Malware Forensics Coursework [MalFor]

Malware Analysis Report

Student No.: 2068521
Word Count: 2089

Disclaimer

By submitting this report for marking, **I fully understand** that and **I agree**:

- This is an independent piece of coursework, and it is expected that I have taken responsibility for all the design, implementation, analysis of results and writing of the report. And, it is 2000 words (+/-10%).
- For the Turnitin plagiarism software, the match score of the report must be below 15% overall and less than 5% to an individual source (excluding appendices). Marks will be investigated accordingly. if appropriate.
- This marking scheme sets out what would be expected for each marking band for this module's coursework.
- The University "general criteria applicable to essays, reports and aspects of projects and dissertations" applies (on Moodle).
- It is blind marked.
- Any technical help, high-level advice and suggestions which may be provided in-person (e.g., labs) and electronically, has no contribution to or an indication of my final mark.
- Knowledge of a peer's work and their final mark is not justification for what my own mark should be.
- Any examples provided were used for ideas only, and "having followed an example" is not justification for what my mark should be.
- The coursework malware was used for analysis only and was isolated to a safe working environment (a virtual machine), to prevent the malware from infecting the host.

Word Count

Section	Word Count
Summary	140
Malware Details	39
Static Analysis	459
Dynamic Analysis	542
Reverse Engineering	557
Origins	137
Removal	175
Conclusions & Recommendations	103
Total	**2152**

Contents

1. Executive Summary

This report refers to figures detailed within the attached disassembly report.

Infected System Details

- Windows 11 Pro 64 bit
- OS Build 22000.2538

Summary of Reports Intentions

This comprehensive report, published by Longtext Cyber Security, aims to give a complete overview of its findings for the malware which infiltrated Gazprom's systems in late 2023. This report offers the malware's full functionality; statically and dynamically, determine the possible creator(s) behind the malware and give security recommendations to prevent a similar attack from occurring.

Summary of Functionality

- The Malfor executable file is a persistent malware that targets Intel 386 or later processors.
- It connects to a remote host over the IRC protocol.
- The infected system can be controlled by a remote botmaster and is a part of a botnet, a collection of infected systems which are controlled by malicious actor(s).

2. Malware Details

Overview

The details in this section are only related to the file features displayed within VirusTotal before and after it was unpacked. This analysis aims to give a foundational understanding of the malware.

Packed Malware Metadata

(Packed & Protected)		
Name & File Format	malfor-cw-sample.exe	.exe
File Size	464.79	KB
File Hash	f3883f8a9bceb7a00b36eac80c042623f87080cbd6ed017878c5ad7f3f0a0ce3	SHA-256
Community Score	**59 / 72**	
Threat Categories	Trojan, Downloader, Botnet	
Target Platform	Intel 386 and compatible processors	
Packer / Protector	Enigma Virtual Box	
Compiler	MinGW (GCC v6.3.0)	
Activity Summary		
Dropped Files	80 Other	
Network Communication	2 HTTP, 10 IP	

Figure 2.1 - Malware details of packed malware

Unpacked Malware Metadata

(Unpacked & Unprotected)		
Name & File Format	malfor-cw-sample_unpacked.exe	Win32.exe
File Size	56.79	KB
File Hash	3b0422bff4061fa53574d3724f58cf9e695069ece1b8c9ad69a2b69fe62a3dcb	SHA-256
Community Score	**56 / 72**	
Threat Categories	Trojan, Botnet	
Target Platform	Intel 386 and compatible processors	
Packer / Protector	N/A	
Compiler	MinGW (GCC v6.3.0)	
Activity Summary		
Dropped Files	14 Other, 1 Text	
Network Communication	5 HTTP, 6 IP	

Figure 2.2 - Malware details of packed malware

3. Static Analysis

3.1 Packer

Overview

A malware packer is a tool used to obfuscate the malware in such a way that makes the disassembly of said malware hard to perform. In the current scenario of ‘malfor-cw-sample.exe’ [Malfor], it made it impossible to analyse the functionality of the malware, as the individual function names and Windows API calls were replaced with byte locations, as seen in examples **Figure 3.11** and **Figure 3.12**.

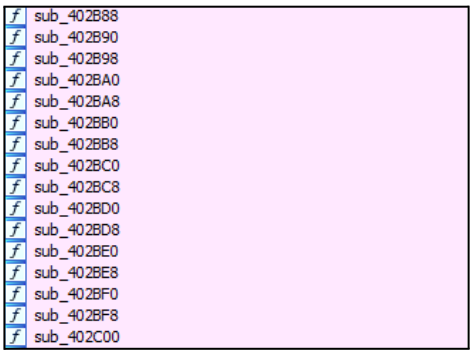


Figure 3.12 - Packed Malware Functions

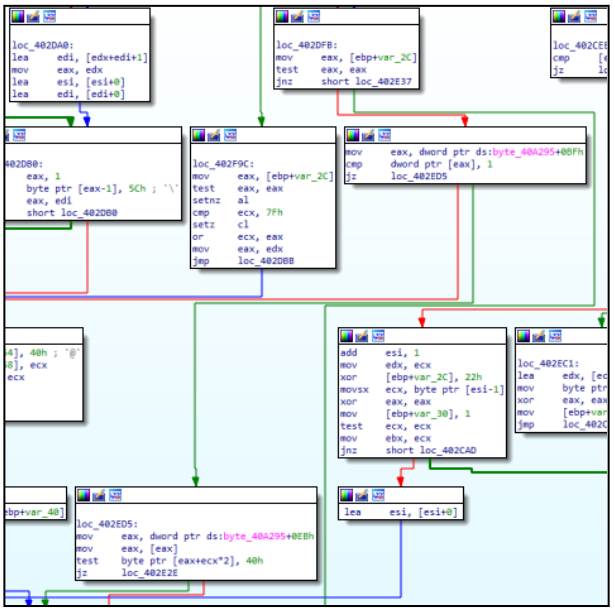


Figure 3.11 - Portion of Packed Malware Function

PEid

When using the tool ‘PEid’ on Malfor, intended for analysing Windows executable programs, it showed no packer was used / present within the malware (**Figure 3.13**, **3.14** and **3.15**), even after using all three scans available (Normal, Deep and Hardcore).

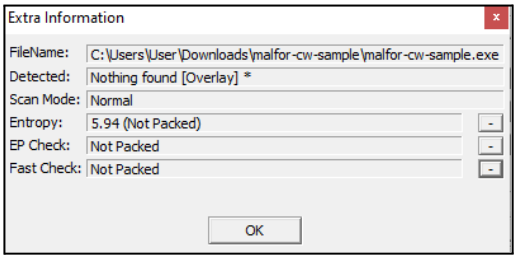


Figure 3.13 - Results of Normal scan by ‘PEid’

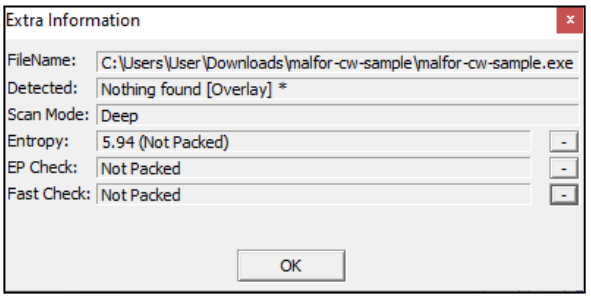


Figure 3.14 - Results of Deep scan by ‘PEid’

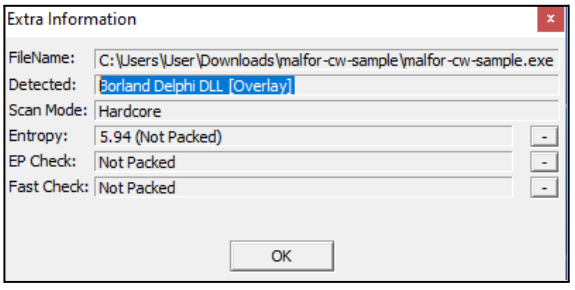


Figure 3.15 - Results of Hardcore scan by ‘PEid’

VirusTotal

The next tool used to attempt to find the correct packer for this malware was VirusTotal, an online service used for analysing and detecting malicious files & URLs.

