

**IN THE UNITED STATES DISTRICT COURT
FOR DISTRICT OF DELAWARE**

ORCA SECURITY LTD.,)	
)	
Plaintiff,)	C.A. No. 23-0758-JLH
)	
v.)	
)	
WIZ, INC.)	
)	
Defendant.)	

**DEFENDANT WIZ INC.’S ANSWER TO SECOND AMENDED COMPLAINT AND
COUNTERCLAIMS**

Defendant and Counterclaim-Plaintiff Wiz, Inc. (“Wiz”) hereby responds to the Second Amended Complaint (“Complaint”) for patent infringement (D.I. 15) of Plaintiff and Counterclaim-Defendant Orca Security Ltd. (“Orca”) as follows. To the extent not specifically admitted in the following paragraphs, the allegations in Orca’s Second Amended Complaint are denied.

INTRODUCTION AND SUMMARY OF THE ACTION¹

1. Wiz denies the allegations in paragraph 1 of the Complaint.
2. Wiz denies the allegations in paragraph 2 of the Complaint.
3. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 3 of the Complaint and on that basis, denies them.

¹ Wiz has incorporated the headings that appear in the Second Amended Complaint. Wiz does not necessarily agree with the characterization of such headings and does not waive any right to object to those characterizations. Accordingly, to the extent that a particular heading can be construed as an allegation, Wiz specifically denies any such allegations.

4. Plaintiff's allegations in paragraph 4 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of Plaintiff's patents.

5. Plaintiff's allegations in paragraph 5 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of Plaintiff's patents.

6. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 6 of the Complaint and on that basis, denies them.

7. Plaintiff's allegations in paragraph 7 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of Plaintiff's patents.

8. Plaintiff's allegations in paragraph 8 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of Plaintiff's patents.

9. Wiz specifically denies that it infringes any valid claim of Plaintiff's patents. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 9 of the Complaint and on that basis, denies them.

10. Wiz admits that the faces of what appear to be U.S. Patent Nos. 11,663,031 (the "'031 patent"), 11,663,032 (the "'032 patent"), 11,693,685 (the "'685 patent"), 11,726,809 (the

“‘809 patent”), 11,740,926 (the “‘926 patent”), and 11,775,326 (the “‘326 patent”) list “Avi Shua” as “Inventor.” Wiz denies that these patents issued on August 22, 2022, and specifically denies that Wiz infringes any valid claim of Plaintiff’s patents.² Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 10 of the Complaint and on that basis, denies them.

11. Wiz denies the allegations in paragraph 11 of the Complaint.

12. Wiz denies the allegations in paragraph 12 of the Complaint.

WIZ AND ITS WIDESPREAD COPYING OF ORCA

13. Wiz admits that it was founded in January 2020 by Messrs. Rappaport, Luttwak, Costica, and Reznik. Wiz further admits that Messrs. Rappaport, Luttwak, Costica, and Reznik joined Microsoft after Microsoft acquired their prior cloud security company, Adallom, Inc. (“Adallom”) and that they served as leaders of Microsoft’s Cloud Security Group. Wiz further admits that, after years in the security industry at Adallom and Microsoft, the founders of Wiz identified prior cloud security solutions as complex, fragmented, and generating too many alerts for security teams. Wiz further admits that on or around December 9, 2020, Wiz emerged from stealth with a cloud security solution that took a new approach and with a new architecture allowing for seamless scanning of the entire cloud environment across compute types and cloud services for vulnerabilities, configuration, network, and identity issues without agents or sidecars. The remaining allegations of paragraph 13 appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the

² Orca is likely referring to U.S. Patent No. 11,431,735, which was the first patent to issue in the family of asserted patents and issued on August 22, 2022. In response to a petition for inter partes review by Wiz, Orca statutorily disclaimed all challenged claims in the ‘735 patent, including all independent claims. See IPR2024-00220, Doc. No. 6.

claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of Plaintiff's patents.

14. Wiz admits that Messrs. Rappaport, Luttwak, Costica, and Reznik left Microsoft at various dates and thereafter founded Wiz in January 2020. Wiz denies the remaining allegations in paragraph 14 of the Complaint, and specifically denies that it infringes any valid claim of Plaintiff's patents or engaged in any actionable copying of Plaintiff.

15. Wiz admits that by in or around December 2020, it had a cloud security solution that allowed for seamless scanning of the entire cloud environment across compute types and cloud services for vulnerabilities, configuration, network, and identity issues without agents or sidecars and that the solution delivered 360 degrees of visibility for cloud security teams by highlighting the critical risks in cloud environments across all risk pillars, and with the goal to empower security teams to know their clouds better than the developer teams. Wiz further admits that Fortune 100 companies were among its early customers. Wiz further admits that it issued the press release available at <https://www.wiz.io/blog/100m-arr-in-18-months-wiz-becomes-the-fastest-growing-software-company-ever> on or around August 10, 2022. The press release speaks for itself. Wiz further admits that by in or around February 2023, it had raised \$300 million at a \$10 billion valuation. Wiz denies the remaining allegations in paragraph 15 of the Complaint and specifically denies that it infringes any valid claim of Plaintiff's patents or engaged in any actionable copying of Plaintiff.

16. Wiz denies that it has engaged in any copying of "Orca's technology." Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 16 of the Complaint and on that basis, denies them.

17. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 17 of the Complaint and on that basis, denies them.

18. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 18 of the Complaint and on that basis, denies them.

19. The allegations of paragraph 19 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies that it infringes any valid claim of Plaintiff's patents or engaged in any actionable copying of Plaintiff. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 19 of the Complaint and on that basis, denies them.

20. The allegations of paragraph 20 include purported citation to a document from Wiz's website; that document speaks for itself. Wiz denies that it infringes any valid claim of Plaintiff's patents or engaged in any actionable copying of Plaintiff. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 20 of the Complaint and on that basis, denies them.

21. The allegations of paragraph 21 include purported citation to Wiz's website; that website speaks for itself. Wiz denies that it infringes any valid claim of Plaintiff's patents or engaged in any actionable copying of Plaintiff. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 21 of the Complaint and on that basis, denies them.

22. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegation that "Wiz showed up with its own coffee machine," and, on that basis, denies it. Wiz denies the remaining allegations of paragraph 22 of the Complaint, and

specifically denies that it infringes any valid claim of Plaintiff's patents or engaged in any actionable copying of Plaintiff.

23. Wiz admits that it was issued U.S. Patent No. 11,374,982; that document speaks for itself. Wiz denies copying Orca's patents, its prosecution strategy, or its prosecuting attorney. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 23 of the Complaint and on that basis, denies them.

24. Wiz denies the allegations in paragraph 24 of the Complaint.

25. Wiz denies hiring Orca's outside corporate counsel to assist Wiz in an attempt to copy Orca. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the remaining allegations in paragraph 25 of the Complaint and on that basis, denies them.

26. Wiz denies the allegations in paragraph 26 of the Complaint.

27. Wiz denies the allegations in paragraph 27 of the Complaint.

28. Wiz denies the allegations in paragraph 28 of the Complaint.

29. Wiz denies the allegations in paragraph 29 of the Complaint.

THE PARTIES

30. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 30 of the Complaint and on that basis, denies them.

31. Wiz admits that Wiz, Inc. is a Delaware company with a principal place of business at One Manhattan West, 57th Floor, New York, New York.

JURISDICTION AND VENUE

32. Wiz admits that this action invokes the United States patent laws, and that this Court has subject matter jurisdiction over patent law claims pursuant to 28 U.S.C. §§ 1331 and 1338(a).

33. Wiz does not contest that this Court has personal jurisdiction solely for the purposes of this particular action. Wiz specifically denies that it has committed any acts of infringement within this district, or any other district. Otherwise denied.

34. Wiz does not contest that this Court has personal jurisdiction solely for the purposes of this particular action. Wiz specifically denies that it has committed any acts of infringement within this district, or any other district. Otherwise denied.

35. Wiz admits that venue is proper in this judicial district for the purposes of this particular action. Wiz specifically denies that it has committed any acts of infringement within this district, or any other district. Otherwise denied.

COUNT I
(INFRINGEMENT OF THE '031 PATENT)

36. Wiz realleges and incorporates by reference its responses to paragraphs 1-35.

37. Wiz admits that what purports to be a copy of U.S. Patent No. 11,663,031 is attached as Exhibit 1 to the Complaint. Wiz further admits that the face of what appears to be the '031 patent indicates that its title is "Techniques for Securing Virtual Cloud Assets at Rest Against Cyber Threats" and that the date of the patent is May 30, 2023. Wiz denies that the '031 patent was duly and legally issued.

38. Wiz denies that Orca has any right to recover damages for infringement of the '031 patent. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 38 of the Complaint and on that basis, denies them.

39. Wiz denies that the '031 patent is valid and enforceable.

40. Orca's allegations in paragraph 40 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '031 patent.

41. Wiz denies the allegations in paragraph 41 of the Complaint.

42. Wiz admits that paragraph 42 of the Complaint reproduces the language of claim 9 of what appears to be the '031 patent.

43. Wiz denies the allegations in paragraph 43 of the Complaint.

44. Wiz admits the claim language quoted in paragraph 44 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 44 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

45. Wiz admits the claim language quoted in paragraph 45 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 45 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

46. Wiz admits the claim language quoted in paragraph 46 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 46 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

47. Wiz admits the claim language quoted in paragraph 47 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in

paragraph 47 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

48. Wiz admits the claim language quoted in paragraph 48 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 48 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

49. Wiz admits the claim language quoted in paragraph 49 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 49 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

50. Wiz admits the claim language quoted in paragraph 50 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 50 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

51. Wiz admits the claim language in paragraph 51 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 51 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

52. Orca's allegations in paragraph 52 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '031 patent.

53. Wiz admits the claim language in paragraph 53 of the Complaint appears in claim 9 of what appears to be the '031 patent. Wiz denies the remaining allegations in paragraph 53 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

54. Wiz denies the allegations in paragraph 54 of the Complaint.

55. Wiz denies the allegations in paragraph 55 of the Complaint.

56. Wiz denies the allegations in paragraph 56 of the Complaint.

57. Wiz denies the allegations in paragraph 57 of the Complaint.

58. Wiz denies the allegations in paragraph 58 of the Complaint.

59. The allegations in paragraph 59 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 59 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

60. The allegations in paragraph 60 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 60 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

61. The allegations in paragraph 61 include purported citation to documents from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 61 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

62. Wiz denies the allegations in paragraph 62 of the Complaint.

63. Wiz denies the allegations in paragraph 63 of the Complaint.

64. Wiz denies the allegations in paragraph 64 of the Complaint.

65. The allegations in paragraph 65 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 65 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

66. Wiz denies the allegations in paragraph 66 of the Complaint.

67. The allegations in paragraph 67 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 67 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

68. The allegations in paragraph 68 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 68 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

69. The allegations in paragraph 69 include purported citation to documents from Wiz's website; that website speaks for itself. Wiz denies the allegations in paragraph 69 of the Complaint, and specifically denies that it infringes any valid claim of the '031 patent.

70. Wiz denies the allegations in paragraph 70 of the Complaint.

71. Wiz denies the allegations in paragraph 71 of the Complaint.

72. Wiz denies the allegations in paragraph 72 of the Complaint.

COUNT II
(INFRINGEMENT OF THE '032 PATENT)

73. Wiz realleges and incorporates by reference its responses to paragraphs 1-35.

74. Wiz admits that what purports to be a copy of U.S. Patent No. 11,663,032 is attached as Exhibit 2 to the Complaint. Wiz further admits that the face of what appears to be the '032 patent indicates that its title is "Techniques for Securing Virtual Machines By Application Use Analysis" and that the date of the patent is May 30, 2023. Wiz denies that the '032 patent was duly and legally issued.

75. Wiz denies that Orca has any right to recover damages for infringement of the '032 patent. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 75 of the Complaint and on that basis, denies them.

76. Wiz denies that the '032 patent is valid and enforceable.

77. Orca's allegations in paragraph 77 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '032 patent.

78. Orca's allegations in paragraph 78 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '032 patent.

79. Wiz denies the allegations in paragraph 79 of the Complaint.

80. Wiz admits that paragraph 80 of the Complaint reproduces the language of claim 1 of what appears to be the '032 patent.

81. Wiz denies the allegations in paragraph 81 of the Complaint.

82. Wiz admits the claim language quoted in paragraph 82 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 82 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

83. Wiz admits the claim language quoted in paragraph 83 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 83 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

84. Wiz admits the claim language quoted in paragraph 84 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in

paragraph 84 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

85. Wiz admits the claim language quoted in paragraph 85 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 85 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

86. Wiz admits the claim language quoted in paragraph 86 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 86 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

87. Wiz admits the claim language quoted in paragraph 87 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 87 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

88. Wiz admits the claim language quoted in paragraph 88 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 88 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

89. Wiz admits the claim language quoted in paragraph 89 of the Complaint appears in claim 1 of what appears to be the '032 patent. Wiz denies the remaining allegations in paragraph 89 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

90. Wiz denies the allegations in paragraph 90 of the Complaint.

91. Wiz denies the allegations in paragraph 91 of the Complaint.

92. Wiz denies the allegations in paragraph 92 of the Complaint.

93. Wiz denies the allegations in paragraph 93 of the Complaint.

94. Wiz denies the allegations in paragraph 94 of the Complaint.

95. The allegations in paragraph 95 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 95 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

96. The allegations in paragraph 96 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 96 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

97. The allegations in paragraph 97 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 97 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

98. Wiz denies the allegations in paragraph 98 of the Complaint.

99. Wiz denies the allegations in paragraph 99 of the Complaint.

100. The allegations in paragraph 100 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 100 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

101. The allegations in paragraph 101 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 101 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

102. Wiz denies the allegations in paragraph 102 of the Complaint.

103. The allegations in paragraph 103 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 103 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

104. The allegations in paragraph 104 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 104 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

105. The allegations in paragraph 96 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 96 of the Complaint, and specifically denies that it infringes any valid claim of the '032 patent.

106. Wiz denies the allegations in paragraph 106 of the Complaint.

107. Wiz denies the allegations in paragraph 107 of the Complaint.

108. Wiz denies the allegations in paragraph 108 of the Complaint.

COUNT III
(INFRINGEMENT OF THE '685 PATENT)

109. Wiz realleges and incorporates by reference its responses to paragraphs 1-35.

110. Wiz admits that what purports to be a copy of U.S. Patent No. 11,693,685 is attached as Exhibit 7 to the Complaint. Wiz further admits that the face of what appears to be the '685 patent indicates that its title is "Virtual Machine Vulnerabilities and Sensitive Data Analysis and Detection" and that the date of the patent is July 4, 2023. Wiz denies that the '685 patent was duly and legally issued.

111. Wiz denies that Orca has any right to recover damages for infringement of the '685 patent. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 111 of the Complaint and on that basis, denies them.

112. Wiz denies that the '685 patent is valid and enforceable.

113. Orca's allegations in paragraph 113 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '685 patent.

114. Wiz denies the allegations in paragraph 114 of the Complaint.

115. Wiz admits that paragraph 115 of the Complaint reproduces the language of claim 1 of what appears to be the '685 patent.

116. Wiz denies the allegations in paragraph 116 of the Complaint.

117. Wiz admits the claim language quoted in paragraph 117 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 117 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

118. Wiz admits the claim language quoted in paragraph 118 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 118 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

119. Wiz admits the claim language quoted in paragraph 119 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 119 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

120. Wiz admits the claim language quoted in paragraph 120 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in

paragraph 120 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

121. Wiz admits the claim language quoted in paragraph 121 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 121 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

122. Wiz admits the claim language quoted in paragraph 122 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 122 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

123. Wiz admits the claim language quoted in paragraph 123 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 123 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

124. Wiz admits the claim language quoted in paragraph 124 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 124 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

125. Wiz admits the claim language quoted in paragraph 125 of the Complaint appears in claim 1 of what appears to be the '685 patent. Wiz denies the remaining allegations in paragraph 125 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

126. Wiz denies the allegations in paragraph 126 of the Complaint.

127. Wiz denies the allegations in paragraph 127 of the Complaint.

128. Wiz denies the allegations in paragraph 128 of the Complaint.

129. Wiz denies the allegations in paragraph 129 of the Complaint.

130. Wiz denies the allegations in paragraph 130 of the Complaint.

131. The allegations in paragraph 131 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 131 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

132. The allegations in paragraph 132 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 132 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

133. The allegations in paragraph 133 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 133 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

134. Wiz denies the allegations in paragraph 134 of the Complaint.

135. Wiz denies the allegations in paragraph 135 of the Complaint.

136. Wiz denies the allegations in paragraph 136 of the Complaint.

137. The allegations in paragraph 137 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 137 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

138. Wiz denies the allegations in paragraph 138 of the Complaint.

139. The allegations in paragraph 139 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 139 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

140. The allegations of paragraph 140 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 140 of the Complaint, and specifically denies that it infringes any valid claim of the '685 patent.

141. Wiz denies the allegations in paragraph 141 of the Complaint.

142. Wiz denies the allegations in paragraph 142 of the Complaint.

143. Wiz denies the allegations in paragraph 143 of the Complaint.

COUNT IV
(INFRINGEMENT OF THE '809 PATENT)

144. Wiz realleges and incorporates by reference its responses to paragraphs 1-35.

145. Wiz admits that what purports to be a copy of U.S. Patent No. 11,726,809 is attached as Exhibit 8 to the Complaint. Wiz further admits that the face of what appears to be the '809 patent indicates that its title is "Techniques for Securing Virtual Machines by Application Existence Analysis" and that the date of the patent is August 15, 2023. Wiz denies that the '809 patent was duly and legally issued.

146. Wiz denies that Orca has any right to recover damages for infringement of the '809 patent. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 146 of the Complaint and on that basis, denies them.

147. Wiz denies that the '809 patent is valid and enforceable.

148. Orca's allegations in paragraph 146 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '809 patent.

149. Wiz denies the allegations in paragraph 149 of the Complaint.

150. Wiz admits that paragraph 150 of the Complaint reproduces the language of claim 1 of what appears to be the '809 patent.

151. Wiz denies the allegations in paragraph 151 of the Complaint.

152. Wiz admits the claim language quoted in paragraph 152 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 152 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

153. Wiz admits the claim language quoted in paragraph 153 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 153 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

154. Wiz admits the claim language quoted in paragraph 154 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 154 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

155. Wiz admits the claim language quoted in paragraph 155 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 155 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

156. Wiz admits the claim language quoted in paragraph 156 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 156 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

157. Wiz admits the claim language quoted in paragraph 157 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 157 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

158. Wiz admits the claim language quoted in paragraph 158 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 158 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

159. Wiz admits the claim language quoted in paragraph 159 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 159 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

160. Wiz admits the claim language quoted in paragraph 160 of the Complaint appears in claim 1 of what appears to be the '809 patent. Wiz denies the remaining allegations in paragraph 160 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

161. Wiz denies the allegations in paragraph 161 of the Complaint.

162. Wiz denies the allegations in paragraph 162 of the Complaint.

163. Wiz denies the allegations in paragraph 163 of the Complaint.

164. Wiz denies the allegations in paragraph 164 of the Complaint.

165. Wiz denies the allegations in paragraph 165 of the Complaint.

166. The allegations in paragraph 166 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 166 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

167. The allegations in paragraph 167 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 167 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

168. The allegations in paragraph 168 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 168 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

169. Wiz denies the allegations in paragraph 169 of the Complaint.

170. Wiz denies the allegations in paragraph 170 of the Complaint.

171. Wiz denies the allegations in paragraph 171 of the Complaint.

172. The allegations in paragraph 172 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 172 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

173. Wiz denies the allegations in paragraph 173 of the Complaint.

174. The allegations in paragraph 174 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 174 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

175. The allegations in paragraph 175 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 175 of the Complaint, and specifically denies that it infringes any valid claim of the '809 patent.

176. Wiz denies the allegations in paragraph 176 of the Complaint.

177. Wiz denies the allegations in paragraph 177 of the Complaint.

178. Wiz denies the allegations in paragraph 178 of the Complaint.

COUNT V
(INFRINGEMENT OF THE '926 PATENT)

179. Wiz realleges and incorporates by reference its responses to paragraphs 1-35.

180. Wiz admits that what purports to be a copy of U.S. Patent No. 11,740,926 is attached as Exhibit 9 to the Complaint. Wiz further admits that the face of what appears to be the '926 patent indicates that its title is "Techniques for Securing Virtual Machines by Analyzing Data for Cyber Threats" and that the date of the patent is August 29, 2023. Wiz denies that the '926 patent was duly and legally issued.

181. Wiz denies that Orca has any right to recover damages for infringement of the '926 patent. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 181 of the Complaint and on that basis, denies them.

182. Wiz denies that the '809 patent is valid and enforceable.

183. Orca's allegations in paragraph 183 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '926 patent.

184. Wiz denies the allegations in paragraph 184 of the Complaint.

185. Wiz admits that paragraph 185 of the Complaint reproduces the language of claim 1 of what appears to be the '926 patent.

186. Wiz denies the allegations in paragraph 186 of the Complaint.

187. Wiz admits the claim language quoted in paragraph 187 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in

paragraph 187 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

188. Wiz admits the claim language quoted in paragraph 188 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 188 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

189. Wiz admits the claim language quoted in paragraph 189 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 189 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

190. Wiz admits the claim language quoted in paragraph 190 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 190 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

191. Wiz admits the claim language quoted in paragraph 191 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 191 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

192. Wiz admits the claim language quoted in paragraph 192 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 192 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

193. Wiz admits the claim language quoted in paragraph 193 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 193 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

194. Wiz admits the claim language quoted in paragraph 194 of the Complaint appears in claim 1 of what appears to be the '926 patent. Wiz denies the remaining allegations in paragraph 194 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

195. Wiz denies the allegations in paragraph 195 of the Complaint.

196. Wiz denies the allegations in paragraph 196 of the Complaint.

197. Wiz denies the allegations in paragraph 197 of the Complaint.

198. Wiz denies the allegations in paragraph 198 of the Complaint.

199. Wiz denies the allegations in paragraph 199 of the Complaint.

200. The allegations in paragraph 200 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 200 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

201. The allegations in paragraph 201 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 201 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

202. The allegations in paragraph 202 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 202 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

203. Wiz denies the allegations in paragraph 203 of the Complaint.

204. Wiz denies the allegations in paragraph 204 of the Complaint.

205. Wiz denies the allegations in paragraph 205 of the Complaint.

206. The allegations in paragraph 206 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 206 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

207. Wiz denies the allegations in paragraph 207 of the Complaint.

208. The allegations in paragraph 208 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 208 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

209. The allegations in paragraph 210 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 210 of the Complaint, and specifically denies that it infringes any valid claim of the '926 patent.

210. Wiz denies the allegations in paragraph 210 of the Complaint.

211. Wiz denies the allegations in paragraph 211 of the Complaint.

212. Wiz denies the allegations in paragraph 212 of the Complaint.

COUNT VI
(INFRINGEMENT OF THE '326 PATENT)

213. Wiz realleges and incorporates by reference its responses to paragraphs 1-35.

214. Wiz admits that what purports to be a copy of U.S. Patent No. 11,775,326 is attached as Exhibit 14 to the Complaint. Wiz further admits that the face of what appears to be the '362 patent indicates that its title is "Techniques for Securing a Plurality of Virtual Machines in a Cloud Computing Environment" and that the date of the patent is October 3, 2023. Wiz denies that the '362 patent was duly and legally issued.

215. Wiz denies that Orca has any right to recover damages for infringement of the '362 patent. Wiz is without knowledge or information sufficient to form a belief as to the truth or falsity of the allegations in paragraph 215 of the Complaint and on that basis, denies them.

216. Wiz denies that the '362 patent is valid and enforceable.

217. Orca's allegations in paragraph 217 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '362 patent.

218. Orca's allegations in paragraph 218 of the Complaint appear intended to reflect the state of the art, claim scope, an appropriate construction of any of the claim terms, the subject matter of the claims, or a claim of infringement, and Wiz therefore denies them. Wiz specifically denies that it infringes any valid claim of the '362 patent.

219. Wiz denies the allegations in paragraph 219 of the Complaint.

220. Wiz admits that paragraph 220 of the Complaint reproduces the language of claim 1 of what appears to be the '362 patent.

221. Wiz denies the allegations in paragraph 221 of the Complaint.

222. Wiz admits the claim language quoted in paragraph 222 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 222 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

223. Wiz admits the claim language quoted in paragraph 223 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in

paragraph 223 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

224. Wiz admits the claim language quoted in paragraph 224 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 224 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

225. Wiz admits the claim language quoted in paragraph 225 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 225 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

226. Wiz admits the claim language quoted in paragraph 226 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 226 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

227. Wiz admits the claim language quoted in paragraph 227 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 227 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

228. Wiz admits the claim language quoted in paragraph 228 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 228 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

229. Wiz admits the claim language quoted in paragraph 229 of the Complaint appears in claim 1 of what appears to be the '362 patent. Wiz denies the remaining allegations in paragraph 229 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

230. Wiz denies the allegations in paragraph 230 of the Complaint.

231. Wiz denies the allegations in paragraph 231 of the Complaint.

232. Wiz denies the allegations in paragraph 232 of the Complaint.

233. Wiz denies the allegations in paragraph 233 of the Complaint.

234. Wiz denies the allegations in paragraph 234 of the Complaint.

235. The allegations in paragraph 235 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 235 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

236. The allegations in paragraph 236 include purported citation to a video posted by Wiz; that video speaks for itself. Wiz denies the remaining allegations in paragraph 236 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

237. The allegations in paragraph 237 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 237 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

238. Wiz denies the allegations in paragraph 238 of the Complaint.

239. Wiz denies the allegations in paragraph 239 of the Complaint.

240. Wiz denies the allegations in paragraph 240 of the Complaint.

241. The allegations in paragraph 241 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 241 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

242. Wiz denies the allegations in paragraph 242 of the Complaint.

243. The allegations in paragraph 243 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 243 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

244. The allegations in paragraph 244 include purported citation to webpages from Wiz's website; that website speaks for itself. Wiz denies the remaining allegations in paragraph 244 of the Complaint, and specifically denies that it infringes any valid claim of the '362 patent.

245. Wiz denies the allegations in paragraph 245 of the Complaint.

246. Wiz denies the allegations in paragraph 246 of the Complaint.

247. Wiz denies the allegations in paragraph 247 of the Complaint.

PRAYER FOR RELIEF

248. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

249. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

250. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

251. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

252. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

253. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

254. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

255. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

256. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

257. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

258. Wiz denies that Orca is entitled to any relief whatsoever, including all relief requested in Orca's "Prayer for Relief." To the extent any statement in the Prayer for Relief is deemed factual and/or requires a response, it is denied.

DEFENSES

259. In addition to denying infringement as to each of Orca's Asserted Patents, and subject to the responses above, Wiz alleges and asserts the following defenses in response to the allegations in Orca's Second Amended Complaint, undertaking the burden of proof only as to those defenses deemed affirmative defenses by law, regardless of how such defenses are denominated herein.

FIRST DEFENSE **(Non-Infringement)**

260. Wiz does not infringe and has not infringed (directly, contributorily, or by inducement), either literally or under doctrine of equivalents, and is not liable for infringement of any valid and enforceable claim of the '031, '032, '685, '809, '926, or '362 patents (collectively, the "Patents-in-Suit")

SECOND DEFENSE **(Invalidity)**

261. The claims of the Patents-in-Suit are invalid and unenforceable under 35 U.S.C § 102 because the claims lack novelty and are taught and suggested by the prior art.

262. The claims of the Patents-in-Suit are invalid and unenforceable under 35 U.S.C § 103 because the claims are obvious in view of the prior art.

263. For example, Orca recently disclaimed every challenged claim of a related patent, U.S. Patent No. 11,431,735, rather than contest that those claims were invalid before the Patent Trial and Appeal Board. *See* Patent Owner's Preliminary Response in *Wiz, Inc. v. Orca Security Ltd.*, Case IPR2024-00220 (April 18, 2024).

264. The claims of the Patents-in-Suit are invalid and unenforceable for failure to satisfy the conditions set forth in 35 U.S.C § 112 including failure to contain a written description, lack of enablement, and indefiniteness because the claims lack novelty and are taught and suggested by the prior art.

THIRD DEFENSE
(Limitations on Patent Damages)

265. Plaintiff's claim for damages, if any, against Wiz for alleged infringement of the Asserted Patents are limited by 35 U.S.C. §§ 286, 287, and/or 288.

FOURTH DEFENSE
(Prosecution History Estoppel)

266. By reason of statements, representations, concessions, admissions, arguments, and/or amendments, whether explicit or implicit, made by or on behalf of the applicant during the prosecution of the patent applications that led to the issuance of the Asserted Patents, Plaintiff's claims of infringement are barred in whole or in part by the doctrine of prosecution history estoppel.

FIFTH DEFENSE
(Patent Marking)

267. Any claim for damages for patent infringement is limited by 35 U.S.C. § 287 to those damages occurring only after the notice of infringement.

SIXTH DEFENSE
(License)

268. Plaintiff's claims are barred in whole or in part by an express or implied license and/or the patent exhaustion doctrine.

SEVENTH DEFENSE
(Non-Exceptional Case)

269. Plaintiff cannot prove that this is an exceptional case that would justify an award of attorney's fees against Wiz pursuant to 35 U.S.C. § 285.

EIGHTH DEFENSE
(Ensnarement)

270. Plaintiff's claims for infringement are barred by the doctrine of ensnarement.

NINTH DEFENSE
(Unclean Hands)

271. Plaintiff's claims are barred by the doctrine of unclean hands. For example, Orca has obtained non-public, proprietary information of Wiz without Wiz's authorization on multiple occasions and, rather than return or destroy that information, has used that information to bring baseless litigation targeting Wiz, including this case.

272. As a specific example, in this action, Exhibit 4 to Orca's original complaint (D.I. 1-1 at 45-55) is from a non-publicly accessible webpage behind a login page specifically designed to prevent access to Wiz's proprietary information unless authorized by Wiz. When Wiz confronted Orca with this information, Orca stated it obtained this document "from a third party some time in 2022." Notably, Orca continued to amend claims of the Asserted Patents in 2022 and later.

273. Similarly, in August 2022, Orca filed a complaint against one of Wiz's employees and sought to prevent her from working for Wiz for a year, accusing her of misusing Orca's confidential information based only on the fact that she had previously worked for Orca and was now working for Wiz. *See Orca Security, Inc. v. Jacques*, 1:22-cv-01048-CFC (D. Del. August 10, 2022), D.I. 10. However, Orca's own briefing showed it was **Orca** that improperly handled Wiz's confidential information. In its submissions in that case, Orca acknowledged that in the industry, "Request For Proposal ('RFP') processes" involve "blind proposals, including confidential price and technological information, such that competitors like Orca and Wiz do not

know what is contained in each other’s proposal or even which competitors they are up against[,]” but Orca brazenly admitted that a potential client had accidentally given Orca documents relating to Wiz’s response to an RFP and—instead of promptly returning or destroying them—Orca used them as the purported basis for the suit. *Id.* at 4, 8. On information and belief, Orca was also asked to delete the information. Orca voluntarily dismissed that suit before Wiz’s employee filed any response. Again, Orca continued to amend claims of the Asserted Patents after August 2022.

TENTH DEFENSE
(Equitable Estoppel/Waiver)

274. Orca’s claims are barred in whole or in part by the doctrines of equitable estoppel or waiver.

RESERVATION OF RIGHTS

275. Wiz reserves the right to amend this Answer to Orca’s Second Amended Complaint and assert further affirmative defenses in the event that discovery indicates that doing so would be appropriate.

PRAYER FOR RELIEF

WHEREFORE, Wiz respectfully requests that the Court enter judgment in favor and against Plaintiff as follows:

1. Dismissing with prejudice Plaintiff’s claims against Wiz;
2. Denying all relief that Plaintiff seeks in its Complaint;
3. Finding this case exceptional under 35 U.S.C. § 285 and awarding Wiz all costs and attorney’s fees; and
4. Awarding any other relief the Court deems just and equitable.

DEMAND FOR A JURY TRIAL

In accordance with Fed. R. Civ. P. 38, Wiz demands a trial by jury on all issues so triable.

WIZ'S COUNTERCLAIMS

Counterclaim-Plaintiff Wiz hereby alleges the following Counterclaims against Counterclaim-Defendant Orca:

Wiz is a Technology Success Story

1. Wiz is a technology success story and one of the hottest startups in the world. Wiz's products secure their customers' use of the "cloud"—*i.e.* the now ubiquitous software running on servers provided by Amazon Web Services, Google Cloud, Microsoft Azure and others. With cloud computing, a user does not need to have a local or personal computer capable of doing all of their tasks; instead, a computing "workload" can be hosted on remote servers in the "cloud." With the incredible importance of the "cloud," Wiz has provided immense value by securing and protecting these assets.

2. Though founded only in 2020, Wiz currently has a \$12 billion dollar valuation, and recorded over \$350 million in sales in annual recurring revenue. Its customers are a "who's who" of private industry, including Morgan Stanley, BMW, DocuSign, Salesforce, Fox Corporation, Colgate-Palmolive, among many others. More than 40% of the Fortune 100 are Wiz customers.

3. Wiz's success was not built overnight, however. Wiz was founded in 2020 by serial cybersecurity entrepreneurs with over a decade of expertise in the industry. The founders of Wiz, Assaf Rappaport, Yinon Costica, Roy Reznik, and Ami Luttwak served in the Israeli Defense Forces ("IDF"), including various members in Unit 8200, an elite intelligence division and Unit 81, a secret technology section part of the Special Operations Division of the Military Intelligence Directorate of the IDF.

4. In 2012, Rappaport, Luttwak, and Reznik created Adallom, a cloud security company that was based in Menlo Park, California. Ahead of its time, it secured companies' use

of enterprise software cloud applications, such as SharePoint, Dropbox and Box. After raising tens of millions of dollars in venture capital funding, Adallom was acquired by Microsoft for approximately \$320 million in 2015. Microsoft then turned to Rappaport to lead its new cloud security division, with Rappaport later operating as general manager for Microsoft's entire research and development center in Israel.

5. After five years at Microsoft running its cloud security division, Rappaport and the future Wiz founders left Microsoft to build a new venture. The entrepreneurship bug continued to push them, and many investors kept encouraging them to start a new venture.

6. Wiz was incorporated in January 2020 and exited stealth in December of that year, when it announced a \$100 million Series A funding.³ Backed by seed funding from venture capital, the Wiz founders considered a few different ideas. But after talking to dozens of prospective buyers about what they needed most, the group quickly focused on cloud visibility.

7. Wiz's products and services have been a hit, generating hundreds of millions of dollars in revenue. Wiz has also made repeated headlines by publicly identifying cloud security vulnerabilities in the industry. *See* Ex. F, Microsoft *Patched Bing Vulnerability That Allowed Snooping on Email and Other Data*, WALL STREET JOURNAL, March 29, 2023, ("The problem was discovered by outside researchers at the security firm Wiz Inc.").

³ <https://www.wiz.io/blog/wiz-comes-out-of-stealth-with-100m-series-a-funding-to-reinvent-cloud-security>

Wiz's Innovative Intellectual Property Portfolio

8. Wiz's investment in research and development of new cloud security technology and features has led to Wiz being one of the most innovative cybersecurity companies in the industry. This is also reflected in Wiz's intellectual property portfolio.

9. Some of the advancements developed by Wiz related to developing a new holistic cloud security solution that, instead of focusing on each "workload" within the cloud to detect vulnerabilities, focuses on providing visibility across the entire cloud environment to provide unified risk analysis and address risk across a user's entire deployment. These solutions include those directed to, among other things, creating a network graph that visually represents network objects for the user, generating unified graph models across multiple cloud computing platforms by genericizing and imputing network entities, and applying policies on a path through the network to mitigate risks. As claimed in Wiz's patents, these were insights by Wiz to change the model for cloud security, driven by thinking about visibility across the entire cloud, rather than focusing on each cloud server or "workload." The marketplace has agreed.

10. Wiz has also innovated in the area of Artificial Intelligence ("AI") and cloud security. For example, Wiz has developed solutions directed to, among other things, detecting cybersecurity risks from AI models and utilizing Large Language Models ("LLMs") to assist in responding to cybersecurity incidents.

11. Wiz applied for and received patent protection from the United States Patent and Trademark Office ("USPTO") for these advancements, including patents infringed by Orca as discussed further below.

Orca’s Lack of Success in the Marketplace, Followed by Copying of Wiz

12. In contrast to Wiz, Orca has lagged in developing features and in the marketplace. Founded in 2019—only a year prior to Wiz—Orca markets itself as a cloud cybersecurity company. Orca originally focused on specific legacy risks such as workload security. On information and belief, Orca’s purportedly new agentless approach to workload scanning, which it would come to call “SideScanning” (*see, e.g.*, D.I. 1-1 at Ex. 3), did not originally include Wiz’s patented features. For example, according to Orca’s own early descriptions, its “SideScanning” approach works on a per-workload basis. With SideScanning, Orca collects workload data and “then reconstructs the workload’s file system – OS, applications, and data – in a virtual read-only view” to perform its risk analysis.⁴ Orca continues to tout “agentless SideScanning” as its primary “innovation” today.⁵ However, agentless security solutions for the cloud have been known in the industry since before Orca existed, and many in the industry use agentless techniques today,⁶ including the cloud service providers themselves.⁷

13. Orca has not experienced as much success with its approach. While the company claimed it expected “triple figure growth” in 2023, it reportedly laid off 15% of its workforce in

⁴ <https://orca.security/platform/agentless-sidescanning/>

⁵ <https://orca.security/about/>

⁶ *See, e.g.*, <https://blogs.cisco.com/security/agentless-threat-detection-for-microsoft-azure-workloads-with-cisco-stealthwatch-cloud>; <https://www.paloaltonetworks.com/cyberpedia/what-is-the-difference-between-agent-based-and-agentless-security>

⁷ *See, e.g.*, <https://aws.amazon.com/about-aws/whats-new/2023/11/amazon-inspector-agentless-assessments-ec2-preview/>; <https://learn.microsoft.com/en-us/azure/defender-for-cloud/concept-agentless-data-collection>

January 2024. It publicly stated that the layoffs were due to “current macroeconomic conditions.”⁸

14. In its efforts to catch up to Wiz, Orca has repeatedly reviewed and kept Wiz confidential materials meant for potential and current customers, including attaching one such document to its complaint against Wiz. Exhibit 4 to Orca’s complaint is a Wiz document that is not public and only accessible through a login page specifically designed to prevent access to Wiz’s proprietary information without Wiz’s authorization. When Wiz asked Orca how it obtained this document, Orca would only respond that it received it “from a third party some time in 2022.”

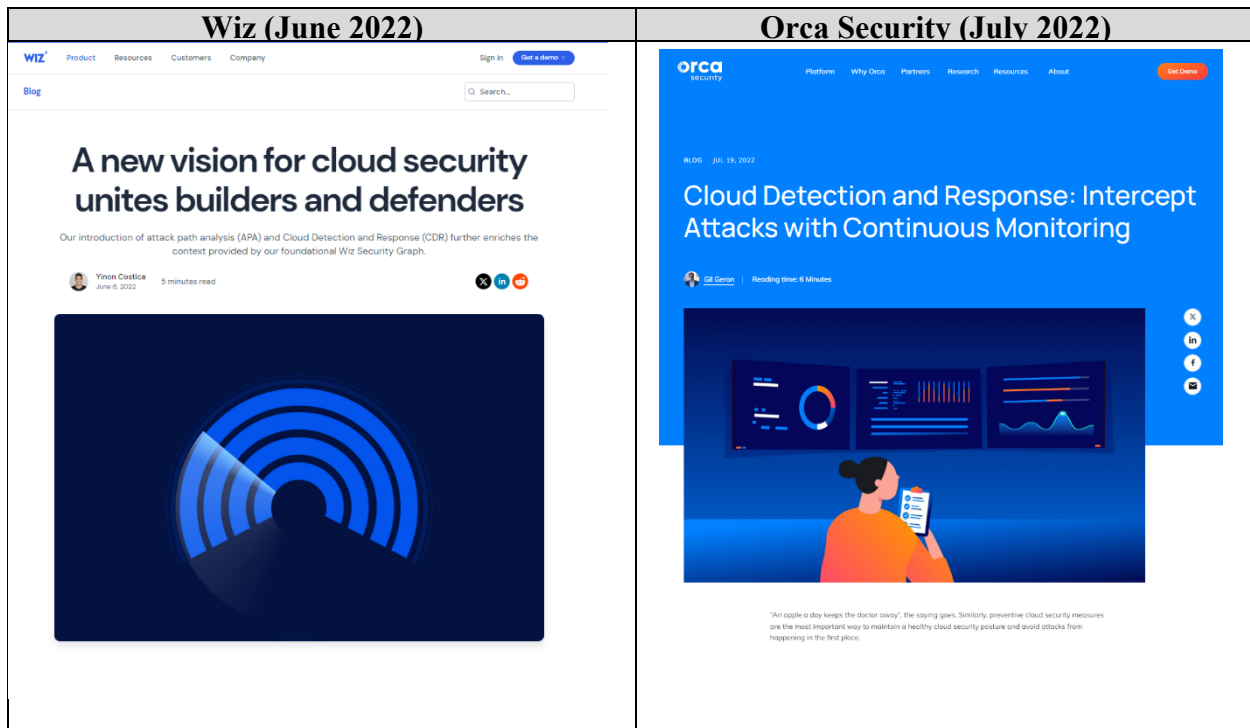
15. As another example, in August 2022, Orca filed a complaint against one of Wiz’s employees, accusing her of misusing Orca’s confidential information because she changed jobs, but Orca’s own briefing showed it was Orca—not Wiz or its employee—that had improperly handled its competitor’s confidential information. *See Orca Security, Inc. v. Jacques*, 1:22-cv-01048-CFC (D. Del. August 10, 2022), D.I. 10. Orca brazenly admitted that a potential client had accidentally given Orca documents relating to Wiz’s response to a Request for Proposal (“RFP”) and—instead of promptly returning or destroying them—Orca kept those documents even though it knew such RFPs are supposed to be “blind” and include “confidential price and technological information, such that competitors like Orca and Wiz do not know what is contained in each other’s proposal or even which competitors they are up against[.]” *Id.* at 4, 8. Orca voluntarily dismissed that suit before Wiz’s employee responded.

⁸ See <https://www.calcalistech.com/ctechnews/article/skm23oz00t>; <https://www.crn.com/news/security/2024/orca-security-cuts-15-percent-of-staff>

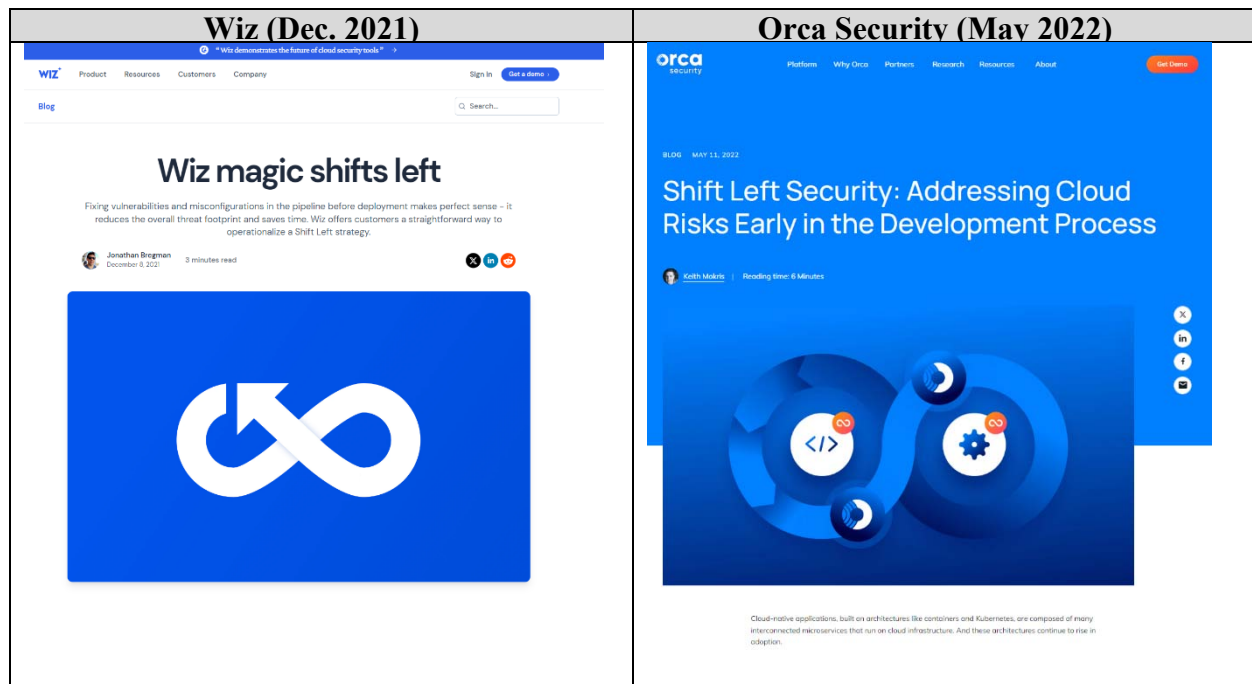
16. After Wiz showed its successful approach in the market, Orca has tried to compete by repeatedly copying features first debuted by Wiz. This includes adopting features patented by Wiz as discussed further below.

17. Rather than “copying” the idea of serving coffee at a conference, Orca’s copying has included adopting Wiz’s patented technology. Orca regularly copies Wiz’s features after they are released:

18. In June 2022, Wiz announced its new Cloud Detection and Response (CDR) feature, which provided capabilities to simulate, detect, and respond to cloud security events. See <https://www.wiz.io/blog/uniting-builders-and-defenders-a-new-vision-for-cloud-security>. A month later, in July 2022, Orca announced a feature with the same name. See <https://orca.security/resources/blog/orca-cloud-security-platform-adds-cloud-detection-and-response/>.



19. In December 2021, Wiz announced it was offering tools for its customers to operationalize a Shift-Left strategy. See <https://www.wiz.io/blog/wiz-magic-shifts-left> (dated December 9, 2021). Approximately six months later, Orca announced its was adding the same features. See <https://orca.security/resources/blog/shift-left-security-platform/> (dated May 11, 2022).

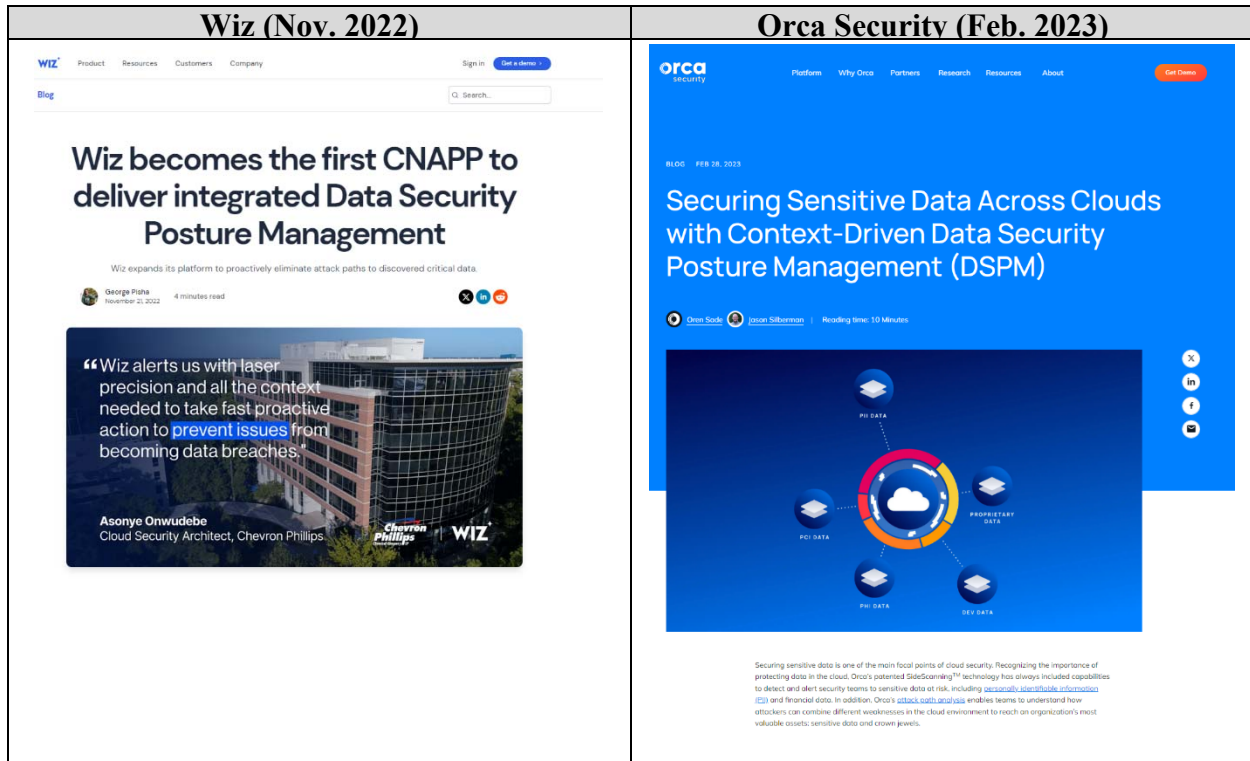


20. Wiz has provided Cloud Infrastructure Entitlement Management or “CIEM” since its earliest product offerings, including using those terms long before Orca. On February 10, 2022, Orca released its CIEM feature. See <https://orca.security/resources/press-releases/orca-platform-expanded-ciem-multi-cloud-security-score> (dated Feb. 10, 2022).

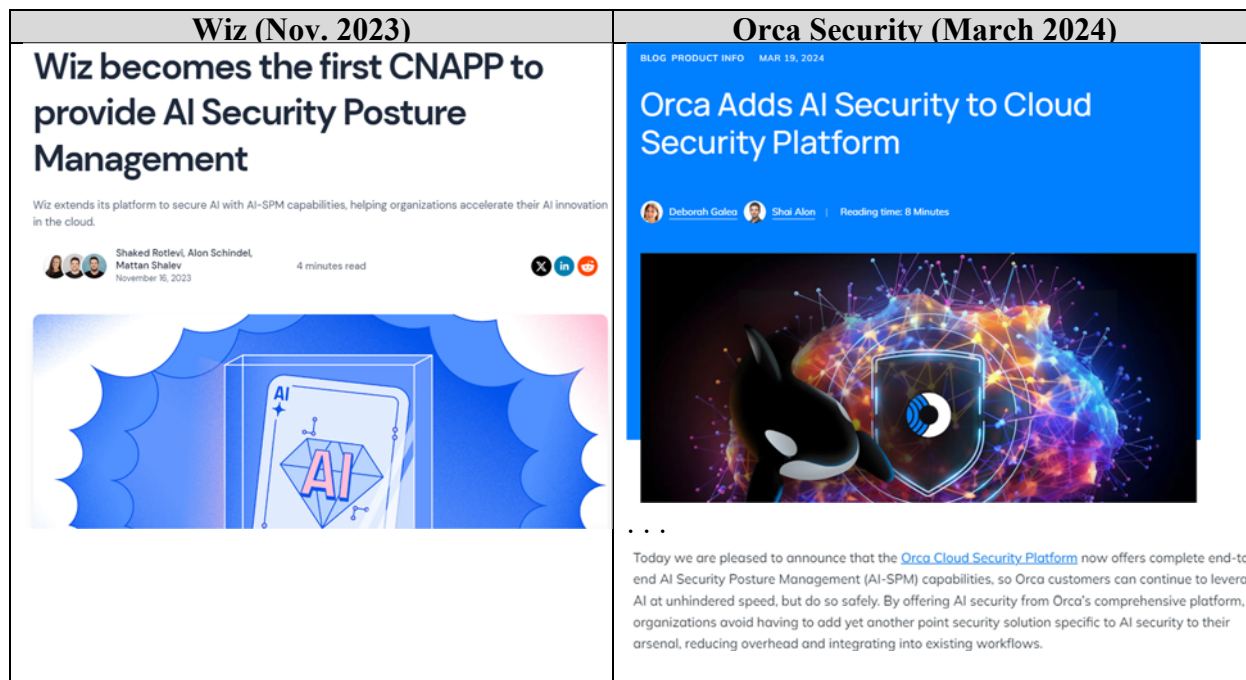
21. By at least August 2021, Wiz had released its Threat Center, a dedicated feed of high-profile security issues with analysis of their impact on your organization. See <https://www.wiz.io/blog/protecting-your-environment-from-chaosdb> (dated Aug. 29, 2021, stating “Wiz Threat Center shows you which assets are at-risk to the most dangerous threats”).

In March 2022, Orca released its version of the same feature, redesigning its interface to be a “news feed” and including a “From the news” functionality similar to Wiz. *See* <https://orca.security/resources/blog/enhancing-the-orca-security-risk-dashboard-with-an-integrated-news-feed/> (dated Mar. 8, 2022).

22. On November 21, 2022, Wiz announced it was the first cloud security platform to offer Data Security Posture Management (DSPM) capabilities to continuously monitor for critical data exposure so organizations can respond before a breach occurs. *See* <https://www.wiz.io/blog/wiz-becomes-first-cnapp-to-deliver-integrated-data-security-posture-management> (dated Nov. 21, 2022). Months later, on Feb. 28, 2023, Orca announced it was launching the same feature. *See* <https://orca.security/resources/blog/securing-sensitive-data-across-clouds-with-data-security-posture-management-dspm/> (blog post dated Feb. 28, 2023 stating, in part: “Today, we are excited to announce that we have now significantly expanded our cloud data security coverage and capabilities, launching a comprehensive offering of Data Security Posture Management (DSPM) as part of the Orca Cloud Security Platform.”).



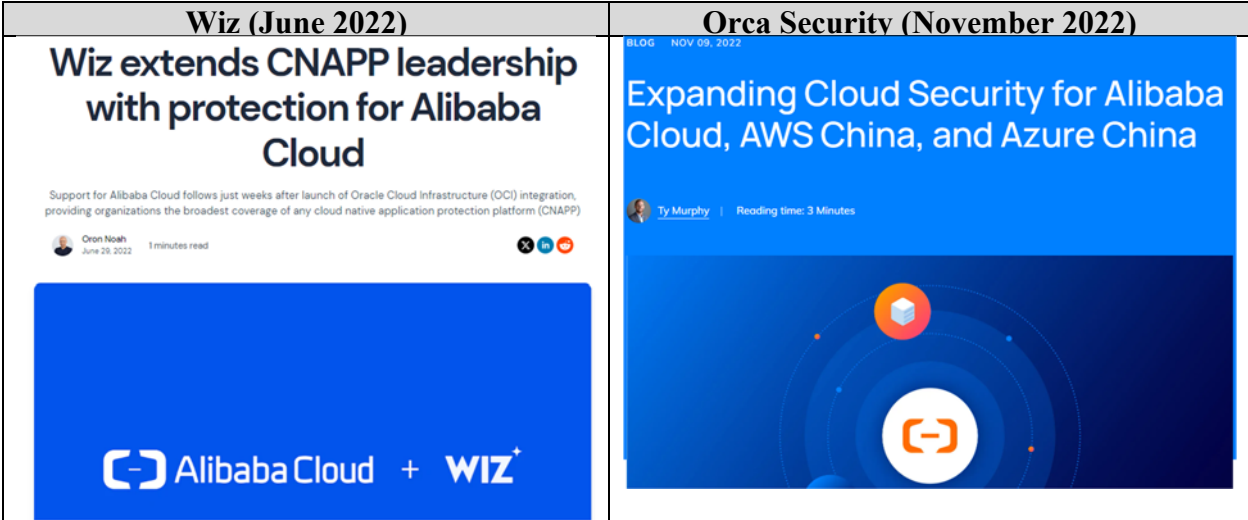
23. On November 16, 2023, Wiz announced it was the first cloud security platform to secure AI with AI Security Posture Management or “AI-SPM capabilities”—a term coined by Wiz—including by announcing an “AI Security Dashboard.” See <https://www.wiz.io/blog/ai-security-posture-management> (dated November 16, 2023). Four months later, on March 19, 2024, Orca announced it would also be offering “AI Security Posture Management (AI-SPM) capabilities”—using the same term coined by Wiz—and an “AI Security dashboard.” See <https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/> (dated March 19, 2024).



24. On June 1, 2022, Wiz announced that its platform could integrate with the Oracle Cloud Infrastructure (OCI). See <https://www.wiz.io/blog/supporting-oracle-cloud-wiz-brings-the-first-graph-based-cloud-security-approach-to-all-major-providers> (dated June 1, 2022). On February 23, 2023, Orca announced its platform could now connect with OCI. See <https://orca.security/resources/blog/expanding-cloud-security-coverage-for-oracle-cloud-infrastructure/> (dated Feb. 23, 2023).



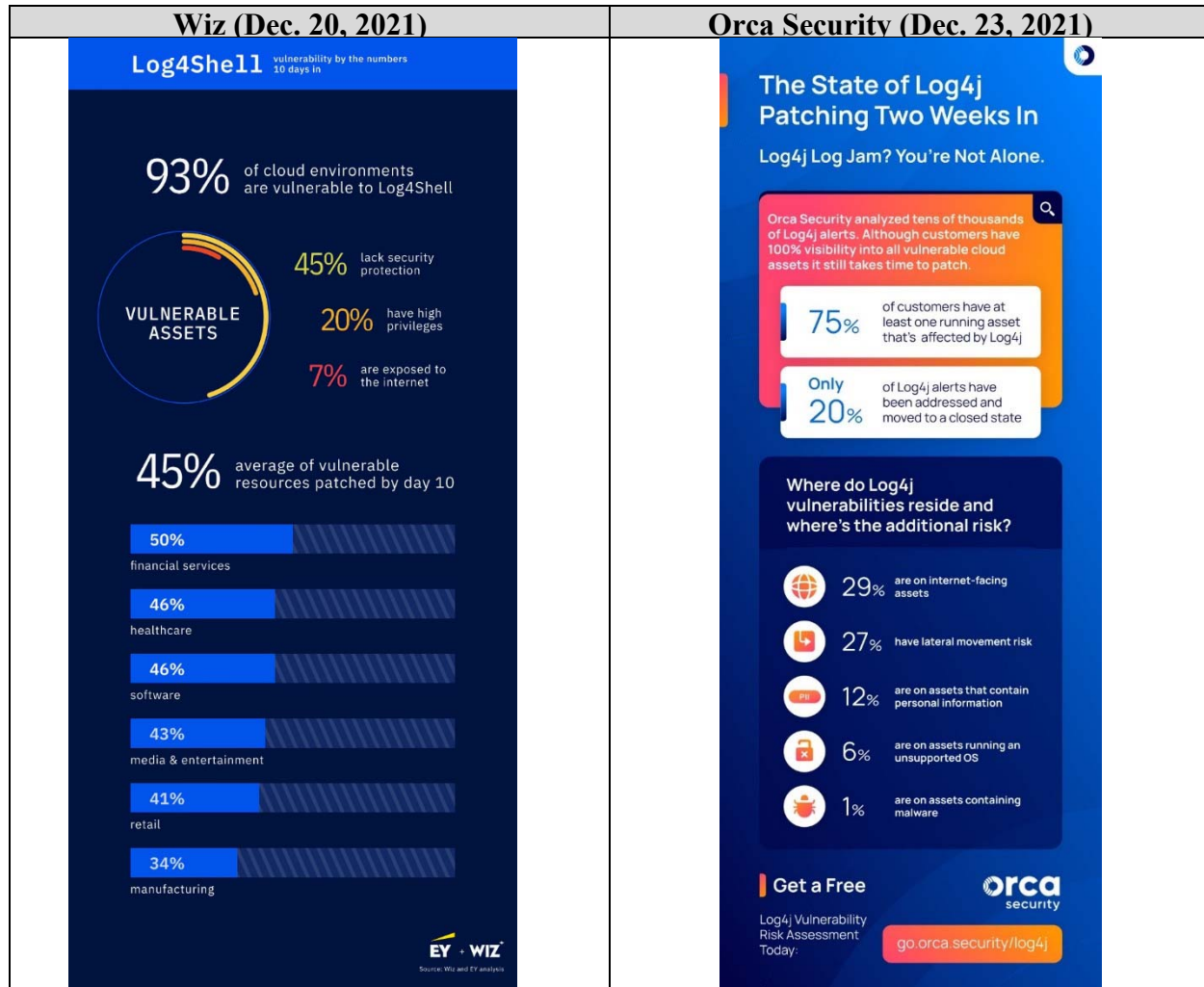
25. Similarly, Wiz announced support for Alibaba Cloud on June 29, 2022. See <https://www.wiz.io/blog/wiz-extends-cnapp-leadership-with-protection-for-alibaba-cloud> (dated June 29, 2022). On November 9, 2022, Orca announced its platform would also support Alibaba Cloud. See <https://orca.security/resources/blog/expanding-cloud-security-for-alibaba-cloud/> (dated Nov. 9, 2022).



26. On June 21, 2023, Wiz announced it became the first CNAPP to provide an end-to-end cloud forensics experience. See <https://www.wiz.io/blog/wiz-becomes-the-first-cnapp-to-provide-end-to-end-cloud-forensics-experience> (dated June 21, 2023). On April 29, 2024, Orca announced it was launching a cloud digital forensics and incident response service. See

<https://orca.security/resources/press-releases/orca-security-launches-cloud-digital-forensics-and-incident-response-service-to-empower-rapid-investigation-of-cloud-incidents/> (dated April 29, 2024).

27. Orca's copying, however, has included the mundane as well, including copying infographics from Wiz. In the wake of the Log4Shell vulnerability that was major news in the cybersecurity industry, Wiz published a statistics blog about the prevalence of the issue, including an infographic. See <https://www.wiz.io/blog/10-days-later-enterprises-halfway-through-patching-log4shell> (dated Dec. 20, 2021). Three days later, Orca published a blog post with a very similar graphic. See <https://orca.security/resources/blog/instantly-detect-log4j-vulnerabilities-on-aws-azure-and-google-cloud/> (dated Dec. 23, 2021).



28. Based on the above examples, Orca appears to have a culture of copying Wiz's innovations, including its patented technology as shown herein.

29. Wiz did not choose to bring this litigation, but faced with Orca's meritless claims, it is now forced to correct the record about Wiz's innovation, Orca's copying of Wiz, and Orca's use of Wiz's intellectual property. Orca is improperly using Wiz's inventions, specifically those claimed in U.S. Patent Nos. 11,722,554 (the "'554 Patent"); 11,929,896 (the "'896 Patent"); 11,936,693 (the "'693 Patent"); 12,001,549 (the "'549 Patent"); and 12,003,529 (the "'529 Patent") (collectively, Wiz's "Asserted Patents"), as discussed in further detail below. Exs. A-E. Wiz brings these counterclaims to address that infringement.

30. On information and belief, Orca was aware the Asserted Patents that issued prior to June 4, 2024, and Orca's infringement thereof prior to these counterclaims at least due to Orca's monitoring of Wiz patents. As one example, in its original complaint in this action, Orca cited and quoted from Wiz's U.S. Patent No. 11,374,982, demonstrating its monitoring of Wiz's patent portfolio. *See, e.g.*, D.I. 1, ¶ 22. Moreover, Orca has a demonstrated culture of copying Wiz, as discussed above. In addition, Orca is aware of each of the Asserted Patents and its infringement thereof at least as of the filing of these counterclaims. Orca has and continues to willfully infringe all of the Asserted Patents.

THE PARTIES

31. Wiz is a Delaware corporation with its principal place of business at One Manhattan West, 57th Floor, New York, New York.

32. On information and belief, Orca is an Israeli company with its principal place of business at 3 Tushia St., Tel Aviv, Israel 6721803.

JURISDICTION AND VENUE

33. This Court has subject matter jurisdiction over the matters asserted herein under 28 U.S.C. §§ 1331 and 1338(a).

34. Orca is subject to this Court's personal jurisdiction at least because Orca initiated this lawsuit and, on information and belief, Orca's business operations include software and services for use in Delaware.

35. Venue is proper in this District pursuant to 28 U.S.C. Section 1391 (b), (c), and/or 1400(b), at least because Orca is a foreign entity that, on information and belief, has committed acts of infringement in this District.

COUNTERCLAIM I
(INFRINGEMENT OF U.S. PATENT NO. 11,722,554)

36. Wiz is the sole and exclusive owner, by assignment, of all rights, title and interest in U.S. Patent No. 11,722,554 (the “’554 patent”), entitled “System and method for analyzing network objects in a cloud environment.” The ’554 patent was duly and legally issued by the U.S. Patent and Trademark Office on August 8, 2023. The named inventors of the ’554 patent are Shai Keren, Danny Shemesh, Roy Reznik, Ami Luttwak, and Avihai Berkovitz. A copy of the ’554 patent is attached as Exhibit A.

37. The ’554 patent generally relates to determining abnormal configuration of network objects in a cloud environment for security purposes. *See* ’554 patent at 2:51-67. This is done through the use of a network graph that includes a visual representation of network objects in the cloud computing environment. Describing an embodiment shown in figure 2 the patent states, “[a] network graph is a data feature describing the various objects included in, and adjacent to, a network, as well as the relationship between such objects.” *Id.* at 8:49-51. The network graph includes network object relationships. “Network object relationships are descriptions of the various connections between the network objects identified at S210. Network relationships may describe aspects of the connections between objects including, without limitation, connected objects, relevant ports of connected objects, connection bandwidths, connection durations, connection protocols, connection names or IDs, connection statuses, and the like, as well as any combination thereof.” *Id.* at 9:4-11.

38. The ’554 patent discloses using the network graph and network object relationships to generate at least one network insight. *See Id.* at 2:63-67. The network insights “are natural-language representations of aspects of the network graph[.]” *Id.* at 10:15-17. “Network insights may include a pure-text descriptions of objects and relationships.” *Id.* at 10:17-

19. “In addition, network insights may include detailed descriptions of objects, relationships and the like as well as any combination thereof.” *Id.* at 10:25-27.

39. Orca has infringed and continues to directly infringe one or more claims of the ’554 patent by making, using, selling, offering for sale, and/or importing into the United States without authority or license, the Orca Platform with Attack Path Analysis in violation of 35 U.S.C. § 271(a). Orca’s infringement includes infringement of, for example, claim 1 of the ’554 patent.

40. Claim 1 of the ’554 patent recites:

1. A method for determining abnormal configuration of network objects deployed in a cloud computing environment, comprising:
collecting network object data on a plurality of network objects deployed in the cloud computing environment;
constructing a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment;
determining relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects;
analyzing the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects; and
tagging network objects in the network graph for which the insight is generated.

41. On information and belief, Orca practices each and every limitation of claim 1 of the ’554 patent by and through the use of the Attack Path Analysis.

42. The preamble of claim 1 recites “[a] method for determining abnormal configuration of network objects deployed in a cloud computing environment, comprising:” To the extent the preamble is limiting, Orca practices this step by, for example, using its Attack Path Analysis product to collect data on assets in cloud computing environments. *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder,*

<https://orca.security/resources/blog/cloud-attack-path-analysis/> (“Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

43. Claim 1 further recites “collecting network object data on a plurality of network objects deployed in the cloud computing environment” Orca’s public blog posts confirm that Orca practices this step by, for example but not limited to, Orca’s Attack Path Analysis “collects and correlates contextual data on each asset” and that “it is important to view risks as an interrelated chain, rather than just siloed risks.” *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/> (“To fully understand where your organization’s most critical weaknesses are, it is important to view risks

as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

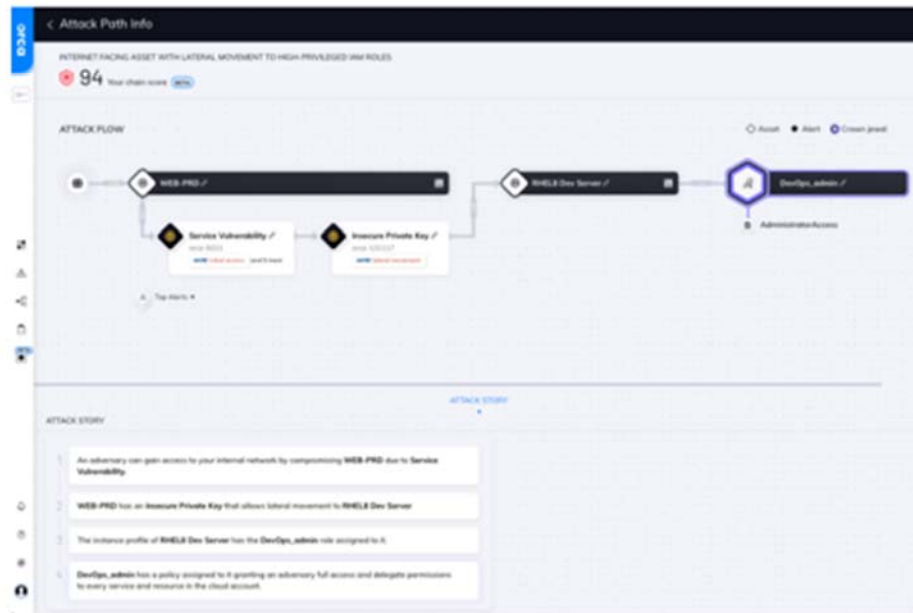
Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

To fully understand where your organization's most critical weaknesses are, it is important to view risks as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.



Orca shows the steps an attacker can take to reach the company's crown jewels

See also, *Data Security and Posture Management*, <https://orca.security/platform/data-security-and-posture-management-dspm/> (“Orca’s DSPM dashboard provides data security teams with an overview of their cloud data stores, sensitive data, and security and compliance alerts. Orca scans managed, unmanaged, and shadow data, giving security teams wide and deep visibility

into where their data resides.”).

Discover and classify data in your cloud

Orca’s DSPM dashboard provides data security teams with an overview of their cloud data stores, [sensitive data](#), and security and compliance alerts. Orca scans managed, unmanaged, and shadow data, giving security teams wide and deep visibility into where their data resides.

- ✓ Get a multi-cloud inventory of data storage assets—including databases, and files in virtual machines, storage buckets, and containers.
- ✓ Know which data stores contain sensitive data and of what type—including PII, [PCI](#), PHI, or financial information—for both security and regulatory purposes.
- ✓ Leverage interactive graphs that show the location and relationship between data stores and other cloud entities.

44. Claim 1 further recites “constructing a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment” Orca’s public blog posts confirm that Orca practices this step by, for example but not limited to, describing operation of Orca’s Attack Path Analysis as “representing attack paths in a visual graph with contextual data on all relevant cloud entities” and by using their “attack path visualization” tool. *See, e.g., Cloud*

Attack Path Analysis: Work Smarter Not Harder, <https://orca.security/resources/blog/cloud-attack-path-analysis/>

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization's most valuable assets, or 'crown jewels', and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

45. Claim 1 further recites “determining relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects” Orca’s public blog posts and marketing videos confirm that Orca practices this step by, for example but not limited to, using Orca’s Attack Path Analysis to identify “risk combinations” between objects including “vulnerability status, misconfiguration risks, trust and authorization[.]” *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*,

<https://orca.security/resources/blog/cloud-attack-path-analysis/>.

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

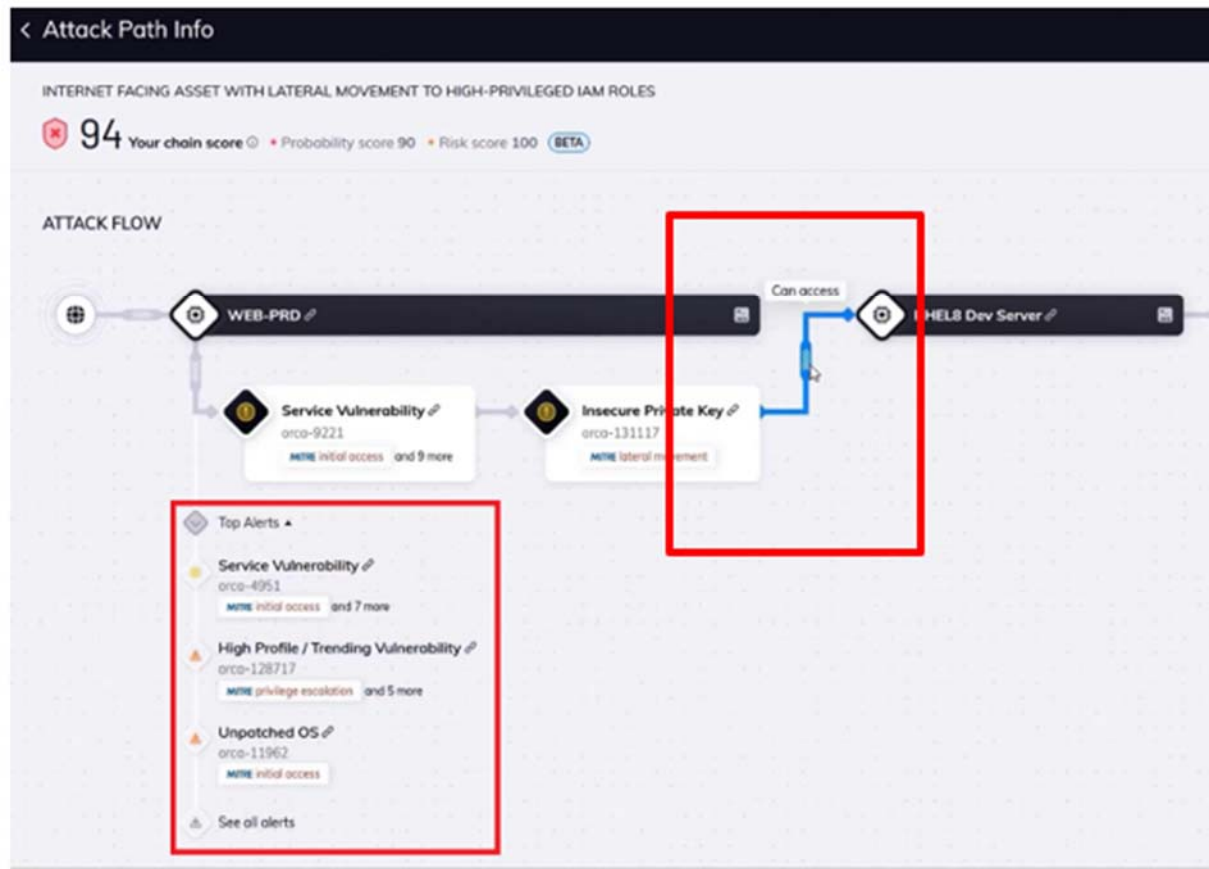
Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization's most valuable assets, or 'crown jewels', and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

See, also, *Orca Bytes: Attack Path Analysis*, <https://www.youtube.com/watch?v=MJkO8UfQa-8>.



46. Claim 1 further recites “analyzing the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects” Orca’s public blog posts and marketing videos confirm that Orca practices this step by, for example but not limited to, using Orca’s Attack Path Analysis’s “analysis and prioritization capabilities” and “representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.” See, e.g., *Cloud Attack Path Analysis: Work Smarter Not Harder*,

<https://orca.security/resources/blog/cloud-attack-path-analysis/>.

Prioritize Attack Paths, Not Just Siloed Alerts

With Orca's new attack path analysis and prioritization capabilities, security teams no longer need to [sift through hundreds of siloed alerts](#) to find out which issues need to be fixed and in what order, but instead can now focus on a much smaller amount of prioritized attack paths, or combinations of alerts, that endanger the company's most critical assets. By prioritizing and scoring each attack path, Orca pinpoints exactly which risks need to be remediated to 'break the chain'.

The screenshot displays the 'Attack Path Info' interface. At the top, it shows a score of 90 and a warning icon. Below this, the 'ATTACK FLOW' section visualizes a sequence of events: a vulnerability (Web-Http://) is exploited to gain administrative access to a JMWEL-03-FR0-7 server, which then connects to a JMWEL-03-Server-7, and finally to a sensitive asset (Server_admin//). Below the flow, the 'ATTACK STORY' section provides a step-by-step narrative of the attack path.

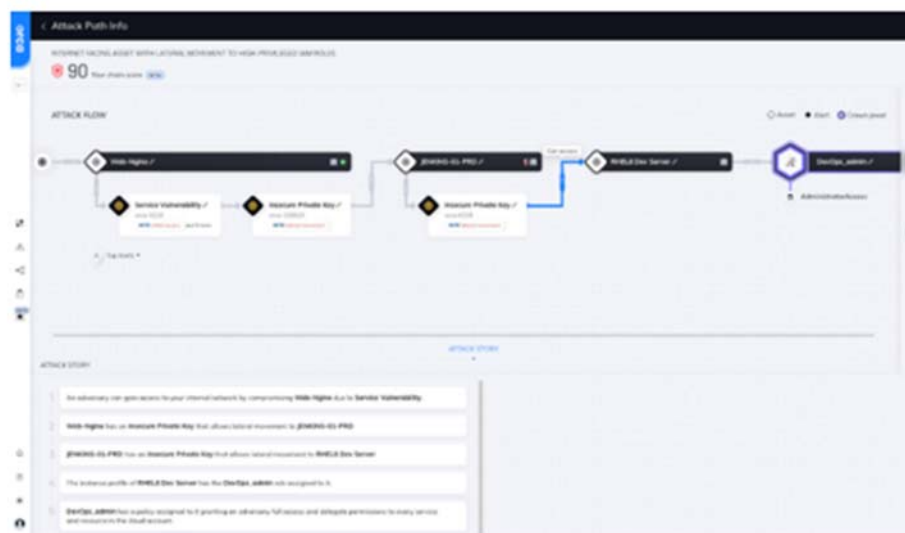
Attack path visualization of how an attacker can access sensitive company data

47. Finally, Claim 1 further recites “and tagging network objects in the network graph for which the insight is generated.” Orca’s public blog posts and marketing videos confirm that Orca practices this step by, for example but not limited to, using its Attack Path Analysis for “prioritizing and scoring each attack path[.]” *See, e.g., Cloud Attack Path Analysis: Work*

Smarter Not Harder, <https://orca.security/resources/blog/cloud-attack-path-analysis/>.

