****

**CT126-3-2-CTF-PRACTICAL CTF STRATEGIES**

**GROUP ASSIGNMENT (40%)**

A green and white flag

AI-generated content may be incorrect.

**[CTF TRY OUT – HTB CTF]**

|  |  |  |
| --- | --- | --- |
| **GROUP NAME: 5MDB** |  | |
| **NAME** | **TP NUMBER** | **WORKLOAD** |
| Luqman Hakim Bin Mat Sa’at | TP086011 | 20% |
| Lee Wei Jie | TP068969 | 20% |
| Brian Tan Xian Hao | TP068853 | 20% |
| Alfred Tan Tin Zhou | TP068762 | 20% |
| Yong Lee Wai | TP068636 | 20% |

Table of Contents

[1.0 Workload Matrix 3](#_Toc197964308)

[2.0 CTF Challenge Write Up 4](#_Toc197964309)

[2.1 Introduction 4](#_Toc197964310)

[2.2 Difficulty Level and Hints 5](#_Toc197964311)

[2.3 Step-by-step Solution 6](#_Toc197964312)

[2.4 Tools, Commands and Scripts 15](#_Toc197964313)

[2.4.1 VMware Workstation and Kali Linux 15](#_Toc197964314)

[2.4.2 GDB (GNU Debugger) 16](#_Toc197964315)

[2.4.3 Ghidra 17](#_Toc197964316)

[2.4.4 Microsoft Excel 17](#_Toc197964317)

[2.5 Vulnerability & Exploits 18](#_Toc197964318)

[2.6 Conclusion 20](#_Toc197964319)

[3.0 References 21](#_Toc197964320)

[4.1 ICTF 2025 23](#_Toc197964321)

[4.2 HTB CTF 2025 24](#_Toc197964322)

**Table of Figures**

[Figure 1: The CTF Challenge 5](#_Toc197964343)

[Figure 2: Points and Difficulty 5](#_Toc197964344)

[Figure 3: Extraction 6](#_Toc197964345)

[Figure 4: Listing files 6](#_Toc197964346)

[Figure 5: Identifying File Type 6](#_Toc197964347)

[Figure 6: View Strings in Binary File 7](#_Toc197964348)

[Figure 7: List Program Functions 8](#_Toc197964349)

[Figure 8: Function Disassembly 8](#_Toc197964350)

[Figure 9: Ghidra Implementation 9](#_Toc197964351)

[Figure 10: prompt\_and\_update\_pos function 10](#_Toc197964352)

[Figure 11: get\_cell function 11](#_Toc197964353)

[Figure 12: get\_flag() function 11](#_Toc197964354)

[Figure 13: breakpoint prompt\_and\_update\_pos 12](#_Toc197964355)

[Figure 14: breakpoint get\_flag 12](#_Toc197964356)

[Figure 15: brute force attack 12](#_Toc197964357)

[Figure 16: get\_flag() breakpoint 13](#_Toc197964358)

[Figure 17: Step/direction to reach the get\_flag() 13](#_Toc197964359)

[Figure 18: connect to remote server 13](#_Toc197964360)

[Figure 19: Final Flag (Hooray) 14](#_Toc197964361)

[Figure 20: Completed challenges 14](#_Toc197964362)

[Figure 21: VMware Workstation Pro 15](#_Toc197964363)

[Figure 22: Kali Linux 15](#_Toc197964364)

[Figure 23: GNU Debugger 16](#_Toc197964365)

[Figure 24: main function 18](#_Toc197964366)

[Figure 25: prompt\_and\_update\_pos function 19](#_Toc197964367)

# 1.0 Workload Matrix

A table with numbers and symbols

AI-generated content may be incorrect.

