RedEnergy Stealer | ThreatLabz

By Shatak Jain, Gurkirat Singh

Published: 2023-06-21 · Archived: 2026-04-05 22:02:52 UTC

Technical Analysis

The RedEnergy malware under investigation exhibits a dual functionality, acting both as a stealer and a ransomware. This .NET file, intentionally obfuscated by its author, possesses advanced capabilities to evade detection and hinder analysis. To establish communication with its command and control servers, the malware utilizes HTTPS, adding an additional layer of encryption and obfuscation.

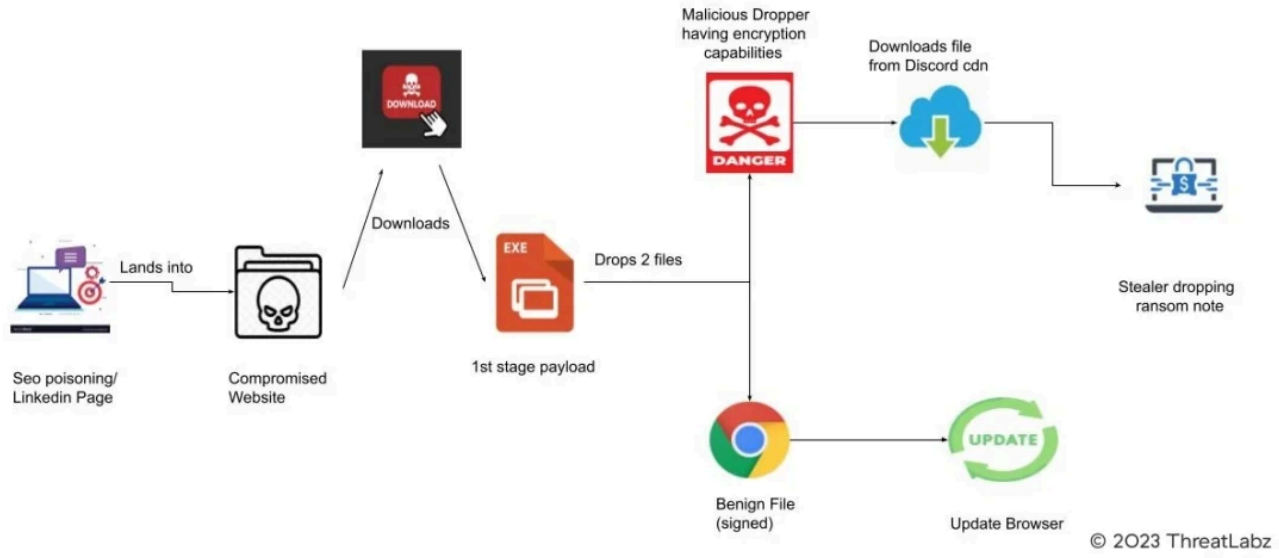

Fig 6. - Infection chain

The execution of this malware unfolds in three distinct stages, each serving a specific purpose. Each stage is outlined in the sections below.

Stage 1: Initial Startup

Upon execution, the malicious RedEnergy executable masquerades as part of a legitimate browser update, depicted in Fig. 7 below. It cleverly disguises itself with a legitimate update from one of the various popular browsers, including Google Chrome, Microsoft Edge, Firefox, and Opera, to deceive the user. Notably, looking at the properties of the malicious executable reveals the presence of an invalid certificate, however at surface level this attack hides behind a genuine signed certificate from the user’s browser as shown by the Google example examined in Fig. 8 below. This deceptive tactic aims to instill trust and convince the victim of the authenticity of the update.

