JOINT
CYBERSECURITY
ADVISORY

TLP:CLEAR

Co-Authored by:

Product ID: AA24-191A

July 9, 2024

POLITIE

Communications Security Establishment
Canadian Centre for Cyber Security

Centre de la sécurité des télécommunications
Centre canadien pour la cybersécurité

State-Sponsored Russian Media Leverages Meliorator Software for Foreign Malign Influence Activity

Summary

The U.S. Federal Bureau of Investigation (FBI) and Cyber National Mission Force (CNMF), in partnership with the Netherlands General Intelligence and Security Service (AIVD), Netherlands Military Intelligence and Security Service (MIVD), the Netherlands Police (DNP), and the Canadian Centre for Cyber Security (CCCS), (hereinafter referred to as the authoring organizations) are releasing this advisory to warn social media companies that Russian state-sponsored actors have leveraged the covert Meliorator software for foreign malign influence activity benefiting the Russian Government.

Affiliates of RT (formerly Russia Today), a Russian state-sponsored media organization, used Meliorator—a covert artificial intelligence (AI) enhanced software package—to create fictitious online personas, representing a number of nationalities, to post content on X (formerly Twitter). Using this tool, RT affiliates disseminated disinformation to and about a number of countries, including the United States, Poland, Germany, the Netherlands, Spain, Ukraine, and Israel.

Although the tool was only identified on X, the authoring organizations' analysis of Meliorator indicated the developers intended to expand its functionality to other social media platforms. The authoring organizations' analysis also indicated the tool is capable of the following:

- Creating authentic appearing social media personas en masse;
- Deploying content similar to typical social media users;
- Mirroring disinformation of other bot personas;
- Perpetuating the use of pre-existing false narratives to amplify malign foreign influence; and
- Formulating messages, to include the topic and framing, based on the specific archetype of the bot.

To report suspicious or criminal activity related to information found in this joint Cybersecurity Advisory, contact your local FBI field office. When available, please include the following information regarding the incident: date, time, and location of the incident; type of activity; number of people affected; type of equipment used for the activity; the name of the submitting company or organization; and a designated point of contact.

The authoring organizations encourage social media companies to leverage the information in this advisory to assist with identifying fictitious personas to reduce Russian malign foreign influence activity.

For additional information, see U.S. Department of Justice (DOJ) press release Justice Department and International and Private Sector Partners Disrupt Covert Russian Government-Operated Social Media Bot Farm. For more information on Russia state-sponsored malicious cyber activity, see the Russia Cyber Threat Overview and Advisories webpage.

Technical Details

Meliorator

As early as 2022, RT had access to Meliorator, an AI-enabled bot farm generation and management software to disseminate disinformation to and about a number of countries, including the United States, Poland, Germany, the Netherlands, Spain, Ukraine, and Israel. Meliorator was designed to be used on social media networks to create "authentic" appearing personas en masse, allowing for the propagation of disinformation, which could assist Russia in exacerbating discord and trying to alter public opinion as part of information operations. As of June 2024, Meliorator only worked on X (formerly known as Twitter). However, additional analysis suggests the software's functionality would likely be expanded to other social media networks.

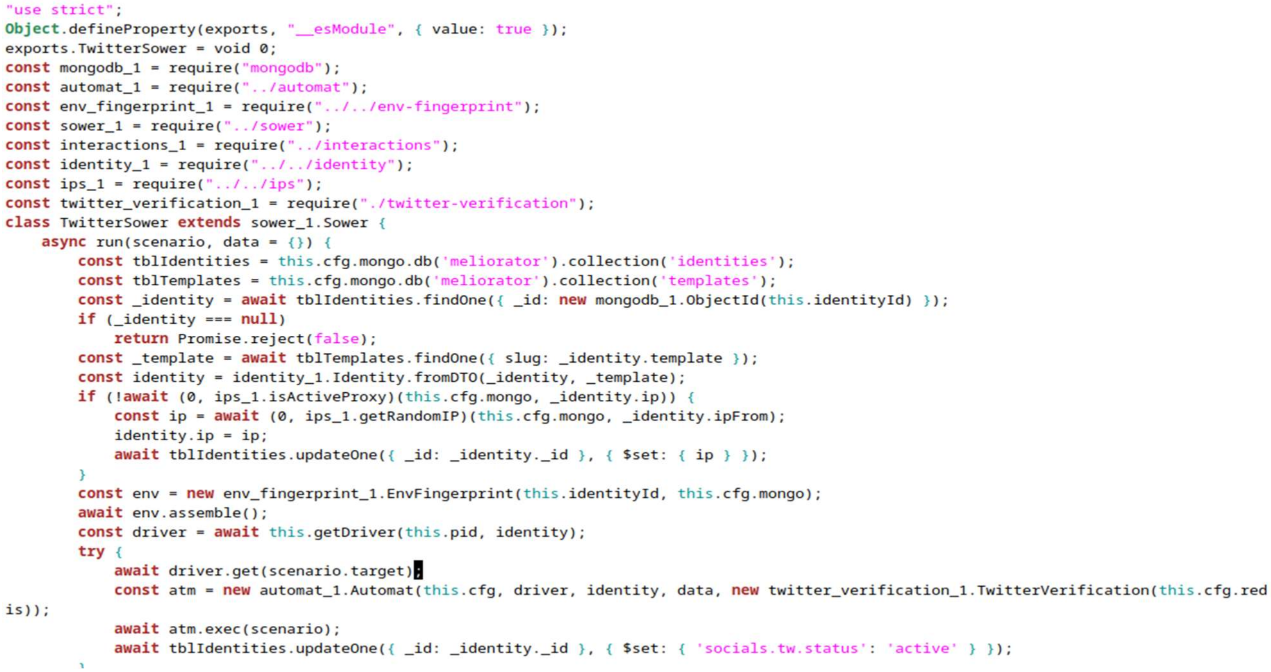
To provide this functionality, Meliorator includes an administrator panel called "Brigadir" and a seeding tool called "Taras." In order to access Meliorator, users would connect by means of a virtual network computing (VNC) connection. Using Redmine software (which supports 49 languages, is multi-platform, and can be used cross-database) for project management, developers hosted Meliorator at `dtxt.mlrtr[.]com`.

Brigadir

Brigadir serves as the primary end user interface of Meliorator and functions as the administrator panel. Brigadir serves as the graphical user interface for the Taras application and includes tabs for "souls," false identities that would create the basis for the bots, and "thoughts," which are the automated scenarios or actions that could be implemented on behalf of the bots, such as sharing content to social media in the future.