# 2.0 CTF Challenge Write Up

## 2.1 Introduction

TunnelMadness is a reverse engineering challenge that tasks participants with analyzing a compiled binary to uncover a hidden flag. In this context, reverse engineering has been used as an approach to analyze a program’s behavior, system structure, and binary content to understand its functionality without access to the source code. The main goal of this challenge is to uncover the mechanism used to “tunnel” back into Vault 8707 which will potentially concealed through obfuscated logic, embedded binaries, or manipulated file system. This writeup outlines the step-by-step process of solving the challenge, analyzing the provided files, and reverse-engineering the logic that leads to successful figuring out the flag.

Reverse-engineering is a process that deconstructing either the program code or system into smaller piece, for the purpose of analysing and understanding the operation insight of the program code (*What Is Reverse Engineering? – PreEmptive*, 2023). There are two totally different purpose uses of reverse engineering, it is either use by attackers or cyber security expertise. Attackers use reverse engineering to have insight of the inner working of the software design and architecture, this shown sensitive code exposure toward the attacker (*What Is Reverse Engineering? – PreEmptive*, 2023). On the other hand, reverse engineering also can be use for malware analysis to allow cyber security expertise to understand malware and its operation so they can come out with ways of mitigation toward the targeted malware. Furthermore, security expertise uses reverse engineering to analyse and prevent any potential software vulnerabilities before being attack by cyber criminals (*What Is Reverse Engineering? – PreEmptive*, 2023).

## 2.2 Difficulty Level and Hints

A screenshot of a computer

AI-generated content may be incorrect.

Figure : The CTF Challenge

When we first saw the TunnelMadness challenge at the CTF event, we were pulled in by its cool story. The challenge told us about Vault 8707, an old storage place for master keys that was sealed up many years ago. There were stories that some people who were trapped inside had dug tunnels to escape.

The hints were tricky and unclear, which is normal for these kinds of contests. One hint said, "Look deeper than what you can see," while another told us that "old ways of connecting computers don't really go away." These clues sent us searching through network data and looking for hidden computer services. The people who made the challenge had hidden the entry points well, making us work like detectives finding old forgotten paths.

A blue rectangular object with white lines

AI-generated content may be incorrect.

Figure : Points and Difficulty

This medium level challenge offered us 1000 points; it was a big prize that showed it was both tricky and important in the contest. The high point value made our team eager to solve it, knowing that cracking this one challenge could really boost our standing on the scoreboard.

## 2.3 Step-by-step Solution

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Extraction

At first, we downloaded the file from HackTheBox TryCTF Challenge and gotten this zip file. After that, we extract this zip file and store it inside the same folder using “Extract Here”.



Figure : Listing files

Furthermore, by using terminal to view what is inside the extracted folder “rev\_tunnelmadness” by using the ‘ls’ command. Ls command also knows as “list”, it is a command in Linux that is used to list content of a directory, by listing out the files and folders within it (Zivanov, 2024). As Shown on the figure 4, there is a file named tunnel containing inside this folder/directory.

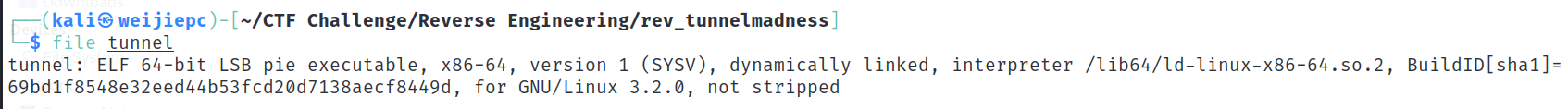


Figure : Identifying File Type

On top of that, using file command to identify the file type of that specific file (S, 2022). In Figure 5, presented that the file type of the ‘tunnel’ file is ELF 64-bit LSB pie executable which mean that this file is executable and is store in an ELF (Executable and linkable Format) along with the PIE (Position Independent Executable).

