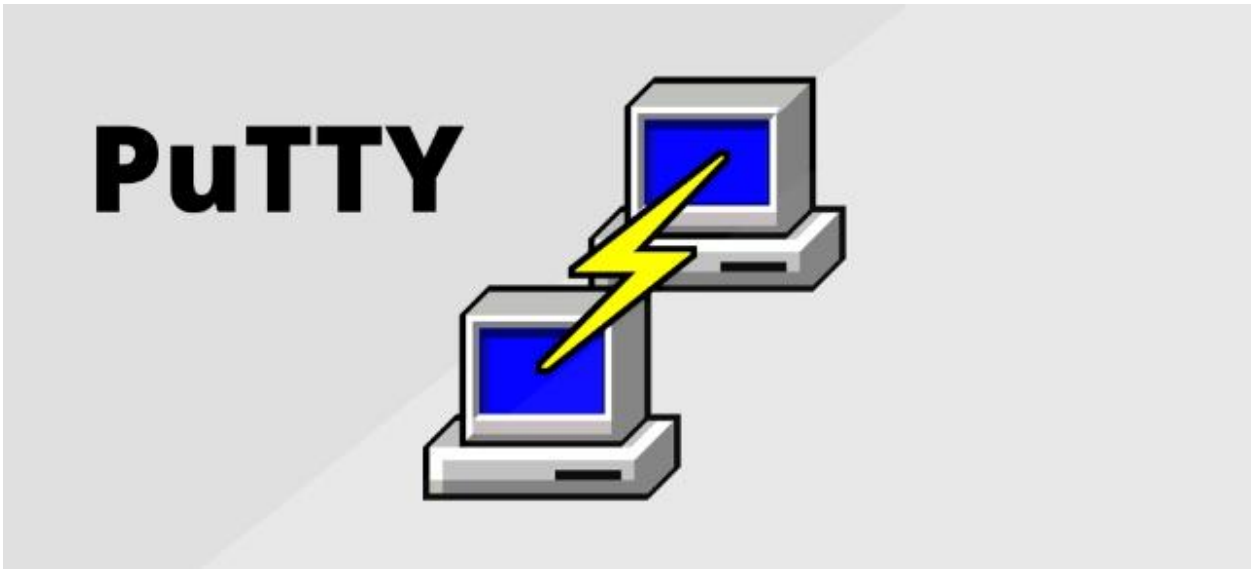
Malware Analysis of a Trojan named “putty.exe”



Contents

Executive Summary

Hi , I Am Don from University of Wollongong bachelor's in computer science major Cybersecurity, and this the first malware in which I have ever analyzed after learning many different concepts and strengthening my foundation just to start learning reverse engineering. It may seem like a simple yet unsophisticated analysis, but it was all done with hours of research and research, and it is my first step into reverse engineering. Special shoutout to TCM security's Practical malware analysis and triage course . It is truly an amazing malware analysis course.

In the course, I was provided with a challenge malware to solve, which is a trojan obfuscated as a legitimate application "putty.exe". I will be documenting the steps I took to conduct a basic manual analysis. As I believe, methodology and fundamental analysis precedes tools. Laying a strong foundation is crucial before turning to decompiling tool such as x64dbg and cutter, which I will be utilizing for other malware later, and I will document its finding too.