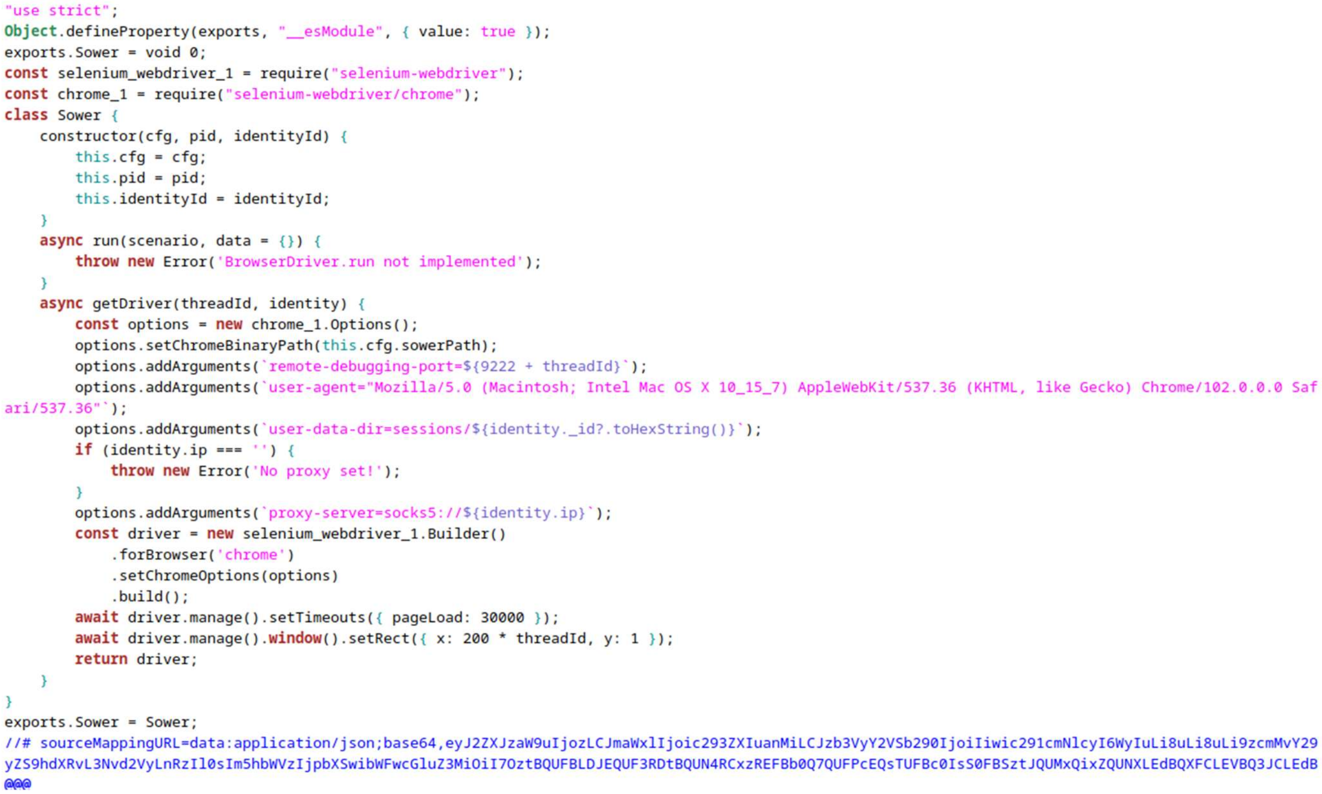
Taras

"Taras" serves as the back end of the Meliorator software package containing `.json` files used to control the personas sowing disinformation on social media. These files are highly decentralized code, which need to be combined with other files upon execution in order to achieve the desired functionality. Two specific files are vital to the functionality of Taras. The first file is designed to aggregate a number of other tools and databases for their use (Figure 1). The second (Figure 2) is designed to aggregate and execute a number of automation tools used by Meliorator.



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.TwitterSower = void 0;
const mongodb_1 = require("mongodb");
const automat_1 = require("../automat");
const env_fingerprint_1 = require("../../env-fingerprint");
const sower_1 = require("../sower");
const interactions_1 = require("../interactions");
const identity_1 = require("../../identity");
const ips_1 = require("../../ips");
const twitter_verification_1 = require("./twitter-verification");
class TwitterSower extends sower_1.Sower {
    async run(scenario, data = {}) {
        const tblIdentities = this.cfg.mongo.db('meliorator').collection('identities');
        const tblTemplates = this.cfg.mongo.db('meliorator').collection('templates');
        const _identity = await tblIdentities.findOne({ _id: new mongodb_1.ObjectId(this.identityId) });
        if (_identity === null)
            return Promise.reject(false);
        const _template = await tblTemplates.findOne({ slug: _identity.template });
        const identity = identity_1.Identity.fromDTO(_identity, _template);
        if (!await (0, ips_1.isActiveProxy)(this.cfg.mongo, _identity.ip)) {
            const ip = await (0, ips_1.getRandomIP)(this.cfg.mongo, _identity.ipFrom);
            identity.ip = ip;
            await tblIdentities.updateOne({ _id: _identity._id }, { $set: { ip } });
        }
        const env = new env_fingerprint_1.EnvFingerprint(this.identityId, this.cfg.mongo);
        await env.assemble();
        const driver = await this.getDriver(this.pid, identity);
        try {
            await driver.get(scenario.target);
            const atm = new automat_1.Automat(this.cfg, driver, identity, data, new twitter_verification_1.TwitterVerification(this.cfg.red
is));
            await atm.exec(scenario);
            await tblIdentities.updateOne({ _id: _identity._id }, { $set: { 'socials.tw.status': 'active' } });
        }
```

Figure 1: Truncated Snippet from a File Aggregator Tool Used to Deploy Databases



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.Sower = void 0;
const selenium_webdriver_1 = require("selenium-webdriver");
const chrome_1 = require("selenium-webdriver/chrome");
class Sower {
    constructor(cfg, pid, identityId) {
        this.cfg = cfg;
        this.pid = pid;
        this.identityId = identityId;
    }
    async run(scenario, data = {}) {
        throw new Error('BrowserDriver.run not implemented');
    }
    async getDriver(threadId, identity) {
        const options = new chrome_1.Options();
        options.setChromeBinaryPath(this.cfg.sowerPath);
        options.addArguments(`remote-debugging-port=${9222 + threadId}`);
        options.addArguments(`user-agent="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Saf
ari/537.36"`);
        options.addArguments(`user-data-dir=sessions/${identity._id?.toHexString()}`);
        if (identity.ip === '') {
            throw new Error('No proxy set!');
        }
        options.addArguments(`proxy-server=socks5://${identity.ip}`);
        const driver = new selenium_webdriver_1.Builder()
            .forBrowser('chrome')
            .setChromeOptions(options)
            .build();
        await driver.manage().setTimeouts({ pageLoad: 30000 });
        await driver.manage().window().setRect({ x: 200 * threadId, y: 1 });
        return driver;
    }
}
exports.Sower = Sower;
//# sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJmaWxlIjoic293ZXIuanMiLCJzb3VyY2VSb290IjoiIiwic291cmNlcyI6WyIuLi8uLi8uLi9zcmMvY29
yZS9hdXRvL3Nvd2VyLnRzIl0sIm5hbWVzIjpbXSwibWFwcGluZ3MiOiI7OztBQUFBLDJEQUF3RDtBQUN4RCxzREFBb0Q7QUFPcEEsTUFBc0IsS0FBSztJQUMxQixZQUNLLEdBQXFCCLEVBQ3JCLEdB
@@@
```

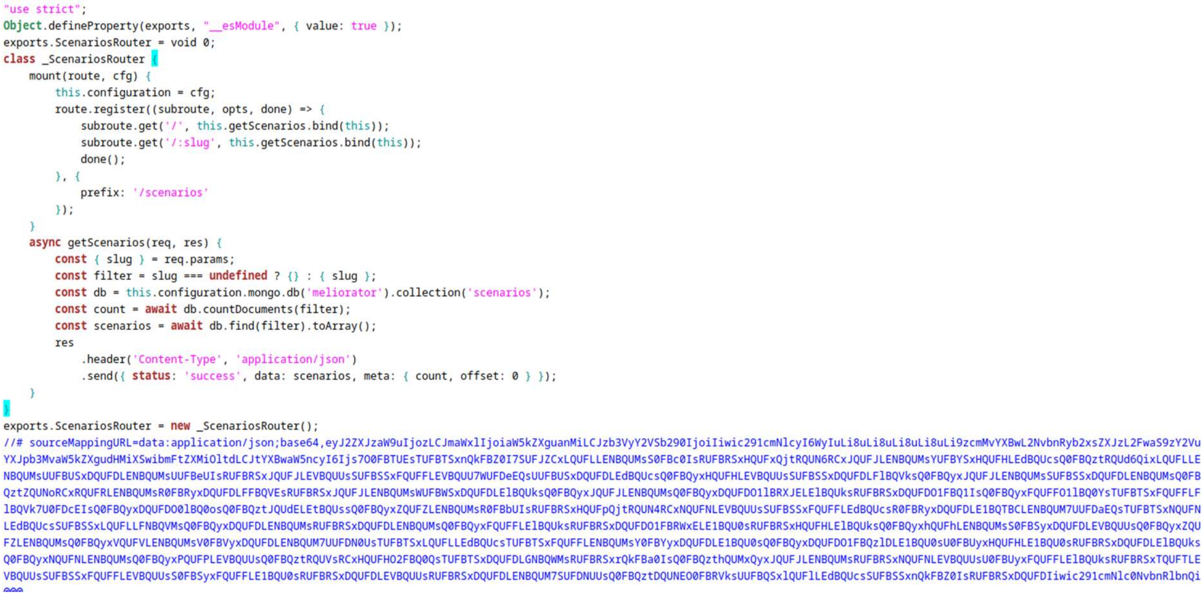
Figure 2: Importation of Other Tools Used in the Automation Process of Meliorator

Souls

The identities or so-called “souls” of these bots are determined based on the selection of specific parameters or archetypes selected by the user. Any field not preselected would be auto-generated. Bot archetypes are then created to group ideologically aligned bots using a specifically crafted algorithm to construct each bot's persona, determining the location, political ideologies, and even biographical data of the persona. These details are automatically filled in based on the selection of the souls’ archetype. Once Taras creates the identity, it is registered on the social media platform. The identities are stored using a MongoDB, which can allow for ad hoc queries, indexing, load-balancing, aggregation, and server-side JavaScript execution.

