2025

Logbook

COMP1829 – NETWORK SECURITY

JAKE CUNNINGHAM

Table of Contents

Malware Propagation Research

Sadmind

Sadmind was a sophisticated computer worm that was discovered in May 2001, it was around 7 months old at the time of its discovery and originated from China. Unique for its cross-platform capabilities, it was malware that was able to propagate through Sun Solaris machines to compromise Windows 2000/NT Internet Information Services (IIS) servers. The worm would deface the web services it could find with anti-United States government messages, along with anti-Chinese cracking group PoizonBOx. The worm would then go on to also replace any pages it could find with the same code.

Fig 1 (Sophos, 2001)

Shows what would happen when sadmind worm was activated

Blaster

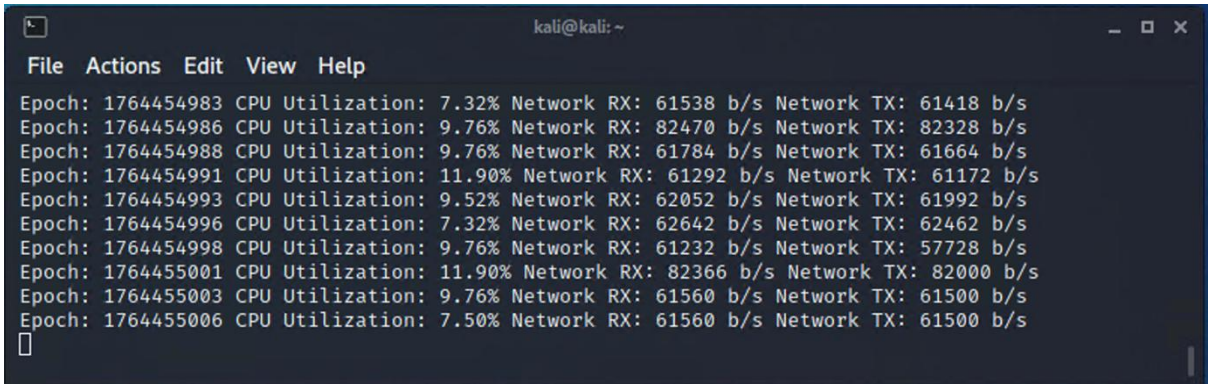
Blaster or Lovsan was also a type of worm that was able to exploit the remote procedure call protocol in Windows 2000 and XP. It was able to connect to TCP port 135 and download msblast.exe in the system32 folder on the machine and amends the registry to ensure it runs on each boot up. It then begins to find other systems on the network to infect, as well as creating a denial of service against windows update, to prevent a patch. Blaster went on to infect hundreds of thousands of machines, causing network slowdowns and crashes. In the early days of the worm it displayed various error messages, such as “I just want to say LOVE YOU SAN!” and “billy gates why do you make this possible? Stop making money and fix your software!”

Conclusion

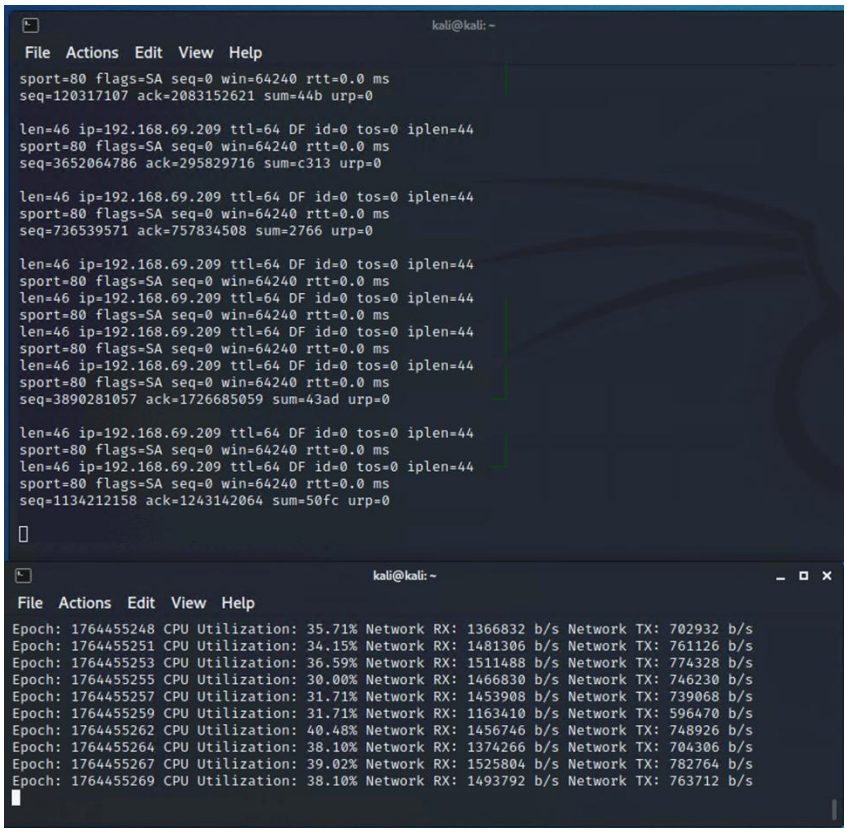
If Malware developers had waited 400 years before releasing, they might have succeeded in their plan, because these ports are frequently used nowadays…

Denial Of Service Impact Analysis

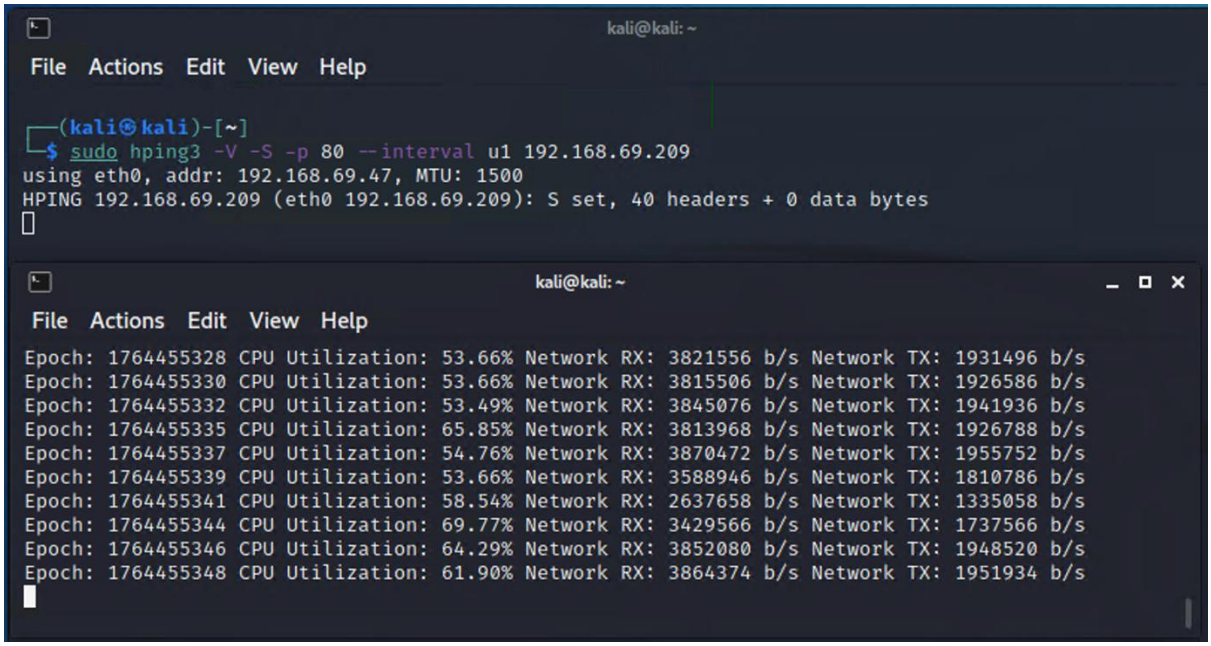
The objective was to evaluate the resilience of a network system against availability attacks, and to see if an attacker can control a victim's resource consumption. The target server was monitored using dos_monitor, which can provide real time feedback on CPU utilisation and network traffic. Standard idle utilisation averages around 9% when no DoS is taking place.