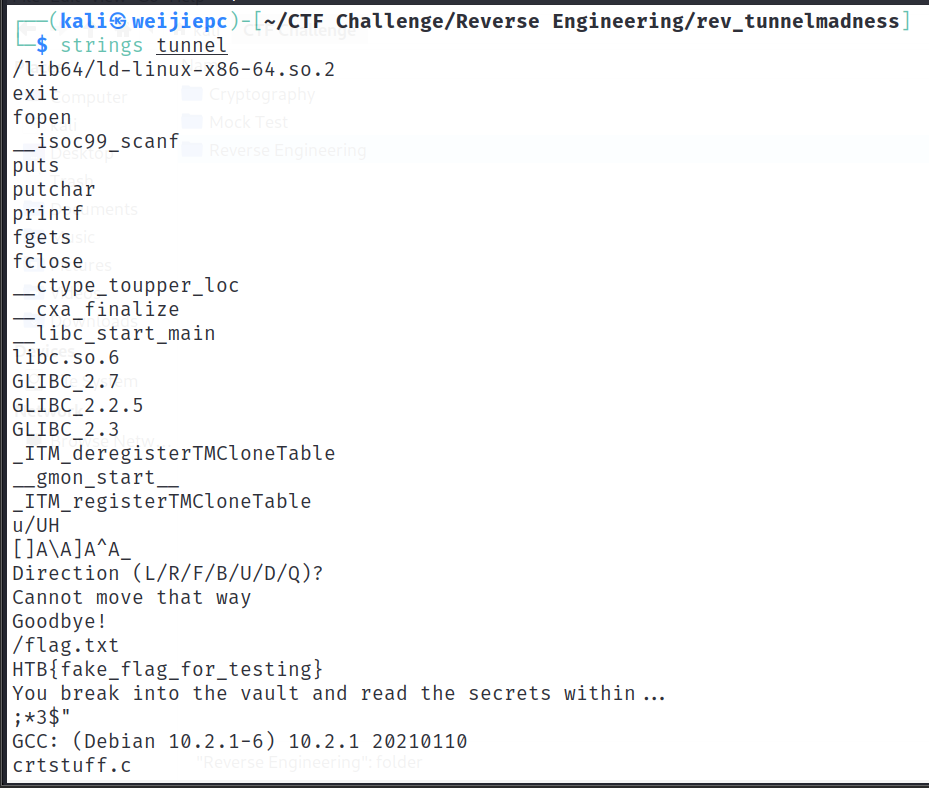


Figure : View Strings in Binary File

Strings command in Linux is used to extract any readable strings from the binary file (Ramuglia, 2023). As presented on above figure, that there are few strings that bring our attention. “/flag.txt” and “HTB{fake\_flag\_for\_testing}”, notice that “flag.txt” is abviously where the flag is located and we found a flag which is “HTB{fake\_flag\_for\_testing}” where it is fake and we tested to submitted to the HTB challenges but failed. At least now we know that the format of the flag is HTB{}.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : List Program Functions

After analysing the file, then using GDB to inspect and debug the program written inside the file. GDB also known as GNU Debugger, it is a powerful debugging tool use to debug program and allow debugger to understand the behaviour of the program (GeeksforGeeks, 2019). We previously analysed that the flag is probably located inside the “/flag.txt”, and now we have to analyse which function inside the program that is probably call out and present the value inside “flag.txt”.



Figure : Function Disassembly

At first, we begin to be analysing the “prompt\_and\_update\_pos” using gdb but it is presented in assembly language which is not too beginner friendly but based on the function named and few known codes, we can guess that this function is use for updating user position once user enter some input. However, the detail of the code is still not explained so we are using other tools for this code.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Ghidra Implementation

Previously step was using GDB as the debugger tools to analyse the program but analysed program were presented in assembly language and doesn’t show details. In the figure above, we are using Ghidra to analyse the code. Ghidra is a well know open-source software reverse engineering (SRE) framework developed by the National Security Agency (NSA), it allows analysts to inspect the code without executing it (Murphy, 2024). The reason that using Ghidra is that it provides a clearer visualise for the structure of the program & more detailed of the program code which is more beginner friendly than GDB. As shown in the Figure 9, there is the main function of the program code. At first the main function defines 3 local variables with the name of local\_14, local\_10 and local\_c and given the value of ‘0’. After that, using the while loop to the process and every user input then “prompt\_and\_update\_pos” will read through user input and update the position of the user. At the beginning of the main function before user input, the system will check user location by using the get\_cell function and if the cell == 3 then the while loop will break, and get\_flag() function will be called and flag will be printed.