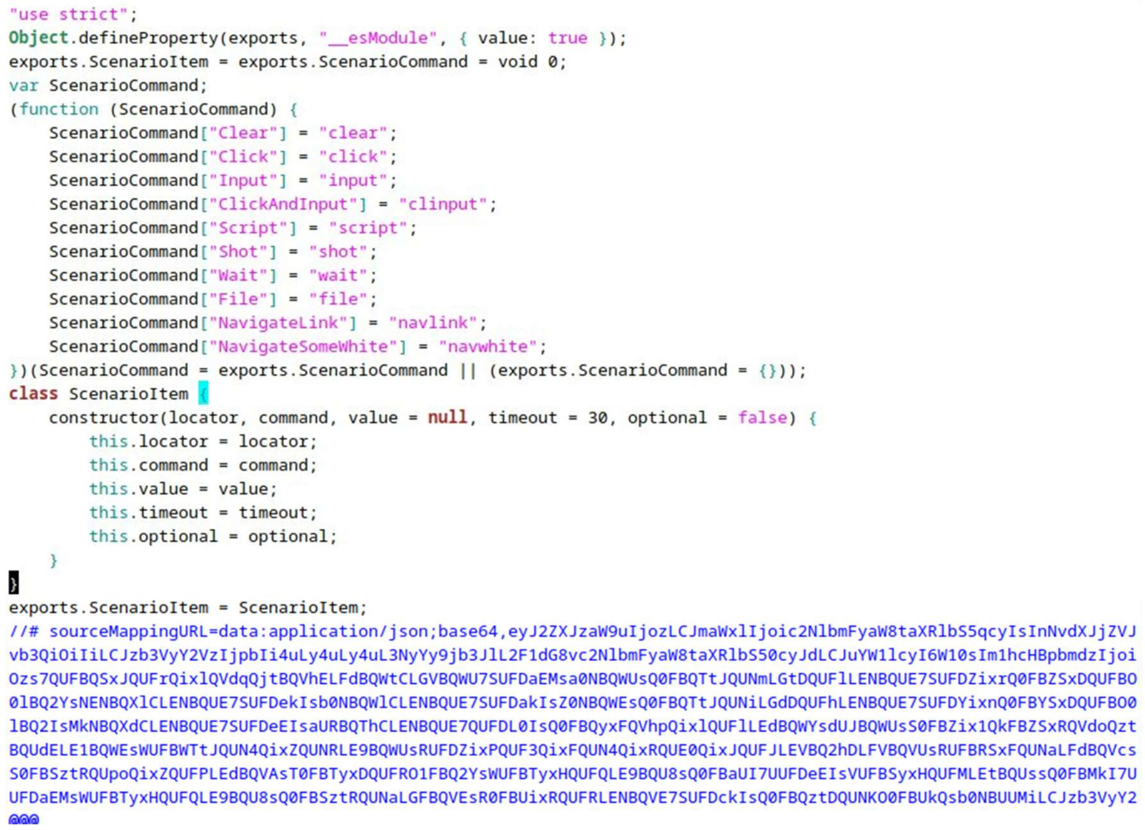
Thoughts

The “thoughts” tab contains automated scenarios or actions which can be completed on behalf of a soul or a group of souls. This allows the personas to like, share, repost, and comment on others’ posts with videos or links. The thoughts tab also allowed for maintenance, creating a new registration for an identity, and logging into already existing profiles. The framework for the thoughts tab and the scenarios it creates can be seen in the code; these files are written separately and a parent file calls to smaller files in order to fulfill the function. Figure 3 is an aggregator file to trigger the functionality for scenarios. Of specific interest, it calls the thoughts from the MongoDB and calls for the files which provide the GUI its functionality making it more user friendly (Figure 4). Notably, the same code contains references to other social media platforms, to include Facebook and Instagram, indicating an intent to expand the project beyond X, as seen in Figure 5.



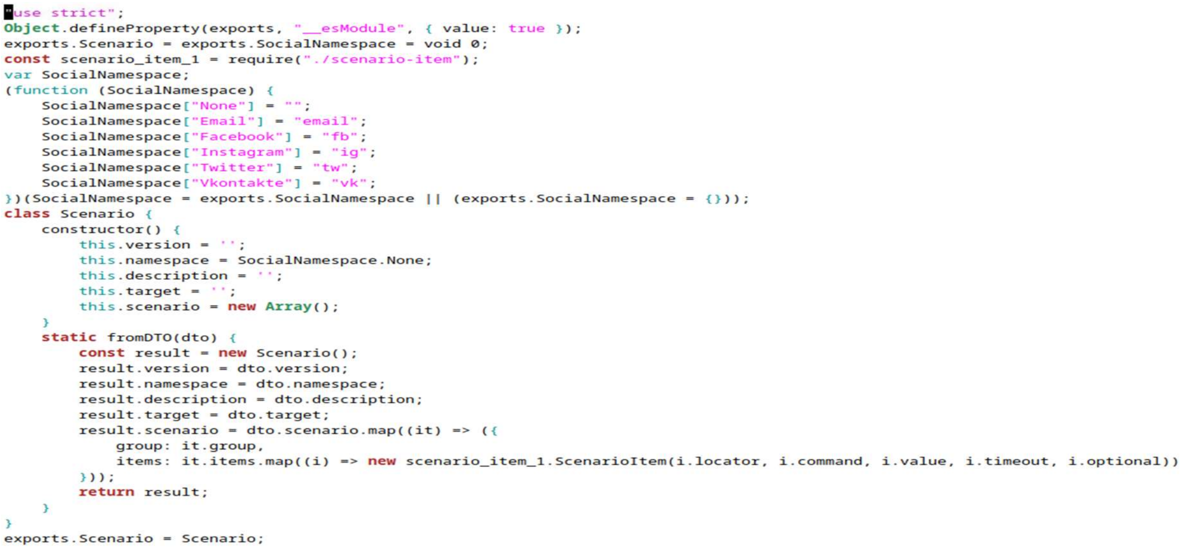
```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.ScenariosRouter = void 0;
class _ScenariosRouter {
    mount(route, cfg) {
        this.configuration = cfg;
        route.register((subroute, opts, done) => {
            subroute.get('/', this.getScenarios.bind(this));
            subroute.get('/:slug', this.getScenarios.bind(this));
            done();
        }, {
            prefix: '/scenarios'
        });
    }
    async getScenarios(req, res) {
        const { slug } = req.params;
        const filter = slug === undefined ? {} : { slug };
        const db = this.configuration.mongo.db('meliorator').collection('scenarios');
        const count = await db.countDocuments(filter);
        const scenarios = await db.find(filter).toArray();
        res
            .header('Content-Type', 'application/json')
            .send({ status: 'success', data: scenarios, meta: { count, offset: 0 } });
    }
}
exports.ScenariosRouter = new _ScenariosRouter();
//# sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJmaWxlIjoiaW5kZXguanMiLCJzb3VyY2VSb290IjoiIiwic291cmNlcyI6WyIuLi8uLi8uLi8uLi8uLi9zcmMvYXBwL2NvbnRyb2xsZXJzL2FwaS9zY2Vu
YXJpb3MvaW5kZXgudHMiXSwibmFtZXMiOltdLCJtYXBwaW5ncyI6Ijs7O0FBTUEsTUFBTSxnQkFBZ0I7SUFJZCxLQUFLLENBQUMsS0FBc0IsRUFBRSxHQUFxQjtRQUN6RCxJQUFJLENBQUMsYUFBYSxHQUFHLEdBQUcsQ0FBQztRQUd6QixLQUFLLE
NBQUMsUUFBUSxDQUFDLENBQUMsUUFBeUIsRUFBRSxJQUFJLEVBQUUsSUFBSSxFQUFFLEVBQUU7WUFDeEQsUUFBUSxDQUFDLEdBQUcsQ0FBQyxHQUFHLEVBQUUsSUFBSSxDQUFDLFlBQVksQ0FBQyxJQUFJLENBQUMsSUFBSSxDQUFDLENBQUMsQ0FB
QztZQUNoRCxRQUFRLENBQUMsR0FBRyxDQUFDLFFBQVEsRUFBRSxJQUFJLENBQUMsWUFBWSxDQUFDLElBQUksQ0FBQyxJQUFJLENBQUMsQ0FBQyxDQUFDO1lBRXJELElBQUksRUFBRSxDQUFDO1FBQ1IsQ0FBQyxFQUFFO1lBQ0YsTUFBTSxFQUFFLF
lBQVk7U0FDcEIsQ0FBQyxDQUFDO0lBQ0osQ0FBQztJQUdELEtBQUssQ0FBQyxZQUFZLENBQUMsR0FBbUIsRUFBRSxHQUFpQjtRQUN4RCxNQUFNLEVBQUUsSUFBSSxFQUFFLEdBQUcsR0FBRyxDQUFDLE1BQTBCLENBQUM7UUFDaEQsTUFBTSxNQUFN
LEdBQUcsSUFBSSxLQUFLLFNBQVMsQ0FBQyxDQUFDLENBQUMsRUFBRSxDQUFDLENBQUMsQ0FBQyxFQUFFLElBQUksRUFBRSxDQUFDO1FBRWxELE1BQU0sRUFBRSxHQUFHLElBQUksQ0FBQyxhQUFhLENBQUMsS0FBSyxDQUFDLEVBQUUsQ0FBQyxZQU
FZLENBQUMsQ0FBQyxVQUFVLENBQUMsV0FBVyxDQUFDLENBQUM7UUFDN0UsTUFBTSxLQUFLLEdBQUcsTUFBTSxFQUFFLENBQUMsY0FBYyxDQUFDLE1BQU0sQ0FBQyxDQUFDO1FBQzlDLE1BQU0sU0FBUyxHQUFHLE1BQU0sRUFBRSxDQUFDLElBQUks
Q0FBQyxNQUFNLENBQUMsQ0FBQyxPQUFPLEVBQUUsQ0FBQztRQUVsRCxHQUFHO2FBQ0QsTUFBTSxDQUFDLGNBQWMsRUFBRSxrQkFBa0IsQ0FBQzthQUMxQyxJQUFJLENBQUMsRUFBRSxNQUFNLEVBQUUsU0FBUyxFQUFFLElBQUksRUFBRSxTQUFTLE
VBQUUsSUFBSSxFQUFFLEVBQUUsS0FBSyxFQUFFLE1BQU0sRUFBRSxDQUFDLEVBQUUsRUFBRSxDQUFDLENBQUM7SUFDNUUsQ0FBQztDQUNEO0FBRVksUUFBQSxlQUFlLEdBQUcsSUFBSSxnQkFBZ0IsRUFBRSxDQUFDIiwic291cmNlc0NvbnRlbnQi
```