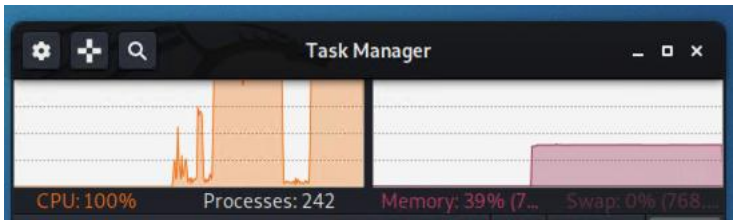


To test whether CPU utilisation can be controlled, to do this hping3 was used, this tool was able to preform a TCP SYN flood by adjusting the --interval, this defines the gap between packets in microseconds, the smaller the number, the more packets that will be sent in a smaller amount of time. Starting with 10 microseconds, which caused the CPU to run at an average of 40% utilisation



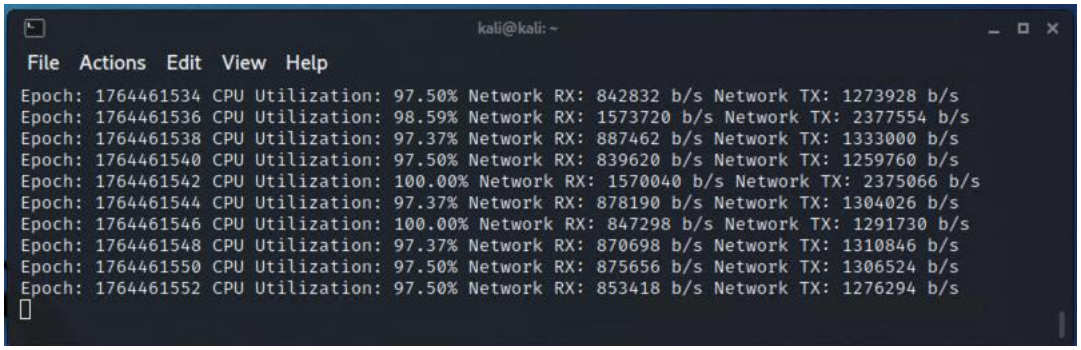
Reducing the interval to 1 microsecond caused the cpu utilisation to spike to around 60% sometimes jumping as high as 70%, this however is the highest utilisation that can be achieved from one attacker as the attacking machine also requires a high cpu usuage to send the packets.



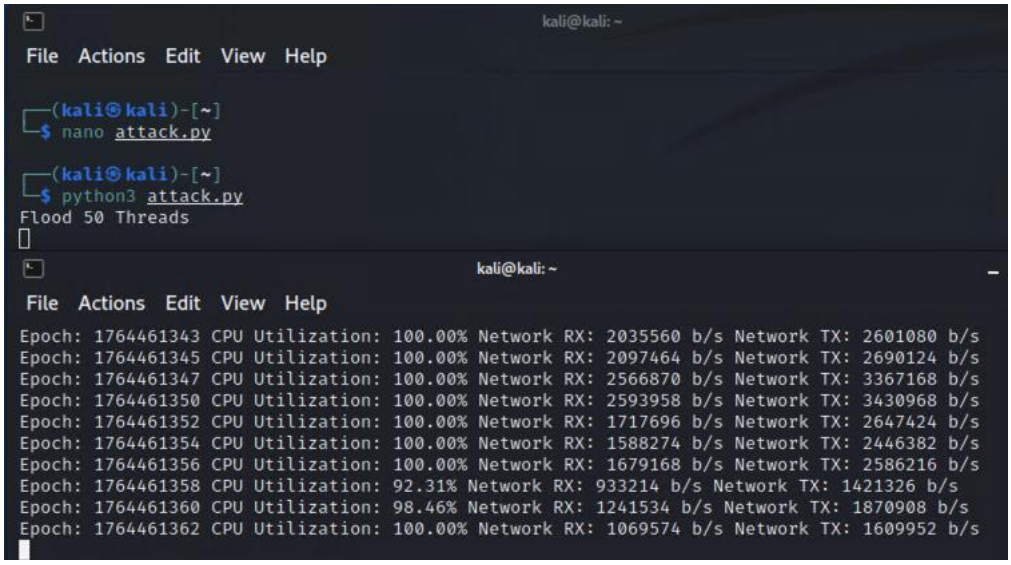


High CPU Usage on attackers machine.

When two attackers are sending DoS signals simultaniously, this is called a DDoS or a distributed denial of service, because the attacks are distributed on multiple machines this causes the CPU usuage of the victim to hit 100%. Resulting in all services on the attacked server to slow down or stop completely. Rendering this system unresponsive to users.



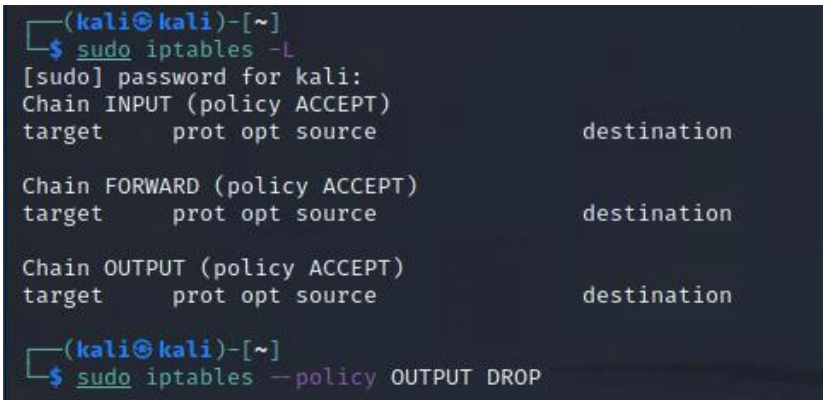
Next compared to a SYN flood where thousands of packets are required to be sent each second to saturate the CPU, a HTTP flood requires significantly fewer requests to achive 100% utilisation. This is because a SYN flood stresses the kernal space where as a HTTP flood stresses the user space.