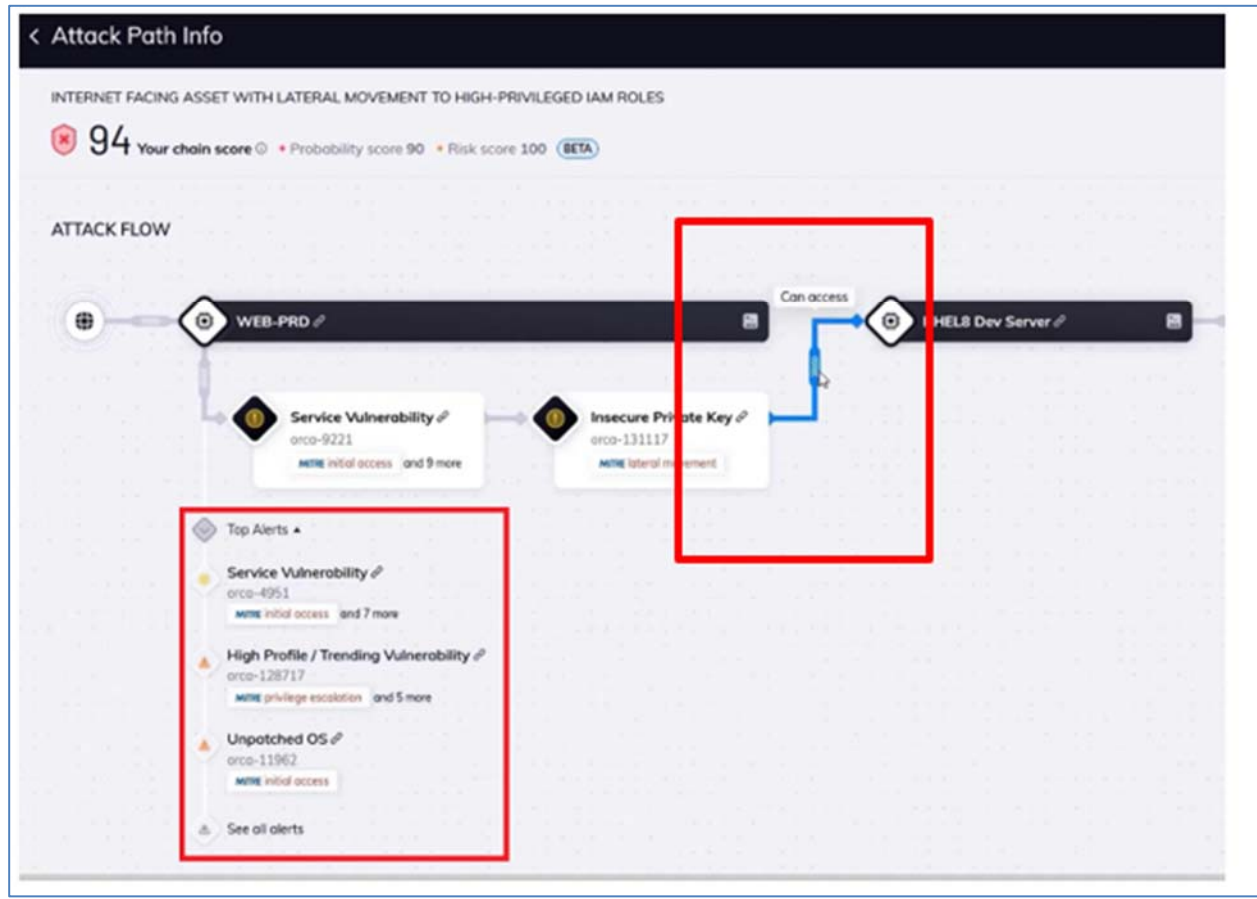
Prioritize Attack Paths, Not Just Siloed Alerts

With Orca's new attack path analysis and prioritization capabilities, security teams no longer need to [sift through hundreds of siloed alerts](#) to find out which issues need to be fixed and in what order, but instead can now focus on a much smaller amount of prioritized attack paths, or combinations of alerts, that endanger the company's most critical assets. By prioritizing and scoring each attack path, Orca pinpoints exactly which risks need to be remediated to 'break the chain'.



Attack path visualization of how an attacker can access sensitive company data

See, e.g., *Orca Bytes: Attack Path Analysis*, <https://www.youtube.com/watch?v=MJkO8UfQa-8>.



48. On information and belief, Orca was aware of the '554 patent and Orca's infringement thereof prior to these counterclaims at least due to Orca's monitoring of Wiz patents as shown by, in its original complaint in this action, that Orca cited and quoted from Wiz's U.S. Patent No. 11,374,982. See, e.g., D.I. 1, ¶ 22. Further, as demonstrated above Orca has repeatedly shown a culture of copying Wiz. This is just one more example of Orca seeing Wiz's success and copying instead of innovating. Moreover, Orca is aware of the '554 patent and Orca's infringement thereof at least as of the filing of these counterclaims. Accordingly, Orca has and continues to willfully infringe the '554 patent.

49. Orca has induced and continues to induce infringement of one or more claims of the '554 patent by, for example but not limited to, encouraging customers to use its Attack Path

Analysis in a manner that directly infringes those claims. Despite its knowledge of the existence of the '554 patent, since at least the filing of this Counterclaim, Orca, upon information and belief, continues to encourage, instruct, enable and otherwise cause its customers to use its Attack Path Analysis in a manner that infringes one or more claims of the '554 patent. Upon information and belief, Orca specifically intends that its customers use its Attack Path Analysis in a manner that infringes one or more claims of the '554 patent by, at a minimum, providing instructions and/or support documentation directing customers on how to use its Attack Path Analysis in an infringing manner, in violation of 35 U.S.C. § 271(b). For example, Orca's public blog posts cited above provide instructions and encourage customers to practice all steps of the claimed method stating: "To fully understand where your organization's most critical weaknesses are, it is important to view risks as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard." *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/>. Further, Orca provides video explanation of Attack Path Analysis stating, "with Orca's new attack path analysis and prioritization capabilities, security teams can now laser focus on a small number of prioritized attack paths or alert on combinations that endanger the company's most critical assets, and every path and link in the path is scored so you can pinpoint exactly which risks need to be remediated." *See Orca Bytes: Attack Path Analysis* (Mar. 31, 2022), <https://www.youtube.com/watch?v=MJkO8UfQa-8>.

50. Orca has contributed and continues to contribute to the infringement of one or more claims of the '554 patent. Upon information and belief, Orca knows that its Attack Path

Analysis feature is especially made and/or adapted for users to infringe one or more claims of the '554 patent and is not a staple article or commodity of commerce suitable for substantial non-infringing use. Because Orca included the features, such as for example but not limited to Attack Path Analysis, in Orca's products, Orca intends for customers to use it. Upon information and belief, its Attack Path Analysis feature has no suitable use that is non-infringing, and therefore Orca intends for customers to use its Attack Path Analysis in an infringing manner. Orca's sales of products including its Attack Path Analysis constitute contributory infringement in violation of 35 U.S.C. § 271(c).

COUNTERCLAIM II
(INFRINGEMENT OF U.S. PATENT NO. 11,929,896)

51. Wiz is the sole and exclusive owner, by assignment, of all rights, title and interest in U.S. Patent No. 11,929,896 (the "'896 patent"), entitled "System and Method for Generation of Unified Graph Models for Network Entities." The '896 patent was duly and legally issued by the U.S. Patent and Trademark Office on Mar. 12, 2024. The named inventors of the '896 patent are Daniel Shemesh, Liran Moysi, Roy Reznik, and Shai Keren. A copy of the '896 patent is attached as Exhibit B.

52. The '896 patent generally relates to generation of network graph models for network entities. *See* '896 patent at 2:48-59. This is done by collecting network entities and network entity properties, and creating a network graph that is a multi-dimensional data structure representing the network entities. *Id.* Describing an embodiment shown in Figure 2 the patent states, a network graph is generated. A graph is a multi-dimensional data feature providing a representation of the contents and structure of a network, cloud, environment, or the like. A graph may include one or more graph vertices, interconnected by one or more graph edges." *Id.* at 11:51-55. The network graph includes vertices and graph edges where "each graph vertex

may correspond with a network entity included in the network, cloud, environment, or the like, and each graph edge may correspond with a connection between such entities.” *Id.* 11:57-60.

The vertices in the network graph may represent “generic entities, such as are described with respect to S220, imputed generic entities, such as are described with respect to S230, as well as any combination thereof.” *Id.* at 62-64.

53. The '896 patent generates the network graph using network entities. “Network entities 105, as may be included in a cloud platform 104, are entities, systems, devices, components, applications, objects, and the like, configured to operate within the cloud platform 104 and provide various functionalities therein. Specifically, the network entities 105 may be, as examples without limitation, entities configured to process data, send data, or receive data, as well as entities configured to provide various other functionalities, and any combination thereof. The network entities 105 may be configured to connect with various other network entities 105, various external entities, and the like, as well as any combination thereof, for purposes including, without limitation, sending data, receiving data, monitoring data transmissions, monitoring network status and activity, and the like, as well as any combination thereof.” *Id.* at 5:29-43. In addition to network entities, the '896 patent includes imputed entities in the network graph. “Imputed entities are generic entities similar or identical to those described with respect to S220, above, which may be constructed to provide representation of network entities which are integrated into host platforms, or network entities which are shielded from, or not otherwise exposed to, a system configured to execute network analysis processes and methods, including the method described with respect to FIG. 2, where such a system may be, without limitation, the graph analysis system 150 of FIG. 1A.” *Id.* at 10:58-67.

54. The network entities may be collected from a plurality of cloud computing platforms. “A cloud platform 104 may be a commercially-available cloud system, provided on a service basis, such as, as examples and without limitation, Amazon AWS®, Microsoft Azure®, and the like. A cloud platform 104 may be a private cloud, a public cloud, a hybrid cloud, and the like. In addition, a cloud platform 104 may include, without limitation, container orchestration or management systems or platforms such as, as an example and without limitation, a Kubernetes deployment, and the like, as well as any combination thereof.” *Id.* 5:1-11 The ’896 patent discloses using the network graph for “network analysis, traffic analysis, entity querying, graph generation and the like, as well as any combination thereof.” *Id.* 6:48-50. “The graph analysis system 150 may be configured as a physical system, device, or component, as a virtual system, device, or component, or in a hybrid physical-virtual configuration.” *Id.* at 6:55-58. Finally, “the network graph is stored in a graph database.” *Id.* 13:41-42.

55. Orca has infringed and continues to directly infringe one or more claims of the ’896 patent by making, using, selling, offering for sale, and/or importing into the United States without authority or license, the Orca Platform with Attack Path Analysis in violation of 35 U.S.C. § 271(a). Orca’s infringement includes infringement of, for example, claim 1 of the ’896 patent.

56. Claim of the ’896 patent recites:

1. A method for generation of unified graph models for network entities, comprising: collecting, for each network entity of a plurality of network entities, network entity data, wherein the network entity data collected for a network entity includes at least a network entity property, wherein the plurality of network entities are deployed in a plurality of cloud computing platforms; genericizing each of the network entities based on the respective collected network entity data to generate a plurality of generic network entities, wherein a generic network entity includes a generic representation of respective network entities from different cloud computing platforms of the plurality of cloud computing platforms;

generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of generic network entities and relations between the generic network entities of the plurality of network entities; and
creating at least one imputed entity, wherein the at least one imputed entity is a generic network entity representing an executed platform functionality, and wherein the executed platform functionality is different than a network entity; and
storing the generated network graph.

57. On information and belief, Orca practices each and every limitation of claim 1 of the '896 patent by and through the use of the Attack Path Analysis.

58. The preamble of claim 1 recites “A method for generation of unified graph models for network entities, comprising:” To the extent the preamble is limiting, Orca practices this step by, for example but not limited to, using its Attack Path Analysis product to generate a unified graph model for network entities. *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/> (“For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and

internal cloud connectivity.”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the core capabilities of a Cloud-Native Application Protection Platform (CNAPP):

59. Claim 1 further recites “collecting, for each network entity of a plurality of network entities, network entity data, wherein the network entity data collected for a network entity includes at least a network entity property, wherein the plurality of network entities are deployed in a plurality of cloud computing platforms;” Orca’s public blog posts confirm that Orca practices this step by, for example but not limited to, Orca’s Attack Path Analysis “collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.” *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/> (“To fully understand where your organization’s most critical weaknesses are, it is important to view risks as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your

critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the core capabilities of a Cloud-Native Application Protection Platform (CNAPP):

60. Orca collects network entity data across a plurality of cloud computing platforms.

See, e.g., Key Security Capabilities in Kubernetes,

<https://orca.security/resources/blog/kubernetes-security-capabilities-policies/>; *AI-Driven Cloud*

Security, <https://orca.security/platform/ai-cloud-security/> (“I’m going to use AI and I’m going to

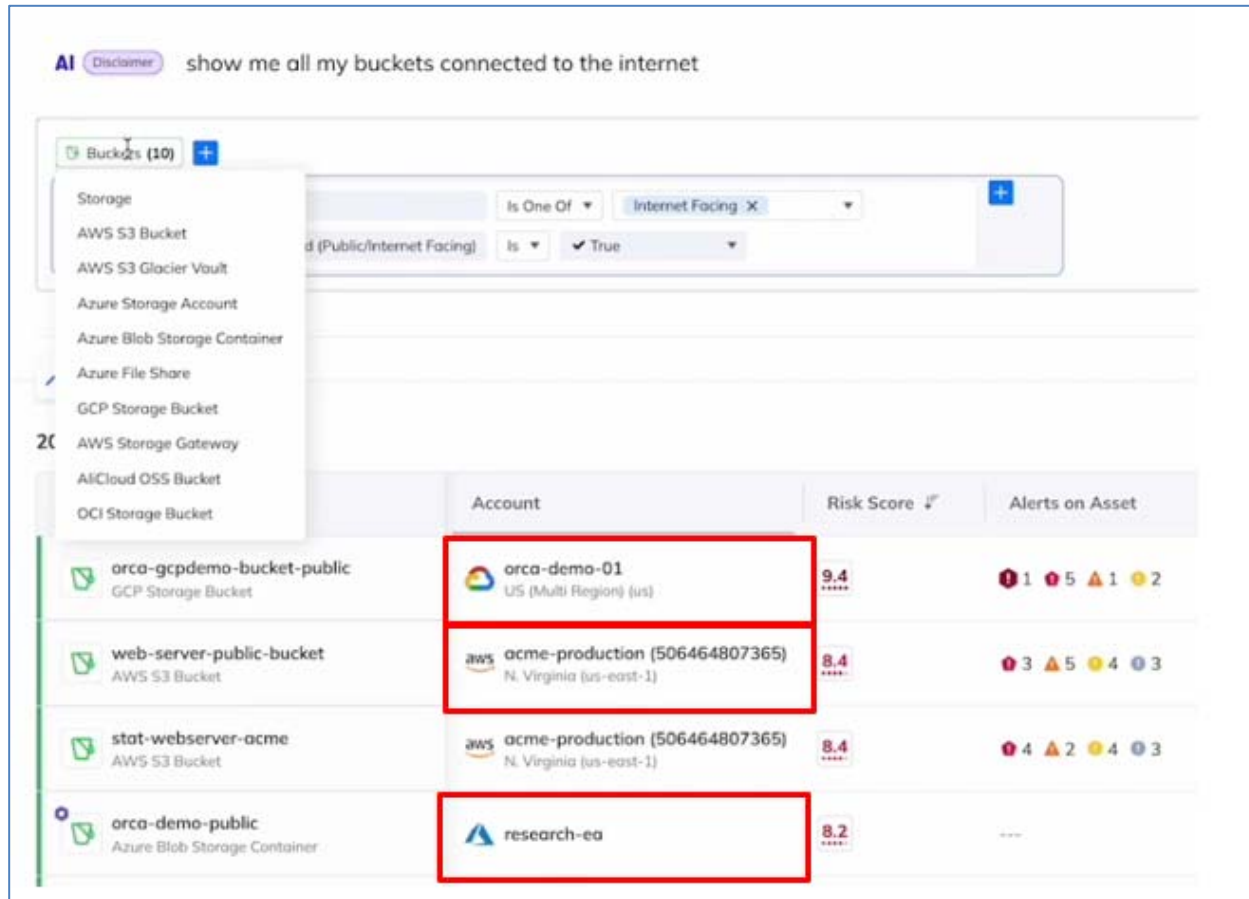
say show me all my buckets connected to the internet. Now we’re taking that query, the natural

language that I used, so think about me being someone who perhaps doesn’t know all the

different types of buckets that exist across all the different cloud service providers. Well, now I

don’t need to know that I’m looking for a GCP storage bucket or an S3 bucket or any other parts

of the storage.”).



61. Claim 1 further recites “genericizing each of the network entities based on the respective collected network entity data to generate a plurality of generic network entities, wherein a generic network entity includes a generic representation of respective network entities from different cloud computing platforms of the plurality of cloud computing platforms;” On information and belief, Orca practices this step by, for example but not limited to, Orca’s Attack Path Analysis “collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.” See, e.g., *Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/> (“To fully understand where your organization’s most critical weaknesses are, it is important to view risks as an interrelated chain,

rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.”). *See Key Security Capabilities in Kubernetes,*

<https://orca.security/resources/blog/kubernetes-security-capabilities-policies/>.

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

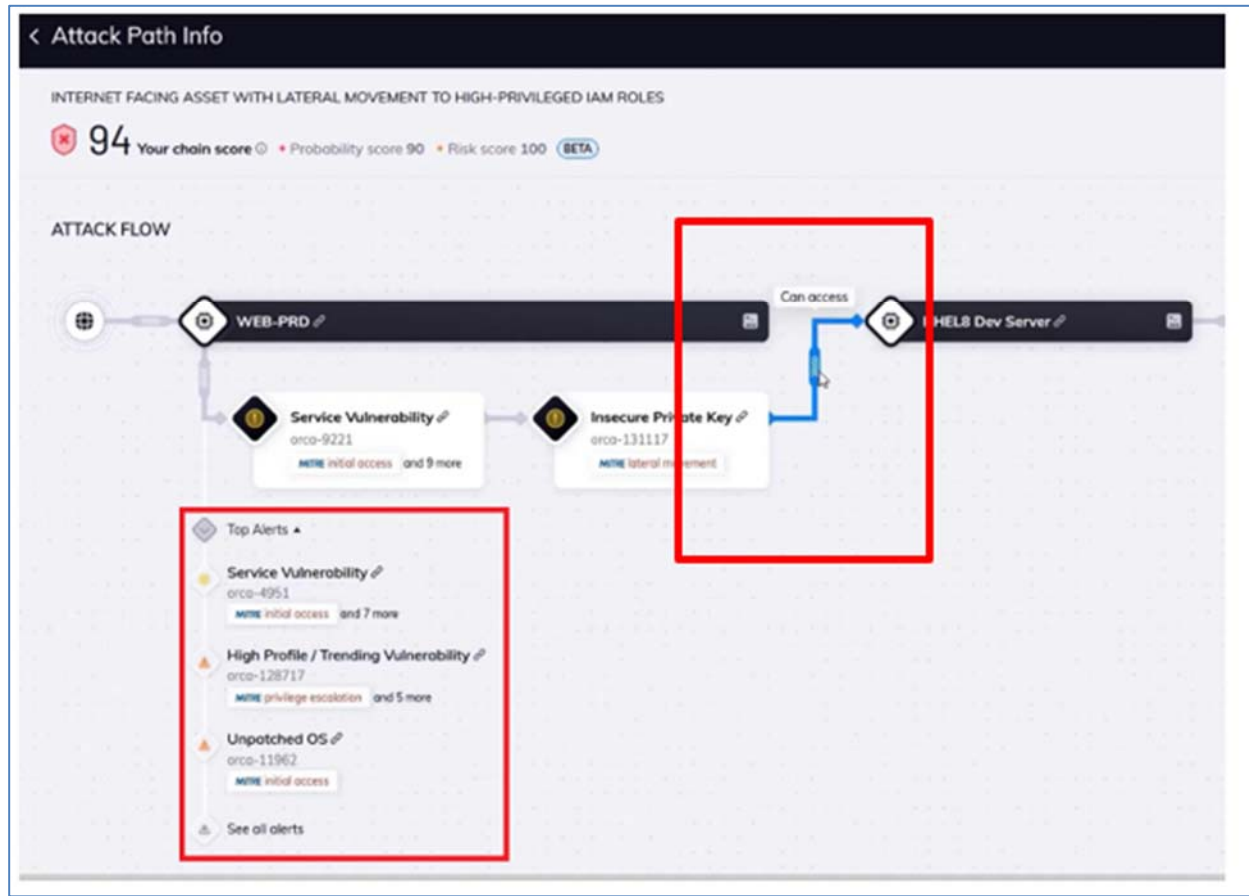
Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the core capabilities of a Cloud-Native Application Protection Platform (CNAPP):

As a further example, Orca displays the generic entities in their attack path visualization. *See id.* (“Attack Path visualization of how an attacker can access sensitive company data”); *see, e.g.,*

Orca Bytes: Attack Path Analysis, <https://www.youtube.com/watch?v=MJkO8UfQa-8>.



62. Claim 1 further recites “generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of generic network entities and relations between the generic network entities of the plurality of network entities; and. . . .” On information and belief, Orca practices this step as shown by, for example but not limited to, describing operation of Orca’s Attack Path Analysis as “representing attack paths in a visual graph with contextual data on all relevant cloud entities” and by using their “attack path visualization” tool. *See, e.g., Cloud Attack Path Analysis: Work*

Smarter Not Harder, <https://orca.security/resources/blog/cloud-attack-path-analysis/>.

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

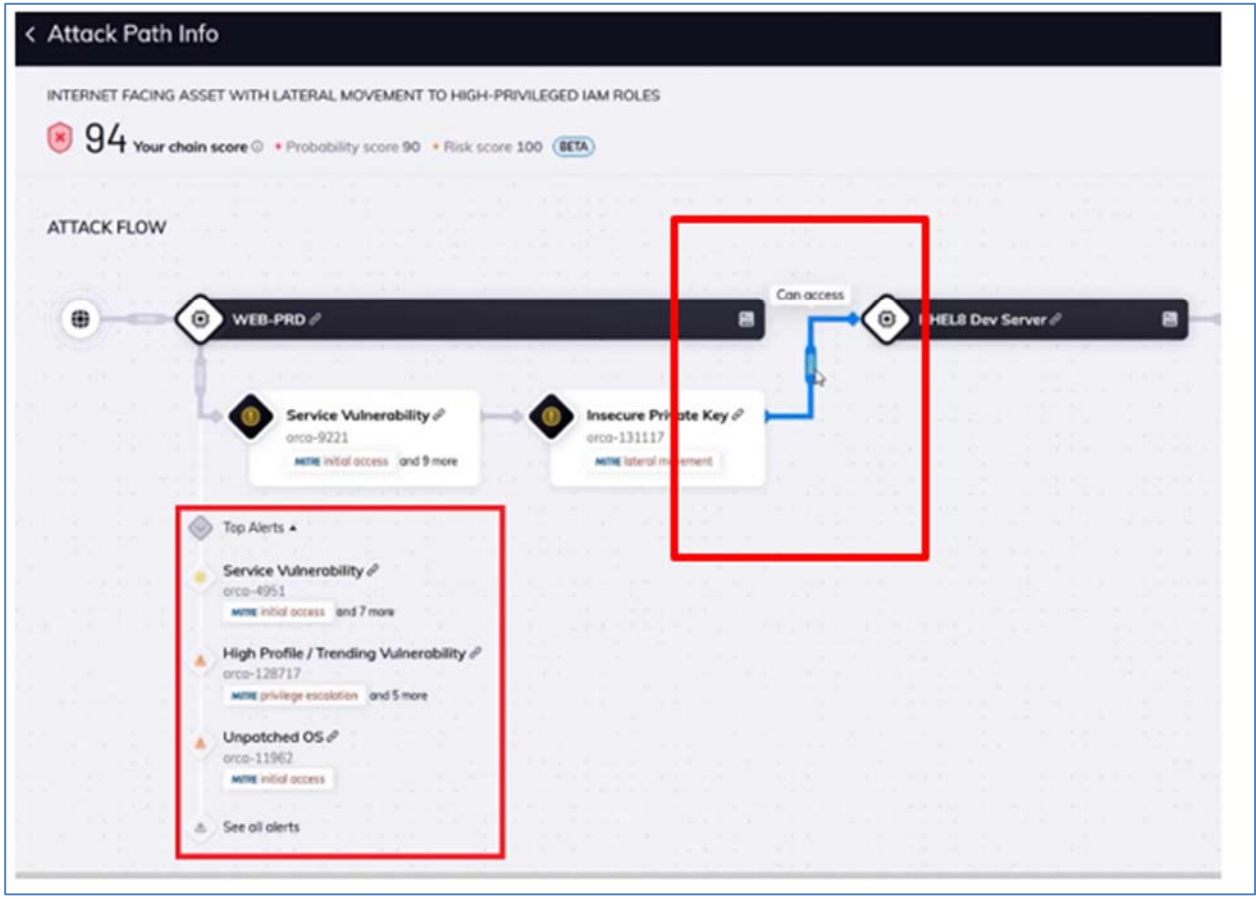
Orca then combines this information with the location of the organization's most valuable assets, or 'crown jewels', and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

63. As a further example, on information and belief, Orca's network graph is a multi-dimensional data structure representing network entities and each entity contains properties,

including but not limited to, entity names and associated alerts.

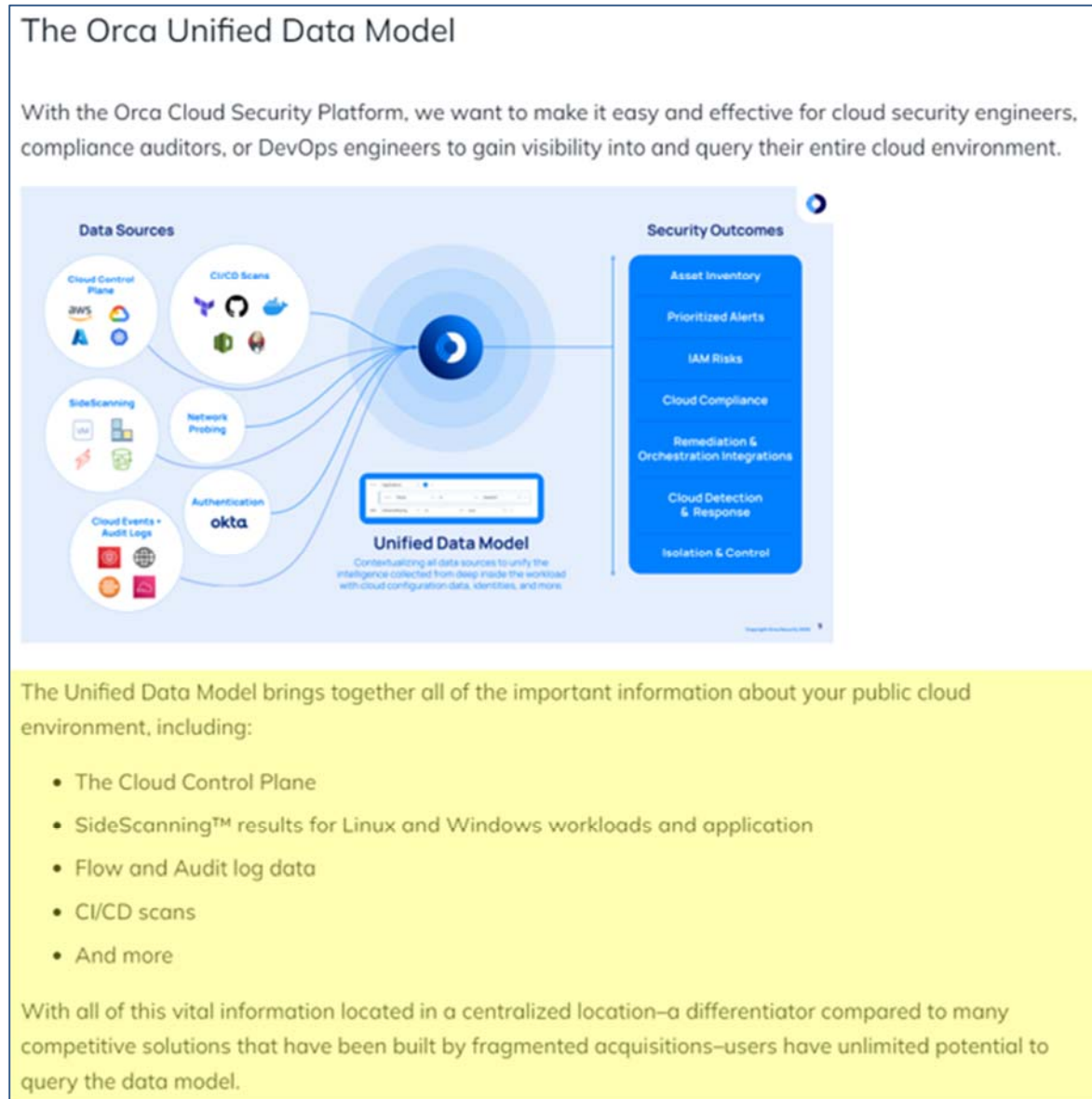


64. Claim 1 further recites “creating at least one imputed entity, wherein the at least one imputed entity is a generic network entity representing an executed platform functionality, and wherein the executed platform functionality is different than a network entity; and” On information and belief, Orca practices this step by, for example but not limited to, including firewall rules in their graph visualization of cloud assets. *See, e.g., Navigating Your Cloud Estate to Understand External Exposure*, <https://orca.security/resources/blog/understanding-external-exposure-with-graph-visualization/>.



65. Finally, claim 1 further recites “storing the generated network graph.” Orca’s public blog posts confirm that Orca practices this step by, for example but not limited to, using Orca’s Unified Data Model to store a network graph. *See, e.g., Cloud Security Simplified: Easily Query Your Entire Cloud Environment*, <https://orca.security/resources/blog/orca-sonar-data-query-builder/> (“The Unified Data Model brings together all of the important information about your public cloud environment . . . ”); and its “Data Security and Posture Management.” *See e.g., Data Security and Posture Management*, <https://orca.security/platform/data-security-and-posture-management-dspm/> (“The Orca Cloud Security Platform performs continuous discovery of data stores across your cloud estate, and alerts to security and compliance risks, without requiring any additional tools. Instead of focusing solely on data security, Orca delivers a comprehensive, context-driven picture of sensitive data exposure, enabling organizations to prioritize risks effectively, reduce alert fatigue, and stay focused on what matters most—from a

single platform.”).



66. On information and belief, Orca was aware of the '896 patent and Orca's infringement thereof prior to these counterclaims at least due to Orca's monitoring of Wiz patents as shown by, in its original complaint in this action, Orca cited and quoted from Wiz's U.S. Patent No. 11,374,982. *See, e.g.*, D.I. 1, ¶ 22. Further, as demonstrated above Orca has repeatedly shown a culture of copying Wiz. This is just one more example of Orca seeing Wiz's success and copying instead of innovating. Moreover, Orca is aware of the '896 patent and

Orca's infringement thereof at least as of the filing of these counterclaims. Accordingly, Orca has and continues to willfully infringement the '896 patent.

67. Orca has induced and continues to induce infringement of one or more claims of the '896 patent by, for example but not limited to, encouraging customers to its Attack Path Analysis in a manner that directly infringes those claims. Despite its knowledge of the existence of the '896 patent, since at least the filing of this Counterclaim, Orca, upon information and belief, continues to encourage, instruct, enable and otherwise cause its customers to use its Attack Path Analysis in a manner that infringes one or more claims of the '896 patent. Upon information and belief, Orca specifically intends that its customers use its Attack Path Analysis in a manner that infringes one or more claims of the '896 patent by, at a minimum, providing instructions and/or support documentation directing customers on how to use its Attack Path Analysis in an infringing manner, in violation of 35 U.S.C. § 271(b). For example, Orca's public blog posts cited above provide instructions and encourage customers to practice all steps of the claimed method stating "To fully understand where your organization's most critical weaknesses are, it is important to view risks as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard." *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/>. Further, Orca provides video explanation of Attack Path Analysis stating, "with Orca's new attack path analysis and prioritization capabilities, security teams can now laser focus on a small number of prioritized attack paths or alert on combinations that endanger the company's most critical assets, and every path and link in the path is scored so you can

pinpoint exactly which risks need to be remediated.” *See Orca Bytes: Attack Path Analysis* (Mar. 31, 2022), <https://www.youtube.com/watch?v=MJkO8UfQa-8>.

68. Orca has contributed and continues to contribute to the infringement of one or more claims of the ’896 patent. Upon information and belief, Orca knows that the Accused Product is especially made and/or adapted for users to infringe one or more claims of the ’896 patent and is not a staple article or commodity of commerce suitable for substantial non-infringing use. Because Orca included features, such as for example but not limited to Attack Path Analysis, in Orca’s products, Orca intends for customers to use it. Upon information and belief, the Attack Path Analysis feature has no suitable use that is non-infringing, and therefore Orca intends for customers to use its Attack Path Analysis in an infringing manner. Orca’s sales of products including Attack Path Analysis constitute contributory infringement in violation of 35 U.S.C. § 271(c).

COUNTERCLAIM III
(INFRINGEMENT OF U.S. PATENT NO. 11,936,693)

69. Wiz is the sole and exclusive owner, by assignment, of all rights, title and interest in U.S. Patent No. 11,936,693 (the “’693 patent”), entitled “System and Method for Applying a Policy on a Network Path.” The ’693 patent was duly and legally issued by the U.S. Patent and Trademark Office on Mar. 19, 2024. The named inventors of the ’693 patent are Roy Reznik, Matilda Lidgi, Shai Keren, and Eliran Marom. A copy of the ’693 patent is attached as Exhibit C.

70. The ’693 patent generally relates to applying a policy on a network path to a reachable resource in a cloud computing environment. *See* ’693 patent at 2:44-54. The policy may include a conditional rule which, if not met, initiates a mitigating action. *Id.* at 2:54-56.

71. The '693 patent discloses “initiating active inspection for each network path of a plurality of network paths; storing an indicator to indicate that a first network path of the plurality of network paths is a valid path, in response to determining that the reachable resource is accessible from the external network; and applying the policy on the first network path.” *Id.* at 2:64-3:3. The '693 patent further discloses “initiating the mitigation action on the reachable resource . . . where the mitigation action includes any one of: revoking access to the reachable resource, revoking access from the reachable resource, closing a port of the reachable resource, generating a notification, generating an alert, and any combination thereof.” *Id.* at 3:11-16.

72. Orca has infringed and continues to directly infringe one or more claims of the '693 patent by making, using, selling, offering for sale, and/or importing into the United States without authority or license, the Orca Platform with Attack Path Analysis and Auto Remediation in violation of 35 U.S.C. § 271(a). Orca's infringement includes infringement of, for example, claim 1 of the '693 patent.

73. Claim 1 of the '693 patent recites:

1. A method for applying a policy on a network path, comprising:
 - selecting a reachable resource having a network path to access the reachable resource, wherein the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment;
 - actively inspecting the network path to determine if the network path of the reachable resource is accessible from the external network;
 - storing an indicator to indicate that the network path is a valid path, in response to determining that the reachable resource is accessible from the external network;
 - applying a policy on the valid path, wherein the policy includes a conditional rule;
 - initiating a mitigation action, in response to determining that the conditional rule is not met; and
 - applying the policy on another network path, in response to determining that the conditional rule is met.

74. On information and belief, Orca practices each and every limitation of claim 1 of the '693 patent by and through the use of the Attack Path Analysis and Auto-Remediation.

75. The preamble of claim 1 recites “[a] method for applying a policy on a network path, comprising:” To the extent the preamble is limiting, Orca practices this step by, for example but not limited to, using its Attack Path Analysis product to tag network paths. *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/> (“For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.”); *see also id* (“Each attack vector in the path includes tags for easy identification and filtering, including applicable MITRE ATT&CK categories.”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

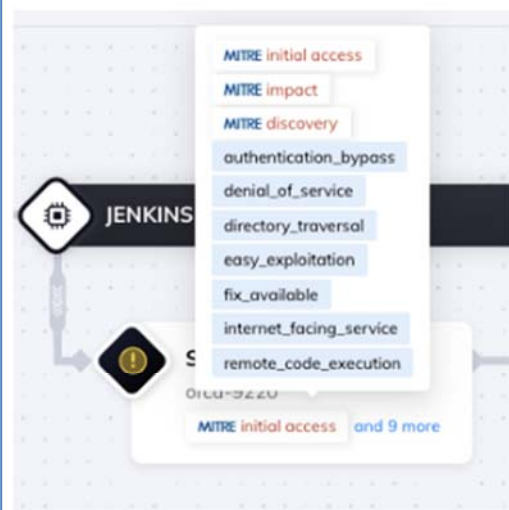
Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization's most valuable assets, or 'crown jewels', and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

Each attack vector in the path includes tags for easy identification and filtering, including applicable [MITRE ATT&CK](#) categories.



Orca shows detailed information for each step

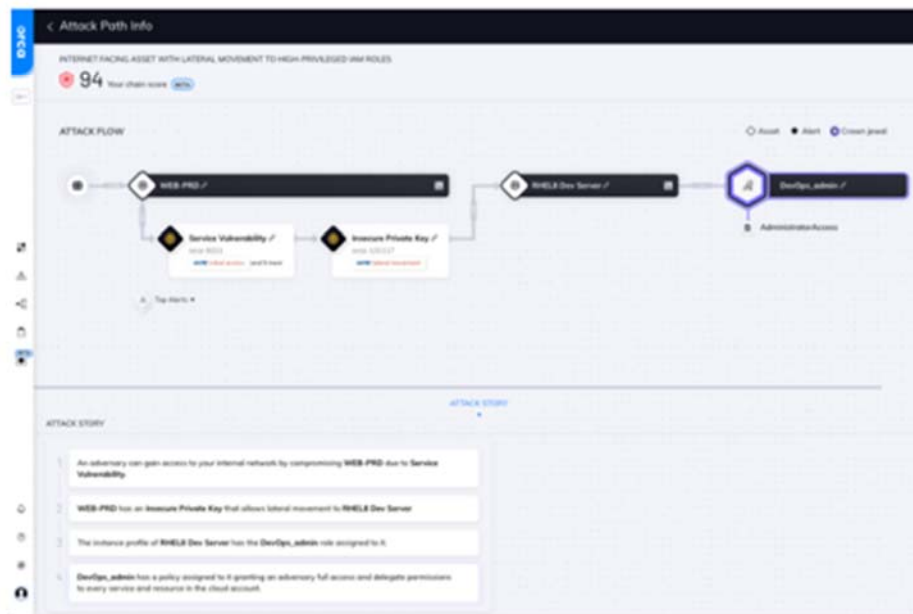
76. Claim 1 further recites “selecting a reachable resource having a network path to access the reachable resource, wherein the reachable resource is a cloud object deployed in a

cloud computing environment, having access to an external network which is external to the cloud computing environment.” Orca’s public blog posts confirm that Orca practices this step by, for example but not limited to, using Orca’s Attack Path Analysis. Orca scans and selects resources that are external or internet facing in a cloud environment. *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis> (“For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.”).

77. Orca further boasts that “[b]y understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.” *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*,

<https://orca.security/resources/blog/cloud-attack-path-analysis/>.

To fully understand where your organization's most critical weaknesses are, it is important to view risks as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.



Orca shows the steps an attacker can take to reach the company's crown jewels

78. Further, Orca considers “Accessibility” as a factor when providing Attack Path Scoring and Prioritization. Accessibility includes whether or not a resource is “internet facing.” See, e.g., *id.* (“Accessibility: Is the attack path Internet facing? How easy is it to exploit the

initial entry point to the attack path?”).

Orca Security Attack Path Scoring and Prioritization

Orca assigns an overall score (from 0 to 99) to each attack path and scores each attack vector that makes up the attack path. To calculate the score, Orca uses an advanced algorithm that takes the following factors into account:

1. **Severity:** How severe is the underlying security issue? For example, what type of threat is it, how likely is it to be exploited, and what is the CVSS score?
2. **Accessibility:** Is the attack path Internet facing? How easy is it to exploit the initial entry point to the attack path?
3. **Lateral Movement Risk:** Is there [lateral movement risk](#)? If so, how easy is it to exploit, and how many hops do you need to reach your end goal?
4. **Access Level:** Does the risk provide read only access, or read and write access? If the risk allows privilege exploitation, what level of privileges does it allow?
5. **Business Impact:** How critical is the target that the attack path exposes? Is it Personal Identifiable Information (PII), or other sensitive information such as intellectual property? Is it a production server that is essential to the business?



How Orca Security scores attack paths

79. Claim 1 further recites “actively inspecting the network path to determine if the network path of the reachable resource is accessible from the external network;” Orca’s public blog posts confirm that Orca practices this step by, for example but not limited to, using Orca’s Attack Path Analysis. Orca states that “it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including

information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.” Further this model is used to determine the accessibility or reachability of an object. *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/> (“Orca assigns an overall score (from 0 to 99) to each attack path and scores each attack vector that makes up the attack path. To calculate the score, Orca uses an advanced algorithm that takes the following factors into account: . . . 2. Accessibility: Is the attack path Internet facing? How easy is it to exploit the initial entry point to the attack path?”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

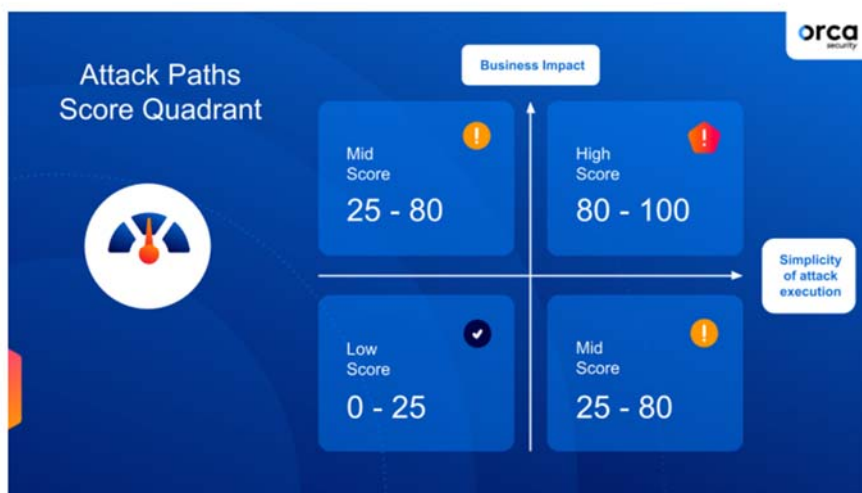
For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

Orca Security Attack Path Scoring and Prioritization

Orca assigns an overall score (from 0 to 99) to each attack path and scores each attack vector that makes up the attack path. To calculate the score, Orca uses an advanced algorithm that takes the following factors into account:

1. **Severity:** How severe is the underlying security issue? For example, what type of threat is it, how likely is it to be exploited, and what is the CVSS score?
2. **Accessibility:** Is the attack path Internet facing? How easy is it to exploit the initial entry point to the attack path?
3. **Lateral Movement Risk:** Is there [lateral movement risk](#)? If so, how easy is it to exploit, and how many hops do you need to reach your end goal?
4. **Access Level:** Does the risk provide read only access, or read and write access? If the risk allows privilege exploitation, what level of privileges does it allow?
5. **Business Impact:** How critical is the target that the attack path exposes? Is it Personal Identifiable Information (PII), or other sensitive information such as intellectual property? Is it a production server that is essential to the business?



How Orca Security scores attack paths

80. Claim 1 further recites “storing an indicator to indicate that the network path is a valid path, in response to determining that the reachable resource is accessible from the external network;” Orca’s public blog posts confirm that Orca practices this step. For example, Orca states that “it is essential that the cloud security platform utilizes a unified data model that collects and correlates contextual data on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.”

Further this model is used to determine the accessibility or reachability of an object. *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder,*

<https://orca.security/resources/blog/cloud-attack-path-analysis/> (“Orca assigns an overall score (from 0 to 99) to each attack path and scores each attack vector that makes up the attack path. To calculate the score, Orca uses an advanced algorithm that takes the following factors into account: . . . 2. Accessibility: Is the attack path Internet facing? How easy is it to exploit the initial entry point to the attack path? . . . Each attack vector in the path includes tags for easy identification and filtering, including applicable MITRE ATT&CK categories”).

Cloud Attack Path Analysis Automatically Correlates Multiple Alerts into One Interactive View

Orca defines Attack Path Analysis as the automatic identification of risk combinations that create dangerous attack paths that can be exploited by attackers. This includes representing attack paths in a visual graph with contextual data on all relevant cloud entities and their risks across vulnerability status, misconfiguration risks, trust and authorization, and data as well as the relations between them.

Orca then combines this information with the location of the organization’s most valuable assets, or ‘crown jewels’, and assigns each attack path a Business Impact Score. This allows security teams to immediately understand which attack paths are the most critical to the business, so they can remediate those first.

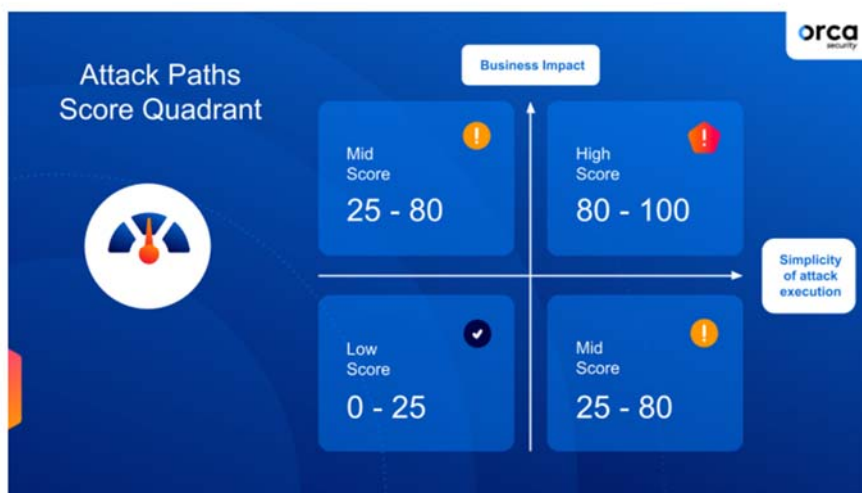
For attack path analysis to be truly beneficial, it is essential that the cloud security platform utilizes a unified data model that collects and correlates [contextual data](#) on each asset, including information on potential risks in the cloud workload and configuration as well as external and internal cloud connectivity.

Industry analysts are increasingly recognizing the value and need for more detailed attack path analysis as part of cloud security solutions. Gartner lists attack path analysis as one of the [core capabilities of a Cloud-Native Application Protection Platform \(CNAPP\)](#):

Orca Security Attack Path Scoring and Prioritization

Orca assigns an overall score (from 0 to 99) to each attack path and scores each attack vector that makes up the attack path. To calculate the score, Orca uses an advanced algorithm that takes the following factors into account:

1. **Severity:** How severe is the underlying security issue? For example, what type of threat is it, how likely is it to be exploited, and what is the CVSS score?
2. **Accessibility:** Is the attack path Internet facing? How easy is it to exploit the initial entry point to the attack path?
3. **Lateral Movement Risk:** Is there [lateral movement risk](#)? If so, how easy is it to exploit, and how many hops do you need to reach your end goal?
4. **Access Level:** Does the risk provide read only access, or read and write access? If the risk allows privilege exploitation, what level of privileges does it allow?
5. **Business Impact:** How critical is the target that the attack path exposes? Is it Personal Identifiable Information (PII), or other sensitive information such as intellectual property? Is it a production server that is essential to the business?



How Orca Security scores attack paths

Each attack vector in the path includes tags for easy identification and filtering, including applicable [MITRE ATT&CK](#) categories.



Orca shows detailed information for each step

See also, *Data Security and Posture Management*, <https://orca.security/platform/data-security-and-posture-management-dspm/> (“Orca’s DSPM dashboard provides data security teams with an overview of their cloud data stores, sensitive data, and security and compliance alerts. Orca scans managed, unmanaged, and shadow data, giving security teams wide and deep visibility

into where their data resides.”).

Discover and classify data in your cloud

Orca’s DSPM dashboard provides data security teams with an overview of their cloud data stores, [sensitive data](#), and security and compliance alerts. Orca scans managed, unmanaged, and shadow data, giving security teams wide and deep visibility into where their data resides.

- ✓ Get a multi-cloud inventory of data storage assets—including databases, and files in virtual machines, storage buckets, and containers.
- ✓ Know which data stores contain sensitive data and of what type—including PII, [PCI](#), PHI, or financial information—for both security and regulatory purposes.
- ✓ Leverage interactive graphs that show the location and relationship between data stores and other cloud entities.

81. Claim 1 further recites “applying a policy on the valid path, wherein the policy includes a conditional rule. . . .” On information and belief, Orca practices this step through the use of Auto Remediation. Orca states for example, that it applies “auto remediation” policies on assets with common and complex security alerts. *See, e.g. Manage Cloud Security Risks with Auto-Remediation*, <https://orca.security/resources/blog/manage-security-risks-auto-remediation/>.

(“Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity. With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon).”).

Automatic Remediation as an Approach to Better, Faster Security

One of the most complex parts of the alert life cycle is the manual remediation process; usually, it will include either step-by-step mitigation instructions on the console or copying and pasting command after command to the CLI.

However, an excessive number of alerts can quickly become unmanageable. To assist with this challenge, Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity.

With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon).



See also, <https://orca.security/resources/video/auto-remediation-demo/> (Demonstrating setting up

a policy on an internet facing asset.)

The screenshot displays a security dashboard for an asset named 'acme-dev2948', which is an AWS S3 Bucket. At the top, it shows 'ALERTS ON ASSET' with a total of 4 alerts: 1 Compromise, 0 Imminent Compromise, and 3 Hazardous. A secondary bar indicates '2 Data at risk', '1 Data protection', and '1' (likely another category). The 'ASSET INFORMATION' section on the right lists 'Top alerts' with a link to 'SEE IN ALERTS'. A table of alerts is shown below:

Alert	Last seen
Malware jeicar.com	8 hours ago 2022, Sep 28
S3 Bucket Should Enforce H... acme-dev2948	8 hours ago 2022, Sep 28
S3 Bucket Allows Public REA... acme-dev2948	8 hours ago 2022, Sep 28
S3 Bucket Allows Public REA... acme-dev2948	8 hours ago 2022, Sep 28
Bucket Allows Unencrypted ... acme-dev2948	8 hours ago 2022, Sep 28

The 'Additional Information' section shows the asset was created 2 years ago (2020, Oct 16) and is up for 6 hours (2022, Sep 28, 03:35). The 'Attack vector' diagram shows a path from 'public' to 'acme-dev2948'.

82. Claim 1 further recites “initiating a mitigation action, in response to determining that the conditional rule is not met; and” Orca’s public blog posts confirm that Orca practices this step. Orca states for example, that it applies “auto remediation” policies on assets with common and complex security alerts when security rules are not met. These remediation actions can include hardening permissive access rules to assets and blocking specific ports. *See, e.g. Manage Cloud Security Risks with Auto-Remediation,*

<https://orca.security/resources/blog/manage-security-risks-auto-remediation/> (“Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity. With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon) For example, you can automatically harden permissive access on insecure security group rules and block specific ports while creating a Jira ticket notifying your DevOps team with more details. Our remediation capabilities give you the option to select the action based on your requirements.”).

Automatic Remediation as an Approach to Better, Faster Security

One of the most complex parts of the alert life cycle is the manual remediation process; usually, it will include either step-by-step mitigation instructions on the console or copying and pasting command after command to the CLI.

However, an excessive number of alerts can quickly become unmanageable. To assist with this challenge, Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity.

With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon).

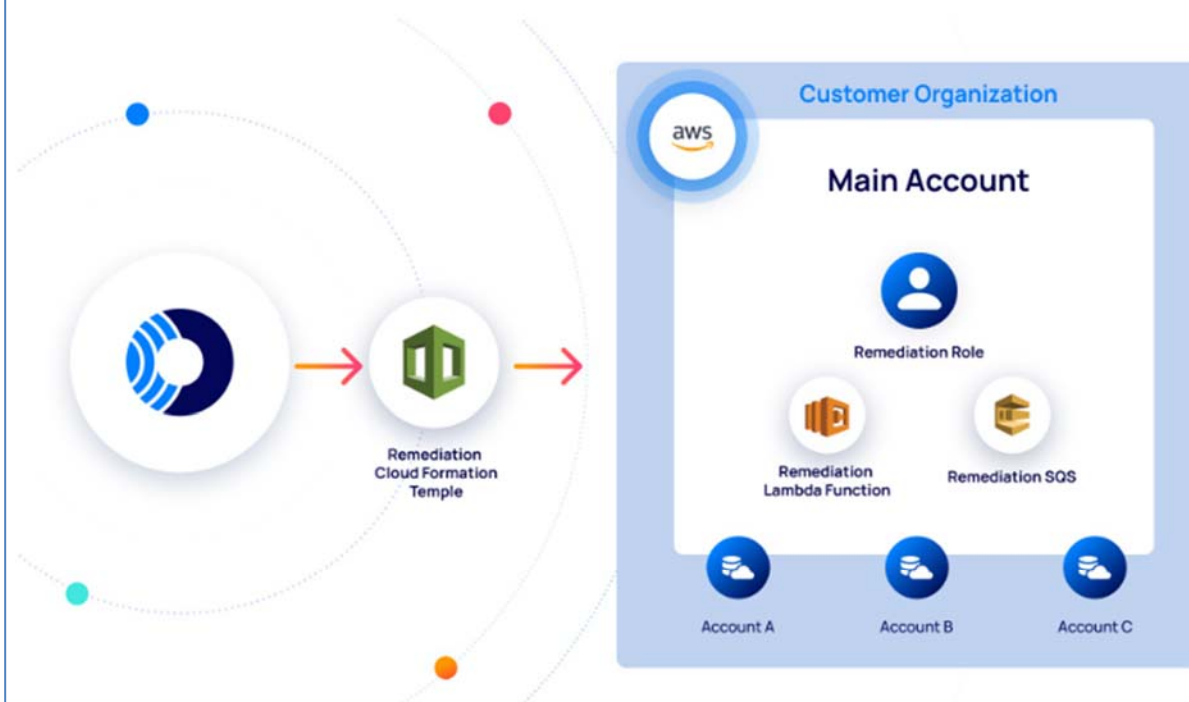


How Does Orca Auto-Remediation Work?

Orca's Auto-Remediation is yet another AWS CloudFormation stack deployed in your environment, which means that you do not need to provide write access to Orca.



Instead, the remediation solution infrastructure resides in the customer's environment.



The Orca Cloud Security Platform sends remediation instructions to an AWS SQS Queue, triggering a Lambda function. The function then calls the appropriate action to remediate the alert(s), as shown in the diagram below.



Using Orca's Auto-Remediation, you can quickly and easily remediate security issues in your cloud environment. For example, you can automatically harden permissive access on insecure security group rules and block specific ports while creating a Jira ticket notifying your DevOps team with more details. Our remediation capabilities give you the option to select the action based on your requirements.


See also, <https://orca.security/resources/video/auto-remediation-demo/> (Demonstrating applying auto remediation to a S3 bucket that allows public read access).

  Data at risk
S3 Bucket Allows Public READ_...

  [Take action](#)

ALERT INFO **ASSET INFO**

Status **Integration**

Open 

Summary

Ensure that your S3 buckets content permissions details cannot be viewed by anonymous users in order to protect against unauthorized access. An S3 bucket that grants READ_ACP (view permissions) access to everyone can

[SHOW MORE](#)

Remediation

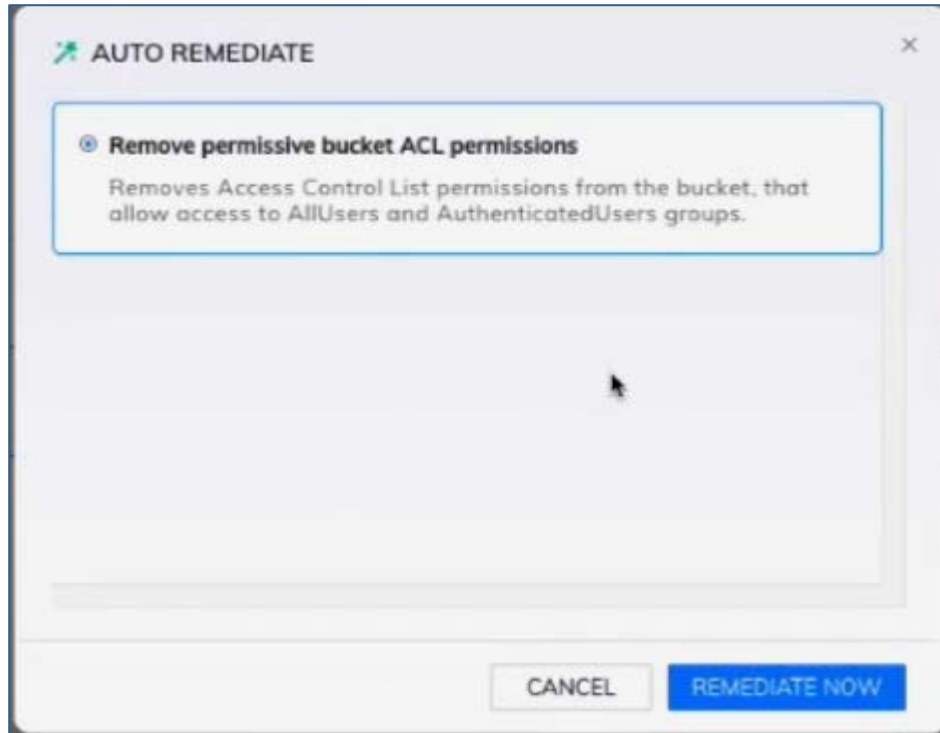
Auto

[Choose auto remediation](#)

or

Manual

Change the acme-dev2948 bucket policy to block public READ_ACP access



83. Finally claim 1 recites “applying the policy on another network path, in response to determining that the conditional rule is met.” On information and belief, Orca practices this step. Orca states for example, that it “Automate[s] routine remediation tasks and processes,” “Reduce[s] redundancy,” and that “you can configure automation rules which remediate alerts as they are detected.” See, e.g. *Manage Cloud Security Risks with Auto-Remediation*, <https://orca.security/resources/blog/manage-security-risks-auto-remediation/>.

84.

85. applies “auto remediation” policies on assets with common and complex security alerts. These remediation actions can include hardening permissive access rules to assets and blocking specific ports in “security groups” impacting multiple network paths. Orca specifically states that applying these policies and mitigation actions across a multi-cloud environment through automation “is the way to improve accuracy, reduce redundancy, and reduce cost and validation time.” This demonstrates how Orca applies remediation polices across network paths.

See, e.g. *Manage Cloud Security Risks with Auto-Remediation*,

<https://orca.security/resources/blog/manage-security-risks-auto-remediation/>; *see also id.* (“Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity. With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon) . . . For example, you can automatically harden permissive access on insecure security group rules and block specific ports while creating a Jira ticket notifying your DevOps team with more details. Our remediation capabilities give you the option to select the action based on your requirements . . . Automation is the way to go to improve accuracy, reduce redundancy, and reduce cost and validation time.”).

Automatic Remediation as an Approach to Better, Faster Security

One of the most complex parts of the alert life cycle is the manual remediation process; usually, it will include either step-by-step mitigation instructions on the console or copying and pasting command after command to the CLI.

However, an excessive number of alerts can quickly become unmanageable. To assist with this challenge, Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity.

With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon).

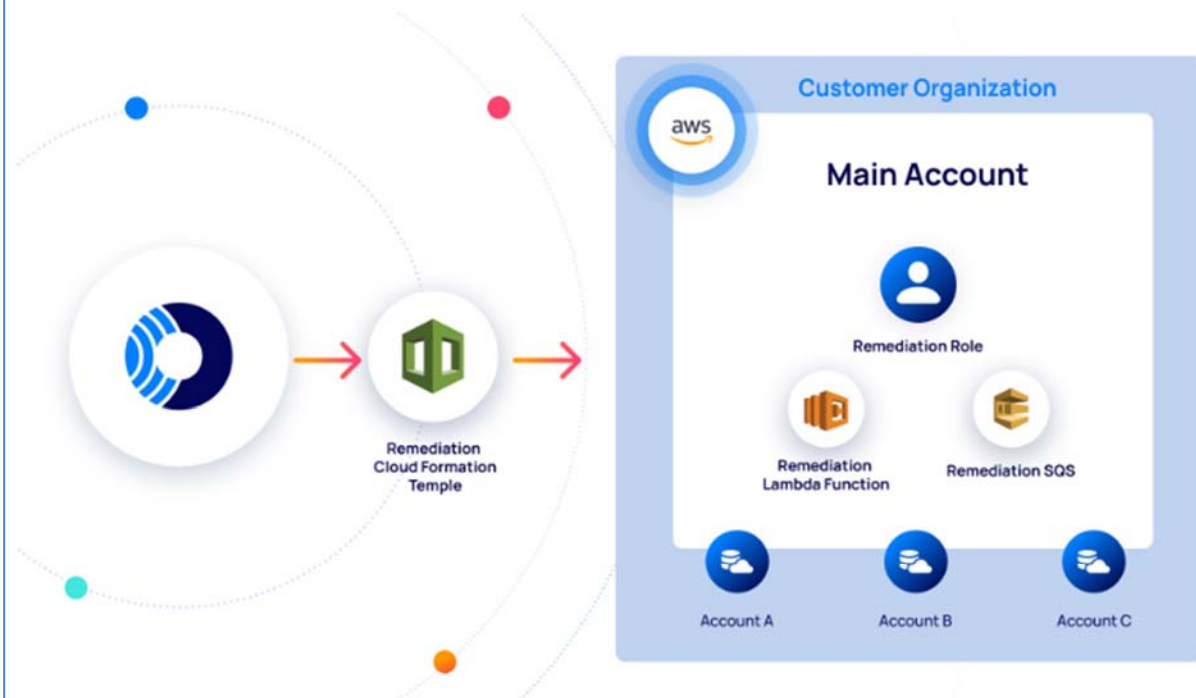


How Does Orca Auto-Remediation Work?

Orca's Auto-Remediation is yet another AWS CloudFormation stack deployed in your environment, which means that you do not need to provide write access to Orca.

Instead, the remediation solution infrastructure resides in the customer's environment.

The Orca Cloud Security Platform sends remediation instructions to an AWS SQS Queue, triggering a Lambda function. The function then calls the appropriate action to remediate the alert(s), as shown in the diagram below.



Using Orca's Auto-Remediation, you can quickly and easily remediate security issues in your cloud environment. For example, you can automatically harden permissive access on insecure security group rules and block specific ports while creating a Jira ticket notifying your DevOps team with more details. Our remediation capabilities give you the option to select the action based on your requirements.

Why Use Orca's Auto-Remediation?

Considering the intricacy of your multi-cloud environments, managing these environments and applications becomes more daunting and complicates your current operational challenges. Automation is the way to go to improve accuracy, reduce redundancy, and reduce cost and validation time.

Auto-Remediation is a self-healing workflow that triggers and responds to alerts or events by executing actions that can prevent or fix the problem. Orca is an event-driven application that uses event-driven automation to resolve policy violations. The Auto-remediation can trigger a serverless function to remediate alerts detected as a result of misconfiguration.

With Orca's Auto-Remediation, your Mean Time to Remediation (MTTR) will be at the bare minimum, thereby improving your security posture and compliance requirements.

Implementing a thorough security framework like [Orca's agentless security solution](#) is the first step in automating your cloud security. Learn how to create custom alerts from queries and integrate these into existing remediation workflows with Orca's platform. Read our [case studies](#) to see how we benefit our customers, or [watch a demo](#) to witness Orca in action. You can also sign up for a [free, no-obligation risk assessment](#) to get started today!

86. Orca is aware of the '693 patent and Orca's infringement thereof at least as of the filing of these counterclaims. Moreover, Orca has a culture of copying Wiz, as explained above. Accordingly, Orca has and continues to willfully infringe the '693 patent.

87. Orca has induced and continues to induce infringement of one or more claims of the '693 patent by encouraging customers to use its Attack Path Analysis and Auto Remediation in a manner that directly infringes those claims. Despite its knowledge of the existence of the '693 patent, since at least the filing of this Counterclaim, Orca, upon information and belief, continues to encourage, instruct, enable and otherwise cause its customers to use its Attack Path Analysis in a manner that infringes one or more claims of the '693 patent. Upon information and belief, Orca specifically intends that its customers use its Attack Path Analysis and Auto Remediation in a manner that infringes one or more claims of the '693 patent by, at a minimum, providing instructions and/or support documentation directing customers on how to use its

Attack Path Analysis and Auto Remediation in an infringing manner, in violation of 35 U.S.C. § 271(b). For example, Orca’s public blog posts cited above provide instructions and encourage customers to practice all steps of the claimed method stating “To fully understand where your organization’s most critical weaknesses are, it is important to view risks as an interrelated chain, rather than just siloed risks. By understanding which combinations are a direct path to your critical assets, security teams can operate strategically by making sure that the risks that break the attack path are remediated first. Orca Security does just that with its new Attack Path Analysis dashboard.” *See, e.g., Cloud Attack Path Analysis: Work Smarter Not Harder*, <https://orca.security/resources/blog/cloud-attack-path-analysis/>. Further, Orca provides instructions on security risk auto remediation stating: “Orca Security introduces Automatic Remediation: a way to quickly resolve common and complex security alerts, such as an Unencrypted S3 Bucket or Security group with permissive access, reducing friction between different groups in the organization and increasing productivity. With Orca Automatic Remediation, you can configure automation rules which remediate alerts as they are detected or click the “Auto-Remediation” button on a specific alert (indicated by a green magic stick icon).” *See, e.g., Manage Cloud Security Risks with Auto-Remediation*, <https://orca.security/resources/blog/manage-security-risks-auto-remediation/>.

88. Orca has contributed and continues to contribute to the infringement of one or more claims of the ’693 patent. Upon information and belief, Orca knows that its Attack Path Analysis and Auto Remediation are especially made and/or adapted for users to infringe one or more claims of the ’693 patent and is not a staple article or commodity of commerce suitable for substantial non-infringing use. Because Orca included the features, such as for example but not limited to, Attack Path Analysis and Auto-Remediation, in Orca’s products, Orca intends for

customers to use it. Upon information and belief, Orca's Attack Path Analysis feature has no suitable use that is non-infringing, and therefore Orca intends for customers to use this feature in an infringing manner. Orca's sales of products including its Attack Path Analysis and Auto Remediation constitute contributory infringement in violation of 35 U.S.C. § 271(c).

COUNTERCLAIM IV
(INFRINGEMENT OF U.S. PATENT NO. 12,001,549)

89. Wiz is the sole and exclusive owner, by assignment, of all rights, title and interest in U.S. Patent No. 12,001,549 (the "'549 patent"), entitled "Cybersecurity Incident Response Techniques Utilizing Artificial Intelligence." The '549 patent was duly and legally issued by the U.S. Patent and Trademark Office on June 4, 2024. The named inventors of the '549 patent are Alon Schindel, Barak Sharoni, Amitai Cohen, Ami Luttwak, Roy Reznik, and Yinon Costica. A copy of the '549 patent is attached as Exhibit D.

90. The '549 patent generally relates to providing a cybersecurity incident response to an incident based on a cybersecurity event and generating a prompt for a large language model to generate a query on a security database for a mitigation action. *See* '549 patent at Abstract. The patent provides a method "where the incident input includes any one of: a query, a statement, and a combination thereof." *Id.* at 4:43-44.

91. The '549 patent discloses "providing the received incident input into a large language model (LLM)[.]" *Id.* at 2:35-36. The disclosed LLM is trained on "a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and any combination thereof." *Id.* at 2:38-41. The '549 patent further discloses "utilizing the LLM to generate an explanation of a security finding." *Id.* at 2:55-56. From this finding, the system may generate and execute a query in a security database. *Id.* at 4:23-41.

92. Finally, the '549 patent discloses that the “[s]ystem may also initiate a mitigation action based on a result of the executed query.” *Id.* at 3:26-28. “A mitigation action includes generating a notification, generating an alert, updating an alert, generating a severity score, updating a severity score, generating a ticket, generating a risk score, updating a risk score, initiating a remediation action, initiating an incident response, a combination thereof, and the like.” *Id.* at 17:15-21.

93. Orca has infringed and continues to directly infringe one or more claims of the '549 patent by making, using, selling, offering for sale, and/or importing into the United States without authority or license, the Orca Platform with AI-Driven Cloud Security in violation of 35 U.S.C. § 271(a). Orca's infringement includes infringement of, for example, claim 1 of the '549 patent.

94. Claim 1 of the '549 patent recites:

1. A method for providing cybersecurity incident response, comprising:
receiving an incident input based on a cybersecurity event;
generating a prompt for a large language model (LLM) based on the received incident input;
configuring the LLM to generate an output based on the generated prompt;
mapping the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, wherein each scenario is associated with an incidence response;
generating a query based on the received incident input and the mapped scenario;
executing the query on a security database, the security database including a representation of a computing environment; and
initiating a mitigation action based on a result of the executed query.

95. On information and belief, Orca practices each and every limitation of claim 1 of the '549 patent by and through the use of AI-Driven Cloud Security.

96. The preamble of claim 1 recites “[a] method for providing cybersecurity incident response, comprising:” To the extent the preamble is limiting, Orca practices this step by, for example but not limited to, using its AI-Driven Cloud Security features to provide a

cybersecurity incident response. *See, e.g., AI-Driven Cloud Security,*

<https://orca.security/platform/ai-cloud-security/> (“Simplify investigations and accelerate

remediation with built-in generative AI.”).



97. Claim 1 further recites “receiving an incident input based on a cybersecurity event. . . .” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, generating alerts in response to a cyber security incident. *See, e.g., AI-Driven Cloud Security,* <https://orca.security/platform/ai-cloud-security/> (Orca’s dynamic alert and asset descriptions greatly simplify investigations, summarizing all the important information that teams need to know in an easily consumable manner, reducing investigation time and improving Mean Time To Remediation (MTTR)).

AI-generated alert and asset descriptions

Orca's dynamic alert and asset descriptions greatly simplify investigations, summarizing all the important information that teams need to know in an easily consumable manner, reducing investigation time and improving Mean Time To Remediation (MTTR).

- ✓ For assets, Orca summarizes which risks are found and of what severity, how many attack paths they are part of, whether the asset is Internet-facing, running or paused, and more.
- ✓ For alerts, Orca explains what the risk is, when it was first found, if it is exploitable, whether there's a fix, how an attacker could abuse it, and more.
- ✓ Where applicable, descriptions contain links to other resources with more information

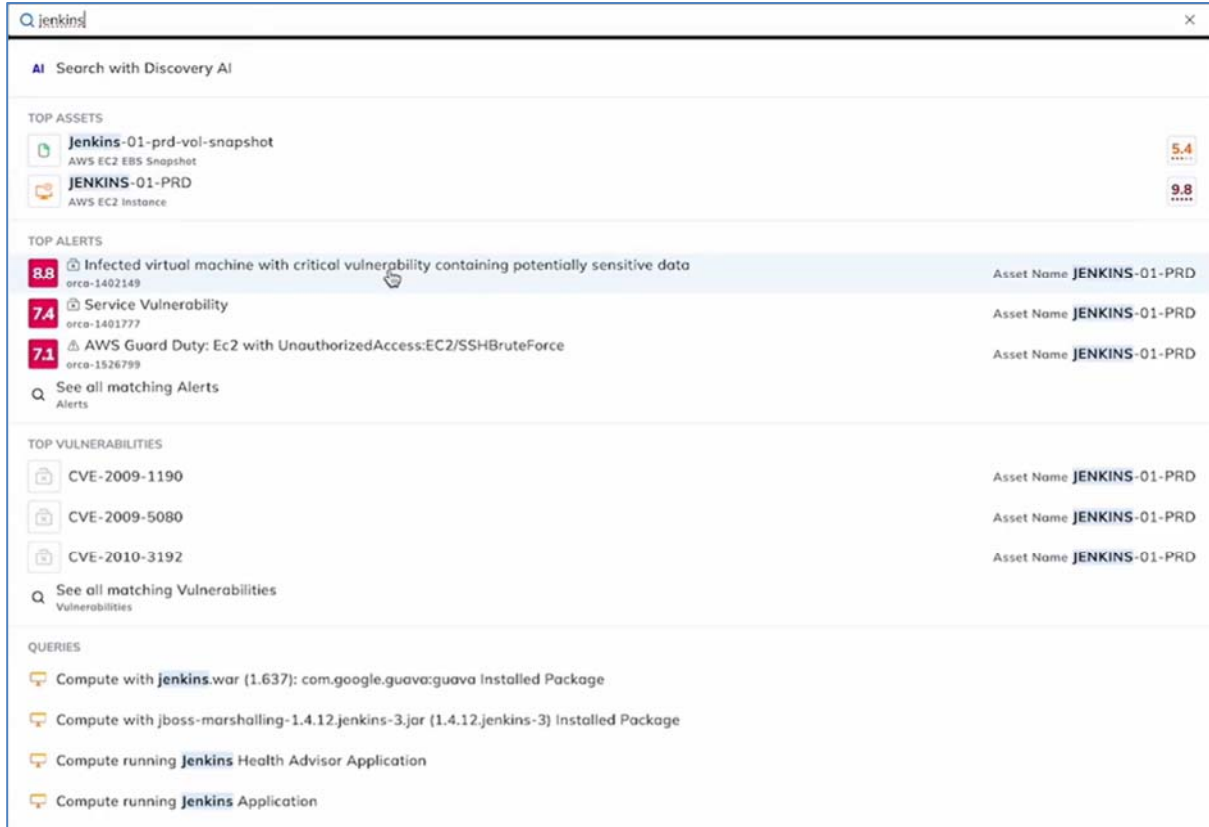
Elevate Cloud Security with Orca AI

"During incident response, seconds matter. Security operators at most firms struggle to build, let alone maintain runbooks that can keep up with the speed of business. Generative AI and LLMs offer proven relief for these teams, so they can remain focused on what matters while continuing to raise the bar on security. Orca's continued use of AI to power remediation shows how it can benefit security teams."



Kathy Wang
CISO and Advisor

Further, Orca provides AI powered search for a user to input a prompt based on the received alert and/or user search. *See, e.g., AI-Driven Cloud Security*, <https://orca.security/platform/ai-cloud-security/> (A user can input the search term “Jenkins” based on an alert).



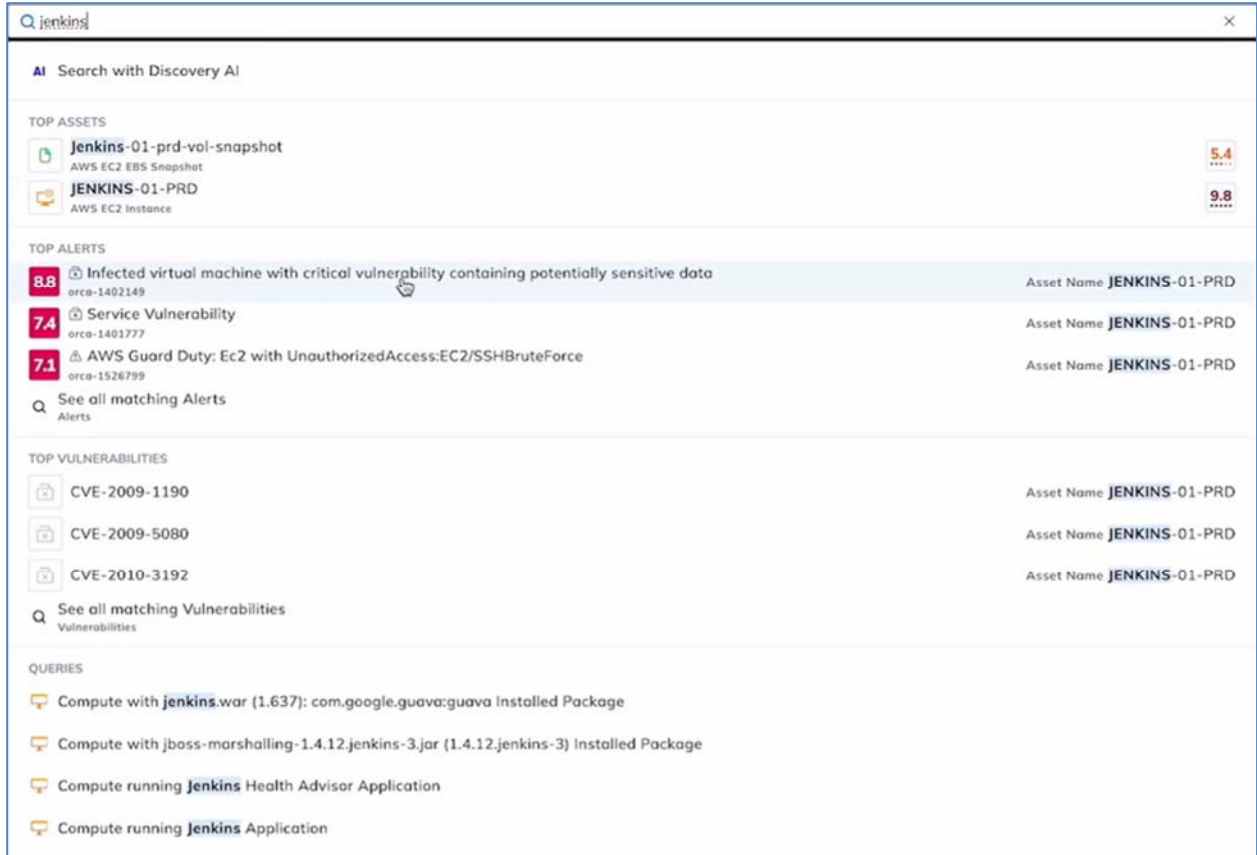
98. Claim 1 further recites “generating a prompt for a large language model (LLM) based on the received incident input” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s “Search with Discovery AI” feature. Orca provides AI powered search for a user to input a prompt based on the received alert and/or user search. *See, e.g., AI-Driven Cloud Security*, <https://orca.security/platform/ai-cloud-security/>

(A user can input the search term “Jenkins” based on an alert).

