LAB TITLE:	YARA RULE DETECTION - MALWARE ANALYSIS
STUDENT NAME:	BRIAN NJIRU
STUDENT ID:	2025/ACTI/6177
COURSE NAME:	AMINU IDRIS
DATE OF SUBMISSION:	7TH APRIL 2025
VERSION:	1.0

1. EXECUTIVE SUMMARY

1.1 Purpose of the lab.

The lab requires a student to analyze a malware sample and write YARA rules to detect the malware. The YARA rules target various indicators associated with the malware.

1.2 Key Activities

The lab mainly involved writing YARA rules to detect the presence of the malware.

1.3 Major Findings

The major findings from this lab involved the conclusion that YARA is a very effective tool in analyzing malware and that the provided strain of malware poses no threat to existing computer systems.

2. LAB OBJECTIVES.

The primary objective of the lab was to write YARA rules to detect the provided sample of malware. The lab requires a thorough analysis of the provided sample in order to find indicators that can be used to track the presence of the malware.

3. TOOLS AND RESOURCES.

To achieve the objectives of the lab, I employed the use of the following tools:

- Kali Linux – The Kali Linux operating system was used as the main operating system for this analysis due to its compatibility with YARA rules.
- YARA Rules – YARA Rules was used to write and execute scripts that were able to detect the presence of the malware in a computer system.
- VMware Hypervisor – I used VMware Workstation as the hypervisor to create a virtual machine running Kali Linux.

4. METHODOLOGY

The following steps were followed to achieve the goals of the lab:

- Download the malware sample
- Write YARA rules to detect malicious strings
- Test your YARA rules.

5. SCREENSHOTS AND EVIDENCE

The following screenshots were taken during the lab:

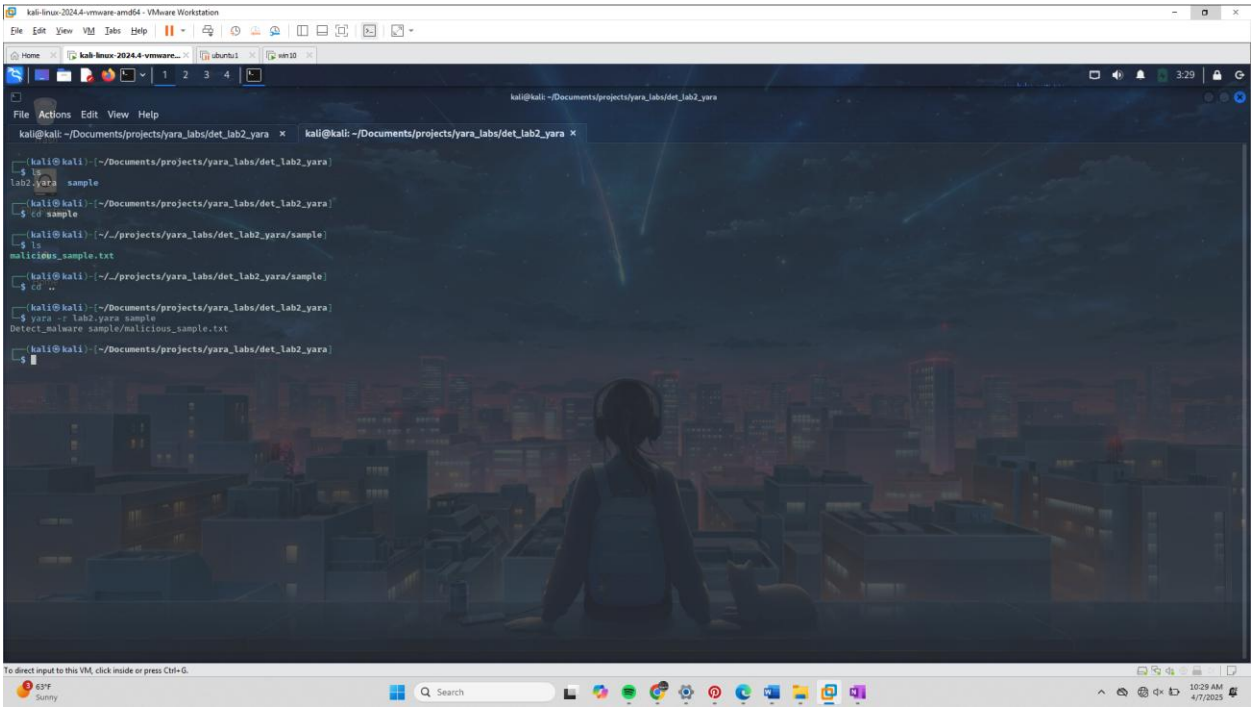

Figure 1 running the YARA rule.