Firewall Configuration

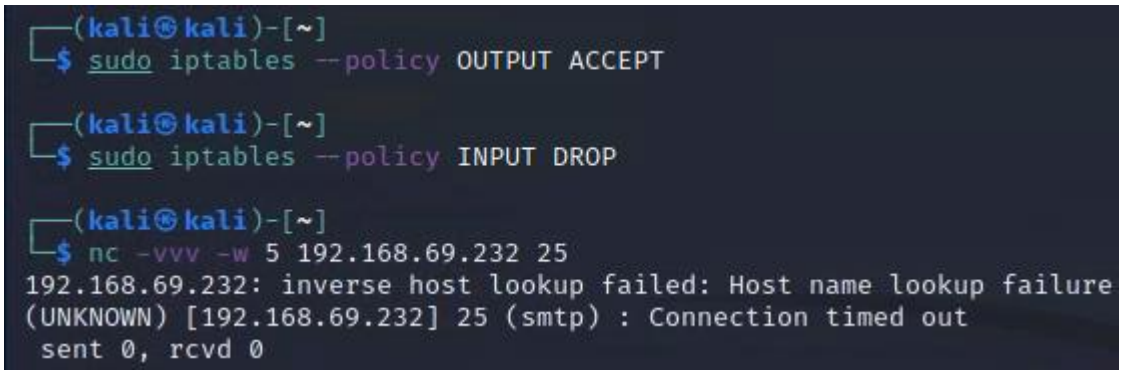
In this section, iptables was utilised which is a command line firewall utility for implementing traffic filtering rules. The goal was to configure the firewall from a default Policy of ACCEPT to a default policy of DENY. This is to significantly reduce an attack by making sure that only authorised traffic can leave or enter the workstation.

Firstly to view the policy “sudo iptables -L”, this shows the default policy for all chains (INPUT, FORWARD, OUTPUT) is ACCEPT meaning there could be a security risk in the event of a user unknowingly executes malware. To prevent this “sudo iptables --policy OUTPUT DROP” is ran. This means that any traffic going out will be dropped.



Then, to allow authorised traffic out the command “sudo iptables -A OUTPUT -p tcp --dport [port number] -j ACCEPT” is used, this is run 3 times. Port 80 for http, port 25 for smtp email and port 8888 for server communication. Listing the iptables again shows that OUTPUT is set to default DROP for INPUT and OUTPUT however listed below are the ports that are set to allow OUTPUT.

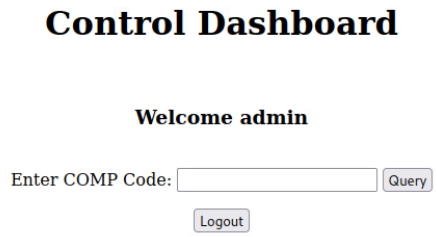
Finally, setting the configuration OUTPUT to ACCEPT and the INPUT to DROP an attempt to connect the web server on 192.168.69.232 was attempted, this connection timed out and failed even though the OUTPUT policy is set to ACCEPT. This is because while the SYN packet is allowed to be sent out, the server replies with the SYN-Ack, acknowledgement packet. Since this packet is received from the server and all INPUT are set to DROP, the SYN-ACK packet is subsequently dropped breaking the by the workstation, breaking the TCP handshake and resulting in a “connection timed out” message.



Web Application Security Laboratory

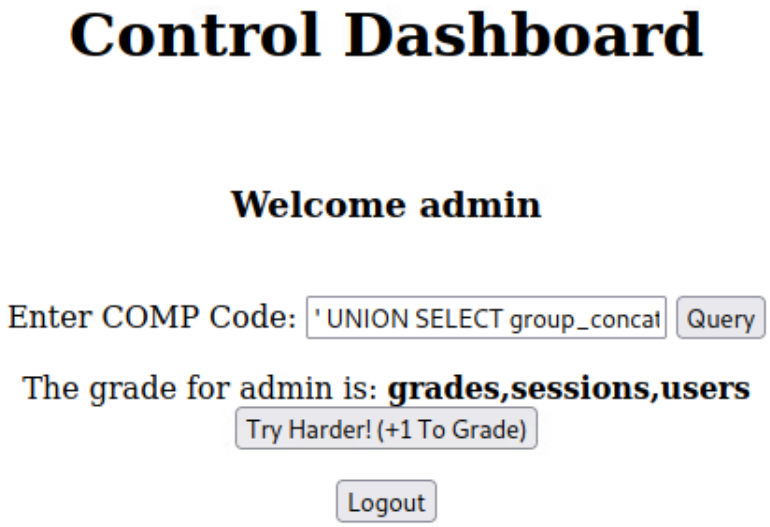
Targeting a web application on http://192.168.69.228, this server hosts a grade management system. It was found to be vulnerable to SQL injections. This means that with some specific commands in SQL query boxes, information about the database can be returned.

Firstly, to bypass the log in screen “ ' OR 1=1 # “was used, this command logged me into the first user on the database without a valid password, this user happened to be the admin account.

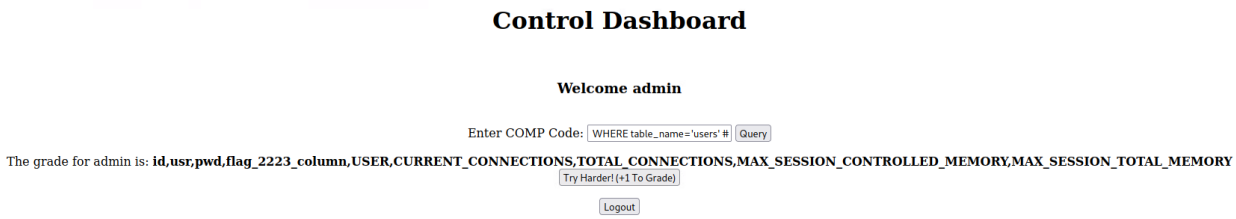


To understand the structure of the database, the UNION based SQL injection was utilised. By injecting “’ORDER BY {1, 2, 3…} #” where each time incrementing the number by 1 until the page errored. This happened after 4, meaning the current table has 3 columns.