Presented under the details section of the malware was ‘Enigma Virtual Box’, a software that combines all dependencies and files into one executable file. After unpacking Malfor with this software, the executable file now had readable function names, with the Windows API function calls used in each function now visible, therefore making the malware significantly more straightforward to analyse and disassemble.

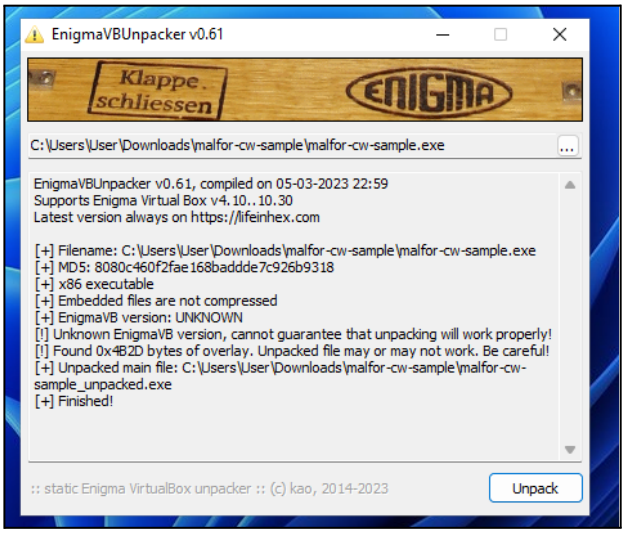


Figure 3.16 - Unpacking malfor-cw-sample.exe

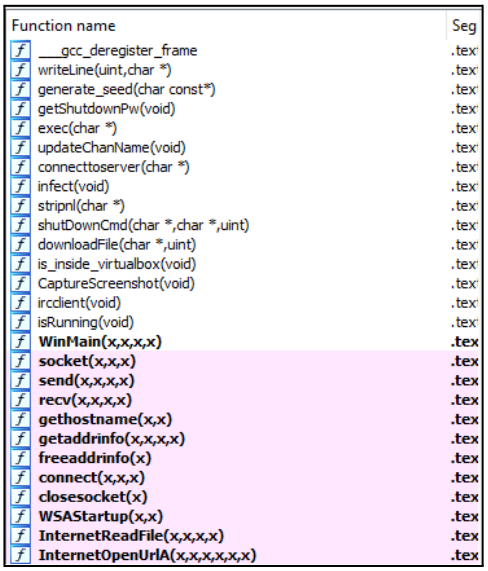


Figure 3.17 - malfor-cw-sample.unpacked.exe function names

3.2 Strings Analysis

Overview

The ‘strings’ command is used to extract readable text from binary files, as they often contain sequences of characters or strings not visible when viewing the file normally.

Results

In this use case, this can be incredibly useful to preview possible function names, embedded text, API calls, libraries and much more, even before the file is unpacked. This is seen in **Figure 3.21** and **Figure 3.22.** Using strings was a notable reason to assume this malware was packed, as these functions were not visible when viewed in IDA.

Figure 3.21 - Possible Function names using strings

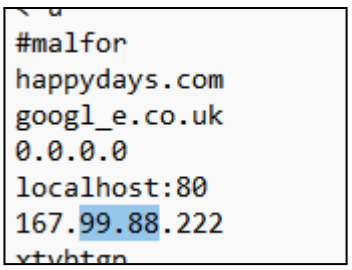


Figure 3.22 - Possible domain and IP address using strings

3.3 Malware Functionality

Overview

This malware is accomplished and multifunctional, covering various end processes. The main objective is for the attacker to be able to remotely connect and breach the integrity of the infected system in addition to this system getting added to a botnet, by which commands can be sent via an Internet Relay Chat (IRC).

Capabilities

The specific attacks which can be performed by the botmaster via the IRC include:

- Distributed Denial of Service (**Figure 15a+b**).
- Downloading files from the infected system.
- Executing remote commands (**Figure 13a+b**).
- Opening applications (**Figure 16a+b**).
- Sending messages to the user (**Figure 14a+b**).
- Executing remote screenshots (**Figure 16a+b**).
- Maintains persistence by adding a key to the Windows registry (**Figure 4**).

MITRE ATT&CK Framework

The following table is in respect to the MITRE ATT&CK Framework, a publicly available database of adversarial strategies and tactics derived from firsthand observations. This table details what tactics were used in this malware and the procedure in which they were implemented.

Tactic	MITRE ID	Technique	Procedure within malware
Initial Access	TA0001	Tactics for gaining a foothold into a system.	Malware entered the system by some means, likely by USB stick but not proven.
Execution	TA0002	Adversarial code is executed on the target system.	Malicious actions are performed on the infected system, like those described in the **Capabilities** subsection.
Persistence	TA0003	Maintaining continuous presence within a target system across restarts.	Adds a key into Windows startup registry.
Command and Control	TA0011	Techniques used by attackers to communicate and execute commands on infected systems.	Malware connects to the IRClient to receive commands from the botmaster for the malware to execute.
Collection	TA0009	Techniques used to gather Information useful to the attackers objective.	Screenshots and downloading files from the system could be used to collect information from the target.
Impact	TA0040	Techniques used to disrupt the availability of that system or compromise the integrity of the target.	DDOS attacks could be executed which could affect the availability of the system.
Defence Evasion	TA0005	Techniques used to avoid detection by the target.	The IRClient channel name is changed periodically as an evasion tactic.

(MITRE, 2023)

(Nation Cyber Security Centre, 2023)