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure : prompt\_and\_update\_pos function

After understanding the main function code, inside main function there are prompt\_and\_update\_pos function. In the figure above showing the code of the prompt\_and\_update\_pos, where this function will be used to validate and update the position of the user, it seems like this program have something to associate with MAZE or 3D grid. Firstly, print out a message “Direction (L/R/F/B/U/D/Q)?” and wait user to input a value then only validate the input and save the current position at the maze. Secondly, the function will be based on validated user input and update the position accordingly. Finally, only if user input is invalid then will exit this function.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : get\_cell function

According to previous analysis, this program is most likely to be a Maze game program where it requires user to reach specific cell then only the flag will be printed. The code present on above figure 11, is basically get the current position of the user and return it in a (x,y,z) coordinate.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : get\_flag() function

Along with the combination of prompt\_and\_update\_pos and get\_cell, once the user has reached the specific cell then the get\_flag() function will be called. Inside this get\_flag(), it will first attempt to read /flag.txt and if file not being found then will print HTB{fake\_flag\_for\_texting} else if the file being found then the value inside the file will be printed. Based on the current and previous finding, we can come to a conclude that this program is most likely to be a MAZE game and for user to get to the specific cell then only the flag will be printed which is located inside /flag.txt.

A screen shot of a computer

AI-generated content may be incorrect.

Figure : breakpoint prompt\_and\_update\_pos



Figure : breakpoint get\_flag

After analysing the program, we have a basic understanding and overview of how the system flow and operate. Using GDB to run the program and using breakpoint at prompt\_and\_update\_pos and get\_flag(), shown on above figure 13,14. From the figure above, after setting a breakpoint at prompt\_and\_update\_pos function then program will stop operating when reaching prompt\_and\_update\_pos function, and the reason why print $rdi is to get the memory address first argument being put into the prompt\_and\_update\_pos function so that by using x/3dw <memory\_address> then can know what is the memory in the memory address that store inside the $rdi register. By doing all that, then can know the exact starting coordinate of the MAZE program which is (x)0,(y)0,(z)0.

A white background with black lines

AI-generated content may be incorrect.

Figure : brute force attack

After knowing the starting coordinate of the MAZE program then remove the breakpoint for prompt\_and\_update\_pos function to prevent the program for stopping every time use input a value. From there on, the method uses to get the flag is through manually brute force the program until gets the flag. By manually inserting “L”, “R”, “F”, “B”, “U”, “D”, “Q” until reach get\_flag breakpoint.

A white background with black and orange text

AI-generated content may be incorrect.

Figure : get\_flag() breakpoint

After brute force and reach the breakpoint of get\_flag(), we have concluded the step/direction to reach the get\_flag function which is shown on the figure 17 below. The coordinate where reach the point of printing the flag is 19, 19, 19. After that, record where the flag contains and each step of the route to the destination.

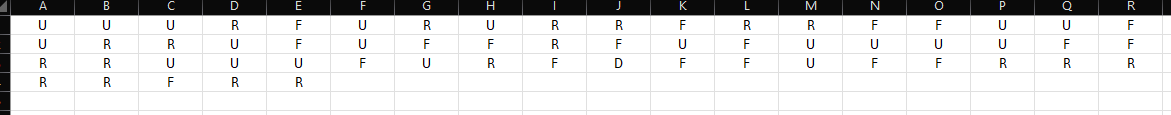


Figure : Step/direction to reach the get\_flag()

A white background with black text

AI-generated content may be incorrect.