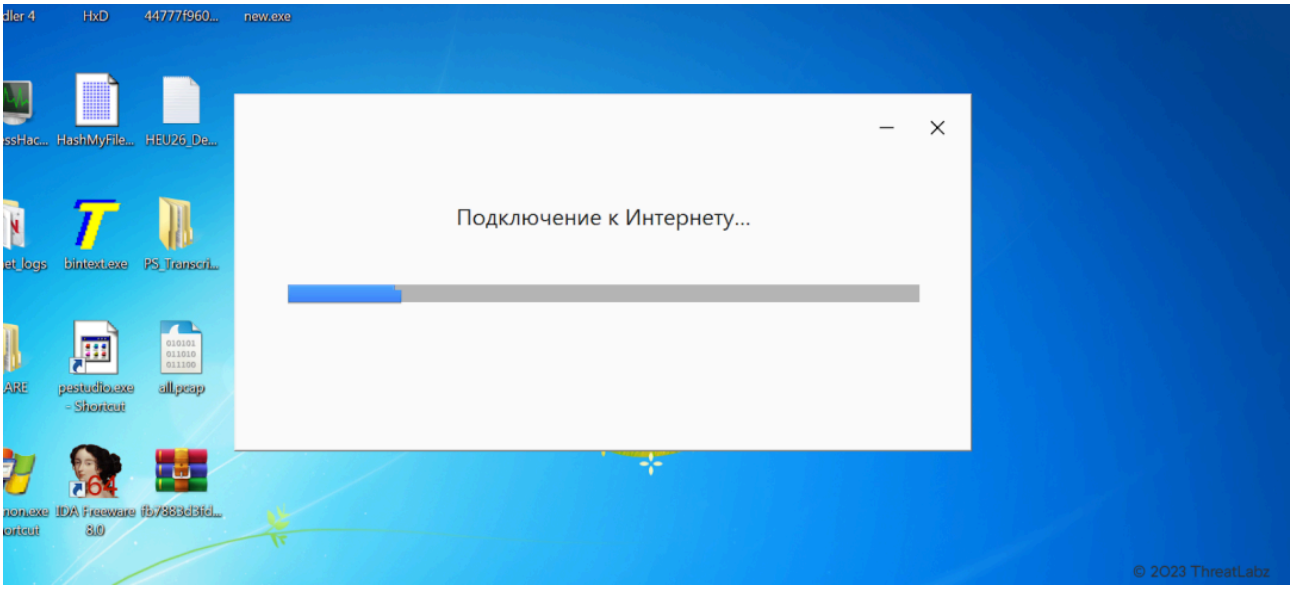

Fig 7. - Google updater executing the malicious RedEnergy binary

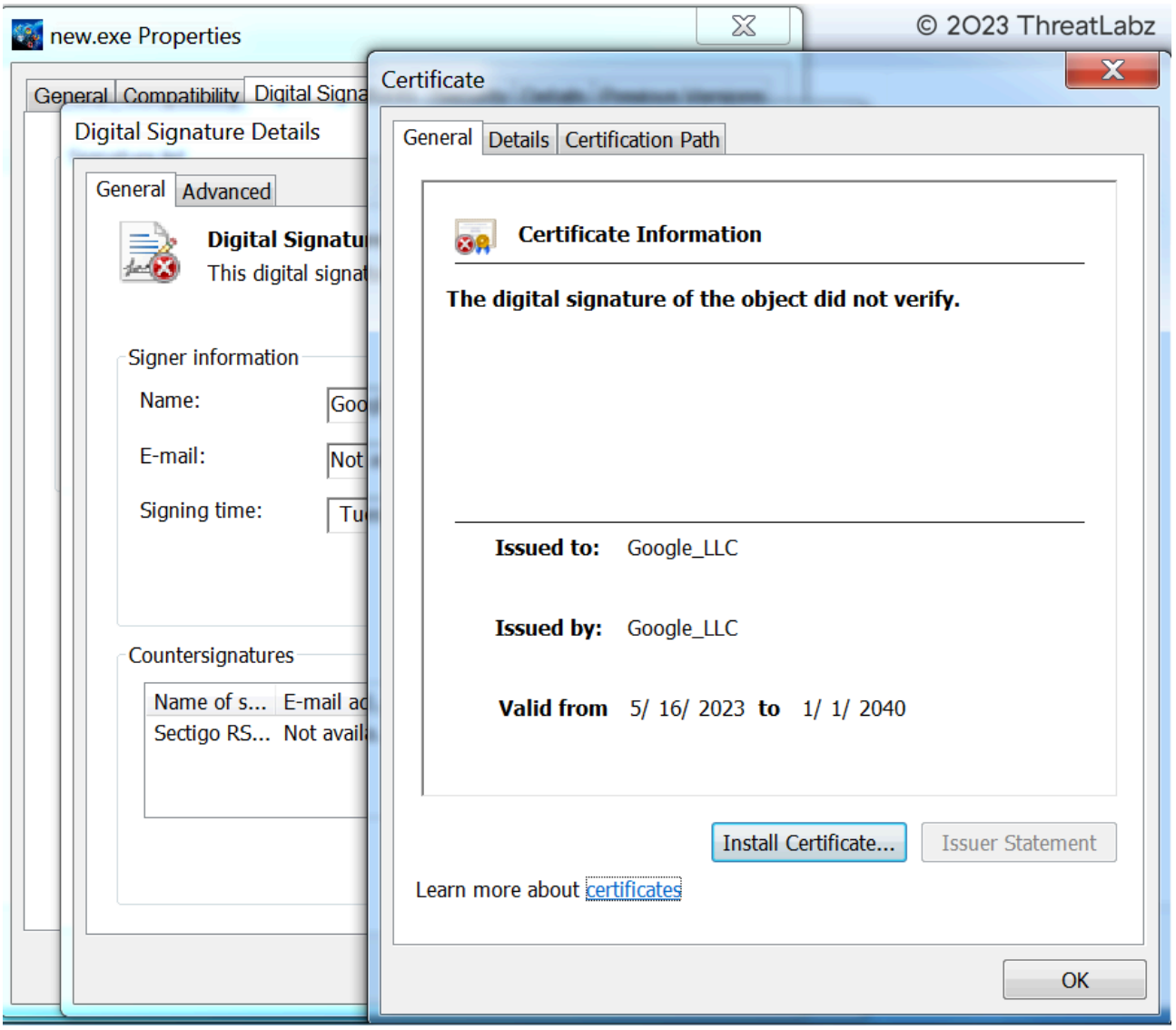

Fig 8. - Fake certificate

Stage 2: Dropping Files, Persistence, Outgoing Requests, Encrypted Files

Dropping Files:

In this stage, the malware drops four files onto the victim's system, shown in Fig. 9 below, precisely within the path **%USERPROFILE%\AppData\Local\Temp**. These dropped files consist of two temporary files and two executables, all following a similar pattern with filenames beginning with "**tmp**" and four randomly generated hexadecimal characters, followed by the "**.exe**" extension: **tmp[4 random hex characters].exe**. Among the executable files, one serves as the malicious payload, while the other disguises itself as the legitimate, digitally signed Google Update. The benign executable possesses the hash value **8911b376a5cd494b1ac5b84545ed2eb2** and is responsible for performing the actual update of Google Chrome, thereby further deceiving the victim. Simultaneously, the malware executes another background process, identified by the MD5 hash **cb533957f70b4a7ebb4e8b896b7b656c**, which represents the true malicious payload. During execution, this payload displays an inappropriate message on the victim's screen, displayed in Fig. 10 below, most likely as part of the threat actor's intent to cause distress or confusion.

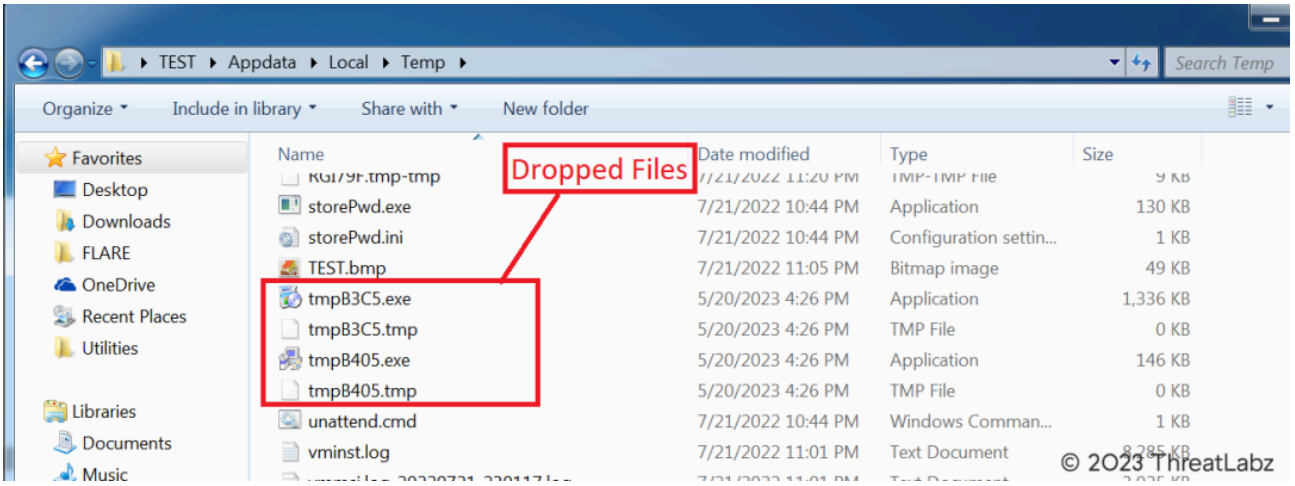

Fig 9. - Dropping malicious file in temp directory

Fig 10. - Display message after executing the binary

Persistence:

Persistence is a critical aspect of malware, enabling it to maintain its presence on an infected system even after rebooting or shutting down. To achieve persistence, the malicious executable stores files in the Windows startup directory. It creates an entry within the start menu (**Start Menu\Programs\Startup**) and initiates an immediate reboot, ensuring that the malware is executed once the system is up and running again. This persistence mechanism guarantees that the malware remains active and continues its malicious operations even after system restarts.

Outgoing Requests:

During the analysis of the malware, researchers utilized Fakenet, a Windows malware analysis tool that simulates network activity, to gain insights into its behavior. Through Fakenet, they discovered that the malicious tmp.exe file established communication with the DNS server **2no.co**, depicted in Fig. 11 below. To delve deeper into the network interactions, the widely used packet analysis tool, Wireshark, was employed. This allowed researchers to identify the specific DNS query made by the malicious tmp.exe file, providing crucial information for further investigation, as shown in Fig. 12 below. It was observed that upon establishing a connection with the DNS server, tmp.exe was expected to initiate the download of an executable file from cdn.discord. Unfortunately, during this particular analysis, the Command and Control (CnC) server was unavailable, making it impossible to obtain a sample. However, another sample resembling the final payload was discovered, which had been hosted on the same domain just two days prior to the current analysis.