Figure 3: Indexed Aggregator File to Deploy Scenarios



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.ScenarioItem = exports.ScenarioCommand = void 0;
var ScenarioCommand;
(function (ScenarioCommand) {
    ScenarioCommand["Clear"] = "clear";
    ScenarioCommand["Click"] = "click";
    ScenarioCommand["Input"] = "input";
    ScenarioCommand["ClickAndInput"] = "clinput";
    ScenarioCommand["Script"] = "script";
    ScenarioCommand["Shot"] = "shot";
    ScenarioCommand["Wait"] = "wait";
    ScenarioCommand["File"] = "file";
    ScenarioCommand["NavigateLink"] = "navlink";
    ScenarioCommand["NavigateSomeWhite"] = "navwhite";
})(ScenarioCommand = exports.ScenarioCommand || (exports.ScenarioCommand = {}));
class ScenarioItem {
    constructor(locator, command, value = null, timeout = 30, optional = false) {
        this.locator = locator;
        this.command = command;
        this.value = value;
        this.timeout = timeout;
        this.optional = optional;
    }
}
exports.ScenarioItem = ScenarioItem;
//# sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJmaWxlIjoic2NlbmFyaW8taXRlbS5qcyIsInNvdXJjZVJ
vb3QiOiIiLCJzb3VyY2VzIjpbIi4uLy4uLy4uL3NyYy9jb3JlL2F1dG8vc2NlbmFyaW8taXRlbS50cyJdLCJuYW1lcyI6W10sIm1hcHBpbmdzIjoi
Ozs7QUFBQSxJQUFrQixlQVdqQjtBQVhELFdBQWtCLGVBQWU7SUFDaEMsa0NBQWUsQ0FBQTtJQUNmLGtDQUFlLENBQUE7SUFDZixrQ0FBZSxDQUFBO
0lBQ2YsNENBQXlCLENBQUE7SUFDekIsb0NBQWlCLENBQUE7SUFDakIsZ0NBQWEsQ0FBQTtJQUNiLGdDQUFhLENBQUE7SUFDYixnQ0FBYSxDQUFBO
lBQ2IsMkNBQXdCLENBQUE7SUFDeEIsaURBQThCLENBQUE7QUFDL0IsQ0FBQyxxFQVhpQixlQUFlLEdBQWYsdUJBQWUsS0FBZix1QkFBZSxRQVdoQzt
BQUdELE1BQWEsWUFBWTtJQUN4QixZQUNRLE9BQWUsRUFDZixPQUF3QixFQUN4QixRQUE0QixJQUFJLEVBQ2hDLFVBQVUsRUFBRSxFQUNaLFdBQVcs
S0FBSztRQUpoQixZQUFPLEdBQVAsT0FBTyxxDQUFRO1FBQ2YsWUFBTyxHQUFQLE9BQU8sQ0FBaUI7UUFDeEIsVUFBSyxHQUFMLEtBQUssQ0FBMkI7UUFDaEMsWUFBTyxHQUFQ
UFDaEMsWUFBTyxHQUFQLE9BQU8sQ0FBSztRQUNaLGFBQVEsR0FBUixRQUFRLENBQVE7SUFDckIsQ0FBQztDQUNKO0FBUkQsb0NBQUMiLCJzb3VyY2
@@@
```

Figure 4: Enhancing User Functionality of the Taras GUI



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.Scenario = exports.SocialNamespace = void 0;
const scenario_item_1 = require("./scenario-item");
var SocialNamespace;
(function (SocialNamespace) {
    SocialNamespace["None"] = "";
    SocialNamespace["Email"] = "email";
    SocialNamespace["Facebook"] = "fb";
    SocialNamespace["Instagram"] = "ig";
    SocialNamespace["Twitter"] = "tw";
    SocialNamespace["Vkontakte"] = "vk";
})(SocialNamespace = exports.SocialNamespace || (exports.SocialNamespace = {}));
class Scenario {
    constructor() {
        this.version = '';
        this.namespace = SocialNamespace.None;
        this.description = '';
        this.target = '';
        this.scenario = new Array();
    }
    static fromDTO(dto) {
        const result = new Scenario();
        result.version = dto.version;
        result.namespace = dto.namespace;
        result.description = dto.description;
        result.target = dto.target;
        result.scenario = dto.scenario.map((it) => ({
            group: it.group,
            items: it.items.map((i) => new scenario_item_1.ScenarioItem(i.locator, i.command, i.value, i.timeout, i.optional))
        }));
        return result;
    }
}
exports.Scenario = Scenario;
```

Figure 5: Truncated Coding File Depicting Evidence of Planned Expansion beyond the X Platform

Logging In

Operators of Taras use the “thoughts” tab to log in to already existing bot farm personas. Once a “soul” is live on the social media platform, the identity card for the persona presents a login screen for the social media platform.