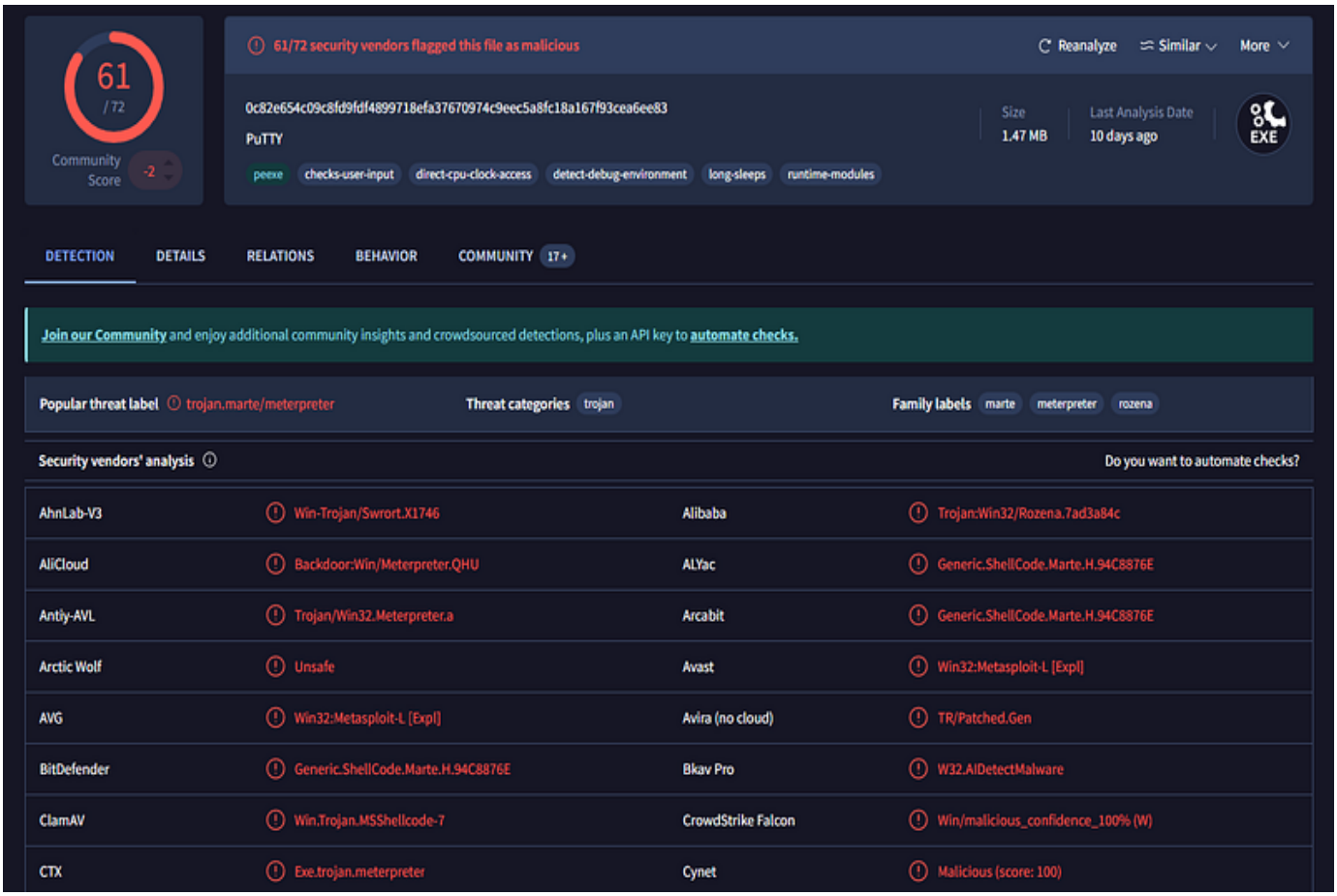
Tools used

1. FlareVM
2. procmon
3. tcpview
4. wireshark
5. netcat
6. inetsim

Static analysis

So, one of the first things i would do was to obtain the signature of putty.exe in SHA256 and MD5 and then submitting the hashes to VirusTotal.com(a known malware databse) for matches.

Figure 1 Hashes



61 vendors have already flagged this as a Trojan malware but let us go under the pretense that it was never there and attempt to analyze its behavior.

Using pestudio

pestudio is a tool which would attempt to identify malicious artifacts within a given executable files(pe format). By using this tool, we can lookout for potential anomalies within the binaries of this executable.

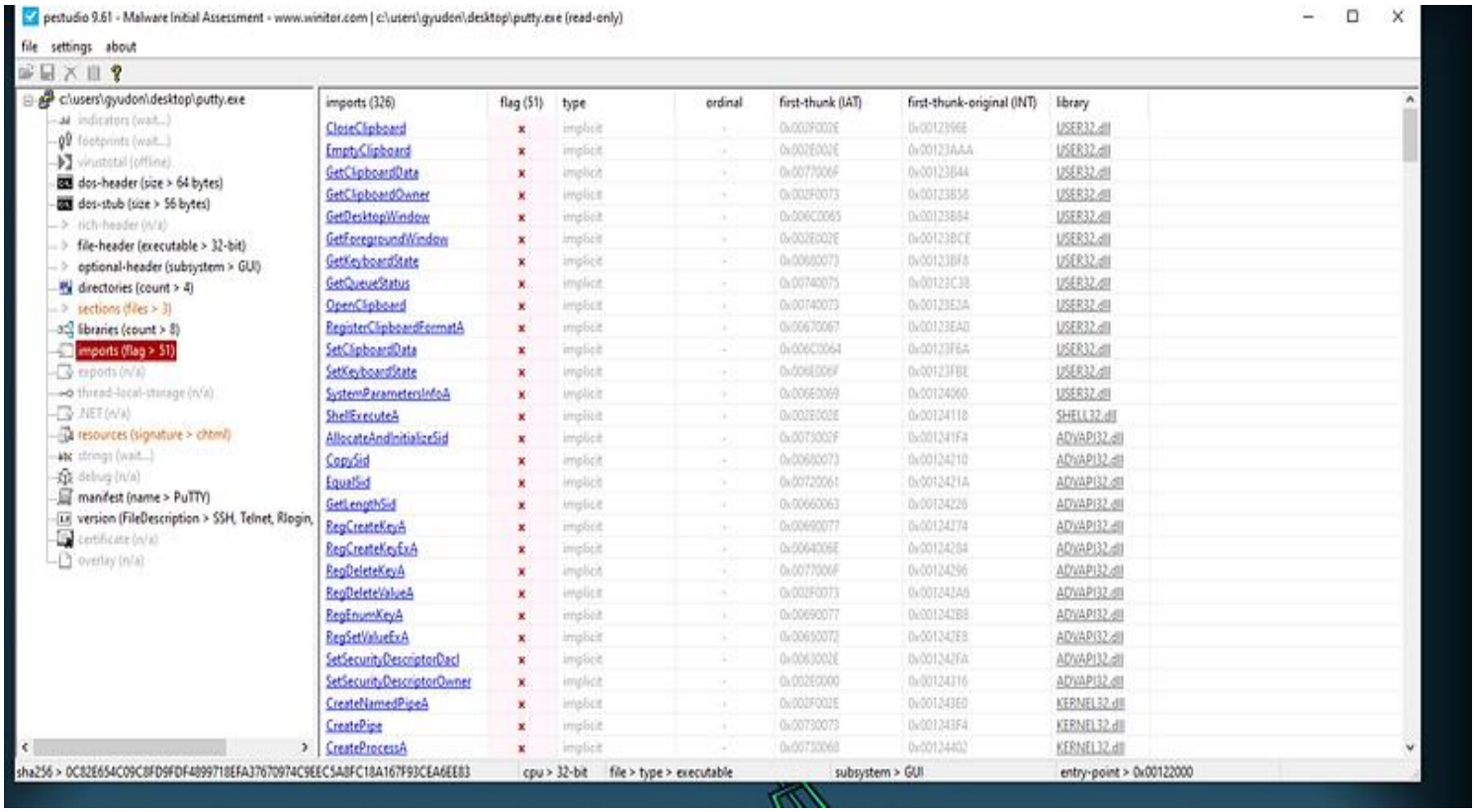


Figure 2 Result from pestudio

These are a list of imported window APIs as well as strings in which pestudio has flagged to be potentially malicious, but then again, for an SSH and Telnet application like putty, these APIs are not out of the ordinary. Even after the static analysis in which we did having no substantial results, i believed I can still work on this further by moving to dynamic analysis.