4. Dynamic Analysis

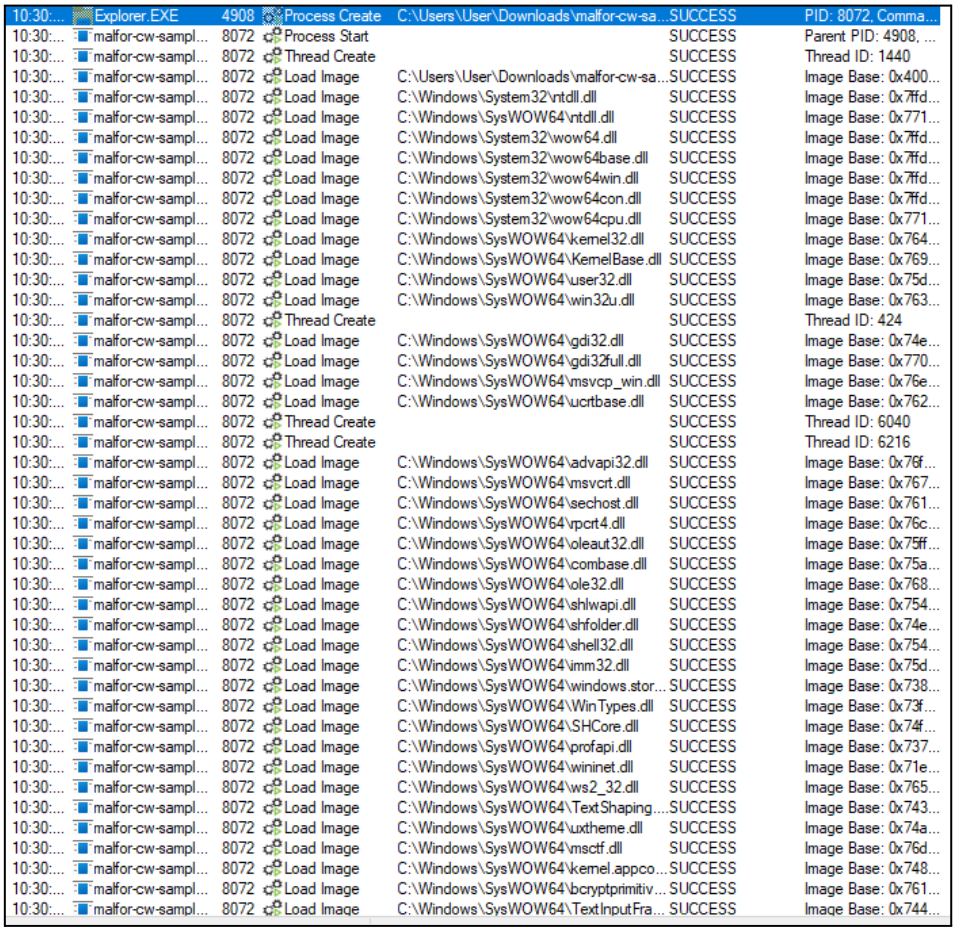
4.1 Process Monitoring

Overview

An integral part of dynamically analysing malware includes monitoring the behaviour of the processes. This malware will be run in a controlled and safe environment, meaning no sequential harm can happen to other systems or devices.

Process Activity

In **Figure 4.11**, we can see the Process and Thread activity linearly after the malware is run. It first creates a process with the PID 8072. This malware's process and thread behaviour after its creation is minimal, as it simply is an accumulation of accessing DLLs and directories necessary for its functionality.



10:30:...	Explorer.EXE	4908	Process Create	C:\Users\User\Downloads\malfor-cw-sa...	SUCCESS	PID: 8072, Comma...
10:30:...	malfor-cw-sampl...	8072	Process Start		SUCCESS	Parent PID: 4908, ...
10:30:...	malfor-cw-sampl...	8072	Thread Create		SUCCESS	Thread ID: 1440
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Users\User\Downloads\malfor-cw-sa...	SUCCESS	Image Base: 0x400...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ffd...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x771...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x7ffd...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\System32\wow64base.dll	SUCCESS	Image Base: 0x7ffd...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x7ffd...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\System32\wow64con.dll	SUCCESS	Image Base: 0x7ffd...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\System32\wow64cpu.dll	SUCCESS	Image Base: 0x771...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Image Base: 0x764...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS	Image Base: 0x769...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\user32.dll	SUCCESS	Image Base: 0x75d...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\win32u.dll	SUCCESS	Image Base: 0x763...
10:30:...	malfor-cw-sampl...	8072	Thread Create		SUCCESS	Thread ID: 424
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\gdi32.dll	SUCCESS	Image Base: 0x74e...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\gdi32full.dll	SUCCESS	Image Base: 0x770...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\msvcp_win.dll	SUCCESS	Image Base: 0x76e...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\ucrtbase.dll	SUCCESS	Image Base: 0x762...
10:30:...	malfor-cw-sampl...	8072	Thread Create		SUCCESS	Thread ID: 6040
10:30:...	malfor-cw-sampl...	8072	Thread Create		SUCCESS	Thread ID: 6216
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\advapi32.dll	SUCCESS	Image Base: 0x76f...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\msvcrt.dll	SUCCESS	Image Base: 0x767...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\sechost.dll	SUCCESS	Image Base: 0x761...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\rpcrt4.dll	SUCCESS	Image Base: 0x76c...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\oleaut32.dll	SUCCESS	Image Base: 0x75ff...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\combase.dll	SUCCESS	Image Base: 0x75a...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\ole32.dll	SUCCESS	Image Base: 0x768...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\shlwapi.dll	SUCCESS	Image Base: 0x754...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\shfolder.dll	SUCCESS	Image Base: 0x74e...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\shell32.dll	SUCCESS	Image Base: 0x754...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\imm32.dll	SUCCESS	Image Base: 0x75d...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\windows.stor...	SUCCESS	Image Base: 0x738...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\WinTypes.dll	SUCCESS	Image Base: 0x73f...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\SHCore.dll	SUCCESS	Image Base: 0x74f...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\profapi.dll	SUCCESS	Image Base: 0x737...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\wininet.dll	SUCCESS	Image Base: 0x71e...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\ws2_32.dll	SUCCESS	Image Base: 0x765...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\TextShaping....	SUCCESS	Image Base: 0x743...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\uxtheme.dll	SUCCESS	Image Base: 0x74a...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\msctf.dll	SUCCESS	Image Base: 0x76d...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\kernel.appco...	SUCCESS	Image Base: 0x748...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\bcryptprimitiv...	SUCCESS	Image Base: 0x761...
10:30:...	malfor-cw-sampl...	8072	Load Image	C:\Windows\SysWOW64\TextInputFra...	SUCCESS	Image Base: 0x744...