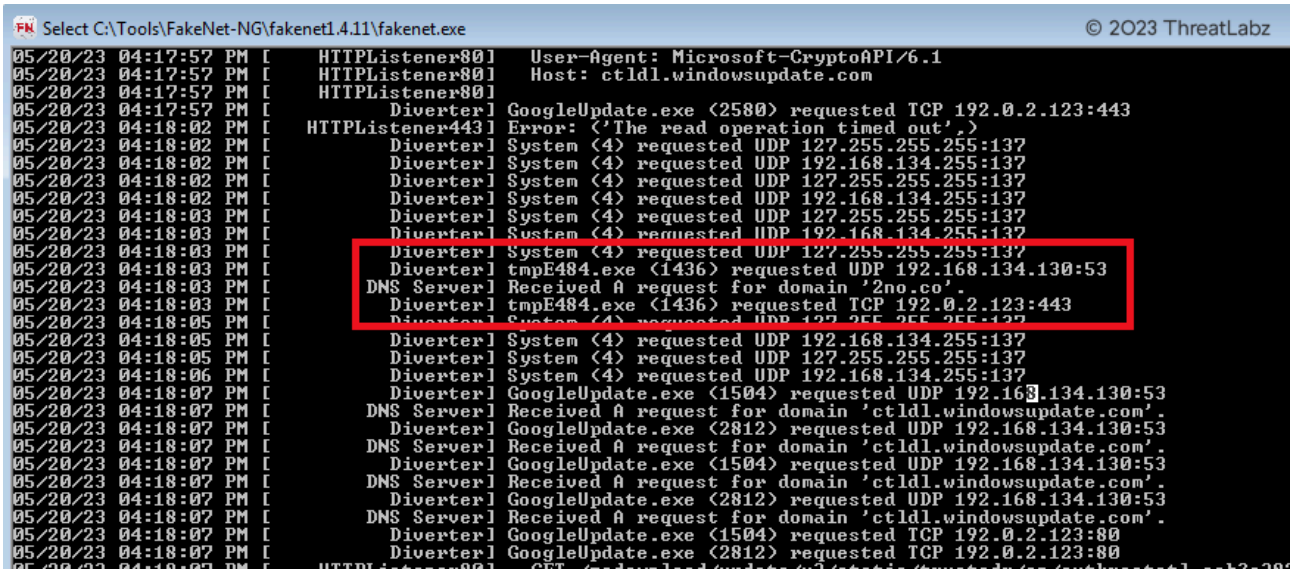

Fig 11. - Malicious binary communication with CnC server

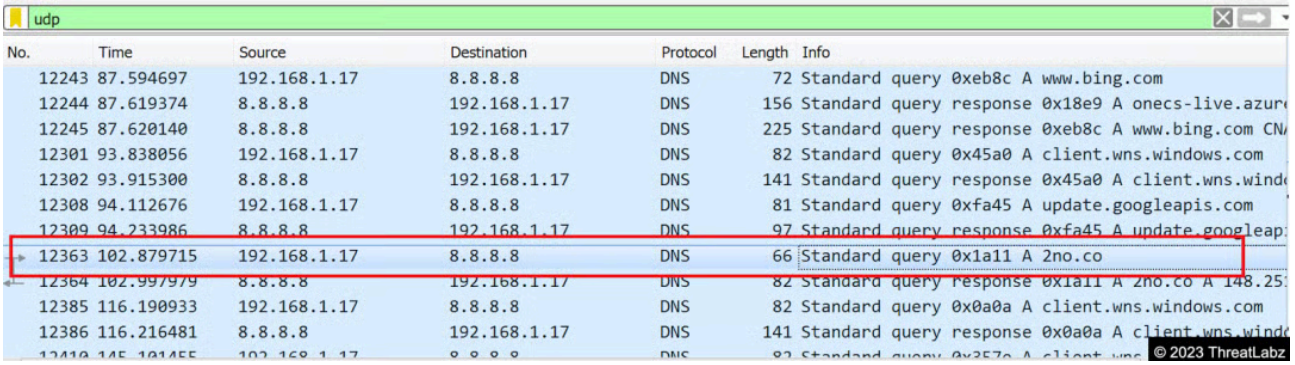

Fig 12. - Network communication seen via Wireshark

Additionally, suspicious activity involving File Transfer Protocol (FTP) was uncovered during the investigation. A user with the username "**alulogrofp**" successfully accessed a private system hosted by OVH, a renowned cloud computing company and one of the largest hosting providers globally. The user's credentials were authenticated, granting them access to a restricted directory, which was identified as the root directory ("/"). Notably, UTF-8 encoding was enabled for file transfers, indicating support for international character sets.