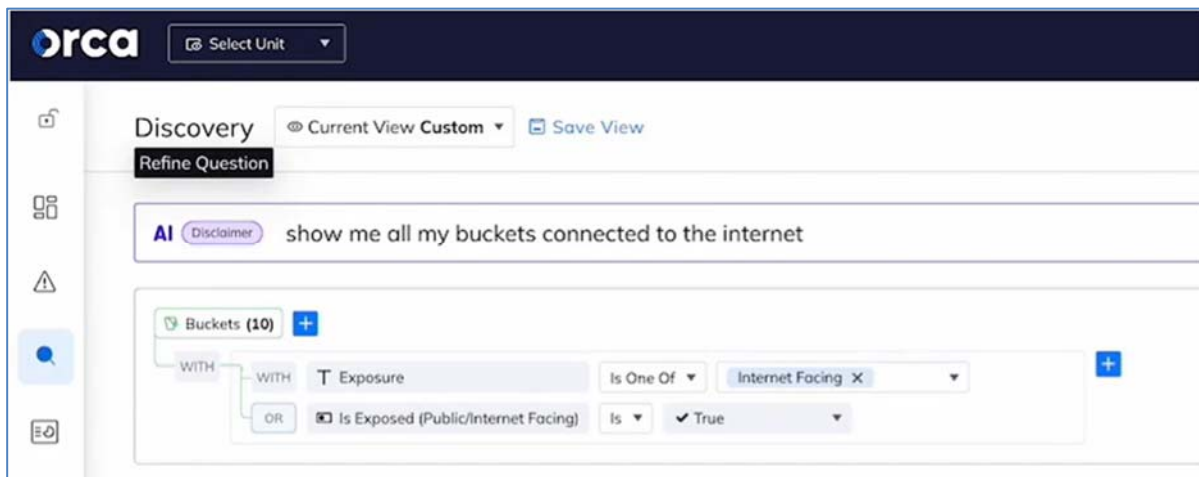
The screenshot displays the Orca AI Search interface with the search term 'jenkins' entered in the search bar. The interface is titled 'AI Search with Discovery AI' and is divided into several sections:

- TOP ASSETS:** Lists two assets: 'Jenkins-01-prd-vol-snapshot' (AWS EC2 EBS Snapshot) with a score of 5.4, and 'JENKINS-01-PRD' (AWS EC2 Instance) with a score of 9.8.
- TOP ALERTS:** Lists three alerts:
 - Alert 8.8: 'Infected virtual machine with critical vulnerability containing potentially sensitive data' (orca-1402149) with Asset Name JENKINS-01-PRD.
 - Alert 7.4: 'Service Vulnerability' (orca-1403777) with Asset Name JENKINS-01-PRD.
 - Alert 7.1: 'AWS Guard Duty: Ec2 with UnauthorizedAccess:EC2/SSHBruteForce' (orca-1526799) with Asset Name JENKINS-01-PRD.
- TOP VULNERABILITIES:** Lists three vulnerabilities: CVE-2009-1190, CVE-2009-5080, and CVE-2010-3192, all with Asset Name JENKINS-01-PRD.
- QUERIES:** Lists four queries:
 - Compute with jenkins.war (1.637): com.google.guava:guava Installed Package
 - Compute with jboss-marshalling-1.4.12.jenkins-3.jar (1.4.12.jenkins-3) Installed Package
 - Compute running Jenkins Health Advisor Application
 - Compute running Jenkins Application

99. Claim 1 further recites “configuring the LLM to generate an output based on the generated prompt” Upon information and belief, Orca practices this step by, for example but not limited to, using Orca’s AI Driven Cloud Security. Using “Search with Discovery AI” to search for the term “jenkins” provides a list of top alerts, top vulnerabilities and queries. *See, e.g., AI-Driven Cloud Security, <https://orca.security/platform/ai-cloud-security/>.*

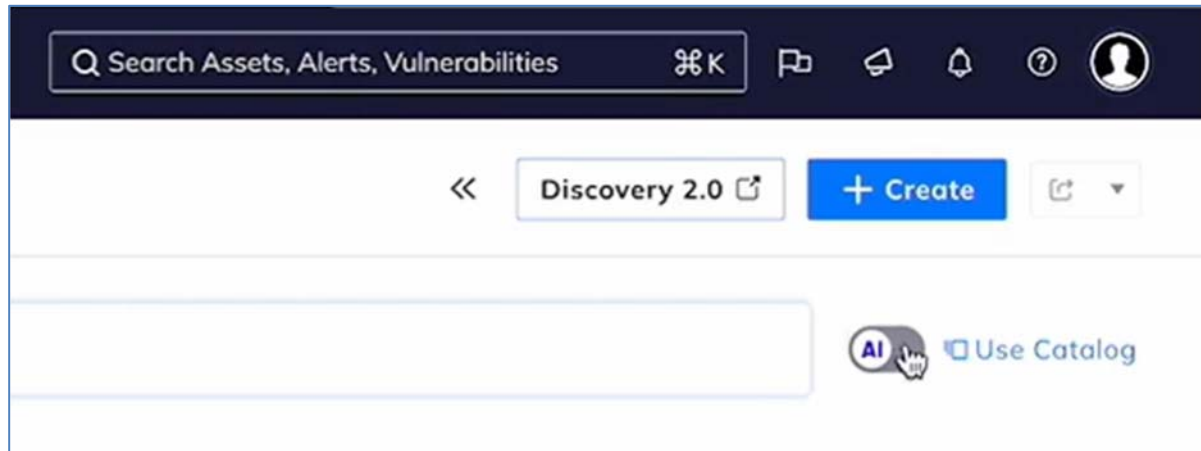


Further, Orca allows user generated prompts for its LLM through its search functionality with AI enabled. See, e.g., *AI-Driven Cloud Security*, <https://orca.security/platform/ai-cloud-security/> (“Show me all my buckets connected to the internet”).



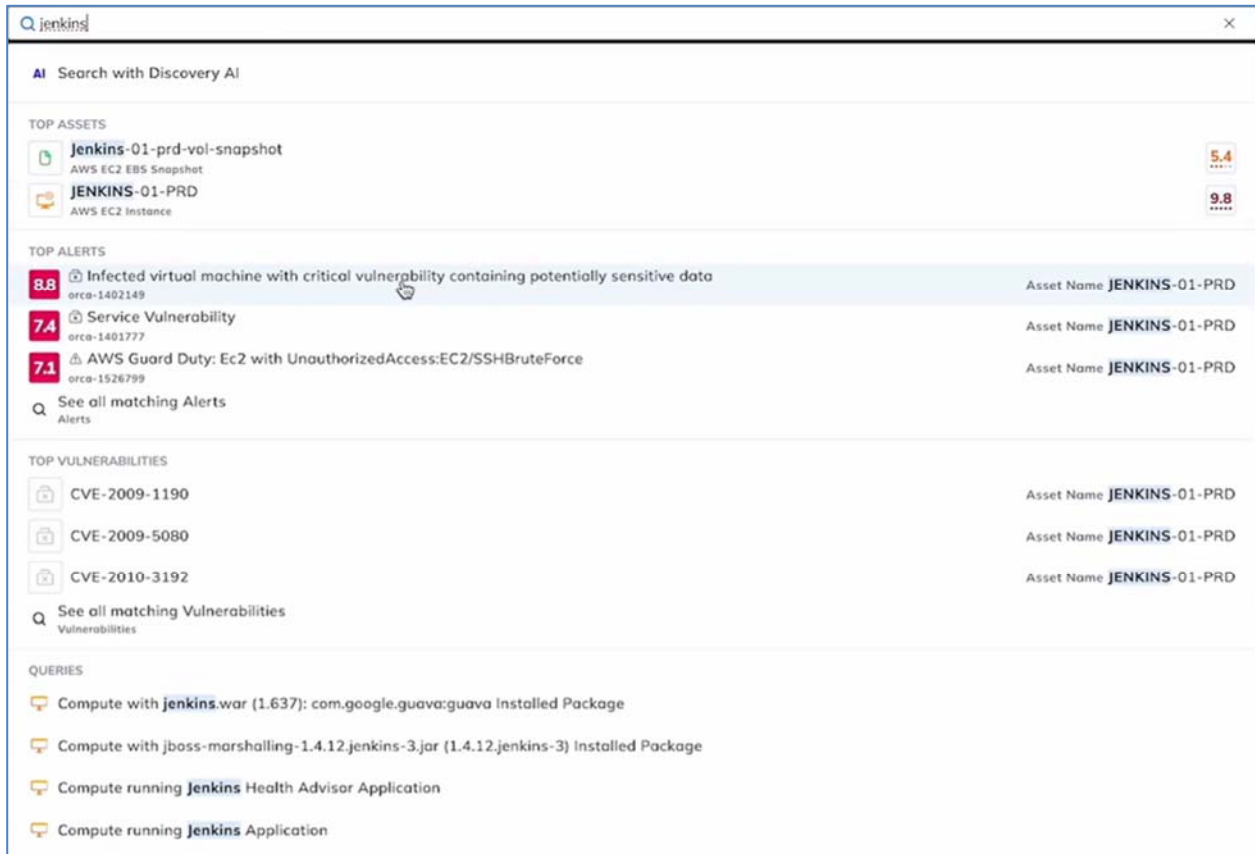
Further, Orca allows users to toggle the LLM to be configured to produce results. See, e.g., *AI-*

Driven Cloud Security, <https://orca.security/platform/ai-cloud-security/>.



100. Claim 1 further recites “mapping the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, wherein each scenario is associated with an incidence response” On information and belief, Orca practices this step by, for example but not limited to, using Orca’s “Search with Discovery AI” to generate a plurality of applicable assets, alerts, vulnerabilities and queries. *See, e.g., AI-Driven Cloud Security, https://orca.security/platform/ai-cloud-security/*. (“I could search for Jenkins. Now, did I mean assets with the name Jenkins? Did I mean alerts that impact assets with the name Jenkins or perhaps vulnerabilities. Or maybe I meant the actual installed package. All of these things derived from that question and clicking into them I can drive further into more details to find all the different types of compute service that has an installed package which is that one there. And all done through that very simple click interface and there I have my two different servers, each

with Jenkins running on them.”).



101. Claim 1 further recites “generating a query based on the received incident input and the mapped scenario” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s “Search with Discovery AI.” Upon information and belief, when a user clicks on a listed alert, vulnerability or query Orca generates a query based on the input of “Jenkins” and the provided list of alerts. *See, e.g., AI-Driven Cloud Security, <https://orca.security/platform/ai-cloud-security/>* (“I could search for Jenkins. Now, did I mean assets with the name Jenkins? Did I mean alerts that impact assets with the name Jenkins or perhaps vulnerabilities. Or maybe I meant the actual installed package. All of these things derived from that question and clicking into them I can drive further into more details to find all the different types of compute service that has an installed package which is that one there. And

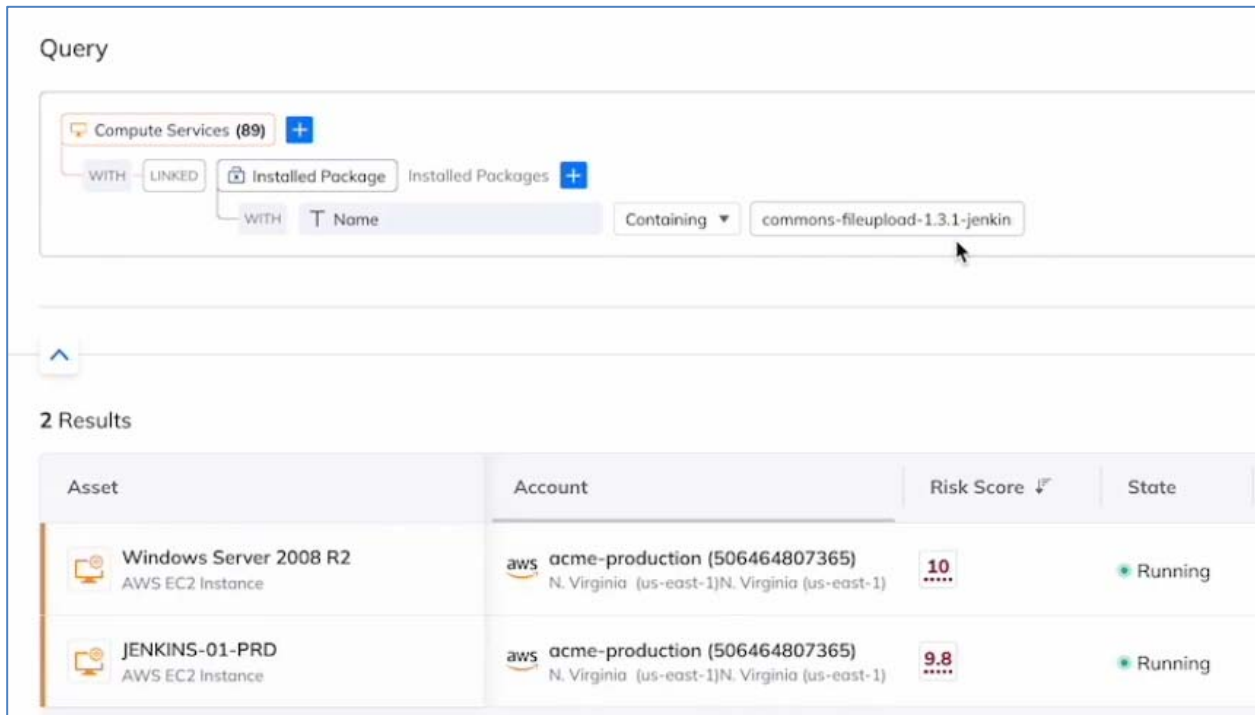
all done through that very simple click interface and there I have my two different servers, each with Jenkins running on them.”).

The screenshot shows a search interface for 'jenkins'. It includes sections for 'TOP ASSETS', 'TOP ALERTS', 'TOP VULNERABILITIES', and 'QUERIES'. The 'TOP ASSETS' section lists 'Jenkins-01-prd-vol-snapshot' (AWS EC2 EBS Snapshot) with a risk score of 5.4 and 'JENKINS-01-PRD' (AWS EC2 Instance) with a risk score of 9.8. The 'TOP ALERTS' section shows three alerts: 'Infected virtual machine with critical vulnerability containing potentially sensitive data' (8.8), 'Service Vulnerability' (7.4), and 'AWS Guard Duty: Ec2 with UnauthorizedAccess:EC2/SSHBruteForce' (7.1). The 'TOP VULNERABILITIES' section lists CVE-2009-1190, CVE-2009-5080, and CVE-2010-3192. The 'QUERIES' section lists four queries related to Jenkins war files, installed packages, and health advisor applications.

The screenshot shows a query builder interface. The query is: 'Compute Services (89) WITH LINKED Installed Package Installed Packages WITH T Name Containing commons-fileupload-1.3.1-jenkin'. Below the query builder, there are 2 Results in a table:

Asset	Account	Risk Score	State
Windows Server 2008 R2 AWS EC2 Instance	aws acme-production (506464807365) N. Virginia (us-east-1)N. Virginia (us-east-1)	10	Running
JENKINS-01-PRD AWS EC2 Instance	aws acme-production (506464807365) N. Virginia (us-east-1)N. Virginia (us-east-1)	9.8	Running

102. Claim 1 further recites “executing the query on a security database, the security database including a representation of a computing environment; and” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s “Search with Discovery AI.” Upon information and belief, when a user clicks on a listed alert, vulnerability or query Orca executes the query on a security database including its “Unified Data Model.” *See, e.g., AI-Driven Cloud Security*, <https://orca.security/platform/ai-cloud-security/>.



The screenshot shows the Orca security platform's query builder interface. The query is defined as follows:

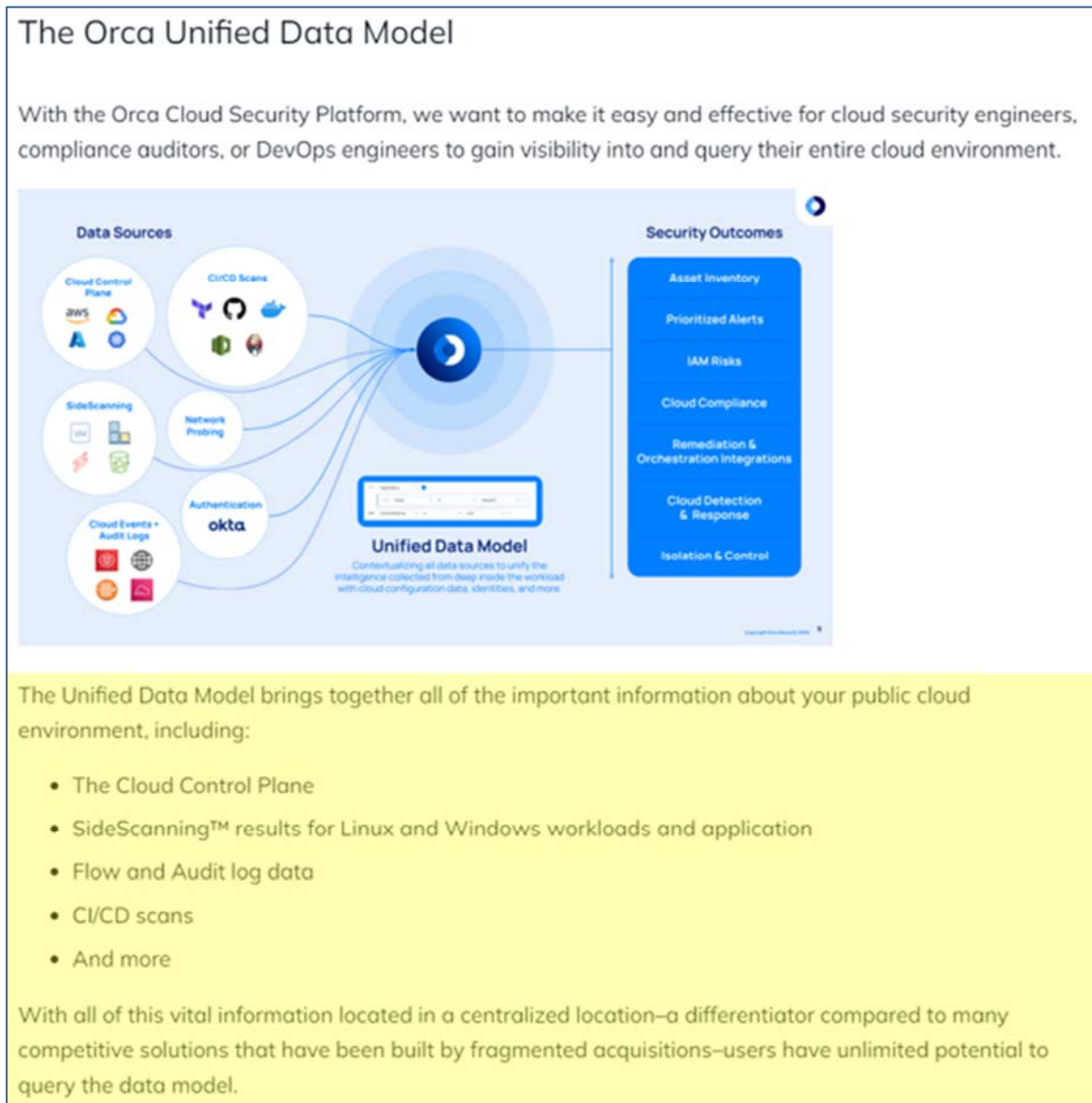
- Compute Services (89) WITH LINKED Installed Package Installed Packages WITH T Name Containing commons-fileupload-1.3.1-jenkin

The results table shows 2 results:

Asset	Account	Risk Score	State
Windows Server 2008 R2 AWS EC2 Instance	aws acme-production (506464807365) N. Virginia (us-east-1)N. Virginia (us-east-1)	10	Running
JENKINS-01-PRD AWS EC2 Instance	aws acme-production (506464807365) N. Virginia (us-east-1)N. Virginia (us-east-1)	9.8	Running

103. Orca also promotes its “Unified Data Model.” *See, e.g., Cloud Security Simplified: Easily Query Your Entire Cloud Environment*, <https://orca.security/resources/blog/orca-sonar-data-query-builder/>, (“The Unified Data Model brings together all of the important information about your public cloud environment”); and its “Data Security and Posture Management.” *See e.g., Data Security and Posture Management*, <https://orca.security/platform/data-security-and-posture-management-dspm/> (“The Orca Cloud Security Platform performs continuous discovery of data stores across your cloud estate, and

alerts to security and compliance risks, without requiring any additional tools. Instead of focusing solely on data security, Orca delivers a comprehensive, context-driven picture of sensitive data exposure, enabling organizations to prioritize risks effectively, reduce alert fatigue, and stay focused on what matters most—from a single platform.”).



104. Finally, claim 1 recites “initiating a mitigation action based on a result of the executed query.” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, allowing users to take remedial steps based on the query results. *See, e.g., AI-*

Driven Cloud Security, <https://orca.security/platform/ai-cloud-security/>.

Asset	Account	Risk Score	State	Alerts on Asset
Windows Server 2008 R2 AWS EC2 Instance	aws acme-production (506464807365) N. Virginia (us-east-1)N. Virginia (us-east-1)	10	Running	1 8 60 81 304
JENKINS-01-PRD AWS EC2 Instance	aws acme-production (506464807365) N. Virginia (us-east-1)N. Virginia (us-east-1)	9.8	Running	2 6 85 62 127

Further, Orca practices this step by, for example, using Orca’s AI Driven Cloud Security to generate policies or remediation steps. *See, e.g., AI-Driven Cloud Security,* <https://orca.security/platform/ai-cloud-security/>.

Recommended Remediation

Delete the infected file immediately, and audit your asset and network for any anomalies or other infected files. Patch the vulnerable service according to the CVE details.

Remediation

Remediation Console:

1. Remove the infected file immediately.
2. Audit your asset and network for any anomalies or other infected files.
3. In case you are not familiar with the file and further security information is needed, you may search the file hash in [VirusTotal](#).
4. In case the file is not known to the website, you may upload it via [VirusTotal files upload](#), and the file will get scanned by various security vendors. **Important:** Files that are uploaded to VirusTotal must not include proprietary information as these are publicly shared by the website.
5. Patch the vulnerable service according to the CVE details.

As a Further example, Orca’s IAM Policy Optimizer powered by Orca AI recommends remediation plans to users. *See id.*



105. Upon information and belief, Orca further infringes multiple additional claims of the '549 patent. For example, but not limited to, claims 2, 3.

106. Orca is aware of the '549 patent and Orca's infringement thereof at least as of the filing of these counterclaims. Moreover, this is another in a long line of examples of Orca copying Wiz. Accordingly, Orca has and continues to willfully infringement the '549 patent.

107. Orca has induced and continues to induce infringement of one or more claims of the '549 patent by encouraging customers to use AI-Driven Cloud Security in a manner that directly infringes those claims. Despite its knowledge of the existence of the '549 patent, since at least the filing of this Counterclaim, Orca, upon information and belief, continues to encourage, instruct, enable and otherwise cause its customers to use AI-Driven Cloud Security in a manner that infringes one or more claims of the '549 patent. Upon information and belief, Orca specifically intends that its customers use AI-Driven Cloud Security in a manner that infringes one or more claims of the '549 patent by, at a minimum, providing instructions and/or support documentation directing customers on how to use AI-Driven Cloud Security in an infringing manner, in violation of 35 U.S.C. § 271(b). For example, Orca's public web posts cited above provide instructions and encourage customers to practice all steps of the claimed method. Further, Orca provides video explanation of how to use Orca's AI-Driven Cloud Security. See <https://orca.security/platform/ai-cloud-security>.

108. Orca has contributed and continues to contribute to the infringement of one or more claims of the '549 patent. Upon information and belief, Orca knows that its AI-Driven Cloud Security features are especially made and/or adapted for users to infringe one or more claims of the '549 patent and is not a staple article or commodity of commerce suitable for substantial non-infringing use. Because Orca included features, such as for example but not limited to, AI-Driven Cloud Security in Orca's products, Orca intends for customers to use it. Upon information and belief, AI-Driven Cloud Security has no suitable use that is non-

infringing, and therefore Orca intends for customers to use AI-Driven Cloud Security in an infringing manner. Orca's sales of products including AI-Driven Cloud Security constitute contributory infringement in violation of 35 U.S.C. § 271(c).

COUNTERCLAIM V
(INFRINGEMENT OF U.S. PATENT NO. 12,003,529)

109. Wiz is the sole and exclusive owner, by assignment, of all rights, title and interest in U.S. Patent No. 12,003,549 (the "'529 patent"), entitled "Techniques for Detecting Artificial Intelligence Model Cybersecurity Risk in a Computing Environment." The '529 patent was duly and legally issued by the U.S. Patent and Trademark Office on June 4, 2024. The named inventors of the '529 patent are Amitai Cohen, Barak Sharoni, Shir Tamari, George Pisha, Itay Arbel, Daniel Velikanski, and Yaniv Shaked. A copy of the '529 patent is attached as Exhibit E.

110. The '529 patent generally relates to detecting an artificial intelligence model cybersecurity risk in a computing environment. *See* '529 patent at 1:61-64. This includes "generating a representation of the AI model in a security database[.]" *Id.* at 1:66-67. The patent further includes "initiating a mitigation action based on the cybersecurity risk." *Id.* at 2:11-12.

111. The '529 patent discloses "generating an inspectable disk based on an original disk of a resource deployed in the computing environment; and inspecting the inspectable disk for the AI model." *Id.* at 2:18-21. The patent discloses that the inspection may include "detecting an artifact of the AI model[.]" detecting "an AI model configured to execute a code object[.]" detecting "metadata of the AI model, where the metadata indicates that the AI model is a cybersecurity risk[.]" and "detecting in the AI model any one of: a secret, a certificate, a code, and any combination thereof." *Id.* at 2:21-29. "In some embodiments, a mitigation action includes revoking access from a principal, revoking access to a resource, revoking access from a resource, sandboxing a resource, revoking access to an AI model, revoking access from an AI

model, denying network communication directed to the AI model, such as network communication including a prompt for the AI model, generating an alert, updating an alert, generating an alert severity, updating an alert severity, various combinations thereof, and the like.” *Id.* at 17:24-32.

112. The ’529 patent further discloses that in response to detecting a cybersecurity risk of an AI the system “my furthermore initiate a mitigation action based on the cybersecurity risk.” *Id.* at 3:11-12. The mitigation action may be “based on a toxic combination, the detected cybersecurity object, an AI model type, a combination thereof, and the like.” *Id.* at 17:20-23.

113. Orca has infringed and continues to directly infringe one or more claims of the ’529 patent by making, using, selling, offering for sale, and/or importing into the United States without authority or license, the Orca Platform with AI Security Posture Management (AI-SPM) in violation of 35 U.S.C. § 271(a). Orca’s infringement includes infringement of, for example, claim 1 of the ’529 patent.

114. Claim 1 of the ’529 patent recites:

1. A method for detecting a cybersecurity risk of an artificial intelligence (AI), comprising:
generating an inspectable disk based on an original disk of a resource deployed in a computing environment;
inspecting the inspectable disk for an AI model;
generating a representation of the AI model in a security database, the security database including a representation of the computing environment;
inspecting the AI model for a cybersecurity risk;
generating a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and
initiating a mitigation action based on the cybersecurity risk.

115. On information and belief, Orca practices each and every limitation of claim 1 of the ’529 patent by and through the use of Orca’s AI Security Posture Management (“AI-SPM”).

116. The preamble of claim 1 recites “[a] method for detecting a cybersecurity risk of an artificial intelligence (AI), comprising:” To the extent the preamble is limiting, Orca practices this step by, for example but not limited to, using its AI Security Posture Management features to detect a cybersecurity risk of an artificial intelligence. *See, e.g., Orca Adds AI Security to Cloud Security Platform*, <https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/> (“Today we are pleased to announce that the Orca Cloud Security Platform now offers complete end-to-end AI Security Posture Management (AI-SPM) capabilities, so Orca customers can continue to leverage AI at unhindered speed, but do so safely.”).

Organizations are increasingly leveraging Generative AI and Large Language Models (LLMs) to optimize business processes and improve products and services. A [recent Gartner report](#) predicts that global spending on AI software will grow to \$298 billion by 2027, with a compound annual growth rate (CAGR) of 19.1%. Some may even be of the opinion that these figures are conservative.

From assets scanned by the Orca Cloud Security Platform, we found that over 37% of organizations have already adopted at least one AI service, with the highest use being Amazon SageMaker and Bedrock (68%) followed by Azure OpenAI (50%), and Vertex AI comes in third at 21%.

Along with all the great business benefits and improved products and services that AI brings, there is however one catch: security. In our [2024 State of Cloud Security Report](#), we found that 82% of Amazon SageMaker users have exposed notebooks, meaning that they are publicly accessible. Especially since AI models often include sensitive data and intellectual property in their training data, these cloud resources are at even higher potential risk than other resources. This has given rise to a new type of security need: AI Security.

While the exposure of SageMaker notebooks is a significant concern, it is a condition that falls within the user’s responsibility under the shared responsibility model. Therefore, it’s important that organizations follow best practices and use the right AI security tools so they can easily identify and remediate these exposures, ensuring the secure deployment of their AI and ML workloads.

Today we are pleased to announce that the [Orca Cloud Security Platform](#) now offers complete end-to-end AI Security Posture Management (AI-SPM) capabilities, so Orca customers can continue to leverage AI at unhindered speed, but do so safely. By offering AI security from Orca’s comprehensive platform, organizations avoid having to add yet another point security solution specific to AI security to their arsenal, reducing overhead and integrating into existing workflows.

117. Claim 1 further recites “generating an inspectable disk based on an original disk of a resource deployed in a computing environment” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s SideScanning Technology “to cover AI models” in combination with Orca’s AI-SPM. *See, e.g., Orca Adds AI Security to Cloud Security Platform*, <https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/> (“Leveraging Orca’s patented agentless SideScanning technology, we’ve extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources . . . Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM).”).

Why is Orca adding AI-SPM?

Many of the security risks facing AI models and LLMs are similar to other cloud assets, including limited visibility, accidental public access, unencrypted sensitive data, shadow data, and unsecured keys. Leveraging Orca's patented agentless [SideScanning technology](#), we've extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources. In addition, we're applying our existing technology for use cases that are unique to AI security, such as detecting sensitive data in training sets, that could later be unintentionally exposed by the LLM or Generative AI application.

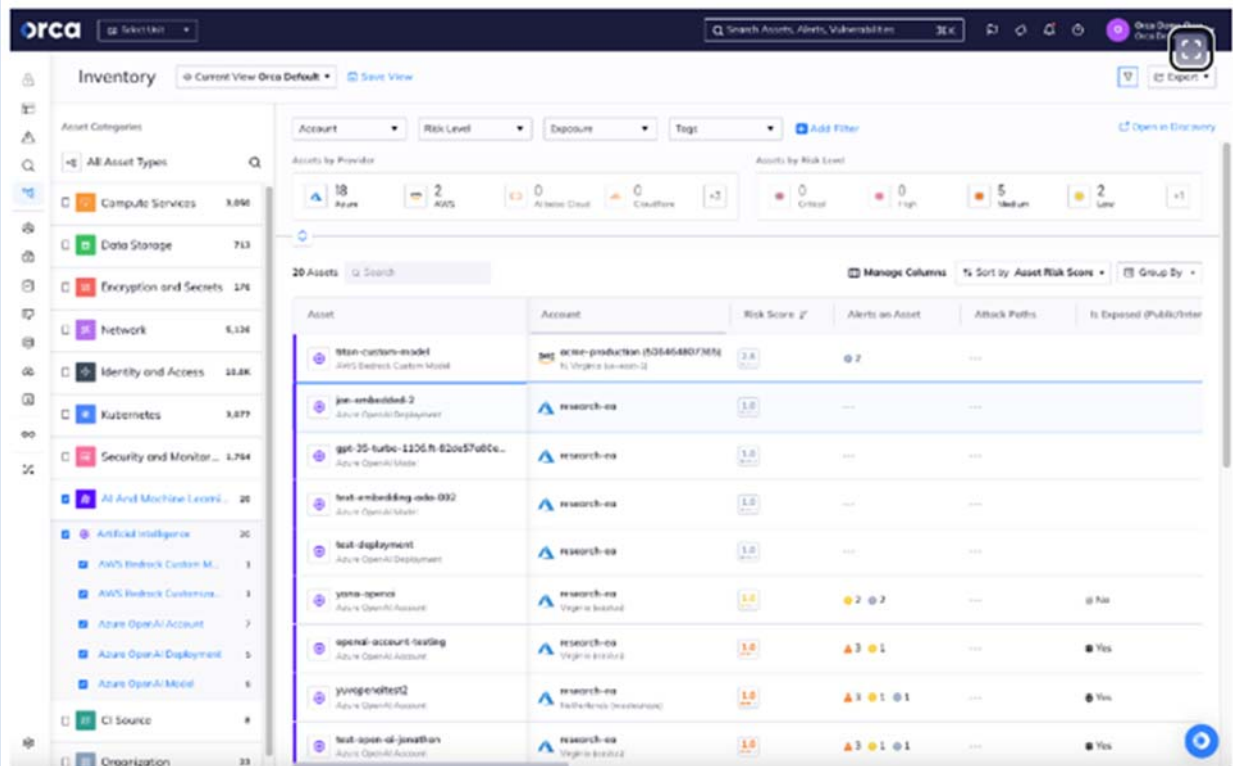
Since the Orca platform does not require agents to inspect cloud resources, coverage is always continuous and 100%, and will immediately scan any new AI resources as soon as they are deployed and alert to any detected risks. However Orca goes beyond just covering AI models – the platform also maps out all the AI related packages that are used to train or infer AI.

By adding AI security to our comprehensive cloud security platform, organizations don't need to procure, deploy, or learn how to use another separate point solution but can instead leverage one unified platform for all cloud security needs.

#1. AI and ML BOM + inventory

Challenge: Much like other resources in the cloud, shadow AI and LLMs are a major concern. In their excitement to explore all the opportunities that Generative AI brings, developers are not always waiting for IT approval before integrating Gen AI services into their flows. And security is probably not the first thing on their mind. This leaves security teams in the dark about which AI projects have been deployed in their environment, whether they contain sensitive data, and whether they are secure.

Orca solution: Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM). Orca detects projects on Azure OpenAI, Amazon Bedrock, Google Vertex AI, AWS Sagemaker and those using one of 50+ most commonly used AI software packages, including Pytorch, TensorFlow, OpenAI, Hugging Face, scikit-learn, and many more.



The Orca Platform shows a full inventory and bill of materials of all deployed AI models

118. Claim 1 further recites “inspecting the inspectable disk for an AI model”

Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s SideScanning Technology “to cover AI models” in combination with Orca’s AI-SPM. See, e.g., *Orca Adds AI Security to Cloud Security Platform*,

<https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/>

(“Leveraging Orca’s patented agentless SideScanning technology, we’ve extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources . . . Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM).”).

Why is Orca adding AI-SPM?

Many of the security risks facing AI models and LLMs are similar to other cloud assets, including limited visibility, accidental public access, unencrypted sensitive data, shadow data, and unsecured keys. Leveraging Orca's patented agentless [SideScanning technology](#), we've extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources. In addition, we're applying our existing technology for use cases that are unique to AI security, such as detecting sensitive data in training sets, that could later be unintentionally exposed by the LLM or Generative AI application.

Since the Orca platform does not require agents to inspect cloud resources, coverage is always continuous and 100%, and will immediately scan any new AI resources as soon as they are deployed and alert to any detected risks. However Orca goes beyond just covering AI models – the platform also maps out all the AI related packages that are used to train or infer AI.

By adding AI security to our comprehensive cloud security platform, organizations don't need to procure, deploy, or learn how to use another separate point solution but can instead leverage one unified platform for all cloud security needs.

#1. AI and ML BOM + inventory

Challenge: Much like other resources in the cloud, shadow AI and LLMs are a major concern. In their excitement to explore all the opportunities that Generative AI brings, developers are not always waiting for IT approval before integrating Gen AI services into their flows. And security is probably not the first thing on their mind. This leaves security teams in the dark about which AI projects have been deployed in their environment, whether they contain sensitive data, and whether they are secure.

Orca solution: Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM). Orca detects projects on Azure OpenAI, Amazon Bedrock, Google Vertex AI, AWS Sagemaker and those using one of 50+ most commonly used AI software packages, including Pytorch, TensorFlow, OpenAI, Hugging Face, scikit-learn, and many more.

The screenshot displays the Orca Platform's 'Inventory' page. On the left, there is a sidebar with 'Asset Categories' including Compute Services (3,896), Data Storage (713), Encryption and Secrets (176), Network (6,134), Identity and Access (11,828), Kubernetes (3,877), Security and Monitor... (1,784), AI And Machine Learn... (26), Artificial Intelligence (26), AWS Bedrock Custom M... (1), AWS Bedrock Custom... (1), Azure OpenAI/Account (7), Azure OpenAI/Deployment (5), Azure OpenAI/Model (6), CI Source (8), and Organization (33). The main area shows a table of 20 assets. The table has columns for Asset, Account, Risk Score, Alerts on Asset, Attack Paths, and Is Exposed (Public/Inter). The assets listed include 'non-custom-model', 'jira-embedded-2', 'gpt-35-turbo-1106-h-82ov57ubCe...', 'text-embedding-ada-002', 'text-deployment', 'yama-igwca', 'openai-account-testing', 'yuvopdites2', and 'text-openai-jasonth'. Each row shows the asset name, account (e.g., 'research-ua'), risk score (e.g., 2.8, 1.0), and exposure status (e.g., 'All', 'Yes').

The Orca Platform shows a full inventory and bill of materials of all deployed AI models

119. Claim 1 further recites “generating a representation of the AI model in a security database, the security database including a representation of the computing environment” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s SideScanning Technology “to cover AI models” in combination with Orca’s AI-

SPM. See, e.g., *Orca Adds AI Security to Cloud Security Platform*,

<https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/>

(“Leveraging Orca’s patented agentless SideScanning technology, we’ve extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources... Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM).”).

Why is Orca adding AI-SPM?

Many of the security risks facing AI models and LLMs are similar to other cloud assets, including limited visibility, accidental public access, unencrypted sensitive data, shadow data, and unsecured keys. Leveraging Orca’s patented agentless [SideScanning technology](#), we’ve extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources. In addition, we’re applying our existing technology for use cases that are unique to AI security, such as detecting sensitive data in training sets, that could later be unintentionally exposed by the LLM or Generative AI application.

Since the Orca platform does not require agents to inspect cloud resources, coverage is always continuous and 100%, and will immediately scan any new AI resources as soon as they are deployed and alert to any detected risks. However Orca goes beyond just covering AI models – the platform also maps out all the AI related packages that are used to train or infer AI.

By adding AI security to our comprehensive cloud security platform, organizations don’t need to procure, deploy, or learn how to use another separate point solution but can instead leverage one unified platform for all cloud security needs.

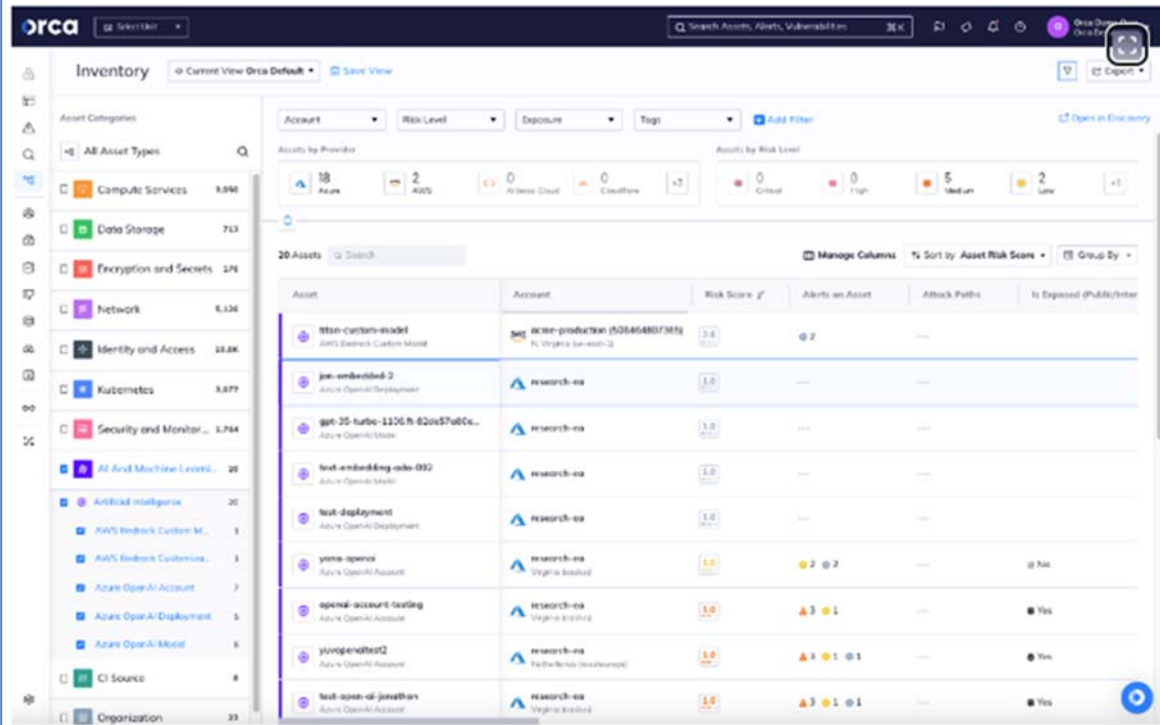
What are Orca's AI Security capabilities?

Orca includes a new AI Security dashboard that provides an overview of the AI models that are deployed in the cloud environment, what data they contain, and whether they are at risk. Orca covers risks end-to-end, from training and fine-tuning to production deployment and inference. Orca connects with your AI data and processes on all levels – the managed cloud AI services, the unmanaged AI models and packages that your developers are using, and even with shift-left detection of leaky secret tokens AI services in your codebase.

#1. AI and ML BOM + inventory

Challenge: Much like other resources in the cloud, shadow AI and LLMs are a major concern. In their excitement to explore all the opportunities that Generative AI brings, developers are not always waiting for IT approval before integrating Gen AI services into their flows. And security is probably not the first thing on their mind. This leaves security teams in the dark about which AI projects have been deployed in their environment, whether they contain sensitive data, and whether they are secure.

Orca solution: Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM). Orca detects projects on Azure OpenAI, Amazon Bedrock, Google Vertex AI, AWS Sagemaker and those using one of 50+ most commonly used AI software packages, including Pytorch, TensorFlow, OpenAI, Hugging Face, scikit-learn, and many more.

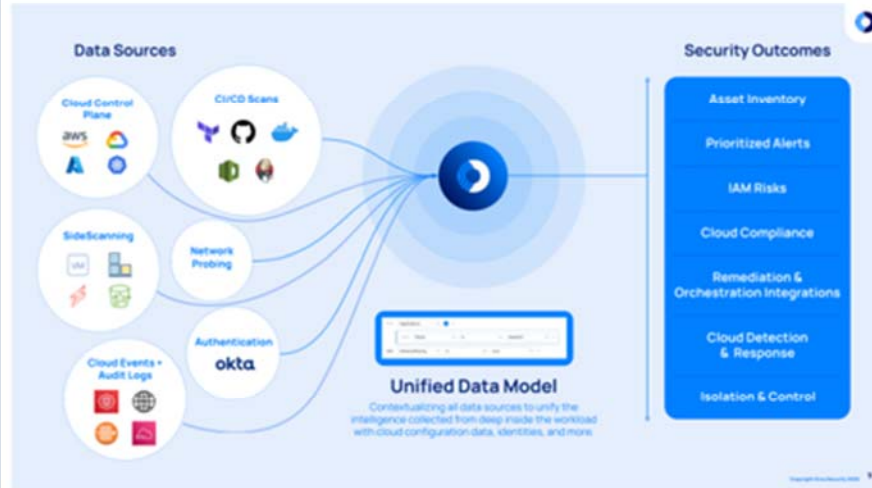


The Orca Platform shows a full inventory and bill of materials of all deployed AI models

120. Further, Orca practices this step by generating a representation of the AI model in its “Unified Data Model.” *See, e.g., Cloud Security Simplified: Easily Query Your Entire Cloud Environment*, <https://orca.security/resources/blog/orca-sonar-data-query-builder/> (“The Unified Data Model brings together all of the important information about your public cloud environment . . .”); and through the use of its “Data Security and Posture Management.” *See e.g., Data Security and Posture Management*, <https://orca.security/platform/data-security-and-posture-management-dspm/> (“The Orca Cloud Security Platform performs continuous discovery of data stores across your cloud estate, and alerts to security and compliance risks, without requiring any additional tools. Instead of focusing solely on data security, Orca delivers a comprehensive, context-driven picture of sensitive data exposure, enabling organizations to prioritize risks effectively, reduce alert fatigue, and stay focused on what matters most—from a single platform.”).

The Orca Unified Data Model

With the Orca Cloud Security Platform, we want to make it easy and effective for cloud security engineers, compliance auditors, or DevOps engineers to gain visibility into and query their entire cloud environment.



The Unified Data Model brings together all of the important information about your public cloud environment, including:

- The Cloud Control Plane
- SideScanning™ results for Linux and Windows workloads and application
- Flow and Audit log data
- CI/CD scans
- And more

With all of this vital information located in a centralized location—a differentiator compared to many competitive solutions that have been built by fragmented acquisitions—users have unlimited potential to query the data model.

OUR APPROACH

Detect and Prioritize Cloud Data Security Risks with Context

The Orca Cloud Security Platform performs continuous discovery of data stores across your cloud estate, and alerts to security and compliance risks, without requiring any additional tools. Instead of focusing solely on data security, Orca delivers a comprehensive, context-driven picture of sensitive data exposure, enabling organizations to prioritize risks effectively, reduce alert fatigue, and stay focused on what matters most—from a single platform.

121. Claim 1 further recites “inspecting the AI model for a cybersecurity risk” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s SideScanning Technology “to cover AI models” in combination with Orca’s AI-SPM. *See, e.g., Orca Adds AI Security to Cloud Security Platform*, <https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/> (“Leveraging Orca’s patented agentless SideScanning technology, we’ve extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources . . . Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM) . . . Since it could be very damaging if this information got reached, it is very important to ensure that AI models are not publicly exposed. However, misconfigurations do happen and especially with Shadow AI, security may not be the first thing developers have on their mind.”).

Why is Orca adding AI-SPM?

Many of the security risks facing AI models and LLMs are similar to other cloud assets, including limited visibility, accidental public access, unencrypted sensitive data, shadow data, and unsecured keys. Leveraging Orca's patented agentless [SideScanning technology](#), we've extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources. In addition, we're applying our existing technology for use cases that are unique to AI security, such as detecting sensitive data in training sets, that could later be unintentionally exposed by the LLM or Generative AI application.

Since the Orca platform does not require agents to inspect cloud resources, coverage is always continuous and 100%, and will immediately scan any new AI resources as soon as they are deployed and alert to any detected risks. However Orca goes beyond just covering AI models – the platform also maps out all the AI related packages that are used to train or infer AI.

By adding AI security to our comprehensive cloud security platform, organizations don't need to procure, deploy, or learn how to use another separate point solution but can instead leverage one unified platform for all cloud security needs.

What are Orca's AI Security capabilities?

Orca includes a new AI Security dashboard that provides an overview of the AI models that are deployed in the cloud environment, what data they contain, and whether they are at risk. Orca covers risks end-to-end, from training and fine-tuning to production deployment and inference. Orca connects with your AI data and processes on all levels – the managed cloud AI services, the unmanaged AI models and packages that your developers are using, and even with shift-left detection of leaky secret tokens AI services in your codebase.

#1. AI and ML BOM + inventory

Challenge: Much like other resources in the cloud, shadow AI and LLMs are a major concern. In their excitement to explore all the opportunities that Generative AI brings, developers are not always waiting for IT approval before integrating Gen AI services into their flows. And security is probably not the first thing on their mind. This leaves security teams in the dark about which AI projects have been deployed in their environment, whether they contain sensitive data, and whether they are secure.

Orca solution: Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM). Orca detects projects on Azure OpenAI, Amazon Bedrock, Google Vertex AI, AWS Sagemaker and those using one of 50+ most commonly used AI software packages, including Pytorch, TensorFlow, OpenAI, Hugging Face, scikit-learn, and many more.

Asset	Account	Risk Score	Alerts on Asset	Attack Paths	Is Exposed (Public/Inter)	
1111n-custom-model	aws-production (520464807356)	2.8	0	---	---	
jira-embedded-2	research-va	1.0	---	---	---	
gpt-35-turbo-1106	research-va	1.0	---	---	---	
test-embedding-ada-002	research-va	1.0	---	---	---	
test-deployment	research-va	1.0	---	---	---	
yama-openai	research-va	1.0	2	---	No	
openai-account-testing	research-va	1.0	3	1	Yes	
yivogentest2	research-va	1.0	4	1	1	Yes
test-asset-of-jonathan	research-va	1.0	3	1	1	Yes

The Orca Platform shows a full inventory and bill of materials of all deployed AI models

#2. Warn when AI projects are publicly accessible

Challenge: Data used to train AI models is often sensitive and can contain proprietary information. Since it could be very damaging if this information got breached, it's very important to ensure that AI models are not publicly exposed. However, misconfigurations do happen, and especially with Shadow AI, security may not be the first thing developers have on their mind. This leaves security teams struggling to ensure that all AI models are being kept private.

Orca solution: Since Orca has insight into the AI model settings and network access, Orca will alert whenever public access is allowed, so security teams can quickly fix the issue to prevent any data breaches.

The screenshot displays the Orca AI Security dashboard. At the top, it shows the 'Average Score' as 55% (including only scored tests) and 'Control Tests' as 11/20 (8 failed, 7 passed, 11 unscored). There are 27 Accounts and 2 failed tests. The dashboard is organized into sections: 1 Network Security (43% score) and 2 Data Protection (56% score). A table lists specific control test results:

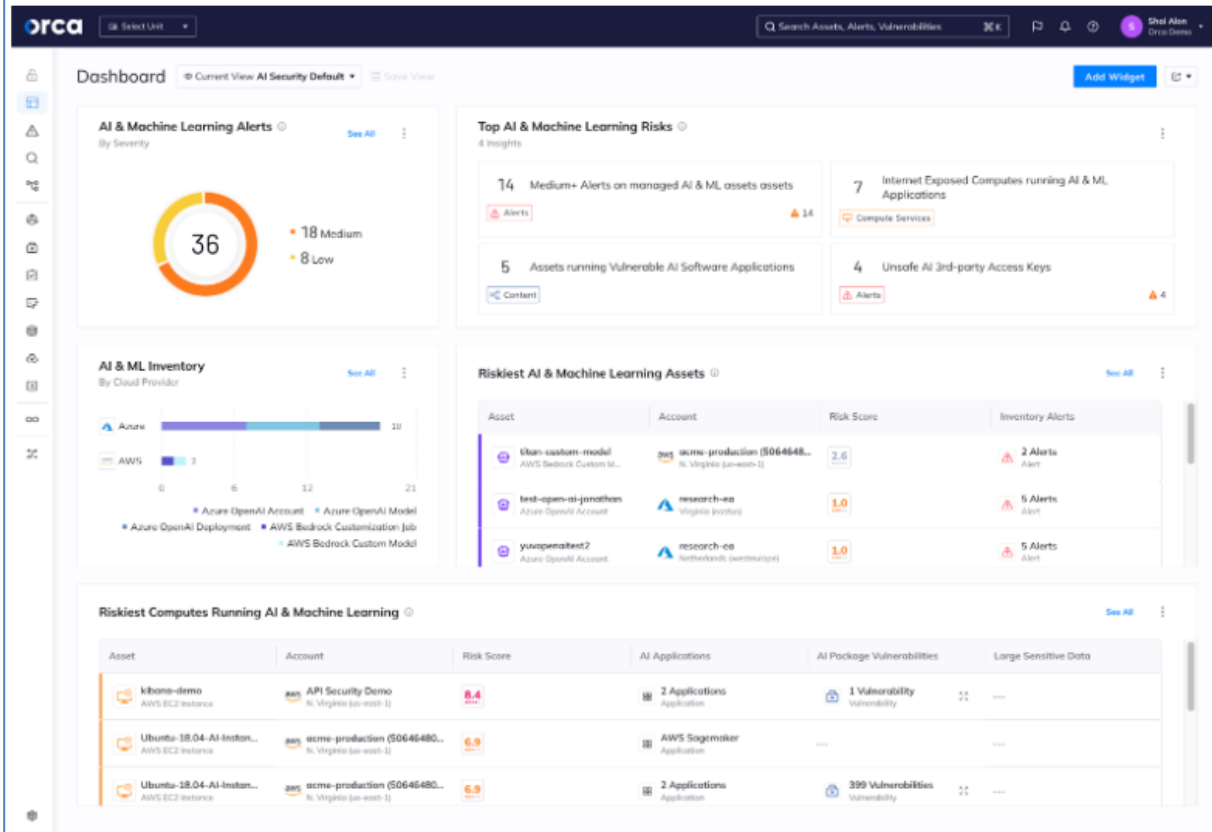
Control ID	Type	Priority	Description	Vendor	Open Alerts
1.2.1	system	low	Azure OpenAI account has no firewall exclusion configured for bypass Azure trusted services	Azure	1
1.2.2	system	low	AWS SageMaker notebook instance is configured with an outdated minimum Instance Metadata Service Version (IMDS)	AWS	0

Orca AI Security best practices compliance performs AI-Security Posture Management

#3. Detect sensitive data in AI projects

Challenge: AI models use vast amounts of data to train models. It's possible that this data inadvertently contains sensitive data, and that developers and security teams are not even aware of the presence of the data, such as PII and access keys. LLMs and Generative AI models could then “spill out” this sensitive data, which could be used by bad actors. In addition, if developers are not aware that the training set contains sensitive data, they may not prioritize security of the resource as much as it needs to be.

Orca solution: Using Orca’s [Data Security Posture Management \(DSPM\)](#) capabilities, Orca scans and classifies all the data stored in AI projects, and alerts if they contain sensitive data, such as email addresses, telephone numbers, email addresses, and social security numbers (PII), or personal health information (PHI). By informing security teams where sensitive data is located, they can make sure that these assets are protected with the highest level of security.



Orca displays pertinent security information in the AI Security dashboard

122. Claim 1 further recites “generating a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and” Orca’s public web posts

confirm that Orca practices this step by, for example but not limited to, using Orca's SideScanning Technology "to cover AI models" in combination with Orca's AI-SPM. Orca further displays a representation of the cybersecurity risk on the AI Security Dashboard including alerting on publicly accessible AI models as well as unsecure AI training data containing sensitive data such as personal addresses or social security information. *See, e.g., Orca Adds AI Security to Cloud Security Platform*, <https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/> ("Leveraging Orca's patented agentless SideScanning technology, we've extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources . . . Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM) . . . Since Orca has insight into the AI model settings and network access, Orca will alert whenever public access is allowed, so security teams can quickly fix the issue to prevent any data breaches . . . Orca scans and classifies all the data stored in AI projects, and alerts if they contain sensitive data, such as email addresses, telephone numbers, email addresses, and social security numbers (PII), or personal health information (PHI).").

Why is Orca adding AI-SPM?

Many of the security risks facing AI models and LLMs are similar to other cloud assets, including limited visibility, accidental public access, unencrypted sensitive data, shadow data, and unsecured keys. Leveraging Orca's patented agentless [SideScanning technology](#), we've extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources. In addition, we're applying our existing technology for use cases that are unique to AI security, such as detecting sensitive data in training sets, that could later be unintentionally exposed by the LLM or Generative AI application.

Since the Orca platform does not require agents to inspect cloud resources, coverage is always continuous and 100%, and will immediately scan any new AI resources as soon as they are deployed and alert to any detected risks. However Orca goes beyond just covering AI models – the platform also maps out all the AI related packages that are used to train or infer AI.

By adding AI security to our comprehensive cloud security platform, organizations don't need to procure, deploy, or learn how to use another separate point solution but can instead leverage one unified platform for all cloud security needs.

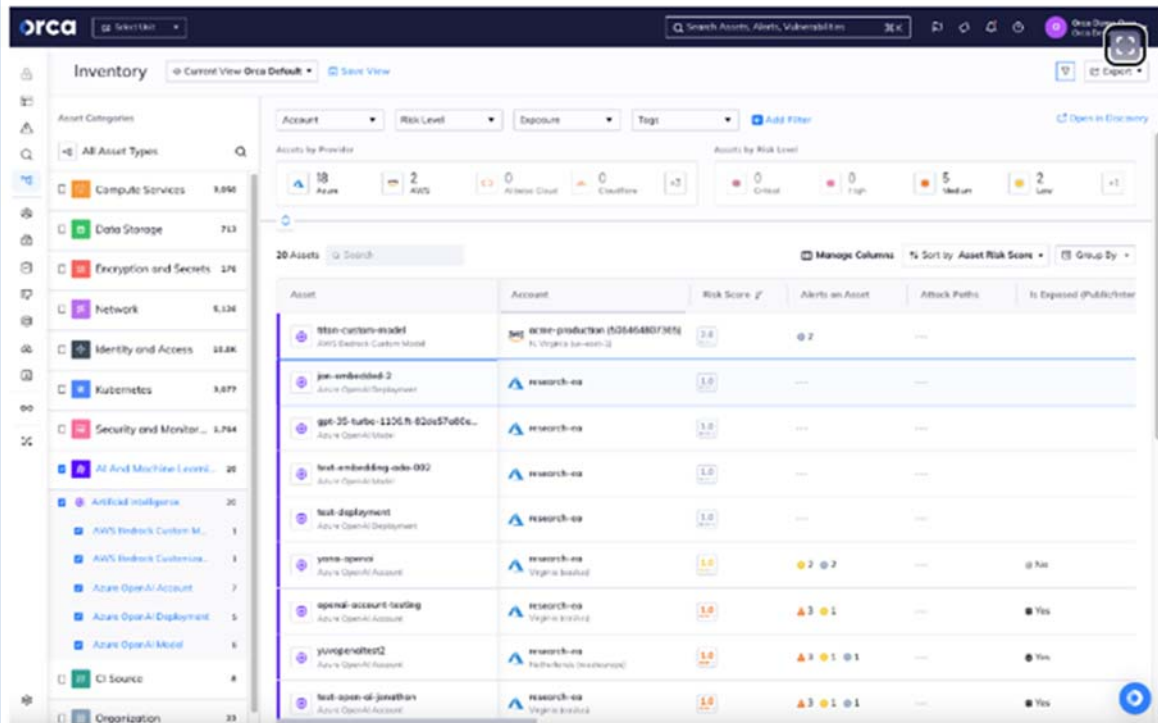
What are Orca's AI Security capabilities?

Orca includes a new AI Security dashboard that provides an overview of the AI models that are deployed in the cloud environment, what data they contain, and whether they are at risk. Orca covers risks end-to-end, from training and fine-tuning to production deployment and inference. Orca connects with your AI data and processes on all levels – the managed cloud AI services, the unmanaged AI models and packages that your developers are using, and even with shift-left detection of leaky secret tokens AI services in your codebase.

#1. AI and ML BOM + inventory

Challenge: Much like other resources in the cloud, shadow AI and LLMs are a major concern. In their excitement to explore all the opportunities that Generative AI brings, developers are not always waiting for IT approval before integrating Gen AI services into their flows. And security is probably not the first thing on their mind. This leaves security teams in the dark about which AI projects have been deployed in their environment, whether they contain sensitive data, and whether they are secure.

Orca solution: Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM). Orca detects projects on Azure OpenAI, Amazon Bedrock, Google Vertex AI, AWS Sagemaker and those using one of 50+ most commonly used AI software packages, including Pytorch, TensorFlow, OpenAI, Hugging Face, scikit-learn, and many more.



The Orca Platform shows a full inventory and bill of materials of all deployed AI models

#2. Warn when AI projects are publicly accessible

Challenge: Data used to train AI models is often sensitive and can contain proprietary information. Since it could be very damaging if this information got breached, it's very important to ensure that AI models are not publicly exposed. However, misconfigurations do happen, and especially with Shadow AI, security may not be the first thing developers have on their mind. This leaves security teams struggling to ensure that all AI models are being kept private.

Orca solution: Since Orca has insight into the AI model settings and network access, Orca will alert whenever public access is allowed, so security teams can quickly fix the issue to prevent any data breaches.

The screenshot displays the Orca AI Security compliance dashboard. At the top, it shows the 'Average Score' as 55% (including only scored tests) and 'Control Tests' as 11/20 (8 failed, 7 passed, 11 unscored). There are 27 accounts in total, with 'acme-production' (50 tests, 2/20 failed) and 'research-ea' (7/20 failed) highlighted. The dashboard is organized into sections: '1 Network Security' (43% score) and '2 Data Protection' (56% score). Under Network Security, '1.1 Network Segmentation' has a 40% score (2/5) and '1.2 Network Access' has a 50% score (1/2). A table lists specific control failures:

Control ID	Type	Priority	Description	Vendor	Open Alerts
1.2.1	system	low	Azure OpenAI account has no firewall exclusion configured for bypass Azure trusted services	Azure	1
1.2.2	system	low	AWS SageMaker notebook instance is configured with an outdated minimum Instance Metadata Service Version (IMDS)	AWS	0

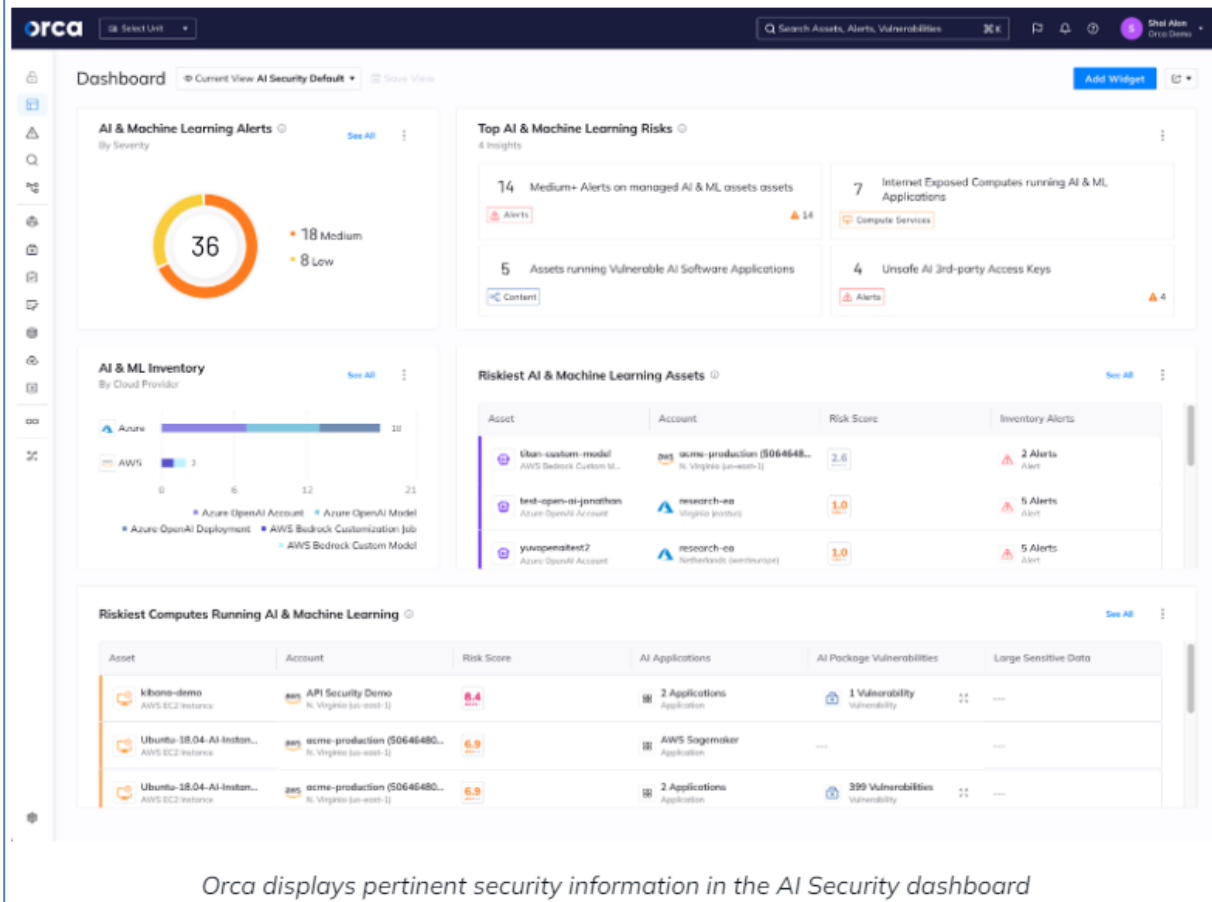
Under Data Protection, '2.1 Data at Rest Encryption' has a 67% score (4/6) and '2.2 Data in Transit Encryption' has a score of 0% (0/1). The interface includes filters for providers (Alibaba Cloud, Amazon Web Services, Microsoft Azure, Cloudflare, Google Cloud, Oracle, Shift Left) and accounts (Alibaba Cloud, AWS, Azure, Cloudflare, GCP).

Orca AI Security best practices compliance performs AI-Security Posture Management

#3. Detect sensitive data in AI projects

Challenge: AI models use vast amounts of data to train models. It's possible that this data inadvertently contains sensitive data, and that developers and security teams are not even aware of the presence of the data, such as PII and access keys. LLMs and Generative AI models could then “spill out” this sensitive data, which could be used by bad actors. In addition, if developers are not aware that the training set contains sensitive data, they may not prioritize security of the resource as much as it needs to be.

Orca solution: Using Orca’s [Data Security Posture Management \(DSPM\)](#) capabilities, Orca scans and classifies all the data stored in AI projects, and alerts if they contain sensitive data, such as email addresses, telephone numbers, email addresses, and social security numbers (PII), or personal health information (PHI). By informing security teams where sensitive data is located, they can make sure that these assets are protected with the highest level of security.

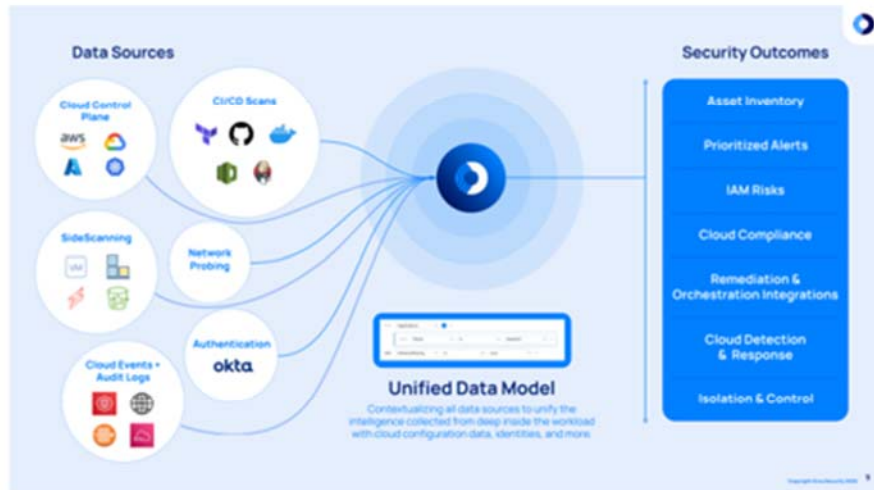


123. Further, Orca practices this limitation through the use of its “Unified Data Model,” *see, e.g., Cloud Security Simplified: Easily Query Your Entire Cloud Environment*, <https://orca.security/resources/blog/orca-sonar-data-query-builder/> (“The Unified Data Model

brings together all of the important information about your public cloud environment”); and its “Data Security and Posture Management.” See e.g., *Data Security and Posture Management*, <https://orca.security/platform/data-security-and-posture-management-dspm/> (“The Orca Cloud Security Platform performs continuous discovery of data stores across your cloud estate, and alerts to security and compliance risks, without requiring any additional tools. Instead of focusing solely on data security, Orca delivers a comprehensive, context-driven picture of sensitive data exposure, enabling organizations to prioritize risks effectively, reduce alert fatigue, and stay focused on what matters most—from a single platform.”).

The Orca Unified Data Model

With the Orca Cloud Security Platform, we want to make it easy and effective for cloud security engineers, compliance auditors, or DevOps engineers to gain visibility into and query their entire cloud environment.



The Unified Data Model brings together all of the important information about your public cloud environment, including:

- The Cloud Control Plane
- SideScanning™ results for Linux and Windows workloads and application
- Flow and Audit log data
- CI/CD scans
- And more

With all of this vital information located in a centralized location—a differentiator compared to many competitive solutions that have been built by fragmented acquisitions—users have unlimited potential to query the data model.

OUR APPROACH

Detect and Prioritize Cloud Data Security Risks with Context

The Orca Cloud Security Platform performs continuous discovery of data stores across your cloud estate, and alerts to security and compliance risks, without requiring any additional tools. Instead of focusing solely on data security, Orca delivers a comprehensive, context-driven picture of sensitive data exposure, enabling organizations to prioritize risks effectively, reduce alert fatigue, and stay focused on what matters most—from a single platform.

124. Finally, claim 1 recites “initiating a mitigation action based on the cybersecurity risk.” Orca’s public web posts confirm that Orca practices this step by, for example but not limited to, using Orca’s AI-SPM to generate alerts. Orca further displays alerts on the AI Security Dashboard. *See, e.g., Orca Adds AI Security to Cloud Security Platform*, <https://orca.security/resources/blog/orca-adds-ai-security-to-cloud-security-platform/> (“Leveraging Orca’s patented agentless SideScanning technology, we’ve extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources . . . Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM) . . . Since it could be very damaging if this information got reached, it is very important to ensure that AI models are not publicly exposed. However, misconfigurations do happen and especially with Shadow AI, security may not be the first thing developers have on their mind. ”).

Why is Orca adding AI-SPM?

Many of the security risks facing AI models and LLMs are similar to other cloud assets, including limited visibility, accidental public access, unencrypted sensitive data, shadow data, and unsecured keys. Leveraging Orca's patented agentless [SideScanning technology](#), we've extended our platform to also cover AI models, providing the same risk insights and deep data that we provide on other cloud resources. In addition, we're applying our existing technology for use cases that are unique to AI security, such as detecting sensitive data in training sets, that could later be unintentionally exposed by the LLM or Generative AI application.

Since the Orca platform does not require agents to inspect cloud resources, coverage is always continuous and 100%, and will immediately scan any new AI resources as soon as they are deployed and alert to any detected risks. However Orca goes beyond just covering AI models – the platform also maps out all the AI related packages that are used to train or infer AI.

By adding AI security to our comprehensive cloud security platform, organizations don't need to procure, deploy, or learn how to use another separate point solution but can instead leverage one unified platform for all cloud security needs.

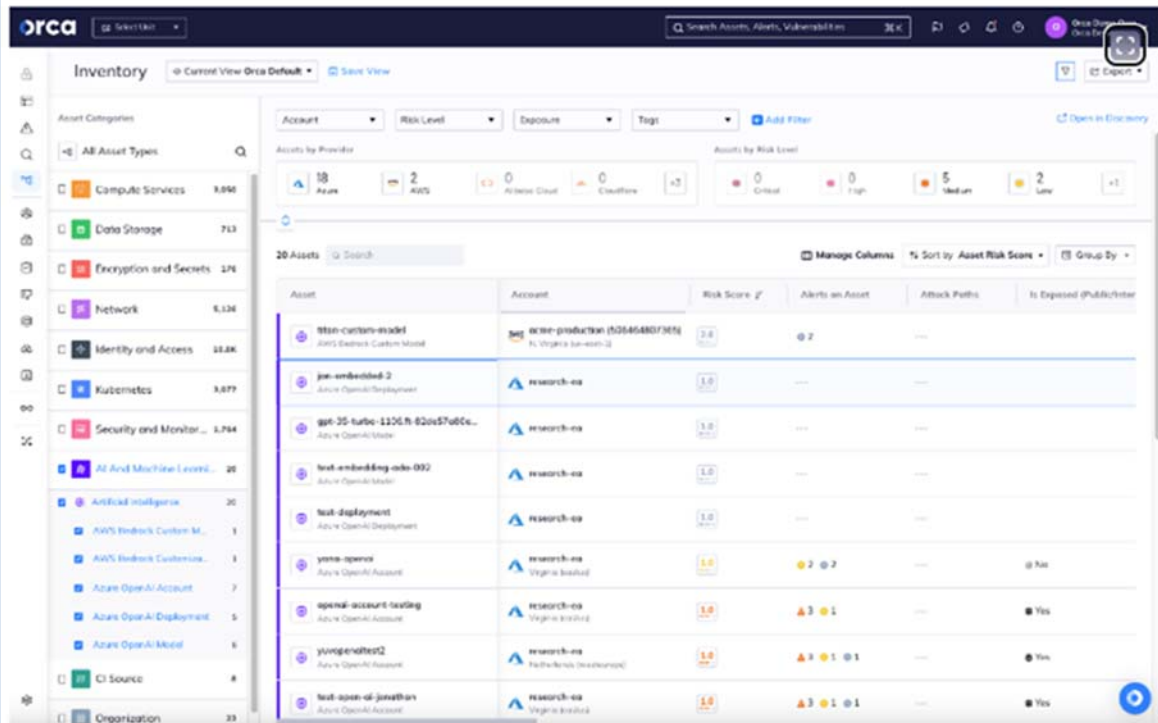
What are Orca's AI Security capabilities?

Orca includes a new AI Security dashboard that provides an overview of the AI models that are deployed in the cloud environment, what data they contain, and whether they are at risk. Orca covers risks end-to-end, from training and fine-tuning to production deployment and inference. Orca connects with your AI data and processes on all levels – the managed cloud AI services, the unmanaged AI models and packages that your developers are using, and even with shift-left detection of leaky secret tokens AI services in your codebase.

#1. AI and ML BOM + inventory

Challenge: Much like other resources in the cloud, shadow AI and LLMs are a major concern. In their excitement to explore all the opportunities that Generative AI brings, developers are not always waiting for IT approval before integrating Gen AI services into their flows. And security is probably not the first thing on their mind. This leaves security teams in the dark about which AI projects have been deployed in their environment, whether they contain sensitive data, and whether they are secure.

Orca solution: Orca scans your entire cloud environment and detects all deployed AI models, providing a full inventory and Bill of Materials (BOM). Orca detects projects on Azure OpenAI, Amazon Bedrock, Google Vertex AI, AWS Sagemaker and those using one of 50+ most commonly used AI software packages, including Pytorch, TensorFlow, OpenAI, Hugging Face, scikit-learn, and many more.



The Orca Platform shows a full inventory and bill of materials of all deployed AI models

#2. Warn when AI projects are publicly accessible

Challenge: Data used to train AI models is often sensitive and can contain proprietary information. Since it could be very damaging if this information got breached, it's very important to ensure that AI models are not publicly exposed. However, misconfigurations do happen, and especially with Shadow AI, security may not be the first thing developers have on their mind. This leaves security teams struggling to ensure that all AI models are being kept private.

Orca solution: Since Orca has insight into the AI model settings and network access, Orca will alert whenever public access is allowed, so security teams can quickly fix the issue to prevent any data breaches.

The screenshot shows the Orca AI Security dashboard for 'Orca AI Best Practices'. Key metrics include an average score of 55% (based on scored tests) and 27 total control tests, with 11 passed and 9 failed. The dashboard is organized into sections: 1 Network Security (43% score) and 2 Data Protection (56% score). A table highlights specific control failures:

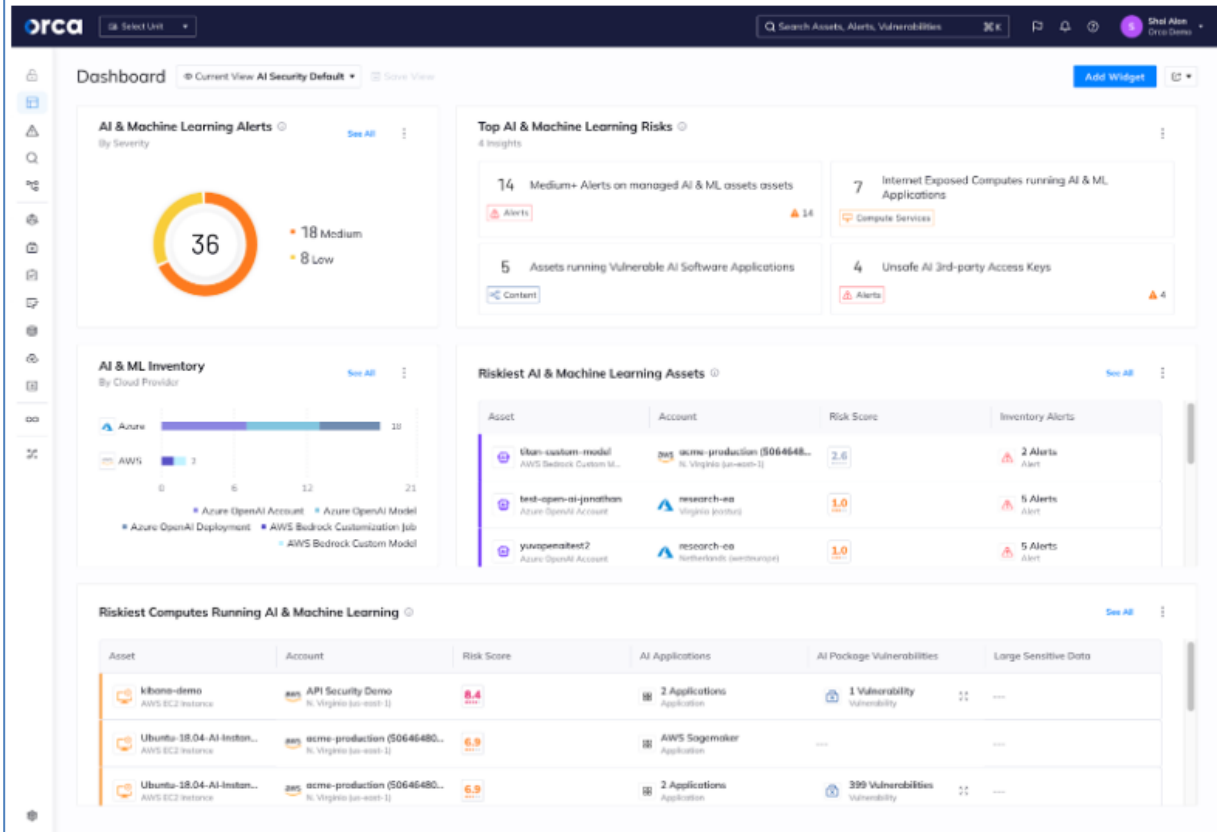
Control ID	Type	Priority	Description	Vendor	Open Alerts
1.2.1	system	low	Azure OpenAI account has no firewall exclusion configured for bypass Azure trusted services	Azure	1
1.2.2	system	low	AWS SageMaker notebook instance is configured with an outdated minimum Instance Metadata Service Version (IMDS)	AWS	0

Orca AI Security best practices compliance performs AI-Security Posture Management

#3. Detect sensitive data in AI projects

Challenge: AI models use vast amounts of data to train models. It's possible that this data inadvertently contains sensitive data, and that developers and security teams are not even aware of the presence of the data, such as PII and access keys. LLMs and Generative AI models could then “spill out” this sensitive data, which could be used by bad actors. In addition, if developers are not aware that the training set contains sensitive data, they may not prioritize security of the resource as much as it needs to be.

Orca solution: Using Orca’s [Data Security Posture Management \(DSPM\)](#) capabilities, Orca scans and classifies all the data stored in AI projects, and alerts if they contain sensitive data, such as email addresses, telephone numbers, email addresses, and social security numbers (PII), or personal health information (PHI). By informing security teams where sensitive data is located, they can make sure that these assets are protected with the highest level of security.