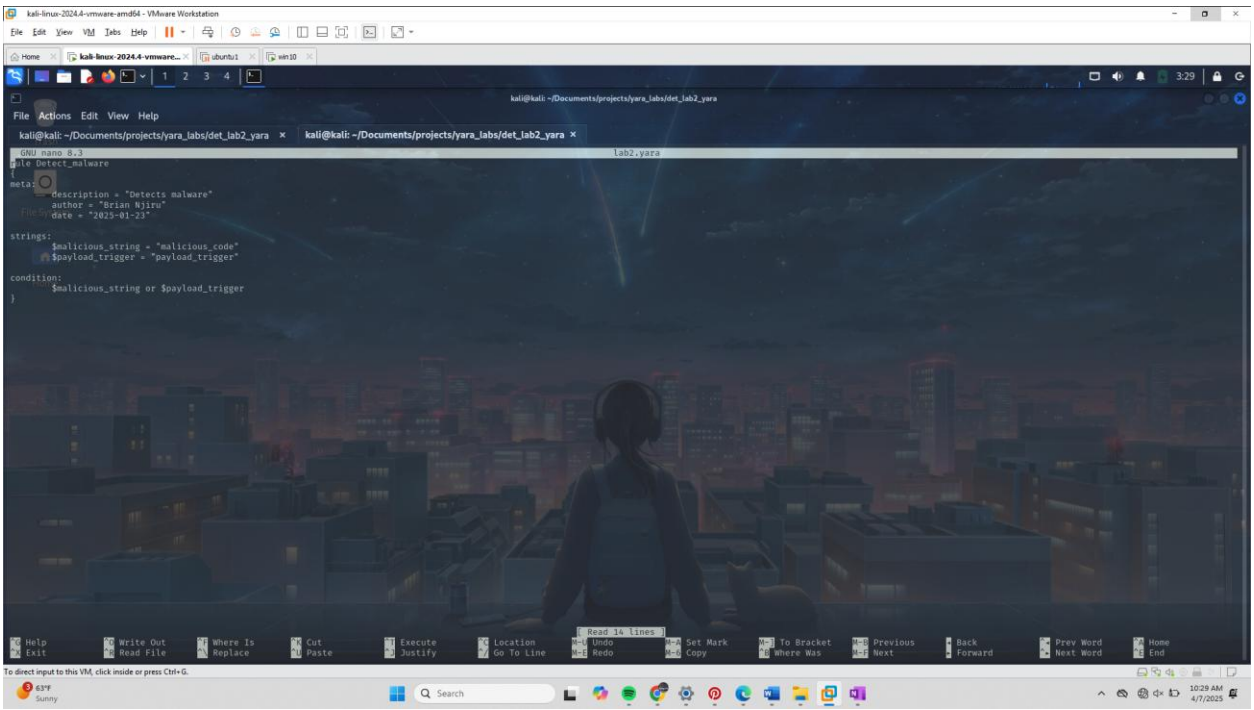

Figure 2 YARA rule

6. ANALYSIS AND FINDINGS.

6.1 Malicious Strings.

The following strings were observed in the malware's execution:

- "malicious_code"
- "payload_trigger"

A YARA rule was written to detect these strings. The YARA rule, **lab2.yara**, detects both "malicious_code" and "payload_trigger" strings.

If the string was obfuscated or encoded, the rule would become unusable.

6.2 File Creation.

The lab provided a sample labelled malware.txt which I suppose is the file created by the malware and not the actual malware itself.

File details:

- Name: malicious_sample.txt
- Type: plain text file
- Content: simulated malware indicators such as: malware payload reference YARA detection phrase, Keywords like malicious_code and payload_trigger and log-style lines mimicking activity like "Malware execution completed successfully".

The lab required me to write a YARA rule that detects the file "malicious_sample.txt" based on its name or content.