Dynamic Analysis

By detonating the malware in a safe and controlled environment (flareVM itself) there are tools in which we can utilize to observe its behavior through dynamic analysis. Always remember to take snapshots so that you can always revert back to a safe snapshot.

Figure 3 Powershell spawned oddly

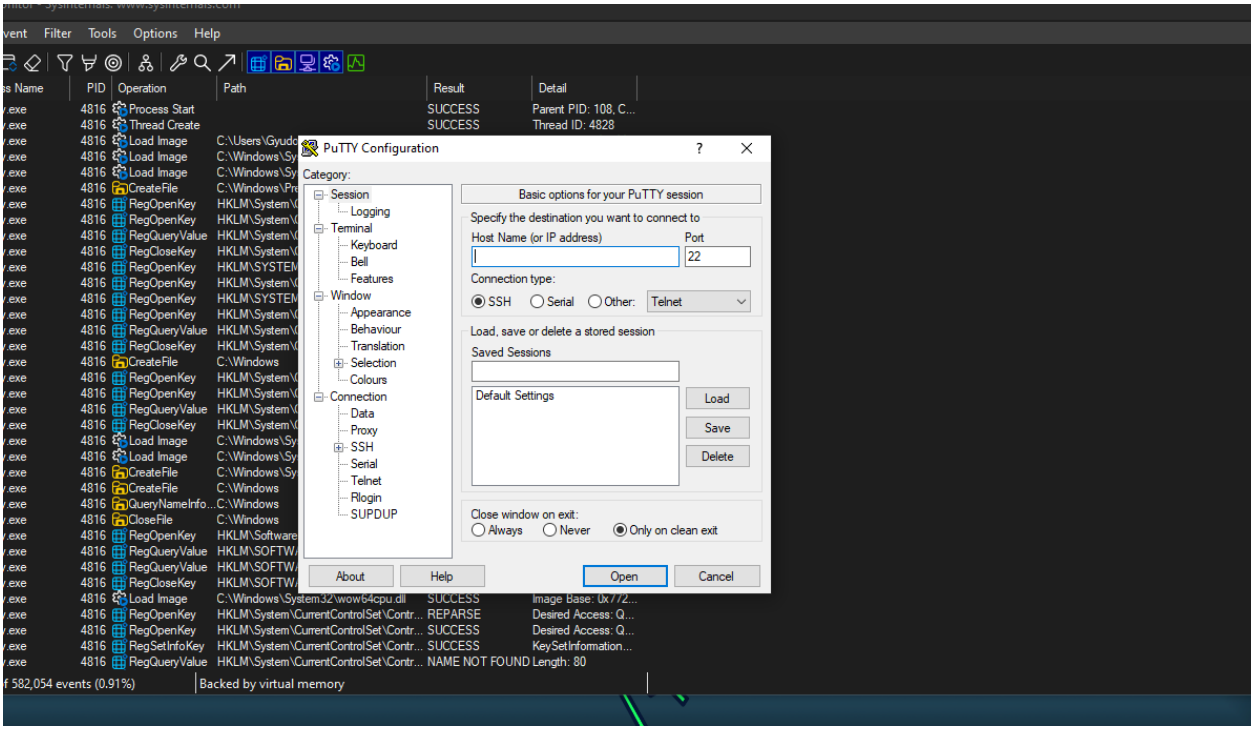
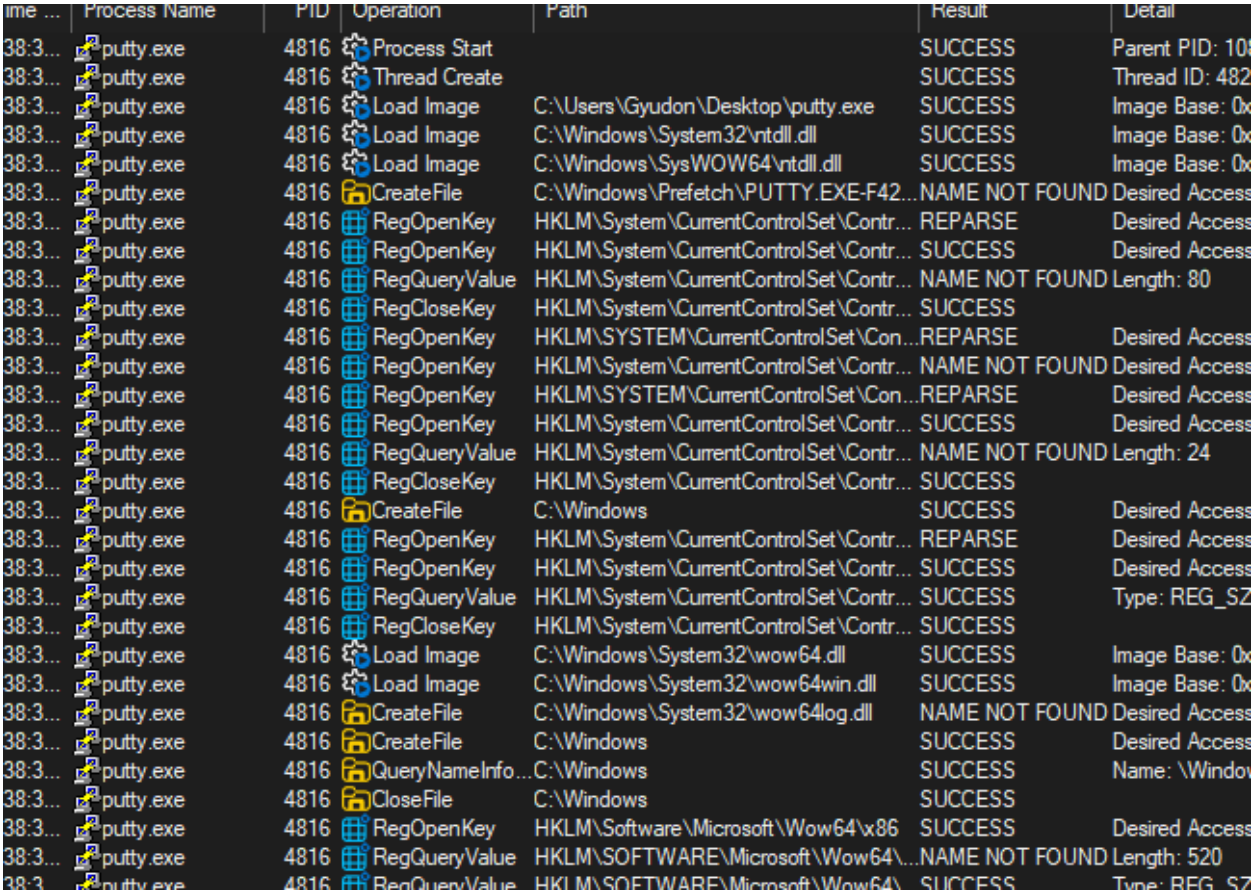


