MEMORY ANALYSIS – RANSOMWARE INVESTIGATION

By Mohammed Choglay

Contents

Acknowledgements 2

Section 1: Introduction 3

Section 1.1: Prerequisites 3

Section 2: infected.vmem memory dump examination 4

Section 3 – Ransomware Attack Preventions 8

Backing up data 8

Patch and Update Software 8

Install and configure AV and Endpoint Security 8

Network segmentation 8

User Training and Awareness 8

Least Privilege Principle 8

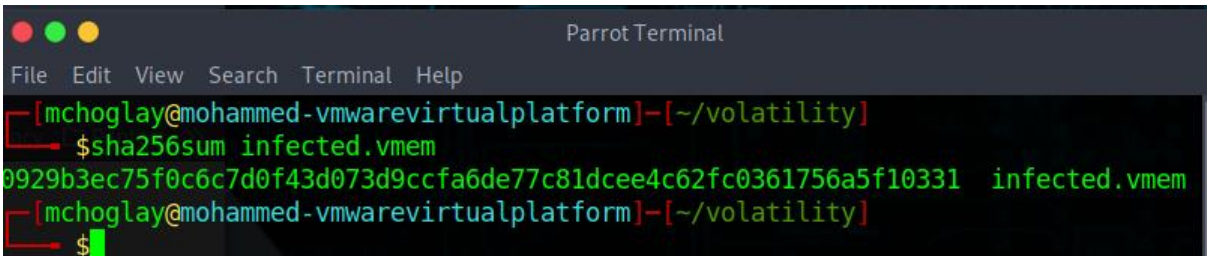
Acknowledgements

BTLO provides all artefacts; without these artefacts, these examinations would not be possible. Without them, there would be no report to write. But if you’re looking to improve your cyber security skills, I recommend BTLO; part of the learning platform is free, a bonus.

Section 1: Introduction

The report is a malware investigation that examines a memory dump (infected.vmem) to see what has potentially occurred on the victims' machine. Learning what happened during the incident is the art of setting future prevention and remediation mechanisms to prevent such attacks.

Before starting any investigation, it's vital that you validate the memory dump that you're working on to ensure that data has not changed from the start of the investigation to the end. Doing the following will ensure you are working on the correct dump. A unique identifier is obtained by taking a hash sum, as shown below in Figure 1.



```
┌─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $sha256sum infected.vmem
0929b3ec75f0c6c7d0f43d073d9ccfa6de77c81dcee4c62fc0361756a5f10331  infected.vmem
┌─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $
```

Figure 1 - Hash value of memory dump (infected.vmem)

Section 1.1: Prerequisites

The Prerequisites are as follows:

1. Windows system or Unix based system (e.g., parrot, kali, ubuntu etc)
2. Volatility An advanced memory forensics framework to analyse the memory dump

Section 2: infected.vmem memory dump examination

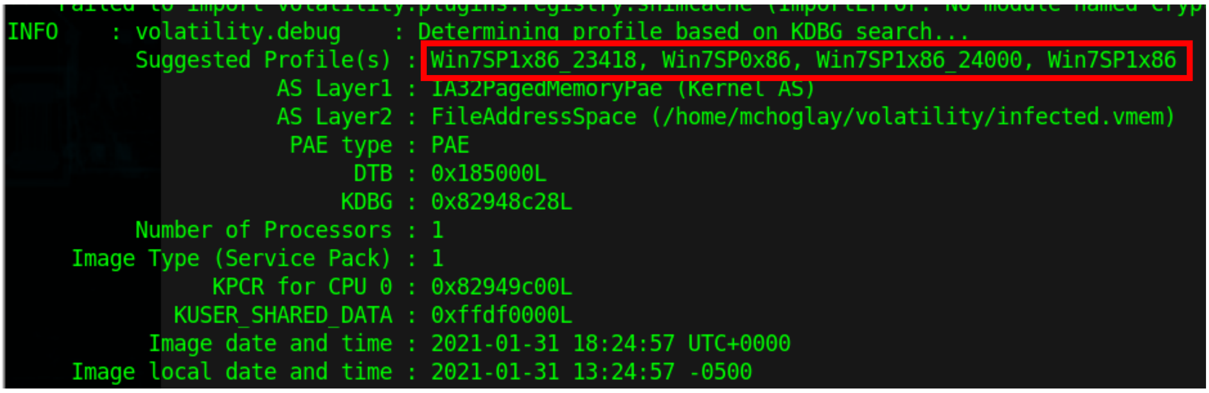
The first initial stage is to obtain a profile for the memory dump. In our case, the memory profile Win7SP1x86. This term means it's a Windows 7 service pack 1, which runs on a 32-bit architecture. This is denoted by x86 within the profile name.