Figure 4.11 - Malware Process and Thread Activity within Process Monitor

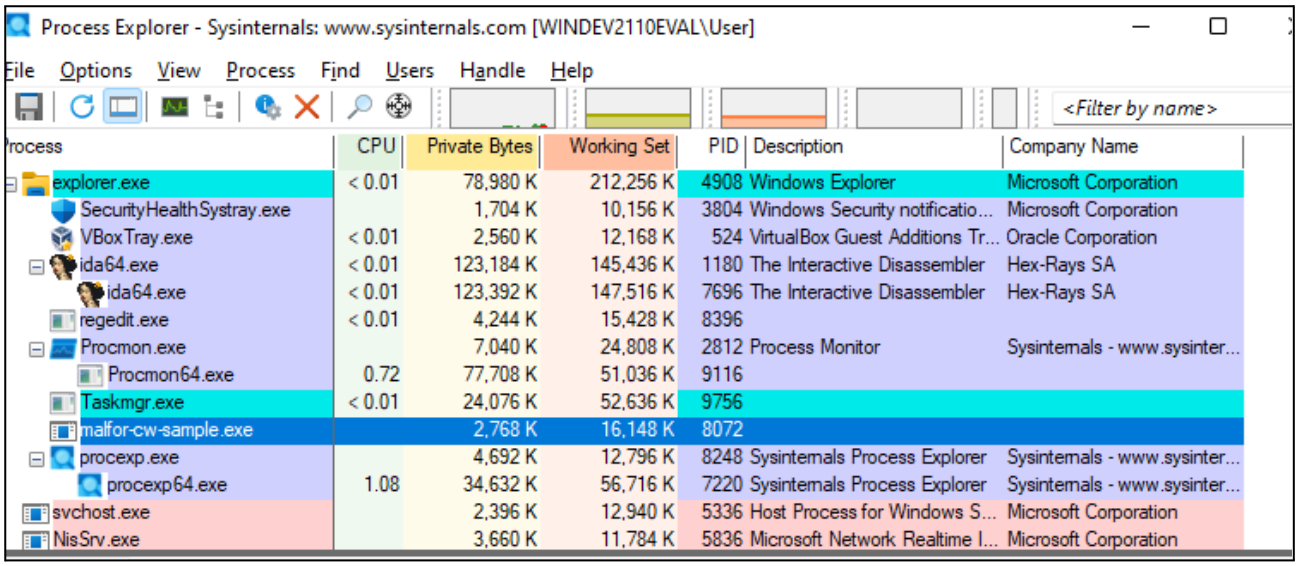


Figure 4.12 - The presence of Malfor's process in Process Explorer

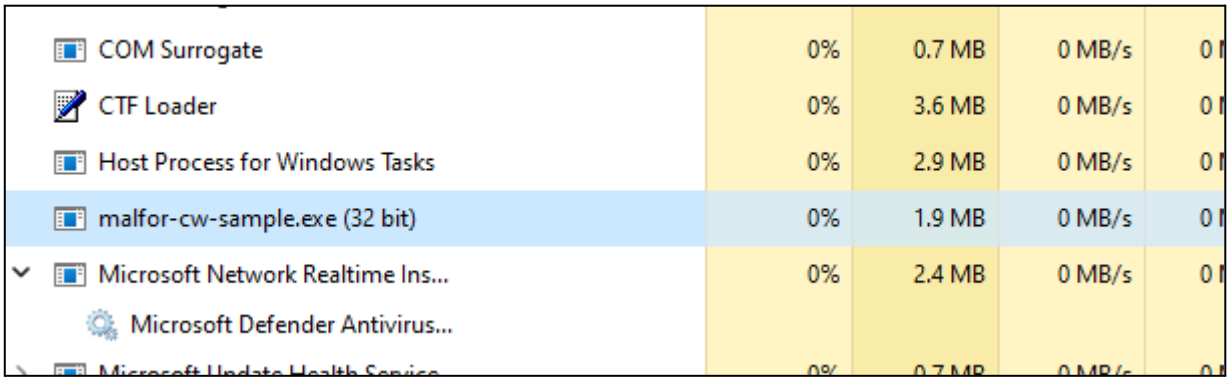


Figure 4.13 - The presence of malfor's process in Task Manager

4.2 Registry Persistence

Overview

Within the Malware Functionality subsection of this report, it was discussed that 'This malware also maintains persistence by adding a key to the Windows registry'. This section will discuss how and why the malware does this.

The Windows registry is a database used to store configuration information and settings. It stores a vast and diverse amount of data.

Persistence is a vital component of creating accomplished malware as it needs to remain active in case of a reboot, in addition it allows an attacker to keep a system compromised without the need to reinfect the system (Perlman, 2020).

Persistence Pathway

As shown in **Figure 4** in the disassembly**,** the malware intends on creating a registry key inside the registry path 'HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run'. This specific path determines what programs run when the Windows Operating System starts. **Figure 4.2** shows a new registry key in the CurrentVersion/Run path, with the data element set to the path of the 'malfor-cw-sample.exe' file.

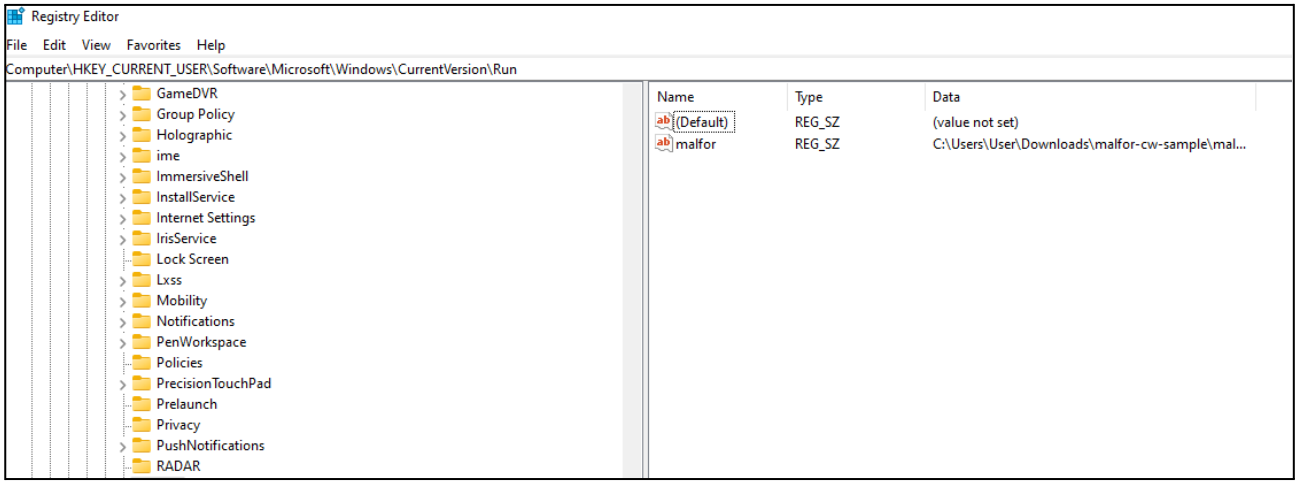


Figure 4.21 - Malfor malware present in Windows registry

4.3 Network Communication

Overview

The main objective of this malware is to add the infected system into a botnet, so by analysing the network communication of this malware after its execution, we can determine how it performs some of its functions more in-depth.

Wireshark

Using wireshark, the first notable activity is a HTTP GET request for the website ‘http://167.99.88.222/cnc.php?id=WinDev2110Eval&uid=User’. As seen in **Figure 4.31**, the website displays the channel name for the IRC botnet.

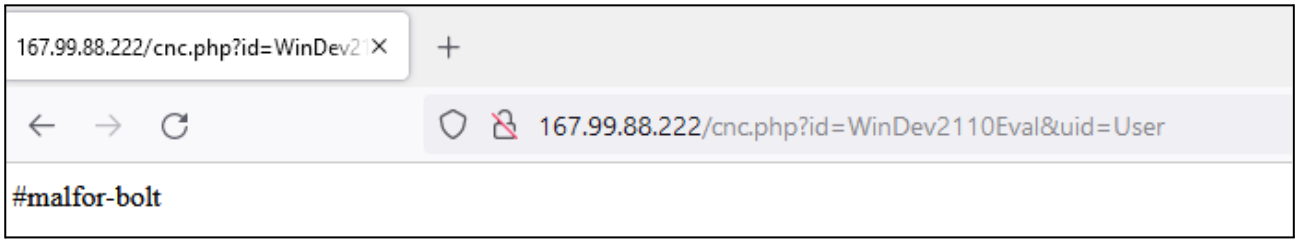


Figure 4.31 - Webpage of http://167.99.88.222/cnc.php?id=WinDev2110Eval&uid=User

This is the URL read from within the **ConnectToServer** function, as seen in **Figure 6** in the disassembly report, as this information is used for the next step in the IRC connection. The reason this is read from a URL and not stored in memory is that the channel name is changed periodically as a possible obfuscation technique by the creator(s).

The next notable activity is the packet that displays the password for the IRC connection: ‘fancybear’.

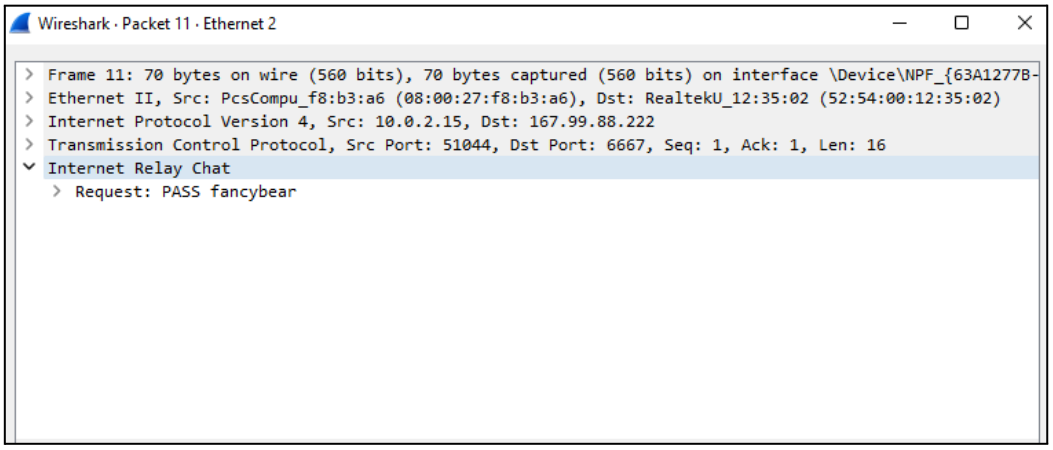


Figure 4.32 - Packet containing the plaintext password for IRC connection

The final notable network activity found by wireshark was the final command used in order to join the specific channel where the botnet activity occurs.