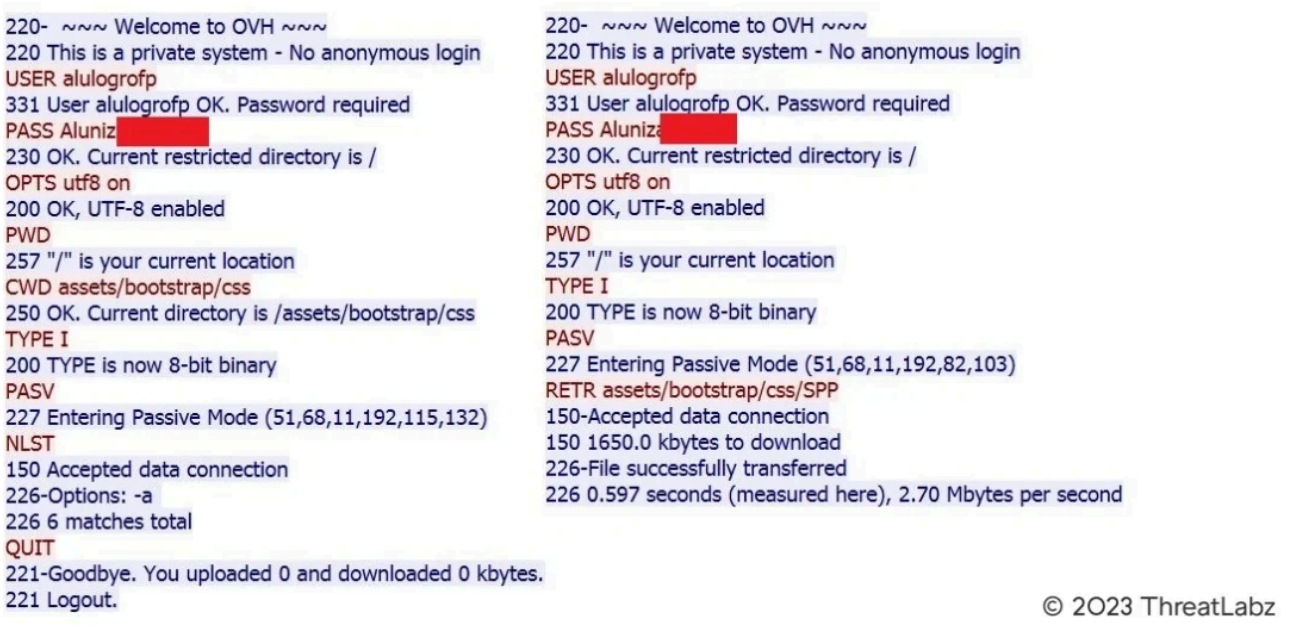

Fig 13. - FTP interaction on OVH private system

Within the FTP session, the user navigated to the "/assets/bootstrap/css" directory, following standard directory traversal practices. To ensure efficient and accurate file transfers, the transfer mode was set to binary (8-bit). Subsequently, the server entered passive mode and provided an IP address and port number, indicated by the message "Entering Passive Mode (51,68,11,192,115,132)". By combining the extracted data, the IP address 51.68.11[.]192 was obtained. Further interactions revealed that the user requested a file list using the "NLST" command, resulting in the retrieval of six matching files.

In another session, the client initiated a file retrieval operation using the "RETR" command, specifying the file path as "assets/bootstrap/css/SPP". The server acknowledged the data connection and confirmed the acceptance of the file transfer.

These FTP interactions raised concerns regarding potential data exfiltration, as well as the possibility of uploading files using the same method.

Encrypted Files:

With ransomware modules integrated into the payload, the malware proceeded to encrypt the user's data, appending the ".FACKOFF!" extension to each encrypted file, as shown in Fig. 14 below. This malicious software is specifically designed to lock the user's files, rendering them inaccessible until a ransom is paid. After the encryption process is completed, the user receives a ransom message, demanding payment in exchange for restoring access to their files. Failure to comply with the ransom demands results in the permanent loss of access to the compromised data.

Furthermore, the malicious executable alters the desktop.ini file, which contains configuration settings for the file system folders. By modifying this file, the malware can manipulate how the file system folders are displayed, potentially further concealing its presence and activities on the infected system. This alteration serves as an attempt to mislead the user and impede the detection of the malware's impact on the file system.