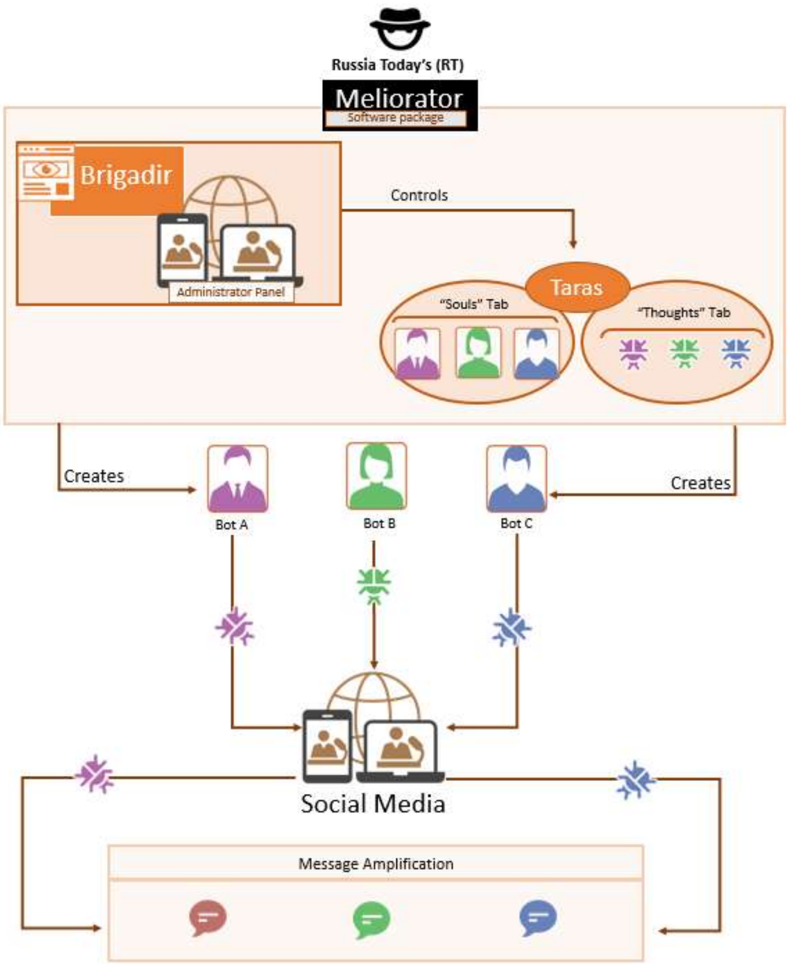
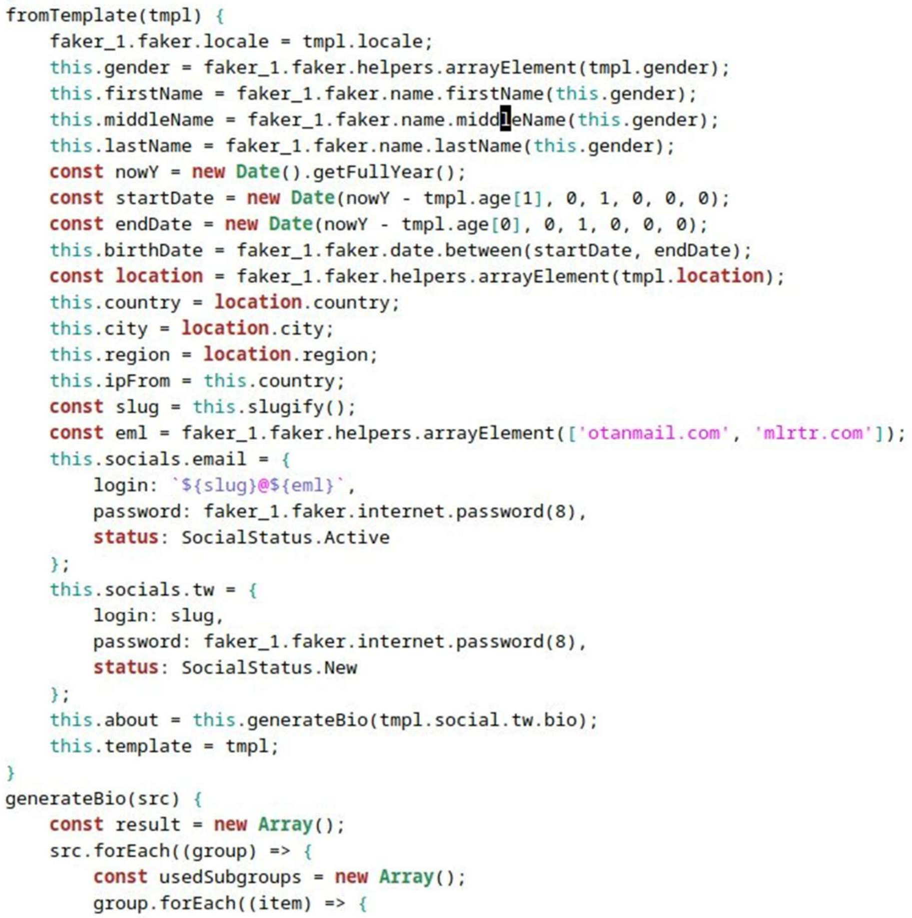


Figure 6: Technical Details Diagram

Bot Characteristics, Capabilities, and Sophistication

Characteristics

To avoid detection in the course of their online activity, each bot account is created with one of three different functions in mind. Using the Souls tab, the persona is generated for specific archetypes which then stay with the bot throughout its lifespan. The first bot archetype gets complete profiles consisting of a profile photo, cover photo, and biographical data, including name and location. These bots also have small biographies indicating their political leanings or ideologies. If a bot has this information, they will be used heavily to propagate information and will conduct the most robust activity. Profile photos for the bots were generated using AI technologies. In these instances, the tool used an open source available tool called Faker to generate photos, biographical information, and other details. (See Figure 7 for the code used in the tool related to Faker). A second bot archetype contains very little information on its profile, if any. Usually, the profile consists of a user name and very little original content, and is used to "like" already shared information. The final bot archetype was created using data compiled by a webcrawler associated with the Nemezida (variant nemez1da) website or by other data repositories to create an authentic appearing persona with no AI-ties. This bot appears real by generating a lot of activity and garnering followers. Of all the bot archetypes, this bot persona appears the most legitimate and is used to mirror and amplify disinformation shared by bot and non-bot accounts.



```
fromTemplate(tmpl) {
    faker_1.faker.locale = tmpl.locale;
    this.gender = faker_1.faker.helpers.arrayElement(tmpl.gender);
    this.firstName = faker_1.faker.name.firstName(this.gender);
    this.middleName = faker_1.faker.name.middleName(this.gender);
    this.lastName = faker_1.faker.name.lastName(this.gender);
    const nowY = new Date().getFullYear();
    const startDate = new Date(nowY - tmpl.age[1], 0, 1, 0, 0, 0);
    const endDate = new Date(nowY - tmpl.age[0], 0, 1, 0, 0, 0);
    this.birthDate = faker_1.faker.date.between(startDate, endDate);
    const location = faker_1.faker.helpers.arrayElement(tmpl.location);
    this.country = location.country;
    this.city = location.city;
    this.region = location.region;
    this.ipFrom = this.country;
    const slug = this.slugify();
    const eml = faker_1.faker.helpers.arrayElement(['otanmail.com', 'mlrtr.com']);
    this.socials.email = {
        login: `${slug}@${eml}`,
        password: faker_1.faker.internet.password(8),
        status: SocialStatus.Active
    };
    this.socials.tw = {
        login: slug,
        password: faker_1.faker.internet.password(8),
        status: SocialStatus.New
    };
    this.about = this.generateBio(tmpl.social.tw.bio);
    this.template = tmpl;
}
generateBio(src) {
    const result = new Array();
    src.forEach((group) => {
        const usedSubgroups = new Array();
        group.forEach((item) => {
```

Figure 7: Truncated json language incorporating Publicly Available Faker API to Create Personas

Sophistication

Bot persona accounts make obvious attempts to avoid bans for terms of service violations and avoid being noticed as bots by blending into the larger social media environment. The majority of accounts being followed by the bot personas boasted more than 100,000 followers, which would be necessary for a bot persona to avoid detection when interacting with other accounts. Additionally, much like authentic accounts, these bots follow genuine accounts reflective of their political leanings and interests listed in their biography. Exceptions to the 100,000 follower rule included following the accounts of other bots and/or highly-publicized accounts which would make sense for an individual interested in US politics to follow, such as well-known politicians. The tool is capable of receiving and replying to direct messages but generally tries to avoid doing so in order to limit the need to respond in real time.

Capabilities

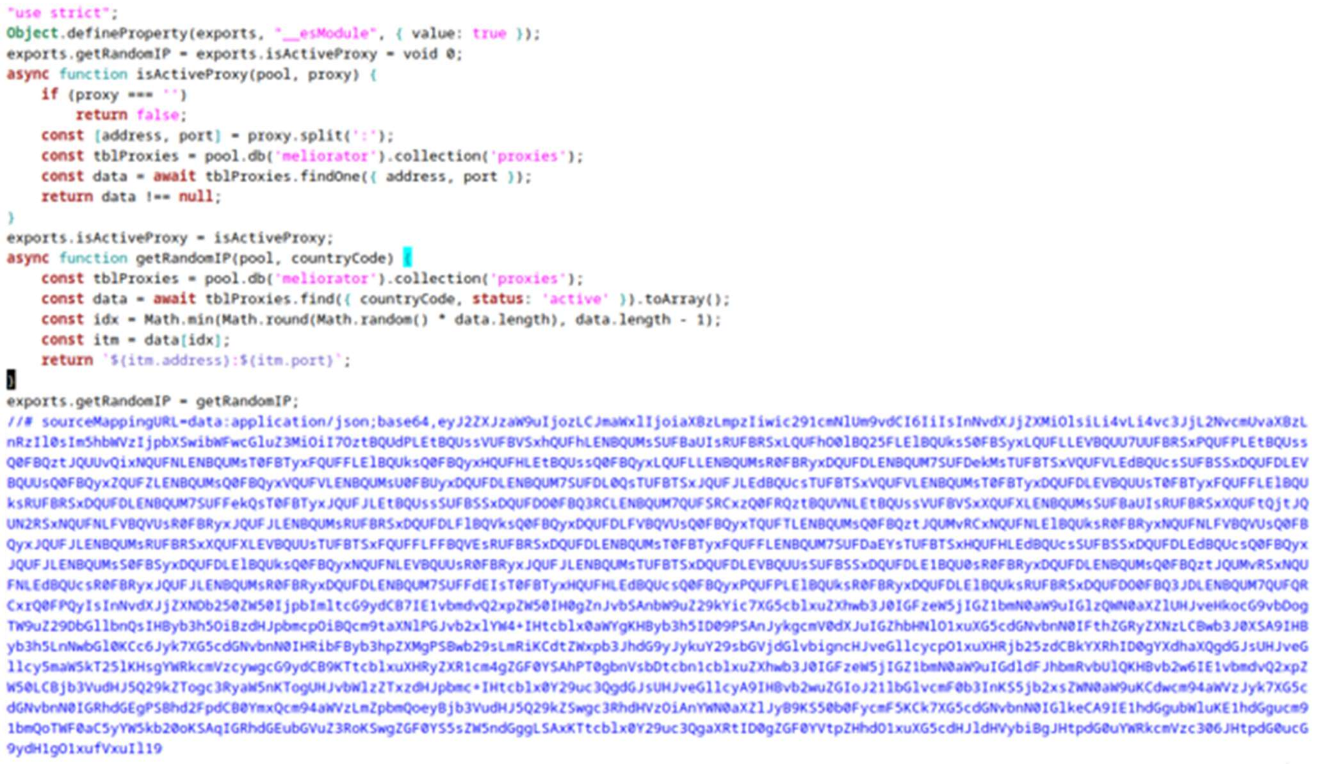
The identified bot personas associated with the Meliorator tool are capable of the following:

- Deploying content similar to typical social media users, such as generating original posts, following other users, “liking,” commenting, reposting, and obtaining followers;
- Mirroring disinformation of other bot personas through their messaging, replies, reposts, and biographies;
- Perpetuating the use of pre-existing false narratives to amplify Russian disinformation; and
- Formulating messaging, to include the topic and framing, based on the specific archetype of the bot.