With the profile, you can gather the information you require, as the profile is an essential part of volatility commands.

In order display the following profiles for the memory dump the following command has to be used:

- python vol.py -f **file_memory_dump_name** imageinfo

The output is shown in Figure 2 below. As you can see, it displays four different types of profiles. One of these will work for the command.



```
  Failed to import volatility.plugins.registry.shimcache (ImportError: No module named cryp
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
                     AS Layer1 : IA32PagedMemoryPae (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/mchoglay/volatility/infected.vmem)
                      PAE type : PAE
                           DTB : 0x185000L
                          KDBG : 0x82948c28L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0x82949c00L
             KUSER_SHARED_DATA : 0xffdf0000L
           Image date and time : 2021-01-31 18:24:57 UTC+0000
     Image local date and time : 2021-01-31 13:24:57 -0500
```

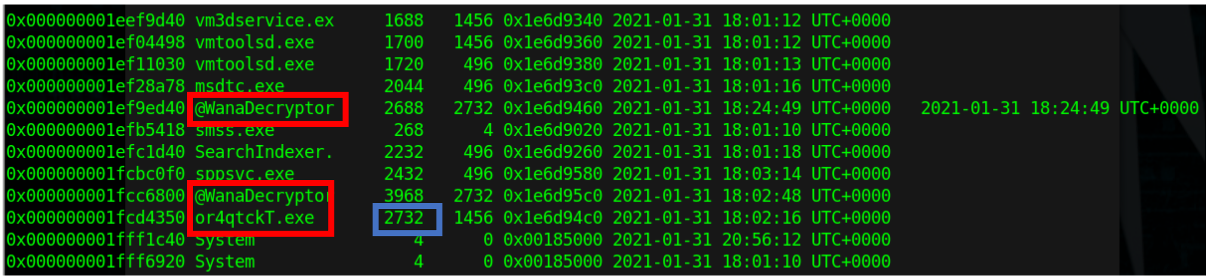
Figure 2 - Profiles for memory dump

Now that the profile has been established, the best next step is to examine the suspicious process. A process is a program within a computer system memory designed to do tasks. Some can be genuine and generic tasks, but others can be malicious.

In Figure 3 below three major processes are identified which are displayed in red outline. However two might go unnoticed for the time being as you need to have knowledge of types malware by name. For now, let’s say most suspicious process is ‘or4qtckT.exe’

Looking at this particular process, it can be seen that it has a process ID is 2732, which is identified within the blue box. To obtain the following information, the following command can be used:

- python vol.py -f **file_memory_dump_name** --profile=**replace_with_identfited_profile** psscan