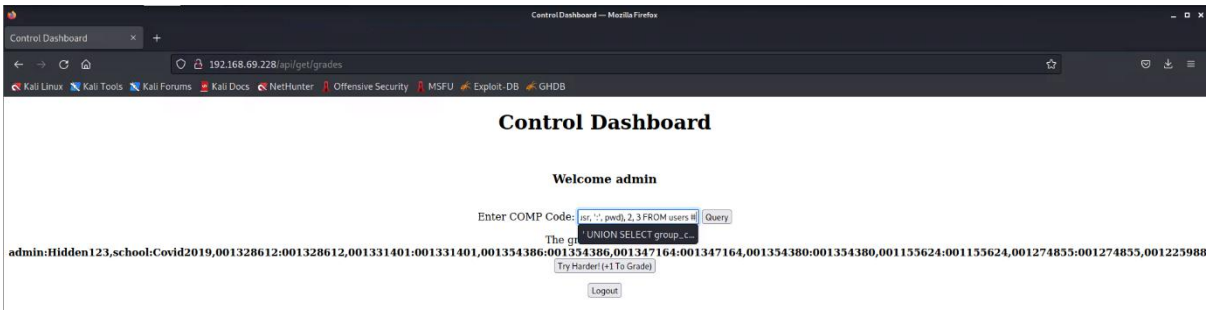
With this information, I was able to then find out the name of the tables in the database, using the “group_concat()” injection. Injecting this into the SQL query “' UNION SELECT group_concat(table_name), 2, 3 FROM information_schema.tables WHERE table_schema=database() #” The result was the names of the tables, which are grades, sessions, and Users.



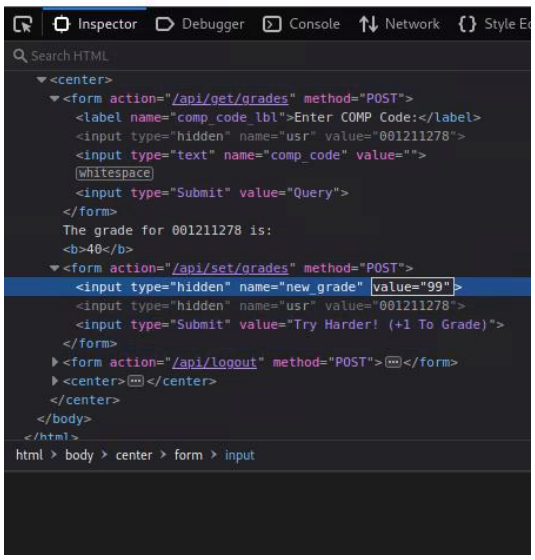
Looking into the user's database, when information_schema is queried, the columns of the table are listed, showing there are usr, pwd and id columns among others, the usr and pwd are the important ones.



With this knowledge, a further injection payload was able to dump the contents of the users table, specifically in usr and pwd. "' UNION SELECT group_concat(usr, ':', pwd), 2, 3 FROM users #" Since the database was stored in plain text, the contents were displayed in the same place as the previous results. The dump revealed the usernames and passwords for all the accounts in the database, namely the admin which has a password of "Hidden123"



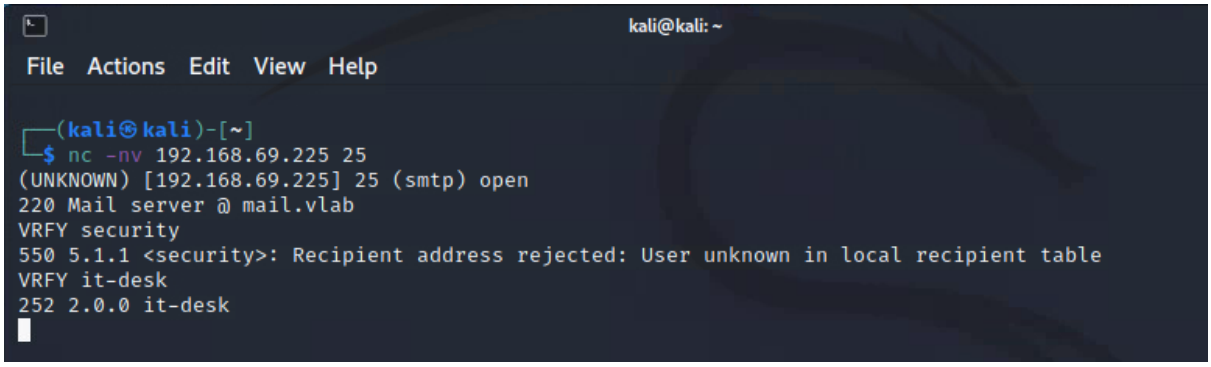
Finally, logging in as myself under 001211278, I was able to modify the HTML code of the website and amend the value of the "+1 To Grade" button, meaning that when pressed, the new grade became 99



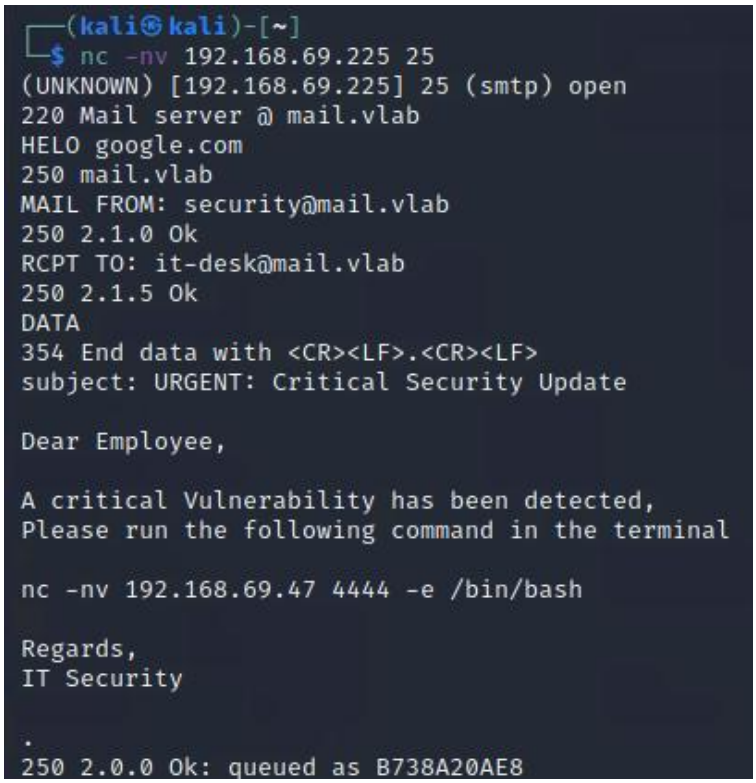
Investigation and Spear Phishing Attack

After shoulder surfing and noticing that the reception uses Outlook within ubuntu, this opens a vulnerability for a possible attacker to take advantage.