Obfuscation Techniques

The creators of the Meliorator tool considered a number of barriers to detection and attempted to mitigate those barriers by coding within the tool the ability to obfuscate their IP, bypass dual factor authentication, and change the user agent string.

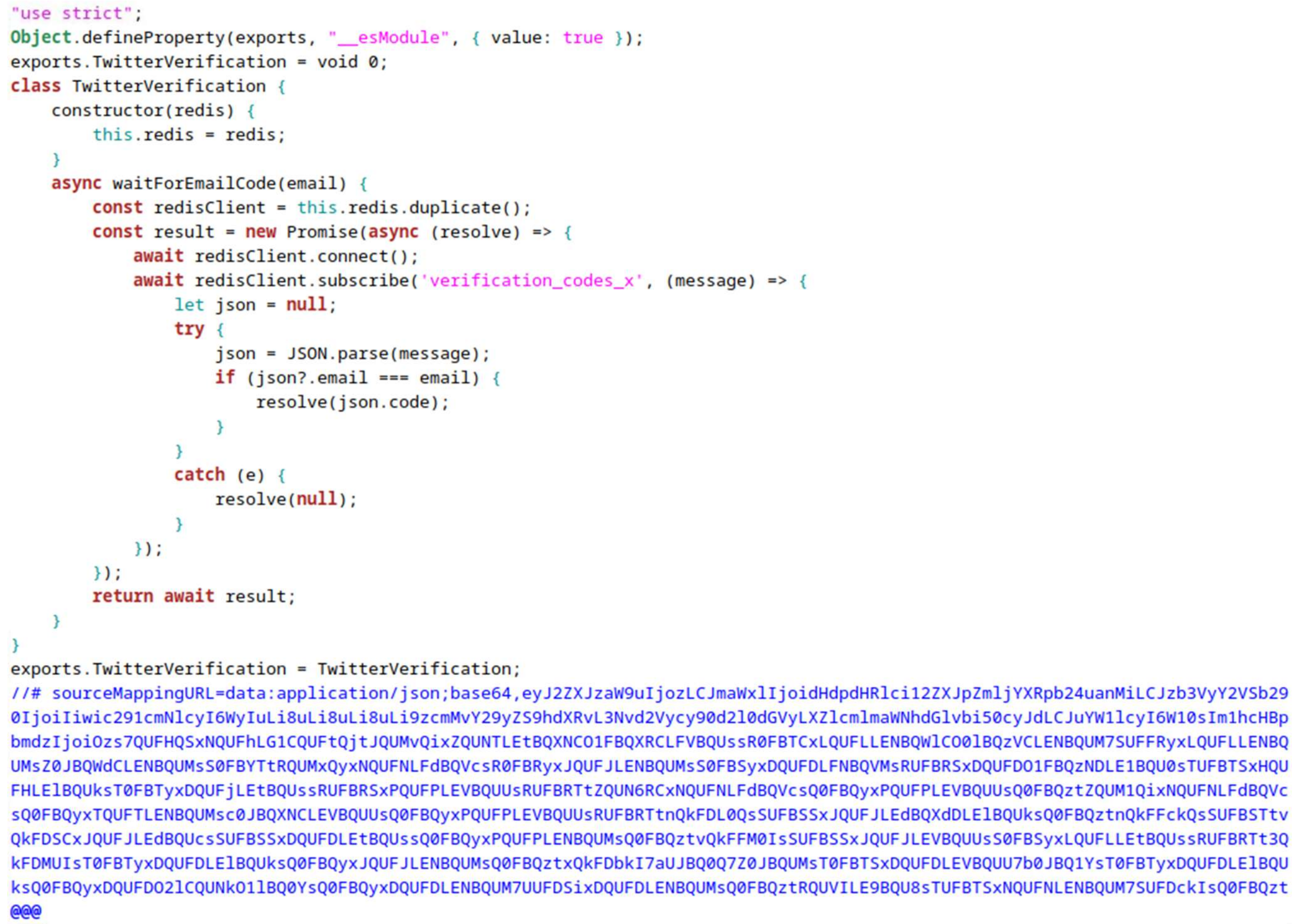
Operators avoid detection by using a backend code designed to auto-assign a proxy IP address to the AI generated persona based on their assumed location. The developer wrote a portion of code to check and see if a proxy and specific port is located in a MongoDB specified in the same code. If not, it then finds an open active IP given a country code value. See Figure 8.