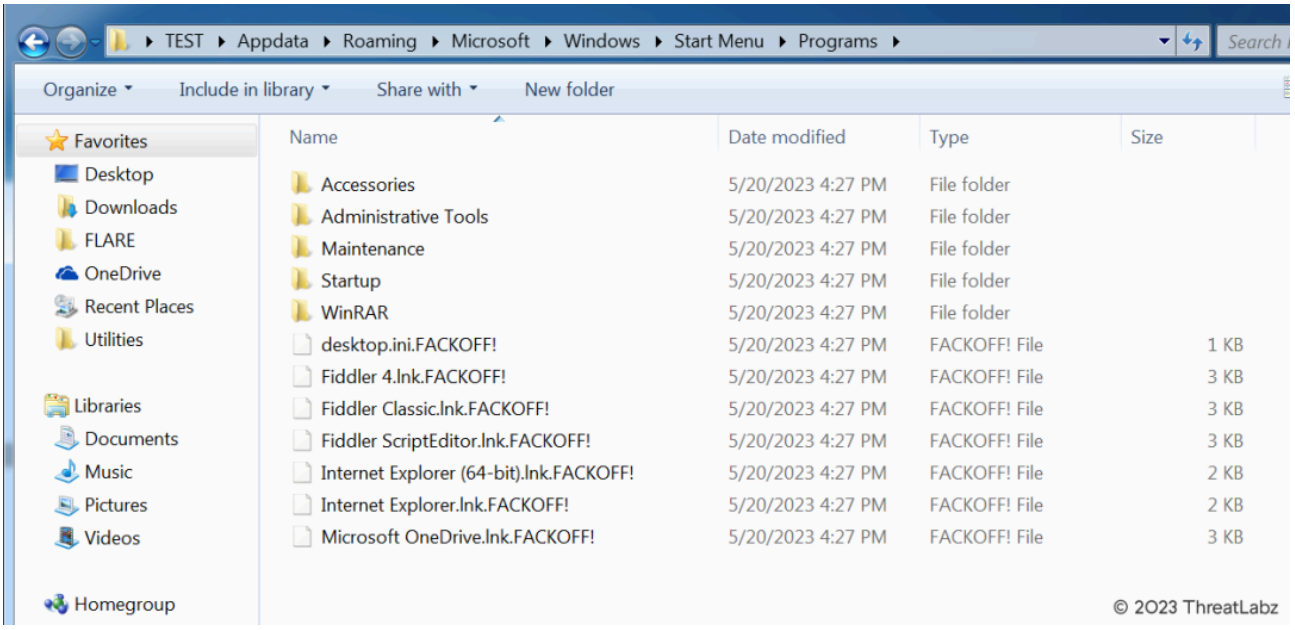

Fig 14. - Encrypted files with .FACKOFF! extension

Stage 3: Decryption Routine

The final stage payload is responsible for various actions, including dropping the ransom note and executing multiple commands and stealer functionalities, and for encryption it uses the **RijndaelManaged** algorithm. Within the payload, numerous functions are named RedEnergy, giving rise to its namesake.

In the second stage, the malware downloads the executable **SystemPropertiesProtection.exe** via the discord cdn. This leads to the third stage, where the malware executes a series of actions typically associated with ransomware. It begins by deleting data from the shadow drive, effectively removing any potential backups. The malware also targets Windows backup plans, further hindering the user's ability to recover their data. Additionally, a batch file is

executed, and a ransom note is dropped, indicating the user's files have been encrypted. Furthermore, the malware possesses stealer capabilities, allowing it to exfiltrate the user's data.

Notably, the Config method, shown in Fig. 15 below, plays a crucial role in decrypting key information. It stores important strings related to the stealer functionality in a dictionary, depicted in Fig. 16, which is used to construct command lines for further operations.