Figure : connect to remote server

Connect to the remote server and repeat the step in figure 17 to reach the coordinate 19,19,19 to get the final flag of the challenges.

A computer screen shot of a computer code

AI-generated content may be incorrect.

Figure : Final Flag (Hooray)

After entering the exact same step and hooray the flag have printed. If the program were run in gdb because it is run locally where there is no /flag.txt file containing inside the extracted folder of rev\_tunnelmadness which mean that even though proceed to get\_flag function, there won’t be /flag.txt to let it open and if /flag.txt file is failed to open then get\_flag function will print the fake flag.

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Completed challenges

After submitted the real flag then the challenge has been solved and that is the end of the write up for rev\_tunnelmadness.

## 2.4 Tools, Commands and Scripts

### 2.4.1 VMware Workstation and Kali Linux

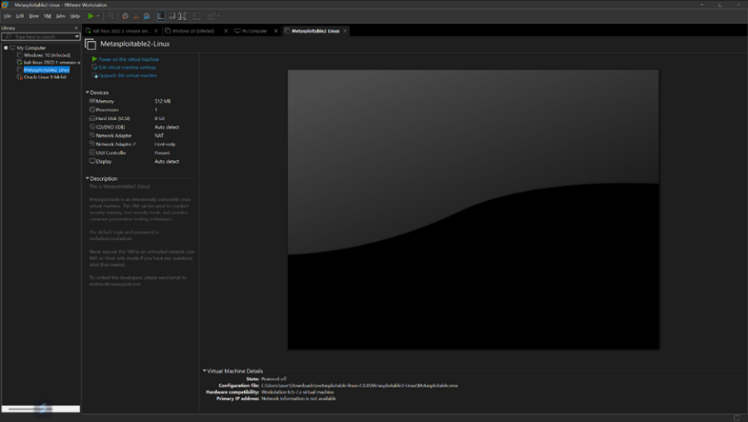


Figure : VMware Workstation Pro

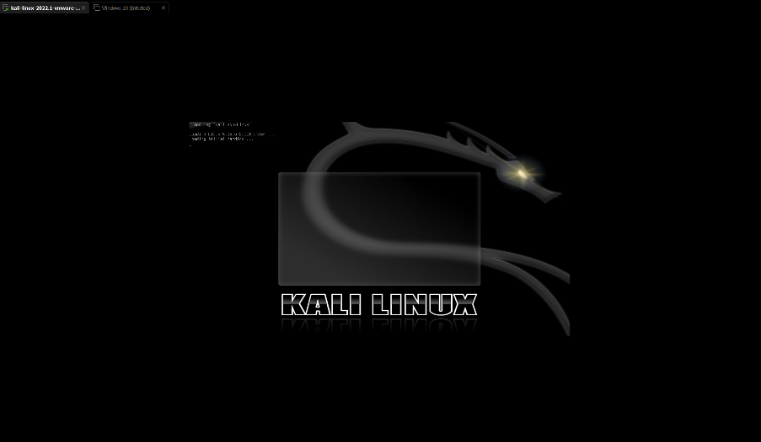


Figure : Kali Linux

Throughout the CTF challenge, Kali Linux was used inside VMware Workstation Pro. The VM Workstation Pro is a virtual machine which is used to produce and create virtual and isolated environment where the user could run different operating systems from a single host machine (*What Is a Virtual Machine? VM Uses and Benefits | Google Cloud*, 2025). This is to ensure that any unwanted malicious software from the challenges were to be contained within the virtual machine itself and not the host PC. Kali Linux on the other hand was used because the features and resources built into the operating system. Kali Linux has a library of programs that are useful for CTF challenges such as reverse engineering which uses GDB.

Commands used:

1. strings
   1. The strings command pulls and prints out texts inside binary files that would otherwise not be possible such as when using cat
2. ls
   1. Lists out files inside the current folder or working directory except for hidden files
3. file
   1. The file command is used to find out the CTF challenge file type. In this case the tunnel file is of a 64-bit Executable and Linkable Format type.