Figure 4 normal putty startup after powershell is terminated

So when I executed putty.exe, it spawned powershell process for a few seconds before it disappeared and the putty.exe executes normally, this is a extremely telling sign of a hidden payload within a legitimate application.

Using procmon to observe processes spawned

So , one thing in which we can do is to use procmon to observe processes spawned by the artifacts in questioned here, which is putty.exe.



Time ...	Process Name	PID	Operation	Path	Result	Detail
38:3...	putty.exe	4816	Process Start		SUCCESS	Parent PID: 10
38:3...	putty.exe	4816	Thread Create		SUCCESS	Thread ID: 482
38:3...	putty.exe	4816	Load Image	C:\Users\Gyudon\Desktop\putty.exe	SUCCESS	Image Base: 0x
38:3...	putty.exe	4816	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x
38:3...	putty.exe	4816	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x
38:3...	putty.exe	4816	CreateFile	C:\Windows\Prefetch\PUTTY.EXE-F42...	NAME NOT FOUND	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access
38:3...	putty.exe	4816	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 80
38:3...	putty.exe	4816	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
38:3...	putty.exe	4816	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Con...	REPARSE	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Con...	REPARSE	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access
38:3...	putty.exe	4816	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	NAME NOT FOUND	Length: 24
38:3...	putty.exe	4816	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
38:3...	putty.exe	4816	CreateFile	C:\Windows	SUCCESS	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	REPARSE	Desired Access
38:3...	putty.exe	4816	RegOpenKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Desired Access
38:3...	putty.exe	4816	RegQueryValue	HKLM\System\CurrentControlSet\Contr...	SUCCESS	Type: REG_SZ
38:3...	putty.exe	4816	RegCloseKey	HKLM\System\CurrentControlSet\Contr...	SUCCESS	
38:3...	putty.exe	4816	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x
38:3...	putty.exe	4816	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x
38:3...	putty.exe	4816	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access
38:3...	putty.exe	4816	CreateFile	C:\Windows	SUCCESS	Desired Access
38:3...	putty.exe	4816	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windo
38:3...	putty.exe	4816	CloseFile	C:\Windows	SUCCESS	
38:3...	putty.exe	4816	RegOpenKey	HKLM\Software\Microsoft\Wow64\x86	SUCCESS	Desired Access
38:3...	putty.exe	4816	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\...	NAME NOT FOUND	Length: 520
38:3...	putty.exe	4816	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\...	SUCCESS	Type: REG_SZ

Figure 5: putty.exe filtered by process name

There is a lot of processes here spawned by putty.exe, but most of them do not seem to be suspicious. However, remember that powershell which spawned? I would then deduced that putty.exe has spawn a powershell child ocess. To confirm my suspicion, I went to filter parent ID of 4816 to see it's child processes.

