



## Table of Contents

<b>Introduction.....</b>	<b>3</b>
<b>Executive summary.....</b>	<b>3</b>
<b>Lab Objectives.....</b>	<b>3</b>
<b>Tools and resources used.....</b>	<b>3</b>
<b>Methodology.....</b>	<b>4</b>
Detecting strings in any file.....	4
Detectingmalware via strings and byte patterns.....	5
Detecting malware with regular expressions.....	5
Hash based detection.....	7
Avoiding false positive.....	8
Comprehensive malware detection rule.....	9
<b>Challenges and Learnings.....</b>	<b>10</b>
<b>Conclusion.....</b>	<b>11</b>

## Introduction

YARA is an open source tool that is used for malware detection and research. It is normally used by malware researchers to analyse and classify malware samples according to strings by bytes, hashes and patterns. YARA is an acronym which stands for Yet Another Recursive Acronym and it is used to create rules that define the characteristics of a malware that is being looked for. It normally has three sections that include the metadata, string and condition section.

Rules made out of YARA can be used by cybersecurity professionals to significantly enhance the ability to identify and respond to malware threats or even share with other professionals and researchers who will use it as references to improve the practice of malware analysis and reverse engineering.

## Executive summary

This lab focuses on how to write, optimize and test YARA rules using practical scenarios and sample files. It also focuses on YARA rule creation and file analysis for malware detection.

## Lab Objectives

1. Creating YARA rules
2. Analysing both benign and malicious files
3. Testing results

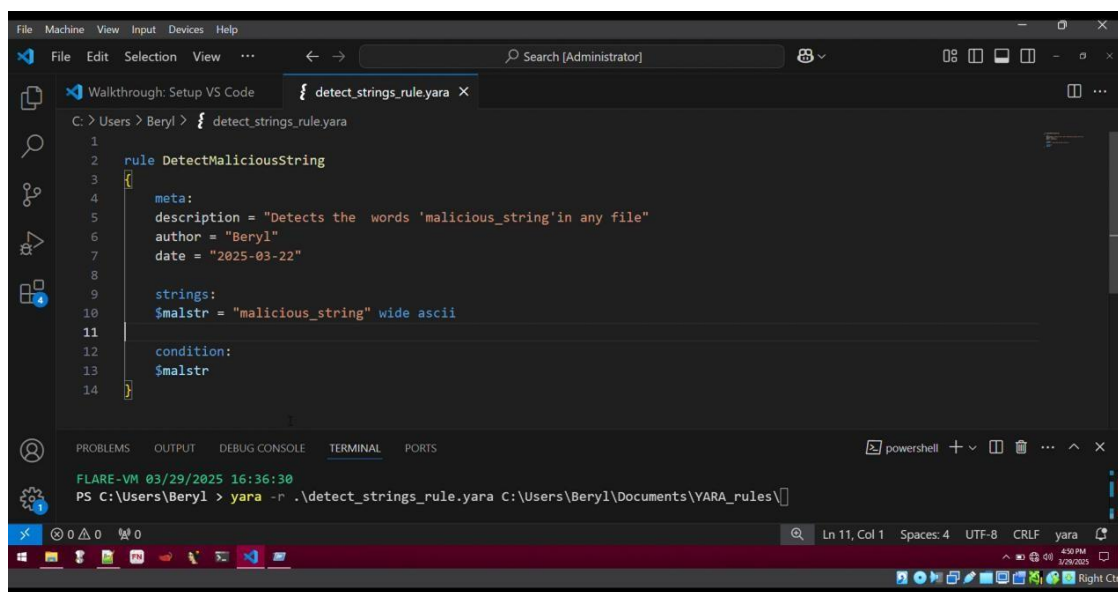
## Tools and resources used

- Visual Studio Code: code editor to write YARA rules
- Notepad: text editor to write YARA rules
- Terminal: to execute YARA commands
- YARA: versions 4.5.2
- Sample Files: with .exe and .txt extensions.

## Methodology

### Detecting strings in any file.

The first task was to create a YARA rule that detects the words ‘malicious\_string’ in any file. I used vs code to write the rules .