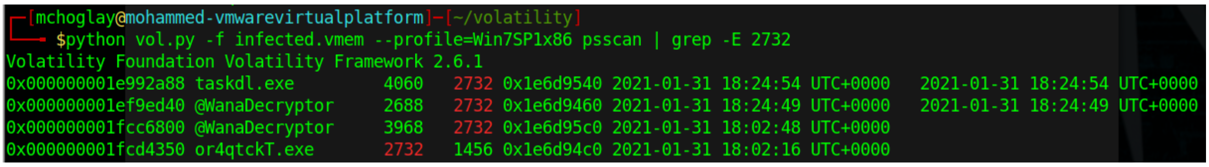
```
0x000000001eef9d40 vm3dservice.ex     1688   1456 0x1e6d9340 2021-01-31 18:01:12 UTC+0000
0x000000001ef04498 vmtoolsd.exe       1700   1456 0x1e6d9360 2021-01-31 18:01:12 UTC+0000
0x000000001ef11030 vmtoolsd.exe       1720    496 0x1e6d9380 2021-01-31 18:01:13 UTC+0000
0x000000001ef28a78 msdtc.exe          2044    496 0x1e6d93c0 2021-01-31 18:01:16 UTC+0000
0x000000001ef9ed40 @WanaDecryptor     2688   2732 0x1e6d9460 2021-01-31 18:24:49 UTC+0000   2021-01-31 18:24:49 UTC+0000
0x000000001efb5418 smss.exe            268      4 0x1e6d9020 2021-01-31 18:01:10 UTC+0000
0x000000001efc1d40 SearchIndexer.     2232    496 0x1e6d9260 2021-01-31 18:01:18 UTC+0000
0x000000001fcbc0f0 sppsvc.exe         2432    496 0x1e6d9580 2021-01-31 18:03:14 UTC+0000
0x000000001fcc6800 @WanaDecryptor     3968   2732 0x1e6d95c0 2021-01-31 18:02:48 UTC+0000
0x000000001fcd4350 or4qtckT.exe       2732   1456 0x1e6d94c0 2021-01-31 18:02:16 UTC+0000
0x000000001fff1c40 System                4      0 0x00185000 2021-01-31 20:56:12 UTC+0000
0x000000001fff6920 System                4      0 0x00185000 2021-01-31 18:01:10 UTC+0000
```

Figure 3 - section of process

Utilising this information, a cross-check can be done to see if that particular process has any sub-process. This can be done by running the same command but with piped grep command attached:

- ➢ python vol.py -f **file_memory_dump_name** --profile=**replace_with_identfited_profile** psscan | grep 2732

Three additional parent process IDs were identified using Figure 4 below. This means that process ID 2732 created three other processes, 4060, 2688, and 3969. These are known as the child process. Also, examining the time differences makes logical sense, as 2732 is being executed first, and 4060 is being executed last.



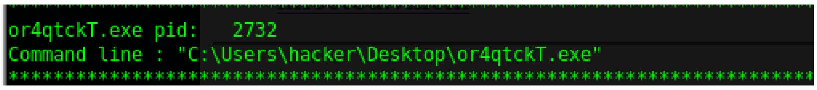
```
┌─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $python vol.py -f infected.vmem --profile=Win7SP1x86 psscan | grep -E 2732
Volatility Foundation Volatility Framework 2.6.1
0x000000001e992a88 taskdl.exe         4060   2732 0x1e6d9540 2021-01-31 18:24:54 UTC+0000   2021-01-31 18:24:54 UTC+0000
0x000000001ef9ed40 @WanaDecryptor     2688   2732 0x1e6d9460 2021-01-31 18:24:49 UTC+0000   2021-01-31 18:24:49 UTC+0000
0x000000001fcc6800 @WanaDecryptor     3968   2732 0x1e6d95c0 2021-01-31 18:02:48 UTC+0000
0x000000001fcd4350 or4qtckT.exe       2732   1456 0x1e6d94c0 2021-01-31 18:02:16 UTC+0000
```

Figure 4 - correlated information regarding PID 2732

Process	**Exit date and time**
Or4qtckT.exe	31/01/2021 18:02:16
@WanaDecryptor	31/01/2021 18:02:48
@WanaDecryptor	31/01/2021 18:24:49
Taskdl.exe	31/01/2021 18:24:54

To see where the execution was initiated from there. This can be done by checking the volatility cmdline option and using the command below. The result of the method is shown in Figure 5 below.