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.getRandomIP = exports.isActiveProxy = void 0;
async function isActiveProxy(pool, proxy) {
    if (proxy === '')
        return false;
    const [address, port] = proxy.split(':');
    const tblProxies = pool.db('meliorator').collection('proxies');
    const data = await tblProxies.findOne({ address, port });
    return data !== null;
}
exports.isActiveProxy = isActiveProxy;
async function getRandomIP(pool, countryCode) {
    const tblProxies = pool.db('meliorator').collection('proxies');
    const data = await tblProxies.find({ countryCode, status: 'active' }).toArray();
    const idx = Math.min(Math.round(Math.random() * data.length), data.length - 1);
    const itm = data[idx];
    return `${itm.address}:${itm.port}`;
}
exports.getRandomIP = getRandomIP;
//# sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJmaWxlIjoiaXBzLmpzIiwic291cmNlUm9vdCI6IiIsInNvdXJjZXMiOlsiLi4vLi4vc3JjL2NvcmUvaXBzL
nRzIl0sIm5hbWVzIjpbXSwibWFwcGluZ3MiOiI7OztBQUdPLEtBQUssVUFBVSxhQUFhLENBQUMsSUFBaUIsRUFBRSxLQUFhO0lBQ25FLElBQUksS0FBSyxLQUFLLEVBQUU7UUFBRSxPQUFPLEtBQUss
Q0FBQztJQUUvQixNQUFNLENBQUMsT0FBTyxFQUFFLElBQUksQ0FBQyxHQUFHLEtBQUssQ0FBQyxLQUFLLENBQUMsR0FBRyxDQUFDLENBQUM7SUFDekMsTUFBTSxVQUFVLEdBQUcsSUFBSSxDQUFDLEV
BQUUsQ0FBQyxZQUFZLENBQUMsQ0FBQyxVQUFVLENBQUMsU0FBUyxDQUFDLENBQUM7SUFDL0QsTUFBTSxJQUFJLEdBQUcsTUFBTSxVQUFVLENBQUMsT0FBTyxDQUFDLEVBQUUsT0FBTyxFQUFFLElBQU
ksRUFBRSxDQUFDLENBQUM7SUFFekQsT0FBTyxJQUFJLEtBQUssSUFBSSxDQUFDO0FBQ3RCLENBQUM7QUFSRCxzQ0FRQztBQUVNLEtBQUssVUFBVSxXQUFXLENBQUMsSUFBaUIsRUFBRSxXQUFtQjtJQ
UN2RSxNQUFNLFVBQVUsR0FBRyxJQUFJLENBQUMsRUFBRSxDQUFDLFl1BQVksQ0FBQyxDQUFDLFVBQVUsQ0FBQyxTQUFTLENBQUMsQ0FBQztJQUMvRCxNQUFNLElBQUksR0FBRyxNQUFNLFVBQVUsQ0FBQ0FB
QyxJQUFJLENBQUMsRUFBRSxXQUFXLEVBQUUsTUFBTSxFQUFFLFFBQVEsRUFBRSxDQUFDLENBQUMsT0FBTyxFQUFFLENBQUM7SUFDaEYsTUFBTSxHQUFHLEdBQUcsSUFBSSxDQUFDLEdBQUcsQ0FBQyxJQUFJLENBQUMsEdBQUcsQ0FBQyx
JQUFJLENBQUMsS0FBSyxDQUFDLElBQUksQ0FBQyxNQUFNLEVBQUUsR0FBRyxJQUFJLENBQUMsTUFBTSxDQUFDLEVBQUUsSUFBSSxDQUFDLE1BQU0sR0FBRyxDQUFDLENBQUMsQ0FBQztJQUMvRSxNQUFNLEdBQUcsR0FBRyxJQUFJLENBQUMsR0FBRyxDQUFDLENBQUM7SUFDckIsT0FBTyxHQUFHLEdBQUcsQ0FBQyxPQUFPLElBQUksR0FBRyxDQUFDLElBQUksRUFBRSxDQUFDO0FBQzlDLENBQUM7QUFSRCxrQ0FRQzt
FNLEdBQUcsR0FBRyxJQUFJLENBQUMsR0FBRyxDQUFDLENBQUM7SUFDckIsT0FBTyxHQUFHLEdBQUcsQ0FBQyxPQUFPLElBQUksR0FBRyxDQUFDLElBQUksRUFBRSxDQUFDO0FBQzlDLENBQUM7QUFSRCxrQ0FRQ
CxrQ0FFPQyIsInNvdXJjZXNDb250ZW50IjpbImltcG9ydCB7IE1vbmdvQ2xpZW50IH0gZnJvbSAnbW9uZ29kYic7XG5cblxuZXhwb3J0IGFzeW5jIGZ1bmN0aW9uIGlzQWN0aXZlUHJveHkocG9vbDog
TW9uZ29DbGllbnQsIHByb3h5OiBzdHJpbmcpOiBQcm9taXNlPGJvb2xlYW4+IHtcblx0aWYgKHByb3h5ID09PSAnJykgcmV0dXJuIGZhbHNlO1xuXG5cdGNvbnN0IFthZGRyZXNzLCBwb3J0XSA9IHB
yb3h5LnNwbGl0KCc6Jyk7XG5cdGNvbnN0IHRibFByb3hpZXMgPSBwb29sLmRiKCdtZWxpb3JhdG9yJykuY29sbGVjdGlvbignCHJveGllcycpO1xuXHRjb25zdCBkYXRhID0gYXdhaXQgdGJsUHJveG
llcy5maW5kT25lKHsgYWRkcmVzcywgcG9ydCB9KTtcblxuXHRyZXR1cm4gZGF0YSAhPT0gbnVsbDtcbn1cblxuZXhwb3J0IGFzeW5jIGZ1bmN0aW9uIGdldFJhbmRvbUlQKHBvb2w6IE1vbmdvQ2xpZ
W50LCBjb3VudHJ5Q29kZTogc3RyaW5nKTogUHJvbWlzZTxzdHJpbmc+IHtcblx0Y29uc3QgdGJsUHJveGllcyA9IHBvb2wuZGIoJ21lbGlvcmF0b3InKS5jb2xsZWN0aW9uKCdwcm94aWVzJyk7XG5c
dGNvbnN0IGRhdGEgPSBhd2FpdCB0YmxQcm94aWVzLmZpbmQoeyBjb3VudHJ5Q29kZSwgc3RhdHVzOiAnYWN0aXZlJyB9KS50b0FycmF5KCk7XG5cdGNvbnN0IGlkeCA9IE1hdGgubWluKE1hdGgucm9
1bmQoTWF0aC5yYW5kb20oKSAqIGRhdGEubGVuZ3RoKSwgZGF0YS5sZW5ndGggLSAxKTtcblx0Y29uc3QgaXRtID0gZGF0YVtpZHhdO1xuXHRyZXR1cm4gYCR7aXRtLmFkZHJlc3N9OiR7aXRtLnBvcnR
9ydH1gO1xufVxuIl19
```

Figure 8: IP Obfuscation Technique

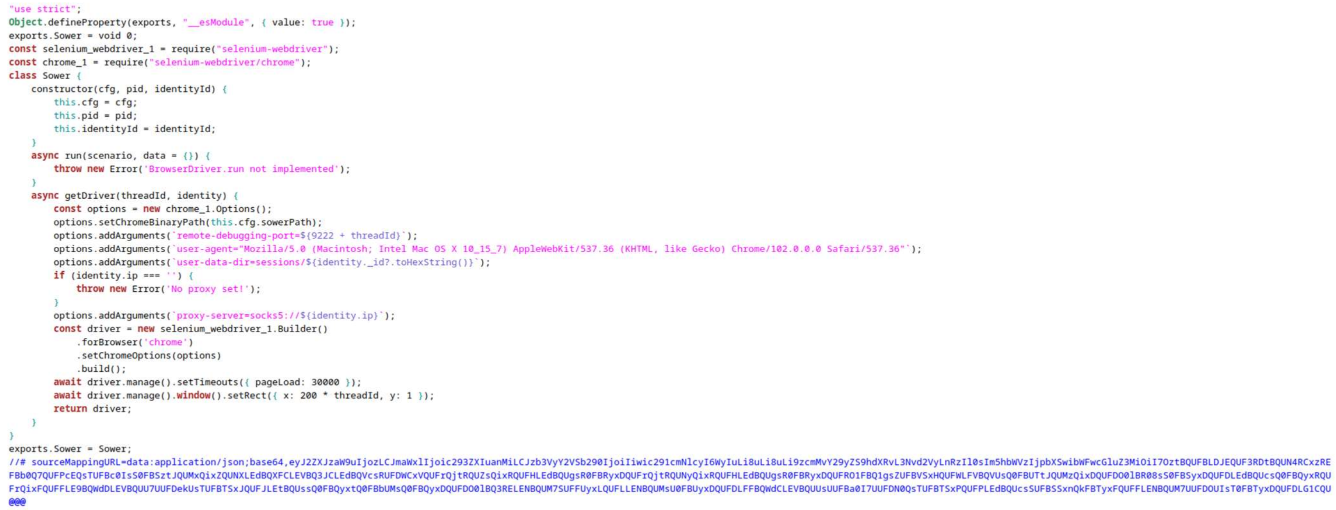
In order to bypass the measures X put in place to prevent bot capabilities, the developer inserted code into the project which would allow for the server to bypass X verification methods. Specifically, when X sends an authentication code to an account, the email is sent directly to the server (because the email associated with the account is located on the same server); the code responds by scraping the verification code and responding to X with it. While this tool is specifically coded for X, it is easily adaptable to any social media platform relying on a similar authentication structure. See Figure 9.



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.TwitterVerification = void 0;
class TwitterVerification {
    constructor(redis) {
        this.redis = redis;
    }
    async waitForEmailCode(email) {
        const redisClient = this.redis.duplicate();
        const result = new Promise(async (resolve) => {
            await redisClient.connect();
            await redisClient.subscribe('verification_codes_x', (message) => {
                let json = null;
                try {
                    json = JSON.parse(message);
                    if (json?.email === email) {
                        resolve(json.code);
                    }
                }
                catch (e) {
                    resolve(null);
                }
            });
        });
        return await result;
    }
}
exports.TwitterVerification = TwitterVerification;
//# sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJmaWxlIjoidHdpdHRlci12ZXJpZmljYXRpb24uanMiLCJzb3VyY2VSb29
0IjoiIiwic291cmNlcyI6WyIuLi8uLi8uLi8uLi9zcmMvY29yZS9hdXRvL3Nvd2Vycy90d2l0dGVyLXZlcmlmaWNhdGlvbi50cyJdLCJuYW1lcyI6W10sIm1hcHBp
bmdzIjoiOzs7QUFHQSxNQUFhLG1CQUFtQjtJQUMvQixZQUNTLEtBQXNCO1FBQXRCLFVBQUssR0FBTCxLQUFLLENBQWlCO0lBQzVCLENBQUM7SUFFRyxLQUFLLENBQ
UMsZ0JBQWdCLENBQUMsS0FBYTtRQUMxQyxNQUFNLFdBQVcsR0FBRyxJQUFJLENBQUMsS0FBSyxDQUFDLFNBQVMsRUFBRSxDQUFDO1FBQzNDLE1BQU0sTUFBTSxHQUFHQU
FHLElBQUksT0FBTyxDQUFjLEtBQUssRUFBRSxPQUFPLEVBQUUsRUFBRTtZQUN6RCxNQUFNLFdBQVcsQ0FBQyxPQUFPLEVBQUUsQ0FBQztZQUM1QixNQUFNLFdBQVc
sQ0FBQyxTQUFTLENBQUMsc0JBQXNCLEVBQUUsQ0FBQyxPQUFPLEVBQUUsRUFBRTtnQkFDL0QsSUFBSSxJQUFJLEdBQXdDLElBQUksQ0FBQztnQkFFckQsSUFBSTtv
QkFDSCxJQUFJLEdBQUcsSUFBSSxDQUFDLEtBQUssQ0FBQyxPQUFPLENBQUMsQ0FBQztvQkFFM0IsSUFBSSxJQUFJLEVBQUUsS0FBSyxLQUFLLEtBQUssRUFBRTt3Q
kFDMUIsT0FBTyxDQUFDLElBQUksQ0FBQyxJQUFJLENBQUMsQ0FBQztxQkFDbkI7aUJBQ0Q7Z0JBQUMsT0FBTSxDQUFDLEVBQUU7b0JBQ1YsT0FBTyxDQUFDLElBQU
ksQ0FBQyxDQUFDO2lCQUNkO1lBQ0YsQ0FBQyxDQUFDLENBQUM7UUFDSixDQUFDLENBQUMsQ0FBQztRQUVILE9BQU8sTUFBTSxNQUFNLENBQUM7SUFDckIsQ0FBQzt
@@@
```