Orca displays pertinent security information in the AI Security dashboard

125. Upon information and belief, Orca further infringes multiple additional claims of the '529 patent. For example, but not limited to, claims 2, 4, 5, 6, 7 and 8.

126. Orca is aware of the '529 patent and Orca's infringement thereof at least as of the filing of these counterclaims. Moreover, this is another in a long line of examples of Orca copying Wiz, as discussed above. Accordingly, Orca has and continues to willfully infringe the '529 patent.

127. Orca has induced and continues to induce infringement of one or more claims of the '529 patent by encouraging customers to use AI-SPM in a manner that directly infringes those claims. Despite its knowledge of the existence of the '529 patent, since at least the filing of this Counterclaim, Orca, upon information and belief, continues to encourage, instruct, enable and otherwise cause its customers to use AI-SPM in a manner that infringes one or more claims of the '529 patent. Upon information and belief, Orca specifically intends that its customers use AI-SPM in a manner that infringes one or more claims of the '529 patent by, at a minimum, providing instructions and/or support documentation directing customers on how to use the AI-SPM in an infringing manner, in violation of 35 U.S.C. § 271(b). For example, Orca's public blog posts cited above provide instructions and encourage customers to practice all steps of the claimed method.

128. Orca has contributed and continues to contribute to the infringement of one or more claims of the '529 patent. Upon information and belief, Orca knows that AI-SPM features are especially made and/or adapted for users to infringe one or more claims of the '529 patent and is not a staple article or commodity of commerce suitable for substantial non-infringing use. Because Orca included features, such as for example but not limited to, AI-SPM in Orca's products, Orca intends for customers to use it. Upon information and belief, AI-SPM has no suitable use that is non-infringing, and therefore Orca intends for customers to use AI-SPM in an

infringing manner. Orca's sales of products including AI-SPM constitute contributory infringement in violation of 35 U.S.C. § 271(c).

PRAYER FOR RELIEF

WHEREFORE, Wiz respectfully asks that the Court enter judgment against Orca and in favor of Wiz as follows:

129. A judgment that Orca has infringed and continues to infringe (either literally or under the doctrine of equivalents) one or more claims of the Asserted Patents under at least 35 U.S.C. § 271(a);

130. A judgment that Orca has induced and continues to induce others to infringe one or more claims of the Asserted Patents under at least 35 U.S.C. § 271(b);

131. A judgment that Orca has contributorily infringed and continues to contribute to the infringement of one or more claims of the Asserted Patents under at least 35 U.S.C. § 271(c);

132. A judgment that Orca's infringement of the Asserted Patents has been and continues to be willful;

133. An award of monetary damages sufficient to compensate Wiz for Orca's patent infringement, with interest, pursuant to at least 35 U.S.C. § 284;

134. A preliminary and permanent injunction prohibiting Orca and its officers, agents, representatives, assigns, licenses, distributors, servants, employees, related entities, attorneys, and all those acting in concert, privity, or participation with them, from:

- (a) infringing or inducing the infringement of any claim of the Asserted Patents; and
- (b) soliciting any new business or new customers using any information or materials that Orca derived from its infringement of the Asserted Patents;

135. An award of enhanced damages of three times the amount found or assessed for Orca's willful patent infringement, pursuant to at least 35 U.S.C. § 284, including interest on such damages;

136. An order finding this case exceptional and awarding Wiz its attorneys' fees, to be obtained from any and all of Orca's assets, pursuant to 35 U.S.C. § 285, including prejudgment interest on such fees;

137. An accounting and supplemental damages for all damages occurring after the period for which discovery is taken, and after discovery closes, through the Court's decision regarding the imposition of a permanent injunction;

138. An award of Wiz's costs and expenses of this suit as the prevailing party; and

139. Any and all other relief in Wiz's favor that the Court deems just and proper.

JURY DEMAND

Orca hereby demands a trial by jury on all issues so triable.

OF COUNSEL:

Jordan R. Jaffe
Catherine Lacy
Callie Davidson
Alex Miller
WILSON SONSINI GOODRICH & ROSATI, P.C.
One Market Plaza
Spear Tower, Suite 3300
San Francisco, CA 94105
(415) 947-2000

Praatika Prasad
Wilson Sonsini
1301 Avenue of the Americas, 40th Floor
New York, NY 10019-6022
(212) 999-5800

Dated: June 4, 2024

/s/ Kelly E. Farnan

Frederick L. Cottrell, III (#2555)
Kelly E. Farnan (#4395)
Christine D. Haynes (#4697)
RICHARDS, LAYTON & FINGER, P.A.
One Rodney Square
920 N. King Street
Wilmington, DE 19801
(302) 658-6541
cottrell@rlf.com
farnan@rlf.com
haynes@rlf.com

Counsel for Defendant Wiz, Inc.

EXHIBIT A



US011722554B2

(12) **United States Patent**
Keren et al.

(10) **Patent No.:** **US 11,722,554 B2**
 (45) **Date of Patent:** ***Aug. 8, 2023**

(54) **SYSTEM AND METHOD FOR ANALYZING NETWORK OBJECTS IN A CLOUD ENVIRONMENT**

(71) Applicant: **Wiz, Inc.**, Palo Alto, CA (US)

(72) Inventors: **Shai Keren**, Tel Aviv (IL); **Danny Shemesh**, Tel Aviv (IL); **Roy Reznik**, Tel Aviv (IL); **Ami Luttwak**, Binyamina (IL); **Avihai Berkovitz**, Tel Aviv (IL)

(73) Assignee: **WIZ, INC.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **17/819,442**

(22) Filed: **Aug. 12, 2022**

(65) **Prior Publication Data**

US 2022/0394082 A1 Dec. 8, 2022

Related U.S. Application Data

(63) Continuation of application No. 17/109,883, filed on Dec. 2, 2020, now Pat. No. 11,431,786.

(51) **Int. Cl.**

G06F 15/173 (2006.01)
H04L 67/10 (2022.01)
H04L 49/00 (2022.01)
H04L 9/40 (2022.01)
H04L 41/50 (2022.01)
H04L 41/046 (2022.01)

(52) **U.S. Cl.**

CPC **H04L 67/10** (2013.01); **H04L 41/046** (2013.01); **H04L 41/5096** (2013.01); **H04L 49/70** (2013.01); **H04L 63/1433** (2013.01)

(58) **Field of Classification Search**

CPC ... H04L 67/10; H04L 41/046; H04L 41/5096; H04L 49/70; H04L 63/1433
 USPC 709/201, 204, 205, 217, 218, 219, 223, 709/224
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,392,539 B2 * 6/2008 Brooks H04L 63/0263 709/224
 10,171,300 B2 * 1/2019 Eggen H04L 41/0896
 10,924,347 B1 * 2/2021 Narsian H04L 47/12
 (Continued)

Primary Examiner — Liang Che A Wang

Assistant Examiner — Liangche A Wang

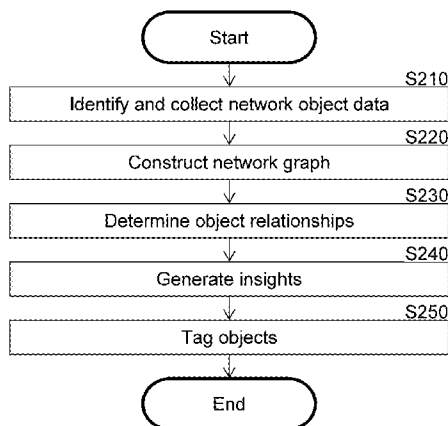
(74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC

(57) **ABSTRACT**

A method and system for determining abnormal configuration of network objects deployed in a cloud computing environment are provided. The method includes collecting network object data on a plurality of network objects deployed in the cloud computing environment; constructing a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment; determining relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects; and analyzing the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects.

21 Claims, 6 Drawing Sheets

200



US 11,722,554 B2

Page 2

(56)

References Cited

U.S. PATENT DOCUMENTS

2004/0019803	A1	1/2004	Jahn	
2014/0157417	A1*	6/2014	Grubel	H04L 63/20 726/25
2016/0044057	A1	2/2016	Chenette et al.	
2016/0048556	A1*	2/2016	Kelly	G06Q 10/10 707/767
2016/0359872	A1*	12/2016	Yadav	H04L 63/1425
2016/0373944	A1*	12/2016	Jain	H04L 43/08
2017/0075981	A1*	3/2017	Carlsson	H04L 41/0826
2018/0024981	A1*	1/2018	Xia	G06F 40/18 715/215
2019/0095530	A1*	3/2019	Booker	G06F 16/9024
2020/0267175	A1	8/2020	Atighetchi et al.	
2020/0322227	A1*	10/2020	Janakiraman	H04L 41/147
2020/0374343	A1*	11/2020	Novotny	G06F 16/9024
2020/0382539	A1*	12/2020	Janakiraman	H04L 63/0428

* cited by examiner

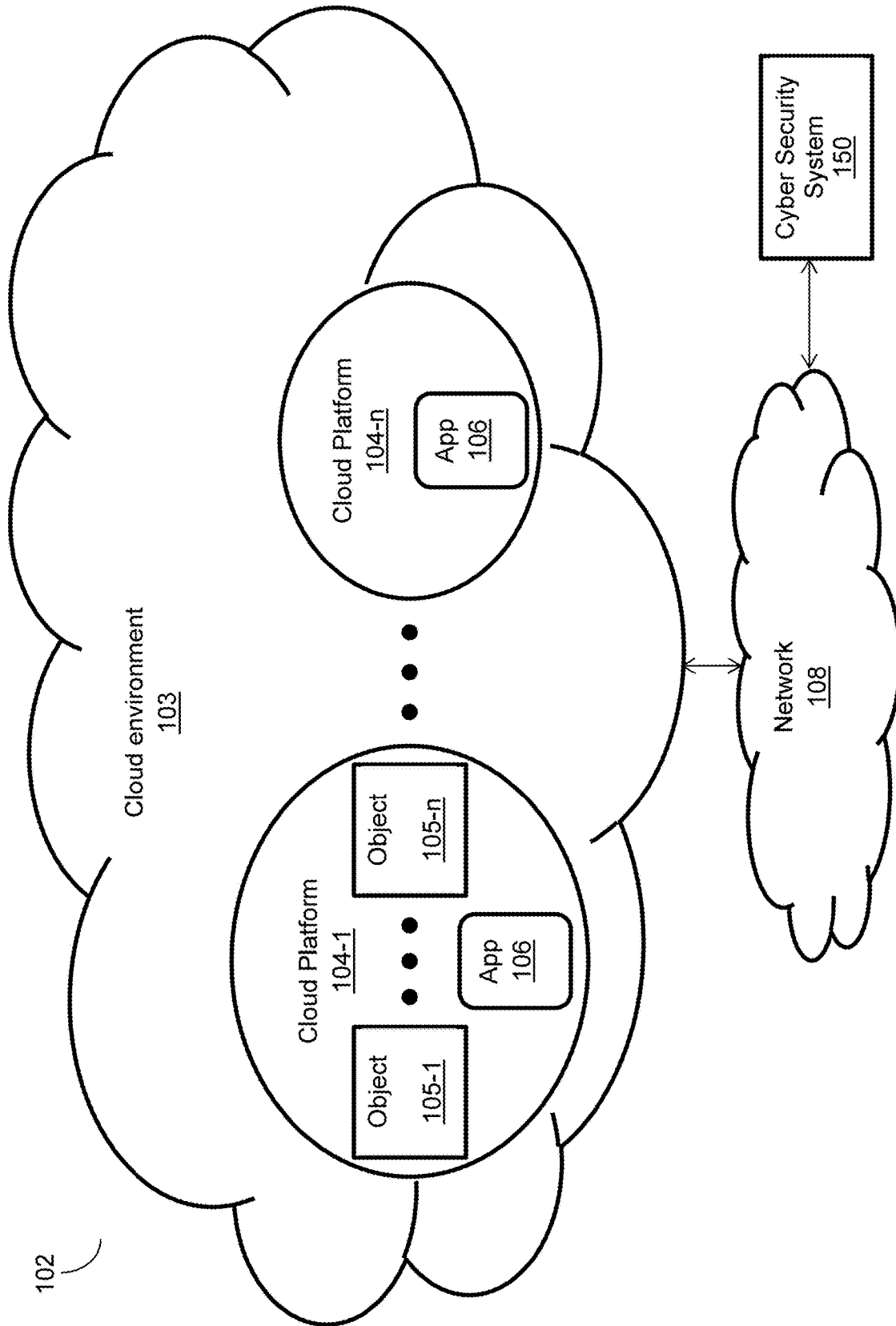


FIG. 1A

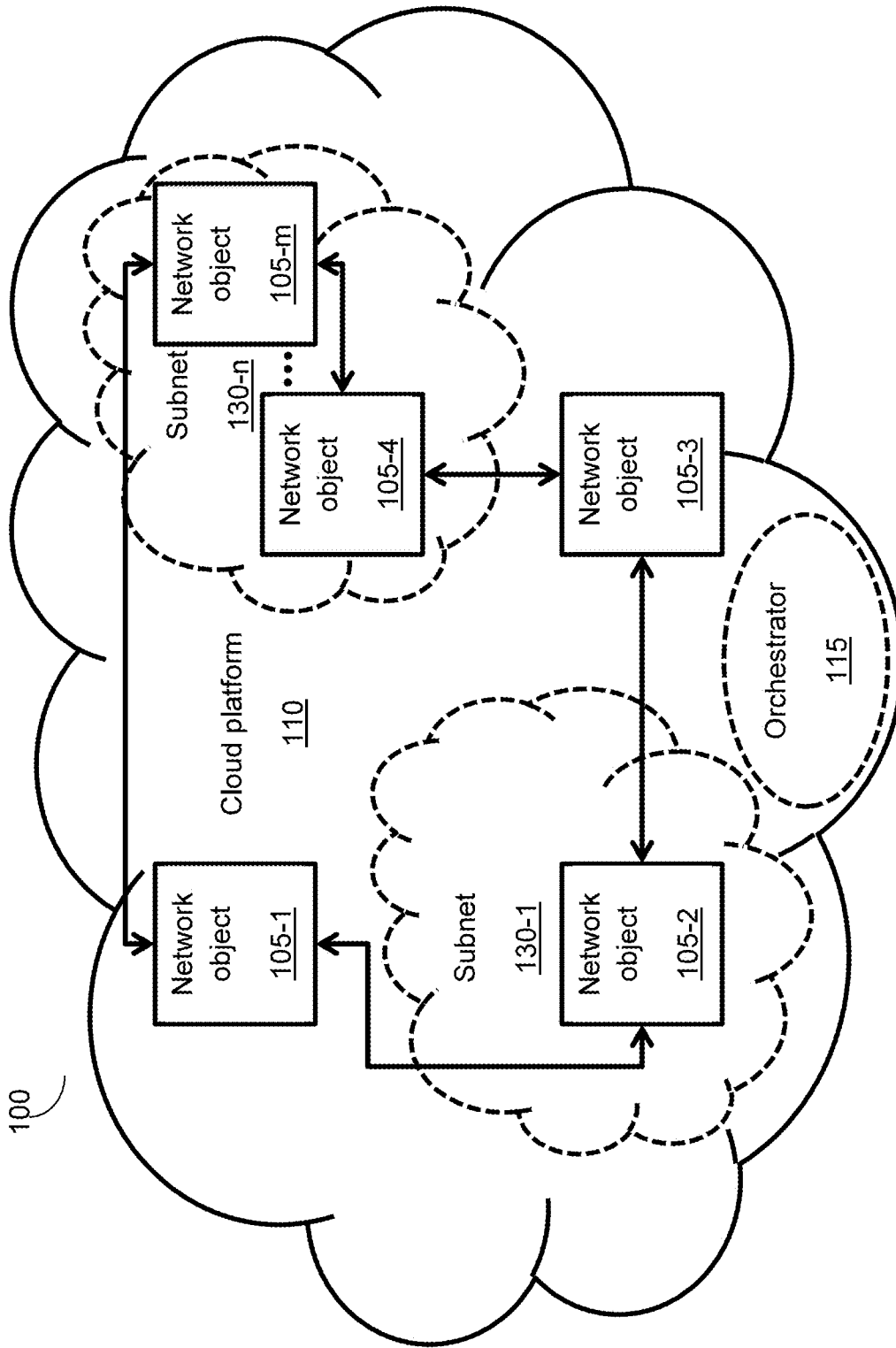


FIG. 1B

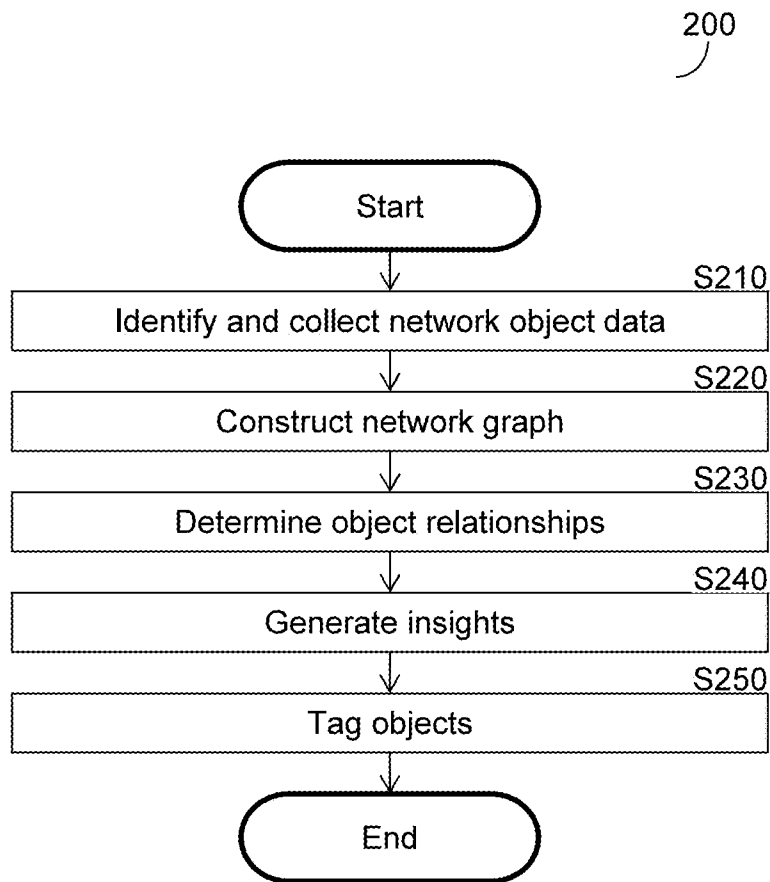


FIG. 2

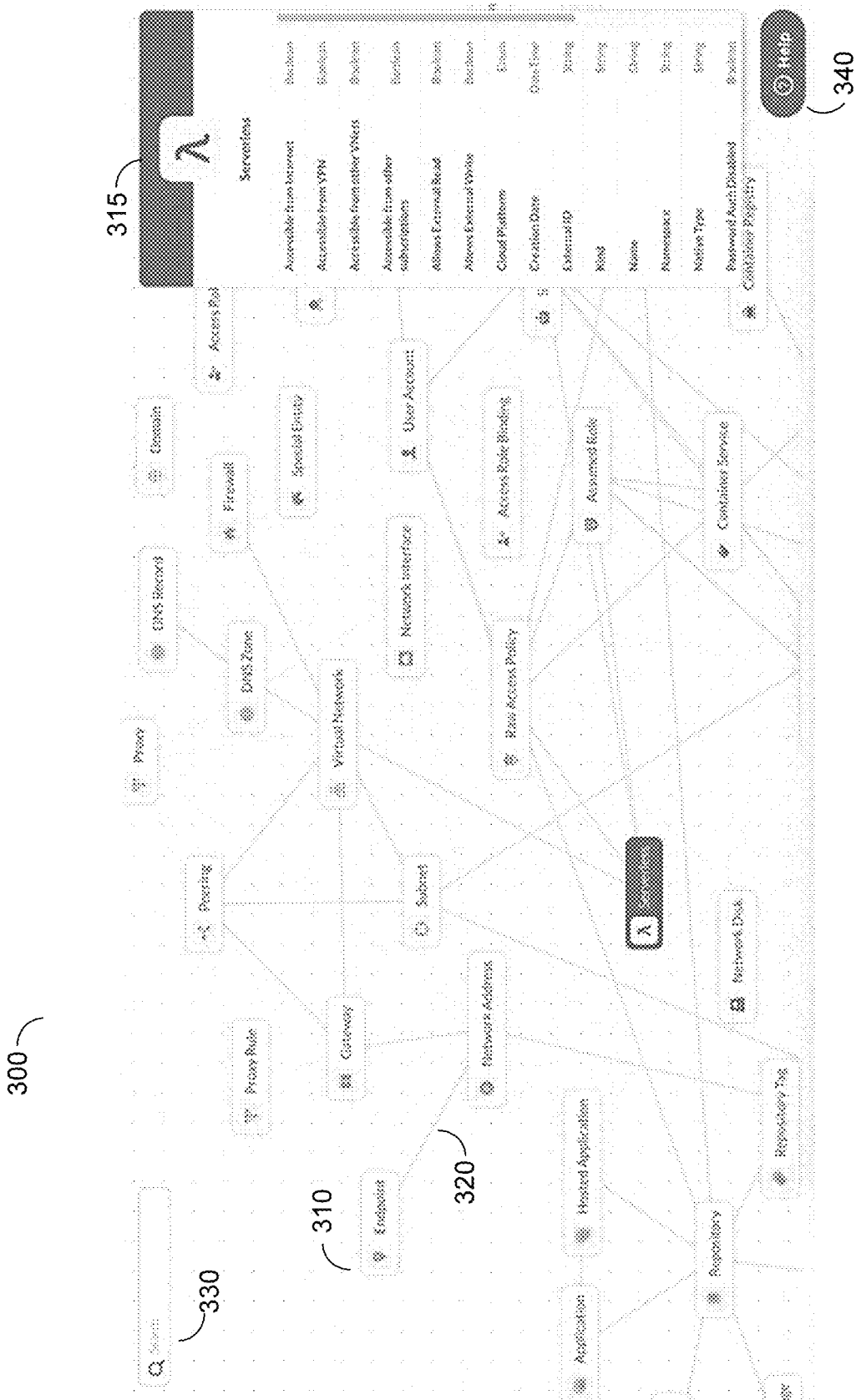


FIG. 3A

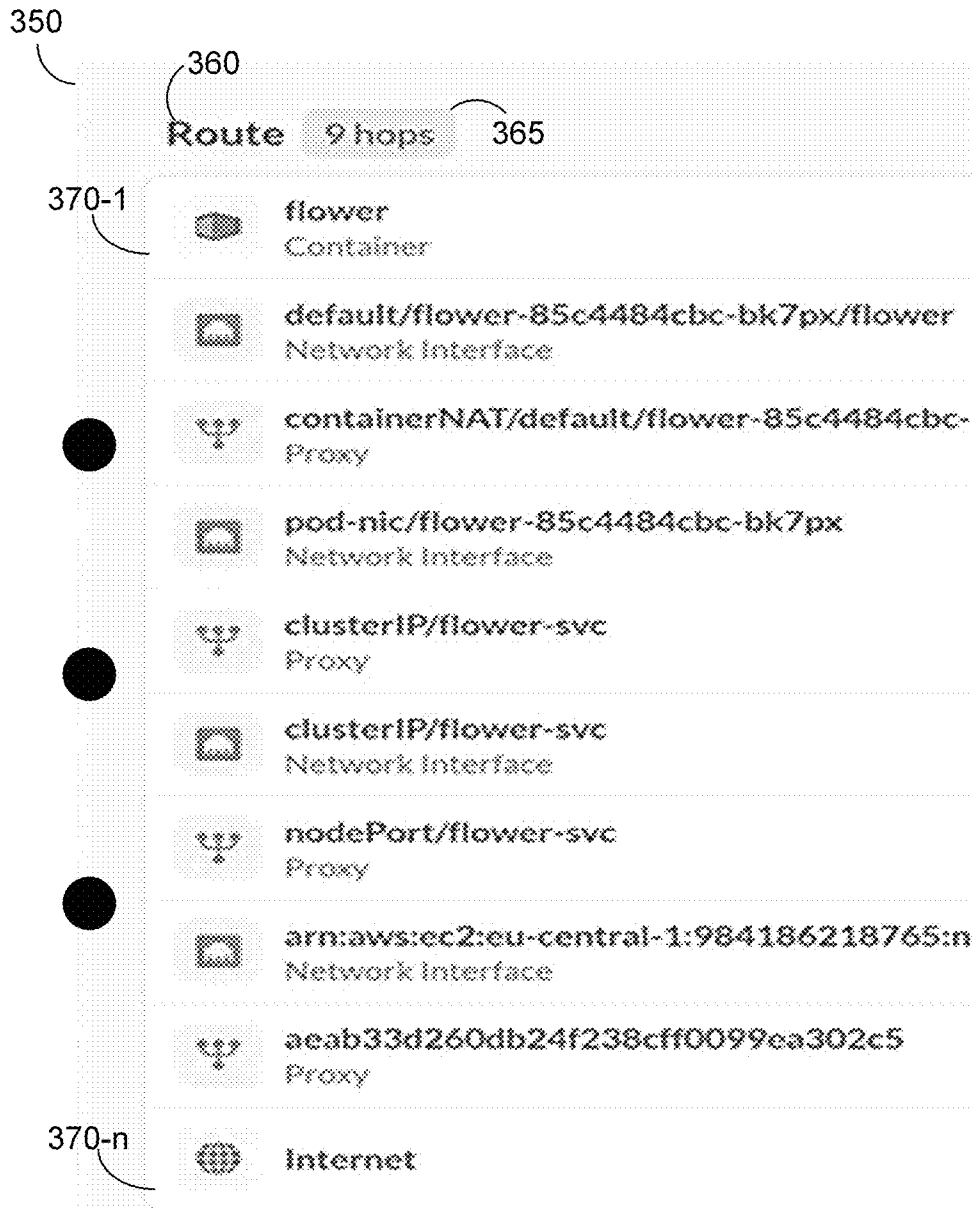


FIG. 3B

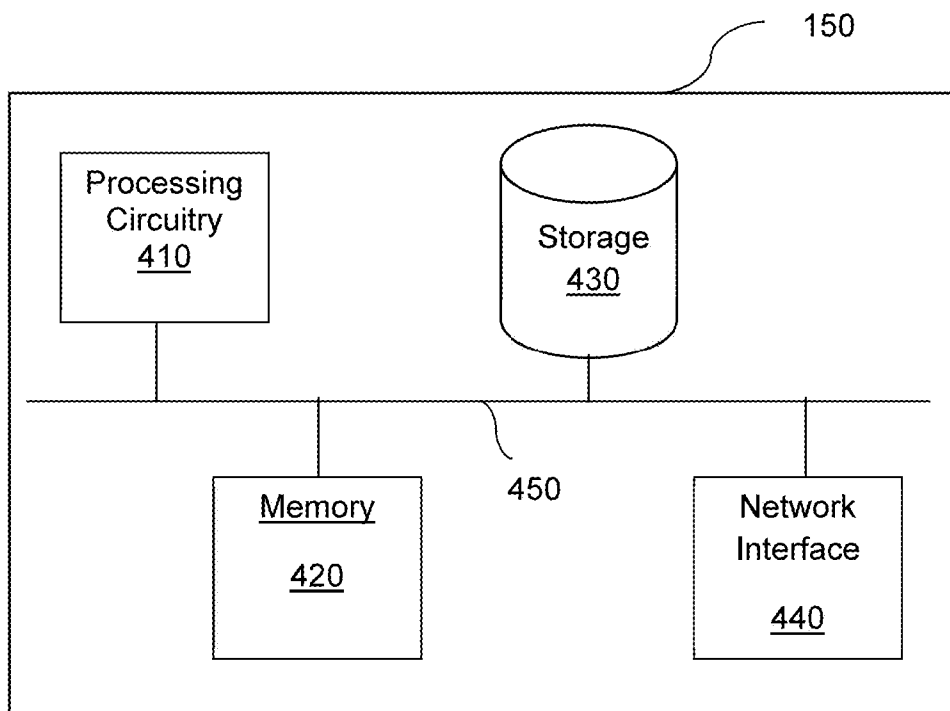


FIG. 4

US 11,722,554 B2

1

SYSTEM AND METHOD FOR ANALYZING NETWORK OBJECTS IN A CLOUD ENVIRONMENT

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 17/109,883 filed Dec. 2, 2020, the contents of which are hereby incorporated by reference.

TECHNICAL FIELD

The present disclosure relates generally to network administration, in particular, to systems and methods for analyzing networks.

BACKGROUND

As businesses, governments, and other organizations expand and increase their digital presence through various computer, network, and web technologies, the same parties may be increasingly vulnerable to developing cyber-threats. While updated solutions provide for management of prior cyber-threats, the same systems may include new vulnerabilities, which attackers may seek to identify and exploit to gain access to sensitive systems and data. Specifically, as organizations transition into multi-level computing systems, implementing computing solutions at the individual, group, team, and cloud levels, these systems, and the links between the elements of the layers, as well as the links between elements of different layers, include vulnerabilities which prior solutions fail to address.

Due to the distributed nature of large, multi-layered network systems, management of network access and use may be difficult or impossible for lone administrators or teams of administrators. Management of such code-to-cloud systems, and protection of the same, may require monitoring of large numbers of devices, systems, and components. Further, as each device, system, or component of a network system may be variously connected with the other elements of the system, including connections with multiple other devices via multiple protocols, management and monitoring of individual devices and connections may be untenable.

To address the need to manage large, distributed network systems, operators and administrators may employ various solutions to provide for network analysis. Certain network analysis solutions include manual review of devices, connections, and networks, providing for thorough, specific analysis of individual elements of a network. However, such manual solutions may require prohibitive outlays of time and effort to successfully review every component and connection of a large, multi-layer network, thus failing to provide a solution for analysis of modern network systems. In addition, various analysis solutions include solutions directed to the monitoring of specific device types, such as, for example, firewall control systems, which may provide for management of all firewalls installed in a given network. Similarly, protocol-specific analysis solutions may provide for monitoring of all traffic occurring over given protocols, within the network. However, such specialized solutions may fail to provide for streamlined monitoring and management of all components and connections of a network, where the network includes multiple types of devices communicating via multiple protocols. Further, protocol-agnostic solutions may provide for overall traffic management, providing monitoring and management solutions for all

2

traffic arising within a network. However, such protocol-agnostic solutions may be over-broad, providing irrelevant or redundant information, and may require specification of connections to monitor, reducing efficacy in network-management contexts, while failing to provide device-specific insights, thereby failing to provide for integrated device and connection analysis within a complex, multi-layer network.

In addition, certain solutions providing for the management of large, distributed network systems may fail to provide for agentless management, non-logging solutions, and the like. Agentless management, whereby such large, distributed network systems are managed without the use of a dedicated management agent system or device, may provide for reduced maintenance requirements, as a management agent may require operation and maintenance in addition to the efforts required by the remainder of the network. In addition to failing to provide for agentless management, various solutions for the management of large, distributed network systems fail to provide for non-logging management of the same. Non-logging management, where network analyses and other management processes are executed without reference to netflow logs, provides for reductions in management computing requirements and resource dependency when compared with logging solutions, which may require, without limitation, the execution of additional processing steps or tasks to analyze or process netflow logs, the dependency of the management solution or process on various netflow log resources or repositories, and the like. In addition to the shortcomings described above, current solutions for management of large, distributed network systems may fail to provide for agentless, non-logging management.

It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the terms “some embodiments” or “certain embodiments” may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

Certain embodiments disclosed herein include a for determining abnormal configuration of network objects deployed in a cloud computing environment. The method comprising: collecting network object data on a plurality of network objects deployed in the cloud computing environment; constructing a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment; determining relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects; and analyzing the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects.

In addition, certain embodiments disclosed herein include a system for determining abnormal configuration of network objects deployed in a cloud computing environment, comprising: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: collect network object data on a plurality of network objects deployed in the cloud computing environment; construct a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment; determine relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects; and analyze the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. 1A is a diagram of a cloud environment utilized to describe the various embodiments.

FIG. 1B is a network diagram depicting a network system and various associated network and external objects, according to an embodiment.

FIG. 2 is a flowchart depicting a method for constructing a network graph for a network system, according to an embodiment.

FIG. 3A is an example network graph schema, according to an embodiment.

FIG. 3B is a network graph object list, configured to provide information describing object-to-object routing within a network graph, according to an embodiment.

FIG. 4 is a hardware block diagram depicting a code compliance system, according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The systems and methods described herein may be applicable to various systems, devices, networks, environments, layers, and the like, as well as cross-connections or multi-entity connections as may be established therebetween. The disclosed systems and methods may be applicable to provide support for various network features including, without limitation, application-layer communications, cloud-native constructs, cross-cloud and Kubernetes-to-cloud communications, third-party features, such as third-party containers and objects, container-management systems, such as Kuber-

netes, as may be virtualized as cloud objects, and the like, as well as any combination thereof.

FIG. 1A is an example diagram **100** of a cloud environment **103** utilized to describe the various embodiments. A cloud environment **103** represents an organization's cloud-based resources, and the various connections between such resources. The cloud environment **103** may include a number of cloud computing platforms, **104-1** through **104-*n*** (hereinafter, "cloud platforms" **104** or "cloud platform" **104**), where a cloud platform may include multiple network objects, **105-1** through **105-*n*** (hereinafter, "network objects" **105** or "network object" **105**), one or more applications (collectively referred to as applications or apps **106**), and the like, as well as any combination thereof. Further, the cloud environment may be configured to connect, via a network **108**, with a cyber-security system **150** for one or more purposes including, without limitation, those described hereinbelow. As is applicable to the cloud platforms **104** and network objects **105**, "*n*" is an integer having a value greater than or equal to two. Further, it may be understood that, while a single configuration of a cloud environment **103** is shown for purposes of simplicity, a cloud environment **103** may include various combinations of platforms **104**, objects **105**, applications **106**, and the like, as well as any combination thereof, without loss of generality or departure from the scope of the disclosure.

A cloud platform **104** is a platform, architecture, or other, like, configuration providing for connectivity of the various objects **106**, applications **106**, and other, like, elements included in a cloud platform **104**, as well as the execution of various processes, instructions, and the like. A cloud platform **104** may be a commercially-available cloud system, provided on a service basis, such as, as examples and without limitation, Amazon AWS®, Microsoft Azure®, and the like. A cloud platform **104** may be a private cloud, a public cloud, a hybrid cloud, and the like. In addition, a cloud platform **104** may include, without limitation, container orchestration or management systems or platforms such as, as an example and without limitation, a Kubernetes® deployment, and the like, as well as any combination thereof.

A cloud platform **104** may be implemented as a physical network of discrete, interconnected objects, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized components. A cloud platform **104** may be, or may replicate or otherwise simulate or emulate, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof. Further, a cloud platform **104** may include one or more subnets, such as the subnets, **130**, of FIG. 1B, below, wherein each subnet may be configured to serve as a cloud platform **104** for the various network objects which may be included in the subnet, while retaining the connectivity and functionalities provided by the cloud platform **104**.

Network objects **105**, as may be included in a cloud platform **104**, are objects, systems, devices, components, applications, entities, and the like, configured to operate within the cloud platform **104** and provide various functionalities therein. Specifically, the network objects **105** may be objects configured to send, receive, or both send and receive, network data. The network objects **105** may be configured to connect with various other network objects **105**, various external objects, and the like, as well as any combination thereof, for purposes including, without limitation, sending

US 11,722,554 B2

5

data, receiving data, monitoring data transmissions, monitoring network status and activity, and the like, as well as any combination thereof.

Examples of network objects **105**, as may be relevant to the methods, processes, and descriptions provided herein include, without limitation, objects providing support for application-layer communications and systems, including application-layer communications and systems relevant to layer seven of the open systems interconnection (OSI) model. Further examples of network objects **105**, relevant to the methods, processes, and descriptions provided herein, include, without limitation, cloud-native constructs, such as private endpoints, transit gateways, tag-based rulesets and objects configured to apply such rules, Kubernetes Istio and Calico services and applications, and the like. In addition, examples of network objects **105** may include, without limitation, third-party containers and images, such as Nginx, web-access firewall (WAF), and firewall implementations, multi-object or cross-object connections, such as cross-cloud connections and Kubernetes-to-cloud connections, as well as container managers, such as Kubernetes, and connections therewith. It may also be understood that network objects **105** may include other objects similar to those described hereinabove, as well as any combination thereof. As another example, network objects may include virtual entities, devices, and the like, to process layer-7 (application layer) traffic, such as objects relevant to Amazon AWS® layer seven services and applications, Amazon Load Balancer® (ALB) layer seven services and applications, Kubernetes ingress, and the like.

The network objects **105** may be configured to include one or more communication ports, where the included communication ports provide for connection of various objects according to one or more protocols, and at different communication layers of the OSI model.

In an example configuration, the network objects **105** are virtual entities or instances of systems, devices, or components, including virtual systems, devices, or components, or any combination thereof. Examples of network objects **105** include, without limitation, virtual networks, firewalls, network interface cards, proxies, gateways, containers, container management objects, virtual machines, subnets **130**, hubs, virtual private networks (VPNs), and the like, as well as any combination thereof.

The applications **106**, as may be executed in one or more cloud platforms **104**, are services, processes, and the like, configured to provide one or more functionalities by execution of various commands and instructions. The applications **106** may be part of a software project of an enterprise or organization. The applications **160** may interact or communicate with other applications, regardless of the platform **104** in which the applications **106** are deployed. It should be understood that a single application, including the same application, may be both present and executed in multiple cloud platforms **104**, including multiple cloud platforms **104** of the same cloud environment **103**, without loss of generality or departure from the scope of the disclosure.

The network **108** is a communication system providing for the connection of the cloud environment **103**, and its various components and sub-parts, with a cyber-security system **150**, as well as other, like, systems, devices, and components, and any combination thereof. The network **108** may be implemented as a physical network of discrete, systems, devices, components, objects, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized compo-

6

nents. The network **108** may be, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof.

The cyber-security system **150** is a system, device, or component, configured to provide one or more network analysis functionalities including, without limitation, network analysis, traffic analysis, object querying, graph generation, and the like, as well as any combination thereof. The cyber-security system **150** may be configured to execute one or more instructions, methods, processes, and the like, including, without limitation, the process described with respect to FIG. 2, other, like, processes, and any combination thereof.

The cyber-security system **150** may be configured as a physical system, device, or component, as a virtual system, device, or component, or in a hybrid physical-virtual configuration. A detailed description of a cyber-security system, **150**, according to an embodiment, is provided with respect to FIG. 4, below. It may be understood that, while the cyber-security system **150** is depicted in FIG. 1A as a discrete element external to the cloud environment **103**, the cyber-security system **150** may be included within any of the various elements of the network system **102**, including the cloud environment **103**, the various cloud platforms **104**, and subparts thereof, and the network **108**, without loss of generality or departure from the scope of the disclosure.

FIG. 1B is an example diagram depicting a network system **100** and various associated network and external objects, according to an embodiment. The depicted network system **100** includes a cloud platform **110**, where the cloud platform **110** may be a cloud platform similar or identical to a cloud platform, **104**, of FIG. 1A, above. The cloud platform **110** includes various subnets, **130-1** through **130-*n*** (hereinafter, “subnets” **130** or “subnet” **130**), and various network objects, **105-1** through **105-*m*** (hereinafter, “network objects” **105** or “network object” **105**). As applicable to the subnets **130**, “*n*” is an integer having a value greater than or equal to two. Further, as applicable to the network objects **105**, “*m*” is an integer having a value greater than or equal to five. In addition, while the network system **100** of FIG. 1B includes certain elements and combinations of elements, as well as connections therebetween, it may be understood that the depiction is provided for illustrative purposes, and that other, like, elements, combinations of elements, and connections therebetween may be implemented without loss of generality or departure from the scope of the disclosure. Other, like, network systems **100** may further include multiple cloud platforms **110**, including variously-interconnected cloud platforms **110**, and other, like, variations and configurations, without loss of generality or departure from the scope of the disclosure.

As described with respect to FIG. 1A, above, the cloud platform **110** is a platform, architecture, or other, like, configuration providing for connectivity of the various systems, devices, and components described with respect to FIG. 1B. The cloud platform **110** may be a commercially-available cloud system, provided on a service basis, such as, as examples and without limitation, Amazon AWS®, Microsoft Azure®, and the like. The cloud platform **110** may be a private cloud, a public cloud, a hybrid cloud, and the like. The cloud platform **110** may be implemented as a physical network of discrete, interconnected objects, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized components. The cloud platform **110** may be, or may

US 11,722,554 B2

7

replicate or otherwise simulate or emulate, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof. Further, as described with respect to FIG. 1A, above, the cloud platform 110 may include one or more subnets 130, wherein each subnet 130 may be configured to serve as a cloud platform 110 for the various network objects 105 included in the subnet 130, while retaining the connectivities and functionalities provided by the cloud platform 110.

The cloud platform 110 may be configured to include an orchestrator 115. The orchestrator 115 is configured to provide for management of the cloud platform 110. The orchestrator 115 may be configured to provide one or more functionalities including, without limitation, monitoring of elements or components of the cloud platform 110, logging and reporting data relating to the cloud platform 110, managing cloud platform 110 updates and maintenance, generating cloud platform 110 alerts, as well as other, like, functionalities, and any combination thereof. The orchestrator 115 may be configured to report one or more data features related to the cloud platform 110, such as may be requested during the execution of network analysis processes, such as those described hereinbelow.

The network objects 105 are network objects similar or identical to those network objects, 105, of FIG. 1A, above. As described with respect to FIG. 1A, the network objects 105 are virtual entities or instances of systems, devices, or components, including virtual systems, devices, or components, or any combination thereof. Examples of network objects 105 include, without limitation, virtual networks, firewalls, network interface cards, proxies, gateways, containers, container management objects, virtual machines, subnets 130, hubs, virtual private networks (VPNs), peering connections, load balancers, route tables, and the like, as well as any combination thereof.

External objects, as may be adjacent or relevant to a cloud platform 110, are objects similar or identical to the network objects 105. The external objects may be configured to communicate with one or more network objects 105, with other, various, external objects, and the like, as well as any combination thereof.

FIG. 2 is an example flowchart 200 depicting a method for constructing a network graph for a network system, according to an embodiment.

At S210, network objects are identified, and network object data is collected. In one embodiment, network objects may be identified by querying a cloud platform, through, for example, an orchestrator (e.g., orchestrator 115, of FIG. 1B, above), and the like. In an embodiment, S210 may include submitting one or more requests to each cloud platform and collecting responses therefrom. The requests may include instructions directing the orchestrator to report information including, without limitation, the number of devices connected to or included in the cloud platform, the names of such devices, the types of such devices, other, like, information, and any combination thereof.

In an embodiment, identification of network objects and collection of network object data at S210 includes querying each cloud platform, where such querying may include generation of one or more queries through an application programming interface (API), such as a REST API. Through the API, network objects' identities and description data are provided in response to such API queries. API queries may be pre-configured data requests, specified in the API, and configured to cause, for example, an orchestrator to return the one or more data features described herein. API queries

8

may be generated based on one or more APIs, or the like, including generic APIs, such as REST, as well as platform-specific APIs, where such platform-specific APIs may be configured to provide for one or more predefined interactions with a cloud platform, such as Amazon AWS®, Microsoft Azure®, and the like, where such predefined interactions may include, without limitation, network object identification and data collection.

Further, at S210, network object data is collected. Network object data is data describing one or more network objects, such as those objects, 105, of FIG. 1B, above. Network object data may include data describing, as examples and without limitation, object types, object names or unique identifiers (IDs), object network addresses, object port configurations, object status, such as online, offline, busy, and available, object input signal configurations, object output signal configurations, object security or access configurations, object processing rules and the like, as well as any combination thereof. Object data may be collected at S210 from one or more sources including, without limitation, various networks, network monitors, subnets, external objects, network objects, and the like, such as the cloud platform, 110, cyber-security system, 150, subnets, 130, external objects, network objects, 105, cloud platform orchestrators, 115, all of FIG. 1B, above, and the like, as well as any combination thereof. Collection of network object data, at S210, may be executed via one or more means including, without limitation, generation and transmission of one or more API queries, such as are described hereinabove, by other, like, means, and any combination thereof.

As a first example, collection of network object data at S210 may include collection of the identities of all objects included in a cloud platform by generation and transmission of an API query. In a second example, where a specific object, such as a given firewall, is specified in an API query, collection of network object data at S210 may include collection of object data from a firewall, including the collection of firewall rules, collection of firewall event logs, collection of firewall port configurations, and the like, as well as any combination thereof. As a third example, collection of network object data at S210 may include collection of object data from all virtual machines (VMs) in a cloud platform, where such VMs are described generally in an API query, including the collection of data resources or libraries internal to the VMs, VM port configurations, VM statuses, and the like, as well as any combination thereof.

At S220, a network graph is constructed. A network graph is a data feature describing the various objects included in, and adjacent to, a network, as well as the relationships between such objects. A network graph may be constructed based on data including, without limitation, data relevant to the objects identified at S210, from which data is collected, and the like, as well as any combination thereof. A network graph may be constructed in one or more various formats including, without limitation, a table, chart, or other, non-visual, data organization format, a list of objects, other, like, formats, and any combination thereof, where such formats may provide network object information including, without limitation, descriptions of network objects, properties, relations, and the like, as well as any combination thereof. In an embodiment, construction of a network graph, at S220, may include construction of a visual "node and link" graph. An example network graph schema, generated in a visual format and presented through a network graph utility, is described with respect to FIG. 3A, below. An additional network graph data feature, including a list of network objects, where the

list is configured to describe an object-to-object path, is described with respect to FIG. 3B, below.

At S230, relationships between network objects are determined. Network object relationships are descriptions of the various connections between the network objects identified at S210. Network relationships may describe aspects of the connections between objects including, without limitation, connected objects, relevant ports of connected objects, connection bandwidths, connection durations, connection protocols, connection names or IDs, connection statuses, and the like, as well as any combination thereof.

In an embodiment, network object relationships may be determined at S230 using a static analysis process. In this embodiment, the static analysis may include analysis of object and protocol code and rules, based on simulated network operation, as collected at S210, to provide for identification of network object relationships based on network object configurations. As an example of a static analytic determination, data collected from a firewall at S210 may specify, in the firewall's port configurations, communication with a first device on a first port using a first protocol and connection with a second device on a second port using a second protocol. Further, according to the same example, the firewall object may include one or more instructions specifying transmission of a specific log file, via a third port, to a connected repository. According to the same example, network object relationships determined at S230 may include connections between the firewall and the first device, connections between the firewall and the second device, and connections between the firewall and the repository.

Determination of network object relationships at S230 may further include updating the graph or graphs constructed at S220 to include the determined relationships. Graphs may be updated at S230 by associating one or more data labels, tags, or other, like, features with a graph entry for a network object determined to have a relationship with another object. The association of data labels and tags may further include the association of labels or tags describing various aspects of the determined relationship or connection including, as examples and without limitation, connection source and destination, connection type, connection direction, connection status, connection protocol, and the like, as well as any combination thereof. Accordingly, as an example, determination, at S230, of a relationship between two objects may include the association of a data label or tag with each object included in the relationship, the data label or tag describing the same relationship for each object. Further, in an embodiment, where a graph is presented as a visual representation of a network system, such as a "node-and-link" graph, updating of the graph, at S230, based on determined relationships, may further include updating the visual graph to include visible "links" or connections between object "nodes," such as by, as examples and without limitation, updating the original visible graph to include such links, adding a second, visible overlay to the graph, including the links, and the like, as well as any combination thereof.

In addition, determination of network object relationships at S230 may include analysis of such determined relationships to identify impermissible relationships. Where determination at S230 includes such permissibility analysis, such analysis may include, without limitation, comparison of determined relationships with one or more dictionaries, or other, like, repositories of object relationship information, to determine whether a given relationship matches a predefined relationship included in the dictionary, where such a pre-

defined relationship may be pre-tagged as "permissible," "not permissible," or the like. Where a determined relationship is determined to match a relationship which has been pre-defined as "not permissible" or as otherwise unacceptable, the relationship may be removed from the graph, such as by updating the graph in a manner similar to that described with respect to adding relationships to the graph, with the update providing for removal of one or more specified unacceptable relationships.

Further, in an embodiment, network object relationships may be determined at S230 by application of observational or active logging methods, such as those methods providing for detection of object-to-object connections by monitoring traffic of a network in use.

At S240, network insights are generated. Network insights are natural-language representations of aspects of the network graph constructed at S220. Network insights may include pure-text descriptions of objects and relationships. An example of a pure-text object-relationship description, generated as an insight at S240, may be "firewall one is connected to object two, which is a VM, and object three, which is a load balancer." Such representations may be in a query format.

In addition, network insights may include detailed descriptions of objects, relationships, and the like, as well as any combination thereof. An example of a detailed object-relationship description, generated as an insight at S240, may be "the gateway is currently active, and is connected to the VM via the second port, using the first protocol."

As another example, including a multiple-step relationship, an insight may be generated at S240, the insight specifying a path from "virtual machine one to load balancer five, where port eighty is routed to port 1337 on virtual machine one, then from load balancer five to firewall two, where port eighty of firewall two is open, then from firewall two to subnet sixteen, where subnet sixteen has a network address of 10.0.1.0/24, then from subnet sixteen virtual network ten, where virtual network ten has a network address of 10.0.0.0/16, then from virtual network ten to virtual network eleven, via peering connection twelve, where peering connection twelve includes a routing rule to virtual network twelve, specifying virtual network twelve's network address, where virtual network twelve's network address is 172.31.0.0/16." The insight described with respect to the second example may be interpreted to describe a virtual machine accessible from a virtual network via a series of hops, where only specific ports and addresses are allowed and routed through the firewall.

Further, generation of network insights at S240 may include the generation of insights providing for network management or anomaly detection. Where generation of insights at S240 includes generation to provide for such functions, as well as other, similar functions, generation of insights may include, as examples and without limitation, generation of insights describing network configurations or events which are rare or novel, such as connection of a new device to a network, connections which are unauthorized, such as re-connection of a user's device to a subnet which the user is not permitted to access, connections which display anomalous behavior, such as connections displaying spikes of network activity, and the like. Such insights may be generated according to one or more pre-defined or user-defined filters, rules, and the like, as well as any combination thereof. As an example, generation of network insights at S240 may include generation of an insight specifying that "VM thirty normally connects to load bal-

US 11,722,554 B2

11

ancer twenty and firewall eight but is currently only connected to load balancer twenty.”

In addition, generation of network insights at S240 may include the generation of high-level insights, where high-level insights are insights similar to those described hereinabove and which are configured to include information describing one or more features of a network which may not be detectable based on the analysis of individual objects. Examples of high-level insights, as may be generated at S240, include, without limitation, “third-party networks A, B, and C currently have access to the internal network,” “objects E, a database, and F, an administrator interface, are currently exposed to external networks,” and “cross-environment exposure has been detected between the development and production environments.”

At S250, network objects are tagged. Tagging of network objects at S250 may include, without limitation, association of one or more data labels, tags, or other, like, features, with graph entries for one or more network objects, including entries in graphs such as those described with respect to S220, above. The data labels, tags, or other, like, features, with graph entries, may include descriptions of aspects of relevant graphs, objects, relationships, and the like, as well as any combination thereof. Examples of relevant descriptions include, as examples and without limitation, insights, such as those generated at S240, object relationships, such as those determined at S230, object details, such as may be collected at S210, descriptions of whether an object appears in another graph, object type counts, per-object connection counts, graph connection counts, descriptions of an object’s open ports and counts thereof, descriptions of an object’s network address or addresses, descriptions of protocols relevant to an object, and other, like, descriptions, as well as any combination thereof.

In an embodiment, tagging of network objects at S250 may provide for enriched querying of graphs, such as through various network graph utilities, including a utility associated with the network graph schema described with respect to FIG. 3A, below, as well as other, like, presentations of network graph information, such as are described with respect to FIG. 3B, below, where such enriched querying may include the searching of one or more graphs for objects associated with one or more data labels or tags, including compound queries specifying multiple data labels or tags.

FIG. 3A is an example screenshot of a network graph schema 300, generated according to an embodiment. The example network graph schema 300 is generated as a visual representation of a network, such as at S230 of FIG. 2, above, and is presented through a network graph utility. A network graph utility may be an application, interface, or other, like, means of providing a visual representation of a network graph schema 300, and the like, where the provided network graph schema 300 may include various interactive features, as described hereinbelow. A network graph utility may be configured as, as examples and without limitation, a web interface, an application or executable installed on a user or administrator device, other, like, configurations, and any combination thereof.

The network graph schema 300 of FIG. 3A is a network graph visualization representing a network, such as the networks described hereinabove, wherein the various objects, systems, devices, components, and the like, of the network are represented as nodes 310, wherein such nodes are variously interconnected by links 320, representing connections between the various objects, systems, devices, components, and the like. It may be understood that while

12

only one node 310 and one link 320 are labeled for purposes of simplicity, other, like, nodes 310 and links 320 may be so labeled without loss of generality or departure from the scope of the disclosure.

The network graph schema 300, and corresponding network graph utility, may be configured to provide for various interactive functionalities. In an embodiment, where a user interacts with a node 310, such as by clicking the node 310 with a mouse or tapping the node 310 through a touchscreen, the graph utility may be configured to display a node overview pane 315. The node overview pane 315 may be an information panel, including data relating to the given node 310 and describing various object data features, such as those object data features collected at S210 of FIG. 2, above. The node overview pane 315 may be configured to provide information relating to the various nodes 310 including, as examples and without limitation, object names, types, statuses, relevant metadata, and the like, as well as any combination thereof.

Further, the network graph schema 300, and corresponding network graph utility, may be configured to include a search tool 330, providing for location and selection of one or more user-specified nodes 310 or links 320 within the graph. The search tool 330 may be configured to provide for search functionality based on one or more user specifications including, as examples and without limitation, object names, types, IDs, statuses, labels or tags associated with various elements of the network graph schema 300, and the like, as well as any combination thereof. In addition, the network graph schema 300, and corresponding network graph utility, may be configured to include a help tool 340, providing for display of one or more resources related to the network graph schema 300 and network graph utility.

It should be noted that a network graph schema 300, shown in FIG. 3A, may be constructed in other formats including, without limitation, a table, chart, or other, non-visual, data organization formats, a list of objects, other, like, formats, and any combination thereof.

FIG. 3B is an example network graph object list 350, configured to provide information describing object-to-object routing within a network graph, according to an embodiment. The network graph object list 350 includes a mode indicator 360, a mode-specific data display 365, and a list of objects, 370-1 through 370-n (hereinafter, “object” 370 or “objects” 370), where ‘n’ is an integer having a value greater than or equal to two. It may be understood that, while the provided network graph object list is configured to provide a list of objects 370 arranged according to a defined mode, other, like, modes and object 370 arrangements may be similarly applicable without loss of generality or departure from the scope of the disclosure.

The network graph object list 350 is a list of objects within a network, a segment of a network, a path of a network, and the like, as well as any combination thereof. A network graph object list 350 may be generated or provided as a function of one or more methods including, without limitation, those methods described herein, other, like, methods, and any combination thereof. A network graph object list 350 may be, without limitation, a feature of a network graph management tool or utility, such as a tool or utility configured to provide the network graph schema of FIG. 3A, above, a stand-alone tool or utility, and the like, as well as any combination thereof. The network graph object list 350 may be configured to list network objects 370 in one or more orders based on factors including, without limitation, object names, object types, object connection latencies, mode-specific factors, other, like, factors, and any combination

thereof. Where the network graph object list **350** is configured to list network objects **370** based on mode-specific factors, the mode indicator **360** may be configured to provide information describing a specific list mode, and the mode-specific data display **365** may be configured to provide information describing the contents of the list **350** in relation to one or more selected modes.

Modes are selected list-organization profiles, configured to provide for population of a network graph object list **350** in one or more configurations. Modes may provide for configuration of a network graph object list **350**, including configurations specific to, as examples and without limitation, routing paths, object utilization or availability descriptions, other, like, configurations, and any combination thereof. Where a given mode is selected, the selected mode may be displayed via a mode indicator **360**, where the mode indicator is configured to provide descriptive information regarding a selected mode. Further, where a mode is selected, the mode-specific data display **365** may be configured to display information regarding a specific network graph object list **350** populated based on the specified mode or modes.

As an example, with reference to the provided FIG. **3B**, a “route” mode may be selected, providing for population of a network graph object list **350** with objects occupying a data path between a first object **370-1** and a destination object **370-n**. According to the same example, the mode indicator **360** may be configured to display “Route,” indicating that the network graph object list **350** is populated to provide information describing a route between the specified objects, and the mode-specific data display **365** may be configured to display “9 hops,” indicating that a transmission from a first object **370-1** makes nine “hops,” or transmissions between objects, before reaching the destination object **370-n**. Further, according to the same example, the network graph object list **350** may be configured to include and display each network object **370** through which the transmission passes, including the first object **370-1** and the destination object **370-n**, in the order of transmission.

FIG. **4** is an example hardware block diagram **400** depicting a cyber-security system **150**, according to an embodiment. The cyber-security system **150** includes a processing circuitry **410** coupled to a memory **420**, a storage **430**, and a network interface **440**. In an embodiment, the components of the cyber-security system **150** may be communicatively connected via a bus **450**.

The processing circuitry **410** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory **420** may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof.

In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage **430**. In another configuration, the memory **420** is configured to store such software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode,

hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the processing circuitry **410**, cause the processing circuitry **410** to perform the various processes described herein.

The storage **430** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or another memory technology, compact disk-read only memory (CD-ROM), Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information.

The network interface **440** allows the cyber-security system **150** to communicate with the various components, devices, and systems described herein for network analysis, as well as other, like, purposes.

It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **4**, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

It should be noted that the computer-readable instructions may be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code, such as in source code format, binary code format, executable code format, or any other suitable format of code. The instructions, when executed by the circuitry, cause the circuitry to perform the various processes described herein.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units (CPUs), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform, such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

As used herein, the phrase “at least one of” followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including “at least one of A, B, and C,” the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and

US 11,722,554 B2

15

embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A method for determining abnormal configuration of network objects deployed in a cloud computing environment, comprising:

collecting network object data on a plurality of network objects deployed in the cloud computing environment; constructing a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment; determining relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects; analyzing the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects; and tagging network objects in the network graph for which the insight is generated.

2. The method of claim 1, wherein the cloud computing environment includes a plurality of different cloud computing platforms.

3. The method of claim 2, wherein collecting network object data further comprises:

connecting through an application programming interface (API) each of the plurality of different cloud computing platforms to collect network object data of network objects deployed in the respective cloud computing platform.

4. The method of claim 1, wherein generating insights to include at least the list of abnormal connections further comprises:

detecting any one of: a rare network configuration, a novel network configuration, a rare network event, a novel network event, a connection of a new device to a network, an unauthorized connection, a re-connection of a user device to a subnet which the user is not permitted to access, a connection demonstrating anomalous behavior or misconfiguration, and a connection demonstrating spikes of network activity.

5. The method of claim 4, further comprising: determining impermissible relationships between the network objects.

6. The method of claim 1, wherein determining relationships between the identified objects further comprises:

determining the relationships using a static analytic method.

7. The method of claim 1, wherein determining the relationships between the identified network objects in further comprises:

determining the relationships using at least one of: observational methods, and active logging methods.

8. The method of claim 1, further comprising: adding visual representations of the determined relationships to the visual representations of the network graph.

9. The method of claim 1, wherein each of the plurality of network objects includes any one of: a virtual network, a firewall, a network interface card, a proxy, a gateway, a

16

software container, container, a management object, a virtual machine, a subnet, a hub, a virtual private network (VPN).

10. The method of claim 1, wherein a generated insight of the generated insights is generated utilizing natural language representation.

11. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process for determining abnormal configuration of network objects deployed in a cloud computing environment, the process comprising:

collecting network object data on a plurality of network objects deployed in the cloud computing environment; constructing a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment; determining relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects; analyzing the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects; and tagging network objects in the network graph for which the insight is generated.

12. A system for determining abnormal configuration of network objects deployed in a cloud computing environment, comprising:

a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

collect network object data on a plurality of network objects deployed in the cloud computing environment; construct a network graph based on the collected network object data, wherein the network graph includes a visual representation of network objects identified in the cloud computing environment; determine relationships between the identified network objects in the network graph, wherein the determined relationships between the identified network objects includes descriptions of connections between the identified network objects; analyze the network graph and the determined relationships to generate insights, wherein the generated insights include at least a list of abnormal connections between the identified network objects; and tagging network objects in the network graph for which the insight is generated.

13. The system of claim 12, wherein the cloud computing environment includes a plurality of different cloud computing platforms.

14. The system of claim 13, wherein the system is further configured to:

connect through an application programming interface (API) each of the plurality of different cloud computing platforms to collect network object data of network objects deployed in the respective cloud computing platform.

15. The system of claim 12, wherein the system is further configured to:

detect any one of: a rare network configuration, a novel network configuration, a rare network event, a novel network event, a connection of a new device to a network, an unauthorized connection, a re-connection

of a user device to a subnet which the user is not permitted to access, a connection demonstrating anomalous behavior or misconfiguration, and a connection demonstrating spikes of network activity.

16. The system of claim 12, wherein the system is further configured to:

determine impermissible relationships between the network objects.

17. The system of claim 12, wherein the system is further configured to:

determine the relationships using a static analytic method.

18. The system of claim 12, wherein the system is further configured to:

determine the relationships using at least one of: observational methods, and active logging methods.

19. The system of claim 12, wherein the system is further configured to:

adding visual representations of the determined relationships to the visual representations of the network graph.

20. The system of claim 12, wherein each of the plurality of network objects includes any one of: a virtual network, a firewall, a network interface card, a proxy, a gateway, a software container, container, a management object, a virtual machine, a subnet, a hub, a virtual private network (VPN).

21. The system of claim 12, wherein a generated insight of the generated insights is generated utilizing natural language representation.

* * * * *

EXHIBIT B



(12) **United States Patent**
Shemesh et al.

(10) **Patent No.:** **US 11,929,896 B1**
(45) **Date of Patent:** **Mar. 12, 2024**

(54) **SYSTEM AND METHOD FOR GENERATION OF UNIFIED GRAPH MODELS FOR NETWORK ENTITIES**

(71) Applicant: **Wiz, Inc.**, Palo Alto, CA (US)
(72) Inventors: **Daniel Hershko Shemesh**, Netanya (IL); **Liran Moysi**, Kfar Saba (IL); **Roy Reznik**, Tel Aviv (IL); **Shai Keren**, Ramat Gan (IL)

(73) Assignee: **WIZ, INC.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **17/161,190**

(22) Filed: **Jan. 28, 2021**

(51) **Int. Cl.**
H04L 43/045 (2022.01)
G06F 16/901 (2019.01)
H04L 43/08 (2022.01)
H04L 67/10 (2022.01)

(52) **U.S. Cl.**
CPC **H04L 43/045** (2013.01); **G06F 16/9024** (2019.01); **H04L 43/08** (2013.01); **H04L 67/10** (2013.01)

(58) **Field of Classification Search**
CPC H04L 43/045; H04L 43/08; H04L 67/10; G06F 16/9024
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,586,254 A * 12/1996 Kondo H04L 41/0846 714/25
5,819,028 A * 10/1998 Manghirmalani H04L 41/046 709/224

5,926,462 A * 7/1999 Schenkel H04L 43/00 370/254
10,009,251 B1 * 6/2018 Koster H04L 67/141
10,326,673 B2 * 6/2019 Kulshreshtha H04L 63/1425
10,862,928 B1 * 12/2020 Badawy H04L 63/104
2002/0147715 A1 * 10/2002 Beyer H04L 41/022 707/999.005
2004/0210654 A1 * 10/2004 Hrastar H04L 63/1408 709/224
2007/0147269 A1 * 6/2007 Ettl H04W 16/18 370/254
2010/0223295 A1 * 9/2010 Stanley G06F 16/284 707/794
2010/0241698 A1 * 9/2010 Hillerbrand G06F 16/13 709/203
2013/0219009 A1 * 8/2013 Bheemarajaiah H04L 67/26 709/217
2014/0130008 A1 * 5/2014 Amulu G06F 8/35 717/105

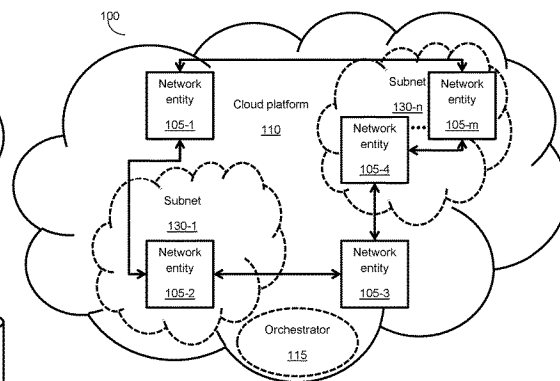
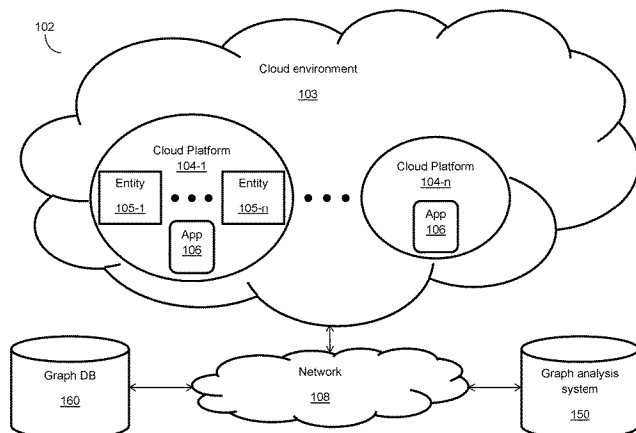
(Continued)

Primary Examiner — Michael Won
(74) Attorney, Agent, or Firm — M&B IP Analysts, LLC

(57) **ABSTRACT**

A system and method for generation of unified graph models for network entities are provided. The method includes collecting, for at least one network entity of a plurality of network entities, at least one network entity data feature, wherein the at least one network entity data feature is a network entity property; genericizing the collected at least one network entity; generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of network entities and relations between the network entities of the plurality of network entities; and storing the generated at least a network graph.

29 Claims, 8 Drawing Sheets



US 11,929,896 B1

Page 2

(56)

References Cited

U.S. PATENT DOCUMENTS

2016/0019033	A1*	1/2016	Ebner	G06F 8/38 717/104
2016/0105350	A1*	4/2016	Greifeneder	H04L 41/046 709/224
2016/0219117	A1*	7/2016	Marlatt	H04L 67/322
2016/0352766	A1*	12/2016	Flacher	H04L 67/10
2017/0127427	A1*	5/2017	Claridge	H04W 4/025
2017/0140040	A1*	5/2017	Gottemukkala	G06Q 10/067
2018/0024981	A1	1/2018	Xia et al.	
2018/0063193	A1*	3/2018	Chandrashekhar ...	H04L 63/029
2018/0261001	A1*	9/2018	Wang	G06T 7/90
2019/0258756	A1*	8/2019	Minwalla	G06N 20/20
2019/0258973	A1*	8/2019	Prabhu	G06Q 10/0637
2019/0289038	A1*	9/2019	Li	H04L 63/205
2020/0050689	A1*	2/2020	Tal	G06F 16/23
2020/0167642	A1*	5/2020	Dhurandhar	G06N 20/00
2020/0236038	A1*	7/2020	Liu	H04L 45/02
2020/0285977	A1*	9/2020	Brebner	G06N 5/022
2020/0320130	A1*	10/2020	Korpman	G06F 16/9027
2020/0322227	A1	10/2020	Janakiraman	
2020/0336376	A1*	10/2020	Mahdi	G06N 20/20
2020/0366759	A1*	11/2020	Sinha	G06F 9/5072
2021/0149858	A1*	5/2021	Xia	G06F 16/2282
2021/0174280	A1*	6/2021	Ratnapuri	G06Q 10/06315
2021/0342685	A1*	11/2021	Dhurandhar	G06N 3/042