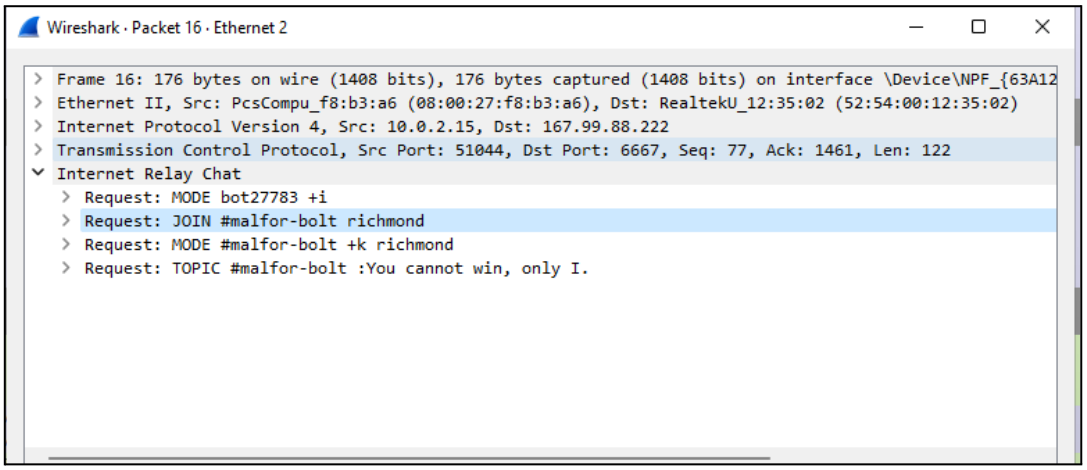


Figure 4.33 - Packet containing the join channel command

4.4 Internet Relay Chat

Overview

IRC is a text-based protocol that allows users to communicate in chat rooms. This IRC chatroom is used as the botnet's botmaster, the user who operates and commands the botnet for execution on a remote target (Radware, 2023).

Infiltrating the botnet

Using HexChat, a tool capable of utilising the IRC protocol, I entered the server's networking information, as seen in **Figure 4.41**.

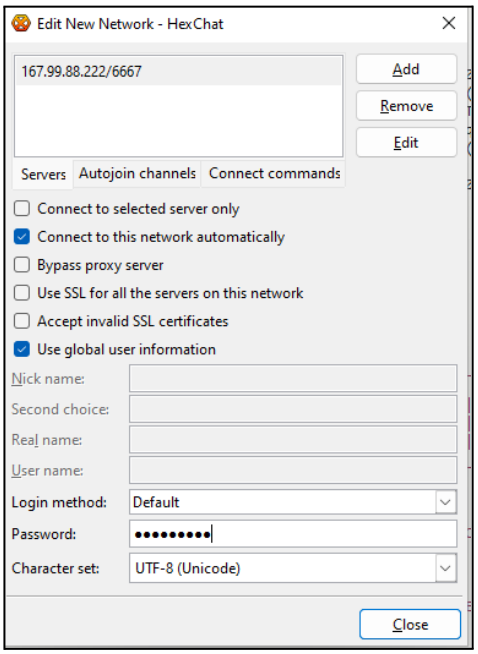


Figure 4.41 - Network Information of botnet

To access the botmaster, the command which was visible in **Figure 4.32** is used. Once inside, a text chat is visible where the commands can be executed, as described previously.