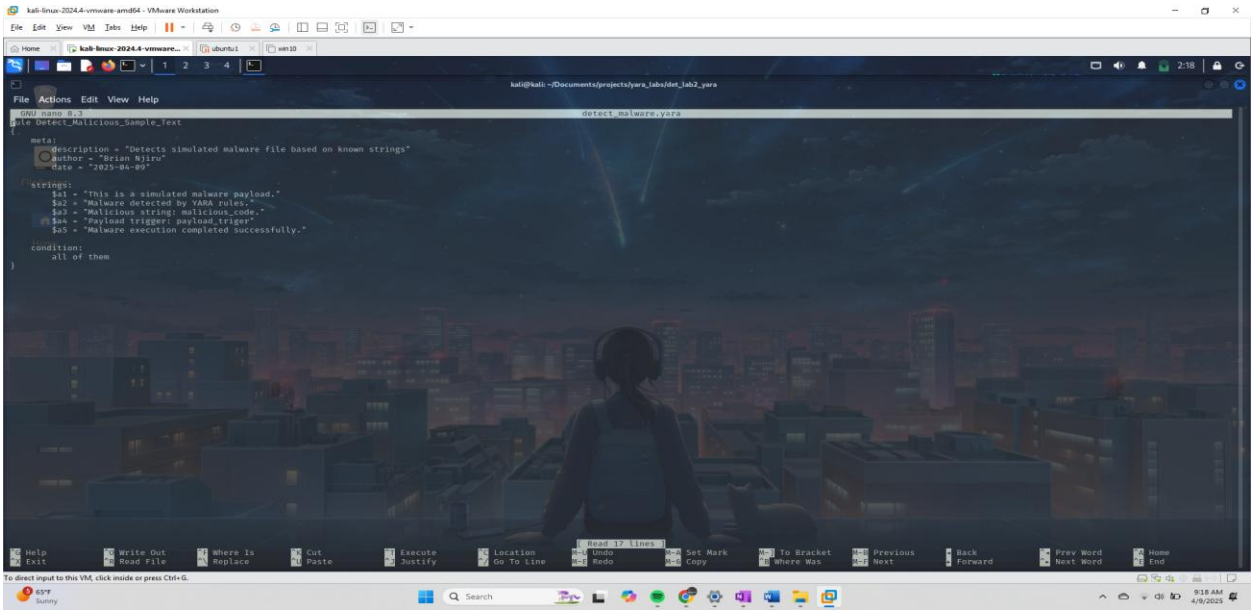

Figure 3 detect_malware.yara

6.3 Byte Patterns

The provided sample is assumed to be a text-based payload, this allows us to convert one of the more unique strings into a byte sequence (ASCII encoded).

I chose the string "**Payload trigger: payload_trigger**" and used the **echo** and **xxd** tools to dins the byte sequence of the chosen string.