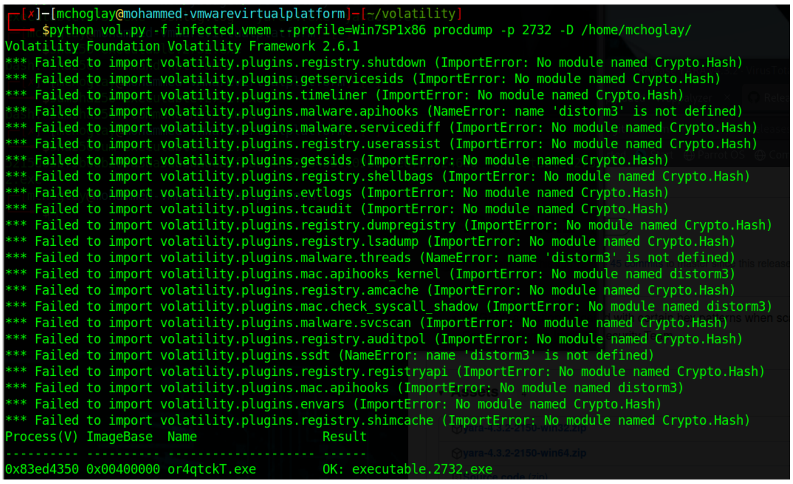
- ➢ python vol.py -f **file_memory_dump_name** --profile=**replace_with_identfited_profile** cmdline



```
or4qtckT.exe pid:    2732
Command line : "C:\Users\hacker\Desktop\or4qtckT.exe"
************************************************************************
```

Figure 5 - Executed location of malware

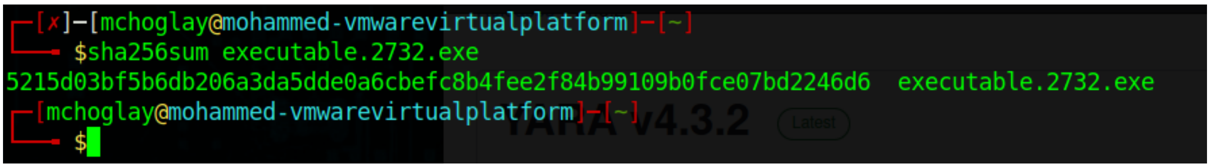
Now that everything is identifiable, we know what artefacts are linked to the possible malware. A sample can be pulled from volatility to learn more about the malware, as shown below in Figure 6



```
┌─[✗]─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $python vol.py -f infected.vmem --profile=Win7SP1x86 procdump -p 2732 -D /home/mchoglay/
Volatility Foundation Volatility Framework 2.6.1
*** Failed to import volatility.plugins.registry.shutdown (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getservicesids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.timeliner (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.apihooks (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.malware.servicediff (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.userassist (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.getsids (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shellbags (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.evtlogs (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.tcaudit (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.dumpregistry (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.lsadump (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.malware.threads (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.mac.apihooks_kernel (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.registry.amcache (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.check_syscall_shadow (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.malware.svcscan (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.auditpol (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.ssdt (NameError: name 'distorm3' is not defined)
*** Failed to import volatility.plugins.registry.registryapi (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.mac.apihooks (ImportError: No module named distorm3)
*** Failed to import volatility.plugins.envars (ImportError: No module named Crypto.Hash)
*** Failed to import volatility.plugins.registry.shimcache (ImportError: No module named Crypto.Hash)
Process(V) ImageBase  Name                 Result
---------- ---------- -------------------- ------
0x83ed4350 0x00400000 or4qtckT.exe         OK: executable.2732.exe
```