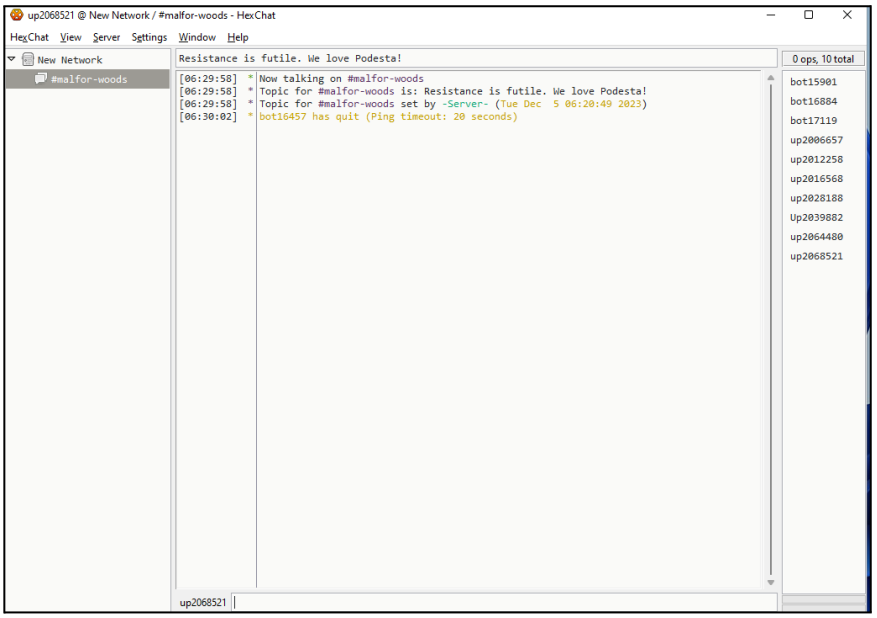


Figure 4.42 - Main screen of the botmaster

Examples of command execution

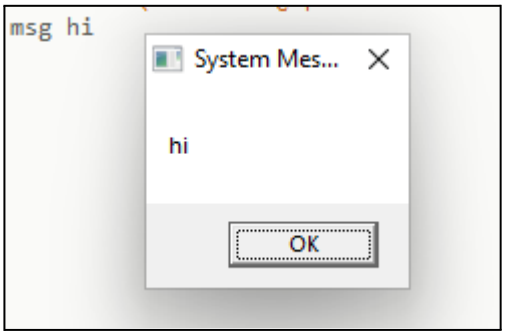


Figure 4.43 - Execution of ‘MSG hi’ command

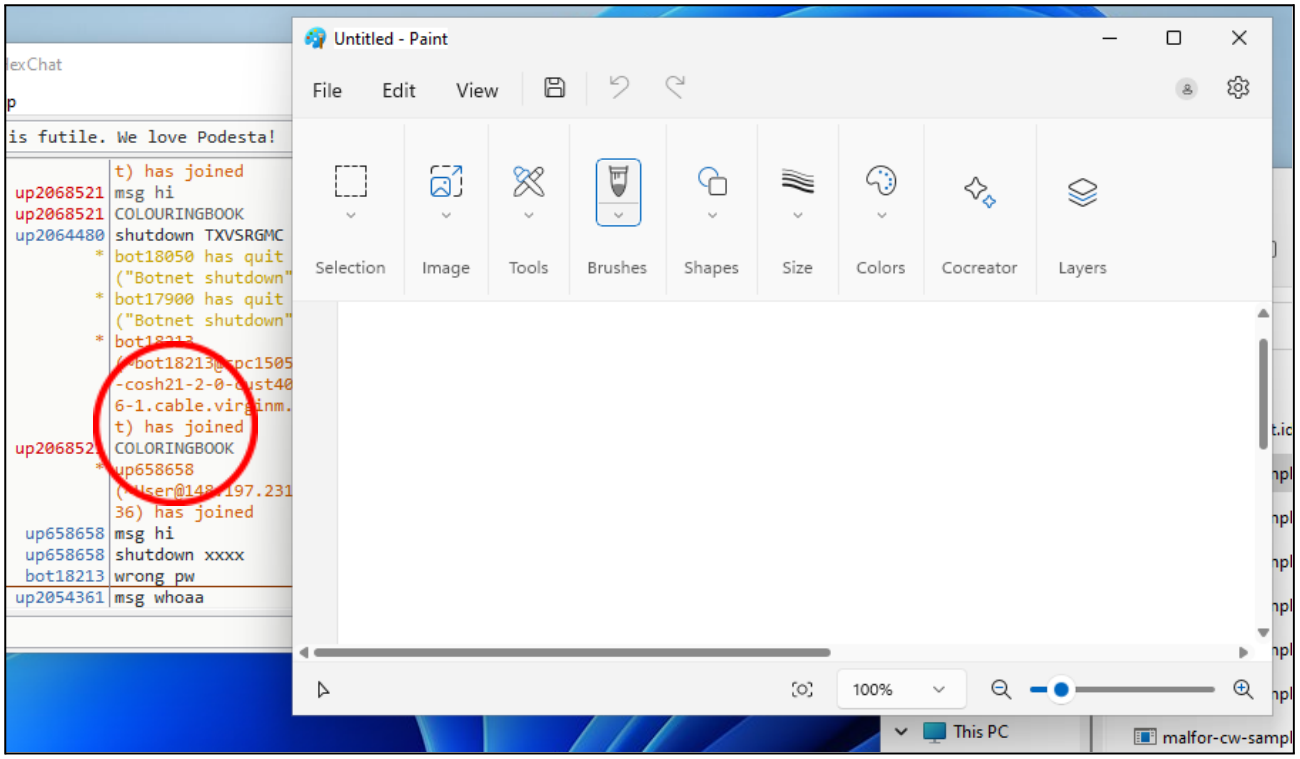


Figure 4.44 - Execution of ‘COLORINGBOOK’ command

5. Reverse Engineering

5.1 Reversing Anti-Debug

Overview

This malware presents some challenges when attempting to analyse and investigate it; specifically Anti-Debugging, a method of ensuring the core fundamentals of a program's functionality are kept protected by not allowing the program to run in a debugger (Head, 2022).

Execution Example

When attempting to run the malware in the IDA software, the following message will appear immediately after execution.

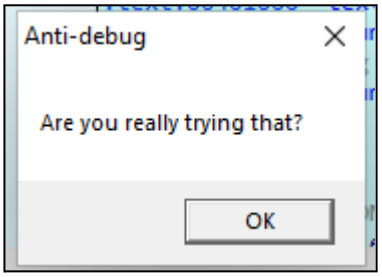


Figure 5.11 - Image showing anti-debugger messagebox

Function Bypass

After the debugger function is called, the program flow goes through a 'jz' instruction; Jump Zero. Meaning if the output of the following instruction is zero, then the program will jump to the location specified. Using a Hex Editor (HxD) it is possible to change the hex code of a 'jz' instruction (74) to another instruction, in this case the JUMP 'jmp' (EB) command. The aftereffect of this is shown in **Figure 5.12**.

Another alternative would have been to use the Jump Not Zero instruction, which has the opposite effect of the Jump Zero command. This would have resulted in the same end result within a debugger as the first method but would have achieved the opposite result when executing the malware outside a debugger.

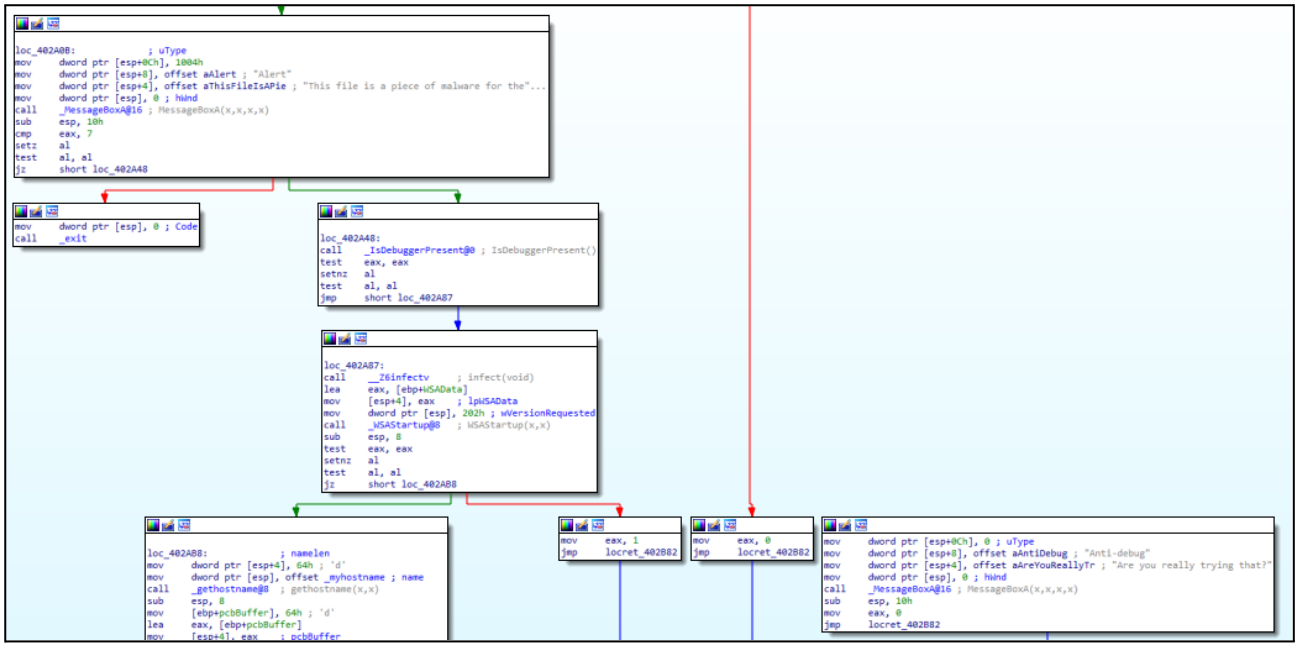


Figure 5.12 - Image showing anti-debugger bypass within IDA

The same function occurs twice within this malware, once just shown and another instance within the **IRClient** function, which is visible in **Figure 7** inside the disassembly report. The same method was executed using the ‘jmp’ command to bypass the function as a whole in this part, and the outcome can be shown in **Figure 5.13**.

Figure 5.13 - Image showing anti-debugger bypass within IDA

5.2 Password Decryption

Overview

Within the malware, there are functions that, when executed correctly, will shut down the botnet. These functions are detailed in **Figures 20, 21** and **22** inside the disassembly. These functions check if the correct password has been entered. The issue is that the password is not stored in memory as plaintext but is instead dynamically unencrypted upon execution.

Malware Function Traversal

The default flow of this malware within a debugger does not go through the previously stated functions; therefore, the program flow must be manipulated to execute these functions. The previous subsection, ‘Reversing Anti-Debug,’ discussed that a ‘jz’ instruction could be changed to a ‘jnz’ instruction to traverse to the alternate route from that function. This is displayed in **Figure 5.21** and **Figure 5.22**.