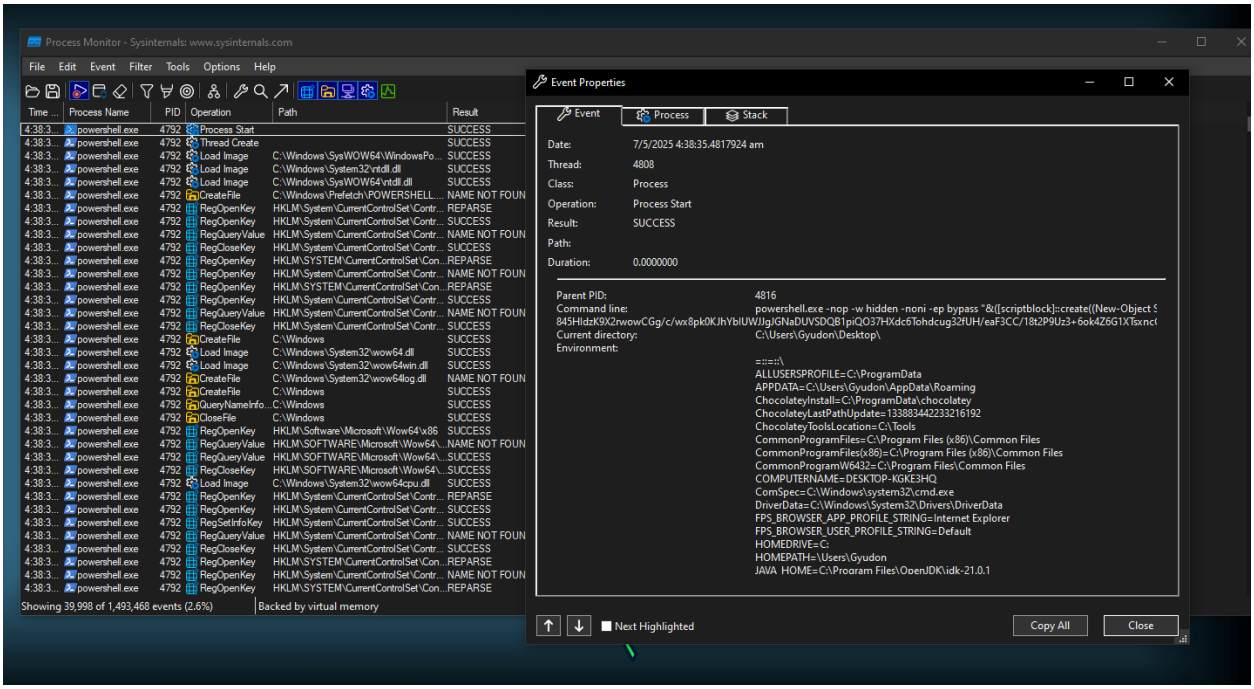
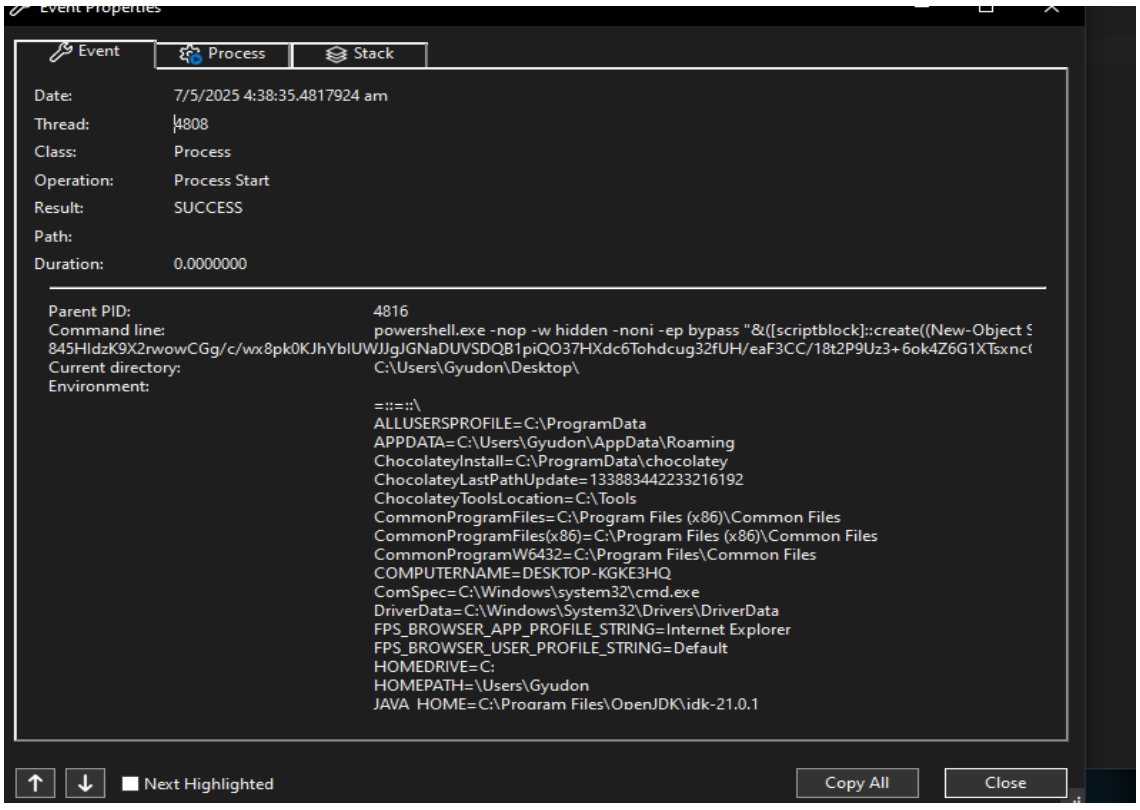


Figure 6:Powershell child process



As suspected, powershell child process is spawned by putty.exe with a suspicious looking powershell command, that has -noni which is non interactive window, and -ep bypass, which bypasses execution policy, to execute a base64 encoded script.

Figure 7: the following script after being word wrapped

Decoding the script and outputting it

The script shows a new object being created in the system with a Gzip format encoded in base64, so what I would do is to decode it and output it using linux to see what is the object/file being created and executed.