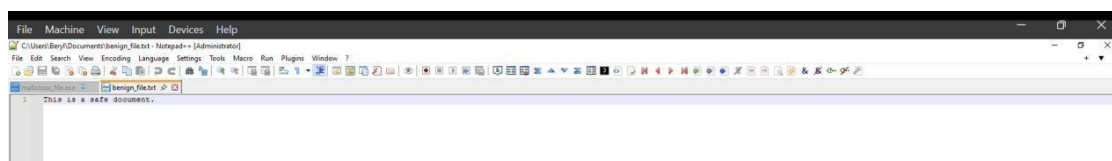


```

1 rule DetectMaliciousString
2 {
3   meta:
4     description = "Detects the words 'malicious_string' in any file"
5     author = "Beryl"
6     date = "2025-03-22"
7
8   strings:
9     $malstr = "malicious_string" wide ascii
10
11   condition:
12     $malstr
13
14 }
```

FLARE-VM 03/29/2025 16:36:30  
PS C:\Users\Beryl> yara -r .\detect\_strings\_rule.yara C:\Users\Beryl\Documents\YARA\_rules\

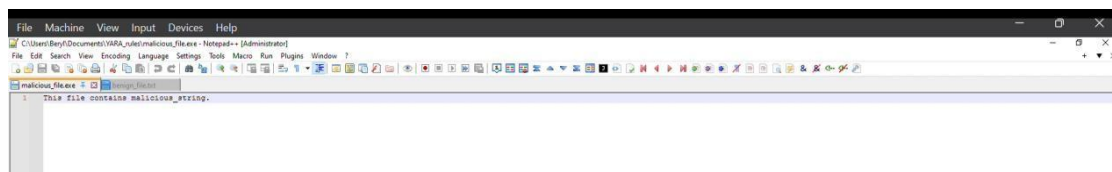
The rules contain strings that detects “malicious\_string” in any file. Wide modifier ensures the UTF-16 encoded strings match in the .exe file extension. On the other hand ascii modifier ensure normal ascii text matches. To test the yara rule, I created two files, one benign file that contains the sentence “This is a safe document”



```

1 This is a safe document.
```

and the another file that contains the words “malicious\_string”.



```

1 This file contains malicious_string.
```

Both files were stored one directory, and running the command “yara -r detect\_strings\_rule.yara and path to the directory” showed the malicious file.

```

1 rule DetectMaliciousString
2 {
3   meta:
4     description = "Detects the words 'malicious_string' in any file"
5     author = "Beryl"
6     date = "2025-03-22"
7
8   strings:
9     $malstr = "malicious_string" wide ascii
10
11   condition:
12     $malstr
13 }
14

```

```

FLARE-VM 03/29/2025 16:36:30
PS C:\Users\Beryl> yara -r .\detect_strings_rule.yara C:\Users\Beryl\Documents\YARA_rules\
DetectMaliciousString C:\Users\Beryl\Documents\YARA_rules\malicious_file.exe
FLARE-VM 03/29/2025 16:50:58
PS C:\Users\Beryl>

```

## Detecting malware via strings and byte patterns

The other task was to write a yara rule to detect malware based on the string “EvilSoftware” and a byte pattern of {E8 34 12 56}

```

1 rule EvilSoftwareDetection
2 {
3   meta:
4     description = "Detects EvilSoftware string and byte pattern"
5     author = "Beryl"
6     date = "2025-03-29"
7
8   strings:
9     $evil_string = "EvilSoftware" wide ascii
10    $evil_byte = {E8 34 12 56}
11
12   condition:
13     $evil_string or $evil_byte
14 }