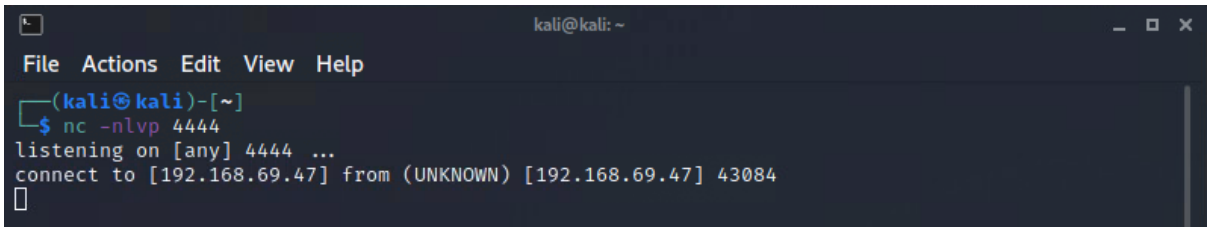
Firstly, the “VRFY” command was used to find out what emails on the system were valid, security returned with “Recipient address rejected” this mean there was no valid email address on the server with security, Then “it-desk” was verified, which was knows to be valid, this returned 252, meaning it was a valid email on the server.



Using a social engineering method, where employing urgency to scare people into acting fast, A spoof email was crafted and sent to the it-desk@mail.vlab email. The email contained a command, like that of the one spied on the victim’s computer already. The email was spoofed from the security@mail.vlab address, despite this not existing on the mail server. The email contained an instruction to enter a command that would execute netcat on the victim’s machine.



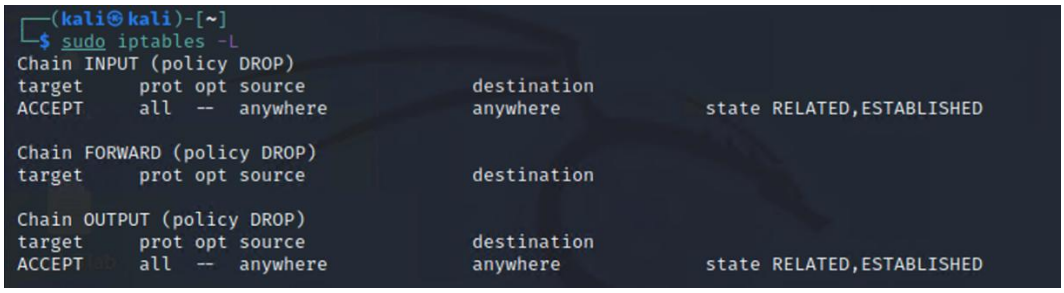
Then once the victim ran the script, a port listener that was running detects the open port and can connect to the victim's computer, granting shell access to the attacker.

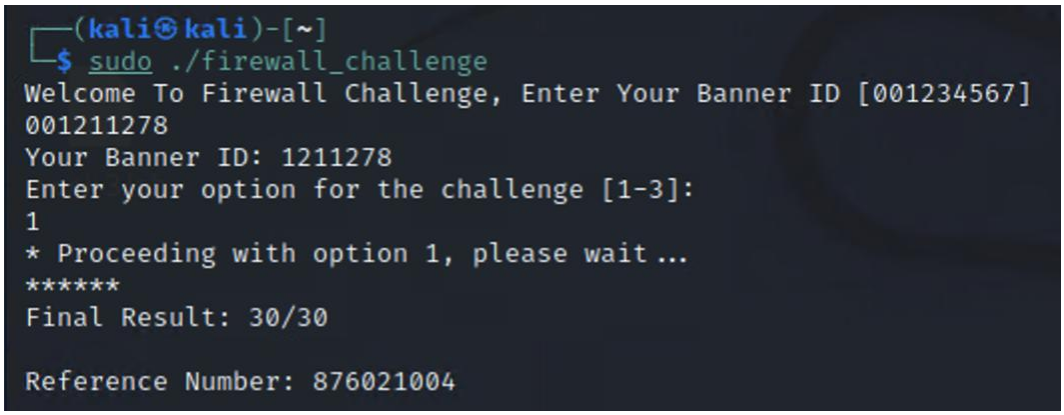


This attack is possible, due to the lack of email source validation and insufficient security awareness. To mitigate these attacks, an organisation must implement email authentication protocols and train staff to verify instructions before executing anything into a terminal.

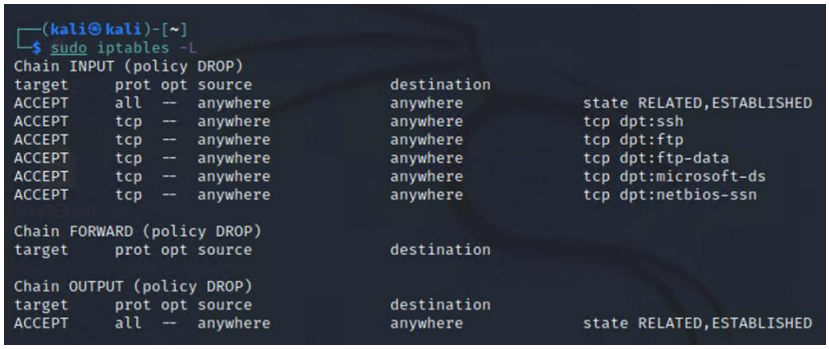
Firewall Use-Case Challenge

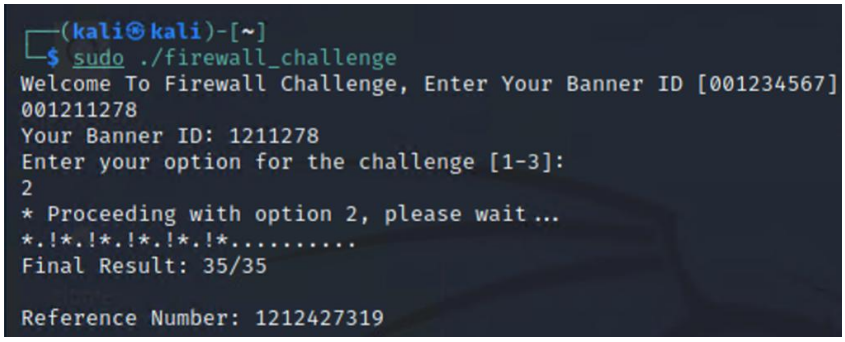
Use case 1is a workstation that is used to browse the web using HTTP and HTTPS, sometimes the occasional SSH will be used to connect to remote workstations in addition to SMTP and POP3 for emails. All traffic that is not this will be subsequently dropped by the firewall