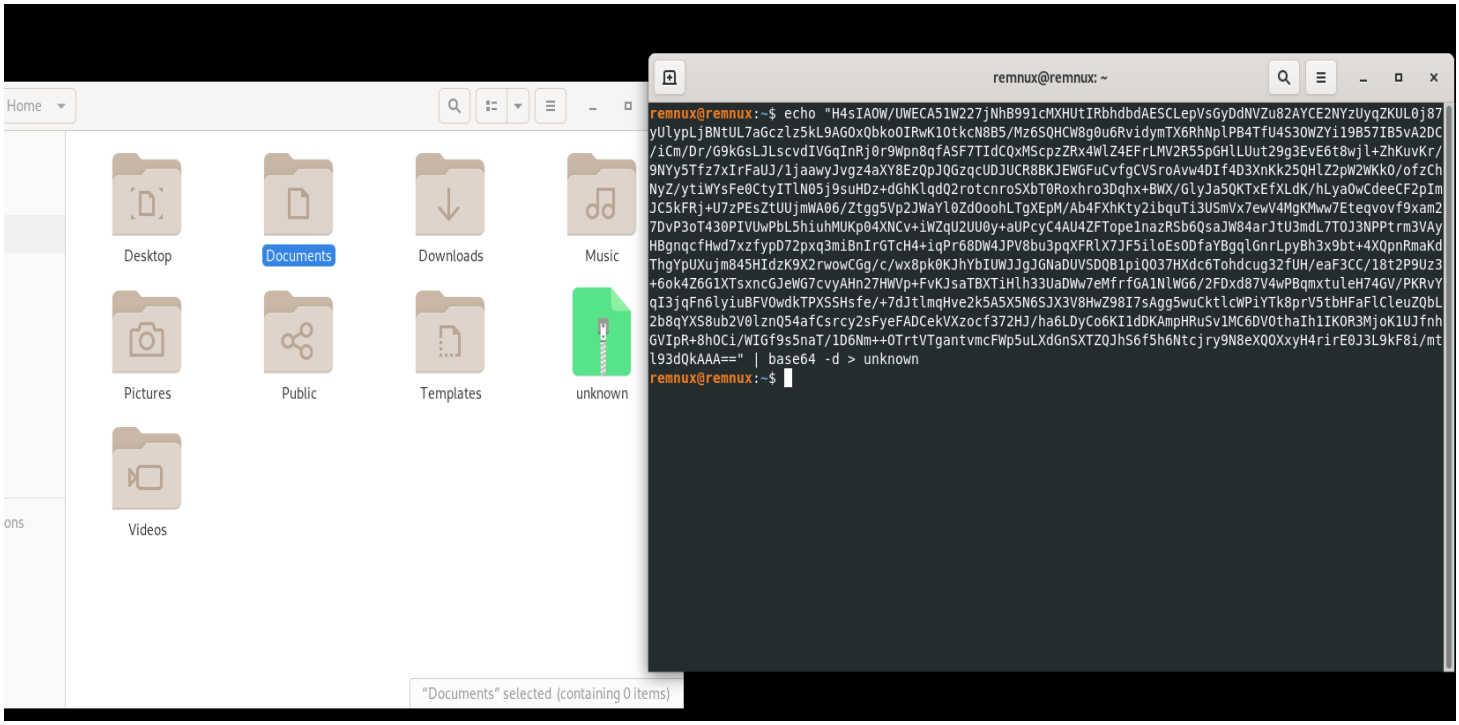


Figure 8:decoding and outputting the file

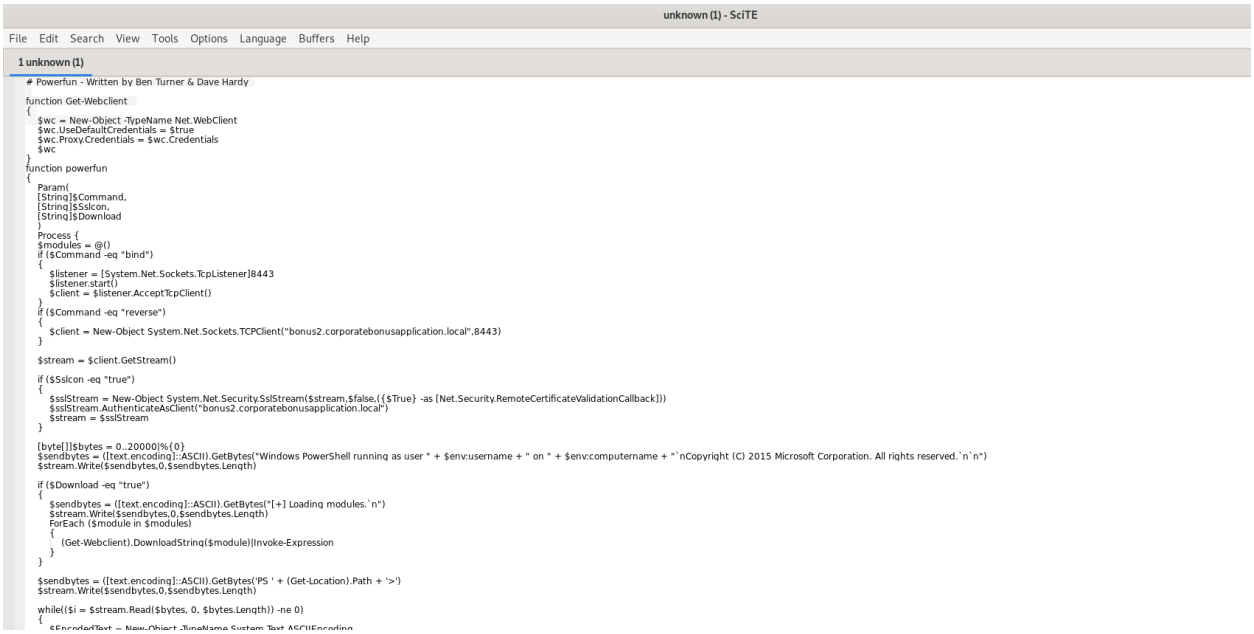


Figure 9:content of the decoded file

As expected, it is a reverse shell script that was hidden in the putty.exe which creates a listening socket on the victim, now I will test out its behavior.

Confirming behavior of the malicious artifact putty.exe

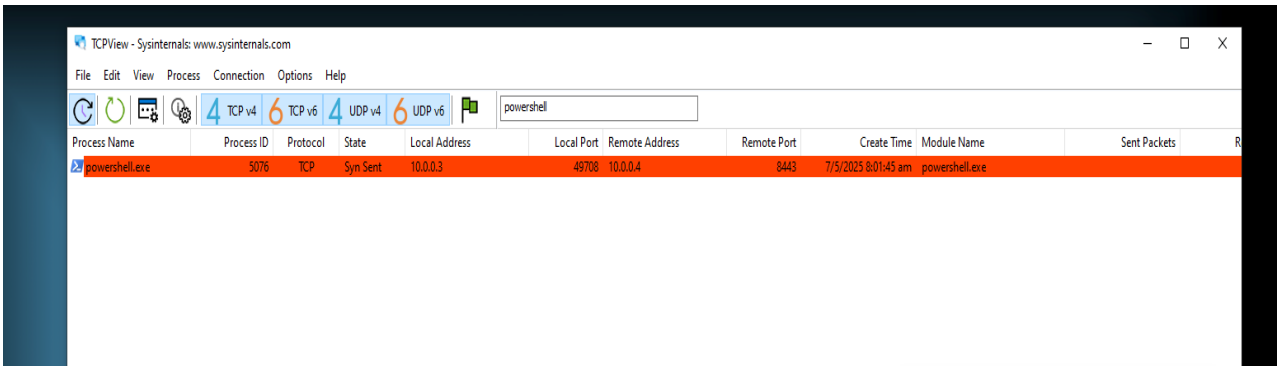


Figure 10: TCP view when putty.exe executes

From the picture above, it seems that the powershell script is attending to reach an remote port of 8443. As regarding the second address, it is an internet simulation setup using inetsim on my second Ubuntu VM, in which I have configured to be my static DNS server.