```

```

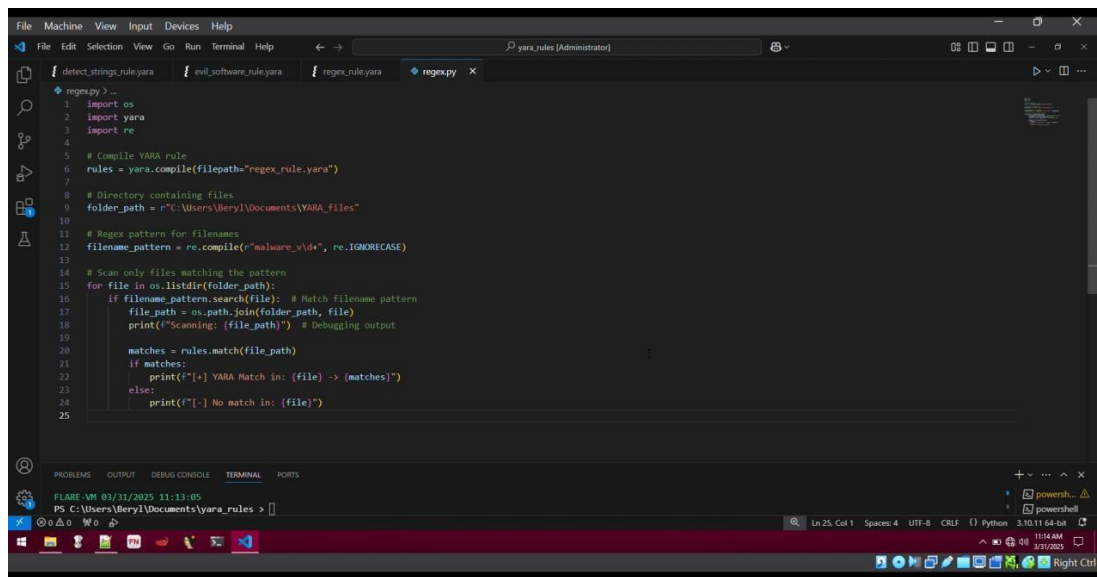
FLARE-VM 03/29/2025 18:16:49
PS C:\Users\Beryl\Documents\yara_rules> yara -r .\evil_software_rule.yara C:\Users\Beryl\Documents\YARA_files\
EvilSoftwareDetection C:\Users\Beryl\Documents\YARA_files\evil_sample.exe
FLARE-VM 03/29/2025 18:16:58
PS C:\Users\Beryl\Documents\yara_rules>

```

The rule detected the evilsoftware in a file called evil\_sample.exe which contained the string EvilSoftware in it.

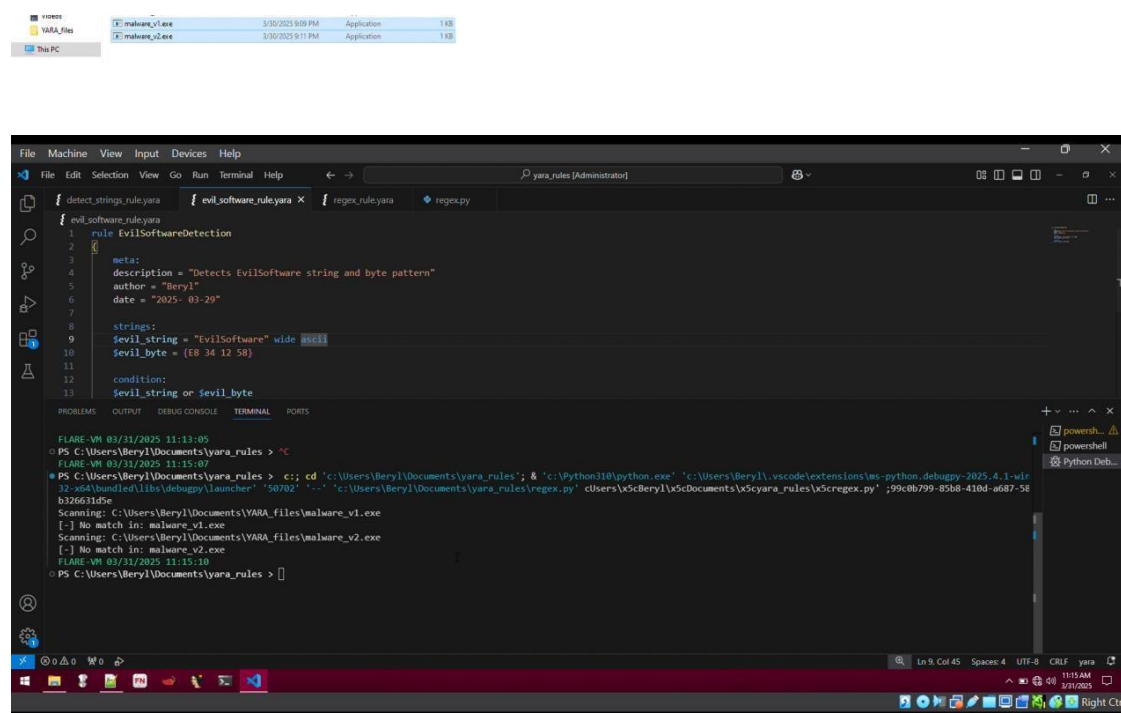
## Detecting malware with regular expressions