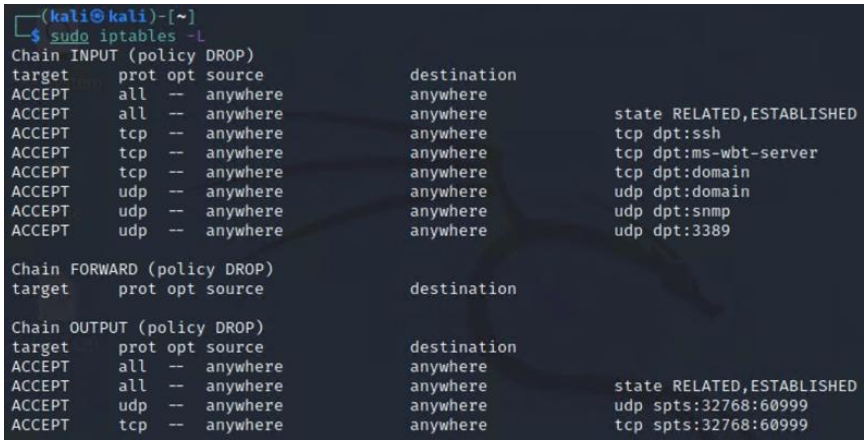


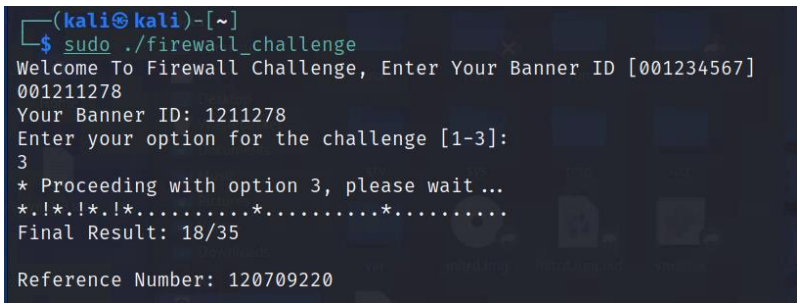
Use case 2 is a server that will provide SSH, FTP SMB and NetBIOS. It will have multiple users connecting remotely via SSH. Any other traffic will be dropped, including incoming.





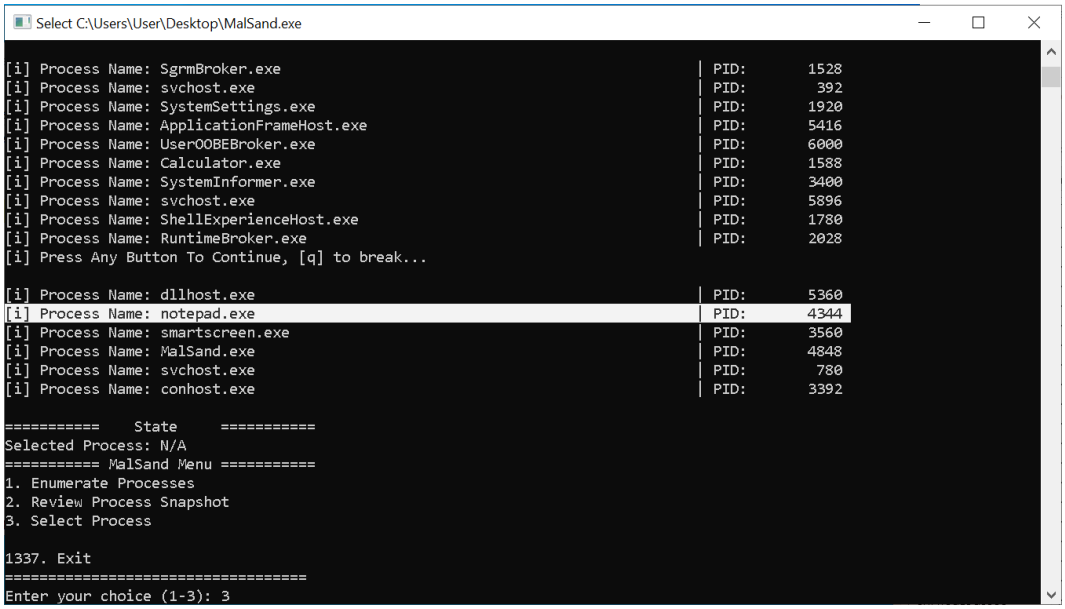
Use case 3 is a prototype domain controller, a combination of a workstation and a server. This means it will provide SSH and RDP as well as DNS and SNMP. Because this is a domain controller, users will need to be able to send traffic via a dynamic port. All other traffic is dropped.