Figure 6 - extracting malware sample

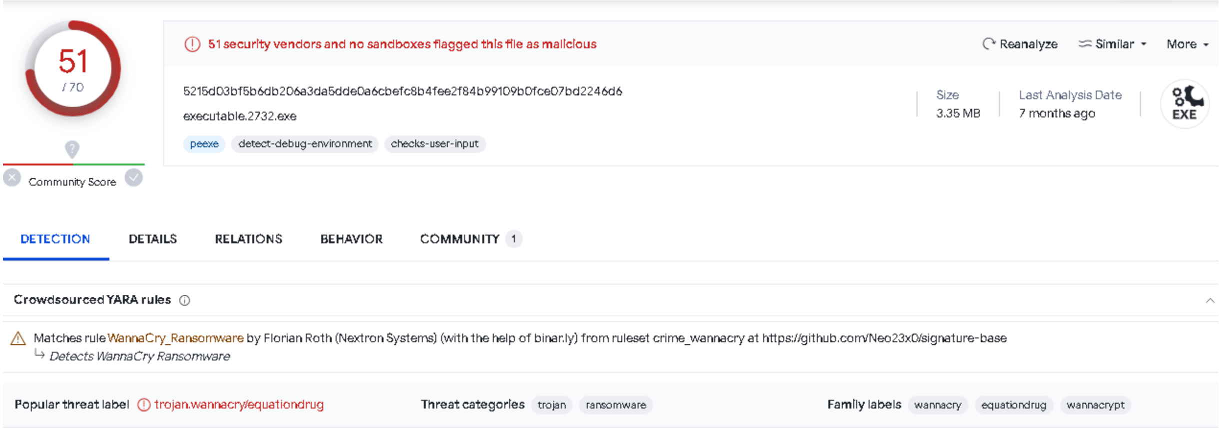
The malware sample is stored in the /home/mchoglay directory. Using the sha256sum command, we can obtain a hash of the sample file. A hash has been taken because it can be used against a b database like virus total.



```
┌─[x]─[mchoglay@mohammed-vmwarevirtualplatform]─[~]
└──╼ $sha256sum executable.2732.exe
5215d03bf5b6db206a3da5dde0a6cbefc8b4fee2f84b99109b0fce07bd2246d6  executable.2732.exe
┌─[mchoglay@mohammed-vmwarevirtualplatform]─[~]
└──╼ $
```

Once the hash was inputted into the virus total, a result appeared, flagged by 51/70 different virus vendors. This is shown in Figure 7 and Figure 8 below. The result also provides the type of malware, a trojan that actually acted as ransomware.

The ransomware has a YARA rule, which is linked to a well-known ransomware crypto worm known as WannaCry. This malware made international news in 2017, affecting over 200,000 computer systems in over 150 countries. One of the most significant targets within the UK was the National Health Service (NHS)



51 security vendors and no sandboxes flagged this file as malicious

5215d03bf5b6db206a3da5dde0a6cbefc8b4fee2f84b99109b0fce07bd2246d6
executable.2732.exe
peexe detect-debug-environment checks-user-input

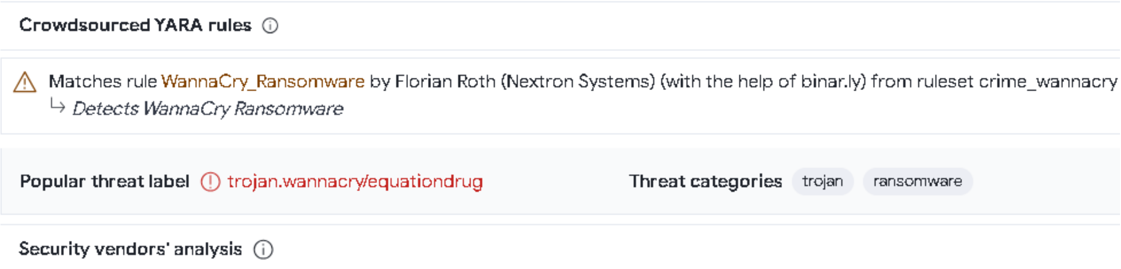
Size 3.35 MB
Last Analysis Date 7 months ago

Crowdsourced YARA rules

Matches rule WannaCry_Ransomware by Florian Roth (Nextron Systems) (with the help of binar.ly) from ruleset crime_wannacry at https://github.com/Neo23x0/signature-base
↳ *Detects WannaCry Ransomware*

Popular threat label trojan.wannacry/equationdrug Threat categories trojan ransomware Family labels wannacry equationdrug wannacrypt

Figure 7 - Virus Total output



Crowdsourced YARA rules

Matches rule WannaCry_Ransomware by Florian Roth (Nextron Systems) (with the help of binar.ly) from ruleset crime_wannacry
↳ *Detects WannaCry Ransomware*