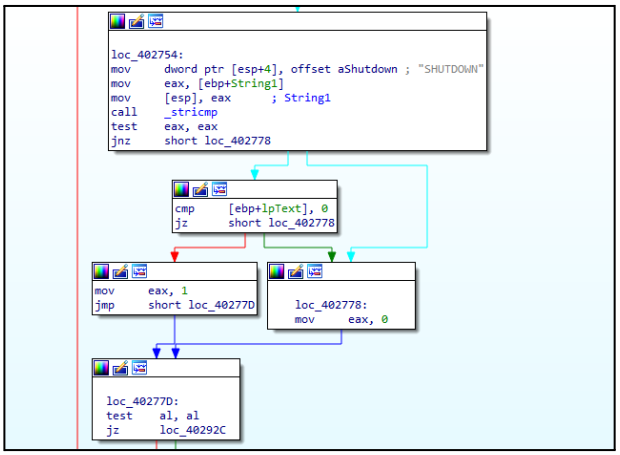


Figure 5.21 - Instruction before modification

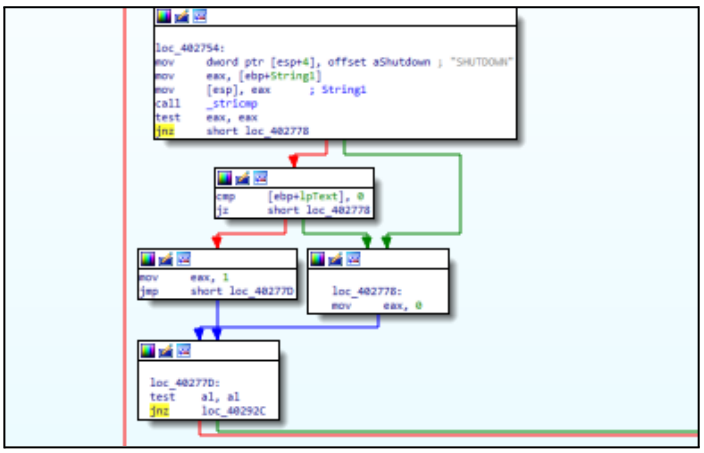


Figure 5.22 - Instruction after modification

Extracting the password

To view the plaintext password, the program will have to be paused at the precise moment where the memory location of the plaintext password is stored in a register; this can be done by using the breakpoint feature in IDA. In **Figure 5.23**, a breakpoint has been added to the instruction where ‘Str2’ is added to the ‘eax’ register. ‘Str2’ was assumed to be the correct password because the ‘Str1’ variable was used in the false condition branch under this function.

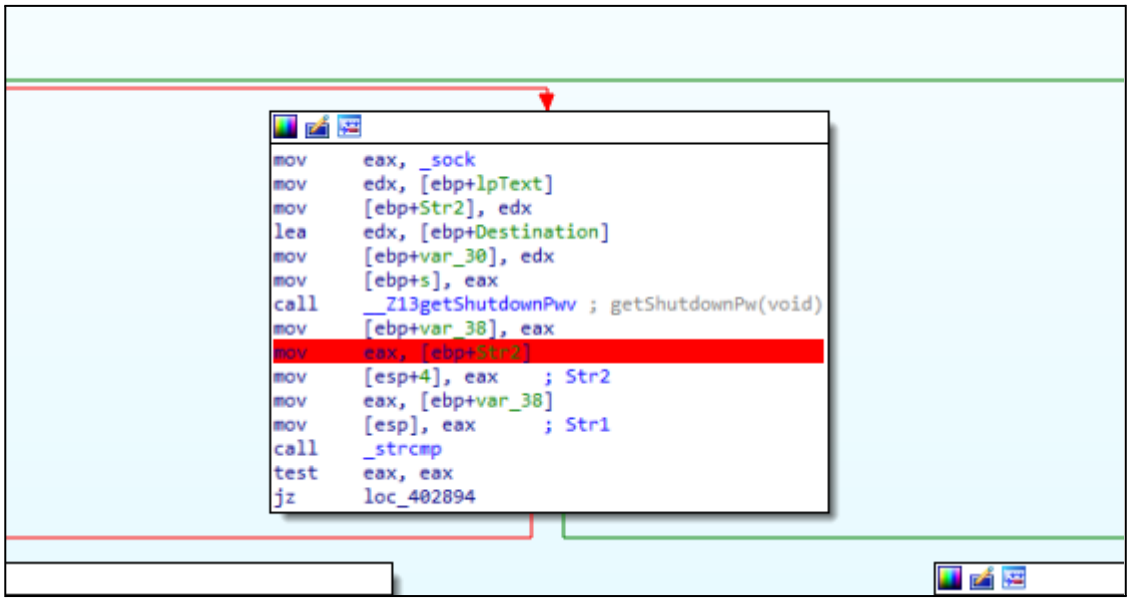


Figure 5.23 - Breakpoint on assembly instruction

After executing the malware and returning to this point in the program's flow, the hex address ‘00A51360’, (this changes every run time), is placed into the ‘eax’ register. Shown in **Figure 5.24**.

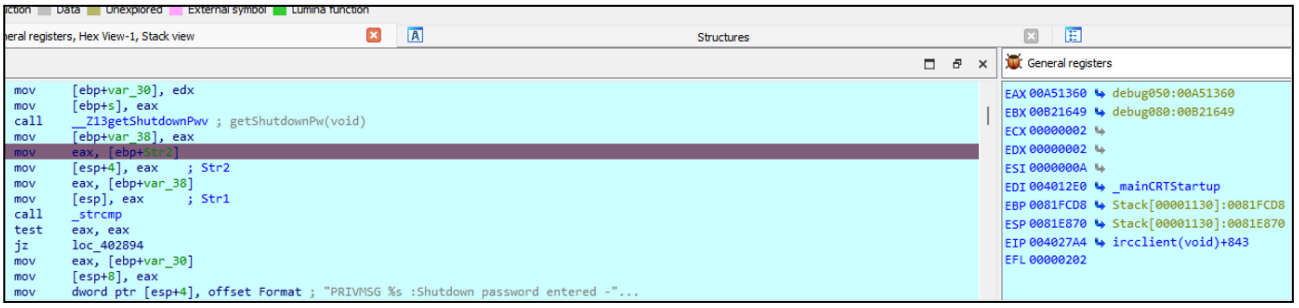


Figure 5.24 - Hex memory location displayed at breakpoint

When going to this location of the memory while the program is still at this breakpoint, the following string is visible ‘TXVSRGMC’, shown in **Figure 2.25.**

```
debug050:00A5135E                db    0
debug050:00A5135F                db  27h ; '
debug050:00A51360                db  54h ; T
debug050:00A51361                db  58h ; X
debug050:00A51362                db  56h ; V
debug050:00A51363                db  53h ; S
debug050:00A51364                db  52h ; R
debug050:00A51365                db  47h ; G
debug050:00A51366                db  4Dh ; M
debug050:00A51367                db  43h ; C
debug050:00A51368                db    0
```

Figure 5.25 - Hex memory location displayed at breakpoint

Using this information, I entered the string alongside ‘SHUTDOWN’ into the IRClient terminal, which resulted in all the bots exiting the IRClient, and a message box popping up on my system. Shown in **Figures 5.26** and **5.27**.

```
up2068521 SHUTDOWN TXVSRGMC
        * bot20141 has quit ("Botnet shutdown")
        * bot16398 has quit ("Botnet shutdown")
        * bot20017 has quit ("Botnet shutdown")
        * bot19808 has quit ("Botnet shutdown")
```

Figure 5.26 - Bots exiting the IRClient after a successful shutdown

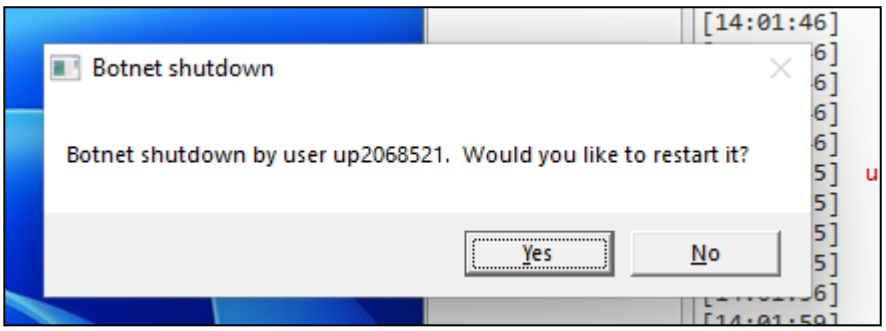


Figure 5.27 - Message box contents after a successful shutdown

6. Origin

6.1 The Equation Group

Overview

Determining the exact developer(s) or creator(s) of this malware is only possible if they claim responsibility publicly. Nonetheless, an informed evaluation can be made using the evidence gathered.