### 2.4.2 GDB (GNU Debugger)

A screenshot of a computer

AI-generated content may be incorrect.

Figure : GNU Debugger

Functions used with GDB:

1. print $rdi
   1. The command print $rdi displays the contents inside the RDI register (destination index)
2. x/3dw
   1. This command is used to examine the arrays of integers, structs and pointers at a certain address
3. disassemble
   1. This command is used to show the assembly instructions or code for a selected function or a certain range of memory
4. info function
   1. Lists out all the functions that are contained inside the tunnel ELF file
5. break
   1. The command pauses the program when it reaches a certain line or function or when the program execution meets a specified condition

### 2.4.3 Ghidra

Ghidra is an open-source reverse engineering suite and an alternative to the IDA Pro. Ghidra is also similar to GDB, but it has a (GUI) graphical user interface which makes it easier for users to navigate and understand the inner workings of the program being analyzed. The program would show the assembly code and instructions. It could also decompile and display in C code.

### 2.4.4 Microsoft Excel

Microsoft Excel was used for keeping track and finding steps needed to find the goal inside the maze program.

## 2.5 Vulnerability & Exploits

A vulnerability is a weakness exist in a system that allow attackers to exploit the vulnerability and deliver a successful attack through either unauthorized access to the system or sensitive data breaching. This attack usually occurs either through human error, features vulnerability or flaws (*Understanding Vulnerabilities*, 2025).

Exploit is a software program or tools that attacker use to take advantages of vulnerabilities of the system. This allows them to gain unauthorized access and install malware or ransomware to the targeted system (*Simplicity Showdown with Cisco Umbrella*, 2025). The main difference between vulnerabilities and exploit is that vulnerabilities is the weakness of the system and exploit is the program tools that attacker use to gain the authorized access toward the targeted system.

A computer screen shot of a program code

AI-generated content may be incorrect.

Figure : main function

According to the figure n on above, the code is written in while loop which mean program will keep running if the Boolean value is true or no break and will keep accepting user input. Once user reach a specific cell then only the program will be stop in the if condition and a message will be printed and get\_flag() function will be process.

A computer screen shot of a program

AI-generated content may be incorrect.

Figure : prompt\_and\_update\_pos function

Inside the prompt\_and\_update\_pos function there will print out a message asking what direction user would like to go by using switch condition to handle different input by user. This function will validate whether the direction is valid to proceed and if not then message “Cannot move that way” will be printed and will proceed back to the original start asking user which direction they would like to move, and this process will keep on going unless user reach the desirable cell location or user enter ‘q’ to terminate the execution of the program.

This mean that, we can implement brute-force strategy to attack these vulnerabilities of the program. Brute-Force attack is a type of hacking method that uses trial and error to crack the password, login credential and encryption keys. It works by keep trying it until the correct output being met, and usually hacker would use wide variety and combination until the correct information being found & use. This tactic is reliable for gaining unauthorized access to accounts, sensitive data and organization networks and systems (*What Is a Brute Force Attack? Definition, Types & How It Works | Fortinet*, 2017).

Since this program doesn’t have input sanitization which mean that user can keep trying possible combination of direction and eventually at the end the correct cell will reach, and the flag will be printed. There is two way to exploit this system, one is through manually (which is what we are using) by using Ghidra to understand the overflow of the system and understand each function contain inside the program then use GDB to set breakpoint at the function to know the coordinate and set breakpoint at the get\_flag() function to know the correct cell coordinate. At the end manually test out, the direction to reach the correct cell and connect to the remote server and enter the step/direction recorded and get the real flag from it.