Since yara does not detect filenames, I needed to filter the filenames using python. Yara only reads the content in a file.



```
1 import os
2 import yara
3 import re
4
5 # Compile YARA rule
6 rules = yara.compile(filepath="regex_rule.yara")
7
8 # Directory containing files
9 folder_path = r"C:\Users\Beryl\Documents\YARA_files"
10
11 # Regex pattern for filenames
12 filename_pattern = re.compile(r"malware_v\d+", re.IGNORECASE)
13
14 # Scan only files matching the pattern
15 for file in os.listdir(folder_path):
16     if filename_pattern.search(file): # Match filename pattern
17         file_path = os.path.join(folder_path, file)
18         print(f"Scanning: {file_path}") # Debugging output
19
20         matches = rules.match(file_path)
21         if matches:
22             print(f"[+] YARA Match in: {file} -> {matches}")
23         else:
24             print(f"[-] No match in: {file}")
25
```

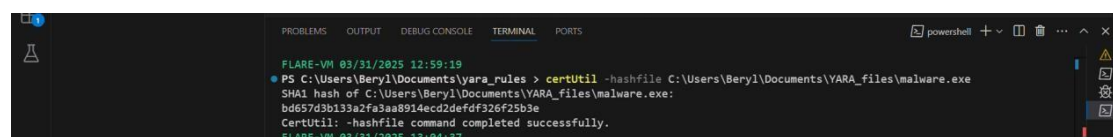
The code read only two files starting with malware\_v



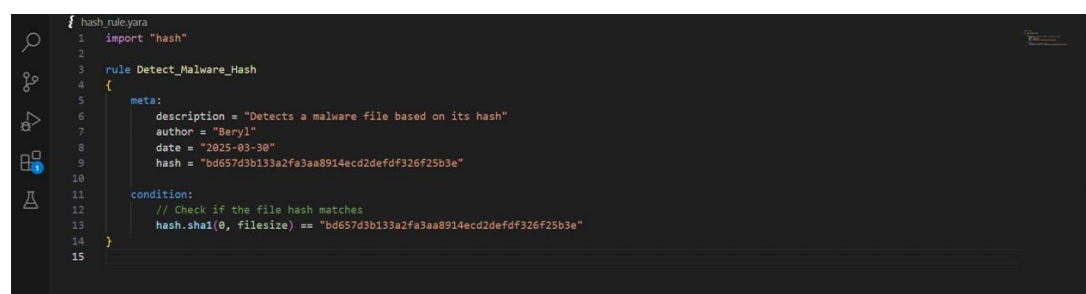
The rule outputs the matching file which is malware\_1.exe which had the sentence 'This is a malicious file with malicious\_behavior' excluding two file in the same directory benign\_file1.txt and malware\_2.exe which had the content 'This is a safe file. Success' and 'This file contains both malicious\_behavior and success' consecutively.

## Hash based detection

In a case of detecting a file hash. I generated a hash from my file using certutil. Yara uses sha1 instead of strings to detect a specific hash. In this case I had two files, one malware.exe which I generated the hash using certUtil



First I imported the hash module before writing the rule.



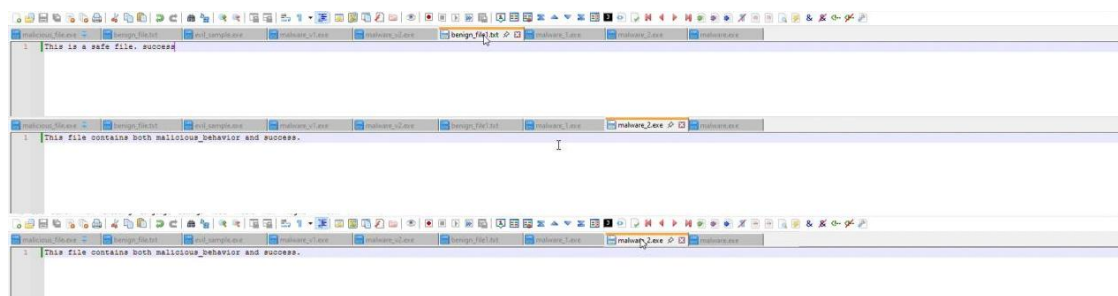
```

FLARE-VM 04/01/2025 11:32:04
PS C:\Users\Beryl\Documents\yara_rules > yara -r .\hash_rule.yara C:\Users\Beryl\Documents\YARA_files\
Detect_Malware_Hash C:\Users\Beryl\Documents\YARA_files\malware.exe
FLARE-VM 04/01/2025 11:35:58
PS C:\Users\Beryl\Documents\yara_rules >