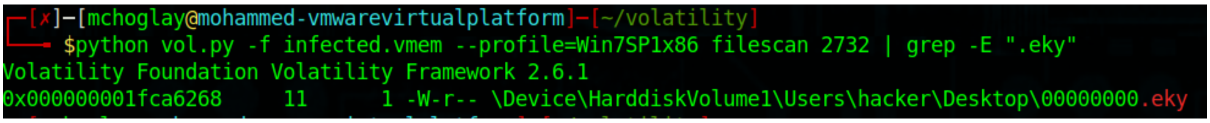
Popular threat label trojan.wannacry/equationdrug Threat categories trojan ransomware

Security vendors' analysis

Figure 8 - Yara rule matched for WannaCry

When conducting research, it was discovered that every infected machine generates private keys RSA-2048. These keys are stored on the local disk, which produces a .eky file after encrypting it with the RSA public key. The purpose of the RSA key is to encrypt every file on the system.

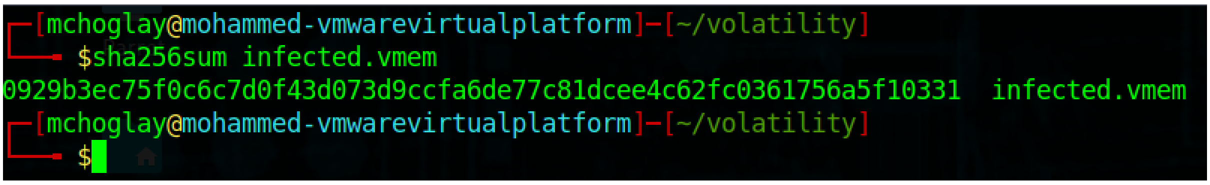
Utilising volatility extraction of ransomware public key filename was identified. The file was known as 00000000.eky, which is shown below in Figure 9.



```
┌─[✗]─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $python vol.py -f infected.vmem --profile=Win7SP1x86 filescan 2732 | grep -E ".eky"
Volatility Foundation Volatility Framework 2.6.1
0x000000001fca6268     11      1 -W-r-- \Device\HarddiskVolume1\Users\hacker\Desktop\00000000.eky
```

Figure 9 - ransomware public key name

Figure 10 proves the correct memory dump has been worked on, and the memory dump has not been modified throughout the investigation if compared to Figure 1.



```
┌─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $sha256sum infected.vmem
0929b3ec75f0c6c7d0f43d073d9ccfa6de77c81dcee4c62fc0361756a5f10331  infected.vmem
┌─[mchoglay@mohammed-vmwarevirtualplatform]─[~/volatility]
└──╼ $
```

Figure 10 - Hash Reverified

Section 3 – Ransomware Attack Preventions

Backing up data

Backups must be taken regularly within any environment. The reason is that if your environment becomes encrypted, you can restore to the previous version. However, it's paramount that these backs are sorted on separate systems to avoid the backup being encrypted.

Patch and Update Software

Everything must be updated to protect systems. These downloads will include the latest security patches. The best update approach is to enable automatic downloads once available. This will always ensure that devices are running the latest versions. A few of these include but are not limited to the following:

- The OS
- Programs running on the OS
- Devices drivers
- Network Infrastructure

Install and configure AV and Endpoint Security

The implementation of the anti-virus is paramount to protect the system from viruses. These will ensure that anything within the AV signature database will not enter the computer system and will be isolated from the system. The AV database is automatically updated or manually needs to be done to catch the latest threats. In addition to this, IPS and IDS can be implemented as extra layers of protection and detection. This will enhance the security posture of the network.

Network segmentation

The purpose behind network segmentation is to isolate your critical systems on the network from less critical ones. This will hinder the malware from travelling to other parts of the network as it will be contained within that section of the segment.

User Training and Awareness

A vital area that has to be carried out is training yourself and other people within the business always to be vigilant when doing daily tasks on different systems. It can take a simple click of an unknown email attachment to mess the whole system up. So, providing training and implementing a wide range of cyber security practices is necessary to protect all assets and people from security threats and social engineering tactics.

Least Privilege Principle

A crucial part that should be applied is the least privilege principle. This idea aims to ensure that individuals on the network are assigned the relevant privileges. At most, everyone should be a Least Privilege user on the network. Least Privilege user is essential because it will Avoid granting unnecessary administrative access, as ransomware often needs elevated privileges to encrypt and remove files from the system.