## 2.6 Conclusion

In Conclusion, completing the TunnelMadness challenge provided valuable hands-on experience in reverse engineering and binary analysis. By stepping through the challenge deeper, we learned how to deconstruct a compiled binary both static and dynamic analysis technique. The challenge highlighted the importance of understanding program structure, functions calls, and system behaviour, especially when navigating a virtual environment like the simulated maze. One of the key takeaways was the effectiveness of combining tools like Ghidra for structure analysis and GDB for runtime behaviour, which helping us in identify key functions and logic paths. Overall, this challenge not only strengthened our technical skills but also improved our problem-solving mindset, persistence, and ability to work collaboratively under pressure which was one of the core qualities essential in real-world cybersecurity scenarios.

# 3.0 References

Zivanov, S. (2024, October 10). *Linux ls Command - How to List Files and Directories*. Knowledge Base by PhoenixNAP. <https://phoenixnap.com/kb/ls-command#:~:text=The%20ls%20(list)%20command%20in,%2C%20sizes%2C%20and%20modification%20dates>.

S, E. (2022, January 27). *The Linux File Command: How to Use It to Determine a File Type*. Hostinger Tutorials. <https://www.hostinger.com/uk/tutorials/linux-file-command#:~:text=To%20organize%20information%20quickly%2C%20Linux,Mail%20Extensions%20(MIME)%20format>.

‌ Ramuglia, G. (2023, December 28). *“strings” Linux Command | Extracting Strings in Binary Files*. Linux Dedicated Server Blog. <https://ioflood.com/blog/strings-linux-command/#:~:text=The%20'strings'%20command%20in%20Linux%20is%20used%20to%20extract%20readable,bin%20.&text=In%20this%20example%2C%20we've,a%20binary%20file%20named%20'myfile>.

Murphy, D. (2024, March 11). *What is Ghidra and How to Use it to Reverse Engineer Malware*. Lepide Blog: A Guide to IT Security, Compliance and IT Operations. <https://www.lepide.com/blog/what-is-ghidra-and-how-to-use-it/>

*What is a Brute Force Attack? Definition, Types & How It Works | Fortinet*. (2017). Fortinet. <https://www.fortinet.com/resources/cyberglossary/brute-force-attack>

Watson, L. (2024, September 25). *Handy Tools for CTF Competitions*. Lnwatson.co.uk; Lewis Watson. <https://lnwatson.co.uk/posts/ctf-tools/>

Lab, O. (2024). *Wireshark - CTF Handbook*. Ctf101.org. <https://ctf101.org/forensics/what-is-wireshark/>

Moisset, S. (2023). *CTF strategies & techniques*. Snyk. <https://snyk.io/articles/ctf/strategies-techniques/>

Roderos, A. (2023, October 30). *How to solve my PCAP CTF challenges | Andrew Roderos*. Andrew Roderos. <https://andrewroderos.com/how-to-solve-my-pcap-ctf-challenges/>

*What Is A Virtual Machine? VM Uses and Benefits | Google Cloud*. (2025). Google Cloud. <https://cloud.google.com/learn/what-is-a-virtual-machine>

*Understanding vulnerabilities*. (2025). Ncsc.gov.uk. <https://www.ncsc.gov.uk/collection/vulnerability-management/understanding-vulnerabilities>

*Simplicity Showdown with Cisco Umbrella*. (2025, April). Cisco. <https://www.cisco.com/site/us/en/learn/topics/security/what-is-an-exploit.html>

‌

*What Is Reverse Engineering? – PreEmptive*. (2023, December 8). PreEmptive In-App Protection. <https://www.preemptive.com/blog/what-is-reverse-engineering/#:~:text=Reverse%20engineering%20is%20a%20process,represents%20a%20dual%2Dedged%20sword>.

‌

‌

‌

4.0 Appendix

## 4.1 ICTF 2025

A group of people sitting at a table

AI-generated content may be incorrect.

A group of people sitting at a table

AI-generated content may be incorrect.

## 4.2 HTB CTF 2025

A person using a computer

AI-generated content may be incorrect.

A person pointing at a computer

AI-generated content may be incorrect.

A group of people using a computer

AI-generated content may be incorrect.

A group of men sitting at a table with a computer

AI-generated content may be incorrect.