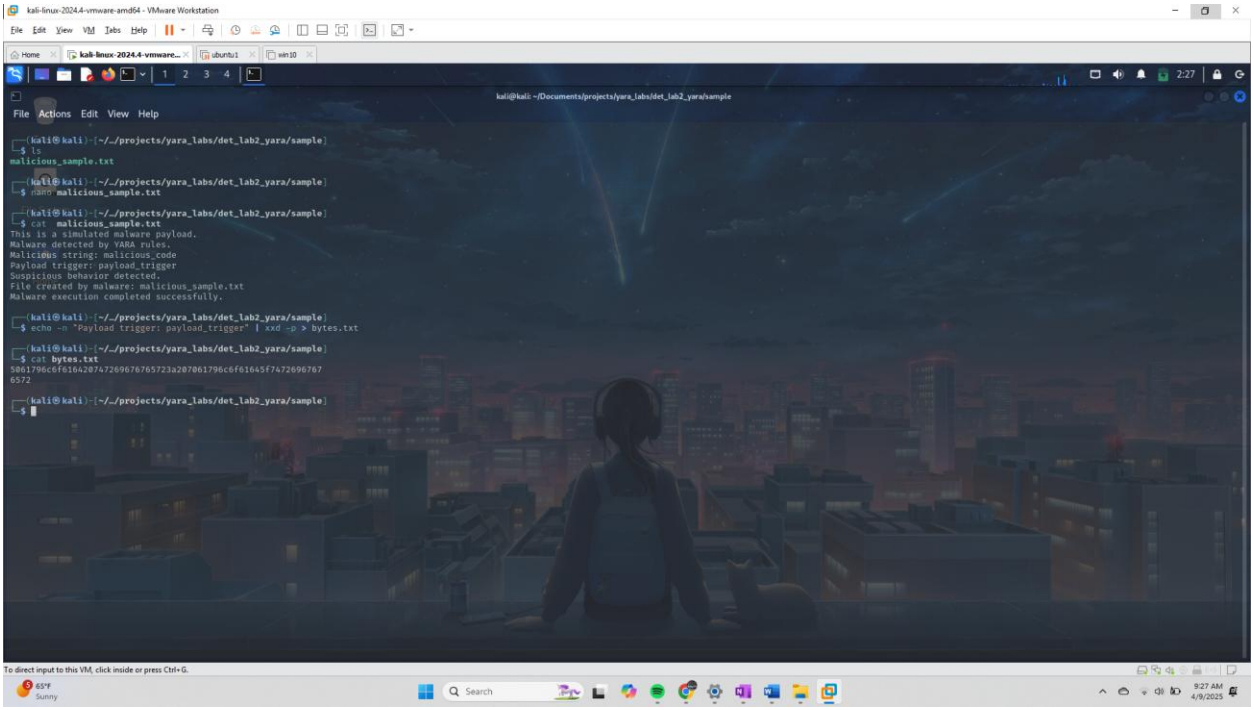

Figure 4 Byte Sequence for chosen string

I then wrote a YARA rule that used the presence of the byte sequence as a condition to detect the malware.

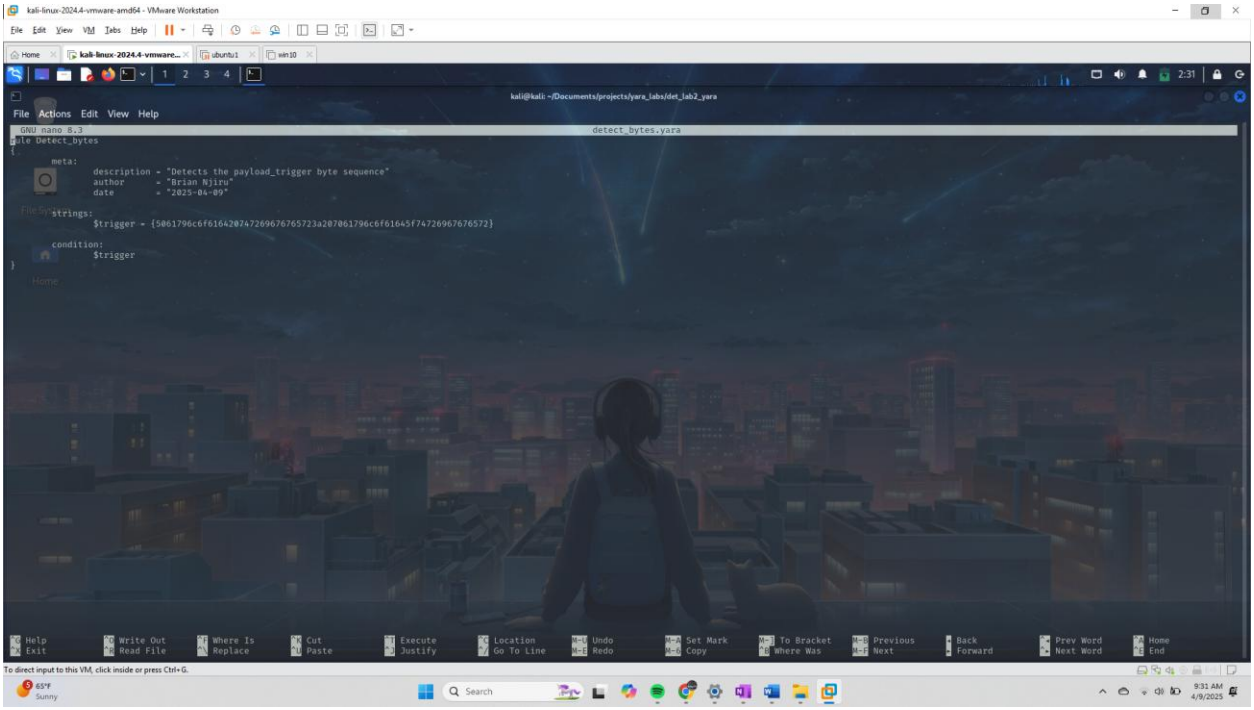

Figure 5 YARA rule to detect byte sequence

I ran the YARA rule which successfully identified the malware in a folder named **sample**.