```

## Avoiding false positive

Sometimes rules can show false positives, to avoid this one has to specify which strings should be excluded. In this case I wrote a rule that detects the string `malicious_file` but excludes the string `success`. Three files were to be detected, one was a benign file which contain the content 'This ia a safe file', the other one had the `malicious_behaviour` in it and the last one had both `malicious_behavior` and `success` strings in it. To avoid false positives I had to exclude `success` so that the output should be the file containing the string `malicious_behavior`.



The rule matched the file with `malware_behavior`.

```

File Machine View Input Devices Help
File Edit Selection View Go Run ...
yara_rules [Administrator]
{ detect_strings_rule.yara } { evil_software_rule.yara } { optimized_rule.yara x } { regex_rule.yara } { regex.py }
{
  rule DetectMaliciousBehavior
  {
    meta:
      description = "Detects malicious behaviur, excluding 'success'"
      author = "Beryl"
      date = "2025-03-30"

    strings:
      $malicious_behavior = "malicious_behavior" wide ascii
      $success = "success" wide ascii

    condition:
      $malicious_behavior and not $success
  }
}

FLARE-VM 03/31/2025 12:11:08
PS C:\Users\Beryl\Documents\yara_rules > yara -r .\optimized_rule.yara C:\Users\Beryl\Documents\YARA_files\

```



```

File Machine View Input Devices Help
File Edit Selection View Go Run ... yara_rules [Administrator]
{
  detect_strings_rule.yara
  evil_software_rule.yara
  optimized_rule.yara
  regex_rule.yara
  regex.py
}

optimized_rule.yara
1 rule DetectMaliciousBehavior
2 {
3   meta:
4     description = "Detects malicious behavior, excluding 'success'"
5     author = "Beryl"
6     date = "2025-03-30"
7
8   strings:
9     $malicious_behavior = "malicious behavior" wide ascii
10
11   condition:
12     $malicious_behavior
13
14 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
FLARE-VM 03/31/2025 12:11:08
PS C:\Users\Beryl\Documents\yara_rules> yara -r .\optimized_rule.yara C:\Users\Beryl\Documents\YARA_files\malware_1.exe
DetectMaliciousBehavior C:\Users\Beryl\Documents\YARA_files\malware_1.exe
FLARE-VM 03/31/2025 12:23:03
PS C:\Users\Beryl\Documents\yara_rules>

```

## Comprehensive malware detection rule

Finally a rule can have multiple conditions such as string, byte and hash to be excluded. In this case we wrote a rule that includes:

- The string "infected\_file".
- The byte sequence { 12 AB 34 CD }.
- The file was modified in the last 30 days.
- Exclude files from trusted sources based on hash.

```

File Machine View Input Devices Help
File Edit Selection View Go Run Terminal Help yara_rules [Administrator]
{
  detect_strings_rule.yara
  evil_software_rule.yara
  optimized_rule.yara
  regex_rule.yara
  hash_rule.yara
  infected_file.yara
  malicious_file.exe
  regex.py
}

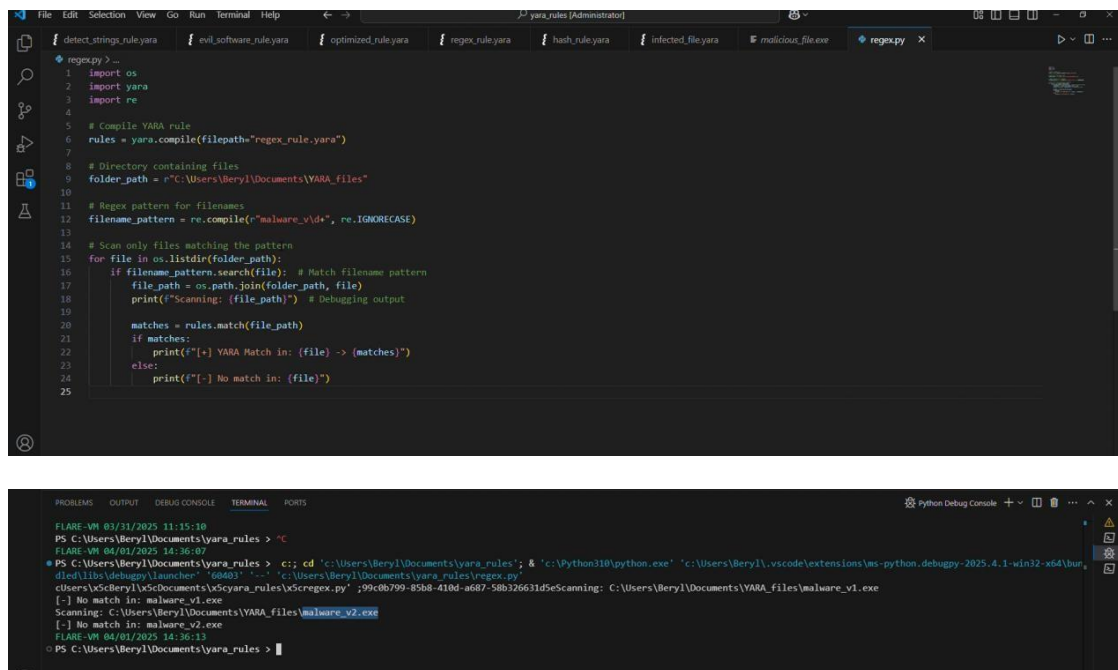
infected_file.yara
1 import "hash"
2 import "time"
3 import "pe"
4
5 rule DetectHarmfulFile
6 {
7   meta:
8     description = "Detecting a malicious file based on malicious string, file hash, time sequence and time of modification"
9     author = "Beryl"
10    date = "2025-03-30"
11
12   strings:
13     $malicious_string = "infected_file" wide ascii
14     $byte = { 12 AB 34 CD }
15
16   condition:
17     ($malicious_string or $byte) or (pe.timestamp >= 1706812800)
18
19 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
FLARE-VM 04/02/2025 07:15:44
PS C:\Users\Beryl\Documents\yara_rules> yara -r .\infected_file.yara C:\Users\Beryl\Documents\YARA_files\infected_file.exe
DetectHarmfulFile C:\Users\Beryl\Documents\YARA_files\infected_file.exe
FLARE-VM 04/02/2025 07:15:46
PS C:\Users\Beryl\Documents\yara_rules>

```

## Challenges and Learnings

Since yara alone does not detect filenames, it only reads the contents of the file. I had to write a python code that detects the regex.