* cited by examiner

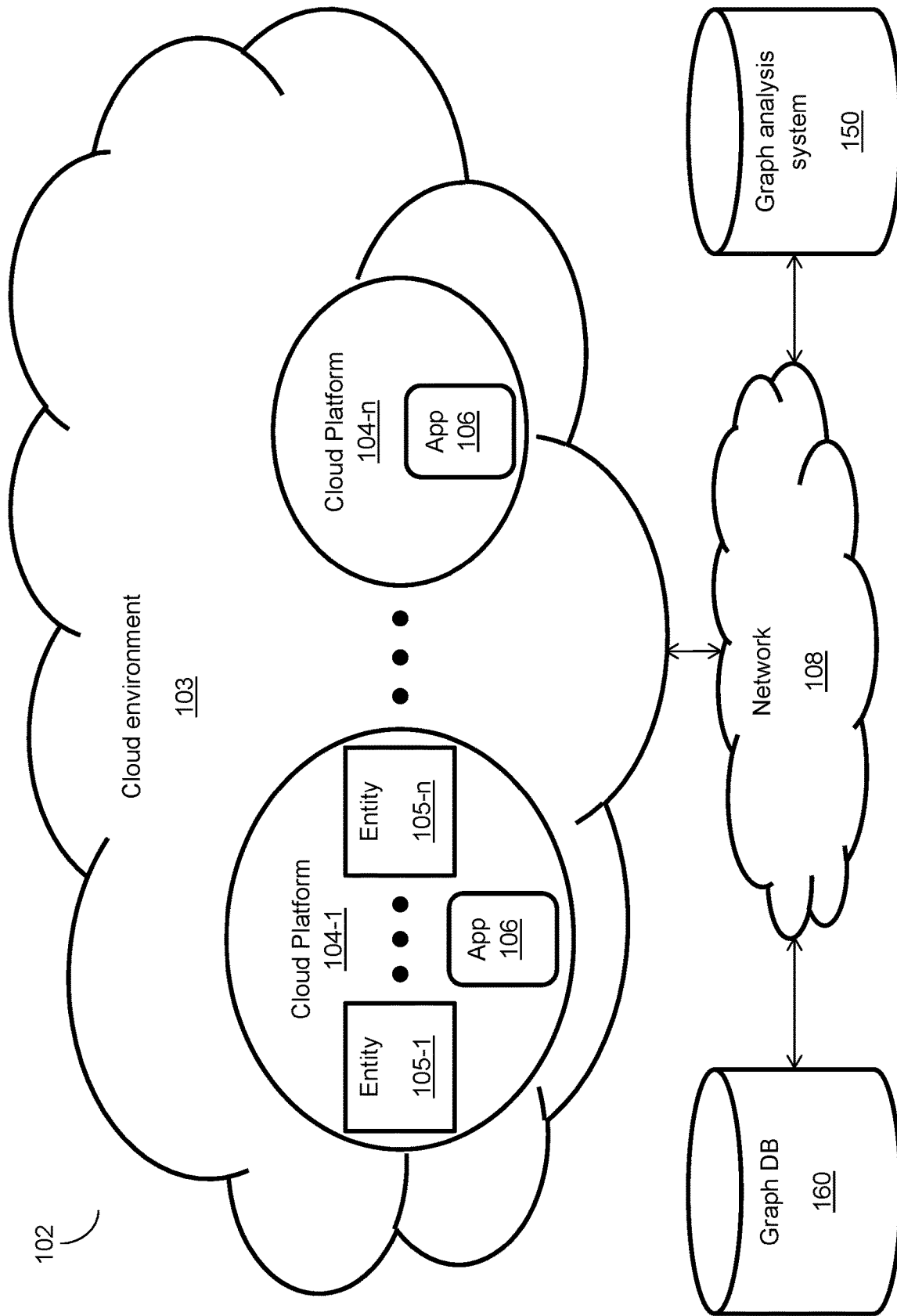


FIG. 1A

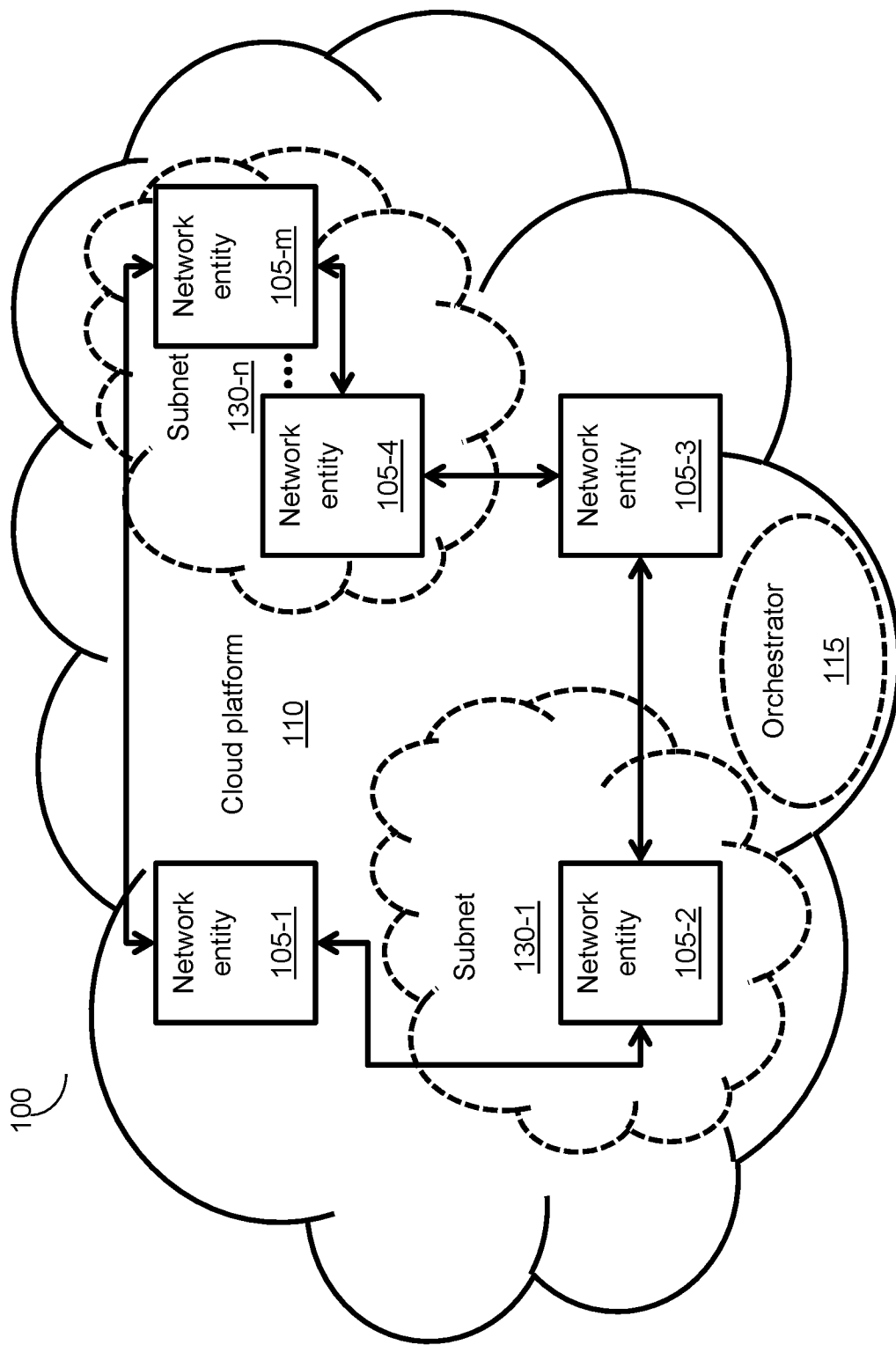


FIG. 1B

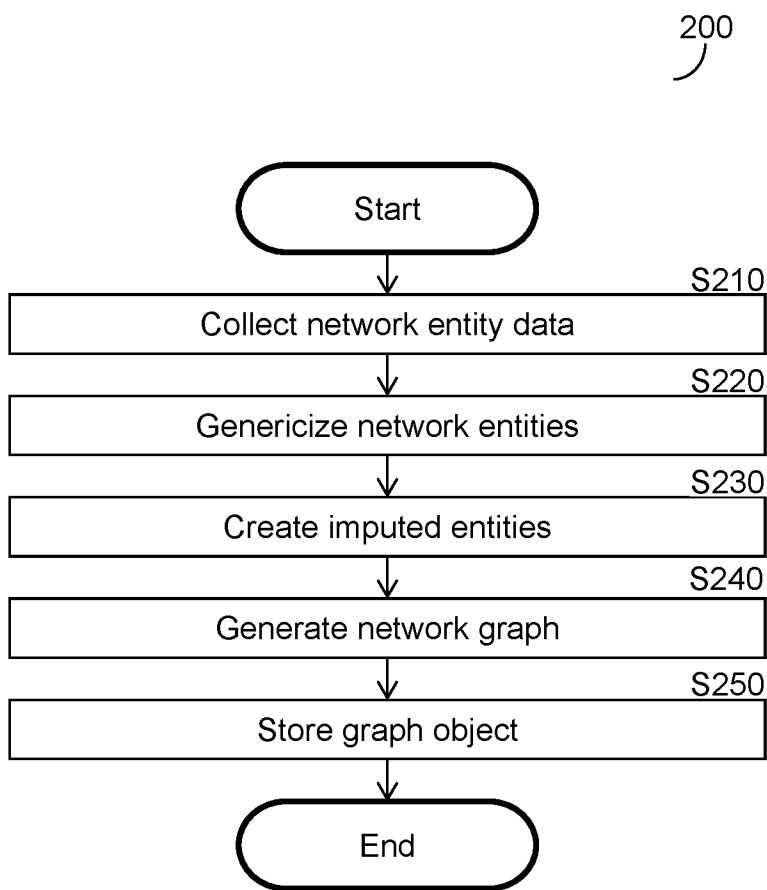


FIG. 2

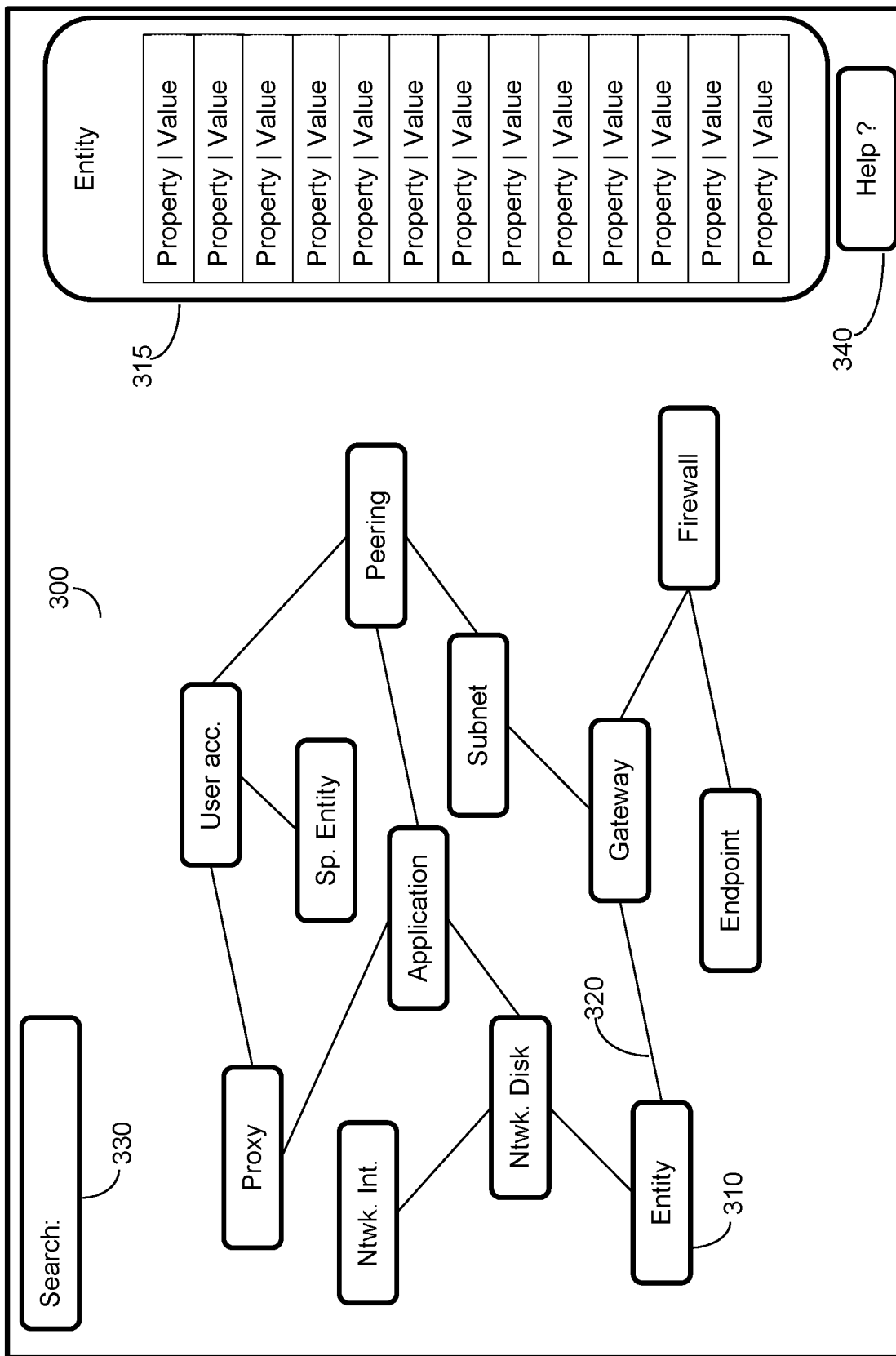


FIG. 3

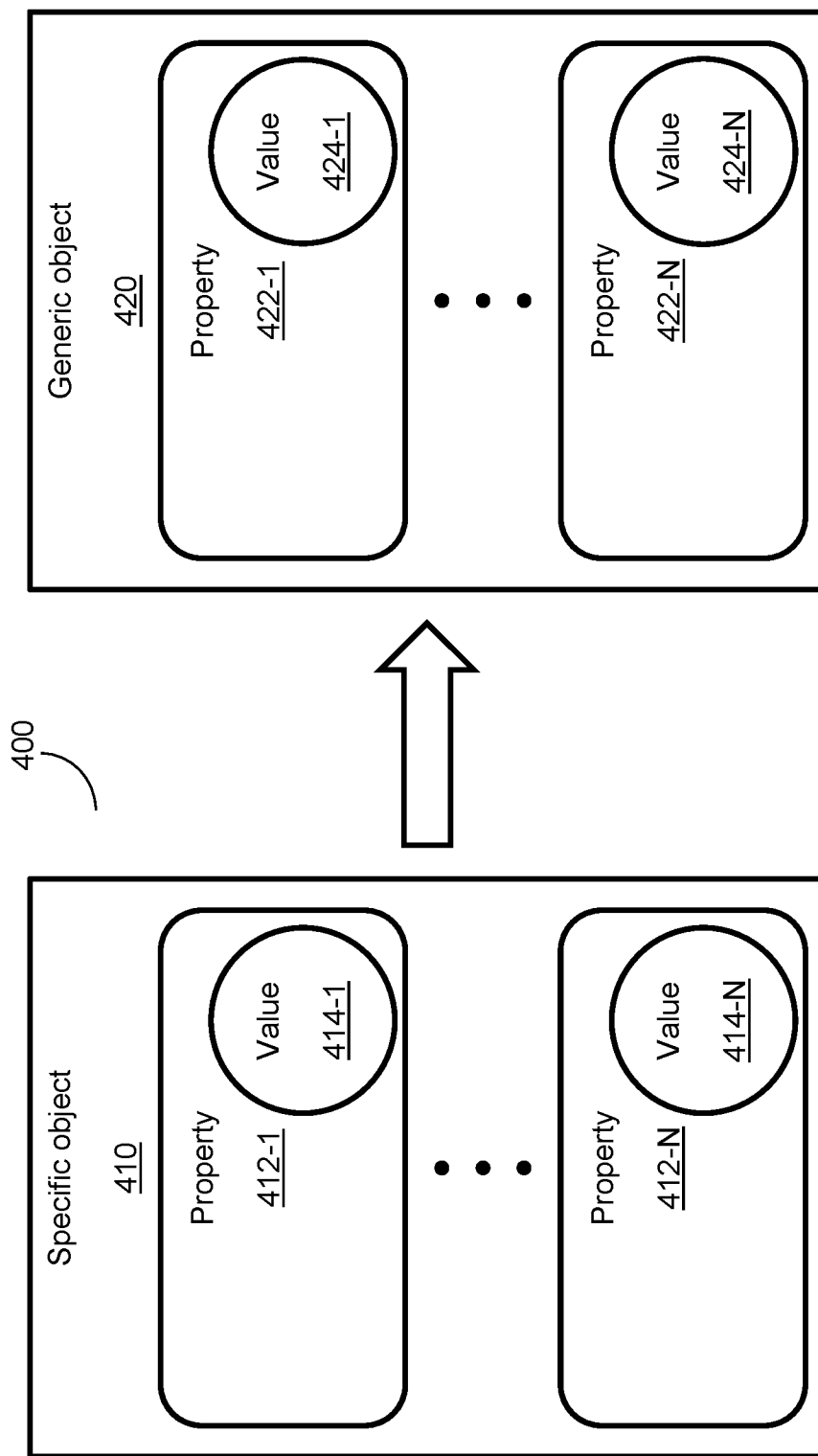


FIG. 4

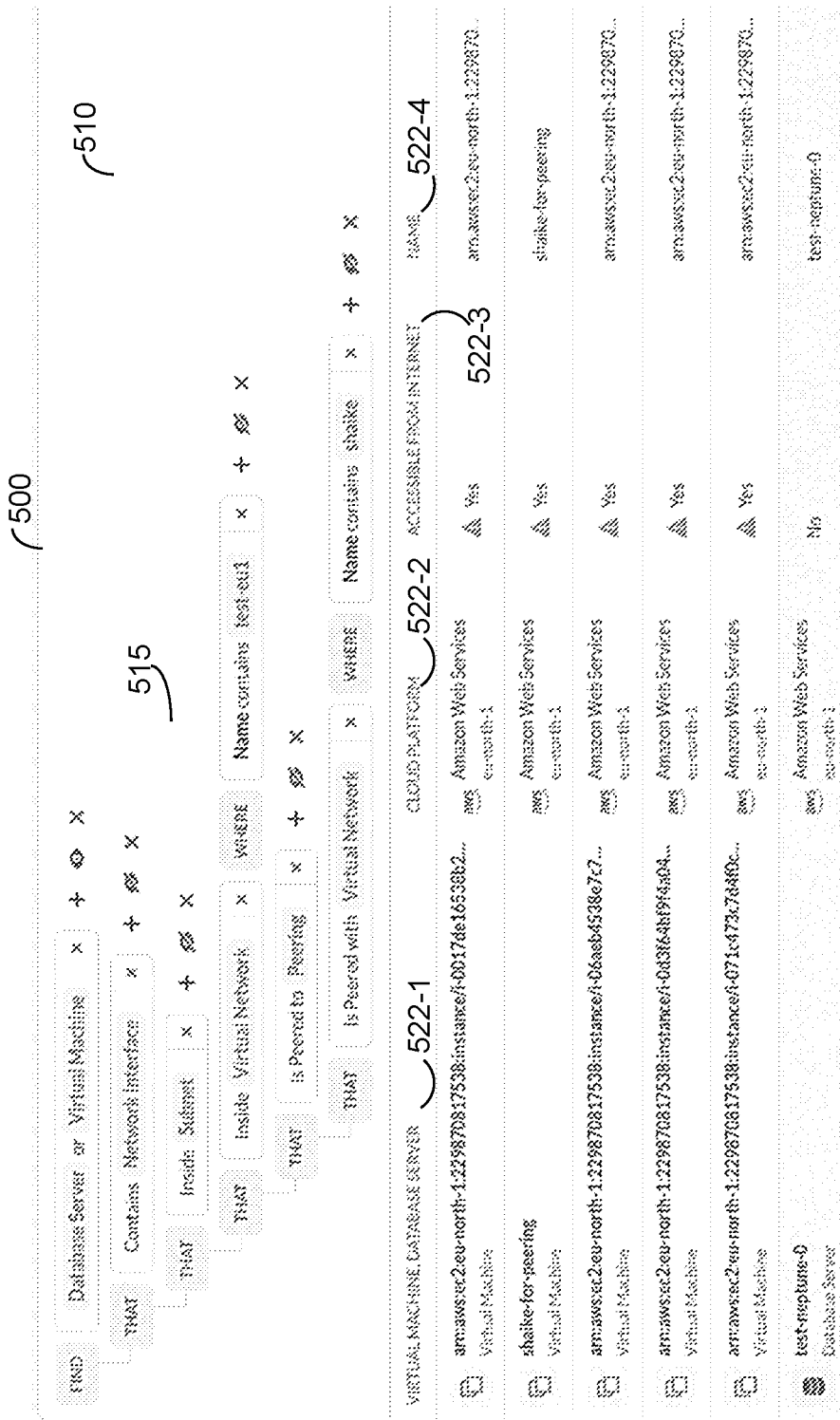


FIG. 5A



FIG. 5B

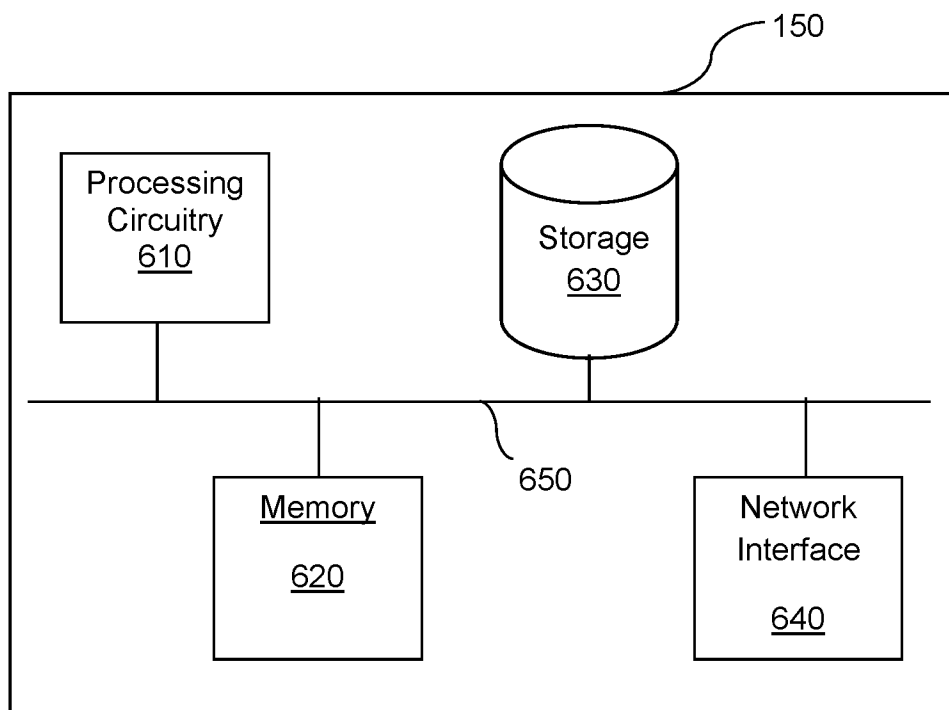


FIG. 6

US 11,929,896 B1

1

SYSTEM AND METHOD FOR GENERATION OF UNIFIED GRAPH MODELS FOR NETWORK ENTITIES

TECHNICAL FIELD

The present disclosure relates generally to cybersecurity, and, in particular, to systems and methods for automated generation of unified graph models for network entities.

BACKGROUND

As users, including large organizations such as businesses and governments, increase deployment of large-scale computing systems for data management, application processing, and other purposes, the same users may seek to better understand the technologies included in such systems, as well as the vulnerabilities thereof. As large-scale computing systems, such as configurations implemented through cloud service providers, such as Microsoft Azure® and Amazon AWS®, may include large numbers of components, devices, systems, and the like, including various types and deployments of each, and as connections between such components, devices, and systems may be similarly numerous and varied, users, administrators, and other interested parties may seek solutions providing for simple, unified understandings of the technologies included in such computing environments.

Current solutions to the analysis of the various systems, services, and the like, which may be included in a computing environment, include solutions directed to the analysis of live or recorded network traffic. Such traffic analysis systems provide for identification of activity within a network, but may fail to represent all components of a computing environment, such as those components which are not actively engaged in network communications during a traffic sampling period. Further, traffic-independent solutions, such as solutions directed to the identification of environment components, and connections therebetween, may provide for a more comprehensive understanding of the components and structure of a network or environment, but may fail to provide for generation of simple, unified views, particularly where networks, environments, and the like, include multiple, similar network components requiring separate analysis and representation. In addition, solutions directed to the identification of environment components and connections may provide for the representation of such environment features, but fail to provide such a representation in a graph format, preventing the execution of graph-specific commands and queries across such datasets.

Where a user's computing environment includes multiple, similar network components, such as dissimilar objects providing similar functionalities, current traffic-independent solutions may fail to provide concise, efficient rendition of such objects in a simplified network view. As a computing environment may include objects drawn from multiple sources which are configured to provide similar functionalities, such as native firewall objects included in Azure® and AWS® configurations, current traffic-independent network analysis systems may require separate analysis and representation of each object, reducing analysis efficiency and increasing cost. Further, where such objects may be "implicit" from a network analysis perspective, such as various traffic management devices incorporated into a cloud platform host system, rather than a user's platform deployment, such implicit objects may not be exposed to an analytic system for representation in a simplified view. As a

2

result, current traffic-independent solutions may fail to provide for the analysis and representation of such objects, where analysis and representation may be necessary to provide a thorough understanding of network or environment components and structures.

Further, in addition to such deficiencies of current traffic-independent solutions, such solutions fail to provide for the integrated representation and analysis of both "explicit" and "implicit" objects, as well as objects included in different layers of a computing environment. As a computing environment may include both "explicit" and "implicit" objects, such as, for example, infrastructure as a service (IaaS) deployments which include visible VMs, network interfaces, storage modules, and the like, and platform as a service (PaaS) deployments which include the same objects at the platform level, rendering such objects "implicit," analysis of such environments may require analysis of both "explicit" and "implicit" objects. However, current traffic-independent solutions fail to provide for the analysis of both types of objects, preventing the generation of network representations which provide for the inclusion and analysis of all relevant objects. Further, where an organization's computing environment includes objects and entities dispersed across multiple layers, current traffic-independent solutions fail to provide for the representation of all objects or entities in a single view or depiction, where such a unified view may provide for enhanced network analysis and complex querying, as well as other functionalities.

It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the terms "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

Certain embodiments disclosed herein include a method for generation of unified graph models for network entities. The method comprises collecting, for at least one network entity of a plurality of network entities, at least one network entity data feature, wherein the at least one network entity data feature is a network entity property; genericizing the collected at least one network entity; generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of network entities and relations between the network entities of the plurality of network entities; and storing the generated at least a network graph.

Certain embodiments disclosed herein also include a non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process for generation of unified graph models for network entities, the process comprising: collecting, for at least one network entity of a plurality of network entities, at least one network entity data feature, wherein the at least one network entity data feature is a network entity property;

US 11,929,896 B1

3

genericizing the collected at least one network entity; generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of network entities and relations between the network entities of the plurality of network entities; and storing the generated at least a network graph.

In addition, certain embodiments disclosed herein include a system for generation of unified graph models for network entities. The system comprises: a processing circuitry; and a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: collect, for at least one network entity of a plurality of network entities, at least one network entity data feature, wherein the at least one network entity data feature is a network entity property; genericize the collected at least one network entity; generate at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of network entities and relations between the network entities of the plurality of network entities; and store the generated at least a network graph.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. 1A is a diagram of a cloud environment utilized to describe the various embodiments.

FIG. 1B is a network diagram depicting a network system and various associated network and external entities utilized to describe the various embodiments.

FIG. 2 is a flowchart depicting a method for generating unified graph models for network entities, according to an embodiment.

FIG. 3 is a network graph visualization, according to an embodiment.

FIG. 4 is an illustration of the genericization of a network entity, utilized to describe the various embodiments.

FIG. 5A is an interactive graph analysis platform, provided through a user interface (UI), according to an embodiment.

FIG. 5B is an interactive graph analysis platform, provided through a user interface (UI), according to an embodiment.

FIG. 6 is a hardware block diagram depicting a graph analysis system, according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The systems and methods described herein may be applicable to various systems, devices, networks, environments, layers, and the like, as well as cross-connections or multi-

4

entity connections as may be established therebetween. The disclosed systems and methods may be applicable to provide support for various network features including, without limitation, application-layer communications, cloud-native constructs, cross-cloud and Kubernetes-to-cloud communications, third-party features, such as third-party containers and entities, container-management systems, such as Kubernetes, as may be virtualized as cloud entities, and the like, as well as any combination thereof.

Further, the systems and methods described herein may be applicable to various network graph models, and applications thereof. The systems and methods described herein may be relevant to unified graph models configured to represent network components from deployments and configurations including, without limitation infrastructure as a service (IaaS) deployments, platform as a service (PaaS) deployments, Kubernetes(R) deployments, multi-cloud configurations, other, like, deployments and configurations, and any combination thereof, including applications relevant to the modeling of cloud elements including, without limitation transit gateways, shared virtual private clients (VPCs), private links, and the like, as well as any combination thereof. Such relevant network graph models may be unified across all cloud environments, computing devices, services, and the like, including, without limitation, AWS® lambda services which are exposed to users, MongoDB® instances running on a container or PaaS database, and the like, as well as any combination thereof. Further, such network graph models may be configured to represent cloud implementations including, without limitation, security-group-based firewall rules, tag-based firewall rules, and the like, as well as any combination thereof. Such network graph models may be configured to include structures and features relevant to the maximization of model efficiency, the maximization of model-element visibility, and the implementation of various query languages, where such structures and features may include, without limitation, minimization of edges and nodes in a model, elimination of redundant edges, and the like, as well as any combination thereof.

FIG. 1A is an example diagram of a cloud environment **103** utilized to describe the various disclosed embodiments. A cloud environment **103** represents an organization's cloud-based resources, and the various connections between such resources. The cloud environment **103** may include a number of cloud computing platforms, **104-1** through **104-n** (hereinafter, "cloud platforms" **104** or "cloud platform" **104**), where a cloud platform may include multiple network entities, **105-1** through **105-n** (hereinafter, "network entities" **105** or "network entity" **105**), one or more applications (collectively referred to as applications or apps **106**), and the like, as well as any combination thereof. Further, the cloud environment **103** may be configured to connect, via a network **108**, with a graph analysis system **150**, and a graph database **160**, for one or more purposes including, without limitation, those described hereinbelow. As is applicable to the cloud platforms **104** and network entities **105**, "n" is an integer having a value greater than or equal to two. Further, it may be understood that, while a single configuration of a cloud environment **103** is shown for purposes of simplicity, a cloud environment **103** may include various combinations of platforms **104**, entities **105**, applications **106**, and the like, as well as any combination thereof, without loss of generality or departure from the scope of the disclosure.

A cloud platform **104** is a platform, architecture, or other, like, configuration providing for connectivity of the various entities **105**, applications **106**, and other, like, elements included in a cloud platform **104**, as well as the execution of

US 11,929,896 B1

5

various processes, instructions, and the like. A cloud platform **104** may be a commercially-available cloud system, provided on a service basis, such as, as examples and without limitation, Amazon AWS®, Microsoft Azure®, and the like. A cloud platform **104** may be a private cloud, a public cloud, a hybrid cloud, and the like. In addition, a cloud platform **104** may include, without limitation, container orchestration or management systems or platforms such as, as an example and without limitation, a Kubernetes deployment, and the like, as well as any combination thereof.

A cloud platform **104** may be implemented as a physical network of discrete, interconnected entities, and the like, a virtual network, providing for interconnection of various virtual systems and devices, as well as a hybrid physical-virtual network, including both physical and virtualized components. A cloud platform **104** may be, or may replicate or otherwise simulate or emulate, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof. Further, a cloud platform **104** may include one or more subnets, such as the subnets, **130**, of FIG. 1B, below, wherein each subnet may be configured to serve as a cloud platform **104** for the various network entities which may be included in the subnet, while retaining the connectivity and functionalities provided by the cloud platform **104**.

Network entities **105**, as may be included in a cloud platform **104**, are entities, systems, devices, components, applications, objects, and the like, configured to operate within the cloud platform **104** and provide various functionalities therein. Specifically, the network entities **105** may be, as examples without limitation, entities configured to process data, send data, or receive data, as well as entities configured to provide various other functionalities, and any combination thereof. The network entities **105** may be configured to connect with various other network entities **105**, various external entities, and the like, as well as any combination thereof, for purposes including, without limitation, sending data, receiving data, monitoring data transmissions, monitoring network status and activity, and the like, as well as any combination thereof.

Examples of network entities **105** include, without limitation, entities providing support for application-layer communications and systems, including application-layer communications and systems relevant to layer seven of the open systems interconnection (OSI) model. Further examples of network entities **105** may include cloud-native constructs, such as private endpoints, transit gateways, tag-based rule-sets and entities configured to apply such rules, Kubernetes Istio and Calico services and applications, and the like. In addition, examples of network entities **105** may include, without limitation, third-party containers and images, such as Nginx, web-access firewall (WAF), and firewall implementations, multi-entity or cross-entity connections, such as cross-cloud connections and Kubernetes-to-cloud connections, as well as container managers, such as Kubernetes, and connections therewith. It may also be understood that network entities **105** may include other entities similar to those described hereinabove, as well as any combination thereof. As another example, network entities **105** may include virtual entities, devices, and the like, to process layer-7 (application layer) traffic, such as entities relevant to Amazon AWS® layer seven services and applications, Amazon Load Balancer® (ALB) layer seven services and applications, Kubernetes ingress, and the like.

6

The network entities **105** may be configured to include one or more communication ports, where the included communication ports provide for connection of various entities according to one or more protocols, and at different communication layers of the OSI model.

In an example configuration, the network entities **105** are virtual entities or instances of systems, devices, or components, including virtual systems, devices, or components, or any combination thereof. Examples of network entities **105** include, without limitation, virtual networks, firewalls, network interface cards, proxies, gateways, containers, container management objects, virtual machines, subnets **130**, hubs, virtual private networks (VPNs), and the like, as well as any combination thereof.

The applications **106** (or cloud applications), as may be executed in one or more cloud platforms **104**, are services, processes, and the like, configured to provide one or more functionalities by execution of various commands and instructions. The applications **106** may be part of a software project of an enterprise or organization. The applications **106** may interact or communicate with other applications **106**, regardless of the platform **104** in which the applications **106** are deployed. Examples of applications **106** include, without limitation, databases, serverless functions, web servers, buckets, and the like, as well as any combination thereof. It should be understood that multiple instances of a single application **106** may be both present and executed in multiple cloud platforms **104**, including multiple cloud platforms **104** of the same cloud environment **103**, without loss of generality or departure from the scope of the disclosure.

The network **108** is a communication system providing for the connection of the cloud environment **103**, and its various components and sub-parts, with a graph analysis system **150**, and a graph database **160**, as well as other, like, systems, devices, and components, and any combination thereof. The network **108** may be a physical network, a virtual network, as well as a hybrid physical-virtual network, including both physical and virtualized components. The network **108** may be, as examples, and without limitation, a local area network, a wide area network, the Internet, the World-Wide Web (WWW), and the like, as well as any combination thereof.

The graph analysis system **150** is a system, device, or component, configured to perform the various disclosed embodiments of graph-related functionalities. Specifically, the system **150** may provide functions including, without limitation network analysis, traffic analysis, entity querying, graph generation, and the like, as well as any combination thereof. The graph analysis system **150** may be configured to execute one or more instructions, methods, processes, and the like, including, without limitation, the process described with respect to FIG. 2, other, like, processes, and any combination thereof.

The graph analysis system **150** may be configured as a physical system, device, or component, as a virtual system, device, or component, or in a hybrid physical-virtual configuration. A detailed description of a graph analysis system, **150**, according to an embodiment, is provided with respect to FIG. 6, below. It may be understood that, while the graph analysis system **150** is depicted in FIG. 1A as a discrete element external to the cloud environment **103**, the graph analysis system **150** may be included within any of the various elements of the network system **102**, including the cloud environment **103**, the various cloud platforms **104**, and subparts thereof, and the network **108**, without loss of generality or departure from the scope of the disclosure.

US 11,929,896 B1

7

The graph database (graph DB) **160** is a storage or memory component, device, system, or the like, configured to provide one or more functionalities relevant to storage of graph-related data. The graph DB **160** may be configured to store graph-related data features of one or more types or formats, including, without limitation, raw data, graphs, graph edges, graph vertices, graph schemas, and the like, as well as any combination thereof, including those types or formats described hereinbelow.

Graphs, as may be included in the graph DB **160**, are data features including one or more graph vertices, where such graph vertices may be variously interconnected by one or more graph edges. Graphs may be configured to provide for one or more representations of various data sets, including, without limitation, presentation of network data according to one or more graph schemas, as described hereinbelow. As an example, a graph relevant to the description of a collection of interconnected network entities may include one or more graph vertices, where each graph vertex corresponds with a network entity, and graph edges between such vertices, the edges corresponding with connections between the various network entities. Graphs, and elements thereof, may be generated based on one or more data sets, including during the execution of a graph generation process, such as is described with respect to FIG. 2, below.

The graph DB **160** may be configured to store one or more graphs, graph, related data features, and the like, as well as any combination thereof. Graphs, as may be stored in the graph DB **160**, may be configured to include one or more functional attributes, including, without limitation, directionality, labeling, and the like, where such functional attributes may provide for the execution or enhancement of one or more processes or methods which would be inapplicable to a graph not including such functional attributes. A graph including directionality may be configured to include connection between graph nodes or vertices, via graph edges, as described herein, where the edges connecting such vertices may be uni-directional or bi-directional, providing for enhanced analysis of network entity structures and relationships. Further, a graph configured to include labeling functionality may be configured to provide for the labeling of graph vertices, graph edges, or both, with one or more labels describing the various properties of the labeled vertices or edges. As an example, a graph vertex representing a virtual machine (VM) may be configured to include a “name” label, describing a name property of the VM. Further, the VM may be configured to run a container entity, where the container entity, as represented in the graph, may be respectively labeled. In addition, according to the same example, the connection, or edge, between the vertices representing the VM and the container entity may be uni-directional and may be labeled as a “run” edge, providing for analysis of the relationship between the vertices, the direction of the relationship, and the type of the relationship.

The graph DB **160** may be configured as a physical system, device, or component, as a virtual system, device, or component, or in a hybrid physical-virtual configuration. It may be understood that, while the graph DB **160** is depicted in FIG. 1A as a discrete element external to the cloud environment **103**, the graph DB **160** may be included within any of the various elements of the network system **102**, including the cloud environment **103**, the various cloud platforms **104**, and subparts thereof, and the network **108**, without loss of generality or departure from the scope of the disclosure. Further, it may be understood that the graph DB **160** may be directly connected to, or realized as a compo-

8

nent of, the graph analysis system **150**, without loss of generality or departure from the scope of the disclosure.

FIG. 1B is an example diagram depicting a network system **100** and various associated network and external entities utilized to describe the various embodiments. The depicted network system **100** includes a cloud platform **110**, where the cloud platform **110** may be a cloud platform similar or identical to a cloud platform, **104**, of FIG. 1A, above. The cloud platform **110** includes various subnets, **130-1** through **130-*n*** (hereinafter, “subnets” **130** or “subnet” **130**), and various network entities, **105-1** through **105-*m*** (hereinafter, “network entities” **105** or “network entity” **105**). As is applicable to the subnets **130**, “*n*” is an integer having a value greater than or equal to two. Further, as is applicable to the network entities **105**, “*m*” is an integer having a value greater than or equal to five. In addition, while the network system **100** of FIG. 1B includes certain elements and combinations of elements, as well as connections therebetween, it may be understood that the depiction is provided for illustrative purposes, and that other, like, elements, combinations of elements, and connections therebetween, may be implemented without loss of generality or departure from the scope of the disclosure. Other, like, network systems **100** may further include multiple cloud platforms **110**, including variously-interconnected cloud platforms **110**, and other, like, variations and configurations, without loss of generality or departure from the scope of the disclosure.

The cloud platform **110** may be configured to include an orchestrator **115**. The orchestrator **115** is configured to provide for management of the cloud platform **110**. The orchestrator **115** may be configured to provide one or more functionalities including, without limitation, monitoring of elements or components of the cloud platform **110**, logging and reporting data relating to the cloud platform **110**, managing cloud platform **110** updates and maintenance, generating cloud platform **110** alerts, as well as other, like, functionalities, and any combination thereof. The orchestrator **115** may be configured to report one or more data features related to the cloud platform **110**, such as may be requested or collected during the execution of graph generation processes, such as those described hereinbelow.

The network entities **105** are network entities similar or identical to those network entities, **105**, of FIG. 1A, above. As described with respect to FIG. 1A, the network entities **105** are virtual entities or instances of systems, devices, or components, including virtual systems, devices, or components, or any combination thereof. Examples of network entities **105** include, without limitation, virtual networks, firewalls, network interface cards, proxies, gateways, containers, container management entities, virtual machines, subnets **130**, hubs, virtual private networks (VPNs), peering connections, load balancers, route tables, and the like, as well as any combination thereof.

External objects, as may be adjacent or relevant to a cloud platform **110**, are entities similar or identical to the network entities **105**. The external entities may be configured to communicate with one or more network entities **105**, with other, various, external entities, and the like, as well as any combination thereof.

FIG. 2 is an example flowchart **200** depicting a method for generating unified graph models for network entities, according to an embodiment.

At S210, network entity data is collected. Network entity data, as may be collected at S210, is data describing one or more entities included in a network, cloud, environment, or other, like, deployment. Network entity data may include

data describing the various network entities including, as examples and without limitation, properties, names, network addresses, permissions, properties, configurations, other, like, network entity data features, and any combination thereof. Network entity data may be collected from one or more network entities, such as those described in detail above.

Network entity data may be collected at S210 via one or more processes including, without limitation, application of various network entity scanning applications or processes, application of network entity application programming interface (API) commands, calls, and the like, collection from one or more network, cloud, or environment orchestrators, including orchestrators similar or identical to those described hereinabove, other, like, processes, and any combination thereof.

At S220, network entities are genericized based on the collected network entity data. Generic network entities, which may be created at S220, are network entities which are configured to include one or more standard properties. Such standard properties may be applicable to all genericized network entities of the same type. As an example, a genericized network entity may be a generic proxy object, and, according to the same example, separate, differently configured specific proxy objects, such as an Azure®-native proxy object and an AWS®-native proxy object, may be genericized at S220 for representation as generic proxy objects.

In an embodiment, S220 may include the generation of new generic network entities, and the conversion of existing specific network entities into generic network entities. Generic network entities may include one or more properties relevant to the type, class, or other, like, category of network entity. Relevant properties may include properties common to specific network entities configured to achieve the same or similar functionalities. Common properties may include one or more properties relevant to the execution or provision of such functionalities.

Generic entity properties may be defined according to one or more property configurations, including, without limitation, pre-defined properties, user-defined properties, properties defined based on sampled specific entities, other, like, bases, and any combination thereof. Further, generic entity properties may be based on analysis of runtime data features such as, without limitation, lists of entities to which a given entity connects during routine operation, as may be determined by processes including, without limitation, analysis of entity codebases or resources, analysis of runtime logs or reports, and the like, as well as any combination thereof. According to the example described hereinabove, a generic proxy object may be configured to include properties relevant to the execution of proxy functionalities, and common to various specific property objects included in separate deployments, such as, as examples and without limitation, proxy rules, port configurations, and the like, as well as any combination thereof.

Genericization at S220 includes, in an embodiment, converting specific objects into generic objects, or generating new generic objects based on specific objects, or a combination thereof. In a further embodiment, S220 may include identifying types, classes, or other, like categories of specific entities, using, for example, comparison of specific entity properties, configurations, and the like, with one or more dictionaries, repositories, or other, like, sources of entity type information. Further, the new generic entities may be entities matching the type, category, or the like, of the identified specific entity or entities, where the new generic

entities may include one or more properties of the corresponding specific entities. The genericization of network entities, as at S220, is further described with respect to FIG. 4, below.

As an example, an AWS®-native proxy objects, network load balancers, application load balancers, application programming interface (API) gateways, and the like, may be, at S220, genericized as generic proxy objects. According to the same example, the AWS®-native specific proxy object may be identified as a proxy-type entity based on the inclusion of data features matching proxy rules within the object's codebase, resource files, and the like. Further, the genericized proxy object, created at S220, may include one or more properties of the AWS®-native proxy object, such as the AWS®-native object's specific proxy rules.

Further, examples of relevant types of generic entities include, without limitation, entities configured to apply routing rules, network interfaces, entities configured to apply security rules, subnets and virtual networks, gateways, and the like, as well as any combination thereof. According to the same example, generic entities representing entities configured to apply routing rules may be generated, or converted from corresponding specific entities, or both, at S220, providing for the genericization of separate, similar entities configured to apply such rules, where the separate entities may include different routing rules or tables, providing for the genericization of such entities into a unified format and, subsequently, providing for the execution of unified queries or commands across such genericized entities.

Likewise, according to the same example, generic entities representing network interfaces may be generated, or converted from corresponding specific entities, or both, at S220, providing for genericization of separate, similar network interfaces, thus providing for encapsulation of the interfaces' network properties. Similarly, according to the same example, generic entities representing entities configured to apply security rules, such as security group rules, network access control lists (ACLs), database ACLs, and the like, may be generated, or converted from corresponding specific entities, or both, at S220, providing for the genericization of such specific entities in a unified format.

In addition, according to the same example, generic entities representing subnets and virtual networks, which may include routing rules and security rules, and which may be applicable to the description of network hierarchies, may be generated, or converted from corresponding specific entities, or both, at S220, providing for genericization of such entities and, subsequently, execution of one or more unified queries across such entities, including graph traversal commands relevant thereto. Further, according to the same example, generic entities representing gateways, such as internet gateways, relevant to searching the internet connectivity of objects, transit gateways, including complex routing rules and attachments, may be generated, or converted from corresponding specific entities, or both, at S220, providing for genericization of such entities.

At S230, imputed entities are created. Imputed entities are generic entities similar or identical to those described with respect to S220, above, which may be constructed to provide representation of network entities which are integrated into host platforms, or network entities which are shielded from, or not otherwise exposed to, a system configured to execute network analysis processes and methods, including the method described with respect to FIG. 2, where such a system may be, without limitation, the graph analysis system 150 of FIG. 1A. Imputed entities may be of various types,

US 11,929,896 B1

11

classes, or categories corresponding to the types, classes, and categories of generic entities, such as the entities generi- cized at S220, and may include one or more properties corresponding to such types, classes, or categories.

In an embodiment, imputed entities may be created using identification of platform or environment functionalities which correspond with functionalities of generic entities. Creation of imputed entities using identification of function- alities may include identification of functionalities which are not provided by any network entities, such as the network entities from which data is collected at S210. Following identification of functionalities not provided by any network entities, creation of imputed entities may include creation of a generic imputed entity with types, properties, and the like, matching an entity configured to provide the described functionality, in addition to other, like, means, and any combination thereof. Where imputed entity creation includes identification of platform or environment function- alities corresponding to entities not included in data collec- tion at S210, such identification may include application of one or more processes or methods including, without limita- tion, identification of functionalities by comparison of platform or environment data flows with data flows relevant to simulated platforms or environments including only the entities included in data collection, identification of func- tionalities by analysis of platform or environment host or provider feature descriptions, identification by other, like, means, and any combination thereof.

As an example, where a cloud platform provider includes a load balancing functionality in all platform deployments by inclusion of a load balancer in a platform host, rather than in client deployments, a generic load balancer entity may be created, as an imputed entity, at S230. According to the same example, the inclusion of a provider-based load balancer functionality may be identified by comparing the data flow of a process executed across the deployed platform with a data flow of the same process executed across a simulated platform including only those entities included in data collection, providing for the identification of load balancer functionality based on differences between the analyzed data flows. According to the same example, a generic, imputed load balancer entity may be created at S230 where a load- balancing process or functionality is identified, but where the entity executing the process or functionality is not identified, providing for representation of a load balancer provided by a platform host as a discrete, generic entity.

It may be understood that the creation of imputed entities at S230 may be executed at any point after the initiation of S220, including simultaneously with S220, without loss of generality or departure from the scope of the disclosure.

At S240, a network graph is generated. A graph is a multi-dimensional data feature providing a representation of the contents and structure of a network, cloud, environment, or the like. A graph may include one or more graph vertices, interconnected by one or more graph edges. Where, as at S240, a graph is generated to represent a network, cloud, environment, or the like, each graph vertex may correspond with a network entity included in the network, cloud, environment, or the like, and each graph edge may corre- spond with a connection between such entities. Further, the entities with which the various graph vertices correspond may be generic entities, such as are described with respect to S220, imputed generic entities, such as are described with respect to S230, as well as any combination thereof.

A graph, as may be generated at S240, may be configured to represent one or more elements of a graph database, such as, as an example, the graph DB 160 of FIG. 1A, above, as

12

well as other, like, graph databases, where such graph databases may be configured to include various tables and other, like, arrangements of network entity information. Where a graph is configured to represent the elements of one or more graph databases, the graph may include various nodes or vertices corresponding with the various entries included in such databases, as well as tables, and the like, included therein. Further, where a graph is configured to represent the elements of one or more graph databases, the graph may include one or more edges, where such edges may represent relationships between the various entries included in such databases. Relationships between graph database entries, as may be represented as graph edges, may be identified by one or more analyses including, without limitation, analysis of graph database entry properties or attributes, analysis of graph database structures or formats, by other, like, analyses, and any combination thereof.

In an embodiment, such analysis of graph database entry properties or attributes may include execution of one or more resolution processes. According to the same embodi- ment, a resolution process is a process configured to identify one or more relationships between one or more network entities based on the known properties of such entities. Further, where execution of such a resolution process includes identification of such a relationship, the identified relationship may be, during graph generation at S240, included in a generated graph as a graph edge, linking the various graph vertices which correspond with the network elements between which the identified relationship exists. As an example, according to the same embodiment, a resolution process may be executed over a network segment including a load balancer configured to assign traffic to either of two connected backend pools, where the backend pools each include a proxy object configured to assign traffic to the various virtual machines (VMs) included in the proxy' objects' respective pools.

In the same example, the load balancer entity may include one or more load balancer rules, extracted at S240 as load balancer properties, where the load balancer rules may specify the related backend pools, such as by reference to backend pool identifiers or IP addresses. Further, according to the same example, each of the proxy objects may include proxy routing rules, which may be extracted as proxy object properties, specifying the assignment of traffic from the proxy objects to the various VMs of the proxy objects' respective pools. According to the same example, the reso- lution process may include the identification of relationships between the load balancer and the proxy objects based on the properties of the load balancer and the proxy objects, as well as identification of the relationships between the proxy objects and the objects' respective VMs, based on the properties of the objects and the VMs. Further, in the same example, the resolution process may include, following the identification of such relationships, the generation of graph edges corresponding with the identified relationships.

Further, a graph, as may be generated at S240, may be configured to provide for execution of one or more queries, commands, instructions, and the like, over the graph, as well as the various data features included therein. The structure of a graph, including several vertices, representing network entities, joined by graph edges, representing entity-to-entity relationships, provides for the execution of various standard, unified queries and commands over the graph, and data thereof. Such queries and commands may be automated, such as by application of one or more algorithms configured to generate and execute such queries and commands, or non-automated, such as by user entry or selection of one or

more queries or commands, such as through a graph user interface (UI). Further, such queries and commands may be configured to provide for the extraction of one or more insights based on a graph, and the contents thereof, where insights may be natural-language descriptions of graph features, such as “entity A is accessible from entity B via port X through entities C and D.” Examples of commands or queries applicable to unified graphs, such as may be generated at S240, include, without limitation, UI queries, back-end queries, graph algorithms, and the like, as well as any combination thereof.

Further, in an embodiment, a graph may be configured to provide for the execution of various user-defined commands, queries, instructions, and the like, where such user-defined commands may include, as an example and without limitation, a custom alert configured to generate a notification upon identification of an undesirable feature of a network graph, such as an unsecured internet connection.

As an example, where a graph database includes separate tables of virtual machines, databases, and proxy entities included in a network, where the graph database is configured to include connections between entries in the same table, as well as entries in different tables, execution of S240 may include the generation of one or more graphs representing the contents of the graph database. The graph generated to represent the graph database, at S240, may include multiple graph vertices, where each graph vertex corresponds to an entry in a table of the graph database, as well as graph edges linking the various vertices, where each graph edge corresponds with a relationship between two entries in the graph database.

In an embodiment, the various graph vertices, and inter-connecting graph edges, may be configured to include one or more descriptive data features. The descriptive data features may be configured to provide representation for one or more properties of the entities or connections corresponding with the various graph vertices and edges, such properties including, without limitation, entity names, addresses, types, configurations, edge direction, and the like, as well as any combination thereof.

At S250, the generated network graph is stored in a graph database. Storage of a graph object at S250 may include storage of a graph object as a unified representation of network or environment entities, and relationships therebetween, providing for functionalities including, without limitation, execution of unified queries across graphs or graph elements, execution of unified graph management commands or instructions, other, like, functionalities, and any combination thereof.

In an embodiment, providing a unified representation for network objects, and generation of the network graph, allows for querying of the graph using a simplified and unified set of queries, configuring network entities using a simplified and unified set of comments, and generating insights with respect to the network entities included in the environment.

In yet another embodiment, the network graph may be utilized to generate and return a graph visualization at S250. A graph visualization is a visual representation of the network graph.

The graph visualization may be generated by, for example, generation of one or more visual overlays, presentations, or other, like, representations of graph data, such as the visualization described with respect to FIG. 3, below. Such generation of visual representations of graph data may include the generation of visual representations where graph vertices are depicted as points, nodes, or other, like, discrete,

visual indications, and where graph edges are depicted as visible interconnections between such points, nodes, and the like, such as lines. Further, generation of such visual representations may include labeling, tagging, or otherwise associating the depicted vertices and edges with one or more visual descriptors, where such visual descriptors may be configured to provide information including, without limitation, element names, types, statuses, edge directionality, and the like, as well as any combination thereof. In an embodiment, such visual descriptors may be configured to display upon selection of a graph element, such as by clicking an element with a computer mouse during presentation of a visualization through a graph visualization presentation platform.

The graph visualization, as may be generated and returned at S250, may be configured to provide for one or more functionalities including, without limitation, visualization of network, cloud, or environment elements and connections, user interactivity, such as by clicking elements or connections to display relevant information, searching or querying of graph elements or underlying graph data features, and the like, as well as any combination thereof. An example of a graph visualization is described with respect to FIG. 3, below.

FIG. 3 is an example of a network graph visualization 300, according to an embodiment. The example network graph visualization 300 is generated as a visual representation of a network, cloud, environment, or the like, where such generation is described with respect to S250, above, and where the visualization is presented through a network graph visualization utility. A network graph visualization utility may be an application, interface, or other, like, means of providing a visual representation of a network graph visualization 300, and the like, where the provided network graph visualization 300 may include various interactive features, as described hereinbelow. A network graph visualization utility may be configured as, as examples and without limitation, a web interface, an application or executable installed on a user or administrator device, other, like, configurations, and any combination thereof.

The network graph visualization 300 of FIG. 3 is a network graph visualization representing a network, such as the networks described hereinabove, where the various entities, systems, devices, components, and the like, of the network are represented as visualizations of graph vertices 310, and where such nodes are variously interconnected by visualizations of graph edges 320, representing connections between the various entities, systems, devices, components, and the like. It may be understood that while only one graph vertex visualization 310 and one graph edge visualization 320 are labeled for purposes of simplicity, other, like, vertex visualizations 310 and edge visualizations 320 may be so labeled without loss of generality or departure from the scope of the disclosure.

The network graph visualization 300, and corresponding network graph visualization utility, may be configured to provide for various interactive functionalities. In an embodiment, where a user interacts with a graph vertex visualization 310, such as by clicking the graph vertex visualization 310 with a mouse or tapping the graph vertex visualization 310 through a touchscreen, the graph visualization utility may be configured to display a graph vertex overview pane 315. The graph vertex overview pane 315 may be an information panel, including data relating to the given graph vertex visualization 310 and describing various entity data features, such as those object data features collected at S210 of FIG. 2, above. The graph vertex overview pane 315 may

US 11,929,896 B1

15

be configured to provide information relating to the various vertex visualizations 310 including, as examples and without limitation, object names, types, statuses, relevant meta-data, and the like, as well as any combination thereof.

In addition, the network graph visualization 300, and corresponding network graph visualization utility, may be configured to provide for the selection of one or more graph visualization 300 views. A graph visualization 300 view is a selective presentation of one or more aspects of a graph visualization 300, including selection of, as examples and without limitation, entities and connections external to a specified network, cloud, environment, or the like, entities and connections included in a first network of a cloud environment, entities and connections relevant to a specified product, service, process, or the like, entities and connections having one or more specified criteria, and the like, as well as any combination thereof. A network graph visualization, and the corresponding network graph utility, may be configured to provide such views by one or more means including, without limitation, generation of displays including only relevant objects and connections, generation of displays including application of identifying highlights, shading, or the like, to such relevant entities and connections, other, like, means, and any combination thereof.

Further, the network graph visualization 300, and corresponding network graph visualization utility, may be configured to include a search tool 330, providing for location and selection of one or more user-specified graph vertex visualizations 310 or graph edge visualizations 320 within the graph. The search tool 330 may be configured to provide for search functionality based on one or more user specifications including, as examples and without limitation, entity or connection names, types, IDs, statuses, labels or tags associated with various elements of the network graph visualization 300, and the like, as well as any combination thereof. Further, the search tool 330 may be configured to provide for the execution of one or more compound queries, such as queries relating to objects or connections including multiple attributes, each attribute matching an element of the query. The search tool 330 may be further configured to provide for the execution of one or more searches or queries, as described herein, including, without limitation, complex queries, as described, to or on one or more views, such as those views described hereinabove. In addition, the network graph visualization 300, and corresponding network graph visualization utility, may be configured to include a help tool 340, providing for display of one or more resources related to the network graph visualization 300 and network graph visualization utility.

It should be noted that a network graph visualization 300, shown in FIG. 3, provides a visual representation of graph data, where such data may be provided in other formats including, without limitation, tables, charts, other, non-visual, data organization formats, lists of entities, other, like, formats, and any combination thereof.

FIG. 4 is an illustration 400 of the genericization of a network entity, utilized to describe the various embodiments. The illustration 400 includes a specific object 410 and a generic object 420, where the generic object corresponds to the specific object 410. The specific object is a network entity including one or more known properties 412-1 through 412-N (hereinafter referred to as “specific object property” 412 or “specific object properties” 412), where each specific object property 412 includes a value 414-1 through 414-N (hereinafter referred to as “specific object property value” 414 or “specific object property values”). The generic object 420 is a network entity generated

16

by one or more means, including, without limitation, the genericization processes described with respect to FIG. 2, above. The generic object 420 includes one or more properties 422-1 through 422-N (hereinafter referred to as “generic object property” 422 or “generic object properties” 422), where each generic object property 422 includes a value 424-1 through 424-N (hereinafter referred to as “generic object property value” 424 or “generic object property values” 424). As is applicable to the specific object properties 412, the specific object property values 414, the generic object properties 422, and the generic object property values 424, “N” is an integer having a value greater than one.

The genericization of an object, as depicted in the illustration 400, is the process of generating a generic object 420 from a specific object 410. The genericization of an object may be achieved using the method described with respect to FIG. 2, other, like, methods, and any combination thereof. The generated generic object 420 is a network entity having a type similar or identical to the type of the specific object 410 and including one or more generic object properties 422 similar or identical to the specific object properties 412. The generic object 420 may be generated according to one or more pre-defined or user-defined formats, schemas, templates, and the like, providing for inclusion of one or more generic object properties 422 for generic objects 420 of a given type. The generic object property values 424 may be values populated, during genericization, based on the corresponding specific object property values 414. Further, generic object property values 424 may be pre-defined or user-defined, such as in a template, schema, or the like, from which the generic object 420 is generated, providing for population of generic object property values 424 with one or more default or pre-configured values.

As an example, where a specific object 410 is genericized, such as during the execution of the method described with respect to FIG. 2, a generic object 420 may be generated during genericization. The generated generic object 420 may be of the same type as the specific object 410, such as, for example, a firewall entity. Where the generic object 420 is generated based on a template firewall entity, the generic object 420 may include one or more pre-configured generic object properties 422, such as, as examples and without limitation, firewall rules, firewall addresses, firewall port configurations, firewall connection configurations, and the like, as well as any combination thereof. During genericization, the generic object property values 424 may be populated based on the corresponding specific object property values 414, such as specific object 410 firewall rules, specific object 410 firewall addresses, specific object 410 firewall port configurations, specific object 410 firewall connection configurations, and the like. Where, according to the same example, the generic object 420 includes one or more generic object properties 422 corresponding to specific object properties 412 which are not defined in the specific object 410, or for which specific object property values 414 are not defined, genericization may include the population of generic object property values 424 with values pre-defined in the template or format from which the generic object 420 is generated.

FIG. 5A is an example interactive graph analysis platform 500, provided through a user interface (UI), according to an embodiment. The example platform 500 provides for user interaction with one or more graph analysis tools through a user device configured to execute the UI, where such a user device may be, as examples and without limitation, a personal computer, a tablet computer, a smartphone, other,

like, devices, as well as various combinations thereof. The example platform **500** includes a query construction pane **510**, including a query construction tool **515**, as well as a per-query unique return display **520**, the per-query unique return display including several query return columns **522-1** through **522-4** (hereinafter, “query return column” **522** or “query return columns” **522**).

The query construction pane **510** is a feature of a graph analysis platform, providing for display of the query construction tool **515**. The query construction pane **510** may be of a fixed size relative to the graph analysis platform, or may, in an embodiment, be dynamically re-sized based on one or more factors including, without limitation, user re-size commands, automatic resizing based on the properties of the query construction tool, such as the number of queries included in a compound query constructed therein, as well as other, like, bases, and any combination thereof.

The query construction tool **515** is an interactive graph analysis tool, providing for user interaction with data included in a graph, such as graphs generated as described herein, on a visual basis. The query construction tool **515** may be configured to accept one or more inputs, which inputs may be, upon execution of a query by a graph analysis system, or other, similar, component, device, system, or the like, configured to provide various graph analysis functionalities, automatically translated into one or more graph-search queries, instructions, commands, or the like, providing for development of compound queries via the tool’s **515** visual interface.

Graph-search queries, instructions, commands, or the like, are graph-specific functions providing for the identification of one or more entities, or other, similar graph elements, based on the various input parameters specified by a user via the tool **515**. Graph-search queries, and the like, as are applicable to the automated retrieval of graph information based on input parameters, are queries, instructions, commands, and the like, generated and structured in one or more formats or languages specific to the structure or format of a graph, such as, as examples and without limitation, JavaScript Object Notation (JSON) queries, other, like, queries, and any combination thereof. Graph-search queries may be configured to accept one or more input parameters, to traverse one or more graphs and identify elements thereof matching the input parameters when executed by a graph analysis system or other, like, system, and to return the one or more identified graph elements. Graph search queries may be generated via one or more means including, without limitation, user-generation by development of specific queries in query formats, conversion of user input selections, such as through the query construction tool **515**, into query formats, by other, like, means, and any combination thereof.

The query construction tool **515** may be configured to provide for user selection of one or more graph query criteria, including query criteria corresponding to designated query search and filter refinements. Query search refinements are top-layer query search criteria, providing for selection of one or more graph entities or elements matching one or more specifications, such specifications including, as an example and without limitation, a type of computing entities. Query filter refinements are lower-layer query search criteria, providing for selection of filtered graph entities or elements, from the group of entities or elements relevant to a given query search refinement, where such filtered graph entities or elements may be filtered on the basis of one or more factors or attributes including, without limitation, entity or element names, types, addresses, connection types, entity or element locations or groups, other,

like, factors or attributes, and any combination thereof. The query construction tool **515** may be configured to provide for user generation of compound queries, wherein compound queries are queries including a search refinement and one or more filter refinements, providing for identification and selection of graph entities or elements including one or more properties or attributes, wherein such properties or attributes are those specified in the compound query.

The per-query unique return display **520** is an information pane configured to provide visual displays of graph information relevant to the query or queries generated via the query construction tool **515**. The per-query unique return display is configured to automatically generate, upon return of graph information relevant to an executed graph query, one or more query return columns **522-1** through **522-4**. The query return columns **522** are columns including information entries, wherein the information entries included in each column correspond with the various properties or attributes of one or more graph entities or elements returned in response to the execution of a graph query. Examples of query return columns, described by the columns’ respective headers, include, without limitation, graph entity or element names, relevant cloud platforms, internet accessibility statuses, and the like, as well as any combination thereof.

Although only four query return columns **522-1** through **522-4** are shown in the example platform **500** for purposes of simplicity, it may be understood that one or more of such columns may be likewise relevant to an executed query without loss of generality or departure from the scope of the disclosure.

As an example, according to the platform **500** described with respect to FIG. **5A**, a graph search query may be generated based on a user’s selection of query inputs providing for a search for databases or virtual machines (VMs) which contain network interfaces, which are inside subnets, which are inside virtual networks with one or more user-specified names, which are connected to one or more peering connections, and which are peered to a virtual network with a specified name. Based on the identification of graph elements or entities including the attributes specified in the compound query developed through the query construction tool **515**, the platform **500** may be configured to return a set of virtual machines and databases matching the described query, and to, through the per-query unique return display **520**, display various attributes of each database or virtual machine identified.

FIG. **5B** is an example interactive graph analysis platform **550**, provided through a user interface (UI), according to an embodiment. The example platform **550** provides for user interaction with one or more graph analysis tools through a user device configured to execute the UI, where such a user device may be, as examples and without limitation, a personal computer, a tablet computer, a smartphone, other, like, devices, as well as various combinations thereof. The example platform **550** includes a query construction pane **510**, including a query construction tool **515**, as well as a per-query unique return display **530**, the per-query unique return display including several query return columns **532-1** through **532-5** (hereinafter, “query return column” **532** or “query return columns” **532**).

The query construction pane **510** is a feature of a graph analysis platform, providing for display of the query construction tool **515**. The query construction pane **510** may be of a fixed size relative to the graph analysis platform, or may, in an embodiment, be dynamically re-sized based on one or more factors including, without limitation, user re-size commands, automatic resizing based on the properties of the

query construction tool, such as the number of queries included in a compound query constructed therein, as well as other, like, bases, and any combination thereof.

The query construction tool **515** is an interactive graph analysis tool, providing for user interaction with data included in a graph, such as graphs generated as described herein, on a visual basis. The query construction tool **515** may be configured to accept one or more inputs, which inputs may be, upon execution of a query by a graph analysis system, or other, similar, component, device, system, or the like, configured to provide various graph analysis functionalities, automatically translated into one or more graph-search queries, instructions, commands, or the like, providing for development of compound queries via the tool's **515** visual interface.

Graph-search queries, instructions, commands, or the like, are graph-specific functions providing for the identification of one or more entities, or other, similar graph elements, based on the various input parameters specified by a user via the tool **515**. Graph-search queries, and the like, as are applicable to the automated retrieval of graph information based on input parameters, are queries, instructions, commands, and the like, generated and structured in one or more formats or languages specific to the structure or format of a graph, such as, as examples and without limitation, JavaScript Object Notation (JSON) queries, other, like, queries, and any combination thereof. Graph-search queries may be configured to accept one or more input parameters, to traverse one or more graphs and identify elements thereof matching the input parameters when executed by a graph analysis system or other, like, system, and to return the one or more identified graph elements. Graph search queries may be generated via one or more means including, without limitation, user-generation by development of specific queries in query formats, conversion of user input selections, such as through the query construction tool **515**, into query formats, by other, like, means, and any combination thereof.

The query construction tool **515** may be configured to provide for user selection of one or more graph query criteria, including query criteria corresponding to designated query search and filter refinements. Query search refinements are top-layer query search criteria, providing for selection of one or more graph entities or elements matching one or more specifications, such specifications including, as an example and without limitation, a type of computing entity. Query filter refinements are lower-layer query search criteria, providing for selection of filtered graph entities or elements, from the group of entities or elements relevant to a given query search refinement, where such filtered graph entities or elements may be filtered on the basis of one or more factors or attributes including, without limitation, entity or element names, types, addresses, connection types, entity or element locations or groups, other, like, factors or attributes, and any combination thereof. The query construction tool **515** may be configured to provide for user generation of compound queries, wherein compound queries are queries including a search refinement and one or more filter refinements, providing for identification and selection of graph entities or elements including one or more properties or attributes, wherein such properties or attributes are those specified in the compound query.

The per-query unique return display **530** is an information pane configured to provide visual displays of graph information relevant to the query or queries generated via the query construction tool **515**. The per-query unique return display is configured to automatically generate, upon return of graph information relevant to an executed graph query,

one or more query return columns **532-1** through **532-5**. The query return columns **532** are columns including information entries, wherein the information entries included in each column correspond with the various properties or attributes of one or more graph entities or elements returned in response to the execution of a graph query. Examples of query return columns, described by the columns' respective headers, include, without limitation, graph entity or element names, relevant network interfaces, relevant subnets, relevant virtual networks, relevant gateways, and the like, as well as any combination thereof.

Although only five query return columns **532-1** through **532-5** are shown in the example platform **550** for purposes of simplicity, it may be understood that one or more of such columns may be likewise relevant to an executed query without loss of generality or departure from the scope of the disclosure.

As an example, according to the platform **550** described with respect to FIG. **5B**, a graph search query may be generated based on a user's selection of query inputs providing for a search for virtual machines (VMs) which contain network interfaces, which are inside subnets, which are inside virtual networks, which contain internet gateways, where the virtual network network address is public, and where the virtual network includes a network security rule which specifies connection allowance for a given source address. Based on the identification of graph elements or entities including the attributes specified in the compound query developed through the query construction tool **515**, the platform **530** may be configured to return a set of virtual machines matching the described query, and to, through the per-query unique return display **530**, display various attributes of each virtual machine identified.

FIG. **6** is an example hardware block diagram **600** depicting a graph analysis system **150**, according to an embodiment. The graph analysis system **150** includes a processing circuitry **610** coupled to a memory **620**, a storage **630**, and a network interface **640**. In an embodiment, the components of the graph analysis system **150** may be communicatively connected via a bus **650**.

The processing circuitry **610** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory **620** may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof.

In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage **630**. In another configuration, the memory **620** is configured to store such software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the processing circuitry **610**, cause the processing circuitry **610** to perform the various processes described herein.

US 11,929,896 B1

21

The storage **630** may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or another memory technology, compact disk-read only memory (CD-ROM), Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information.

The network interface **640** allows the graph analysis system **150** to communicate with the various components, devices, and systems described herein for generation of unified graph models for network entities, as well as other, like, purposes.

It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. **6**, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

It should be noted that the computer-readable instructions may be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code, such as in source code format, binary code format, executable code format, or any other suitable format of code. The instructions, when executed by the circuitry, cause the circuitry to perform the various processes described herein.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units (CPUs), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform, such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

As used herein, the phrase "at least one of" followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including "at least one of A, B, and C," the system can include A alone; B alone; C alone; A and B in combination; B and C in combination; A and C in combination; or A, B, and C in combination.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the

22

future, i.e., any elements developed that perform the same function, regardless of structure.

What is claimed is:

1. A method for generation of unified graph models for network entities, comprising:

collecting, for each network entity of a plurality of network entities, network entity data, wherein the network entity data collected for a network entity includes at least a network entity property, wherein the plurality of network entities are deployed in a plurality of cloud computing platforms;

genericizing each of the network entities based on the respective collected network entity data to generate a plurality of generic network entities, wherein a generic network entity includes a generic representation of respective network entities from different cloud computing platforms of the plurality of cloud computing platforms;

generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of generic network entities and relations between the generic network entities of the plurality of network entities; and creating at least one imputed entity, wherein the at least one imputed entity is a generic network entity representing an executed platform functionality, and wherein the executed platform functionality is different than a network entity; and

storing the generated network graph.

2. The method of claim **1**, wherein the network entity property relates to at least one of: a network entity type, a network entity class, a network entity category, and a network entity configuration.

3. The method of claim **1**, wherein genericizing each of the network entities further comprises:

generating a new generic network entity, wherein the new generic network entity includes at least a network entity property of network entities.

4. The method of claim **1**,

wherein a new generic network entity includes at least a network entity property of at least one network entity.

5. The method of claim **1**, wherein creating at least one imputed entity comprises:

identifying at least one cloud computing platform of the plurality of cloud computing platforms or environment functionality which corresponds with a functionality of a generic network entity.

6. The method of claim **5**, wherein the generated network graph is a unified representation of the plurality of network entities including the imputed network entities and the generic network entities.

7. The method of claim **1**, wherein the generated network graph includes at least one graph vertex, wherein the at least one graph vertex represents at least one of: a network entity, a generic entity, and an imputed entity.

8. The method of claim **7**, wherein the at least one graph vertex includes at least a property label, wherein the at least a property label includes at least a description of a property of the at least one graph vertex.

9. The method of claim **7**, wherein the generated at least a network graph includes at least one graph edge, wherein the at least one graph edge is a connection between two graph vertices, and wherein the at least one graph edge represents a relationship between two connected entities, wherein a connected entity is at least one of: a network entity, a generic entity, and an imputed entity.

US 11,929,896 B1

23

10. The method of claim 9, wherein the at least one graph edge includes at least one of: a property label, and a directionality indicator, wherein a property label includes at least a description of a property of the at least one graph edge, and wherein a directionality indicator includes at least a description of a direction of the at least one graph edge.

11. The method of claim 1, wherein storing the generated network graph further comprises:

returning at least a visualization of the generated at least one network graph.

12. The method of claim 1, wherein storing the generated at network graph further comprises:

storing the generated network graph in a graph database.

13. The method of claim 1, wherein a network entity of the plurality of network entities includes any one of: a private endpoint, a transit gateway, a tag-based ruleset, an entity configured to implement a tag-based ruleset, a container-management service, a container-management application, a third-party container, a third-party image, a web-access firewall, a firewall implementation, a multi-entity connection, a cross-entity connection, a container manager, and a container manager connection.

14. A non-transitory computer readable medium having stored thereon instructions for causing a processing circuitry to execute a process for generation of unified graph models for network entities, the process comprising:

collecting, for each network entity of a plurality of network entities, network entity data, wherein the network entity data collected for a network entity includes at least a network entity property, wherein the plurality of network entities are deployed in a plurality of cloud computing platforms;

generalizing each of the network entities based on the respective collected network entity data to generate a plurality of generic network entities, wherein a generic network entity includes a generic representation of respective network entities from different cloud computing platforms of the plurality of cloud computing platforms;

generating at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of generic network entities and relations between the network entities of the plurality of network entities;

creating at least one imputed entity, wherein the at least one imputed entity is a generic network entity representing an executed platform functionality, and wherein the executed platform functionality is different than a network entity; and

storing the generated network graph.

15. A system for generating unified graph models for network entities, comprising:

a processing circuitry; and

a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:

collect, for each network entity of a plurality of network entities, network entity data, wherein the network entity data collected for a network entity includes at least a network entity property, wherein the plurality of network entities are deployed in a plurality of cloud computing platforms;

generalize each of the network entities based on the respective collected network entity data to generate a plurality of generic network entities, wherein a generic network entity includes a generic representation of

24

respective network entities from different cloud computing platforms of the plurality of cloud computing platforms;

generate at least a network graph, wherein the generated network graph is a multi-dimensional data structure providing a representation of the plurality of network entities and relations between the network entities of the plurality of network entities;

creating at least one imputed entity, wherein the at least one imputed entity is a generic network entity representing an executed platform functionality, and wherein the executed platform functionality is different than a network entity; and

store the generated network graph.

16. The system of claim 15, wherein the network entity property relates to at least one of: a network entity type, a network entity class, a network entity category, and a network entity configuration.

17. The system of claim 15, wherein the system is further configured to:

generate a new generic network entity, wherein the new generic network entity includes at least a network entity property of network entities.

18. The system of claim 15,

wherein a new generic network entity includes at least a network entity property of at least one network entity.

19. The system of claim 15, wherein the system is further configured to:

identify at least one cloud computing platform of the plurality of cloud computing platforms or environment functionality which corresponds with a functionality of a generic network entity.

20. The system of claim 19, wherein the generated network graph is a unified representation of the plurality of network entities including the imputed network entities and the generic network entities.

21. The system of claim 15, wherein the generated at least a network graph includes at least one graph vertex, wherein the at least one graph vertex represents at least one of: a network entity, a generic entity, and an imputed entity.

22. The system of claim 21, wherein the at least one graph vertex includes at least a property label, wherein the at least a property label includes at least a description of a property of the at least one graph vertex.

23. The system of claim 21, wherein the generated at least a network graph includes at least one graph edge, wherein the at least one graph edge is a connection between two graph vertices, and wherein the at least one graph edge represents a relationship between two connected entities, wherein a connected entity is at least one of: a network entity, a generic entity, and an imputed entity.

24. The system of claim 23, wherein the at least one graph edge includes at least one of: a property label, and a directionality indicator, wherein a property label includes at least a description of a property of the at least one graph edge, and wherein a directionality indicator includes at least a description of a direction of the at least one graph edge.

25. The system of claim 15, wherein the system is further configured to:

return at least a visualization of the network graph.

26. The system of claim 15, wherein the system is further configured to:

store the generated network graph in a graph database.

27. The system of claim 15, wherein a network entity of the plurality of network entities includes any one of: a private endpoint, a transit gateway, a tag-based ruleset, an entity configured to implement a tag-based ruleset, a con-

tainer-management service, a container-management application, a third-party container, a third-party image, a web-access firewall, a firewall implementation, a multi-entity connection, a cross-entity connection, a container manager, and a container manager connection.

5

28. The method of claim **3**, wherein the generic network entity represents at least an entity configured to apply security rules.

29. The system of claim **17**, wherein the generic network entity represents at least an entity configured to apply security rules.

10

* * * * *

EXHIBIT C



(12) **United States Patent**
Reznik et al.

(10) **Patent No.:** **US 11,936,693 B2**
 (45) **Date of Patent:** **Mar. 19, 2024**

(54) **SYSTEM AND METHOD FOR APPLYING A POLICY ON A NETWORK PATH**

(71) Applicant: **Wiz, Inc.**, New York, NY (US)

(72) Inventors: **Roy Reznik**, Tel Aviv (IL); **Matilda Lidgi**, Tel Aviv (IL); **Shai Keren**, Tel Aviv (IL); **Eliran Marom**, Yehud-Monoson (IL)

(73) Assignee: **WIZ, INC.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **18/357,845**

(22) Filed: **Jul. 24, 2023**

(65) **Prior Publication Data**
 US 2023/0370499 A1 Nov. 16, 2023

Related U.S. Application Data

(63) Continuation-in-part of application No. 17/818,898, filed on Aug. 10, 2022, and a continuation-in-part of application No. 17/659,163, filed on Apr. 13, 2022, and a continuation-in-part of application No. 17/659,164, filed on Apr. 13, 2022, and a continuation-in-part of application No. 17/659,165, filed on Apr. 13, 2022.

(51) **Int. Cl.**
H04L 9/40 (2022.01)

(52) **U.S. Cl.**
 CPC **H04L 63/20** (2013.01)

(58) **Field of Classification Search**
 CPC H04L 63/20
 USPC 726/1
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

10,574,675 B2	2/2020	Peppe et al.	
2016/0140352 A1*	5/2016	Nickolov	H04W 12/02 726/26
2016/0366185 A1	12/2016	Lee et al.	
2017/0034198 A1	12/2017	Powers et al.	
2018/0004950 A1	1/2018	Gupta et al.	
2020/0389469 A1	12/2020	Litichever et al.	

(Continued)

OTHER PUBLICATIONS

International Search Report of PCT/IB2023/058074, dated Nov. 20, 2023. Searching Authority United States Patent and Trademark Office, Alexandria, Virginia.

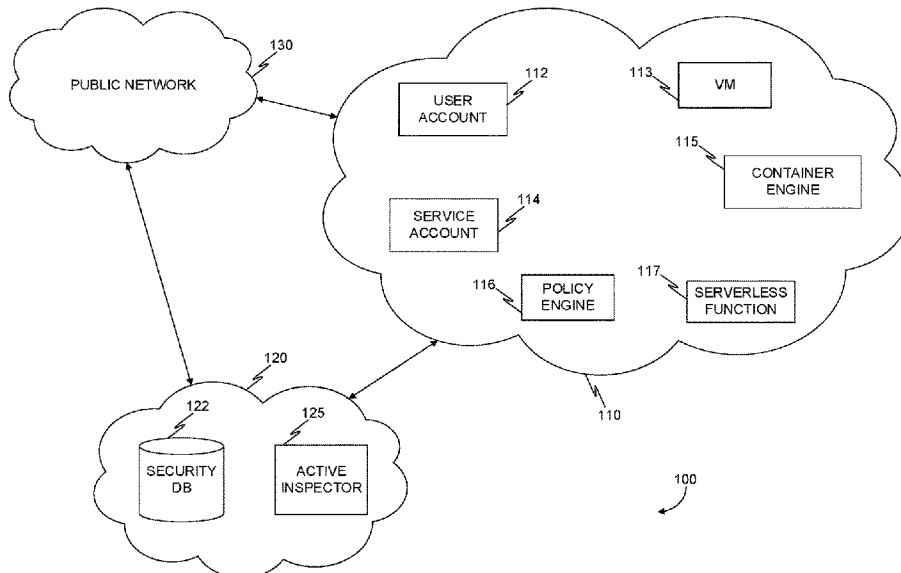
(Continued)

Primary Examiner — Anthony D Brown
 (74) *Attorney, Agent, or Firm* — M&B IP Analysts, LLC

(57) **ABSTRACT**

A system and method for applying a policy on a network path is disclosed. The method includes: selecting a reachable resource having a network path to access the reachable resource, wherein the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment; actively inspecting the network path to determine if the network path of the reachable resource is accessible from the external network; applying a policy on the accessible network path, wherein the policy includes a conditional rule; initiating a mitigation action, in response to determining that the conditional rule is not met; and applying the policy on another network path, in response to determining that the conditional rule is met.

21 Claims, 10 Drawing Sheets



US 11,936,693 B2

Page 2

(56)

References Cited

U.S. PATENT DOCUMENTS

2021/0105304 A1 4/2021 Kraning et al.
2023/0136839 A1* 5/2023 Sundararajan G06N 20/00
711/154

OTHER PUBLICATIONS

Written Opinion of the Searching Authority of PCT/IB2023/058074, dated Nov. 20, 2023. Searching Authority United States Patent and Trademark Office, Alexandria, Virginia.