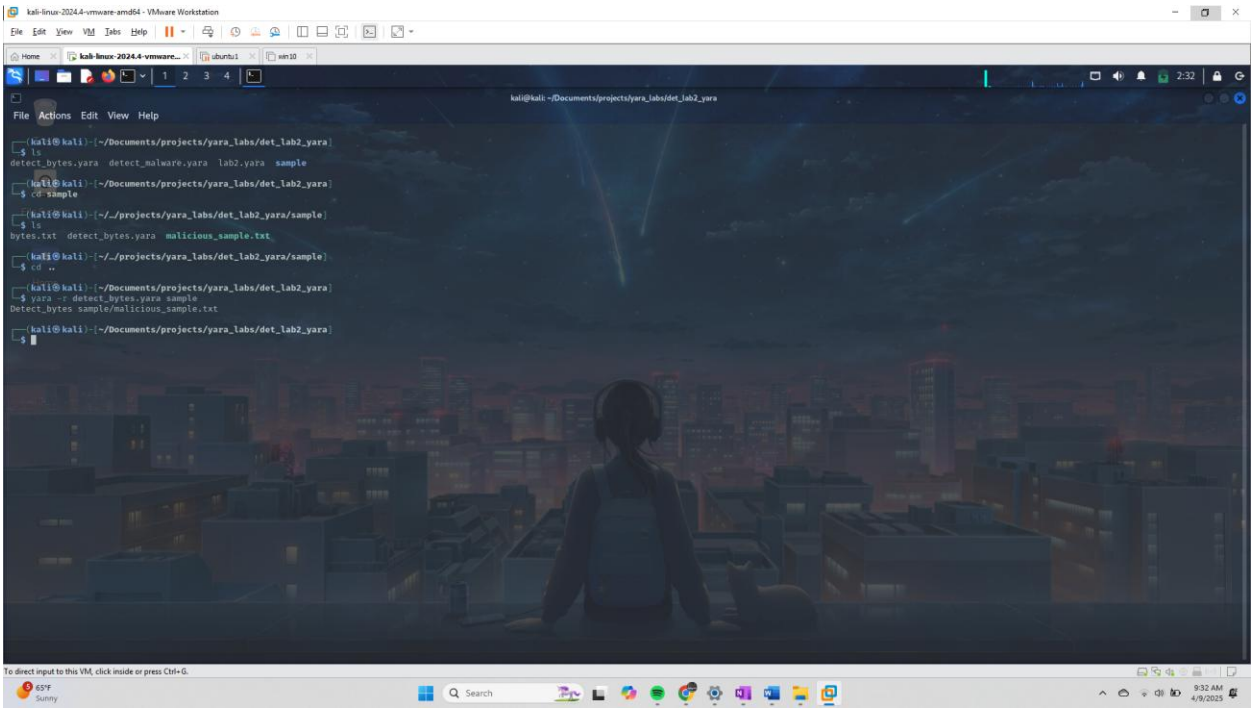

Figure 6 running the detect_bytes YARA rule

6.4 Behavioral Detection.

A YARA rule that detects the console output string is quite useful since it boasts the following advantages:

- Quick detection of malware artifacts printed during runtime.
- Useful during sandbox analysis or dynamic execution tracing.
- Useful in CI/CD pipeline scans or automated security checks.

However, it does pose the following risks:

- The malware can include dummy harmless messages to throw off detection.
- Legitimate tools may coincidentally print similar messages.
- If the malware does not print or if the malware runs silently, this rule won't help.
- Console strings can vary based on environment, language or obfuscation.

6.5 Rule Optimization.

The YARA rules I have written may generate false positives or take too long to scan large datasets, various techniques may be used to optimize the rules:

- Using **nocase** and wildcards when necessary to make exact matches faster.
- Minimizing the number of strings by combining related string into one if possible.
- Targeting rare, long or unique strings to reduce false positives.
- Avoiding large regex patterns or wildcards.
- Using the **filesize < 1MB** condition of you know the file size range.

7. CONCLUSION.

This lab aimed to provide a hands-on experience in detecting and analyzing malware using YARA rules. I encountered real-world challenges, such as rule optimization, reducing false positives and testing your detections against a range of sample files. By the end of the exercise I acquired a deeper understanding of how YARA rules work, how they can be applied to detect malicious activity and ow they can be used to improve malware detection in a real-world environment.