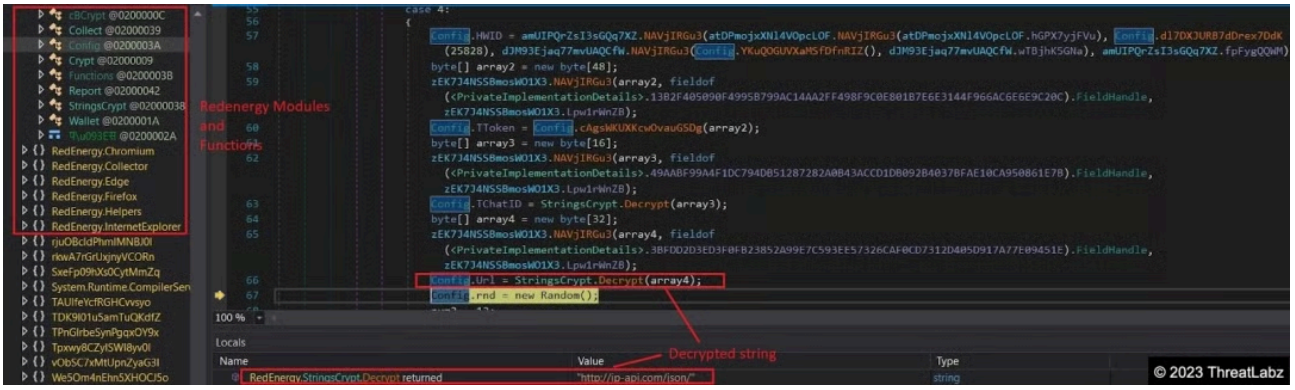

Fig 15. - Config decryption function

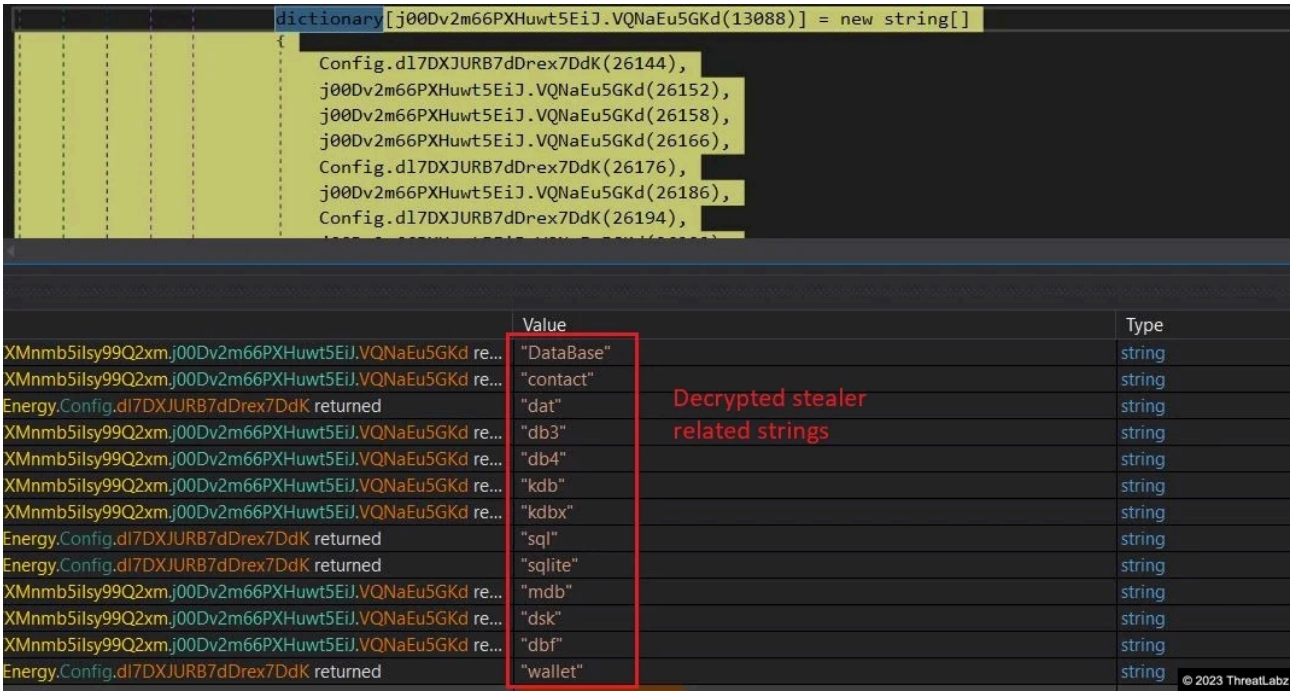

Fig 16. - Malware showcasing stealer capabilities

One such decrypted command line, shown in Figure 17, modifies the boot configuration to ignore failures and disables the automatic recovery options in Windows. The payload also drops specific files in the Temp directory, as seen in Figure 18, using it as a camouflage to conceal its malicious intent. Among the files dropped, C.bin serves as a payload, while a batch file contains commands to terminate processes and perform cleanup tasks associated with the payload. Figure 19 illustrates the instructions executed by the batch file.