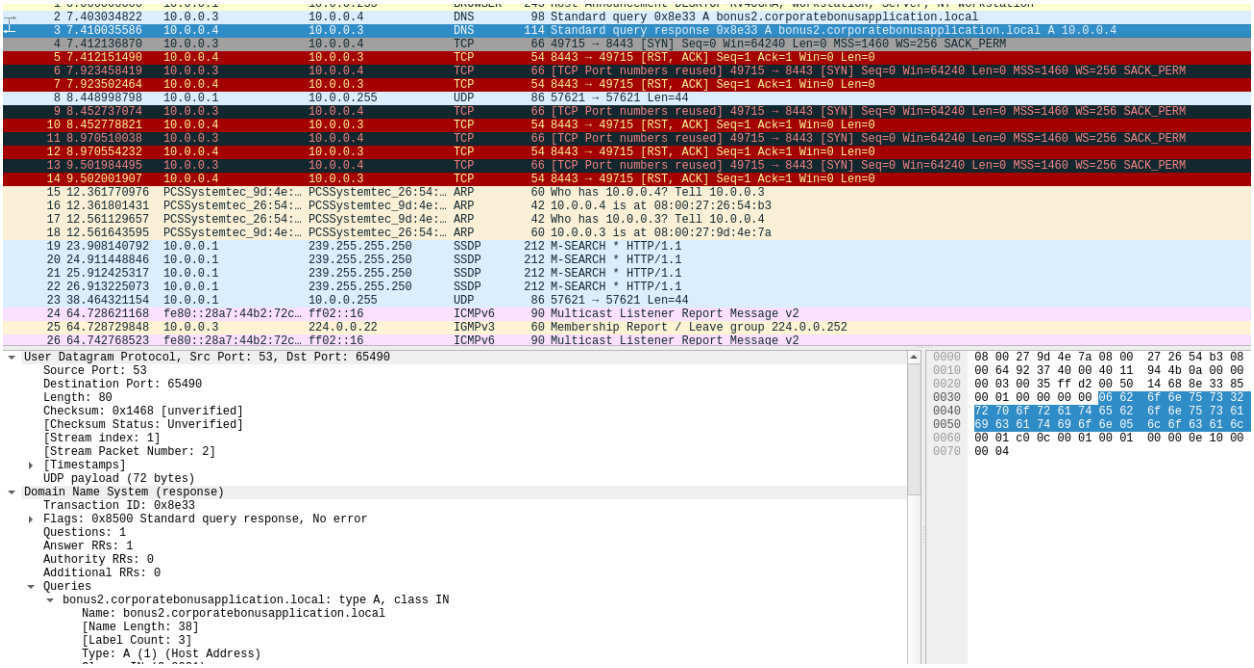


Figure 10 DNS query

As we can see from the picture above, there was an DNS query for bonus2.corporatebonusapplication.local on my inetsim ip address on port 8443.

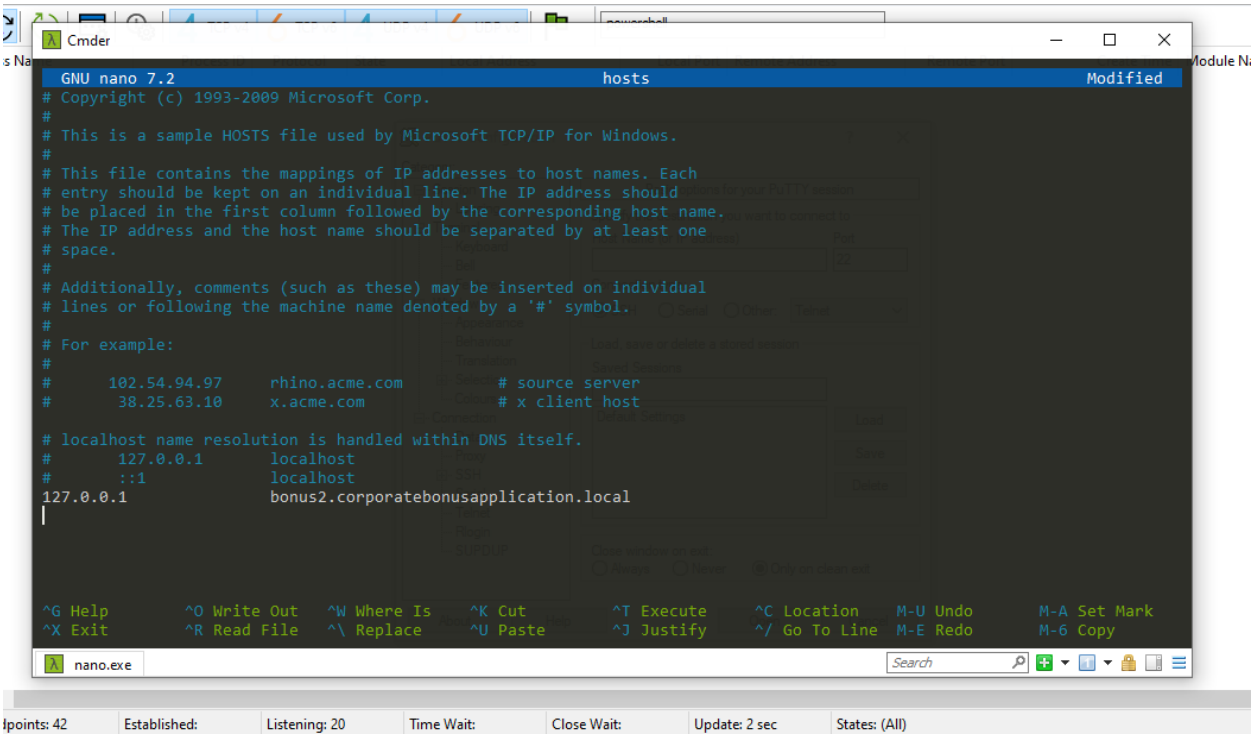
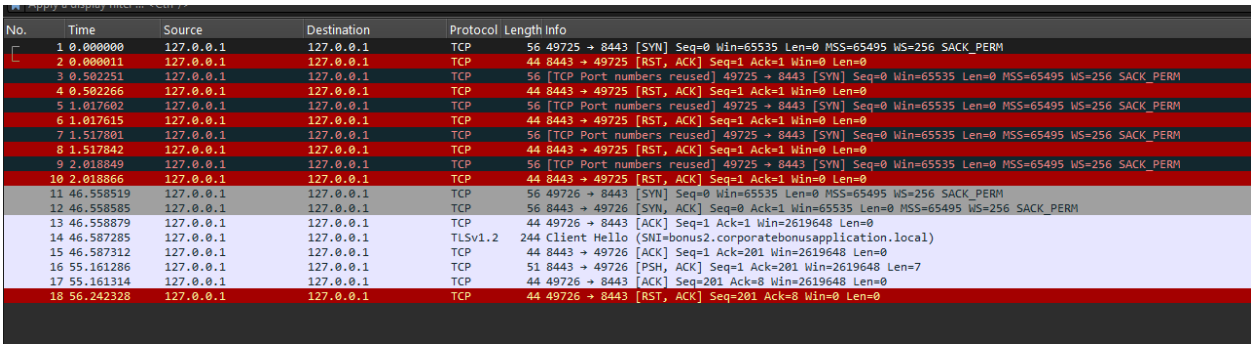