* cited by examiner

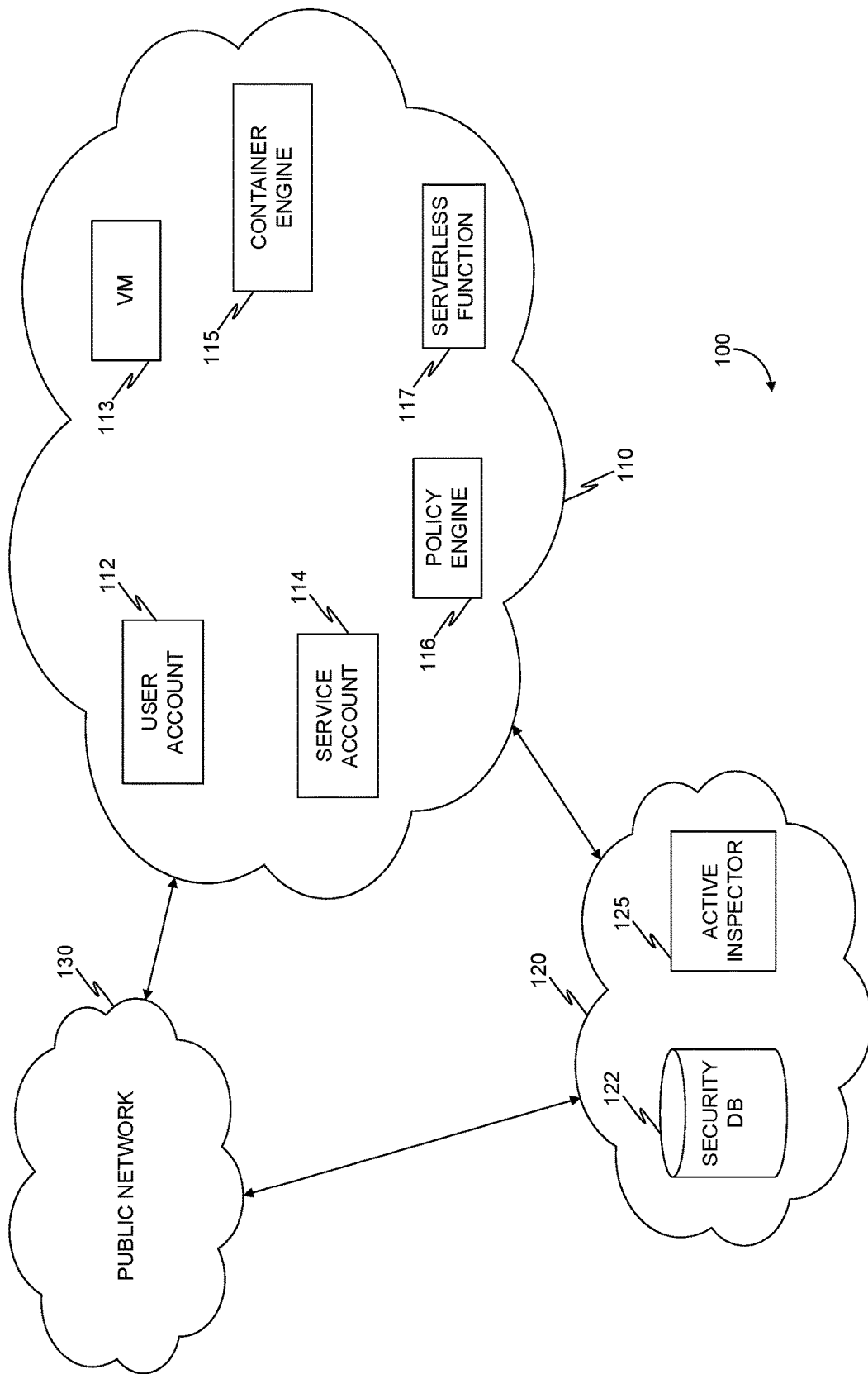


FIG. 1

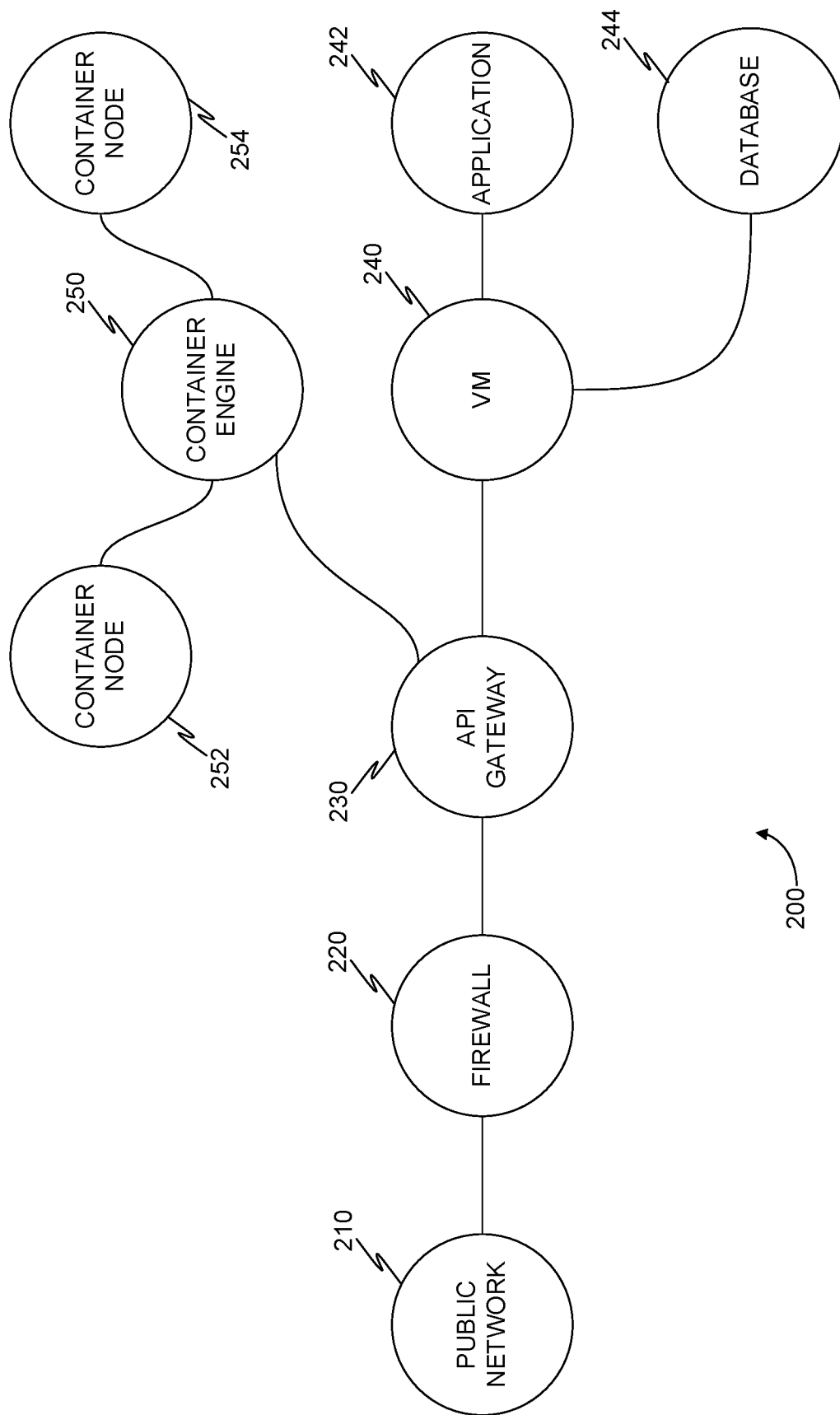


FIG. 2

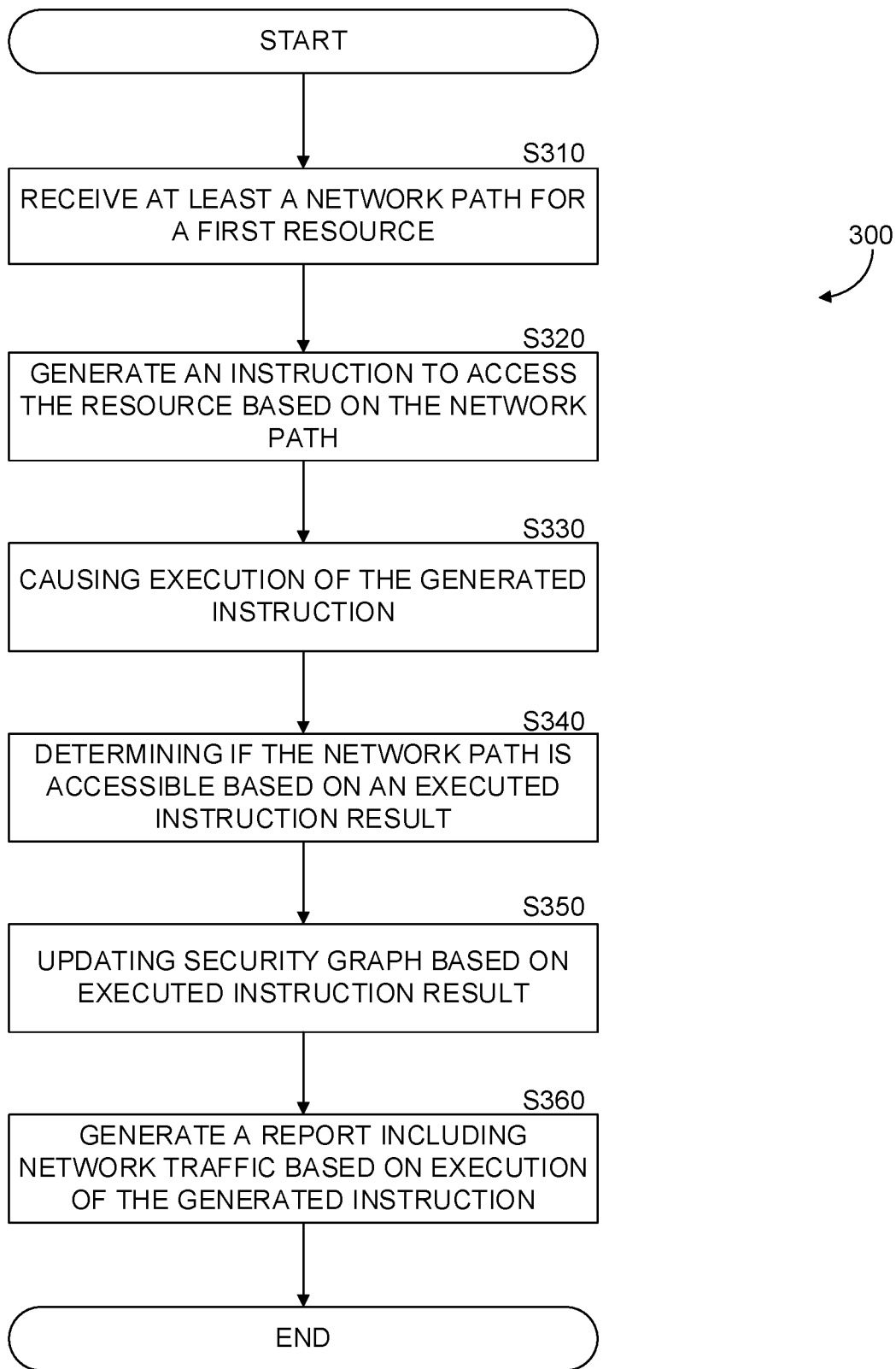


FIG. 3

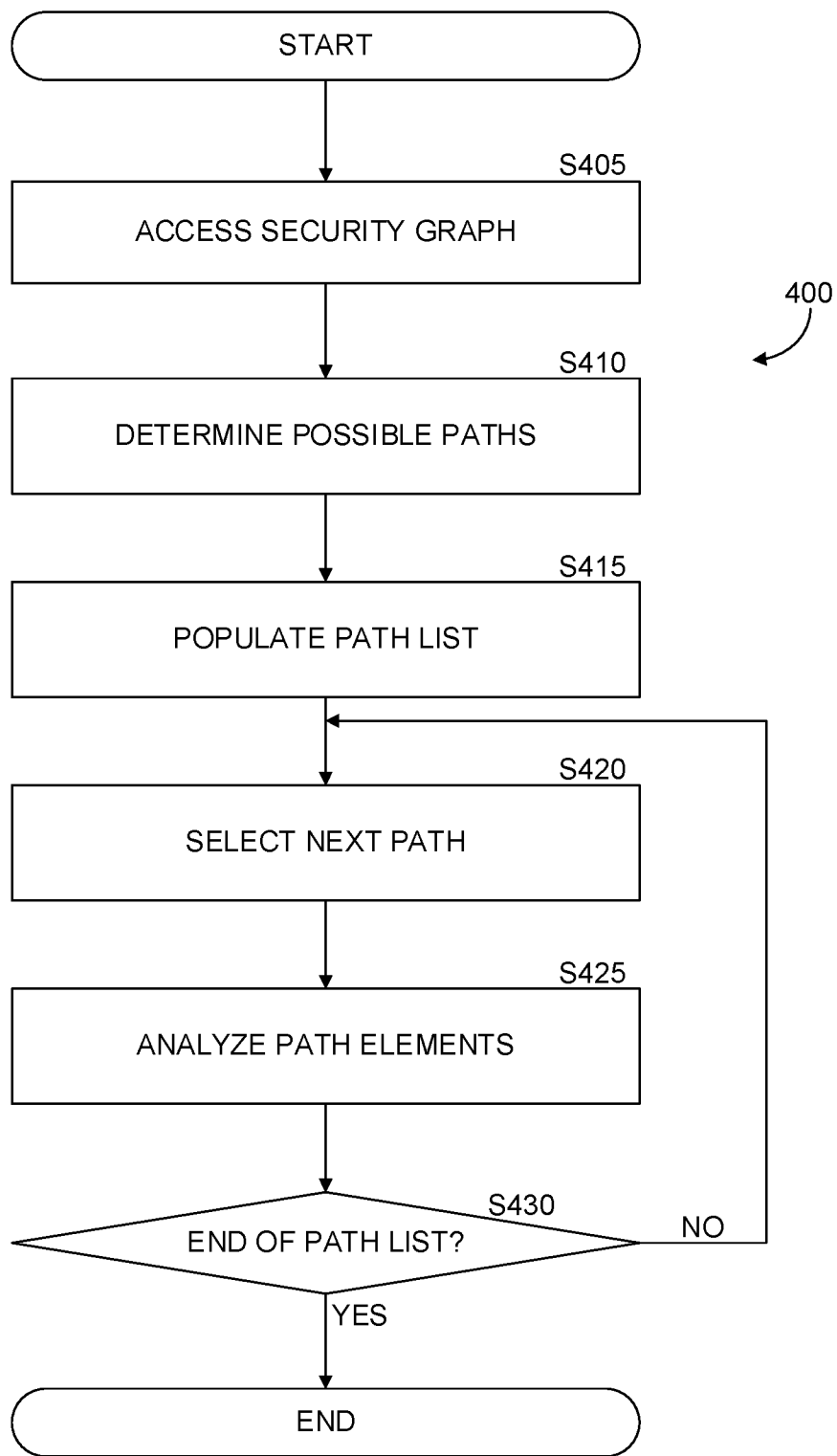


FIG. 4A

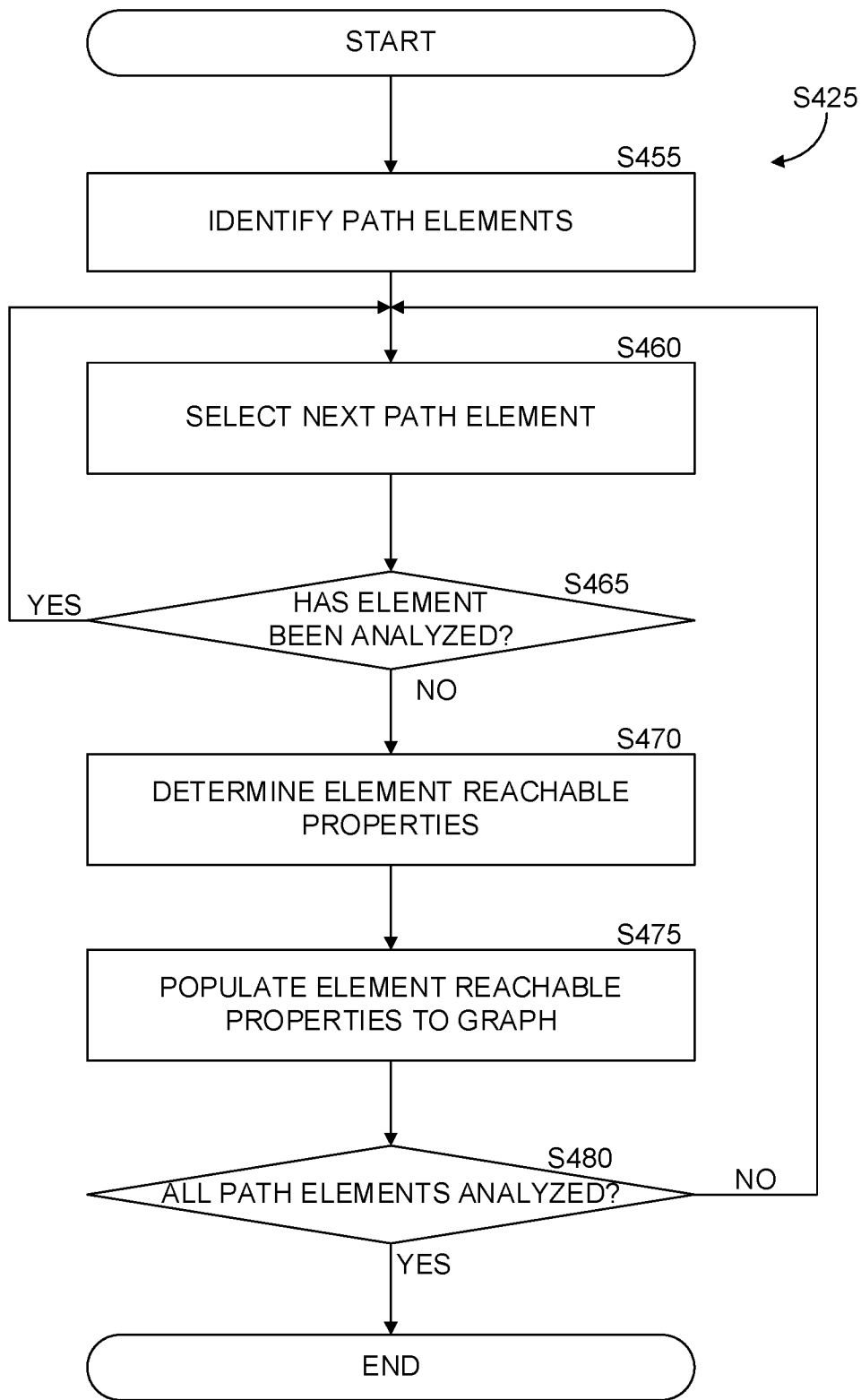


FIG. 4B

500

403 Forbidden — 510

512
514
516
518

- * Code: AccessDenied
- * Message: Access Denied
- * RequestId: 09PP3N3590KVTGWW
- * HostId: uGXESSiwa0tojg3nla9X81xD4e3eD1W1aAttrL61yDrelHU8puUX24XUyoCD/4RQHFParJjAuuDI=

FIG. 5

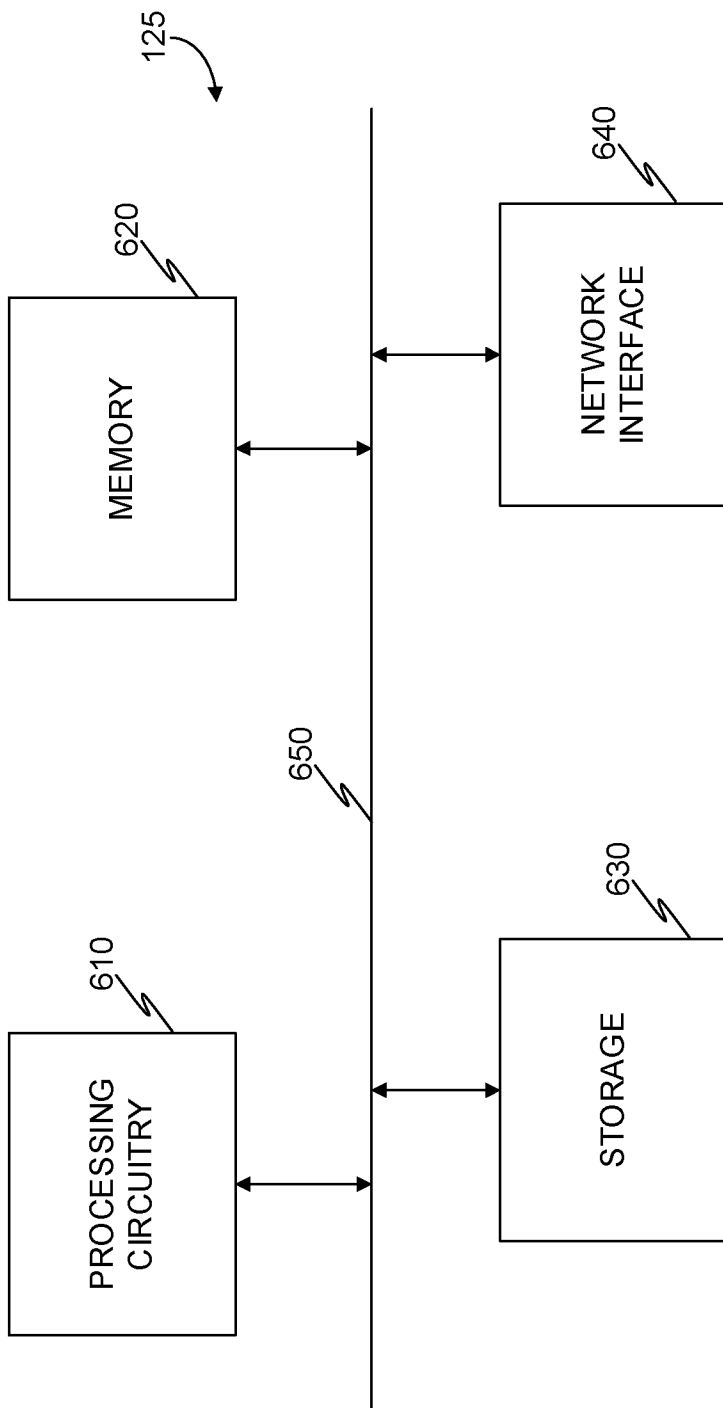


FIG. 6

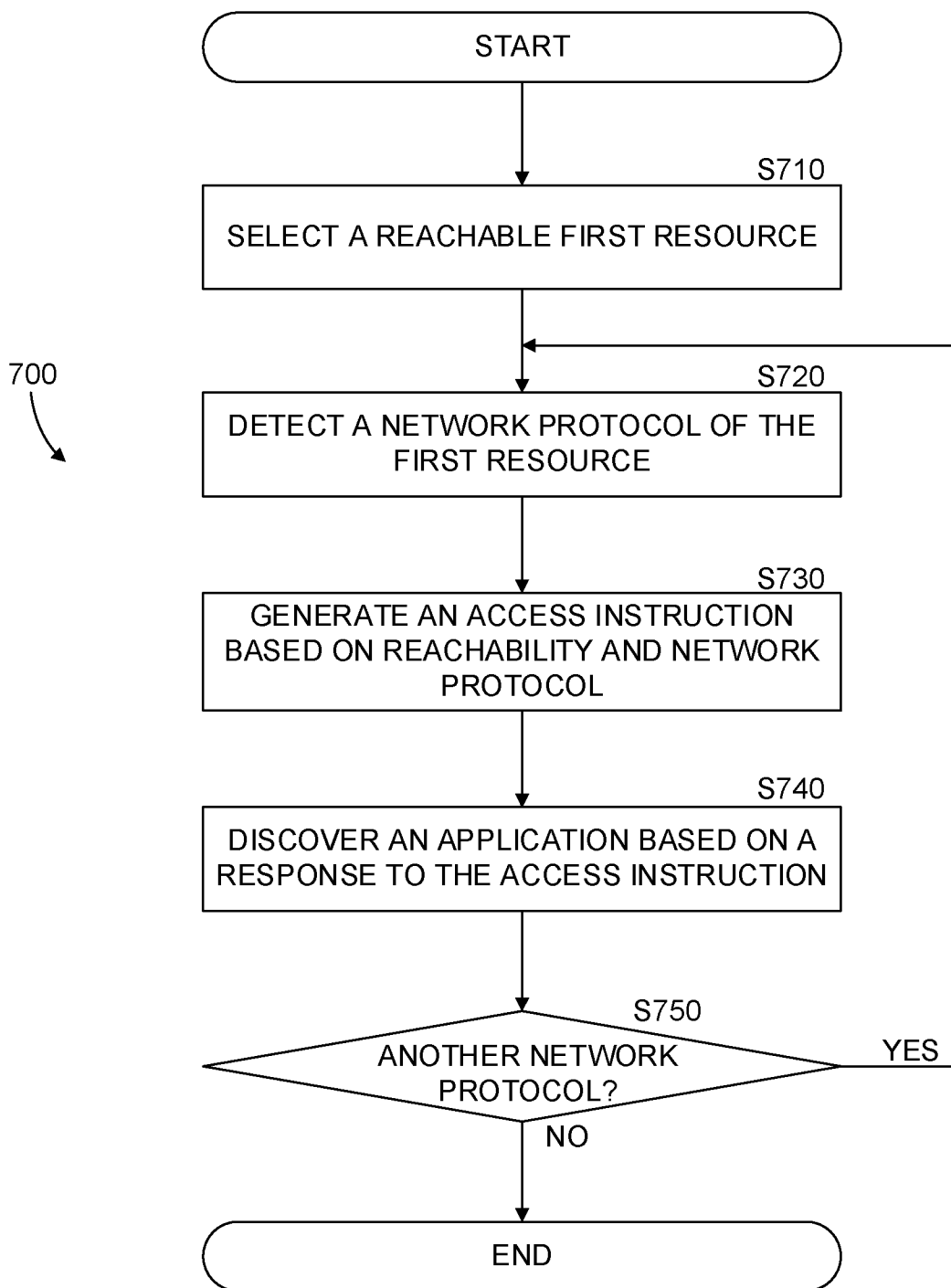


FIG. 7

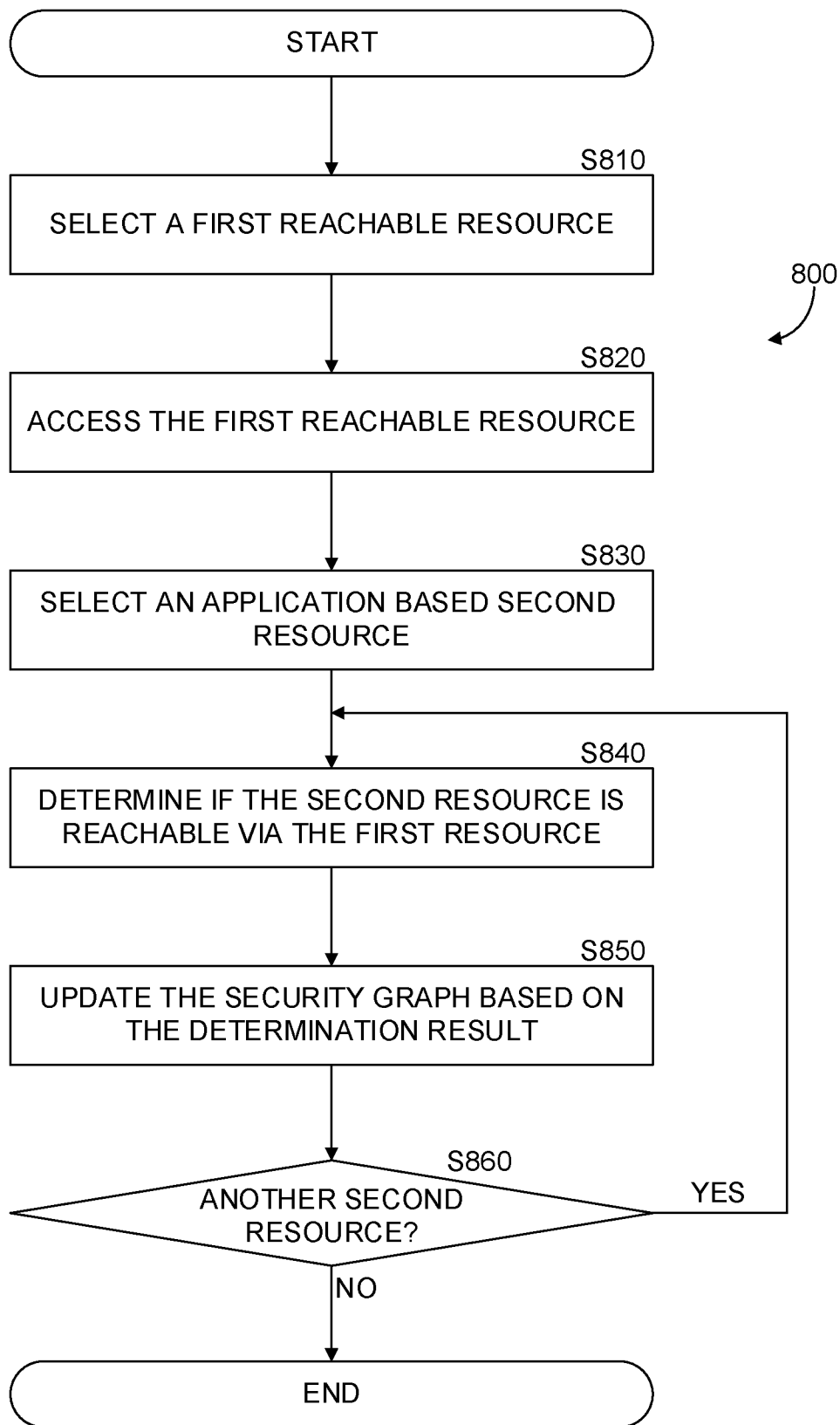


FIG. 8

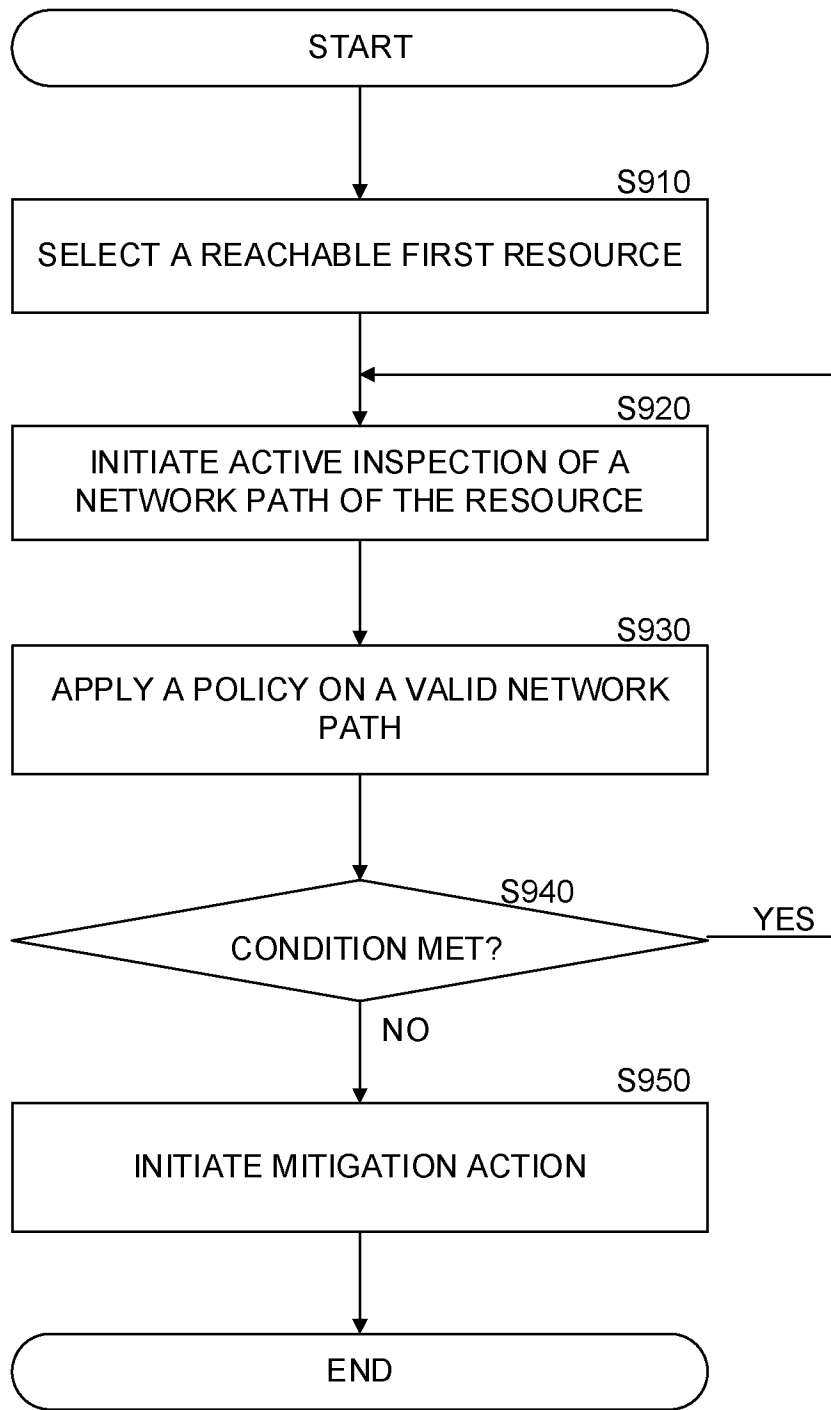


FIG. 9

US 11,936,693 B2

1

**SYSTEM AND METHOD FOR APPLYING A
POLICY ON A NETWORK PATH****CROSS-REFERENCE TO RELATED
APPLICATIONS**

This application is a continuation-in-part of U.S. Non-Provisional patent application Ser. No. 17/659,165 filed on Apr. 13, 2022. This application is also a continuation-in-part of U.S. Non-Provisional patent application Ser. No. 17/659,163 filed on Apr. 13, 2022. This application is further a continuation-in-part of U.S. Non-Provisional patent application Ser. No. 17/659,164, filed Apr. 13, 2022, and a continuation-in-part of U.S. Non-Provisional patent application Ser. No. 17/818,898 filed Aug. 10, 2022. The contents of all the above referenced applications are hereby incorporated by reference.

TECHNICAL FIELD

The present disclosure relates generally to exposure detection in cloud environments, and specifically to active detection of exposure in cloud environments.

BACKGROUND

External attack surface management (EASM) is a term which for a technology field and best practices which are utilized in cybersecurity to describe what vulnerabilities an organization has within their network infrastructure, which may include cloud computing environments, local network environments, and the like. For example, an organization may have a virtual private cloud (VPC) implemented in Amazon® Web Services (AWS), Microsoft® Azure, Google® Cloud Platform (GCP), and the like, which serves as a cloud computing environment. The cloud computing environment may include a plurality of workloads, such as virtual machines, container engines, serverless functions, and the like, any of which may pose a security risk, for example by having a vulnerability, allowing an attacker to infiltrate the organization's network in an unintended manner.

EASM technologies aim to discover where an organization is vulnerable, in order for a network administrator to secure the discovered vulnerabilities. For example, discovering an out-of-date operating system (OS) having a known vulnerability running on a virtual machine may require the network administrator to update the OS version, or apply a software patch, in order to address the vulnerability. This is also known as minimizing the external attack surface.

One such technology which may be deployed in order to discover the external attack surface is known as active scanning. Active scanning attempts to infiltrate a network (e.g., access resources in the above mentioned VPC). For example, by sending packets to endpoints in the network. Thus, an active scanner may attempt to access random domains, at random ports, in order to gain access to a network or to a network resource.

This method has some serious drawbacks. For example, attempting to guess random domains, random ports, and the like, creates a large volume of network traffic which the target (i.e., organization's network) must deal with. This may congest the network, and further risks malfunctions, such as a denial of service to other clients, data corruption from incompatible queries, and the like. It is often of upmost importance to an organization to keep a production environment in a fully operational state. Therefore, using an

2

active scanner to test accessibility of an active production environment may be detrimental to this objective, since it would require devotion of substantial resources at least in terms of network bandwidth to perform such tests.

5 A cloud computing environment may limit the number of open ports the network provides, however in practice that limitation is often limited, opting instead to limit port access at the resource level. Thus, in order to discover if a resource includes an open port, a port scan may be utilized. A port scan involves determining what ports in a network, or network element (such as the resource), are open. An open port receives information. Port numbers range from 0 to 65535, thus performing a full scan for each port, to determine if the port of a particular resource is open, is computationally intensive.

15 It would therefore be advantageous to provide a solution that would overcome at least the challenges noted above.

SUMMARY

20 A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

25 A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

30 In one general aspect, method may include selecting a reachable resource having a network path to access the reachable resource, where the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment. Method may also include actively inspecting the network path to determine if the network path of the reachable resource is accessible from the external network. Method may furthermore include applying a policy on the accessible network path, where the policy includes a conditional rule. Method may in addition include initiating a mitigation action, in response to determining that the conditional rule is not met. Method may moreover include applying the policy on another network path, in response to determining that the conditional rule is met. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

35 Implementations may include one or more of the following features. Method may include: initiating active inspection for each network path of a plurality of network paths; storing an indicator to indicate that a first network path of the plurality of network paths is a valid path, in response to

US 11,936,693 B2

3

determining that the reachable resource is accessible from the external network; and applying the policy on the first network path. Method may include: initiating active inspection by generating an access instruction for the reachable resource; and executing the access instruction on a network path. Method may include: applying the policy only on each network path of the plurality of network paths which is a valid path. Method may include: initiating active inspection on each network path of the plurality of network paths to determine if the network path is a valid network path. Method may include: initiating the mitigation action on the reachable resource. Method where the mitigation action includes any one of: revoking access to the reachable resource, revoking access from the reachable resource, closing a port of the reachable resource, generating a notification, generating an alert, and any combination thereof. Method may include: generating an exception to the policy based on a valid network path. Method may include: initiating active inspection of the network path to detect an application path. Method may include: generating the mitigation action further based on the application path. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

In one general aspect, non-transitory computer-readable medium may include one or more instructions that, when executed by one or more processors of a device, cause the device to: select a reachable resource having a network path to access the reachable resource, where the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment. Medium may furthermore include one or more instructions that, when executed by one or more processors of a device, cause the device to: actively inspect the network path to determine if the network path of the reachable resource is accessible from the external network. Medium may in addition include one or more instructions that, when executed by one or more processors of a device, cause the device to: apply a policy on the accessible network path, where the policy includes a conditional rule. Medium may moreover include initiating a mitigation action, in response to determining that the conditional rule is not met. Medium may also include one or more instructions that, when executed by one or more processors of a device, cause the device to: apply the policy on another network path, in response to determining that the conditional rule is met. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

In one general aspect, system may include a processing circuitry. System may also include a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: select a reachable resource having a network path to access the reachable resource, where the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment. System may in addition include instructions that, when executed by the processing circuitry, configure the system to: actively inspect the network path to determine if the network path of the reachable resource is accessible from the external network. System may moreover include instructions that, when executed by the processing circuitry, configure the system to: apply a policy on the accessible network path, where the policy includes a conditional rule. System may also include instructions that,

4

when executed by the processing circuitry, configure the system to: initiate a mitigation action, in response to determining that the conditional rule is not met. System may furthermore include instructions that, when executed by the processing circuitry, configure the system to: apply the policy on another network path, in response to determining that the conditional rule is met. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: initiate active inspection for each network path of a plurality of network paths; store an indicator to indicate that a first network path of the plurality of network paths is a valid path, in response to determining that the reachable resource is accessible from the external network; and apply the policy on the first network path. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: initiate active inspection by generating an access instruction for the reachable resource; and execute the access instruction on a network path. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: apply the policy only on each network path of the plurality of network paths which is a valid path. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: initiate active inspection on each network path of the plurality of network paths to determine if the network path is a valid network path. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: initiate the mitigation action on the reachable resource. System where the mitigation action includes any one of: revoking access to the reachable resource, revoking access from the reachable resource, closing a port of the reachable resource, generating a notification, generating an alert, and any combination thereof. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate an exception to the policy based on a valid network path. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: initiate active inspection of the network path to detect an application path. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate the mitigation action further based on the application path. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. 1 is a diagram of a cloud computing environment monitored by an active inspector, implemented in accordance with an embodiment.

US 11,936,693 B2

5

FIG. 2 is a security graph illustrating a network path, implemented in accordance with an embodiment.

FIG. 3 is a flowchart of a method for performing active inspection of a cloud computing environment, implemented in accordance with an embodiment.

FIG. 4A is a flowchart depicting a method for determining reachable properties of security objects, according to an embodiment.

FIG. 4B is a flowchart depicting the analysis of a network path to determine reachable properties of objects included in the path, according to an embodiment.

FIG. 5 is a screenshot generated by an active inspector, implemented in accordance with an embodiment.

FIG. 6 is a schematic diagram of an active inspector according to an embodiment.

FIG. 7 is a flowchart of a method for detecting a technology stack utilizing active inspection, implemented in accordance with an embodiment.

FIG. 8 is a flowchart of a method for detecting application paths, according to an embodiment.

FIG. 9 is a flowchart of a method for applying a policy on a network path, implemented in accordance with an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The various disclosed embodiments include a system and a method for applying a policy on a network path. According to an embodiment,

Active inspection allows to generate instructions directed at the reachable resource over a network path, which when received by the reachable resource cause the reachable resource to send a response. For example, a reachable resource may have indicated that port 22 is an open port. Port 22 is typically associated with secure shell (SSH) protocol. In certain embodiments, a port may be associated with a plurality of applications, each application associated with an access instruction. An SSH session may be initiated with the reachable resource having an IP address at 10.0.0.256 on port 22, by providing an access instruction including “ssh user@10.0.0.256:22”.

Discovering an application using the methods described herein allows to validate a tech stack as part of an external attack surface management (EASM) procedure in an efficient way, since only ports which are indicated, for example by static analysis, as being open are inspected to determine if access may occur. In an embodiment, only predetermined applications are checked for each open port, which further reduces the number of access instructions required to probe open ports, as the alternative is to test each probe for each application, which can easily result in hundreds of thousands of access instructions required to determine what applications are deployed on just a single resource.

Various techniques of static analysis can be used in order to determine reachability properties of a resource deployed in a cloud computing environment. Reachability properties, or parameters, may be utilized to establish a network path to

6

the resource from an external network through the cloud computing environment. An access instruction may be generated based on the network path to determine if a network path generated through static analysis is indeed a viable path to reach the resource. Determining what network paths are viable is advantageous as it exposes what network paths can be used to access the cloud computing environment from external networks, and therefore what parts of the cloud computing environment are in practice opened to attack. These network paths should be addressed by system administrators as early as possible to minimize the effect of a cyber-attack.

Another aspect of the disclosed embodiments allows discovering a technology stack on a resource deployed in a cloud computing environment. In an embodiment a technology stack (or “tech stack”) includes a collection of software files, such as an application, an operating system, a driver, a file system, and the like which are installed or deployed on a resource, such as a virtual machine, which in turn is deployed in a cloud computing environment. For example, Nginx® deployed on a virtual machine having a Linux® operating system has a technology stack (also referred to as a tech stack) which includes Nginx and Linux. In an embodiment a reachable resource is selected. A network protocol is detected for the reachable resource. In some embodiments, a security graph representing the cloud computing environment is traversed to determine what network protocol is available on the reachable resource. A network protocol may be, for example, a TCP protocol, UDP protocol, and the like. The network protocol may include an open port, on which an application of the reachable resource is listening. An open port does not necessarily indicate though what application is listening on that port.

FIG. 1 is an example diagram 100 of a cloud computing environment monitored by an active inspector, implemented in accordance with an embodiment. A first cloud environment 110 includes a plurality of principals and resources. A resource is a cloud entity which supplies functionality, such as processing power, memory, storage, communication, and the like. A resource may supply more than one functionality. Resources may include, for example, virtual machines (VMs) such as VMs 113, container engines such as container engines 115, serverless functions such as serverless functions 117, and the like. A VM may be implemented using Oracle® VirtualBox. A container engine may be implemented using Kubernetes® or Docker®. A serverless function may be implemented using Lambda®.

A principal is a cloud entity which acts on a resource, meaning it can request, or otherwise initiate, actions or operations in the cloud environment which cause a resource to perform a function. A principal may be, for example, a user account such as user account 112, a service account such as service account 114, a role, and the like. In an embodiment a user account 112 is implemented as a data structure which includes information about an entity, such as username, a password hash, an associated role, and the like.

The first cloud environment 110 may be implemented utilizing a cloud infrastructure, such as Amazon® Web Services (AWS), Microsoft® Azure, Google® Cloud Platform (GCP), and the like. In an embodiment, the first cloud environment 110 may be implemented as a virtual private cloud (VPC) on such a cloud infrastructure. The first cloud environment 110 may be, for example, a production environment for an organization. A production environment is a computing environment which provides services, for example, to client devices within the production environment and outside of it. An organization may also have a

staging environment, which is a computing environment substantially identical to the production environment in at least some deployments of resource (e.g., workloads) which is used for the purpose of testing new policies, new permissions, new applications, new appliances, new resources, and the like, which are not present in the production environment.

It is often of utmost importance to an organization to keep the production environment in a fully operational state. Therefore, using an active scanner to test accessibility to the first cloud environment **110** may be detrimental to this objective, since it would require devotion of substantial resources at least in terms of network bandwidth to perform such tests.

An inspection environment **120** is communicatively connected with the first cloud environment **110**, and a public network **130**. The public network **130** is also communicatively connected with the first cloud environment **110**. In an embodiment, the public network **120** may be, but is not limited to, a wireless, cellular or wired network, a local area network (LAN), a wide area network (WAN), a metro area network (MAN), the Internet, the worldwide web (WWW), similar networks, and any combination thereof.

The inspection environment **120** may be implemented as a VPC in a cloud infrastructure. In an embodiment, the cloud infrastructure of the inspection environment **120** may be the same cloud infrastructure as the first cloud environment **110**. In some embodiments, the inspection environment may be implemented as multiple cloud environments, each utilizing a cloud infrastructure. The inspection environment includes a security graph database (DB) **122** for storing a security graph, and at least an active inspector **125**.

In an embodiment, the security graph stored in the security graph DB **122** represents at least the first cloud environment **110** using a predefined data schema. For example, each resource and each principal of the first cloud environment **110** may be represented as a corresponding resource node or principal node in the security graph. The various nodes in the security graph may be connected, for example, based on policies, roles, permissions, and the like, which are detected in the first cloud environment **110**. A predefined data schema may include data structures including into which values can be inputted to represent a specific cloud entity. For example, a resource may be represented by a template data structure which includes data attributes, whose values uniquely identify the resource, such as address, name, type, OS version, and the like.

The active inspector **125** is configured to receive a network path to access a resource in the first cloud environment **110**. In an embodiment, a network path may be stored as a data string which includes one or more reachability parameters. Such parameters include host names, protocols, IP addresses, ports, usernames, passwords, and the like. In certain embodiments, the active inspector **125** is further configured to receive a list of network paths. The network paths may be received periodically. In certain embodiments, the active inspector **125** is also configured to generate an instruction which includes a query for the security graph, such instruction or instructions when executed by the security graph database **122** cause(s) generation of an output including one or more network paths. For example, network paths may be generated every 24 hours, while active inspection may occur once per day, once per week, once per month, and so on.

An example of a static analysis process for generating network paths, also known as determining reachability to a resource, is discussed in more detail in U.S. Pat. No.

11,374,982, the contents of which are hereby incorporated by reference herein. In an embodiment, the active inspector **125** may generate an instruction based on the network path to access the resource associated with the network path. For example, the instruction may be to send a data packet to an IP address of the resource, and receive an acknowledgement (ACK) response. The active inspector **125** may generate a log which includes, for example, the network path, the instruction sent by the active inspector **125**, and any response(s) received from the resource. For example, if the active inspector **125** sends an HTTP (hypertext transfer protocol) request, a response may be a 404 error, a 403 error, 500 error, 502 error, and the like.

In an embodiment the active inspector **125** initiates active inspection of a network path to determine if a resource is accessible via the network path from a network which is external to the first cloud environment **110**.

In some embodiments, the first cloud environment **110** further includes a policy engine **116**. In certain embodiments, the policy engine **116** is implemented in the inspection environment **120**. In an embodiment, the policy engine **116** includes a plurality of policies. In some embodiments, a policy includes a statement, a conditional rule, a combination thereof, and the like. In an embodiment, the policy engine **116** is configured to apply a policy on a network path, an application path, a combination thereof, and the like. In some embodiments, the active inspector **125** is configured to access a policy stored on the policy engine **116** and apply the policy on a network path, a valid network path, and the like.

FIG. 2 is an example of a security graph **200** illustrating a network path, implemented in accordance with an embodiment. The security graph **200** includes a plurality of nodes, each node connected to at least another node by an edge. In certain embodiments, a pair of nodes may be connected by a plurality of edges. In some embodiments, each edge may indicate a type of connection between the nodes. For example, an edge may indicate a "can access," to indicate that a cloud entity represented by a first node can access the cloud entity represented by a second node.

A first enrichment node **210** (also referred to as public network node **210**) represents a public network, such as public network **130** of FIG. 1 above. An enrichment node, such as enrichment node **210**, is a node generated based off of insights determined from data collected from a computing environment, such as the first cloud computing environment **110** of FIG. 1 above. An enrichment node may also represent, for example, a vulnerability. By connecting resource nodes in the graph to the enrichment node representing a vulnerability, the security graph **200** may indicate that the resources contain the vulnerability. This allows a compact representation as the security graph does not redundantly store multiple data fields of the same vulnerability in each resource node.

The public network node **210** is connected to a first resource node **220** (also referred to as firewall node **220**) representing a firewall workload. The firewall represented by the firewall node **220** may be implemented, for example, as a virtual machine in the first cloud computing environment. Connecting the public network node **210** to the firewall node **220** represents that the firewall is open to transceiving communication between itself and the public network.

The firewall node **220** is further connected to a second resource node **230** (also referred to as API gateway node **230**) which represents an API (application programming interface) gateway. An API gateway is a workload, for example a serverless function, which can act as a reverse

proxy between a client and resources, accepting API calls, directing them to the appropriate service, workload, resource, etc. and returning a result to the client when appropriate.

The API gateway node **230** is connected to a first principal node **240** (also referred to as VM node **240**) representing a virtual machine hosting an application and a database, and is also connected to a second principal node **250** (also referred to as container engine node **250**) which hosts a plurality of container nodes. The VM node **240** is connected to an application node **242**, and a database node **244**. The application node **242** may indicate, for example, that a certain application, having a version number, binaries, files, libraries, and the like, is executed on the VM which is represented by the VM node **240**.

In an embodiment, the VM node **240** may be connected to a plurality of application nodes. The database node **244** represents a database which is stored on the VM (represented by VM node **240**) or stored on a storage accessible by the VM. The database node **244** may include attributes which define a database, such as type (graph, columnar, distributed, etc.), version number, query language, access policy, and the like.

FIG. 3 is an example flowchart **300A** of a method for performing active inspection of a cloud computing environment, implemented in accordance with an embodiment.

At **S310**, at least one network path for a first resource in a cloud computing environment is received. The network path, also known as object reachability, includes data (e.g. reachability parameters) for accessing the first resource from a public network, which is not the cloud computing environment of the first resource, such as the Internet. In an embodiment, an active inspector may receive the at least a network path, for example from a security graph. In an embodiment, **S320** includes generating an instruction (or instructions) which when executed by a database system storing the security graph return a result of one or more resources, and a respective network path for each of the one or more resources. In certain embodiments, the network paths may be received periodically.

In some embodiments, the first resource may be one of a plurality of first resources, which are each substantially identical. For example, a group of virtual machines which are generated based on the same code or image are substantially identical, since their initial deployment would be identical other than a unique identifier assigned to each machine. In such embodiments it may be beneficial to inspect the at least one network path for a subset of the plurality of first resources, in order to decrease the computation and network resources required. This may be acceptable in such embodiments, as the expectation is that the plurality of VMs would be accessible in similar network paths. In some embodiments, the subset includes one or more first resources.

In an embodiment, each of the received network paths includes a set of reachability parameters to reach a specific cloud object in the cloud environment. The reachability parameters, and hence the network paths are generated by statically analyzing the cloud environment. An example method for such static analysis is described with reference to FIGS. 4A and 4B below.

At **S320**, an access instruction is generated to access the first resource based on the network path. In an embodiment, the access instruction is generated by the active inspector deployed outside of the cloud environment where the first resource resides. In certain embodiments, the instruction includes one or more access parameters. Such parameters

may include, but are not limited to, a host name, an IP address, a communication protocol, a port, a username, a password, and the like, or combination thereof. A communication protocol may be, for example, HTTP or UDP (user datagram protocol). For example, the instruction may be a ping, GET, CONNECT, or TRACE request over HTTP.

In certain embodiments, a plurality of access instructions may be generated. For example, a plurality of generated access instructions may include a first access instruction having a first request, and a second access instruction having a second request which is different from the first request. For example, the first access instruction may include a CONNECT request, and the second access instruction may include a GET request. In certain embodiments, a plurality of first access instructions may be generated. In such embodiments, each first access instruction may include a same type of request (e.g., CONNECT) with different values (e.g., different web address, different port, and so on). For example, a resource may be reachable at IP address 10.0.0.127, at ports 800 through 805. The IP address and ports would be reachability parameters, based on which an active inspector can generate a plurality of first access instructions based on an HTTP GET request, such as:

```
GET /bin HTTP/1.1
```

```
Host:10.0.0.127:800
```

and further generate another HTTP GET request:

```
GET /bin HTTP/1.1
```

```
Host:10.0.0.127:801
```

and so on, which when executed attempt to access a /bin folder in the resource which has an IP address of 10.0.0.127. In certain embodiments, the active inspector (e.g., the active inspector **125** of FIG. 1) may connect to a proxy server (not shown) through the public network **130**, and send a first access instruction to a resource in the cloud environment **110** through a first proxy server, and send a second access instruction (which may or may not be identical to the first access instruction) through a second proxy server. In such embodiments, each proxy server may show as originating from a different country of origin, therefore the source would receive access requests from seemingly different sources. This is advantageous to determine, for example, if a resource is configured to block certain network traffic based on geographic location.

At **S330**, execution of the generated access instruction is caused. The access instruction, when executed, causes an attempt to actually access the resource. In an embodiment, the attempt may result in network traffic being generated, including requests sent to the resource and answers (i.e., data packets) received. While static analysis provides a possible path to access a resource, executing the access instruction provides a real result of an attempt to utilize the possible path, in order to determine which paths are really viable, and which are not. For example, a path may be possible based on static analysis, but not viable, where, for example, an application deployed on the resource prevents such an access from occurring. In an embodiment a network path is determined to be viable (or accessible), if the access instruction, when executed does not return an error message. An error message may be, for example, a timeout (e.g., in response to a "ping" request), a 403 Forbidden (e.g., in response to an HTTP GET request), and the like. In some embodiments, the access instruction may be executed by the active inspector **125**.

At **S340**, a determination is performed to determine if the network path is accessible, based on the execution of the generated access instruction. Performing an active inspection of a cloud environment allows to determine which of the

US 11,936,693 B2

11

reachability paths (i.e., network paths) are indeed vulnerable, meaning that paths that can be used to gain access into the cloud environment, and which reachability paths (network paths) are not vulnerabilities since the active inspector could not gain access to the resource, therefore the reachability path is not possible in practice. Reachability paths which have been confirmed through both static analysis (i.e., analysis using the security graph) and active inspection are paths which should therefore be considered more vulnerable. In an embodiment, if the network path results in successfully reaching the resource, the network path is determined to be accessible (or viable). If the resource is not reachable by the network path, the network path is determined to be inaccessible (or unviable).

At S350, a security graph is updated based on the network path determination. In certain embodiments, the active inspector may update the security graph, which includes a representation of the cloud environment in which the first resource is deployed, to indicate whether a reachability path is confirmed (i.e., is viable) by active inspection or not, where a confirmed path is a path through which the active inspector successfully accessed a resource. In turn, the security graph may update an alert generated based on determining that a resource has a reachability path through a public network.

At S360, a report is generated based on the execution of the generated instruction. In an embodiment, the report may be generated by the active inspector, which performs this method. In certain embodiments, generating a report may include updating a log with network traffic between the active inspector and the resource. For example, the active inspector may record (e.g., write to a log) the generated instruction, the resource identifier, and a response received from the resource. A response may include, for example, a response code. A response code may indicate success, redirection, client error, server error, and the like, where the client is the active inspector, and the server is the resource. In certain embodiments the security graph stored in the security DB 122 may be updated based on the determined viability of the network paths. For example, if a resource is successfully accessed, or successfully un-accessed (i.e., an attempt was made to access the resource and the attempt was not successful in accessing the resource), this result can be stored as an attribute of a node representing the resource in the security graph. For example, the VM node 240 of FIG. 2 may have an attribute which indicates a reachability status, which may have values corresponding to: successfully reached (i.e., an active inspector successfully accessed this resource), successfully not reach (i.e., an active inspector was not successful in accessing this resource), and undetermined (the active inspector has not yet attempted to access the resource through a network path). In some embodiments, certain network paths may be determined (i.e., as viable or unviable) while others may be undetermined. A node may be associated with a plurality of network paths, each having its own active inspection indicator.

In some embodiments, the active inspector may communicate with a virtual private network (VPN) or a proxy, in order to mask the IP address from which the active inspector is attempting access. This may be useful to test, for example, if a firewall, such as represented by the firewall node 220 of FIG. 2, will let communication through based on blocking or allowing certain IP addresses. In such embodiments, multiple similar instructions may be generated, each originating from a different IP address of the active inspector.

In some embodiments network path may include a plurality of resources. The method above may be performed on

12

each resource of the plurality of resources, to determine the reachability of each resource.

Utilizing an active inspector using network paths generated from a security graph is advantageous, as attempting to access resources in this manner to determine the viability of a network path (i.e., reachability) requires less resources than, for example, randomly guessing network paths in an attempt to access resources.

In certain embodiments the active inspector may generate a screenshot of a user interface used to access the resource through the network path. FIG. 5 below is one such example of a screenshot of a user interface, implemented in accordance with an embodiment.

Furthermore, utilizing the active inspector to validate network paths and updating the security graph with the results allows to detect workloads which both contain a vulnerability, and have a validated network path. This allows generating an alert to a user of the cloud environment in order to address such problems by accurately characterizing cybersecurity threats. This in turn allows to utilize resources more efficiently, since the most vulnerable gaps in the cloud environment will be addressed first.

FIG. 4A is an example flowchart 400 depicting a method for determining reachable properties of security objects, according to an embodiment. A reachable property defines if and how an object on the generated security graph can be reached from an external or internal network, and/or an external or internal object. External means outside of the cloud environment of an organization. An object may be any computing or network object designated in a security graph generated as discussed above.

At S405, a security graph is accessed or otherwise obtained from the graph database. Within a security graph, various objects or entities, as may be included in a network or cloud environment of an organization, may be represented as “nodes” or “vertices,” and such “nodes” or “vertices” may be interconnected by one or more “links” or “edges,” the “links” or “edges” representing the relationships between the various objects included in a network or environment. Each object in the graph may be associated with known properties of the object. Examples for such properties may include an object’s name, its IP address, various predefined security rules or access rules, and the like.

At S410, possible network paths within the obtained security graph are identified. A network path is a connection of two or more security objects accessible from an external or internal network, and/or an external or internal object. That is, a network path may include sequential representations of possible data/control flows between two or more objects in a graph. In an embodiment, where two objects in a graph are represented as vertices, and where the vertices are joined by an edge, a path may be constructed between the two vertices. A path may be a vertex-only path, describing a sequence of vertex-to-vertex “hops,” an edge-only path, describing only the edges included in the sequence without description of the associated vertices, or a combined edge-vertex path, describing both edges and vertexes included in the sequence.

According to disclosed embodiments, a path shows a connection between security objects and/or computing objects that communicate over a network. An object may be a virtual, physical, or logical entity.

In an embodiment, paths can be identified by traversing the security graph. The traversal can start or end at objects that are connected to an external network (the internet). The traversal of the security graph can be performed using

solutions disclosed in the related art, e.g., a breadth-first search (BFS), a tree traversal, and the like, as well as any combination thereof.

In another embodiment, paths can be identified by querying the graph database storing the security graph. Examples of applicable queries include, without limitation, queries configured to identify all paths between a first graph object (node) and a second graph object, queries configured to identify all paths between all graph vertices of a first object type and all graph vertices of a second object type, other, like, queries, and any combination thereof.

Following as performed at S410 through S430, the list of paths are iteratively identified to determine the reachability properties of the path. Specifically, at S415, a path list is populated to include all identified paths. A path list may be a table, list, or other type of data structure. A path list may be unordered or ordered, including ordering according to one or more path properties.

At S420, a path from the path list is selected. At a first run of the method a first path in the list is selected.

At S425, path elements are analyzed to determine reachable properties. Path element analysis, as at S425, is an iterative analysis of each element included in the path selected at S420. The operation of S425 is discussed in detail with reference to FIG. 4B.

At S430, it is determined whether the last path of the path list has been analyzed, and if so, execution terminates; otherwise, execution returns to S420.

FIG. 4B is an example flowchart S425 depicting the analysis of a network path to determine reachable properties of objects included in the path, according to an embodiment.

At S455, elements within a selected network path are identified. Elements are network and/or computing objects and relationships (or connections) between such objects. Identification of elements within the selected path may include, without limitation, identification based on properties, and other, like, data, included in the elements, identification of elements based on element identifications provided during the execution of S410 of FIG. 4A, above, and the like, as well as any combination thereof. Further, identification of in-path elements may include identification of element properties or attributes including, without limitation, names, network addresses, rulesets, port configurations, and the like, as well as any combination thereof.

Then, at S460 through S480, the list of paths are iteratively processed in order to determine reachable properties of the elements. Specifically, at S460, the next element is selected. The next element is a subsequent element of the set of elements, within the selected path, identified at S455. Where execution of S460 follows the execution of S480, the next element may be an element which, in the selected network path, immediately follows the element relevant to the preceding execution of S470 and S475. Where execution of the method described with respect to FIG. 4B includes a first execution of S460, the first execution of S460 may include the selection of a first element of the selected path.

For exemplary purposes, a network path may be a path from a virtual machine (VM), connected to a NIC, connected to a load balancer, connected to a firewall. According to a first example, where S460 is executed for the first time, the first execution of S460 may include the selection of the VM as the selected element. Further, according to a second example, where execution of S460 follows execution of S480, selection of a next element at S460 may include selection of, following the VM, selection of the NIC, or, following the NIC, selection of the load balancer, or, following the load balancer, selection of the firewall.

At S465, it is determined whether the selected element has been analyzed. Determination of whether the selected element may include the determination of whether the selected element may include the determination of whether the selected element may include the determination of whether the selected element. As execution of S475 provides for the population of reachable properties into the security graph, an element which does not include such reachable properties in the graph may be assumed to have not been analyzed.

Where, at S465, it is determined that the selected element has been analyzed, execution continues with S460. Where, at S465, it is determined that the selected element has not been analyzed, execution continues with S470.

At S470, reachable properties are determined. Reachable properties are object properties describing if, and how, a given path element is reachable through the selected path, and, specifically, from an external network, an internal network, both, and a combination thereof. Examples of reachable properties include, without limitation, binary properties describing whether an element is reachable, protocols by which the element is reachable, network addresses at which an element is reachable, ports by which an element is reachable, access rules, and the like, as well as any combination thereof.

In an embodiment, a reachable property is determined as a minimal set of reachable properties of all other objects in the path. As a simple example, if a path includes two objects, where one object can receive traffic from any source IP address through port 1515, and the other object can receive traffic only from a source IP address of 173.54.189.188, the reachable property of the second object may be that the second object is reachable through “source IP address 173.54.189.188 and port 1515.”

At S475, reachable properties are populated into the security graph. Reachable properties, as may be determined at S470, may be populated into the graph by processes including, without limitation, labeling or tagging graph vertices (or “nodes”), updating network or graph object properties, generating one or more graph overviews, layers, or graph-adjacent data features, and the like, as well as any combination thereof.

In an embodiment, population of reachable properties into the security graph may include, for each object, population of object network access control lists (NACLs) as described hereinbelow, into the security graph elements corresponding with the various path elements, as well as the population of scope specific NACLs, and other, like, properties into the graph. Scope-specific NACLs are NACLs describing object, path, or network accessibility properties specific to a given scope, where a given scope may be the internet, various given accounts, various given environments, and the like. Scope-specific NACLs may, for example, describe the properties of an object with respect to the object’s internet accessibility, where the object may be configured to include different access control properties for internet access and local intranet access.

Further, population of reachable properties into the graph may include population of one or more paths into the graph, including by population processes similar or identical to those described with respect to population of individual objects. Population of paths into the graph may include, without limitation, population of one or more paths into the graph, including a presently-analyzed path, population of one or more path properties, and the like, as well as any combination thereof. Path properties, as may be populated to a graph, are properties describing various attributes of a path, including, without limitation, NACLs applicable to path elements, path segments, or full paths, including full-

path aggregate NACLs, and the like, as well as any combination thereof. Further, population of path properties into the graph may include the population of one or more scope-specific path properties, where such scope-specific path properties may be properties relevant to specific scopes, such as those described herein.

Where population of reachable properties includes labeling or tagging a graph, or elements thereof, one or more graph vertices or edges, the corresponding objects or relationships, or both, may be labeled, tagged, or otherwise associated with one or more data features describing relevant reachable properties. In addition, where population of reachable properties to the graph includes updating graph objects, graph vertices and edges, the corresponding objects and relationships, or both, may be directly updated to explicitly include the calculated properties.

Further, where population of reachable properties includes the generation of one or more graph layers or overlays, the generated graph layers or overlays may be data features independent of, but corresponding to, the relevant graphs, where the generated overlays or layers may include one or more data features describing the reachable properties of the various graph elements.

At S480, it is determined whether all elements in the selected path have been analyzed. Determination of whether all elements in the selected path have been analyzed may include, without limitation, determination of whether the immediately preceding execution of S475 relates to the last element in the selected path, determination of whether additional elements remain in the path, determination of whether any additional in-path elements have been analyzed, and the like, as well as any combination thereof.

Where, at S480, it is determined that all elements in the selected path have not been analyzed, execution continues with S460. Where, at S480, it is determined that all elements in the selected path have been analyzed, execution terminates.

FIG. 5 is an example of a screenshot 500 generated by an active inspector, implemented in accordance with an embodiment. A screenshot is an image which shows the contents of a computer display. In an embodiment, an active inspector, such as the active inspector 125 of FIG. 1, may include a web browser application for executing access instructions. The web browser application may generate a user interface intended for a display. The screenshot 500 includes a portion of such a user interface, which includes a response header 510 received based on a request to access a resource. In this case the response header 510 includes an HTTP code 403 (i.e., forbidden), meaning that the request to access the resource was denied. A detailed code 512 includes a message which is associated with the 403 code (i.e., "access denied"), a message 514, a request identifier 516, and a host identifier 518.

FIG. 6 is an example schematic diagram of an active inspector 125 according to an embodiment. The active inspector 125 includes a processing circuitry 610 coupled to a memory 620, a storage 630, and a network interface 640. In an embodiment, the components of the active inspector 125 may be communicatively connected via a bus 650.

The processing circuitry 610 may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose

microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory 620 may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof.

In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage 630. In another configuration, the memory 620 is configured to store such software. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the processing circuitry 610, cause the processing circuitry 610 to perform the various processes described herein.

The storage 630 may be magnetic storage, optical storage, and the like, and may be realized, for example, as flash memory or other memory technology, compact disk-read only memory (CD-ROM), Digital Versatile Disks (DVDs), or any other medium which can be used to store the desired information.

The network interface 640 allows the active inspector 125 to communicate with, for example, a cloud environment, a security graph database, resources from the cloud environment, and the like.

It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. 6, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

FIG. 7 is an example flowchart of a method for detecting a technology stack utilizing active inspection, implemented in accordance with an embodiment. Technology stack is a term used to describe application, operating systems, and the like which are installed or deployed on a resource, such as a virtual machine, which in turn is deployed in a cloud computing environment. For example, Nginx® deployed on a virtual machine having a Linux® operating system has a technology stack (also referred to as a tech stack) which includes Nginx and Linux.

At S710, a reachable first resource is selected. In an embodiment, a reachable first resource may be selected from a list of reachable resources. The list may be stored, for example, as a table in a database. The list may include an identifier of each reachable resource, and at least one viable network path. A reachable resource is a resource which is reachable from an external network, in that the external network is external to a cloud computing environment in which the resource is deployed. In an embodiment, a reachable resource includes a network path, reachability parameters, and the like, for example as discussed in more detail above. A reachable resource is a resource which includes at least a viable network path, having reachability parameters which allow access from an external network to the resource, the resource deployed in a cloud computing environment. In an embodiment, a security graph may be queried to generate a result which includes at least a reachable resource. In certain embodiments, the generated result includes a plurality of reachable resources, each having its own at least a viable network path. In other embodiments, the result includes a plurality of network paths for a reachable resource (i.e., the resource is reachable from a plurality of network paths).

US 11,936,693 B2

17

At **S720**, a network protocol is detected for the reachable first resource. In an embodiment, the network protocol is a transport layer protocol. For example, the network protocol may be any one of: a TCP protocol, and a UDP protocol. In an embodiment, the network protocol is any one of: hyper-text transfer protocol (HTTP), file transfer protocol (FTP), secure shell (SSH), simple mail transfer protocol (SMTP), post office protocol (POP3), internet message access protocol (IMAP), internet relay chat (IRC), HTTP secure (HTTPS), and the like. In certain embodiments, a port may be determined for the network protocol. For example, the first reachable resource may be indicated as listening on (or having open) any one of port: 80, 20, 21, 22, 25, 110, 143, 194, 443, and the like. A port is a communication endpoint, and may be implemented as a 16-bit number.

In some embodiments, a security graph may be queried to determine if the reachable first resource includes an open port. A resource will not actively listen on all ports, as this is over sixty-five thousand different port numbers which are possible. In an embodiment, determining that a resource is actively listening on a port may be determined as part of the reachability parameters, for example as detailed above.

An open port is a port which a resource is actively listening for network traffic. While certain ports are considered well known ports (i.e., port 80 is used for HTTP) most are open for general use. Some network ports may be used by multiple applications. For example, port 6600 is used by Microsoft® Hyper-V Live, and by Music Player Daemon, which are two separate applications. Therefore, knowing that a port, or range of ports, is open, does not necessarily indicate what applications are deployed on the first resource.

As another example, port 8080 is used as an alternative for HTTP, used by Apache® Tomcat, and by Atlassian® JIRA applications. Therefore, knowing that a machine is listening on port 8080 does not indicate what applications are present.

At **S730**, an access instruction for accessing an application of the first resource is generated. In an embodiment the access instruction is based on the network protocol. In certain embodiments, accessing the first resource includes providing the first resource with credentials which allow access to the first resource. For example, a private key may allow accessing an SSH server. As another example, an API gateway, such as the API gateway **230** of FIG. **2** may be accessed by providing credentials. As yet another example, a load balancer (first resource) may provide access to a server (second resource), which listens on an application address. For example, an SSH server may be exposed behind a load balancer, listening on a local application address (e.g., 10.0.0.115). By accessing the load balancer, which includes an external network path, and from there accessing the application address, an attacker may gain access to the SSH server. In certain embodiments, the access instruction is generated based on a predetermined application which is associated with a port number. For example, a table may include a port number in a first column, and a network protocol in a corresponding second column.

In the example above, the first column may indicate 8080, the second column HTTP, another second column Tomcat, and the like. In some embodiments, a predetermined instruction may be stored which corresponds to the predetermined application. For example, the predetermined application may be HTTP, and the predetermined instruction may be a GET command. The access instruction may be further generated based on the predetermined instruction.

At **S740**, an application is discovered based on a response to the access instruction. In an embodiment, the access

18

instruction, when executed, configures a computing device to initiate a communication over the network path. A response may be, for example, an ack (acknowledgement), which is received from the first resource in response to the access instruction. In an embodiment, a response to an HTTP request may include a status code, such as 500, 404, 200, 202, and the like. In certain embodiments, a security graph may be updated based on the discovered application.

In an embodiment, an application node may be generated in a security graph to represent the discovered application. The application node may be connected with an edge to a resource node representing the reachable first resource node, on which the application is deployed.

Discovering an application using the methods described herein allows to validate a tech stack as part of an external attack surface management (EASM) procedure in an efficient way, since only ports which are indicated, for example by static analysis, as being open are inspected to determine if access may occur. In an embodiment, only predetermined applications are checked for each open port, which further reduces the number of access instructions required to probe open ports, as the alternative is to test each probe for each application, which can easily result in hundreds of thousands of access instructions required to determine what applications are deployed on just a single resource.

At **S750**, a check is performed to determine if an additional network protocol should be checked. If 'yes' execution may continue at **S720**. Otherwise, execution may terminate. In an embodiment, the check may be further performed based on a response received based on execution of the access instruction. For example, if the access instruction was not successful, a second access instruction may be generated for example for the same port using a different protocol. In an embodiment the access instruction, second access instruction, and the like, are delivered over the network path of the reachable first resource.

FIG. **8** is an example flowchart **800** of a method for detecting application paths, according to an embodiment. An application path is a path an attacker may use when gaining access to a cloud computing environment, which includes a first resource through which the attacker gains access to the cloud computing environments, and subsequent second resources which the attacker is able to access in the cloud computing environment, after gaining access to the first resource, wherein the first resource corresponds to a virtual machine, container, and the like, and the second resource corresponds to an application executed (or deployed) on the first resource. In an embodiment, detecting an application path includes detecting a vulnerability which allows to reach a first resource, gain access to it, and through the first resource gain access to a second resource which is accessible to the first resource, but should not be accessible, for example, to the user account or service account accessing the first resource.

At **S810**, a reachable first resource is selected. In an embodiment, a reachable first resource may be selected from a list of reachable resources. The list may be stored, for example, as a table in a database. The list may include an identifier of each reachable resource, and at least one viable network path. A reachable resource is a resource which is reachable from an external network, in that the external network is external to a cloud computing environment in which the resource is deployed. In an embodiment, a reachable resource includes a network path, reachability parameters, and the like, for example as discussed in more detail above. A reachable resource is a resource which includes at least a viable network path, having reachability parameters

which allow access from an external network to the resource, the resource deployed in a cloud computing environment. In an embodiment, a security graph may be queried to generate a result which includes at least a reachable resource. In certain embodiments, the generated result includes a plurality of reachable resources, each having its own at least a viable network path. In other embodiments, the result includes a plurality of network paths for a reachable resource (i.e., the resource is reachable from a plurality of network paths).

At **S820**, the first resource is accessed. In an embodiment, accessing the first resource includes providing the first resource with credentials which allow access to the first resource. For example, a private key may allow accessing an SSH server. As another example, an API gateway, such as the API gateway **230** of FIG. **2** may be accessed by providing credentials. As yet another example, a load balancer (first resource) may provide access to a server (second resource), which listens on an application address. For example, an SSH server may be exposed behind a load balancer, listening on a local application address (e.g., 10.0.0.115). By accessing the load balancer, which includes an external network path, and from there accessing the application address, an attacker may gain access to the SSH server.

At **S830**, a second resource is selected. In an embodiment, the second resource is an application exposed through the first resource. In an embodiment a security graph is queried to determine the second resource. In some embodiments, the cloud computing environment in which the first resource is deployed is represented in the security graph, for example as detailed in FIG. **2** above. In some embodiments, querying the security graph includes causing a query to be executed on a database hosting the security graph, and receiving as a result an identifier of a node which represents a second resource which is connected to the node representing the first resource.

For example, a security graph may be traversed to detect a node representing an application (application node) which is connected to the node representing the first resource. A second network path may be determined, to the application node from the first resource node. For example, the first resource may be accessed by accessing "example.com:80," while the second resource (i.e., application node) is accessed by using an application address, or other listening address. In an embodiment, an application may be predetermined to be listening on an address, port, and the like. The second network path may be generated based on the predetermined listening.

As another example, the application node may indicate that a web server application (second resource) is deployed on a virtual machine (first resource). An access attempt may include generating an access instruction on the first resource (first network path) using port 80 (second network path), which is a predetermined port used for Internet web traffic.

At **S840**, the second resource is actively determined to be reachable via the first resource. In an embodiment, a second resource is reachable from the first resource, if the first resource can be used to access the second resource. In the example above, if the web server is reachable through the virtual machine, then the web server is reachable. Thus, if the first resource is a reachable resource, meaning that a network path is found which is viable, and the second resource is accessible from the first resource, then an attacker which gains access to the first resource may also gain access to the second resource. Thus, while the second resource may not have a direct viable network path, it can

still become reachable by accessing the first resource, meaning that there is a second network path, which is the network path between the first resource and the second resource. In an embodiment, a resource is accessible if, for example, it can be sent instructions which are then executed by the resource. For example, a SQL database may be determined to be reachable if a network path is determined to a virtual machine hosting the SQL database application, and a second network path allows access to the SQL database (i.e., application path), and further an instruction for performing an SQL injection is generated for execution by the SQL database application.

At optional **S850**, a security graph is updated based on the determination. In some embodiments, a node may include an indicator to indicate if a resource is reachable. In other embodiments, an edge may be added between a node representing the first resource and a node representing the second resource, to indicate that the second resource is reachable from the first resource. Actively inspecting second resources in this manner allows to detect certain vulnerabilities in a cloud computing environment, which is of course desirable.

At **S860**, a check is performed to determine if another second resource should be checked for reachability from the first resource. In an embodiment execution continues at **S840** if another resource should be checked to determine reachability, otherwise execution terminates. For example, if another application is determined to be deployed on the first resource, e.g., by traversing a security graph and detecting another application node connected to the first resource node, a second application path may be determined and reachability thereto may be determined by performing the method detailed herein.

FIG. **9** is an example flowchart of a method for applying a policy on a network path. According to an embodiment, a network path includes resources, protocols, interfaces, combinations thereof, and the like, between a first resource in a cloud computing environment, and another interface, such as an interface of another computing device, an application programming interface (API), a second resource, and the like. In some embodiments, the network path is between the first resource and includes a network portion of a public network, such as the Internet.

According to some embodiments, it is advantageous to determine if a network path is a valid network path, i.e., a network path through which a resource is really reachable, as opposed to an invalid network path, which seems reachable, but in practice cannot be used to access the resource. For example, a resource might be listening on a certain port, which is blocked by a load balancer deployed on the network path of the resource. Therefore, the resource would seem to be accessible through the port, when in practice, it is not.

Therefore, in some embodiments, it is advantageous to determine which network paths, reachability paths, application paths, and the like, are valid paths, for example by performing active inspection of the same, as detailed above.

Furthermore, it is advantageous, in certain embodiments, to apply a policy on a network path, so that certain paths which are viable paths, and should not be viable paths, are addressed as cybersecurity threats. Also, it is further advantageous to apply such a policy, in some embodiments, only on a network path which is a valid network path, for example in order to reduce a number of alerts which are generated by a cybersecurity system, since a network path which is not a viable (or reachable) path is a network path which does not currently pose a cybersecurity threat, according to some embodiments.

US 11,936,693 B2

21

At **S910**, a reachable first resource is selected. In an embodiment, a reachable first resource is selected from a list of reachable resources. In an embodiment, the list is stored, for example, as a table in a database. In some embodiments, the list includes an identifier of each reachable resource, and at least one viable network path.

In certain embodiments, a reachable resource is a resource which is reachable from an external network, in that the external network is external to a cloud computing environment in which the resource is deployed. In an embodiment, a reachable resource includes a network path, reachability parameters, and the like, for example as discussed in more detail above.

According to an embodiment, a reachable resource is a resource which includes at least a viable network path, having reachability parameters which allow access from an external network to the resource, the resource deployed in a cloud computing environment. For example, in an embodiment, the external network is a public network, such as the Internet.

In an embodiment, a security graph is queried to generate a result which includes at least a reachable resource. In certain embodiments, the generated result includes a plurality of reachable resources, each having a respective at least a viable network path. In some embodiments, the result includes a plurality of network paths for a reachable resource (i.e., the resource is reachable from a plurality of network paths).

At **S920**, active inspection of a network path is initiated. In an embodiment, the network path is a network path of the selected reachable first resource. In some embodiments, action inspection of the network path includes the methods described in more detail herein. For example, according to an embodiment, active inspection of the reachable first resource includes providing credentials which allow access to the first resource. For example, a private key may allow accessing an SSH server. As another example, an API gateway, such as the API gateway **230** of FIG. **2** is accessed by providing credentials, according to an embodiment.

As yet another example, a load balancer (first resource) is configured to provide access to a server (second resource), which listens on an application address. For example, an SSH server may be exposed behind a load balancer, listening on a local application address (e.g., 10.0.0.115). By accessing the load balancer, which includes an external network path, and from there accessing the application address, an attacker may gain access to the SSH server.

In an embodiment, active inspection includes generating an instruction which when executed configures a computing device to send a data packet through a network path. In certain embodiments, the data packet, a plurality of data packets, and the like, include an instruction which when executed by the reachable resource configure the resource to generate a response to the instruction.

At **S930**, a policy is applied on a valid network path. In some embodiments, an active inspector is configured to determine that a network path is a valid network path. In certain embodiments, in response to determining that a network path is a valid network path, the active inspector is further configured to apply a policy to the network path.

In some embodiments, applying a policy on a network path includes applying a conditional rule. For example, in some embodiments, applying the policy on a network path includes generating an instruction which, when executed, generates an output on which the conditional rule can be applied.

22

In certain embodiments, the instruction includes an additional active inspection. For example, according to an embodiment, a policy includes a statement that a network path to/from a first reachable resource should only be accessible via a predefined port (e.g., port 80) using a predefined protocol (e.g., HTTPS). In some embodiments, where a viable network path is detected, the policy is applied to determine if the network path adheres to the policy or not.

For example, in such an embodiment, an active inspection instruction is generated which, when executed, configures the active inspector to attempt to access the first reachable resource over the network path, using port 8080 and HTTPS. As another example, the action inspection instruction, when executed, configures the active inspector to attempt to access the first reachable resource over the network path, using port 80 and HTTP (which is the unsecure version of HTTPS).

In some embodiments, a plurality of active inspection instructions are generated and executed, to determine if the policy is adhered to. In some embodiments, a policy includes an exception. For example, in certain embodiments, a certain resource is exempt from a certain policy a priori. In some embodiments, a policy exemption is generated based on a list of predefined resources, resource types, a combination thereof, and the like, which are exempt from a particular policy, a particular policy type, a combination thereof, and the like.

In certain embodiments, active inspection is further performed to detect an application path. In some embodiments, the policy is applied to the application path, the network path, a combination thereof, and the like. For example, in some embodiments, a policy includes a statement such that a valid network path violates the policy, however an application path utilizing the network path, allows to create an exemption for the network path. For example, a policy includes a statement, according to an embodiment, that no resource should allow communication over port 80, unless the communication is specifically for a predefined application which is configured to communicate over this port.

At **S940**, a check is performed to determine if a condition is met. In an embodiment, the condition is a condition of the policy, such as a conditional rule. In some embodiments, the condition includes determining if the policy is adhered to by the network path. In some embodiments, this is determined, for example, as detailed above (e.g., by applying a rule and checking the result of the outcome). Where the condition is satisfied (e.g., the network path adheres to the policy), execution continues at **S920**, according to an embodiment. In certain embodiments, where the condition is not satisfied (e.g., the network path does not adhere to the policy), execution continues at **S950**.

At **S950**, a mitigation action is initiated. In some embodiments, the mitigation action includes generating a policy exemption. In certain embodiments, the mitigation action includes revoking access to the reachable resource, revoking access from the reachable resource, closing a port of the reachable resource, generating a notification, generating an alert, any combination thereof, and the like.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing

US 11,936,693 B2

23

units (“CPUs”), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

It should be understood that any reference to an element herein using a designation such as “first,” “second,” and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

As used herein, the phrase “at least one of” followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including “at least one of A, B, and C,” the system can include A alone; B alone; C alone; 2A; 2B; 2C; 3A; A and B in combination; B and C in combination; A and C in combination; A, B, and C in combination; 2A and C in combination; A, 3B, and 2C in combination; and the like.

What is claimed is:

1. A method for applying a policy on a network path, comprising:
 selecting a reachable resource having a network path to access the reachable resource, wherein the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment; actively inspecting the network path to determine if the network path of the reachable resource is accessible from the external network;
 storing an indicator to indicate that the network path is a valid path, in response to determining that the reachable resource is accessible from the external network;
 applying a policy on the valid path, wherein the policy includes a conditional rule;
 initiating a mitigation action, in response to determining that the conditional rule is not met; and
 applying the policy on another network path, in response to determining that the conditional rule is met.

24

2. The method of claim 1, further comprising:
 initiating active inspection for each network path of a plurality of network paths; and
 applying the policy only on a network path of the plurality of network paths which is a valid path.