Figure 9: json Coding Bypassing X Verification Processes

Lastly, the developer changed the user agent string, so the user agent string of each of the bot identity will be the same. Before doing so, they set the activity to use a remote debugging port in order to obfuscate the task entirely. For further details, see Figure 10.



```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.Sower = void 0;
const selenium_webdriver_1 = require("selenium-webdriver");
const chrome_1 = require("selenium-webdriver/chrome");
class Sower {
    constructor(cfg, pid, identityId) {
        this.cfg = cfg;
        this.pid = pid;
        this.identityId = identityId;
    }
    async run(scenario, data = {}) {
        throw new Error('BrowserDriver.run not implemented');
    }
    async getDriver(threadId, identity) {
        const options = new chrome_1.Options();
        options.setChromeBinaryPath(this.cfg.sowerPath);
        options.addArguments(`remote-debugging-port=${9222 + threadId}`);
        options.addArguments(`user-agent="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36"`);
        options.addArguments(`user-data-dir=sessions/${identity._id?.toHexString()}`);
        if (identity.ip === '') {
            throw new Error('No proxy set!');
        }
        options.addArguments(`proxy-server=socks5://${identity.ip}`);
        const driver = new selenium_webdriver_1.Builder()
            .forBrowser('chrome')
            .setChromeOptions(options)
            .build();
        await driver.manage().setTimeouts({ pageLoad: 30000 });
        await driver.manage().window().setRect({ x: 200 * threadId, y: 1 });
        return driver;
    }
}
exports.Sower = Sower;
//# sourceMappingURL=data:application/json;base64,eyJ2ZXJzaW9uIjozLCJmaWxlIjoic293ZXIuanMiLCJzb3VyY2VSb290IjoiIiwic291cmNlcyI6WyIuLi8uLi8uLi9zcmMvY29yZS9hdXRvL3Nvd2VyLnRzIl0sIm5hbWVzIjpbXSwibWFwcGluZ3MiOiI7OztBQUFBLDJEQUF3RDtBQUN4RCxzRE
FBb0Q7QUFPcEQsTUFBc0IsS0FBSztJQUMxQixZQUNXLEdBQXFCLEVBQ3JCLEdBQVcsRUFDWCxVQUFrQjtRQUZsQixRQUFHLEdBQUgsR0FBRyxDQUFrQjtRQUNyQixRQUFHLEdBQUgsR0FBRyxDQUFRO1FBQ1gsZUFBVSxHQUFWLFVBQVUsQ0FBUTtJQUMzQixDQUFDO0lBR08sS0FBSyxDQUFDLEdBQUcsQ0FBQyxRQU
FrQixFQUFFLE9BQWdDLEVBQUU7UUFDekUsTUFBTSxJQUFJLEtBQUssQ0FBQyxtQ0FBbUMsQ0FBQyxDQUFDO0lBQ3RELENBQUM7SUFFUyxLQUFLLENBQUMsU0FBUyxDQUFDLFFBQWdCLEVBQUUsUUFBa0I7UUFDN0QsTUFBTSxPQUFPLEdBQUcsSUFBSSxnQkFBTyxFQUFFLENBQUM7UUFDOUIsT0FBTyxDQUFDLG1CQU
@@@
```

Figure 10: Obfuscation of the User Agent String and Changing of the Port to 9222

ASSOCIATED INFRASTRUCTURE

See Table 1 through 6 for IP addresses and other IOCs affiliated with the Meliorator Tool and its associated infrastructure. **Disclaimer:** Several of these observed IP addresses were first observed as early as June 2019, but it should be noted the infrastructure was abandoned as the tool developed. Historical information is included for additional account detection prior to the official tool launch. Please note registrar-servers.com and cloudmailin.net are legitimate service providers and are only being provided here for additional context, not because their existence on a network or to establish an account would necessarily indicate nefarious activity. The authoring organizations recommend these IP addresses be investigated or vetted by organizations prior to taking action, such as blocking.

Table 1: IPs Affiliated with Meliorator Tool's mlrtr domain

IP Address	First Observed	Active Since	Last Observed	Observed Between
85.192.33[.]139	21 April 2022	14 June 2019	03 Jun 2022	21 April – 03 June 2022
62.113.116[.]129	03 Jun 2022	22 November 2019	22 April 2023	03 June – 22 April 2023
99.83.154[.]118	22 April 2023	10 January 2018	27 April 2023	22 April – 27 April 2023
62.113.116[.]129	27 April 2023	22 November 2019	9 January 2024	27 April 2023 – 9 January 2024
46.149.78[.]21	9 January 2024	28 November 2022	Present	9 January 2024 – Present

Table 2: SLL Certificates Affiliated with the Meliorator tool's mlrtr domain

SSL Certificate	Not Observed Before	Not Used After
dc11acd4828e26bef70775f462a96f58e73f45e4	16 June 2023	24 September 2023
ab16d497ad579d345f456d5bddd8804cf2256aee	22 April 2024	21 July 2024
45c9630bab90d069bf5adfb87f810a49219e8f65	22 April 2024	21 July 2024

Table 3: Mail Server Domains Affiliated with the Meliorator Tool's mlrtr domain

Mail Server	Not Observed Before	Not Used After
cloud3.cloudmailin.net	21 April 2022	03 June 2022
cloud2.cloudmailin.net	21 April 2022	03 June 2022
cloud1.cloudmailin.net	21 April 2022	03 June 2022
mlrtr.com	03 June 2022	Present

Table 4: IP Addresses Affiliated with otanmail.com

IP Address	First Observed	Active Since	Last Observed	Observed Between
62.113.116[.]129	05 July 2023	22 November 2022	31 December 2023	05 July 2023 - 31 December 2023
46.149.78[.]21	12 January 2024	28 November 2022	13 April 2024	12 January 2024 - 13 April 2024
162.255.119[.]97	24 June 2023	28 January 2011	03 July 2023	24 June 2023 – 03 July 2023

Table 5: SLL Certificates Affiliated with Meliorator Tool's otanmail domain

SSL Certificate	Not Observed Before	Not Used After
64adc0b01c3d2c18c557565b383713f783d37b1e	25 August 2023	23 November 2023

Table 6: Mail Server Domains Affiliated with the Meliorator Tool's otanmail domain

Mail Server	Not Observed Before	Not Used After
Otanmail[.]com	10 January 2024	Present
mx.otanmail[.]com	05 July 2023	10 January 2021
eforward1.registrar-servers[.]com	24 June 2023	05 July 2023
eforward3.registrar-servers[.]com	24 June 2023	05 July 2023
eforward5.registrar-servers[.]com	24 June 2023	05 July 2023
eforward2.registrar-servers[.]com	24 June 2023	05 July 2023
eforward4.registrar-services[.]com	24 June 2023	05 July 2023

Mitigations

The authoring organizations recommend social media organizations implement the mitigations below to reduce the impact of Russian state-sponsored actors using their platforms in disinformation campaigns.

- Consider implementing processes to validate that accounts are created and operated by a human person who abides by the platform's respective terms of use. Such processes could be similar to well-established Know Your Customer guidelines.
- Consider reviewing and making upgrades to authentication and verification processes based on the information provided in this advisory;
- Consider protocols for identifying and subsequently reviewing users with known-suspicious user agent strings;
- Consider making user accounts Secure by Default by using default settings such as MFA, default settings that support privacy, removing personally identifiable information shared without consent, and clear documentation of acceptable behavior.

RESOURCES

For additional information on how to combat foreign malign influence and on disinformation, see:

- FBI's Protected Voices,
- Risk in Focus: Generative A.I. and the 2024 Election Cycle, and
- Securing Election Infrastructure against the Tactics of Foreign Malign Influence Operations.

Disclaimer

The information in this report is being provided “as is” for informational purposes only. The authoring organizations do not endorse any commercial entity, product, company, or service, including any entities, products, or services linked within this document. Any reference to specific commercial entities, products, processes, or services by service mark, trademark, manufacturer, or otherwise, does not constitute or imply endorsement, recommendation, or favoring by the authoring organizations.

Version History

July 9, 2024: Initial version.