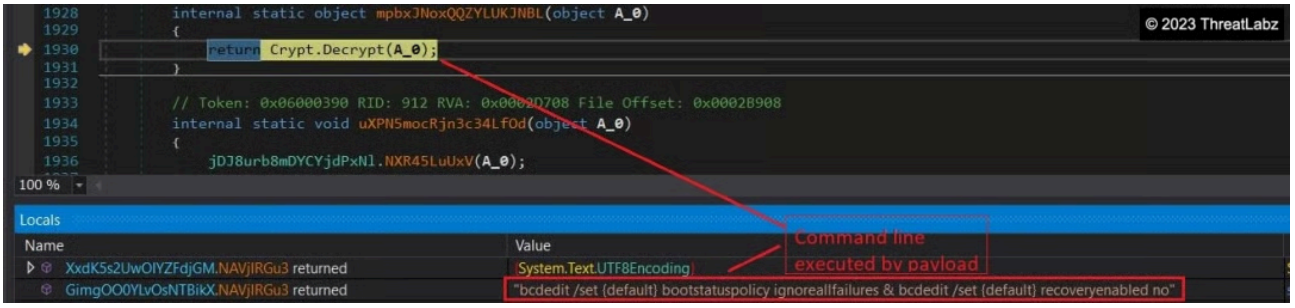

Fig 17. - Command line executed post decryption

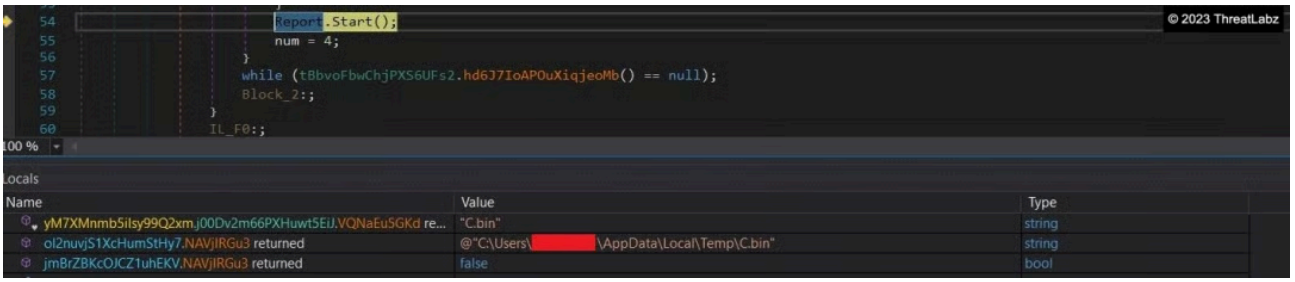

Fig 18. - Dropping supporting files in temp directory

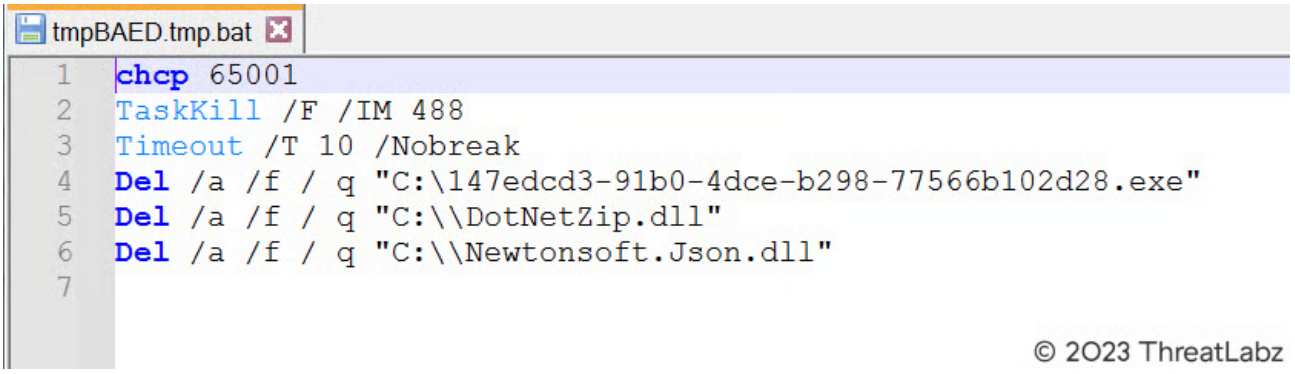

Fig 19. - Content inside batch file

Furthermore, the payload is programmed to delete all volume shadow copies (VSS), the backup catalog, and shadow copies using the Windows Management Instrumentation Command-line (WMIC). The following command lines exemplify this process:

- C:\Windows\System32\cmd.exe /C vssadmin delete shadows /all /quiet & wmic shadowcopy delete
- C:\Windows\System32\cmd.exe /C wbadmin delete catalog -quiet

Additionally, the payload undergoes a three-stage process to gather antivirus (AV) information. Based on this information, it generates a string that it sends to the Command and Control (CnC) server as a User Agent, as depicted in Figure 20 below. During the analysis, it was observed that the AV detected is Windows Defender. STM, RSM, and RZ likely provide additional information related to Windows Defender.

Lastly, the payload is responsible for dropping the final ransom note, **read_it.txt**, shown in Figure 21. This note is placed in all the folders where file encryption occurs, serving as a notification to the user that their files have been encrypted and demanding a ransom for their release.

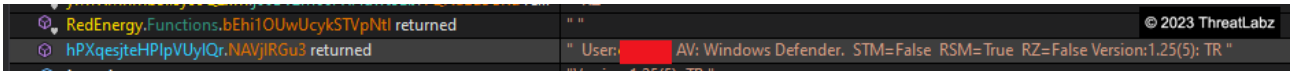

Fig 20. - User Agent built from malicious code storing AV information

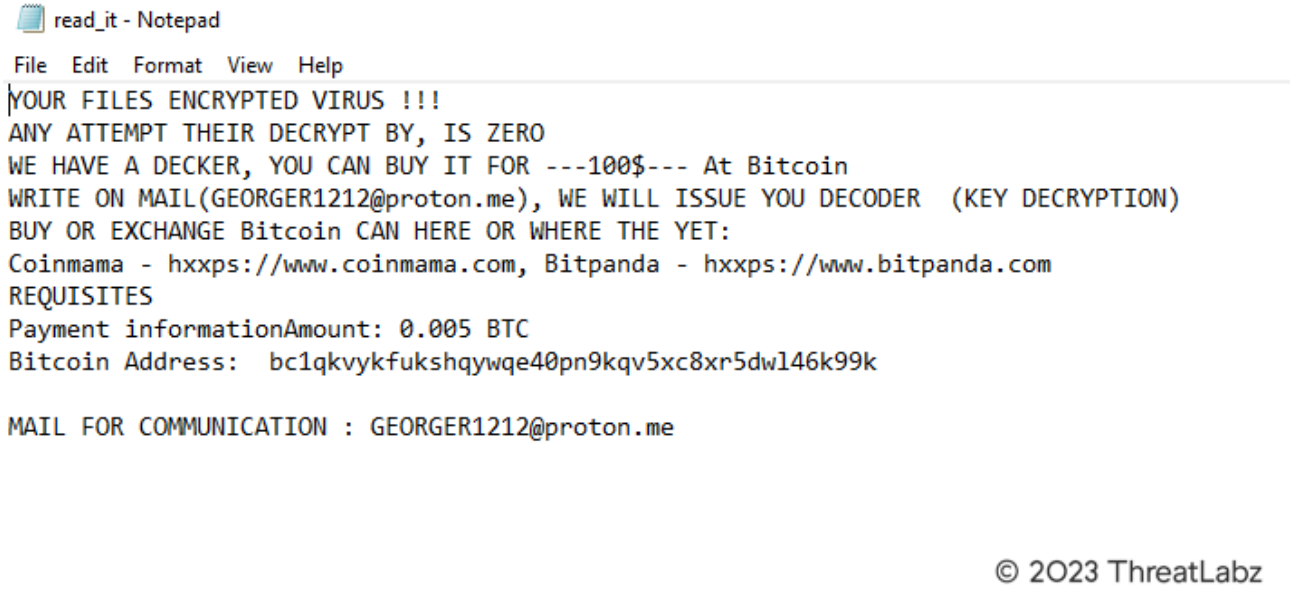

Fig 21. - Screenshot of the ransom note

Source: https://www.zscaler.com/blogs/security-research/ransomware-redefined-redenergy-stealer-ransomware-attacks