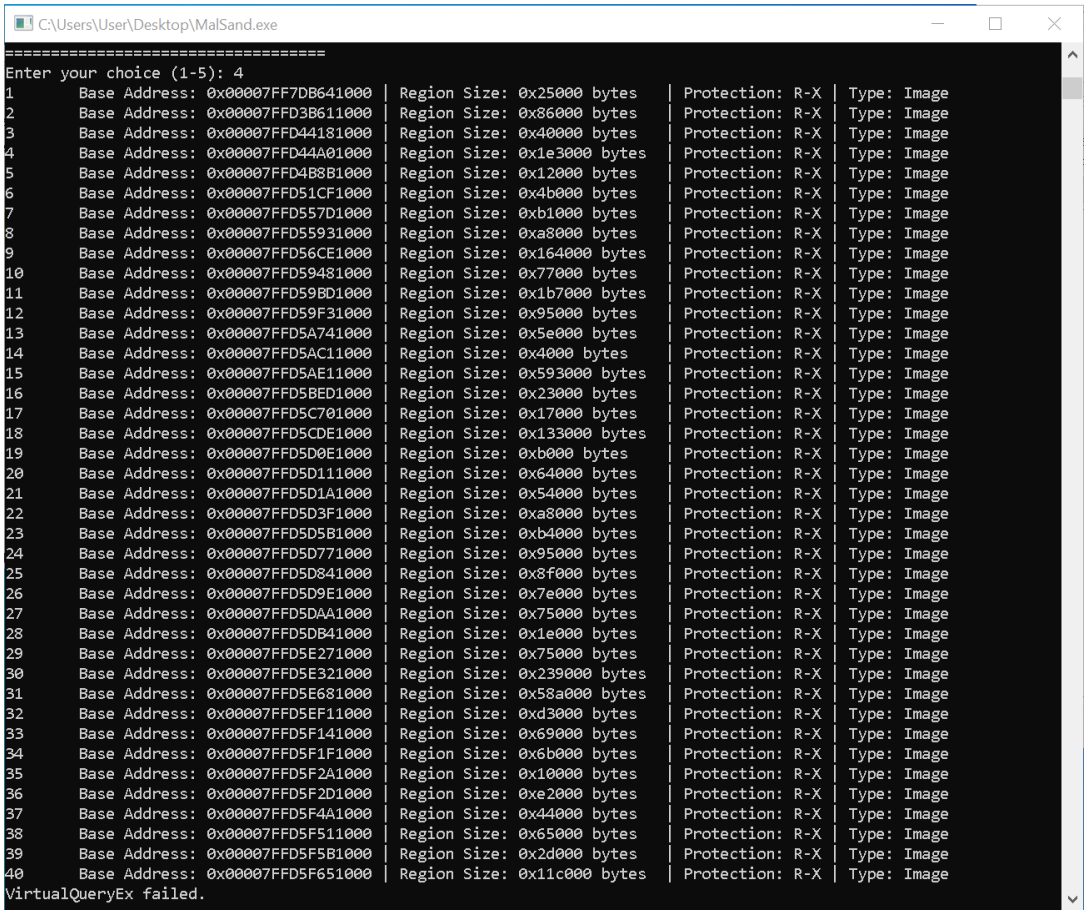


Investigation of Malware Propagation using Sandbox

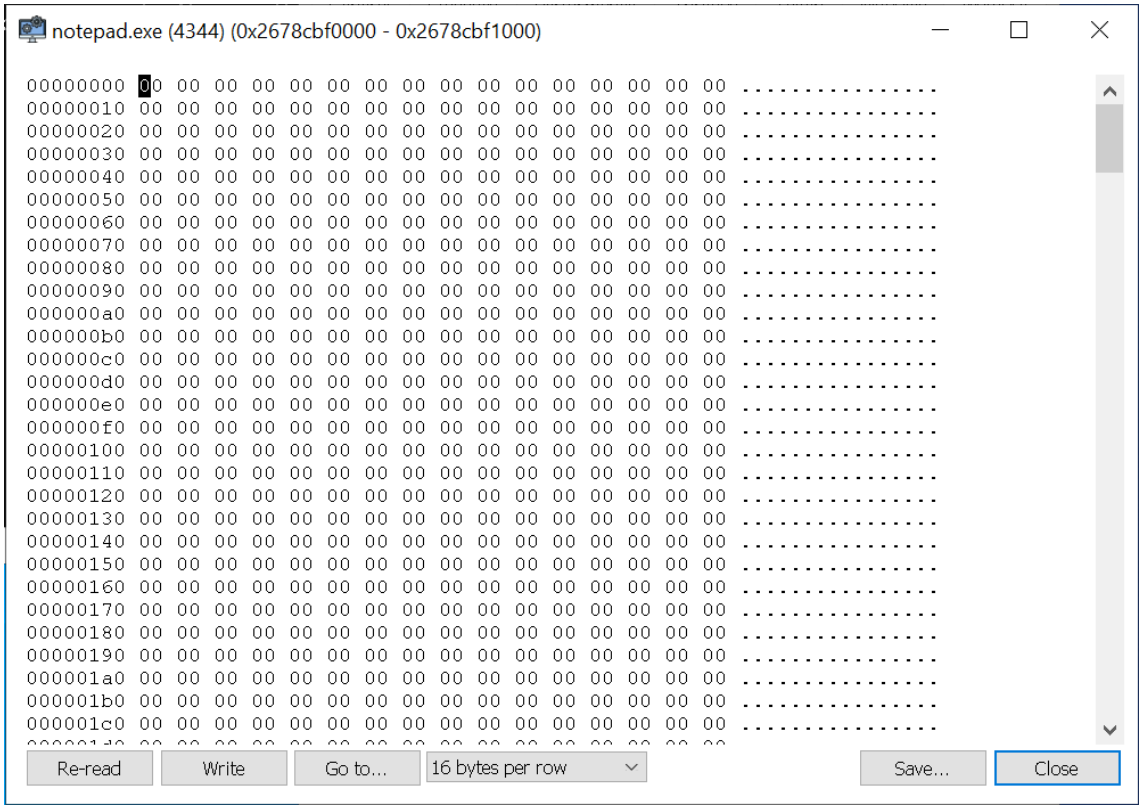
After dumping the active processes in widows, the process ID or PID was noted for notepad.exe. The PID was 4344.



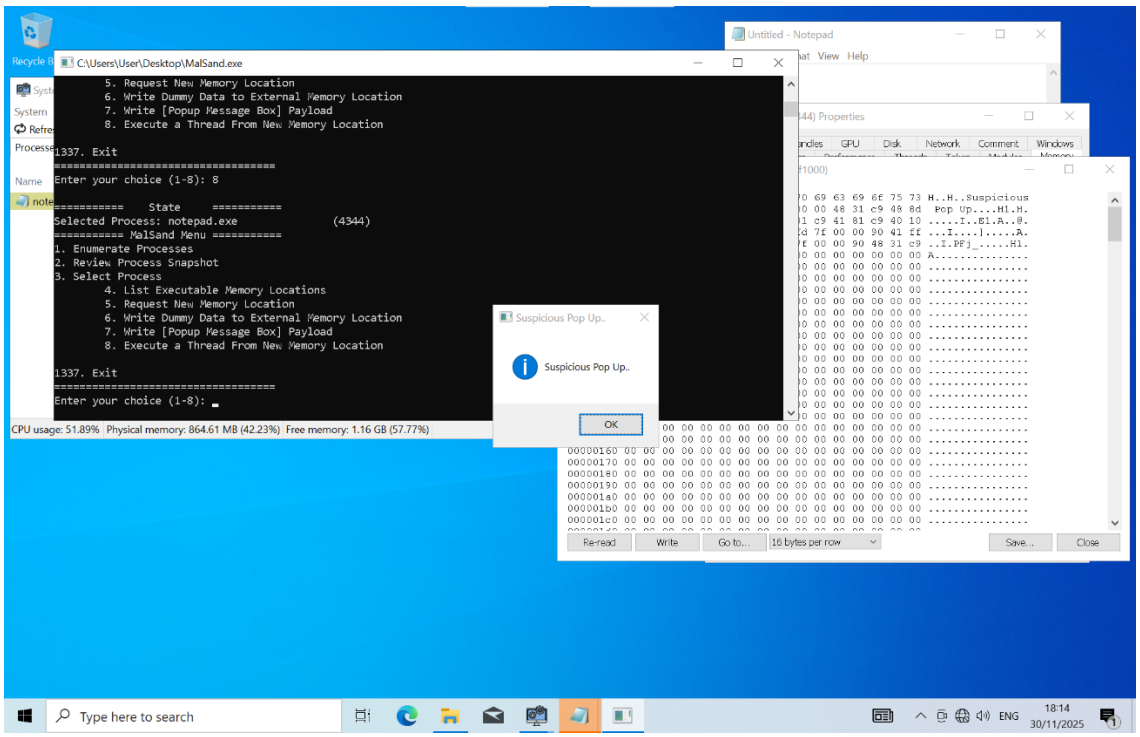
Notepad.exe had 40 active memory allocations with execution permissions.