The Equation Group

The sole Internet Protocol (IP) address used within this malware, including that of which the botmaster client is executed by and detailed in **Figure 3.22**, is ‘167.99.88.22’. When accessing that IP address in a search engine, you are greeted by this page.

Figure 6.01 - Message box contents after a successful shutdown

The Equation group is tied to a unit within the United States National Security Agency (NSA). They are described as one of the most sophisticated cyber attack groups ever documented, surpassing anything documented in terms of complexity (Kaspersky, 2021).

Our incident response analysts found this malware to be introduced into the Gazprom System by a rogue employee.

7. Removal

7.1 Malware Deletion Process

To ensure the malware is completely removed from Gazprom's system, three vital steps will have to taken, these are:

1. Shutdown of the current malware process
2. Removal of the registry key
3. Removal of the malware executable file

Shutdown of Process

To shut down the process, Task Manager can be utilised. Right-click on the 'malfor-cw-sample.exe' process name, and click 'end task'. This will stop the malware from running on the system immediately.

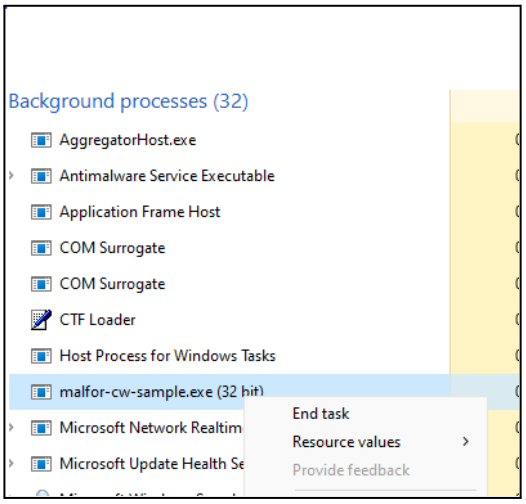


Figure 7.01 - Task Manager view of malware process

Registry Key Removal

To remove the malware's key in the registry, which was done to achieve persistence as detailed in a previous section, the Windows Registry Editor will be used. Traverse to the previously stated path and delete the Malfor key.

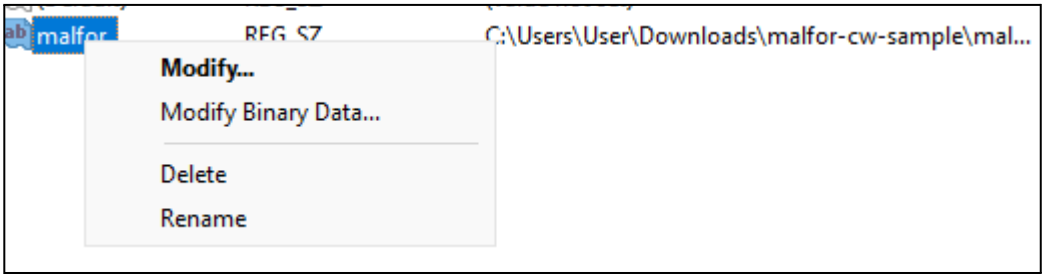


Figure 7.02 - Registry Editor view of malware key

Removal of the executable file

To remove the malware executable file, proceed to the path where the malware was being executed from; this will be visible in the registry key and task manager process but will differ in most cases, so not one standard location can be stated. Highlight the file and use ‘Shift+Delete’ to remove the malware from the system fully.

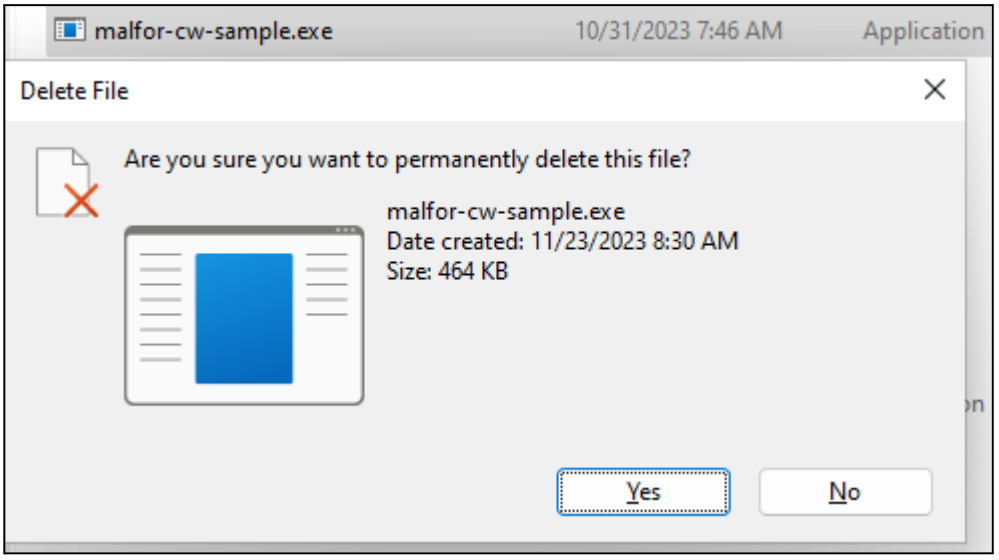


Figure 7.03 - Permanently deleting pop up when the malware

8. Conclusions and Recommendations

8.1 Conclusion

In conclusion, the extensive malware analysis carried out by Longtext Cyber Security has illuminated the malware's significant effects on Gazprom's systems. The research in this report highlights how urgent it is to patch the malware's exploits and put adequate safety controls in place.

Gazprom can considerably improve its cybersecurity posture, protect critical assets, and lower the likelihood of malware attacks in the future through these suggestions, which are detailed in the following table. These recommendations are to prevent and reduce the likelihood of a similar attack occurring. It should be known that this is not a complete vulnerability control table of Gazprom's systems.

8.2 Table of Recommendations

Name of Control / Safeguard	Description
Improve Network Security	● Implement a comprehensive Intrusion Detection System (IDS). ● Update all Network Security Hardware and Software to latest release.
Incident Response Plan (IRP)	● Implement a comprehensive IRP to reduce the impact of any future cyber security event.
USB Port blocking	● Implement a USB port blocking system where there is a complete restriction of external hard drives or USB sticks being entered into the system. ● This lowers the risk of data infiltration and exfiltration.
Security Vulnerability Assessments	● Complete regular third party security vulnerability assessments to assess current weaknesses.

9. References

Freepik. (n.d.-a). *Computer*. Retrieved November 6, 2023, from 32https://www.flaticon.com/free-icons/computer

Freepik. (n.d.-b). *Malware*. Retrieved November 6, 2023, from https://www.flaticon.com/free-icons/malware

Freepik. (n.d.-c). *Server*. Retrieved November 6, 2023, from https://www.flaticon.com/free-icons/server

Head, G. B., Global Business. (2022, October 4). *Anti-Debugging – A Quick Guide to Avoid Malwares and Mobile App Hacks*. AppSealing. https://www.appsealing.com/anti-debugging/#:~:text=One%20of%20the%20common%20methods

kaspersky. (2021, May 26). *Equation Group: The Crown Creator of Cyber-Espionage*. Www.kaspersky.com. https://www.kaspersky.com/about/press-releases/2015_equation-group-the-crown-creator-of-cyber-espionage

MITRE. (2023). *MITRE ATT&CK™*. Mitre.org. https://attack.mitre.org/

Nation Cyber Security Centre. (2023). *Malware Analysis Report Jaguar Tooth Executive summary*.

Perlman, A. (2020, June 11). *The Art of Persistence*. Cynet. https://www.cynet.com/attack-techniques-hands-on/the-art-of-persistence/#:~:text=The%20malware%20needs%20to%20be

Radware. (2023). *Botmaster*. Www.radware.com. https://www.radware.com/security/ddos-knowledge-center/ddospedia/botmaster/#:~:text=A%20botmaster%20is%20a%20person