Figure 11 Loopback configuration on the specified domain name

So in order to confirm it's a reverse shell behavior , I configured my local hosts mapping of ip addresses to hosts name to loopback the connection to my own flareVM hosts .



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	56	49725 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
2	0.000011	127.0.0.1	127.0.0.1	TCP	44	8443 → 49725 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3	0.502251	127.0.0.1	127.0.0.1	TCP	56	[TCP Port numbers reused] 49725 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
4	0.502266	127.0.0.1	127.0.0.1	TCP	44	8443 → 49725 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	1.017602	127.0.0.1	127.0.0.1	TCP	56	[TCP Port numbers reused] 49725 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
6	1.017615	127.0.0.1	127.0.0.1	TCP	44	8443 → 49725 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	1.517801	127.0.0.1	127.0.0.1	TCP	56	[TCP Port numbers reused] 49725 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
8	1.517842	127.0.0.1	127.0.0.1	TCP	44	8443 → 49725 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
9	2.018849	127.0.0.1	127.0.0.1	TCP	56	[TCP Port numbers reused] 49725 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
10	2.018866	127.0.0.1	127.0.0.1	TCP	44	8443 → 49725 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	46.558519	127.0.0.1	127.0.0.1	TCP	56	49726 → 8443 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
12	46.558585	127.0.0.1	127.0.0.1	TCP	56	8443 → 49726 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
13	46.558879	127.0.0.1	127.0.0.1	TCP	44	49726 → 8443 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
14	46.587285	127.0.0.1	127.0.0.1	TLSv1.2	244	Client Hello (SNI=bonus2.corporatebonusapplication.local)
15	46.587312	127.0.0.1	127.0.0.1	TCP	44	8443 → 49726 [ACK] Seq=1 Ack=201 Win=2619648 Len=0
16	55.161286	127.0.0.1	127.0.0.1	TCP	51	8443 → 49726 [PSH, ACK] Seq=1 Ack=201 Win=2619648 Len=7
17	55.161314	127.0.0.1	127.0.0.1	TCP	44	49726 → 8443 [ACK] Seq=201 Ack=8 Win=2619648 Len=0
18	56.242328	127.0.0.1	127.0.0.1	TCP	44	49726 → 8443 [RST, ACK] Seq=201 Ack=8 Win=0 Len=0

Figure 12 Wireshark from my flarevm

Wireshark on my flareVM has confirmed that there was indeed a TLS transmission going on port 8443 with that domain name.

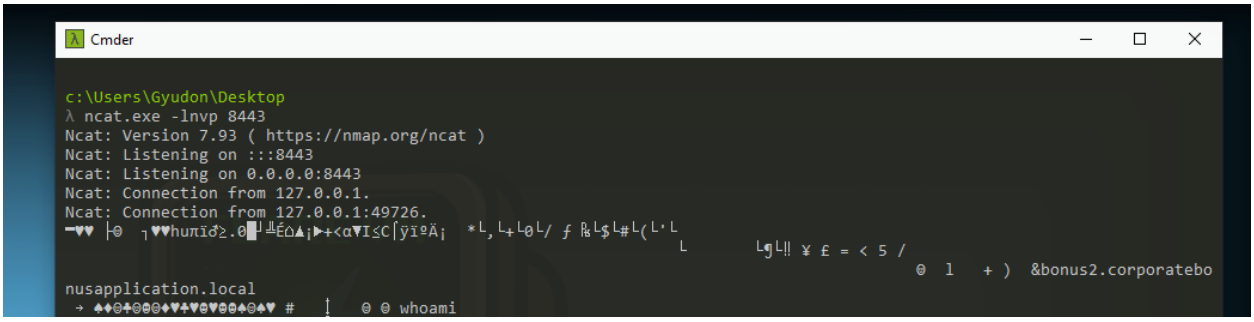


Figure 13:netcat on my own flareVM

I set up a netcat listener on my own flareVM, and I have successfully formed a reverse shell back to my own host, therefore, confirming the reverse shell behavior.

Conclusion

So here goes my documentation regarding my first ever reverse engineering of a Trojan malware, I hope it has been informative, and I will continue to reverse engineer malwares from malware repositories to get better at reverse engineering. Thank you for reading