The screenshot shows a VS Code editor with a file named `regex.py` open. The script imports `os`, `yara`, and `re`. It compiles a YARA rule from `regex_rule.yara` and scans a directory `C:\Users\Beryl\Documents\YARA_files` for files matching a regex pattern `malware_v\d+`. The terminal output shows the script running successfully, scanning `malware_v1.exe` and `malware_v2.exe`, and reporting no matches.

```

1 import os
2 import yara
3 import re
4
5 # Compile YARA rule
6 rules = yara.compile(filepath="regex_rule.yara")
7
8 # Directory containing files
9 folder_path = r"C:\Users\Beryl\Documents\YARA_files"
10
11 # Regex pattern for filenames
12 filename_pattern = re.compile(r"malware_v\d+", re.IGNORECASE)
13
14 # Scan only files matching the pattern
15 for file in os.listdir(folder_path):
16     if filename_pattern.search(file): # Match filename pattern
17         file_path = os.path.join(folder_path, file)
18         print(f"Scanning: {file_path}") # Debugging output
19
20         matches = rules.match(file_path)
21         if matches:
22             print(f"[+] YARA Match in: {file} -> {matches}")
23         else:
24             print(f"[-] No match in: {file}")
25

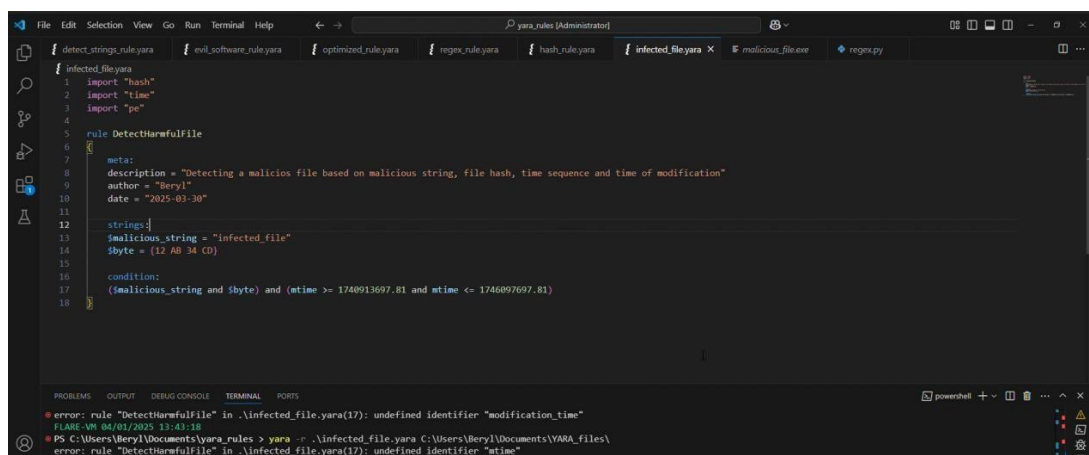
```

```

FLARE-VN 03/31/2025 11:15:10
PS C:\Users\Beryl\Documents\yara_rules > cd "C:\Users\Beryl\Documents\yara_rules"; & 'c:\Python310\python.exe' 'c:\Users\Beryl\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bin\debugpy_launcher' '50403' '-.' 'c:\Users\Beryl\Documents\yara_rules\regex.py'
FLARE-VN 04/01/2025 14:36:07
PS C:\Users\Beryl\Documents\yara_rules > c:; cd "C:\Users\Beryl\Documents\yara_rules"; & 'c:\Python310\python.exe' 'c:\Users\Beryl\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bin\debugpy_launcher' '50403' '-.' 'c:\Users\Beryl\Documents\yara_rules\regex.py'
c:\Users\Beryl\Documents\yara_rules\regex.py:99:OSError: [Errno 9] Bad file descriptor: C:\Users\Beryl\Documents\YARA_files\malware_v1.exe
Scanning: C:\Users\Beryl\Documents\YARA_files\malware_v2.exe
[-] No match in: malware_v2.exe
FLARE-VN 04/01/2025 14:36:13
PS C:\Users\Beryl\Documents\yara_rules >

```

Another challenge was including timestamp in a yara rule since yara does not support mtime.



The screenshot shows a VS Code editor with a file named `infected_file.yara` open. The rule is designed to detect malicious files based on a malicious string, file hash, time sequence, and time of modification. The terminal output shows an error: `error: rule "DetectHarmfulFile" in .\infected_file.yara(17): undefined identifier "modification_time"`.

```

1 import "hash"
2 import "time"
3 import "pe"
4
5 rule DetectHarmfulFile
6 {
7     meta:
8         description = "Detecting a malicious file based on malicious string, file hash, time sequence and time of modification"
9         author = "Beryl"
10        date = "2025-03-30"
11
12     strings:
13         $malicious_string = "infected_file"
14         $byte = {12 AB 34 CD}
15
16     condition:
17         ($malicious_string and $byte) and (mtime >= 1740913697.81 and mtime <= 1746097697.81)
18

```

```

error: rule "DetectHarmfulFile" in .\infected_file.yara(17): undefined identifier "modification_time"
FLARE-VN 04/01/2025 13:43:18
PS C:\Users\Beryl\Documents\yara_rules > yara -r .\infected_file.yara C:\Users\Beryl\Documents\YARA_files\
error: rule "DetectHarmfulFile" in .\infected_file.yara(17): undefined identifier "mtime"

```

## **Conclusion**

This lab teaches how to write basic YARA rules. With YARA you can create whatever malware discription you want. This is essential for understanding the nature of a threat and developing effective countermeasures. In the cybersecurity space YARA is used to identify presense of malware in a compromised system by searching indicators of compromise to analyse malicious files to determine if they are malicious. It can be kntergrated with tools like SIEM, IDS and antivirus softwares.