3. The method of claim 2, further comprising:
 initiating active inspection by generating an access instruction for the reachable resource; and
 executing the access instruction on a network path.

4. The method of claim 2, further comprising:
 applying the policy only on each network path of the plurality of network paths which is a valid path.

5. The method of claim 4, further comprising:
 initiating active inspection on each network path of the plurality of network paths to determine if the network path is a valid network path.

6. The method of claim 1, further comprising:
 initiating the mitigation action on the reachable resource.

7. The method of claim 6, wherein the mitigation action includes any one of: revoking access to the reachable resource, revoking access from the reachable resource, closing a port of the reachable resource, generating a notification, generating an alert, and any combination thereof.

8. The method of claim 1, further comprising:
 generating an exception to the policy based on a valid network path.

9. The method of claim 1, further comprising:
 initiating active inspection of the network path to detect an application path.

10. The method of claim 9, further comprising:
 generating the mitigation action further based on the application path.

11. A non-transitory computer-readable medium storing a set of instructions for applying a policy on a network path, the set of instructions comprising:
 one or more instructions that, when executed by one or more processors of a device, cause the device to:
 select a reachable resource having a network path to access the reachable resource, wherein the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment; actively inspect the network path to determine if the network path of the reachable resource is accessible from the external network;
 storing an indicator to indicate that the network path is a valid path, in response to determining that the reachable resource is accessible from the external network;
 apply a policy on the valid path, wherein the policy includes a conditional rule;
 initiate a mitigation action, in response to determining that the conditional rule is not met; and
 apply the policy on another network path, in response to determining that the conditional rule is met.

12. A system for applying a policy on a network path comprising:
 a processing circuitry; and
 a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:
 select a reachable resource having a network path to access the reachable resource, wherein the reachable resource is a cloud object deployed in a cloud computing environment, having access to an external network which is external to the cloud computing environment;

US 11,936,693 B2

25

actively inspect the network path to determine if the network path of the reachable resource is accessible from the external network;
store an indicator to indicate that the network path is a valid path, in response to determining that the reachable resource is accessible from the external network;
apply a policy on the valid path, wherein the policy includes a conditional rule;
initiate a mitigation action, in response to determining that the conditional rule is not met; and
apply the policy on another network path, in response to determining that the conditional rule is met.

13. The system of claim 12, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

initiate active inspection for each network path of a plurality of network paths; and
apply the policy only on a network path of the plurality of network paths which is a valid path.

14. The system of claim 13, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

initiate active inspection by generating an access instruction for the reachable resource; and
execute the access instruction on a network path.

15. The system of claim 13, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

apply the policy only on each network path of the plurality of network paths which is a valid path.

26

16. The system of claim 15, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

initiate active inspection on each network path of the plurality of network paths to determine if the network path is a valid network path.

17. The system of claim 12, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

initiate the mitigation action on the reachable resource.

18. The system of claim 17, wherein the mitigation action includes any one of:

revoking access to the reachable resource, revoking access from the reachable resource, closing a port of the reachable resource, generating a notification, generating an alert, and any combination thereof.

19. The system of claim 12, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

generate an exception to the policy based on a valid network path.

20. The system of claim 12, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

initiate active inspection of the network path to detect an application path.

21. The system of claim 20, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

generate the mitigation action further based on the application path.

* * * * *

EXHIBIT D



(12) **United States Patent**
Schindel et al.

(10) **Patent No.:** **US 12,001,549 B1**
 (45) **Date of Patent:** ***Jun. 4, 2024**

(54) **CYBERSECURITY INCIDENT RESPONSE TECHNIQUES UTILIZING ARTIFICIAL INTELLIGENCE**

- (71) Applicant: **Wiz, Inc.**, New York, NY (US)
- (72) Inventors: **Alon Schindel**, Tel Aviv (IL); **Barak Sharoni**, Tel Aviv (IL); **Amitai Cohen**, Kfar Saba (IL); **Ami Luttwak**, Binyamina (IL); **Roy Reznik**, Tel Aviv (IL); **Yinon Costica**, Tel Aviv (IL)
- (73) Assignee: **Wiz, Inc.**, New York, NY (US)
- (*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **18/428,794**
 (22) Filed: **Jan. 31, 2024**

Related U.S. Application Data

- (63) Continuation of application No. 18/466,882, filed on Sep. 14, 2023, which is a continuation-in-part of application No. 18/457,054, filed on Aug. 28, 2023.
- (51) **Int. Cl.**
G06F 21/55 (2013.01)
G06F 16/2452 (2019.01)
- (52) **U.S. Cl.**
 CPC **G06F 21/552** (2013.01); **G06F 16/24522** (2019.01)
- (58) **Field of Classification Search**
 CPC G06F 21/552; G06F 16/24522
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,645,122 B1	2/2014	Di Fabrizio et al.
10,158,677 B1	12/2018	Dicorpo et al.
10,417,346 B2	9/2019	Kim et al.
11,301,502 B1	4/2022	Dijamco et al.
11,663,382 B1	5/2023	Cascioli
11,670,062 B1 *	6/2023	Bhushan G06F 9/451 345/633
11,670,288 B1 *	6/2023	Das G06F 16/24578 704/9
11,675,473 B1 *	6/2023	Breeden G06F 3/04847 715/738
11,675,816 B1 *	6/2023	Chandrasekharan G06F 11/3082 707/737
11,676,072 B1 *	6/2023	Chandrasekharan ... G06F 18/23 706/12
11,676,345 B1 *	6/2023	Bhushan G06T 7/73 705/7.27
11,687,413 B1 *	6/2023	Chen G06F 3/04842 707/649

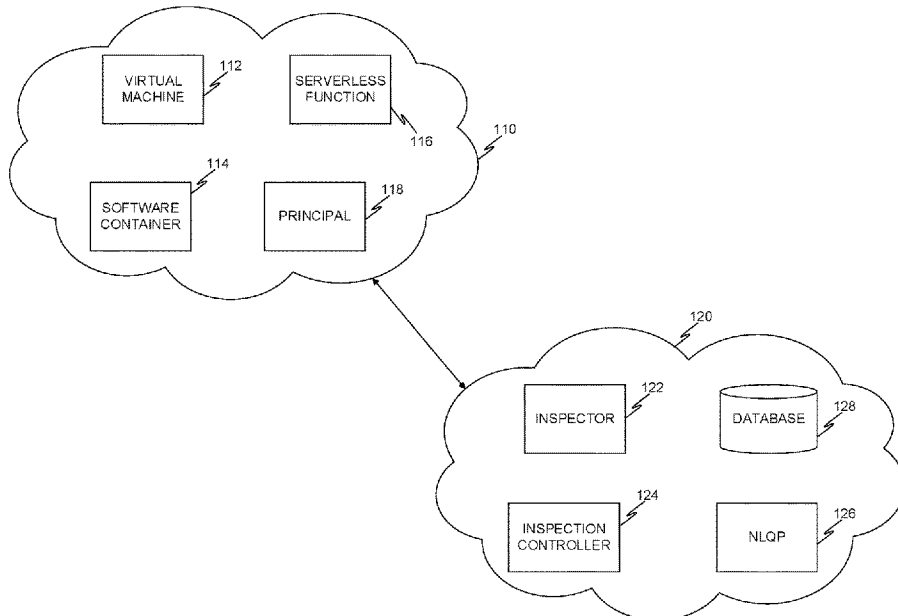
(Continued)

Primary Examiner — Stephen T Gundry
 (74) Attorney, Agent, or Firm — M&B IP Analysts, LLC

(57) **ABSTRACT**

A system and method for providing cybersecurity incident response utilizing a large language model. The method includes: mapping a received incident input into a scenario of a plurality of scenarios, each scenario including a plurality of sub-scenarios; generating a query based on the received incident input and a selection of a sub-scenario of the plurality of sub-scenarios; executing the query on a security database, the security database including a representation of the computing environment; and initiating a mitigation action based on a result of the executed query.

21 Claims, 7 Drawing Sheets



US 12,001,549 B1

Page 2

(56)

References Cited

U.S. PATENT DOCUMENTS

11,693,710 B1 *	7/2023	Aleti	G06F 16/22 707/741	2008/0091681 A1	4/2008	Dwivedi et al.
11,704,219 B1 *	7/2023	Lerner	G06F 11/327 714/57	2013/0018920 A1	1/2013	Griffin
11,714,683 B1 *	8/2023	Roberts	G06F 9/44536 709/223	2014/0095469 A1	4/2014	Chen et al.
11,720,686 B1 *	8/2023	Cross	G06F 21/577 726/25	2018/0062916 A1	3/2018	Eda et al.
11,727,643 B1 *	8/2023	Bhushan	H04L 65/80 345/420	2020/0244700 A1	7/2020	Moon et al.
11,729,074 B1 *	8/2023	Mishra	H04L 43/067 709/224	2020/0349919 A1	11/2020	Wanas et al.
11,734,886 B1 *	8/2023	Bhushan	G06T 17/10 345/420	2022/0229832 A1	7/2022	Li et al.
11,741,131 B1 *	8/2023	Dwivedi	G06F 16/2474 707/758	2022/0382752 A1	12/2022	Yadav et al.
11,755,405 B1 *	9/2023	Satish	G06F 11/0769 714/57	2023/0061234 A1	3/2023	Calado et al.
11,762,869 B1 *	9/2023	Werner	G06T 11/206 707/722	2023/0222029 A1 *	7/2023	Vutukuru G06N 20/10 714/48
11,860,914 B1	1/2024	Qadrud-Din et al.		2023/0224324 A1 *	7/2023	Karabey H04L 63/1416 726/23
11,861,320 B1	1/2024	Gajek et al.		2023/0224377 A1 *	7/2023	Bathla H04L 67/1097 726/22
11,861,321 B1	1/2024	O'Kelly et al.		2023/0252140 A1 *	8/2023	Coulter G06F 21/554 726/23
11,928,569 B1	3/2024	Douthit		2023/0274086 A1	8/2023	Tunstall-Pedoe et al.
				2023/0274094 A1	8/2023	Tunstall-Pedoe et al.
				2023/0316001 A1	10/2023	Araki
				2023/0319074 A1	10/2023	Murphy et al.
				2023/0319097 A1	10/2023	Murphy et al.
				2023/0325725 A1	10/2023	Lester et al.
				2023/0351026 A1 *	11/2023	Cross G06F 9/455
				2024/0020538 A1	1/2024	Socher et al.
				2024/0062016 A1	2/2024	Tong et al.

* cited by examiner

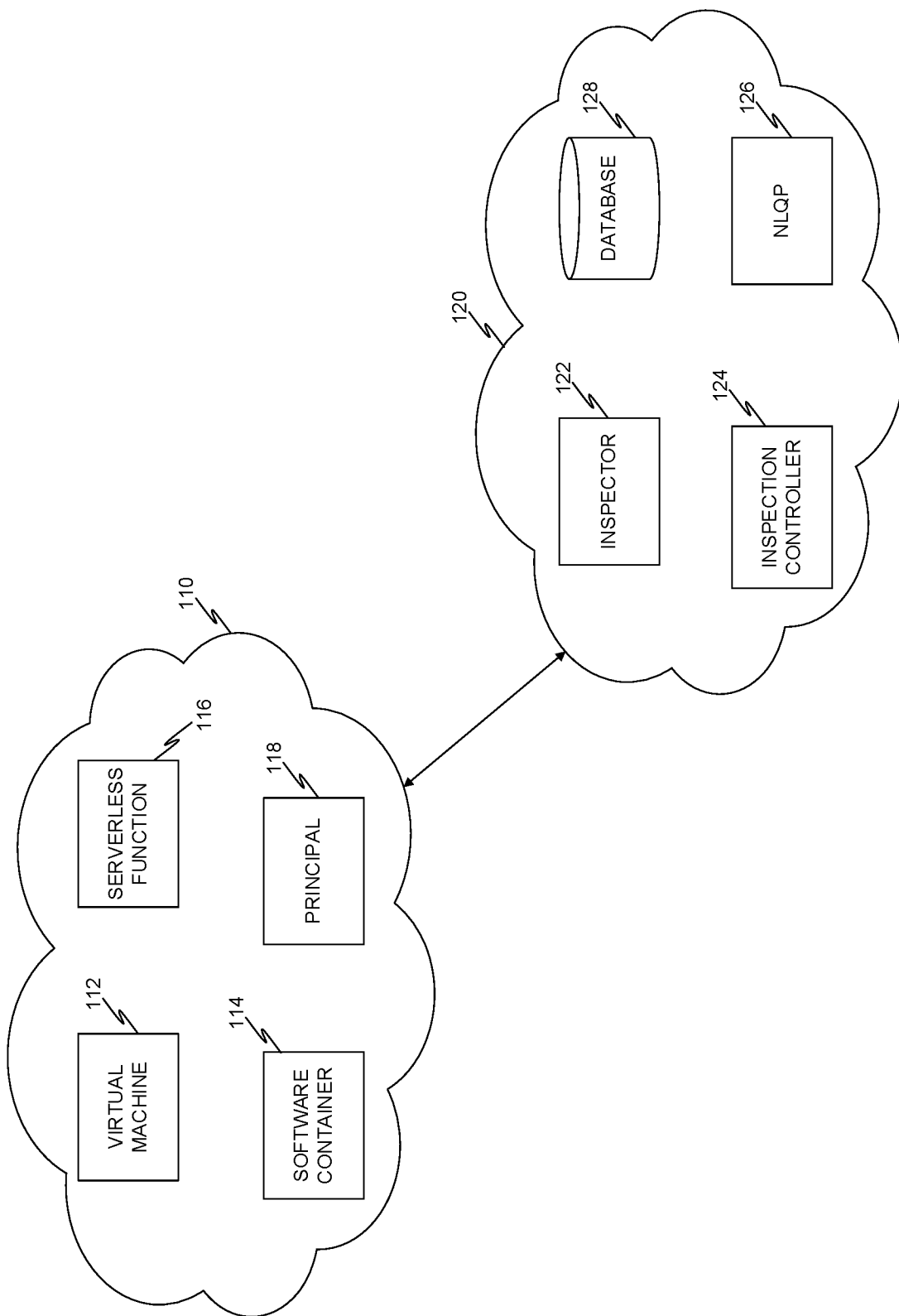


FIGURE 1

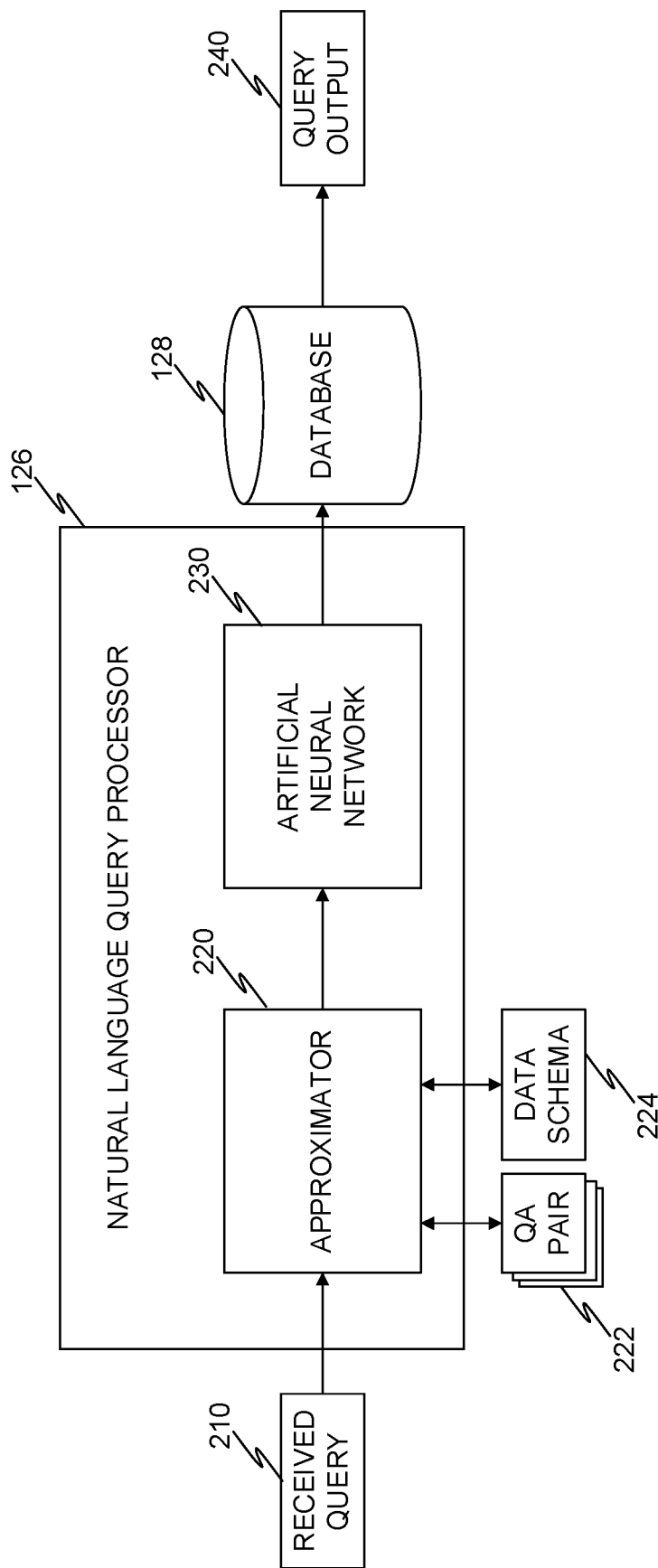


FIGURE 2

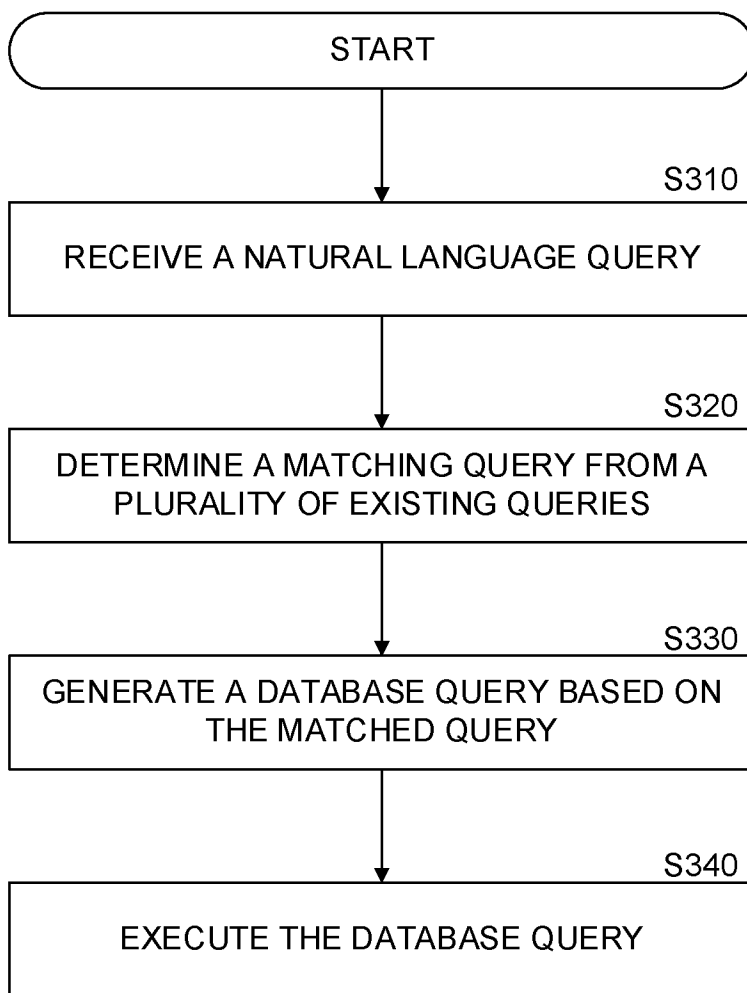


FIGURE 3

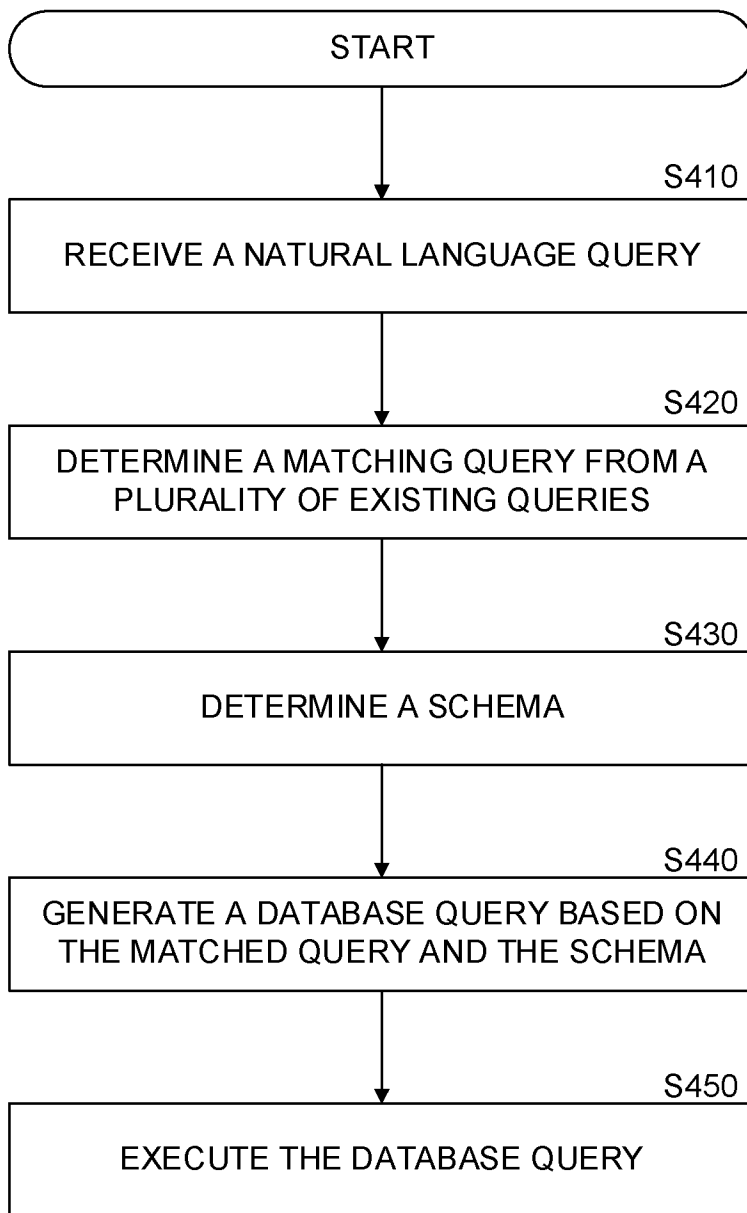


FIGURE 4

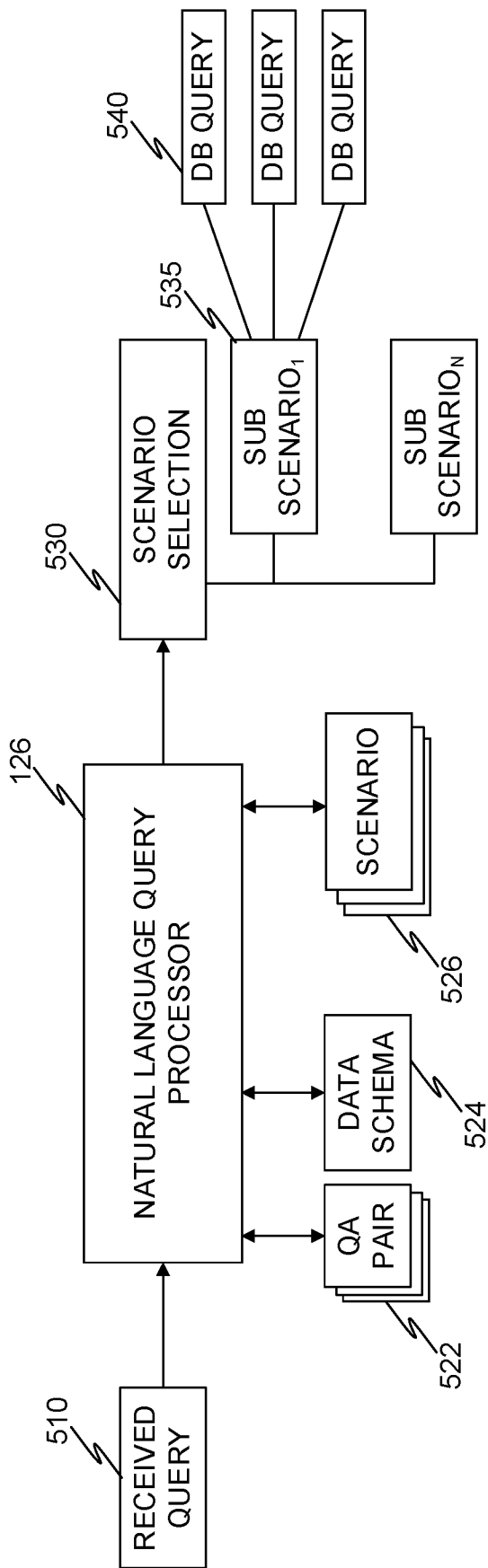


FIGURE 5

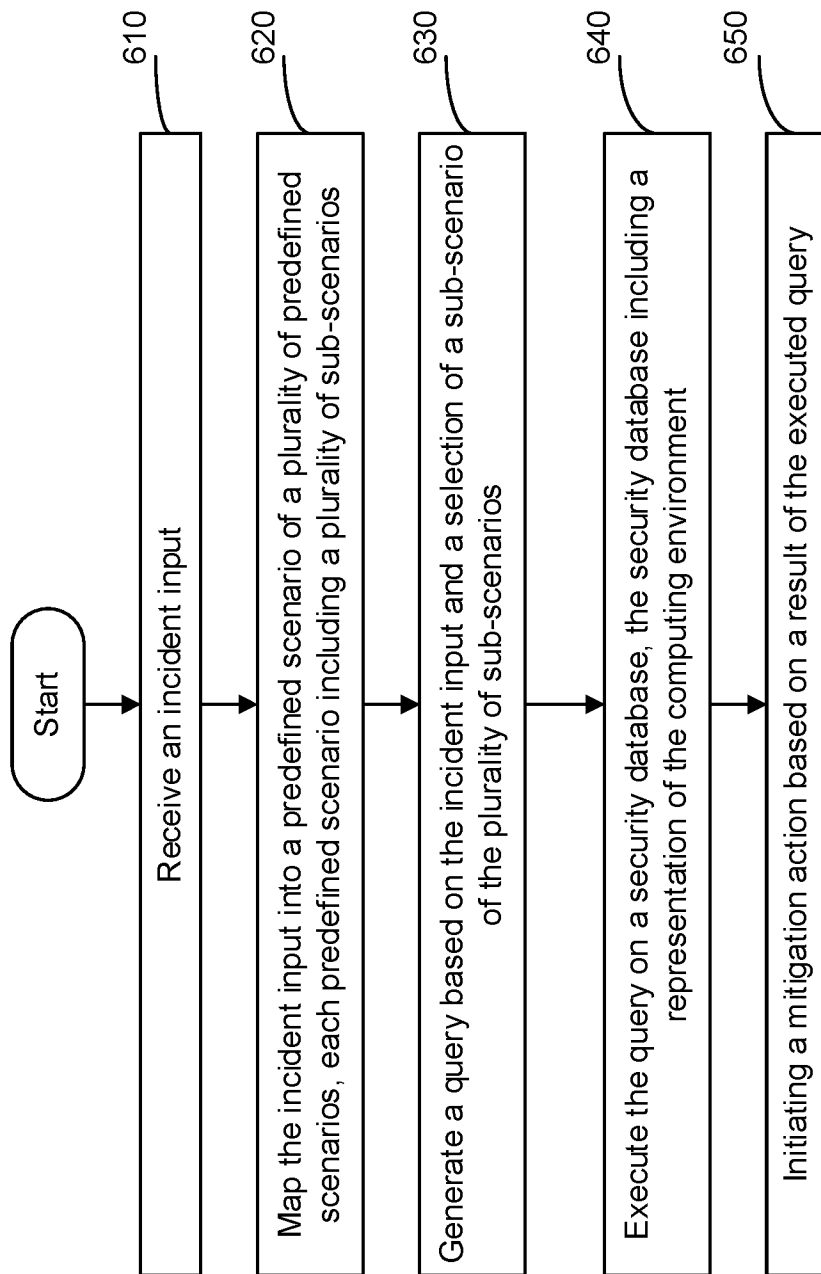


FIGURE 6

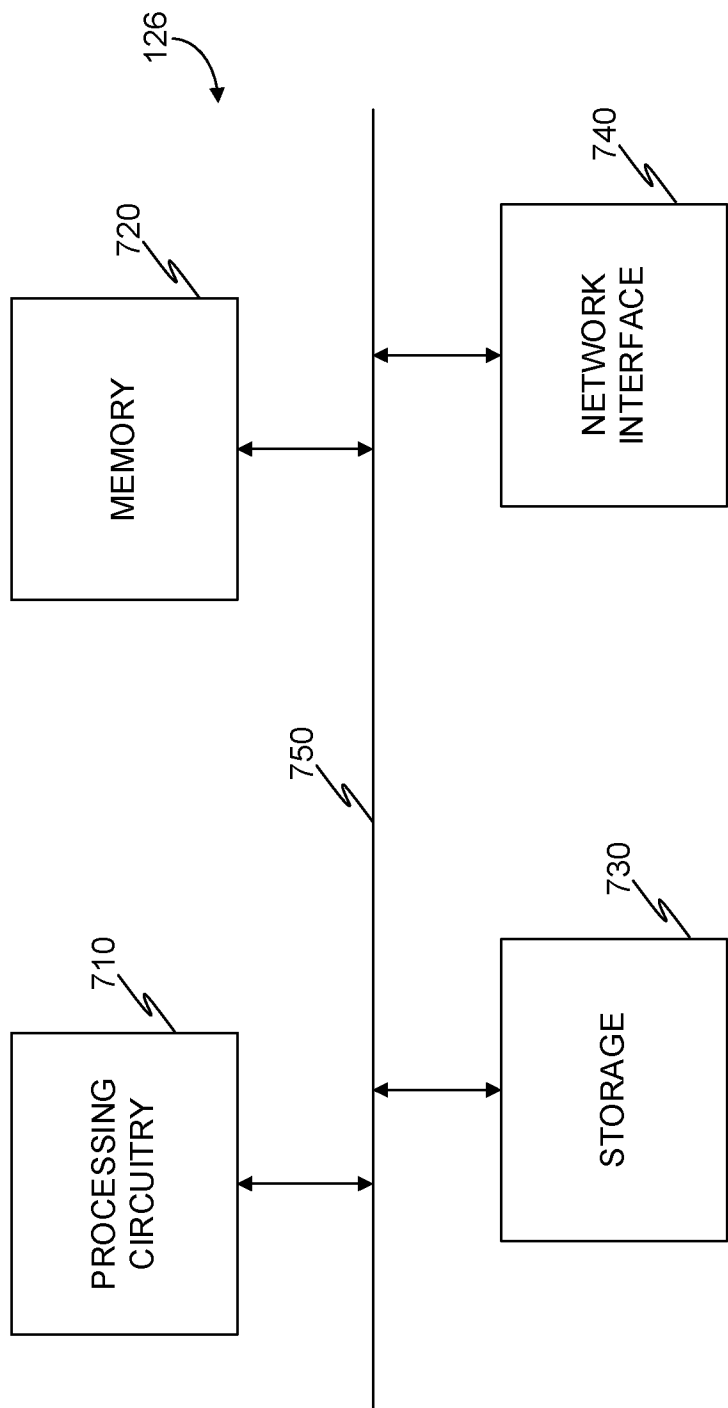


FIGURE 7

US 12,001,549 B1

1

**CYBERSECURITY INCIDENT RESPONSE
TECHNIQUES UTILIZING ARTIFICIAL
INTELLIGENCE**

CROSS-REFERENCE TO RELATED
APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 18/466,882 filed Sep. 14, 2023, which itself is a continuation-in-part of U.S. patent application Ser. No. 18/457,054 filed on Aug. 28, 2023, all contents of which are hereby incorporated by reference.

TECHNICAL FIELD

The present disclosure relates generally to cybersecurity incident response and specifically to initiating mitigation actions in response to detected cybersecurity threats.

BACKGROUND

Computer systems generate increasingly more data. As more and more data is generated, solutions arise to problems relating to storing, accessing, deleting, and managing this data.

One method of organizing and storing data is referred to as structured data storage. Structured data is implemented where data is structured, e.g., using a data schema, data model, and the like, and a persistent order to the data is realized.

Structured data solutions are extremely useful for computer systems, however, such solutions are not always human-friendly. In other words, a data structure, such as a SQL database, makes it easier for a machine to store data, retrieve data, manage data, etc., but requires a human to learn a special query language which the machine uses to retrieve and store data, for example.

Humans tend to converse in natural language, which does not have the rigid structure of machine languages. Increasingly, natural language processing techniques allow users to generate statements, queries, and the like, which a machine translates to a computer language, and executes on an appropriate data set.

A recurring issue with such processes is a lack of context, and a reliance on statistics of what other users search for. For example, for the natural language query "what is jay?", a computer has no way of discerning between the English letter "J", the given name "Jay", and a commonly used name of a North American bird species, just to give a few examples.

As such, specifically for cybersecurity solutions, an operator will often receive an alert that lacks context and information which is presented in a manner which is machine readable but does not immediately convey context, does not provide a root cause, or indicate what, if at all, should be done to remediate, mitigate, and the like.

It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither

2

identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

In one general aspect, method may include mapping a received incident input into a scenario of a plurality of scenarios, each scenario including a plurality of sub-scenarios. Method may also include generating a query based on the received incident input and a selection of a sub-scenario of the plurality of sub-scenarios. Method may furthermore include executing the query on a security database, the security database including a representation of a computing environment; and initiating a mitigation action based on a result of the executed query. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. Method where the incident input includes any one of: a query, a statement, and a combination thereof. Method may include: providing the received incident input into a large language model (LLM); and mapping the incident input into the scenario based on an output of the LLM. Method where the LLM is trained on: a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and any combination thereof. Method may include: generating the query further using the LLM. Method may include: training the LLM on a plurality of database queries, each database query executable on the security database. Method may include: receiving a user input through a user interface, the user interface configured to render a graphical representation of a group of sub-scenarios of the plurality of sub-scenarios; selecting the sub-scenario further based on the received user input. Method may include: generating a prompt for the LLM based on the received user input, the prompt, when executed configuring the LLM to output a sub-scenario selection. Method may include: receiving an user input through a component of a graphical user interface to initiate generation of an explanation of a security finding; utilizing the LLM to generate an explanation of a security finding, the explanation including any one of: a base observation regarding the security finding, an analysis of a symptomatic nature of the security finding, and a combination thereof; and rendering for display the generated explanation of the security finding. Method may include: processing a user input through a component of a graphical user interface to initiate investigation of a custom incident; and generating a request to receive additional contextual information where the custom incident is unrelated to any specific resource. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

US 12,001,549 B1

3

In one general aspect, non-transitory computer-readable medium may include one or more instructions that, when executed by one or more processors of a device, cause the device to: map a received incident input into a scenario of a plurality of scenarios, each scenario including a plurality of sub-scenarios; generate a query based on the received incident input and a selection of a sub-scenario of the plurality of sub-scenarios; execute the query on a security database, the security database including a representation of a computing environment; and initiate a mitigation action based on a result of the executed query. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

In one general aspect, system may include a processing circuitry. System may also include a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: map a received incident input into a scenario of a plurality of scenarios, each scenario including a plurality of sub-scenarios. System may in addition generate a query based on the received incident input and a selection of a sub-scenario of the plurality of sub-scenarios. System may moreover execute the query on a security database, the security database including a representation of a computing environment. System may also initiate a mitigation action based on a result of the executed query. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. System where the incident input includes any one of: a query, a statement, and a combination thereof. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: provide the received incident input into a large language model (LLM); and map the incident input into the scenario based on an output of the LLM. System where the LLM is trained on: a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and any combination thereof. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate the query further using the LLM. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: train the LLM on a plurality of database queries, each database query executable on the security database. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: receive a user input through a user interface, the user interface configured to render a graphical representation of a group of sub-scenarios of the plurality of sub-scenarios; and select the sub-scenario further based on the received user input. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate a prompt for the LLM based on the received user input, the prompt, when executed configuring the LLM to output a sub-scenario selection. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: receive an user input through a component of a graphical user interface to initiate generation of an explanation of a security finding; utilize the LLM to generate an explanation of a security finding, the explana-

4

tion including any one of: a base observation regard the security finding, an analysis of a symptomatic nature of the security finding, and a combination thereof; and render for display the generated explanation of the security finding. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: process a user input through a component of a graphical user interface to initiate investigation of a custom incident; and generate a request to receive additional contextual information where the custom incident is unrelated to any specific resource. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

In one general aspect, method may include receiving an incident input based on a cybersecurity event. Method may also include generating a prompt for a large language model (LLM) based on the received incident input. Method may furthermore include configuring the LLM to generate an output based on the generated prompt. Method may in addition include mapping the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, where each scenario is associated with an incidence response. Method may moreover include generating a query based on the received incident input and the mapped scenario. Method may also include executing the query on a security database, the security database including a representation of a computing environment; and initiating a mitigation action based on a result of the executed query. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. Method where the incident input includes any one of: a query, a statement, and a combination thereof. Method where the LLM is trained on any one of: a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and a combination thereof. Method where generating the query further comprises: generating a second prompt for the LLM which when executed by the LLM outputs the query, where the second prompt is generated based on any one of: the received incident input, the data schema, the plurality of scenarios, and a combination thereof. Method may include: training the LLM further on a plurality of database queries, each database query executable on the security database. Method may include: receiving an user input through an user interface, the user interface configured to render a graphical representation of the plurality of scenarios; selecting a sub-scenario of a scenario of the plurality of scenarios based on the received user input. Method may include: generating a prompt for the LLM based on the received user input, the prompt, when executed configuring the LLM to output a sub-scenario selection. Method may include: processing an user input through a component of a graphical user interface to initiate investigation of a custom incident; and generating a request to receive additional contextual information where the custom

US 12,001,549 B1

5

incident is unrelated to any specific resource. Method may include: traversing the security database to detect a cybersecurity finding, the finding associated with a resource, the resource associated with an incident of the incident response. Method may include: receiving an user input through a component of a graphical user interface to initiate generation of an explanation of the security finding; utilizing the LLM to generate an explanation of the security finding, the explanation including any one of: a base observation regarding the security finding, an analysis of a symptomatic nature of the security finding, and a combination thereof; and rendering for display the generated explanation of the security finding. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

In one general aspect, non-transitory computer-readable medium may include one or more instructions that, when executed by one or more processors of a device, cause the device to: receive an incident input based on a cybersecurity event; generate a prompt for a large language model (LLM) based on the received incident input; configure the LLM to generate an output based on the generated prompt; map the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, where each scenario is associated with an incidence response; generate a query based on the received incident input and the mapped scenario; execute the query on a security database, the security database including a representation of a computing environment; and initiate a mitigation action based on a result of the executed query. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

In one general aspect, system may include a processing circuitry. System may also include a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: receive an incident input based on a cybersecurity event. System may in addition generate a prompt for a large language model (LLM) based on the received incident input. System may moreover configure the LLM to generate an output based on the generated prompt. System may also map the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, where each scenario is associated with an incidence response. System may furthermore generate a query based on the received incident input and the mapped scenario. System may in addition execute the query on a security database, the security database including a representation of a computing environment. System may moreover initiate a mitigation action based on a result of the executed query. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. System where the incident input includes any one of: a query, a statement, and a combination thereof. System where the LLM is trained on any one of: a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and a combination thereof. System where the memory contains further instructions that, when executed by the processing circuitry for generating the query, further configure the system to: generate a second prompt for the LLM which when executed by the LLM outputs the query,

6

where the second prompt is generated based on any one of: the received incident input, the data schema, the plurality of scenarios, and a combination thereof. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: train the LLM further on a plurality of database queries, each database query executable on the security database. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: receive an user input through an user interface, the user interface configured to render a graphical representation of the plurality of scenarios; and select a sub-scenario of the plurality of scenarios based on the received user input. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate a prompt for the LLM based on the received user input, the prompt, when executed configuring the LLM to output a sub-scenario selection. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: process an user input through a component of a graphical user interface to initiate investigation of a custom incident; and generate a request to receive additional contextual information where the custom incident is unrelated to any specific resource. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: traverse the security database to detect a cybersecurity finding, the finding associated with a resource, the resource associated with an incident of the incident response. System where the memory contains further instructions which when executed by the processing circuitry further configure the system to: receive an user input through a component of a graphical user interface to initiate generation of an explanation of the security finding; utilize the LLM to generate an explanation of the security finding, the explanation including any one of: a base observation regard the security finding, an analysis of a symptomatic nature of the security finding, and a combination thereof; and render for display the generated explanation of the security finding. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. 1 is an example schematic diagram of a computing environment communicatively coupled with a cybersecurity inspection environment, utilized to describe an embodiment.

FIG. 2 is an example schematic illustration of a natural language query processor (NLQP), implemented in accordance with an embodiment.

FIG. 3 is an example flowchart of a method for generating a database query based on a natural language query, implemented in accordance with an embodiment.

FIG. 4 is an example flowchart of a method for generating a database query based on a natural language query utilizing a large language model, implemented in accordance with an embodiment.

US 12,001,549 B1

7

FIG. 5 is an example schematic illustration of a natural language query processor utilized in providing a cybersecurity incidence response system, implemented in accordance with an embodiment.

FIG. 6 is an example flowchart of a method for initiating an incident response mitigation action, implemented in accordance with an embodiment.

FIG. 7 is an example schematic diagram of an NLQP according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The various disclosed embodiments include a method and system for providing cybersecurity incident response. Cybersecurity systems are complex both in terms of the computing environments which they monitor, and in terms of objects which they monitor for, such as cybersecurity threats, vulnerabilities, exposures, misconfigurations, combinations thereof, and the like.

Often times, a vulnerability (or other threat) may become known to an operator of a cybersecurity system, without the operator knowing which of the workloads in their computing environment might be affected. A prominent example of this is the SolarWinds® attack in 2019-2020, during which attackers gained access to multiple different computing environments, all having in common a SolarWinds network monitoring software.

According to an embodiment, a natural language query processor is utilized to provide a system for cybersecurity incident response, for example as a software wizard. In an embodiment, a natural language query, an unstructured query, an alert, a combination thereof, and the like, are provided to a natural language query processor (NLQP) as an incident input.

In an embodiment, the NLQP is configured to receive the incident input and select a scenario. In some embodiments, the scenario is selected from a plurality of scenarios. In an embodiment, at least a scenario of the plurality of scenarios is a predefined scenario. In some embodiments, a scenario is generated by a neural network, as discussed below. In an embodiment, each scenario corresponds to a cybersecurity situation.

For example, a cybersecurity situation is, according to an embodiment, a workload having a malware, a workload generating network traffic with a prohibited domain, combinations thereof, and the like. In some embodiments, a scenario includes a plurality of sub-scenarios. In an embodiment, the cybersecurity situation includes a cybersecurity risk, such as a workload with an exposed database, a workload connected to a public network (e.g., the Internet), combinations thereof, and the like.

As another example, a sub-scenario includes a prompt, a question, a security finding, a combination thereof, and the like. In an embodiment, where the scenario is a workload with malware, a sub-scenario corresponds to a question, a prompt, and the like, for example: how was the workload infected, what lateral movement can an attacker perform

8

from the workload, what other workloads are at risk, what is the impact of a specific workload being compromised, etc.

In certain embodiments, the NLQP is configured to generate a database query based on any one of: the scenario, the sub-scenario, the incident input, a combination thereof, and the like.

In some embodiments, sub-scenarios are received continuously. In an embodiment, a second sub-scenario selection is performed based on a result of a database query executed based on a first sub-scenario selection. For example, in an embodiment, a first sub-scenario includes a database query corresponding to a question “how was the workload infected”. A result is received in response to executing the database query. In an embodiment, the result includes a network path between the workload and a public network, including a load balancer, a firewall, and a gateway. In an embodiment, a second sub-scenario is selected based on the result (e.g., the network path), to generate a database query corresponding to “are additional workloads compromised”. Such a database query, when executed, returns a result which corresponds to the status of each workload in the network path, to determine if such workloads are also compromised. In an embodiment, the second sub-scenario includes generating an instruction which, when executed, initiates inspection for a cybersecurity object, a vulnerability, an exposure, a misconfiguration, a malware, a combination thereof, and the like.

In an embodiment, in response to detecting a cybersecurity threat, a mitigation is initiated in the computing environment where the workload is deployed.

Such a software wizard is advantageous as it increases the usability of a cybersecurity monitoring solution, and improves the incident response time. By improving the scenario selection, for example utilizing a large language model to map an incident input to a scenario, the improved software wizard decreases incidence response time, and therefore decreases time to mitigation in the event of a cybersecurity breach.

FIG. 1 is an example schematic diagram of a computing environment communicatively coupled with a cybersecurity inspection environment, utilized to describe an embodiment. A computing environment **110** is, according to an embodiment, a cloud computing environment, a networked environment, an on-premises environment, a combination thereof, and the like.

For example, in an embodiment, a cloud computing environment is implemented as a virtual private cloud (VPC), a virtual network (VNet), and the like, on a cloud computing infrastructure. A cloud computing infrastructure is, according to an embodiment, Amazon® Web Services (AWS), Google® Cloud Platform (GCP), Microsoft® Azure, and the like.

In certain embodiment, the computing environment **110** includes a plurality of entities. An entity in a computing environment **110** is, for example, a resource, a principal **118**, and the like. A resource is, according to an embodiment, a hardware, a baremetal machine, a virtual machine, a virtual workload, a provisioned hardware (or portion thereof, such as a processor, a memory, a storage, etc.), and the like.

A principal **118** is an entity which is authorized to perform an action on a resource, initiate an action in the computing environment **110**, initiate actions with respect to other principals, a combination thereof, and the like. According to an embodiment, a principal is a user account, a service account, a role, a combination thereof, and the like.

In certain embodiments, a resource in a computing environment is a virtual machine **112**, a software container **114**,

US 12,001,549 B1

9

a serverless function **116**, and the like. For example, in an embodiment, a virtual machine **112** is implemented as an Oracle® VirtualBox®. In some embodiments, a software container **114** is implemented utilizing a Docker® Engine, a Kubernetes® platform, combinations thereof, and the like. In certain embodiments, a serverless function **116** is implemented in AWS utilizing Amazon Lambda®.

In some embodiments, the computing environment **110** is implemented as a cloud environment which includes multiple computing environments. For example, a first cloud computing environment is utilized as a production environment, a second cloud computing environment is utilized as a staging environment, a third cloud computing environment is utilized as a development environment, and so on. Each such environment includes, according to an embodiment, a resource, a principal, and the like, having a counterpart in the other environments.

For example, according to an embodiment, a first virtual machine **112** is deployed in a production environment, and a corresponding first virtual machine is deployed in a staging environment, which is essentially identical to the production environment.

In an embodiment, the computing environment **110** is monitored by an inspection environment **120**. According to an embodiment, the inspection environment **120** is configured to inspect, scan, detect, and the like, cybersecurity threats, cybersecurity risks, cybersecurity objects, misconfigurations, vulnerabilities, exploitations, malware, combinations thereof, and the like.

In certain embodiments, the inspection environment **120** is further configured to provide a mitigation action, a remediation action, a forensic finding, a combination thereof, and the like.

In some embodiments, an inspector **122** is configured to detect a cybersecurity object in a workload deployed in the computing environment **110**. For example, in an embodiment, the inspector is a software container pod configured to detect a predetermined cybersecurity object in a disk, access to which is provided to the inspector **122** by, for example, the inspection controller **124**.

In an embodiment, a cybersecurity object is a password stored in cleartext, a password stored in plaintext, a hash, a certificate, a cryptographic key, a private key, a public key, a hash of a file, a signature of a file, a malware object, a code object, an application, an operating system, a combination thereof, and the like.

In certain embodiments, the inspector **122** is assigned to inspect a workload in the computing environment **110** by an inspection controller **124**. In an embodiment, the inspection controller initiates inspection by, for example, generating an inspectable disk based on an original disk. In an embodiment, generating the inspectable disk include generating a copy, a clone, a snapshot, a combination thereof, and the like, of a disk of a workload deployed in the computing environment **110**, and providing access to the inspectable disk (for example by assigning a persistent volume claim) to an inspector **122**.

In an embodiment, where an inspector **122** detects a cybersecurity object in a disk of a workload, a representation is generated and stored in a security database **128**. In certain embodiments, the database is a columnar database, a graph database, a structured database, an unstructured database, a combination thereof, and the like. In certain embodiments, the representation is generated based on a predefined data schema. For example, a first data schema is utilized to generate a representation of a resource, a second data schema is utilized to generate a representation of a principal,

10

a third data schema is utilized to generated a representation of a cybersecurity object, etc.

For example, according to an embodiment, the representation is stored on a graph database, such as Neo4j®. In certain embodiments, a resource is represented by a resource node in the security graph, a principal is represented by a principal node in the security graph, etc.

In some embodiments, the inspection environment **120** further includes a natural language query processor **126** (NLQP **126**). In an embodiment, the NLQP **126** is configured to receive a query in a natural language, and generate, based on the received query, a structured query which is executable on the database **128**.

In certain embodiments, it is advantageous to provide a user with an interface to query the database **128** in a natural language. It is further advantageous to provide a system and method that provides accurate translation between a query received in natural language and a database query, in order to provide a user with a relevant result to their query.

FIG. 2 is an example schematic illustration of a natural language query processor, implemented in accordance with an embodiment. In certain embodiments, the natural language query processor **126** (NLQP **126**) is implemented as a virtual workload in an inspection environment. In some embodiments, the NLQP **126** includes an approximator **220**, and an artificial neural network (ANN) **230**. In some embodiments, the ANN **230** is a large language model, such as GPT, BERT, and the like.

In an embodiment, the NLQP **126** receives a query **210**. In some embodiments, the received query **210** is a query in natural language, such as an English language query. In an embodiment, the received query **210** cannot be executed on a database, such as security database **128**. In certain embodiments, the security database **128** includes a representation of a computing environment, such as the computing environment **110** of FIG. 1 above.

In an embodiment, the received query **210** is provided to the approximator **220**. In an embodiment, the approximator **220** includes a large language model (LLM), such as GPT, BERT, and the like.

In some embodiments, the LLM (e.g., of the approximator **220**, the ANN **230**, etc.) includes a fine-tuning mechanism. In an embodiment, fine-tuning allows to freeze some weights of a neural network while adapting others based on training data which is unique to a particular set of data.

In certain embodiments, an LLM cannot be fine-tuned, for example due to a lack of access to weights of the model. In such embodiments, it is advantageous to provide the LLM with additional data in order to generate a result which is accurate and relevant.

For example, in an embodiment, the approximator **220** is provided with a plurality of query-answer (QA) pairs **222**, and a data schema **224**. In an embodiment, the QA pairs **222** include each a database query and a corresponding response. In some embodiments, the query of the QA pair **222** is a query which was previously executed on the database **128**.

In some embodiments, the data schema **224** is a data schema of the database **128**. In some embodiments, a plurality of data schemas **224** are utilized. For example, in an embodiment, the plurality of data schemas **224** include a data schema for a principal, a data schema for a resource, a data schema of a cloud computing environment, combinations thereof, and the like.

In an embodiment, the approximator **220** is configured to generate a prompt based on a predetermined template, the received query **210**, a QA pair **222**, and the data schema **224**. In some embodiments, the approximator is configured to

US 12,001,549 B1

11

receive the query **210** and generate a selection of a QA pair **222** from a plurality of QA pairs. For example, in an embodiment, the approximator is configured to receive the query **210**, and generate a prompt for an LLM to detect from a plurality of QA pairs, a QA pair **222** which is the closest match to the received query **222**. In some embodiments, the prompt further includes the data schema **224**.

In an embodiment, the output of the approximator **220** is a QA pair **222** which an LLM of the approximator **220** outputs as being the closest match to the received query **210**. In some embodiments, the approximator **220** outputs a group of QA pairs from the plurality of QA pairs.

In certain embodiments, the output of the approximator **220** is provided to the ANN **230**. In an embodiment, the ANN **230** is configured to generate a database query (i.e., a query which is executable by a database, database management system, etc.) based on the output of the approximator **220**. In some embodiments, the ANN **230** includes an LLM, and is configured to generate a prompt for the LLM based on the received output, the received query **210**, and the data schema **224**.

For example, in an embodiment, the ANN **230** is configured to receive the query **210**, a QA pair **222** selected by the approximator **220**, and the data schema **224** as inputs. The ANN **230** is further configured to generate a prompt for an LLM based on the received inputs, which, according to an embodiment, configures the LLM to output a database query based on the received inputs.

In an embodiment, the outputted database query is executed on a database **128** to provide a query output **240**. In an embodiment, a plurality of database queries are outputted by the NLQP **126**, each of which is executed on a database, such as database **128**. In such embodiments, a plurality of query outputs **240** are generated.

In some embodiments, the query output **240** is provided to a client device, a user account, a user interface, rendered for display on a graphical user interface, a combination thereof, and the like.

FIG. 3 is an example flowchart of a method for generating a database query based on a natural language query, implemented in accordance with an embodiment. In an embodiment, the method is performed by utilizing an artificial neural network.

At **S310**, a natural language query is received. In an embodiment, the natural language query is received through a user interface, a graphical user interface, and the like. In some embodiments, a natural language query is an unstructured query, a partially structured query, and the like. For example, a structured query is a query which can be executed on a database to produce a result, whereas an unstructured query, a partially structured query, and the like, cannot be executed on a database to produce a result, according to an embodiment.

For example, according to an embodiment, a natural language query is “public ECRs with container images that contain cloud keys”, “find all vulnerabilities that can be exploited remotely”, “find all vulnerabilities that lead to information disclosure”.

In some embodiments, the natural language query is processed for tokenization. In an embodiment, each word in the natural language query is mapped to a tokenized word, tokenized word portion, and the like. For example, in an embodiment, vulnerability, vulnerabilities, vulnerabilites (with an incorrect spelling) are all mapped to a single term (e.g., “vulnerable”), and the single term is tokenized. This is advantageous as the context is preserved while tokenization

12

is minimized, since only a single term is tokenized, rather than having to tokenize each different term.

At **S320**, an existing query is selected. In an embodiment, the existing query is an existing database query. In some embodiments, the selection includes a query pair, including a database query and a response, result, and the like, which is generated based on execution of the database query on a database.

In an embodiment, the existing query is selected from a group of preselected queries. In some embodiments, a match is determined between the natural language query and a plurality of existing queries. In certain embodiments, generating a match includes determining a match score. For example, in an embodiment, a match score is generated between a natural language query and a preexisting database query based on natural language processing (NLP) techniques, such as the distance-based Word2Vec.

For example, in an embodiment, a distance is determined between the received natural language query and a first preexisting database query, and between the received natural language query and a second preexisting database query. In certain embodiments, the preexisting query having a shorter distance to the natural language query is selected as the matched query.

At **S330**, a database query is generated. In an embodiment, the database query is generated based on the received natural language query and the selected existing query. In certain embodiments, the database query is generated by adapting the existing query to the received natural language query. In an embodiment, adapting the existing query based on the received natural language query is performed by an artificial neural network, such as a generative ANN. In some embodiments, the adaptation is performed by a generative adversarial network (GAN), which includes a generator network and a discriminator network.

At **S340**, the database query is executed. In an embodiment, executing a database query includes configuring a database management system to receive a database query, execute the database query on one or more datasets stored in the database, and generate a result.

In certain embodiments, where a plurality of database queries are generated, each query is executed on a database. According to an embodiment, each query is executed on the same database, a different database, a combination thereof, and the like.

FIG. 4 is an example flowchart of a method for generating a database query based on a natural language query utilizing a large language model, implemented in accordance with an embodiment. In an embodiment, the method is performed by utilizing an artificial neural network such as an LLM. For example, an LLM is, according to an embodiment, GPT, BERT, and the like.

At **S410**, a natural language query is processed. In an embodiment, the natural language query is received through a user interface, a graphical user interface, and the like. In some embodiments, a natural language query is an unstructured query, a partially structured query, and the like. For example, a structured query is a query which can be executed on a database to produce a result, whereas an unstructured query, a partially structured query, and the like, cannot be executed on a database to produce a result, according to an embodiment.

For example, according to an embodiment, a natural language query is “public ECRs with container images that contain cloud keys”, “find all vulnerabilities that can be exploited remotely”, “find all vulnerabilities that lead to information disclosure”.

US 12,001,549 B1

13

In some embodiments, the natural language query is processed for tokenization. In an embodiment, each word in the natural language query is mapped to a tokenized word, tokenized word portion, and the like. For example, in an embodiment, vulnerability, vulnerabilities, vulnerabilites (with an incorrect spelling) are all mapped to a single term (e.g., “vulnerable”), and the single term is tokenized. This is advantageous as the context is preserved while tokenization is minimized, since only a single term is tokenized, rather than having to tokenize each different term.

At **S420**, an existing query is selected. In an embodiment, the existing query is an existing database query. In some embodiments, the selection includes a query pair, including a database query and a response, result, and the like, which is generated based on execution of the database query on a database.

In an embodiment, the existing query is selected from a group of preselected queries. In an embodiment, an LLM is provided with a generated prompt to select a query from the group of preselected queries. In certain embodiments, the prompt is generated based on a preexisting template. For example, in an embodiment, the prompt is generated based on a template, the received query, a data schema, a combination thereof, and the like. In some embodiments, the LLM is configured to select a database query from the preselected queries which mostly resembles the natural language query.

At **S430**, a data schema is determined. In certain embodiments a plurality of data schemas are determined. In an embodiment, the data schema is determined based on the natural language query. For example, in an embodiment, a keyword, a phrase, and the like, are detected in the natural language query.

In some embodiments, the natural language query is received as a text input which is parsed, and a keyword is detected in the parsed text. In an embodiment, the keyword, phrase, and the like, is matched to a data schema.

For example, in the natural language query “public ECRs with container images that contain cloud keys”, the keyword “container” corresponds to a data schema of a resource, and the keyword “cloud keys” corresponds to a data schema of an identity.

At **S440**, a database query is generated. In an embodiment, the database query is generated based on the received natural language query and the selected existing query. In certain embodiments, the database query is generated by adapting the existing query to the received natural language query.

In an embodiment, the database query is generated as an output of an LLM. For example, according to an embodiment, an LLM is configured to receive a prompt, which is generated based on a template. In an embodiment, the template is adapted to the prompt based on the received natural language query, the selected database query, the determined schema, a combination thereof, and the like.

At **S450**, the database query is executed. In an embodiment, executing a database query includes configuring a database management system to receive a database query, execute the database query on one or more datasets stored in the database, and generate a result.

In certain embodiments, where a plurality of database queries are generated, each query is executed on a database. According to an embodiment, each query is executed on the same database, a different database, a combination thereof, and the like.

For example, in an embodiment, a database query is executed on a graph database, such as Neo4j®, which includes, stored thereon, a representation of a computing

14

environment. In an embodiment, the representation is generated based on a data schema, a unified data schema, a data template, a combination thereof, and the like.

According to an embodiment, a unified data schema is a data schema which is utilized across a plurality of workloads, across a plurality of computing environments, combinations thereof, and the like. For example, according to an embodiment, a unified data schema for workloads is utilized to represent different resources, such as a virtual machine, a software container, a serverless function, and the like. In some embodiment, a unified data schema utilizes the same data fields to represent a virtual machine in a first computing environment (e.g., AWS), a virtual machine in a second computing environment (e.g., GCP), a software container in the first computing environment, a software container in the second computing environment, and so on.

FIG. 5 is an example schematic illustration of a natural language query processor utilized in providing a cybersecurity incidence response software wizard, implemented in accordance with an embodiment.

According to an embodiment, an incident input **510** is received by the natural language query processor (NLQP) **126**. In an embodiment, the incident input **510** includes an alert, a natural language query, a combination thereof, and the like.

In an embodiment, the NLQP **126** is configured to access a plurality of query-answer pairs **522**, a data schema **524**, and a plurality of scenarios **526**. In some embodiments, the query-answer pairs **522** includes a plurality of database queries and a corresponding answer, result, and the like, generated by executing the database query on a database.

In an embodiment, the data schema **524** includes a plurality of data fields, each data field for storing a value corresponding to the data field. For example, in an embodiment, the data schema **524** is a unified data schema, which is utilized in generating a representation of a computing environment, such as a cloud computing environment. In some embodiments, the data schema **524** includes a template for representing a resource, a principal, a cloud entity, and the like. For example, according to an embodiment, a unified data schema includes a template for representing a workload, wherein the workload is a virtual machine, a software container, a serverless function, and the like.

According to an embodiment, the NLQP **126** is configured to receive the input **510** and select a scenario of the plurality of scenarios **526**. For example, in an embodiment, the NLQP **126** is configured to receive the incident input **510** and generate a prompt (e.g., based on a predefined template) for a large language model (LLM), such that the prompt, when executed by the LLM, configures the LLM to select a scenario from the plurality of scenarios **526**. In an embodiment, the selected scenario is a scenario which most likely corresponds to the received incident input **510**. A scenario may be, for example “A workload is infected with malware”, according to an embodiment. In an embodiment, the prompt is further generated based on the data schema **524**, the plurality of query-answer pairs **522**, a combination thereof, and the like.

In some embodiments, the NLQP **126** is configured to generate a scenario for the plurality of scenarios **526**. In certain embodiments, the NLQP **126** is configured to receive an incident input **510** and generate a scenario based thereon. For example, in an embodiment, the NLQP **126** is configured to receive the incident input **510** and generate a prompt (e.g., based on a template) that when executed by the LLM, configures the LLM to generate a scenario. In some embodiments, the plurality of scenarios **526** includes a group of

US 12,001,549 B1

15

predefined scenarios. In an embodiment, the prompt is further generated based on the data schema 524, the plurality of query-answer pairs 522, a combination thereof, and the like.

In an embodiment, each scenario includes a plurality of sub-scenario. A sub-scenario includes, according to an embodiment, a query, a natural language query, a database query, combinations thereof, and the like. In some embodiments, a sub-scenario corresponds to a plurality of queries, natural language queries, database queries, combinations thereof, and the like.

In certain embodiments, a sub-scenario is selected. In some embodiments, a plurality of sub-scenarios are selected. In an embodiment, the NLQP 126 is configured to receive a scenario selection 520, and generate a prompt for an LLM, which when executed by the LLM, configured the LLM to select a sub-scenario. In an embodiment, the prompt is further generated based on the data schema 524, the plurality of query-answer pairs 522, a combination thereof, and the like.

In an embodiment, a sub-scenario 535 corresponds to a plurality of database queries, such as database query 540. In certain embodiments, a software wizard is configured to generate a graphical user interface (GUI) for rendering on a display of a user device (e.g., a personal computer, a tablet, a smartphone, a desktop computer, a wearable device, a combination thereof, and the like). In some embodiments, the GUI includes an input for receiving a user input. For example, in an embodiment, a group of the plurality of scenarios are each rendered on the GUI, and a user input is received. In an embodiment, the received user input corresponds to a selection of a scenario of the group of scenarios, wherein the group of scenarios is selected by the NLQP 126 as having the best correlation to the incident input 510.

In some embodiments, the GUI further includes a group of sub-scenarios of the plurality of sub-scenarios, and a received user input corresponds to a selection of a sub-scenario of the group of sub-scenarios, wherein the group of sub-scenarios is selected by the NLQP 126 as having the best correlation to the incident input 510.

FIG. 6 is an example flowchart of a method for initiating an incident response mitigation action, implemented in accordance with an embodiment.

At S610, an incident input is processed. In an embodiment, the incident response is received. In some embodiments, the incident response includes an alert, a query, a natural language query, a database query, a combination thereof, and the like.

For example, in an embodiment, an alert is “EC2 virtual machine with ID of i-012abcd34efghi56 infected with malware with SHA1 hash of 3395856ce81f2b7382dee72602f798b642f14141”, which indicates that a virtual machine having an identifier of “i-012abcd34efghi56” is infected with a malware corresponding to a hash “3395856ce81f2b7382dee72602f798b642f14141”.

At S620, the incident input is mapped to a scenario. In an embodiment, the incident input is provided to a natural language query processor (NLQP) which is configured to generate a prompt for an LLM based on the incident input. In some embodiments, the prompt is generated based on a template, for example a predefined template.

In certain embodiments, the prompt is further generated based on a data schema, a plurality of query-answer pairs, a combination thereof, and the like. In an embodiment, the prompt, when processed by an LLM, configures the LLM to select a scenario from a plurality of scenarios.

16

In some embodiments, the plurality of scenarios includes predefined scenarios, scenarios generated by an LLM (the LLM configured to generate a scenario based on an incident input, a data schema, a combination thereof, and the like), a combination thereof, and the like.

In an embodiment, a data schema includes a plurality of data fields, integrity constraints, and the like, which are utilized to represent a computing environment. For example, in an embodiment, a representation of a computing environment is stored in a database (e.g., a graph database) based on the data schema. In some embodiments, the data schema is a unified data schema, based on which a plurality of different types of computing environments (e.g., cloud computing environments, hybrid computing environments, AWS, Azure, GCP, etc.) are represented. In some embodiments, the unified data schema is utilized to represent a plurality of different types of entities, such as resources, principals, workloads, virtual machines, software containers, serverless functions, combinations thereof, and the like.

At S630, a database query is generated. In an embodiment, the database query is generated based on a selection of a scenario, a sub-scenario, a data schema, an incident input, a combination thereof, and the like. For example, in an embodiment, the database query is generated by providing a prompt to a large language model. A large language model is, for example, GPT, BERT, and the like.

In certain embodiments, the prompt is generated based on a template. In some embodiments, the template is modified based on a selection of a scenario, a sub-scenario, a data schema, an incident input, a combination thereof, and the like.

In an embodiment, the database query is generated based on a predefined database query. In some embodiments, the database query is generated based on the predefined database query, modified by, for example, the incident input. In an embodiment, an incident input is parsed to detect values (e.g., an identifier of a resource, an identifier of a cloud computing environment, an identifier of a cybersecurity object, an identifier of a cybersecurity threat, a combination thereof, and the like).

At S640, the database query is executed. In an embodiment, executing the database query produces a database answer, a database result, and the like. In an embodiment, the database result includes a textual result. For example, in an embodiment, the textual result includes an identifier of a resource, an identifier of a workload, an identifier of a principal, a cybersecurity object, a representation of a mitigation action, a combination thereof, and the like.

In some embodiments, the database query is executed on a graph database having stored thereon a security graph, the security graph including a representation of a computing environment, such as a cloud computing environment. In an embodiment, representation is generated based on the data schema. For example, in some embodiments, a resource, such as a workload, virtual machine, software container, serverless function, and the like, is represented by a node. In an embodiment, the node representing the resource is connected to other nodes representing resources, such as software applications, appliances, operating systems, etc.

In an embodiment, the security graph further includes representations of cybersecurity objects, representations of cybersecurity threats, remediation actions, mitigation actions, combinations thereof, and the like. For example, in an embodiment, a node representing a resource is connected to a node representing a cybersecurity threat, such as a malware, where a malware object was detected on the resource, for example by inspection. In some embodiments,

a node representing a remediation action is connected to a node representing the cybersecurity threat to which the remediation action corresponds. In certain embodiments, a remediation action, a mitigation action, and the like, are represented as: metadata on a node representing a resource, metadata on a node representing a cybersecurity risk, metadata on a node representing a cybersecurity object, a combination thereof, and the like.

For example, according to an embodiment, a malware object is remediated by an action which includes sandboxing the infected application, revoking access to a workload, revoking access from a workload, a combination thereof, and the like.

At **S650**, a mitigation action is initiated. In an embodiment, a mitigation action includes generating a notification, generating an alert, updating an alert, generating a severity score, updating a severity score, generating a ticket, generating a risk score, updating a risk score, initiating a remediation action, initiating an incident response, a combination thereof, and the like.

In an embodiment, a remediation action includes revoking access to a resource, revoking access from a resource, revoking a permission from a principal, revoking access to a principal, uninstalling an application, sandboxing an application, sandboxing a workload, a combination thereof, and the like. In an embodiment, revoking access includes revoking network access to a resource, from a resource, etc. In some embodiments, revoking access to a principal includes removing a permission from a first principal to assume another principal. For example, in an embodiment, a first service account is revoked permission to assume a role of a second service account.

In certain embodiments, the result of executing the database query at **S640** is provided to an LLM with a modified prompt. In an embodiment, the prompt is modified based on the result of the database query execution. In some embodiments, the prompt includes a predefined template. In certain embodiments, the predefined template includes data fields which are modified based on the result. In an embodiment, the prompt, when processed by the LLM, configures the LLM to output an explanation for the result.

For example, in an embodiment, the prompt is generated based on a template such as “how did RESOURCE_ID become infected with MALWARE_ID”. In an embodiment, the data field of “RESOURCE_ID” is modified with a value received as a result of executing the database query, and MALWARE_ID is a data field which is modified based on the incident input. In some embodiments, a plurality of data fields (e.g., both RESOURCE_ID AND MALWARE_ID), are modified based on the incident input.

FIG. 7 is an example schematic diagram of an NLQP **126** according to an embodiment. The NLQP **126** includes a processing circuitry **710** coupled to a memory **720**, a storage **730**, and a network interface **740**. In an embodiment, the components of the NLQP **126** may be communicatively connected via a bus **750**.

The processing circuitry **710** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose microprocessors, microcontrollers, digital signal processors

(DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory **720** may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof. In an embodiment, the memory **720** is an on-chip memory, an off-chip memory, a combination thereof, and the like. In certain embodiments, the memory **720** is a scratch-pad memory for the processing circuitry **710**.

In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage **730**, in the memory **720**, in a combination thereof, and the like. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the processing circuitry **710**, cause the processing circuitry **710** to perform the various processes described herein.

The storage **730** is a magnetic storage, an optical storage, a solid-state storage, a combination thereof, and the like, and is realized, according to an embodiment, as a flash memory, as a hard-disk drive, or other memory technology, or any other medium which can be used to store the desired information.

The network interface **740** is configured to provide the NLQP **126** with communication with, for example, the security database **128**, an inspector **122**, an inspection controller **124**, and the like.

It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. 7, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

Furthermore, in certain embodiments the [other system] may be implemented with the architecture illustrated in FIG. 7. In other embodiments, other architectures may be equally used without departing from the scope of the disclosed embodiments.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units (“CPUs”), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment

US 12,001,549 B1

19

and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

It should be understood that any reference to an element herein using a designation such as “first,” “second,” and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

As used herein, the phrase “at least one of” followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including “at least one of A, B, and C,” the system can include A alone; B alone; C alone; 2A; 2B; 2C; 3A; A and B in combination; B and C in combination; A and C in combination; A, B, and C in combination; 2A and C in combination; A, 3B, and 2C in combination; and the like.

What is claimed is:

1. A method for providing cybersecurity incident response, comprising:
 - receiving an incident input based on a cybersecurity event;
 - generating a prompt for a large language model (LLM) based on the received incident input;
 - configuring the LLM to generate an output based on the generated prompt;
 - mapping the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, wherein each scenario is associated with an incidence response;
 - generating a query based on the received incident input and the mapped scenario;
 - executing the query on a security database, the security database including a representation of a computing environment; and
 - initiating a mitigation action based on a result of the executed query.
2. The method of claim 1, wherein the incident input includes any one of: a query, a statement, and a combination thereof.
3. The method of claim 1, wherein the LLM is trained on any one of: a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and a combination thereof.
4. The method of claim 3, wherein generating the query further comprises:
 - generating a second prompt for the LLM which when executed by the LLM outputs the query, wherein the second prompt is generated based on any one of: the received incident input, the data schema, the plurality of scenarios, and a combination thereof.

20

5. The method of claim 3, further comprising: training the LLM further on a plurality of database queries, each database query executable on the security database.
6. The method of claim 3, further comprising:
 - receiving a user input through a user interface, the user interface configured to render a graphical representation of the plurality of scenarios;
 - selecting a sub-scenario of a scenario of the plurality of scenarios based on the received user input.
7. The method of claim 6, further comprising:
 - generating a prompt for the LLM based on the received user input, the prompt, when executed configuring the LLM to output a sub-scenario selection.
8. The method of claim 7, further comprising:
 - processing a user input through a component of a graphical user interface to initiate investigation of a custom incident; and
 - generating a request to receive additional contextual information wherein the custom incident is unrelated to any specific resource.
9. The method of claim 6, further comprising:
 - traversing the security database to detect a cybersecurity finding, the finding associated with a resource, the resource associated with an incident of the incident response.
10. The method of claim 9, further comprising:
 - receiving a user input through a component of a graphical user interface to initiate generation of an explanation of the security finding;
 - utilizing the LLM to generate an explanation of the security finding, the explanation including any one of: a base observation regarding the security finding, an analysis of a symptomatic nature of the security finding, and a combination thereof; and
 - rendering for display the generated explanation of the security finding.
11. A non-transitory computer-readable medium storing a set of instructions for providing cybersecurity incident response, the set of instructions comprising:
 - one or more instructions that, when executed by one or more processors of a device, cause the device to:
 - receive an incident input based on a cybersecurity event;
 - generate a prompt for a large language model (LLM) based on the received incident input;
 - configure the LLM to generate an output based on the generated prompt;
 - map the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, wherein each scenario is associated with an incidence response;
 - generate a query based on the received incident input and the mapped scenario;
 - execute the query on a security database, the security database including a representation of a computing environment; and
 - initiate a mitigation action based on a result of the executed query.
12. A system for providing cybersecurity incident response comprising:
 - a processing circuitry;
 - a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:
 - receive an incident input based on a cybersecurity event;
 - generate a prompt for a large language model (LLM) based on the received incident input;

US 12,001,549 B1

21

configure the LLM to generate an output based on the generated prompt;
map the received incident input into a scenario of a plurality of scenarios based on the output of the LLM, wherein each scenario is associated with an incidence response;
generate a query based on the received incident input and the mapped scenario;
execute the query on a security database, the security database including a representation of a computing environment; and
initiate a mitigation action based on a result of the executed query.

13. The system of claim 12, wherein the incident input includes any one of:
a query, a statement, and a combination thereof.

14. The system of claim 12, wherein the LLM is trained on any one of:
a data schema utilized in representing the computing environment, incident data classified to a scenario, the plurality of scenarios, and a combination thereof.

15. The system of claim 14, wherein the memory contains further instructions that, when executed by the processing circuitry for generating the query, further configure the system to:

generate a second prompt for the LLM which when executed by the LLM outputs the query, wherein the second prompt is generated based on any one of:
the received incident input, the data schema, the plurality of scenarios, and a combination thereof.

16. The system of claim 14, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

train the LLM further on a plurality of database queries, each database query executable on the security database.

17. The system of claim 14, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

22

receive a user input through a user interface, the user interface configured to render a graphical representation of the plurality of scenarios; and
select a sub-scenario of a scenario of the plurality of scenarios based on the received user input.

18. The system of claim 17, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:
generate a prompt for the LLM based on the received user input, the prompt, when executed configuring the LLM to output a sub-scenario selection.

19. The system of claim 18, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:
process a user input through a component of a graphical user interface to initiate investigation of a custom incident; and
generate a request to receive additional contextual information wherein the custom incident is unrelated to any specific resource.

20. The system of claim 17, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

traverse the security database to detect a cybersecurity finding, the finding associated with a resource, the resource associated with an incident of the incident response.

21. The system of claim 20, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

receive a user input through a component of a graphical user interface to initiate generation of an explanation of the security finding;

utilize the LLM to generate an explanation of the security finding, the explanation including any one of:

a base observation regard the security finding, an analysis of a symptomatic nature of the security finding, and a combination thereof; and

render for display the generated explanation of the security finding.

* * * * *

EXHIBIT E



(12) **United States Patent**
Cohen et al.

(10) **Patent No.:** US 12,003,529 B1
(45) **Date of Patent:** Jun. 4, 2024

(54) **TECHNIQUES FOR DETECTING ARTIFICIAL INTELLIGENCE MODEL CYBERSECURITY RISK IN A COMPUTING ENVIRONMENT**

(71) Applicant: **Wiz, Inc.**, New York, NY (US)

(72) Inventors: **Amitai Cohen**, Kfar Saba (IL); **Barak Sharoni**, Tel Aviv (IL); **Shir Tamari**, Tel Aviv (IL); **George Pisha**, Giv'atayim (IL); **Itay Arbel**, Tel Aviv (IL); **Daniel Velikanski**, Tel Aviv (IL); **Yaniv Shaked**, Tel Aviv (IL)

(73) Assignee: **Wiz, Inc.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **18/584,659**

(22) Filed: **Feb. 22, 2024**

(51) **Int. Cl.**
H04L 9/40 (2022.01)

(52) **U.S. Cl.**
CPC **H04L 63/1441** (2013.01); **H04L 63/1433** (2013.01)

(58) **Field of Classification Search**
CPC H04L 63/1441; H04L 63/1433
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

- 10,558,823 B2 2/2020 Schroeder et al.
- 11,935,416 B1 * 3/2024 Motamedi G07C 5/008
- 11,936,622 B1 * 3/2024 Gonshorowitz H04L 63/1433

- 11,936,785 B1 * 3/2024 Shemesh H04L 9/088
- 11,941,054 B2 * 3/2024 Shu H04L 63/1425
- 11,947,698 B2 * 4/2024 Struttmann H04L 9/3239
- 11,949,690 B2 * 4/2024 Lichtenstein H04L 63/14
- 2019/0050683 A1 * 2/2019 Gupta Hyde G06V 10/7747
- 2021/0258160 A1 * 8/2021 Kannan H04L 9/32
- 2022/0030009 A1 1/2022 Hasan
- 2022/0345457 A1 * 10/2022 Jeffords G06F 21/6218
- 2023/0269272 A1 * 8/2023 Dambrot H04L 63/1408
- 726/22
- 2023/0289604 A1 * 9/2023 Chan G06F 21/577
- 2023/0351026 A1 * 11/2023 Cross G06F 9/455
- 2024/0064159 A1 * 2/2024 Crabtree H04L 63/0876
- 2024/0080329 A1 * 3/2024 Reed H04L 67/306
- 2024/0080338 A1 * 3/2024 Crabtree H04L 9/0643
- 2024/0089272 A1 * 3/2024 Gilad H04L 63/1416
- 2024/0098072 A1 * 3/2024 Verzun H04L 63/123
- 2024/0104118 A1 * 3/2024 Herzberg G06F 16/285
- 2024/0104235 A1 * 3/2024 Herzberg G06F 16/9024
- 2024/0104240 A1 * 3/2024 Herzberg G06F 16/9024
- 2024/0106846 A1 * 3/2024 Kapoor G06F 16/9038
- 2024/0106847 A1 * 3/2024 Yadav H04L 63/1433

* cited by examiner

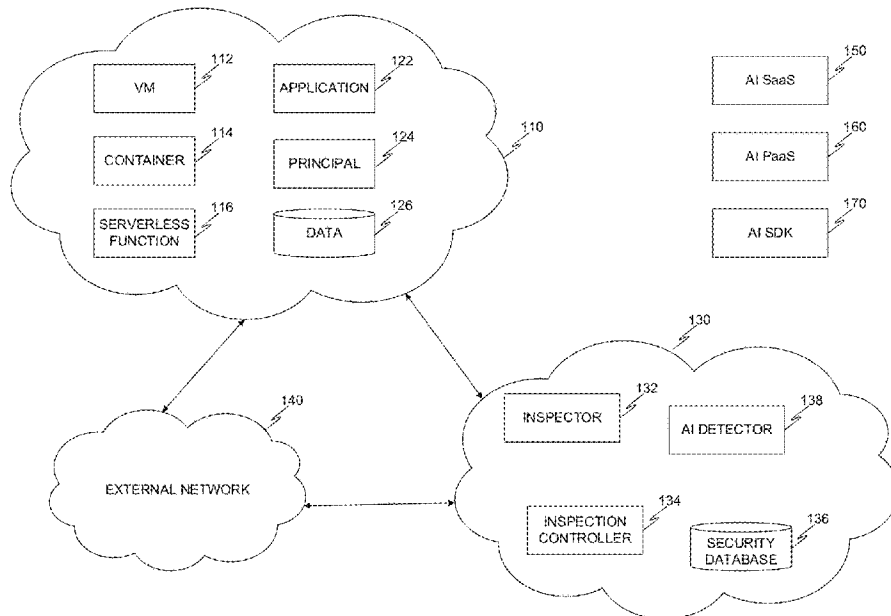
Primary Examiner — Stephen T Gundry

(74) Attorney, Agent, or Firm — M&B IP Analysts, LLC

(57) **ABSTRACT**

A system and method for detecting a cybersecurity risk of an artificial intelligence (AI), is presented. The method includes: inspecting a computing environment for an AI model deployed therein; generating a representation of the AI model in a security database, the security database including a representation of the computing environment; inspecting the AI model for a cybersecurity risk; generating a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and initiating a mitigation action based on the cybersecurity risk.