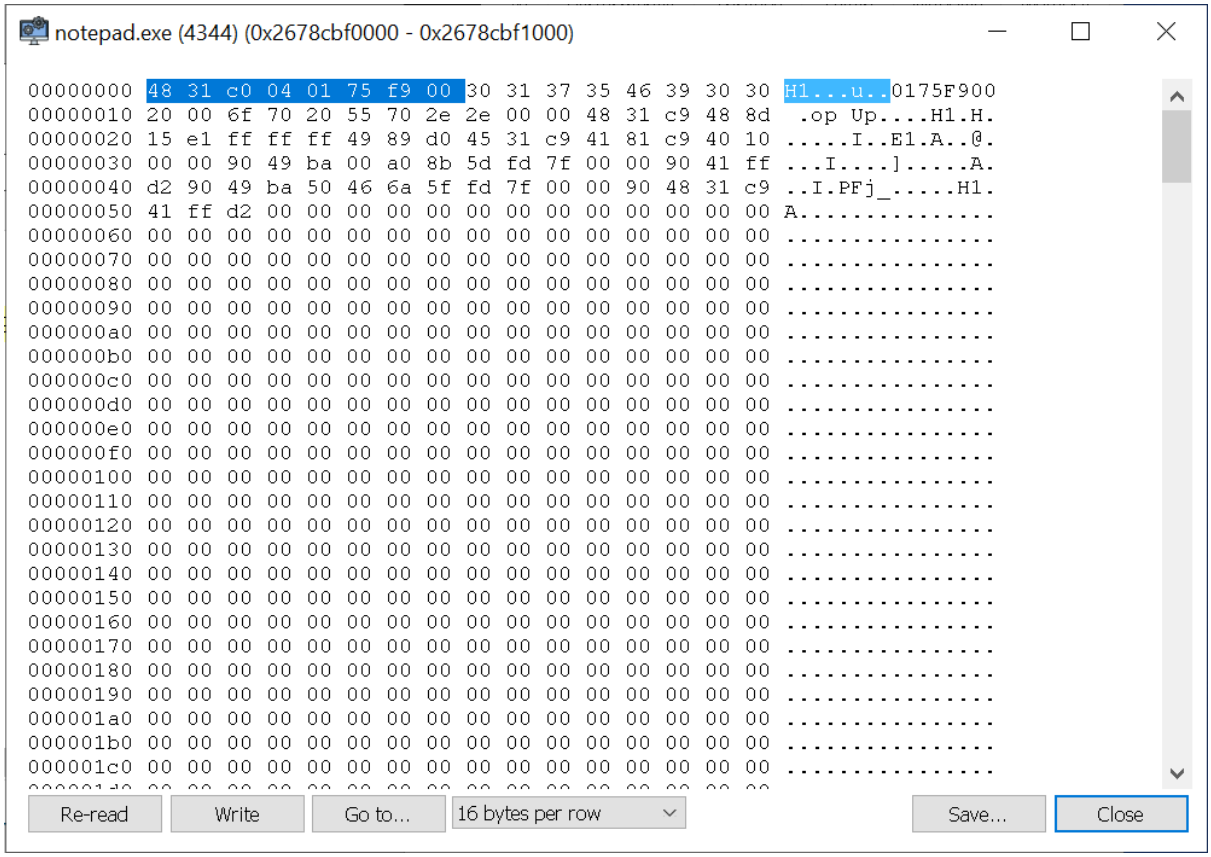
MalSand then allocated a blank memory location its address was 0x2678cbf0000meaning there were now 41 memory allocations in notepad

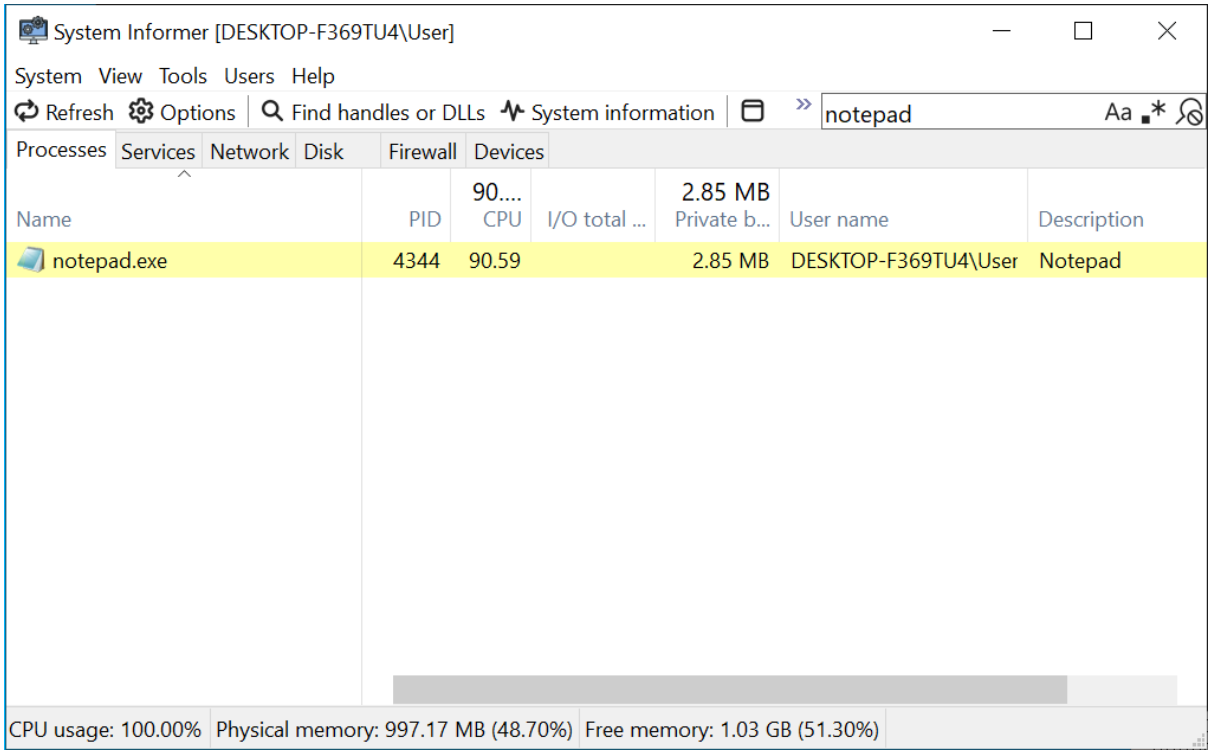


Arbitrary data was then written into the remote memory region, where a popup in windows was created.



Furthermore, a payload to simulate crypto mining on the machine was ran, this data was 48 31 c0 04 01 75 f9 00, this was confirmed to be running by the CPU hitting 100% utilisation, majority of that coming from notepad.exe.





This exercise was able to demonstrate the strengths and dangers of process injection. This malware was able to operate entirely within the memory meaning an attacker would effectively bypass any antivirus scanners since it has no files on the disk. In running a crypto mining simulation, this was able to prove how attackers can not only infect computes for malicious intent, but for financial gain, even without the victim knowing due to the difficulty of detection. Monitoring of CPU spikes would need to be seen to suspect there was a malware operating within the memory.

Individual Summative Reflection

This module has enhanced my understanding of theoretical and practical network defence strategies.

References

Anon., n.d. *Sadmind*. [Online]
Available at: https://www.f-secure.com/v-descs/sadmind.shtml

Cherbaka, P., 2001. *Advanced Incident Handling and Hacker Exploits Practical,* s.l.: GCIH.

Feder, N. L., 2001. SADMIND/IIS Worm. *Global Information Assurance Certification Paper*.

Sophos, 2001. *Unix/SadMind*. [Online]
Available at: https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Unix~SadMind