17 Claims, 7 Drawing Sheets



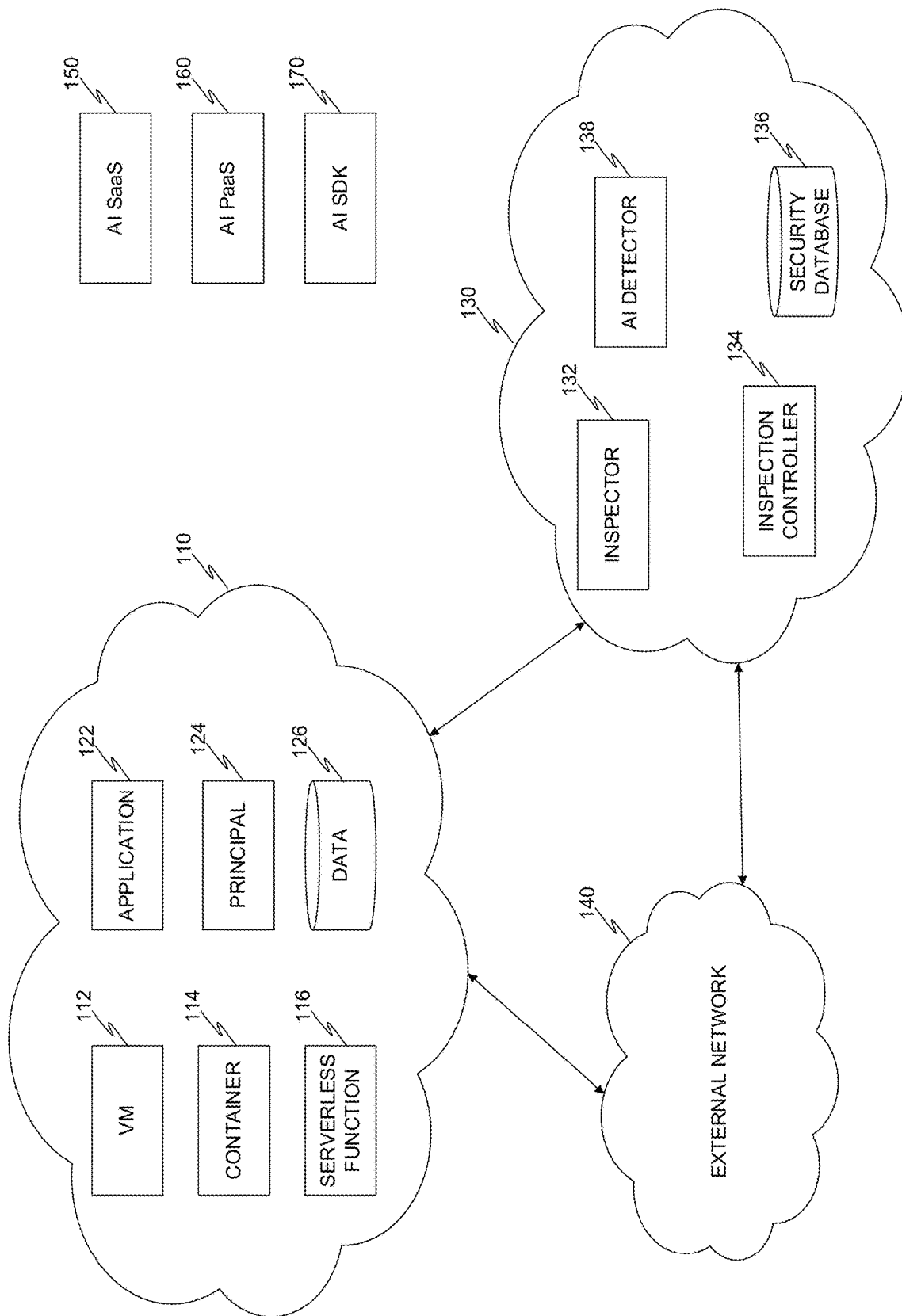


FIGURE 1

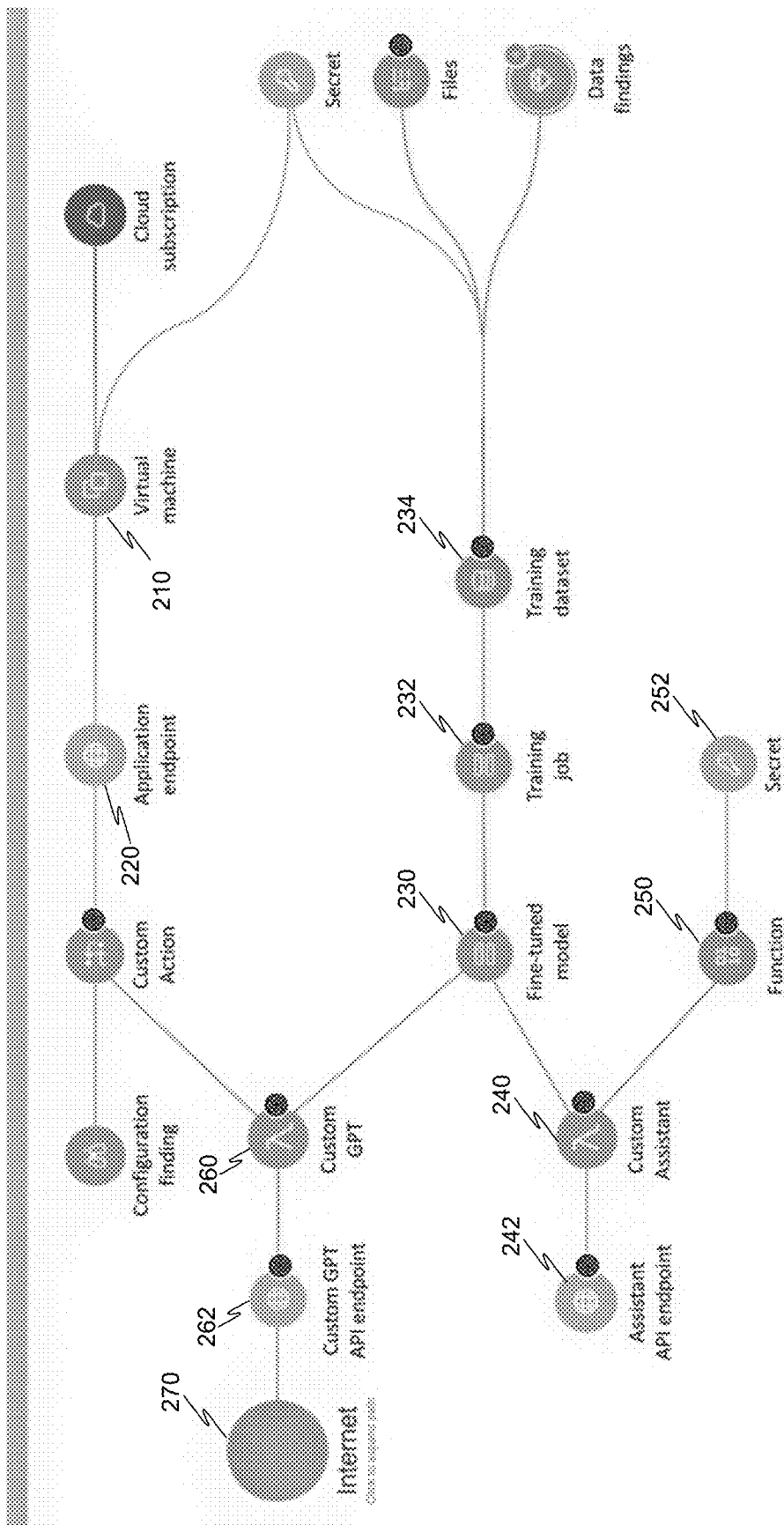


FIGURE 2

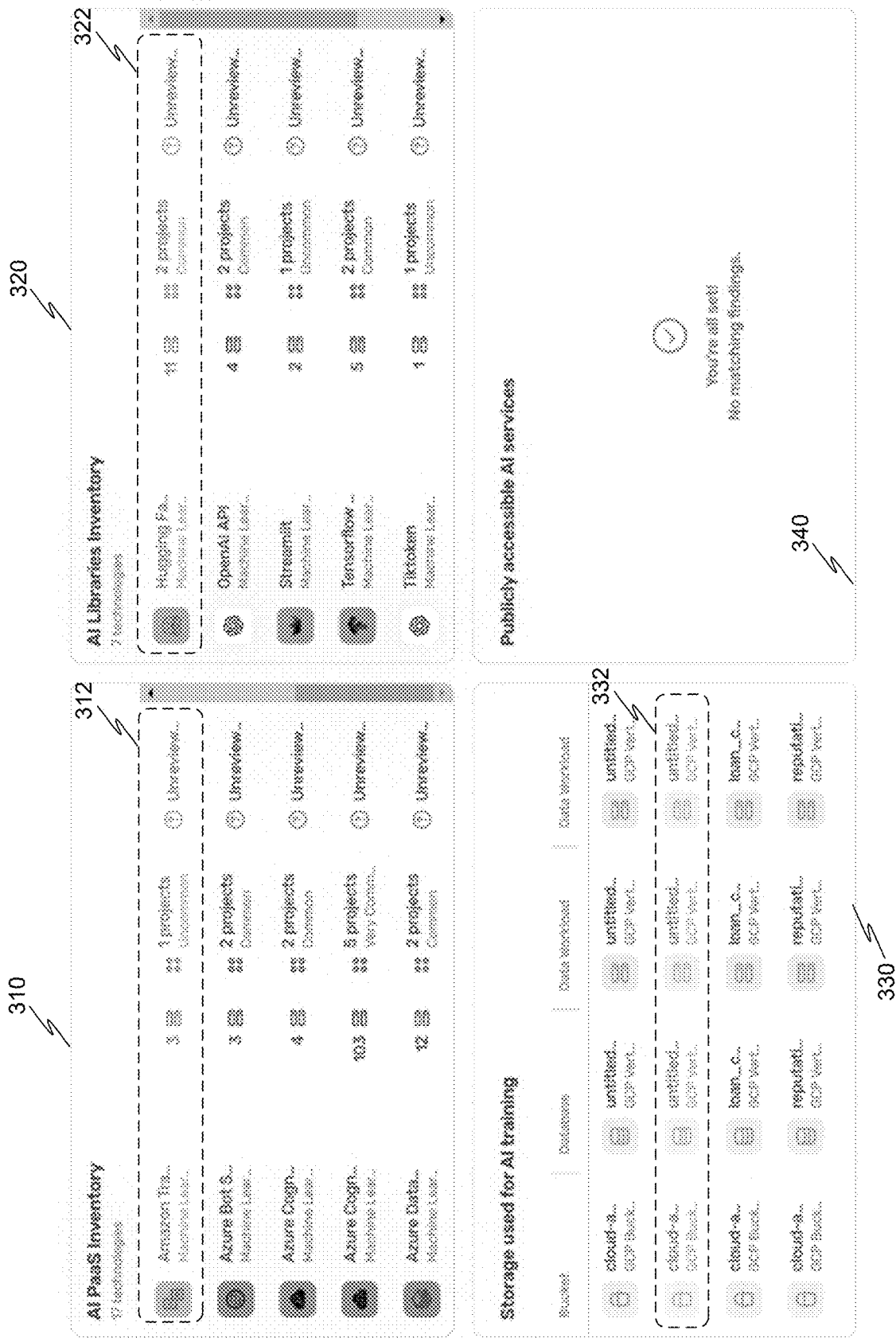


FIGURE 3

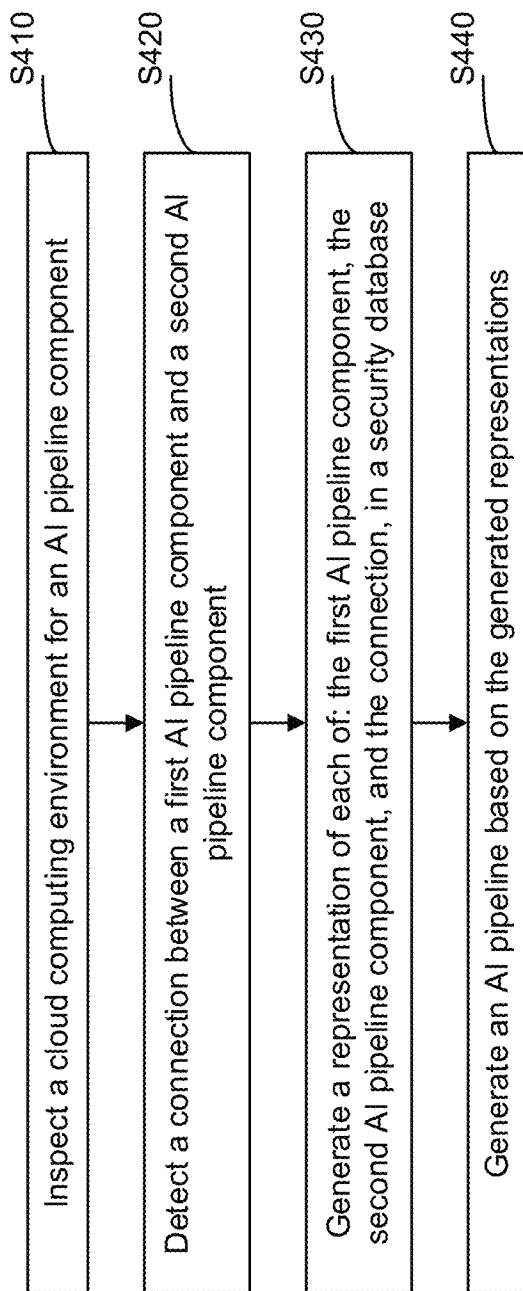


FIGURE 4

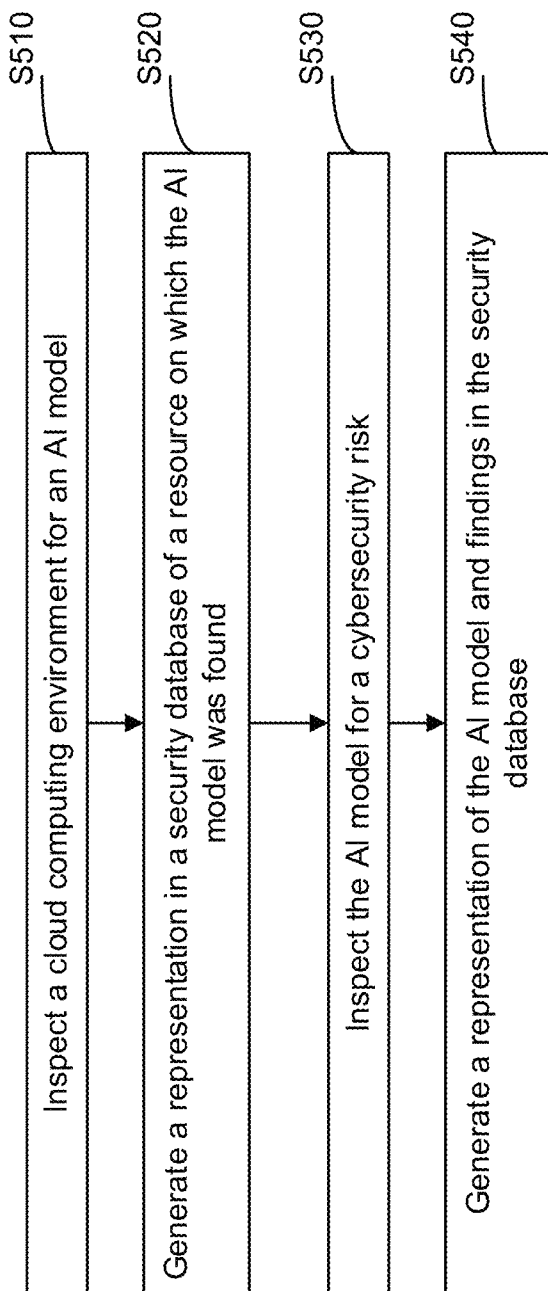


FIGURE 5

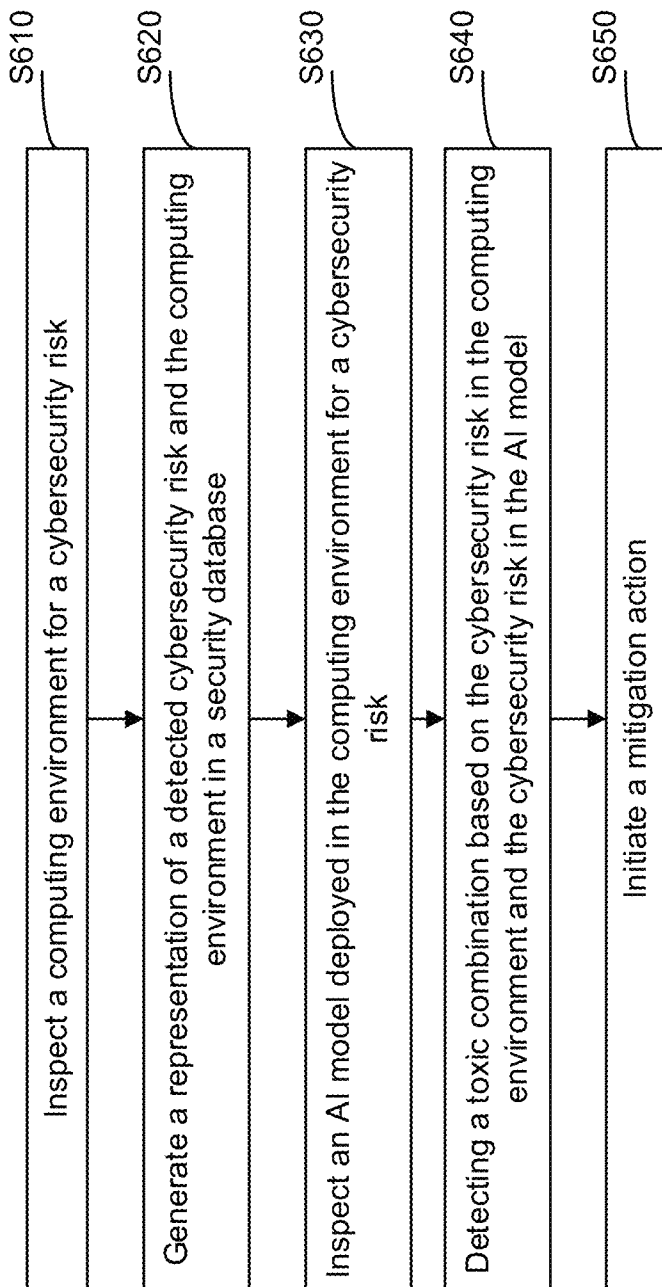


FIGURE 6

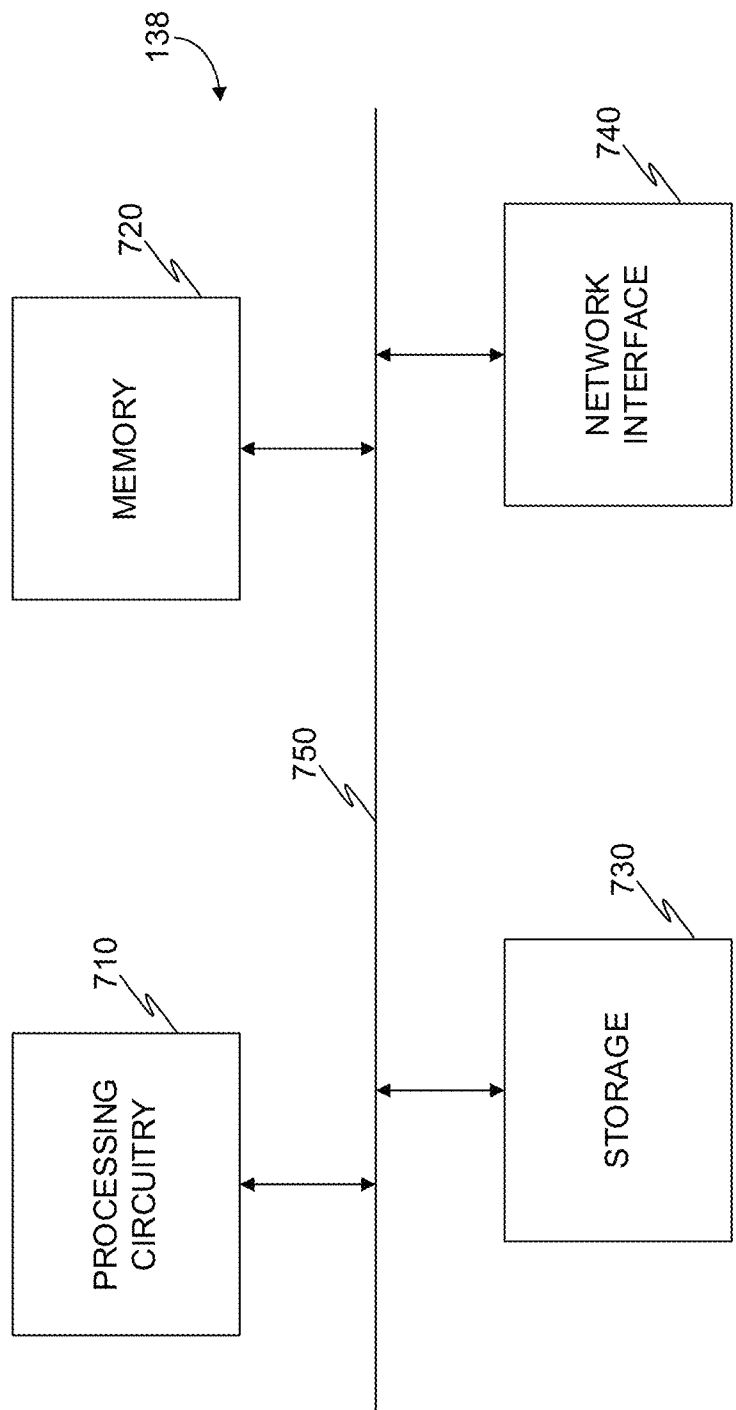


FIGURE 7

US 12,003,529 B1

1

**TECHNIQUES FOR DETECTING
ARTIFICIAL INTELLIGENCE MODEL
CYBERSECURITY RISK IN A COMPUTING
ENVIRONMENT**

TECHNICAL FIELD

The present disclosure relates generally to detecting artificial intelligence models, and specifically to implementing cybersecurity monitoring solutions on an AI development pipeline of a cloud computing environment.

BACKGROUND

Artificial intelligence (AI) applications are increasingly prevalent, as costs of computing hardware have plummeted significantly, and as AI models have improved their computational resource consumption, leading to an overall performance improvement.

This explosion in AI applications, with a rush to deploy AI solutions in various endeavors, presents new cybersecurity risks. Security teams lack knowledge and experience in AI systems, which can lead to vulnerabilities in implementations. AI systems are complex and ever-evolving, require new software tools that security teams may not be aware of, and also do not always have cybersecurity awareness about.

For example, an AI model may leak data, for example exposing sensitive data, secrets, etc. An AI model may be vulnerable to manipulation, such as by poisoning the training data. As with any rapidly evolving technology, the pace of evolution means that attackers can find an advantage over security teams.

It would therefore be advantageous to provide a solution that would overcome the challenges noted above.

SUMMARY

A summary of several example embodiments of the disclosure follows. This summary is provided for the convenience of the reader to provide a basic understanding of such embodiments and does not wholly define the breadth of the disclosure. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor to delineate the scope of any or all aspects. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later. For convenience, the term "some embodiments" or "certain embodiments" may be used herein to refer to a single embodiment or multiple embodiments of the disclosure.

A system of one or more computers can be configured to perform particular operations or actions by virtue of having software, firmware, hardware, or a combination of them installed on the system that in operation causes or cause the system to perform the actions. One or more computer programs can be configured to perform particular operations or actions by virtue of including instructions that, when executed by data processing apparatus, cause the apparatus to perform the actions.

In one general aspect, a method for detecting a cybersecurity risk of an artificial intelligence (AI) may include inspecting a computing environment for an AI model deployed therein. A method for detecting a cybersecurity risk of an artificial intelligence (AI) may also include generating a representation of the AI model in a security database, the security database including a representation of

2

the computing environment. A method for detecting a cybersecurity risk of an artificial intelligence (AI) may furthermore include inspecting the AI model for a cybersecurity risk. A method for detecting a cybersecurity risk of an artificial intelligence (AI) may in addition include generating a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk. A method for detecting a cybersecurity risk of an artificial intelligence (AI) may moreover include initiating a mitigation action based on the cybersecurity risk. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. The method further may include: generating an inspectable disk based on an original disk of a resource deployed in the computing environment; and inspecting the inspectable disk for the AI model. The method further may include: detecting an artifact of the AI model on the inspectable disk. The method further may include: inspecting the AI model to detect an AI model configured to execute a code object. The method further may include: detecting a metadata of the AI model, where the metadata indicates that the AI model is a cybersecurity risk. The method further may include: detecting in the AI model any one of: a secret, a certificate, a code, and any combination thereof. The method further may include: generating a lateral movement path, where the lateral movement path includes the AI model, and a representation of a resource, where the resource is accessible utilizing the AI model. The method further may include: applying a policy on the representation of the computing environment. The method further may include: initiating the mitigation action based on the policy. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

In one general aspect, a non-transitory computer-readable medium storing a set of instructions for detecting a cybersecurity risk of an artificial intelligence (AI), the set of instructions may include one or more instructions that, when executed by one or more processors of a device, cause the device to: inspect a computing environment for an AI model deployed therein; generate a representation of the AI model in a security database, the security database including a representation of the computing environment; inspect the AI model for a cybersecurity risk; generate a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and initiate a mitigation action based on the cybersecurity risk. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

In one general aspect, a system for detecting a cybersecurity risk of an artificial intelligence (AI) may include a processing circuitry. A system for detecting a cybersecurity risk of an artificial intelligence (AI) may also include a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to: inspect a computing environment for an AI model deployed therein. A system for detecting a cybersecurity risk of an artificial intelligence (AI) may in addition generate a representation of the AI model in a security database, the

US 12,003,529 B1

3

security database including a representation of the computing environment. A system for detecting a cybersecurity risk of an artificial intelligence (AI) may moreover inspect the AI model for a cybersecurity risk. A system for detecting a cybersecurity risk of an artificial intelligence (AI) may also generate a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk. A system for detecting a cybersecurity risk of an artificial intelligence (AI) may furthermore initiate a mitigation action based on the cybersecurity risk. Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods.

Implementations may include one or more of the following features. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate an inspectable disk based on an original disk of a resource deployed in the computing environment; and inspect the inspectable disk for the AI model. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: detect an artifact of the AI model on the inspectable disk. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: inspect the AI model to detect an AI model configured to execute a code object. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: detect a metadata of the AI model, where the metadata indicates that the AI model is a cybersecurity risk. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: detect in the AI model any one of: a secret, a certificate, a code, and any combination thereof. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: generate a lateral movement path, where the lateral movement path includes the AI model, and a representation of a resource, where the resource is accessible utilizing the AI model. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: apply a policy on the representation of the computing environment. The system where the memory contains further instructions which when executed by the processing circuitry further configure the system to: initiate the mitigation action based on the policy. Implementations of the described techniques may include hardware, a method or process, or a computer tangible medium.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter disclosed herein is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the disclosed embodiments will be apparent from the following detailed description taken in conjunction with the accompanying drawings.

FIG. 1 is an example schematic illustration of a cloud computing environment for supporting an artificial intelligence (AI) pipeline, utilized to describe an embodiment.

4

FIG. 2 is an example security graph representation of an AI pipeline, implemented in accordance with an embodiment.

FIG. 3 is an example dashboard visualization showing an AI software bill of materials (SBOM), implemented in accordance with an embodiment.

FIG. 4 is an example flowchart of a method for detecting an artificial intelligence software pipeline, implemented in accordance with an embodiment.

FIG. 5 is an example flowchart of a method for inspecting an AI model deployed in a computing environment, implemented in accordance with an embodiment.

FIG. 6 is an example flowchart for initiating a mitigation action on a cybersecurity risk of an AI model, implemented in accordance with an embodiment.

FIG. 7 is an example schematic diagram of an AI detector according to an embodiment.

DETAILED DESCRIPTION

It is important to note that the embodiments disclosed herein are only examples of the many advantageous uses of the innovative teachings herein. In general, statements made in the specification of the present application do not necessarily limit any of the various claimed embodiments. Moreover, some statements may apply to some inventive features but not to others. In general, unless otherwise indicated, singular elements may be in plural and vice versa with no loss of generality. In the drawings, like numerals refer to like parts through several views.

The various disclosed embodiments include a method and system for generating an AI pipeline. In an embodiment, an AI pipeline representation is generated. In some embodiments, the AI pipeline representation includes AI components detected across multiple cloud computing environments. In certain embodiments, the multiple cloud computing environments are not otherwise connected.

FIG. 1 is an example schematic illustration of a cloud computing environment for supporting an artificial intelligence (AI) pipeline, utilized to describe an embodiment. In an embodiment, a cloud computing environment **110** includes cloud entities, such as resources, principals, and the like.

In certain embodiments, the cloud computing environment **110** is implemented as a tenant, as a virtual private cloud (VPC), as a virtual network (VNet), a combination thereof, and the like. In some embodiments, the cloud computing environment **110** is deployed on a cloud computing infrastructure, such as Amazon® Web Service (AWS), Google® Cloud Platform (GCP), Microsoft® Azure, and the like.

In an embodiment, a resource is a cloud entity which exposes hardware resources (e.g., provides access to a processor, a memory, a storage, a combination thereof, and the like), exposes services, exposes an application programming interface (API), a combination thereof, and the like.

In some embodiments, a principal is a cloud entity which is authorized to initiate actions in the cloud computing environment **110**, authorized to act on a resource, a combination thereof, and the like.

For example, according to an embodiment, a principal is a user account, a service account, a role, a combination thereof, and the like. In some embodiments, a resource is a virtual machine, a software container, a serverless function, an application, a database, a combination thereof, and the like.

US 12,003,529 B1

5

In certain embodiments, a virtual machine **112** is deployed in the cloud computing environment **110**. In some embodiments, the virtual machine **112** is implemented as Oracle® VirtualBox®, for example. In an embodiment, the virtual machine **112** is associated with a disk (not shown).

In an embodiment, a software container **114** is deployed in the cloud computing environment **110**. In some embodiments, the software container **114** is implemented utilizing a Kubernetes® Platform, a Docker® container engine, a combination thereof, and the like.

According to an embodiment, a serverless function **116** is deployed in the cloud computing environment **110**. In certain embodiments, the serverless function **116** is implemented as an Amazon® Lambda® service.

In some embodiments, an application **122** is deployed in the cloud computing environment. In an embodiment, an application is deployed utilizing a resource, a plurality of resources, and the like. In some embodiments, an application **122** includes a software library, a software binary, an executable code, a combination thereof, and the like.

In an embodiment, the application **122** is a component of an artificial intelligence (AI) pipeline. According to an embodiment, an AI pipeline is utilized to provide a service utilizing an AI model. For example, in an embodiment, an AI pipeline includes software components, such as an AI model, a training data, an API, a combination thereof, and the like. In some embodiments, an AI pipeline includes resource components such as a serverless function **116**, a software container **114**, a virtual machine **112**, a combination thereof, and the like.

For example, in an embodiment, an application **122** is hosted on a virtual machine **112**. A serverless function **116** is configured to receive a prompt, which is then directed to an AI model, such as a large language model (LLM) to produce an output.

In an embodiment, the AI model is trained on a data set stored, for example, in a database **126**. In some embodiments, an AI model is trained on a database **126** in a first cloud computing environment (e.g., a development environment), and the application **122** is deployed in a second cloud computing environment (e.g., a production environment). According to some embodiments, this presents a challenge, for example where the database **126** includes sensitive information which is used to train the AI model, but the data itself should not become exposed through use of the AI model.

In an embodiment, an AI model includes an artificial neural network (ANN), a recurrent neural network (RNN), a convolutional neural network (CNN), a generative adversarial neural network (GAN), a Bayesian network, a hidden Markov model, a large language model (LLM), a combination thereof, and the like.

In an embodiment, an AI pipeline includes resources, principals, and the like, which are utilized in providing a service based, for example, on an AI model. In an embodiment, a first component of the AI pipeline is connected to a second component of the AI pipeline, for example by a network policy allowing the first component to access the second component.

In some embodiments, the cloud computing environment **110** is accessible by an external network **140**. In an embodiment, the external network **140** includes computing devices, user accounts, and the like, which are not affiliated with the cloud computing environment **110**, but receive a service, access to a resource, access to an application, a combination thereof, and the like. According to an embodiment, the external network **140** is, for example, the Internet.

6

In certain embodiments, the cloud computing environment **110** is monitored for cybersecurity objects, threats, risks, and the like, by an inspection environment **130**. In an embodiment, the inspection environment **130** is configured to monitor the cloud computing environment **110** for cybersecurity objects, threats, risks, combinations thereof, and the like.

In some embodiments, an inspector **132** is configured to inspect a workload of the cloud computing environment **110** for a cybersecurity object. An inspector **132** is implemented, in certain embodiments, as a resource, such as a virtual machine, a software container, a serverless function, a combination thereof, and the like. In some embodiments, it is advantageous to implement the inspector **132** as a node in software container, as this allows scaling of the inspection environment **130** to accommodate the inspection needs of the cloud computing environment **110**.

In certain embodiments, an inspector **132** is configured to inspect a workload for a cybersecurity object such as a hash, a malware, a signature, an application, an operating system, a binary, a library, a computer code object, a code object of an infrastructure as code file, a registry file, a password stored as text (e.g., cleartext, plaintext, etc.), a certificate, a cryptographic key, a vulnerability, an exposure, a misconfiguration, a combination thereof, and the like.

For example, according to an embodiment, an inspector **132** is configured to detect a misconfiguration based on configuration rules. In an embodiment, a configuration rule is applied to a representation of a cybersecurity object, a representation of a workload, a representation of a principal, a combination thereof, and the like.

In an embodiment, an inspector **132** is configured to detect a component of an AI pipeline. In an embodiment, a component of an AI pipeline includes a model (e.g., stored as a file on a storage, a repository, and the like), a database, a training dataset, a serverless function configured to train the AI model using the training dataset stored on the database, an AI application, an AI application endpoint, an API, a combination thereof, and the like.

In certain embodiments, the inspector **132** is configured to detect a network path between an external network **140**, and a component of the AI pipeline. For example, in an embodiment, a network path includes resources, applications, and the like, between the external network path **140**, and the application **122**. In some embodiments, where the network path exposes a resource, exposes data, and the like, to the external network, the network path is determined to be an exposure path.

In some embodiments, the inspector **132** is configured to query a cloud API to detect cloud entities, such as resources and principals, in the cloud computing environment **110**. In certain embodiments, the inspector **132** is configured to query a Platform as a Service (PaaS) to detect an assistant, a list of assistant files, a list of runs belonging to a thread, a list of files associated with a principal, a list of models, a list of available AI models, metadata related to each AI model, a list of messages for a thread, an action, a function, a training dataset, a document, a training job, a thread, a combination thereof, and the like.

According to an embodiment, the inspector **132** is configured to inspect a network and detect network objects, such as resources deployed in a cloud computing environment **110**. For example, in an embodiment, a network object is a server, a firewall, a web-access firewall (WAF), a load balancer, a gateway, a proxy server, a combination thereof, and the like.

In some embodiments, the inspector **132** is configured to detect identities deployed in the cloud computing environment **110**. In an embodiment, an identity is a user account, a service account, a role, an email address, a security credential associated with an identity, a combination thereof, and the like.

In certain embodiments, the inspector **132** is configured to inspect a registry, a command-line interface (CLI), a PaaS, a SaaS, an SDK, a network, an IAM service, a resource, a combination thereof, and the like, to detect a cybersecurity object.

According to an embodiment, an inspector **132** is configured to inspect a first cloud computing environment (e.g., a production environment) and a second cloud computing environment (e.g., a development environment). This is advantageous as AI pipelines are often split among a plurality of computing environments. For example, according to an embodiment, an AI model is trained in a development environment, and deployed in a production environment.

In some embodiments, it is advantageous to generate a pipeline as a view which allows to see a complete AI pipeline through, and including, multiple cloud computing environments. This allows visibility into, for example, an AI model trained on sensitive data, which could potentially leak in a production environment, despite the sensitive data being actually stored in the development environment.

In an embodiment, the inspection environment **130** further includes an inspection controller **134**. In some embodiments, the inspection controller **134** is configured to provision inspector workloads in response to detecting a demand for inspection workloads. In certain embodiments, the inspection controller **134** is configured to generate a copy, a clone, a snapshot, a combination thereof, and the like, in the cloud computing environment **110**, in the inspection environment **130**, and the like, based on a disk associated with a workload in the cloud computing environment **110**.

In certain embodiments, it is advantageous to generate a clone of a disk, as a clone utilizes less resources than a snapshot, for example. In an embodiment, a clone is generated by generating a pointer to the same storage address of the original data, thereby providing instantaneous access to the 'cloned' disk. In certain embodiments, access to a snapshot is only provided after the snapshot process has completed generating the snapshot.

In an embodiment, the inspection environment **130** further includes a security database **136**. In some embodiments, the security database **136** is configured to store a representation of the cloud computing environment **110**, for example based on a predetermined data schema. In an embodiment, a unified data schema is utilized, including templates for example for representing a resource, representing a principal, representing an enrichment, representing a cybersecurity issue, a combination thereof, and the like.

In some embodiments, the security database **136** is implemented as a columnar database, a SQL database a non-SQL database, a table, a plurality of tables, a graph database, combinations thereof, and the like. For example, in an embodiment, a security database **136** is implemented as a graph database, such as Neo4j®.

In certain embodiments, the inspection environment **130** further includes an AI detector **138**. In an embodiment, the AI detector **138** is configured to generate an AI pipeline. In some embodiments, an AI pipeline is a representation, for example stored in a security database **136**. In an embodiment, the AI pipeline includes representations of AI components, and connections between the AI components.

For example, according to an embodiment, a dataset used for training an AI model is an AI component. In some embodiments, a dataset is determined to be used for training by detecting that a dataset stored in, for example, a storage bucket, is accessed by a serverless function configured to train an AI model. In an embodiment, the storage bucket is represented as a resource in the security database, and the representation of the storage bucket is further associated with the AI pipeline.

In certain embodiments, the AI detector **138** is configured to detect various AI components across multiple cloud computing environments. For example, in an embodiment, an AI model is trained in a first cloud computing environment (e.g., a development environment) and deployed in a second cloud computing environment (e.g., a production environment).

Therefore, in some embodiments, an AI pipeline includes components which are deployed in a plurality of cloud computing environments. In certain embodiments, the AI detector **138** is configured to detect that a component deployed in a first cloud computing environment is accessible by, or is configured to access, a component deployed in a second cloud computing environment.

In some embodiments, the cloud computing environment **110** is configured to access a service, access a platform, utilize a software development kit (SDK), and the like. For example, in an embodiment, the cloud computing environment **110** is configured to access an AI SaaS **150**, an AI PaaS **160**, an AI SDK **170**, a combination thereof, and the like.

According to an embodiment, an AI SaaS **150** is, for example Google® TensorFlow®, and the like. In some embodiments, an AI PaaS **160** is, for example, OpenAIR, Hugging Face®, and the like.

FIG. 2 is an example security graph representation of an AI pipeline, implemented in accordance with an embodiment. In an embodiment, a security database is implemented as a graph database, based on a unified data schema. In some embodiments, the security database includes a plurality of nodes and connections between the nodes.

For example, in an embodiment, a node represents a resource, a principal, a cybersecurity object, a misconfiguration, a vulnerability, an exposure, a malware object, an enrichment, a remediation action, combinations thereof, and the like.

In some embodiments, a connection between nodes indicates a relationship between the objects represented by the nodes. For example, in an embodiment, a node representing an application **220** is connected to a node representing a virtual machine **210**, indicating that the application represented by application node **220** is deployed on the virtual machine represented by the virtual machine node **210**.

In an embodiment, a representation of a fine-tuned model **230** is generated in response to detecting a fine-tuned AI model. In some embodiments, detecting a fine-tuned model includes accessing an API of an AI SaaS provider, and pulling a list of AI models accessible by an account. In an embodiment, the account identifier is provided, for example, to an AI detector.

In certain embodiments, detecting a fine-tuned model includes detecting API calls in a networked environment directed to a known predefined AI SaaS, AI PaaS, combination thereof, and the like.

In an embodiment, a serverless function is utilized to implement a custom assistant application. In some embodiments, the serverless function is detected, for example by an inspector configured to detect virtual instances in a computing environment. In an embodiment, a representation of the

serverless function is generated in a security database, for example as custom assistant node **240**.

In an embodiment, an inspector is configured to detect that the custom assistant represented by the custom assistant node **240** accesses the fine-tuned model represented by fine-tuned model node **230**. In some embodiments, the serverless function deploying the custom assistant includes an API through which the serverless function is configured to receive calls, represented by AI endpoint **242**.

In certain embodiments, the serverless function is further configured to access a function, represented by function node **250**, which in turn includes a secret, represented by secret node **252**. In some embodiments, the serverless function is deployed in a production environment.

In an embodiment, a trainer is configured to train an AI model, such as a generative pre-trained transformer (GPT) model, into the fine-tuned model represented by the fine-tuned model node **230**. In some embodiments, the trainer is implemented as a virtual instance in development environment, for example as a serverless function. In an embodiment, the trainer is represented by a training job node **232**.

In certain embodiments, a first inspector is configured to detect the serverless function in the production environment (e.g., represented by the custom assistant node **240**), and a second inspector is configured to detect the trainer (represented by the training job node **232**) in the development environment.

In an embodiment, an AI detector is configured to detect a connection between the trainer and the serverless function (i.e., between the training job node **232** and the custom assistant node **240**). For example, in an embodiment, the AI detector is configured to detect that the trainer is configured to store a trained AI model in a bucket, which is accessed by the serverless function.

Generating a representation including a connection between multiple computing environment to generate a pipeline is advantageous, as this allows to detect vulnerabilities, exposures, and the like, including risk combinations (e.g., vulnerability and exposure) which are not otherwise apparent when a representation includes only components of one computing environment, where a pipeline utilizes multiple components from multiple computing environments.

According to an embodiment, a trainer is configured to train an AI model using a training dataset, represented by training dataset node **234**. In some embodiments, the training dataset is stored, for example as a file, a plurality of files, and the like, on a storage, such as a network-accessible storage device, a cloud based storage service, a combination thereof, and the like.

In certain embodiments, the training dataset includes sensitive data. In an embodiment, it is advantageous to maintain a separation between sensitive data in a development environment, and an application (such as the custom assistant, a custom GPT, etc.) in a production environment, where the application is exposed to an external network, such as the Internet.

For example, in an embodiment, an API endpoint (represented by custom GPT API endpoint **262**) is configured to receive an input from a user account over an external network (i.e., external to where the custom GPT is deployed). In some embodiments, the API endpoint is an endpoint for a custom GPT (represented by custom GPT node **260**). In an embodiment, the custom GPT is configured to utilize a fine-tuned model (represented by fine-tuned model node **230**).

In some embodiments, the custom GPT is deployed in a production environment, while the trainer is deployed in a development environment. Representing the pipeline allows to detect a cybersecurity risk which is not apparent when each computing environment is viewed individually. In this example, the custom GPT is exposed to an external network (i.e., represented by the Internet node **270**). In an embodiment, the custom GPT is configured to utilize the fine-tuned AI model which utilizes a training dataset that includes sensitive data.

In certain embodiments, this AI pipeline indicates a cybersecurity risk of exposing sensitive data through the custom GPT, by utilizing the fine-tuned model trained on the sensitive data.

FIG. 3 is an example dashboard visualization showing an AI software bill of materials (SBOM), implemented in accordance with an embodiment. In an embodiment, an AI SBOM is generated based on detection of a plurality of AI components deployed in a computing environments, in a combination of environments, etc.

For example, in an embodiment, an AI detector is configured to generate an AI SBOM based on data detected by an inspector. In some embodiments, an AI SBOM includes identifiers of software, an application, an endpoint, a storage device, a library, a binary, a SaaS application, a PaaS connection, a resource identifier, a principal identifier, a dataset identifier, a database identifier, combinations thereof, and the like.

In an embodiment, the dashboard includes an AI PaaS Inventory widget **310**. In some embodiments, the AI PaaS Inventory widget **310** includes a plurality of rows, each row representing an AI PaaS connected to the computing environment. For example, a first row **312** includes a visual representation of an Amazon® Transcribe® service. In some embodiments, the AI PaaS inventory widget **310** includes representations of Azure® bot service, Azure cognitive services, and the like.

In certain embodiments, the dashboard includes an AI library inventory widget **320**. In an embodiment, the AI library inventory widget includes various platforms, systems, libraries, and the like, which are accessible by the computing environment. For example, in an embodiment, a first row **322** includes a representation of Hugging Face®. In some embodiments, the AI library inventory widget includes representations of, for example, OpenAI® API, Streamlit, Google® Tensorflow®, Tiktoken, a combination thereof, and the like.

In an embodiment, the dashboard further includes a storage widget **330**. In some embodiments, the storage widget **330** includes a visual representation of a storage device, storage service, and the like, utilized by a computing environment for AI applications, AI data, AI services, etc. For example, in an embodiment, a first row **332** represents a Google® Cloud Platform (GCP) bucket. In some embodiments, a bucket is utilized to store, for example, training data.

In certain embodiments, the dashboard includes an AI services widget **340**. In some embodiments, the AI services widget **340** includes a visual representation of AI services which are publicly accessible (e.g., accessible from a network external to the computing environment, such as the Internet). In an embodiment, an AI service is determined to be a publicly accessible service in response to detecting an API endpoint associated with the AI service.

In certain embodiments, an AI detector is configured to detect an AI pipeline, and further to generate an AI dashboard based on the detected AI pipeline. In some embodi-

US 12,003,529 B1

11

ments, a widget of the AI dashboard is updated with data based on querying a representation of the AI pipeline, such as stored on a graph in a graph database, for example as discussed in more detail in FIG. 2 above.

FIG. 4 is an example flowchart of a method for detecting an artificial intelligence software pipeline, implemented in accordance with an embodiment. In some embodiments, an AI pipeline is deployed across multiple computing environments, multiple accounts, utilizing multiple resources, principals, SaaS providers, PaaS provides, combinations thereof, and the like. It is therefore advantageous to generate a representation of the entire AI pipeline, including all components, in order to detect cybersecurity risks, mitigate cybersecurity issues, remediate cybersecurity exposures, exploited vulnerabilities, and the like. It is further advantageous as an AI pipeline is often also deployed across time, e.g., the AI model is trained in a first environment at a first time, and deployed in a second environment at a second time. An AI pipeline lends visibility to this process of bridging computing environments, and bridging time.

At S410, a computing environment is inspected for an AI component. In an embodiment, an AI component is utilized by an AI system. For example, in certain embodiments, an AI model utilizes a storage for storing training data, a training workload utilized to train the AI model, an application for providing input to the trained model, an API to receive calls and utilize the application which utilizes the AI model, and API endpoint to connect the API to an external network, various combinations thereof, and the like. In this example, each such component is an AI component.

In an embodiment, an AI component is detected by an inspector configured to inspect a computing environment, such as a cloud computing environment, a hybrid computing environment, a networked computing environment, a combination thereof, and the like.

In some embodiments, an inspector is configured to detect a cybersecurity object, such as an application, an operating system, a nested workload, a certificate, a password, a cryptographic key, an identity, a library, a binary, a software development kit (SDK), a registry file, a combination thereof, and the like.

In certain embodiments, an inspector is configured to inspect a network of a computing environment (i.e., network inspection) and detect network components, such as firewalls, load balancers, gateways, endpoints, open ports, network paths (including, for example, TCP/IP communication, UDP communication, combinations thereof, and the like).

In some embodiments, an inspector is configured to inspect a cloud computing environment (e.g., cloud scanning) to detect various resources, principals, cloud entities, and the like, which are deployed in the cloud computing environment.

In an embodiment, an inspector is configured to perform static analysis of applications, of SDKs, of registry files, infrastructure as code (IaC) code objects, command line interface (CLI) inspection, a combination thereof, and the like. In some embodiments, the inspector is configured to detect AI components by performing such static analysis.

In certain embodiments, an AI detector is configured to query an identity and access management (IAM) service to detect identities, principals, user accounts, service accounts, roles, combinations thereof, and the like, which are utilized as AI components (i.e., in the AI pipeline). In an embodiment, an identity is determined to be utilized as an AI component in response to detecting a permission which

12

authorizes the identity to act on another AI component (e.g., act on a resource of the AI pipeline).

In some embodiments, an identity is determined to be utilized as an AI component, and a further detection is performed to determine if the identity is authorized to access components, resources, and the like, which are not already identified as AI components.

For example, according to an embodiment, a first identity is determined to be utilized in an AI pipeline in response to determining that the first identity includes a permission to initiate an action in a cloud computing environment on a resource identified as an AI component. In an embodiment, the AI detector is configured to detect additional resources which the first identity has a permission to initiate an action thereupon. In some embodiments, the additional resources are further determined to be AI components.

In some embodiments, an AI detector is configured to determine if a component detected by the inspector is an AI component.

At S420, a connection is detected between AI components. In an embodiment, the connection is detected between a first AI component, and a second AI component. In some embodiments, the connection is detected in response to detecting a shared resource that both AI components act on. In certain embodiments, a connection is detected by querying an IAM service to detect resources which a principal (determined to be an AI component) is authorized to act on.

In an embodiment, an AI detector is configured to detect a connection between AI components, based on, for example, a result of an inspector performing inspection in a computing environment.

In some embodiments, the AI detector is configured to detect a connection between a first AI component deployed in a first computing environment (e.g., a production environment), and a second AI component deployed in a second computing environment (e.g., a development environment).

At S430, a representation is generated in a security database. In an embodiment, a representation is generated for each of: the first AI component, for the second AI component, the connection, etc. In some embodiments, a representation is generated based on a unified data model, such that a resource detected in a first computing environment is represented using the same model as a resource detected in a second computing environment which is different than the first computing environment.

In certain embodiments, the security database is implemented as a graph database, such as a Neo4j® database. In an embodiment, a representation is stored as a node in the graph database. In some embodiments, a relationship between a first AI component and a second AI component, is represented as a connection between representations, e.g., a connection between nodes.

At S440, an AI pipeline is generated. In an embodiment, the AI pipeline includes a representation of an AI pipeline, such as a visual representation. In some embodiments, the AI pipeline includes a plurality of AI components, and the connections between such AI components.

In an embodiment, an AI pipeline includes any of: a cybersecurity finding, a representation of an API endpoint, a representation of an application, a representation of an AI model, a representation of a function, a representation of a virtualization, a representation of a training job, a representation of a training dataset, a representation of a secret, a representation of a computing environment, a representation of an account, a representation of a data, a representation of a data type, any combination thereof, and the like.

US 12,003,529 B1

13

In an embodiment, generating an AI pipeline includes generating a visual representation, including a corresponding graphic for each data representation of an AI component, and a graphic representing connections between the different graphics. A visual representation of an AI pipeline is discussed in more detail with respect to FIG. 2, which is an example of one such visual representation.

In some embodiments, it is advantageous to generate a visual representation of an AI pipeline, as AI applications are often deployed across multiple computing environments, include multiple external software providers (e.g., SaaS and PaaS providers), etc.

In certain embodiments, an AI detector is further configured to detect cybersecurity risks based on detecting a plurality of AI components, and further based on detecting a connection between a plurality of AI components.

FIG. 5 is an example flowchart of a method for inspecting an AI model deployed in a computing environment, implemented in accordance with an embodiment. In some embodiments, an AI model includes a software file, a software library, a software binary, a combination thereof, and the like. In certain embodiments, an AI model is a cybersecurity object.

At S510, an AI model is detected. In an embodiment, an inspector is configured to detect an AI model, an artifact of an AI model, a combination thereof, and the like. For example, according to an embodiment, an artifact of an AI model is a file, a software library, a software binary, a folder name, various combinations thereof, and the like, which indicate that a known (e.g., previously determined) AI model is installed on a disk, resource, and the like, which is being inspected.

For example, in an embodiment, an inspector is configured to detect a PyTorch framework (e.g., TAR files, including sys_info, pickle, storages, tensors, etc.), a PT file, a SAFETENSORS file, a ZIP file including a PKL file, an H5 file, a PB file, a KERAS file, an ONNX file, a GGUF file, a GGML file, a combination thereof, and the like.

In some embodiments, a JSON configuration (e.g., "config.json") file is inspected to detect a format predetermined to be associated with an AI model, for example as utilized by HuggingFace.

In certain embodiments, some file names are changed to a hash. For example, where a file name is imported from an external source, such as when using HuggingFace Transformers to fetch a model from HuggingFace Hub, which is an AI model repository. In such an embodiment, an inspector is configured to inspect a JSON (or other markup language) configuration file (e.g., config.json) for a "\model_type\" string which indicates an AI model type. For example, according to an embodiment, detecting a config.json file in a certain (e.g., predetermined) folder, path, etc., is an indicator that a model is stored in the same location.

In other embodiments, files having a certain size, having a size which is less than a predetermined size (e.g., less than 2 KB), and the like, are inspected to detect if they contain therein a "\model_type\" string. As another example, in an embodiment, such files are searched for in paths that match a ".*/.cache/huggingface/hub/./blobs/*" string.

At S520, a representation is generated. In an embodiment, a representation of the detected AI model is generated in a security database, wherein the security database further includes a representation of the computing environment.

In some embodiments, the representation of the computing environment includes a representation generated on a predefined unified data schema. In an embodiment, a resource is represented in a security database, for example

14

by a node in a security graph (e.g., utilizing Neo4j). In certain embodiments, a representation of the AI model is connected to a representation of the resource on which it was detected.

For example, in an embodiment, an inspector detects an AI model based on a file in a disk of the virtual machine. According to an embodiment, a representation of the AI model, and a representation of the virtual machine are each generated in a security database, and connected to each other to indicate that the AI model is detected on the virtual machine.

In some embodiments, the representation of the AI model is enriched in the security database. In an embodiment, an inspection controller, enricher, and the like, are configured to enrich the representation based on data, metadata, and the like, extracted by an inspector, extracted from another source, a combination there, and the like.

In an embodiment, an enrichment includes a file name, a file extension, a file type, extracted metadata from GGUF files, extracted metadata from ONNX files, extracted metadata from KERAS files, local metadata extracted from a configuration file (e.g., architecture, model type, model name, transformer version, license, etc.), author identifier, malware detection list, pickle imports, suspicious pickle imports, various combinations thereof, and the like.

At S530, the AI model is inspected. In an embodiment, the AI model is inspected for a cybersecurity object, a cybersecurity risk, a misconfiguration, a vulnerability, an exposure, a combination thereof, and the like.

For example, according to an embodiment, inspecting the AI model includes determining if the model is configured to run code (e.g., through a pickle file), a list of functions which are allowed, a list of functions which are blocked, and the like.

In an embodiment, inspecting an AI model includes determining if the AI model matches a version, identifier, and the like, of an AI model which is previously determined as a cybersecurity risk.

In certain embodiments, inspecting an AI model includes detecting a file, a code, and the like, which the AI model is configured to execute. In some embodiments, inspecting the AI model includes detecting a secret, a code, and the like, for example stored in a pickle file, which the AI model, when prompted with an appropriate prompt, is configured to output, leading to leaked data, leaked secrets, etc.

In some embodiments, inspection of an AI model includes generating a finding. In an embodiment, a finding is a data record indicating a detected cybersecurity risk, misconfiguration, vulnerability, exposure, combination thereof, and the like.

In an embodiment, a security database, inspection controller, and the like, is configured to generate a lateral movement path. For example, in an embodiment, where a secret is detected in an AI model, a representation of the secret is generated. In some embodiments, the representation of the secret already preexists in the security database, and a representation of the AI model is connected to the preexisting representation of the secret.

In some embodiments, a lateral movement path is generated to include resources, nodes, representations, and the like, between the AI model and another resource. For example, in an embodiment, the secret detected in the AI model is also utilized by an administrator account to access a storage in a cloud computing environment, therefore a lateral movement path includes the AI model, the secret, the administrator account, and the storage.

In certain embodiments, inspecting an AI model further includes detecting an anomaly. For example, according to an embodiment, detecting an anomaly respective of an AI model includes detecting a training set including data not used for training other models. In an embodiment, detecting an anomaly includes detecting a suspicious prompt provided to the AI model (e.g., a prompt utilizing code, attempting to execute code, attempting to extract a secret, etc.).

At S540, a representation of the AI model is generated. In an embodiment, the representation of the AI model is generated in a security graph. In some embodiments, a representation of a finding based on inspection of the AI model is further stored in the security database, and connected to the representation of the AI model.

In certain embodiments, the security representation is utilized to apply a policy on a computing environment, the computing environment having a representation stored in the security database as part of the security representation.

For example, in an embodiment, a policy includes a rule, a conditional rule, a plurality of each, a combination thereof, and the like, which applied to the representation of the computing environment, the representation of the AI model, a representation of a cybersecurity object, a combination thereof, and the like.

FIG. 6 is an example flowchart of a method for initiating a mitigation action on a cybersecurity risk of an AI model, implemented in accordance with an embodiment. In an embodiment, a cybersecurity risk is detected with respect to an AI model utilizing, for example, the method of FIG. 5 above.

At S610, a computing environment is inspected. In an embodiment, the computing environment is inspected for a cybersecurity object, a cybersecurity risk, a cybersecurity threat, a vulnerability, an exposure, a misconfiguration, a combination thereof, and the like.

In an embodiment, a cybersecurity object includes a code object, a secret, a certificate, a hash, a file, a folder, a software artifact, a software application, an operating system, a malware object, a combination thereof, and the like.

In certain embodiments, a cybersecurity object indicates a cybersecurity risk, a cybersecurity threat, a misconfiguration, a vulnerability, an exposure, a combination thereof, and the like.

In an embodiment, inspecting a computing environment includes generating an inspectable disk based on an original disk. In some embodiments, generating an inspectable disk includes generating a clone of an original disk, a snapshot of an original disk, a copy of an original disk, a combination thereof, and the like, and inspecting the cloned disk for a cybersecurity object.

In some embodiments, an inspectable disk is released once inspection is complete. In an embodiment, an inspector, inspection controller, and the like, is configured to generate a finding of a cybersecurity object on an inspectable disk, wherein the finding is a data record stored in a security database. In an embodiment, the finding (i.e., the representation of the finding, the data record, etc.) is connected to a representation of a resource on which the finding is detected.

At S620, a representation is generated. In an embodiment, a representation is generated for a resource, a cybersecurity object detected on the resource, a finding based on a cybersecurity object detected on the resource, a combination thereof, and the like.

In some embodiments, the representation is stored in a security database, wherein the security database is further

configured to store a representation of a computing environment in which the resource is deployed.

In some embodiments, a representation is enriched in the security database. In an embodiment, an inspection controller, enricher, and the like, are configured to enrich the representation based on data, metadata, and the like, extracted by an inspector, extracted from another source, a combination thereof, and the like.

At S630, an AI model is inspected. According to an embodiment, an AI model is detected on a resource, in a resource, etc., wherein the resource is deployed in the computing environment. In an embodiment, the AI model is inspected for a cybersecurity object, a cybersecurity risk, a misconfiguration, a vulnerability, an exposure, a combination thereof, and the like.

For example, according to an embodiment, inspecting the AI model includes determining if the model is configured to run code (e.g., through a pickle file), a list of functions which are allowed, a list of functions which are blocked, and the like.

In an embodiment, inspecting an AI model includes determining if the AI model matches a version, identifier, and the like, of an AI model which is previously determined as a cybersecurity risk.

In certain embodiments, inspecting an AI model includes detecting a file, a code, and the like, which the AI model is configured to execute. In some embodiments, inspecting the AI model includes detecting a secret, a code, and the like, for example stored in a pickle file, which the AI model, when prompted with an appropriate prompt, is configured to output, leading to leaked data, leaked secrets, etc.

In some embodiments, inspection of an AI model includes generating a finding. In an embodiment, a finding is a data record indicating a detected cybersecurity risk, misconfiguration, vulnerability, exposure, combination thereof, and the like.

In an embodiment, a security database, inspection controller, and the like, is configured to generate a lateral movement path. For example, in an embodiment, where a secret is detected in an AI model, a representation of the secret is generated. In some embodiments, the representation of the secret already preexists in the security database, and a representation of the AI model is connected to the preexisting representation of the secret.

In some embodiments, a lateral movement path is generated to include resources, nodes, representations, and the like, between the AI model and another resource. For example, in an embodiment, the secret detected in the AI model is also utilized by an administrator account to access a storage in a cloud computing environment, therefore a lateral movement path includes the AI model, the secret, the administrator account, and the storage.

In certain embodiments, inspecting an AI model further includes detecting an anomaly. For example, according to an embodiment, detecting an anomaly respective of an AI model includes detecting a training set including data not used for training other models. In an embodiment, detecting an anomaly includes detecting a suspicious prompt provided to the AI model (e.g., a prompt utilizing code, attempting to execute code, attempting to extract a secret, etc.).

At S640, a toxic combination is detected. In an embodiment, a toxic combination is detected based on detecting a cybersecurity object, and detecting an AI model. For example, in an embodiment, the cybersecurity object is a secret, such as a secret utilized by the AI model.

In an embodiment, a toxic combination is detected by utilizing the representation stored in the security database.

For example, in some embodiments, utilizing the representation includes applying a policy on the representation, querying the representation, traversing a graph utilized as the representation, a combination thereof, and the like.

In certain embodiments, the security representation is utilized to apply a policy on a computing environment, the computing environment having a representation stored in the security database as part of the security representation.

For example, in an embodiment, a policy includes a rule, a conditional rule, a plurality of each, a combination thereof, and the like, which applied to the representation of the computing environment, the representation of the AI model, a representation of a cybersecurity object, a combination thereof, and the like.

In some embodiments, a policy includes a rule to detect a toxic combination. For example, in an embodiment, a policy includes a rule applied to the representation to detect a finding connected to a resource, and an AI model connected to the resource.

At **S650**, a mitigation action is initiated. In an embodiment, a mitigation action is initiated based on a toxic combination, the detected cybersecurity object, an AI model type, a combination thereof, and the like.

In some embodiments, a mitigation action includes revoking access from a principal, revoking access to a resource, revoking access from a resource, sandboxing a resource, revoking access to an AI model, revoking access from an AI model, denying network communication directed to the AI model, such as network communication including a prompt for the AI model, generating an alert, updating an alert, generating an alert severity, updating an alert severity, various combinations thereof, and the like.

In certain embodiments, an inspection environment is configured to provide a remediation infrastructure. In an embodiment, the inspection environment is configured to deploy the remediation infrastructure in the computing environment. In some embodiments, the remediation infrastructure includes a plurality of remediation scripts, mitigation actions, a combination thereof, and the like.

FIG. 7 is an example schematic diagram of an AI detector **138** according to an embodiment. The AI detector **138** includes a processing circuitry **710** coupled to a memory **720**, a storage **730**, and a network interface **740**. In an embodiment, the components of the AI detector **138** may be communicatively connected via a bus **750**.

The processing circuitry **710** may be realized as one or more hardware logic components and circuits. For example, and without limitation, illustrative types of hardware logic components that can be used include field programmable gate arrays (FPGAs), application-specific integrated circuits (ASICs), Application-specific standard products (ASSPs), system-on-a-chip systems (SOCs), graphics processing units (GPUs), tensor processing units (TPUs), general-purpose microprocessors, microcontrollers, digital signal processors (DSPs), and the like, or any other hardware logic components that can perform calculations or other manipulations of information.

The memory **720** may be volatile (e.g., random access memory, etc.), non-volatile (e.g., read only memory, flash memory, etc.), or a combination thereof. In an embodiment, the memory **720** is an on-chip memory, an off-chip memory, a combination thereof, and the like. In certain embodiments, the memory **720** is a scratch-pad memory for the processing circuitry **710**.

In one configuration, software for implementing one or more embodiments disclosed herein may be stored in the storage **730**, in the memory **720**, in a combination thereof,

and the like. Software shall be construed broadly to mean any type of instructions, whether referred to as software, firmware, middleware, microcode, hardware description language, or otherwise. Instructions may include code (e.g., in source code format, binary code format, executable code format, or any other suitable format of code). The instructions, when executed by the processing circuitry **710**, cause the processing circuitry **710** to perform the various processes described herein.

The storage **730** is a magnetic storage, an optical storage, a solid-state storage, a combination thereof, and the like, and is realized, according to an embodiment, as a flash memory, as a hard-disk drive, or other memory technology, or any other medium which can be used to store the desired information.

The network interface **740** is configured to provide the AI detector **138** with communication with, for example, the inspector **132**, the inspection controller **134**, the security database **136**, and the like.

It should be understood that the embodiments described herein are not limited to the specific architecture illustrated in FIG. 7, and other architectures may be equally used without departing from the scope of the disclosed embodiments.

Furthermore, in certain embodiments the inspector **132**, the inspection controller **134**, the security database **136**, and the like, may be implemented with the architecture illustrated in FIG. 7. In other embodiments, other architectures may be equally used without departing from the scope of the disclosed embodiments.

The various embodiments disclosed herein can be implemented as hardware, firmware, software, or any combination thereof. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage unit or computer readable medium consisting of parts, or of certain devices and/or a combination of devices. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units ("CPUs"), a memory, and input/output interfaces. The computer platform may also include an operating system and microinstruction code. The various processes and functions described herein may be either part of the microinstruction code or part of the application program, or any combination thereof, which may be executed by a CPU, whether or not such a computer or processor is explicitly shown. In addition, various other peripheral units may be connected to the computer platform such as an additional data storage unit and a printing unit. Furthermore, a non-transitory computer readable medium is any computer readable medium except for a transitory propagating signal.

All examples and conditional language recited herein are intended for pedagogical purposes to aid the reader in understanding the principles of the disclosed embodiment and the concepts contributed by the inventor to furthering the art, and are to be construed as being without limitation to such specifically recited examples and conditions. Moreover, all statements herein reciting principles, aspects, and embodiments of the disclosed embodiments, as well as specific examples thereof, are intended to encompass both structural and functional equivalents thereof. Additionally, it is intended that such equivalents include both currently known equivalents as well as equivalents developed in the future, i.e., any elements developed that perform the same function, regardless of structure.

US 12,003,529 B1

19

It should be understood that any reference to an element herein using a designation such as “first,” “second,” and so forth does not generally limit the quantity or order of those elements. Rather, these designations are generally used herein as a convenient method of distinguishing between two or more elements or instances of an element. Thus, a reference to first and second elements does not mean that only two elements may be employed there or that the first element must precede the second element in some manner. Also, unless stated otherwise, a set of elements comprises one or more elements.

As used herein, the phrase “at least one of” followed by a listing of items means that any of the listed items can be utilized individually, or any combination of two or more of the listed items can be utilized. For example, if a system is described as including “at least one of A, B, and C,” the system can include A alone; B alone; C alone; 2A; 2B; 2C; 3A; A and B in combination; B and C in combination; A and C in combination; A, B, and C in combination; 2A and C in combination; A, 3B, and 2C in combination; and the like.

What is claimed is:

1. A method for detecting a cybersecurity risk of an artificial intelligence (AI), comprising:
 - generating an inspectable disk based on an original disk of a resource deployed in a computing environment;
 - inspecting the inspectable disk for an AI model;
 - generating a representation of the AI model in a security database, the security database including a representation of the computing environment;
 - inspecting the AI model for a cybersecurity risk;
 - generating a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and
 - initiating a mitigation action based on the cybersecurity risk.
2. The method of claim 1, further comprising: detecting an artifact of the AI model on the inspectable disk.
3. The method of claim 1, further comprising: inspecting the AI model to detect an AI model configured to execute a code object.
4. The method of claim 1, further comprising: detecting a metadata of the AI model, wherein the metadata indicates that the AI model is a cybersecurity risk.
5. The method of claim 1, further comprising: detecting in the AI model any one of: a secret, a certificate, a code, and any combination thereof.
6. The method of claim 1, further comprising: generating a lateral movement path, wherein the lateral movement path includes the AI model, and a representation of a resource, wherein the resource is accessible utilizing the AI model.
7. The method of claim 1, further comprising: applying a policy on the representation of the computing environment.
8. The method of claim 7, further comprising: initiating the mitigation action based on the policy.
9. A non-transitory computer-readable medium storing a set of instructions for detecting a cybersecurity risk of an artificial intelligence (AI), the set of instructions comprising: one or more instructions that, when executed by one or more processors of a device, cause the device to:
 - generate an inspectable disk based on an original disk of a resource deployed in a computing environment;
 - inspect the inspectable disk for an AI model;

20

- generate a representation of the AI model in a security database, the security database including a representation of the computing environment;
- inspect the AI model for a cybersecurity risk;
- generate a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and
- initiate a mitigation action based on the cybersecurity risk.

10. A system for detecting a cybersecurity risk of an artificial intelligence (AI) comprising:

- a processing circuitry;
- a memory, the memory containing instructions that, when executed by the processing circuitry, configure the system to:
 - generate an inspectable disk based on an original disk of a resource deployed in a computing environment;
 - inspect the inspectable disk for an AI model;
 - generate a representation of the AI model in a security database, the security database including a representation of the computing environment;
 - inspect the AI model for a cybersecurity risk;
 - generate a representation of the cybersecurity risk in the security database, the representation of the cybersecurity risk connected to the representation of the AI model in response to detecting the cybersecurity risk; and
 - initiate a mitigation action based on the cybersecurity risk.

11. The system of claim 10, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

- detect an artifact of the AI model on the inspectable disk.
12. The system of claim 10, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:
- inspect the AI model to detect an AI model configured to execute a code object.

13. The system of claim 10, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

- detect a metadata of the AI model, wherein the metadata indicates that the AI model is a cybersecurity risk.

14. The system of claim 10, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

- detect in the AI model any one of:
 - a secret, a certificate, a code, and any combination thereof.

15. The system of claim 10, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

- generate a lateral movement path, wherein the lateral movement path includes the AI model, and a representation of a resource, wherein the resource is accessible utilizing the AI model.

16. The system of claim 10, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:

- apply a policy on the representation of the computing environment.

US 12,003,529 B1

21

22

17. The system of claim 16, wherein the memory contains further instructions which when executed by the processing circuitry further configure the system to:
initiate the mitigation action based on the policy.

* * * * *

EXHIBIT F

This copy is for your personal, non-commercial use only. Distribution and use of this material are governed by our Subscriber Agreement and by copyright law. For non-personal use or to order multiple copies, please contact Dow Jones Reprints at 1-800-843-0008 or visit www.djreprints.com.

<https://www.wsj.com/articles/microsoft-patched-bing-vulnerability-that-allowed-snooping-on-email-and-other-data-25b58831>

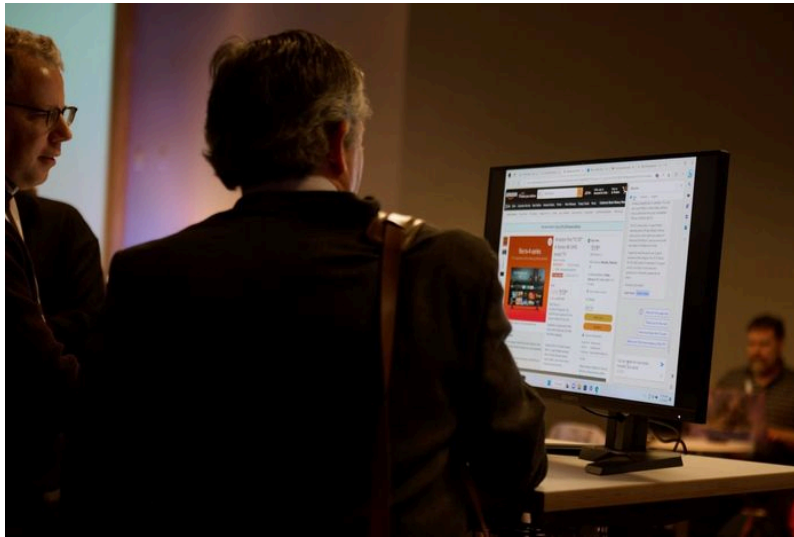
EXCLUSIVE TECH

Microsoft Patched Bing Vulnerability That Allowed Snooping on Email and Other Data

The issue was fixed days before the software company launched Bing with AI

By [Robert McMillan](#) [Follow](#)

Updated March 29, 2023 4:34 pm ET



Microsoft has been adding generative AI capabilities to much of its software and services. PHOTO: CHONA KASINGER/BLOOMBERG NEWS

Microsoft Corp. **MSFT -0.25%** ▼ patched a dangerous security issue in Bing last month, days before it launched a new artificial intelligence-powered version of the search engine.

The problem was discovered by outside researchers at the security firm Wiz Inc. It was created by a mistake in the way that Microsoft configured applications on Azure, its cloud-computing platform, and could be exploited to gain access to emails and other documents of people who used Bing, the researchers said.

Microsoft fixed the problem on Feb. 2, according to Ami Luttwak, Wiz's chief technology officer. Five days later Satya Nadella introduced the new generative-AI capabilities to Bing, bringing a renewed interest in Microsoft's 14-year-old search engine. Usage of Bing has jumped, rising to more than 100 million daily active users in the month since the upgrade, Microsoft said in a recent blog.

Microsoft has been adding generative-AI capabilities to much of its software and services. The new Bing can help users track down information using a chatbot backed by the technology behind ChatGPT.

Microsoft is adding the technology to its popular Microsoft 365 suite of business software. This week it unveiled plans to use AI to help cybersecurity experts monitor and categorize threats and attacks.

A Microsoft spokesman said the misconfiguration issue affected a small number of the company's applications that used its login management service, called Azure Active Directory.

"We appreciate the collaboration with Wiz, which helped us mitigate a potential risk and further harden our services and thank them for working with us to protect the ecosystem," the company said.

In a blog post, Microsoft said the issues pointed out by Wiz had been fixed and outlined ways for companies and consumers to protect themselves. The company said it had taken steps to prevent this type of issue from occurring in the future.

Microsoft shares closed nearly 2% higher Wednesday, in line with the rise of the Nasdaq Composite Index.

Wiz said there is no evidence anyone has taken advantage of the issue. It isn't clear how long it was available for hackers to use, although the issue may have been exploitable for years, the cybersecurity company said.

Hillai Ben-Sasson, a researcher at Wiz, said the misconfiguration allowed him to access a website used by Microsoft employees to set up trivia quizzes on Bing. Because it was misconfigured, anyone with a free Microsoft account could use it to change what results popped up on Bing for search queries.

It should only have been viewable to Microsoft employees, Wiz's Mr. Luttwak said. "We should have never seen it," he said.

The Wiz team discovered they could change some Bing search results by changing data on the Bing trivia page. They were able to make specific results show up for any search query by tinkering with the trivia page. They made the 1995 film "Hackers" pop up for anyone who searched for the term "best soundtracks."

Then they discovered something more serious: a way to get access to Bing users' Microsoft 365 emails, documents, calendars and other data.

This kind of access would be extremely valuable to hackers who could use it to steal sensitive information, send fraudulent emails and gain access to computer systems.

"A potential attacker could have influenced Bing search results and compromised Microsoft 365 emails and data of millions of people," Mr. Luttwak said. "It could have been a nation-state trying to influence public opinion or a financially motivated hacker."

In addition to the trivia site, Wiz researchers found about 1,000 other websites on Microsoft's cloud that appeared to have similar problems. Most of the pages looked like they belonged to Azure customers but at least 10 of them were Microsoft's.

Microsoft has emerged as one of the world's largest cybersecurity companies. It has also been plagued by security issues recently as it tries to lock down both its legacy products, which run on personal computers and in corporate data centers while integrating them with its fast-growing cloud computing platform.

Write to Robert McMillan at robert.mcmillan@wsj.com

Appeared in the March 30, 2023, print edition as 'Microsoft Patched a Bing Security Issue'.

Buy Side from WSJ

Expert recommendations on products and services, independent from The Wall Street Journal newsroom.



HOME

Upgrade Your Kitchen With These Nonstick Pans



ELECTRONICS

The Best Basic Printers for Your Home Office



GIFTS

23 Best Gift Ideas for a Dad Who Has Everything



WELLNESS

How to Choose an LED Therapy Mask, According to Dermatologists



WELLNESS

The 22 Best Mineral Sunscreens Derms Want You to Know About



GIFTS

17 Best Gifts for a First-Time Dad