Table of Contents

Introduction

YARA is an open source tool that is used for malware detection and research. It is normaly used by malware researchers to analyse and classify malware samples according to stringsby 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 normaly has three sections that include the metadata, string and condition section.

Rules made out of YARA can be uses by cybersecurity professionals to significantly enhance the ability to 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

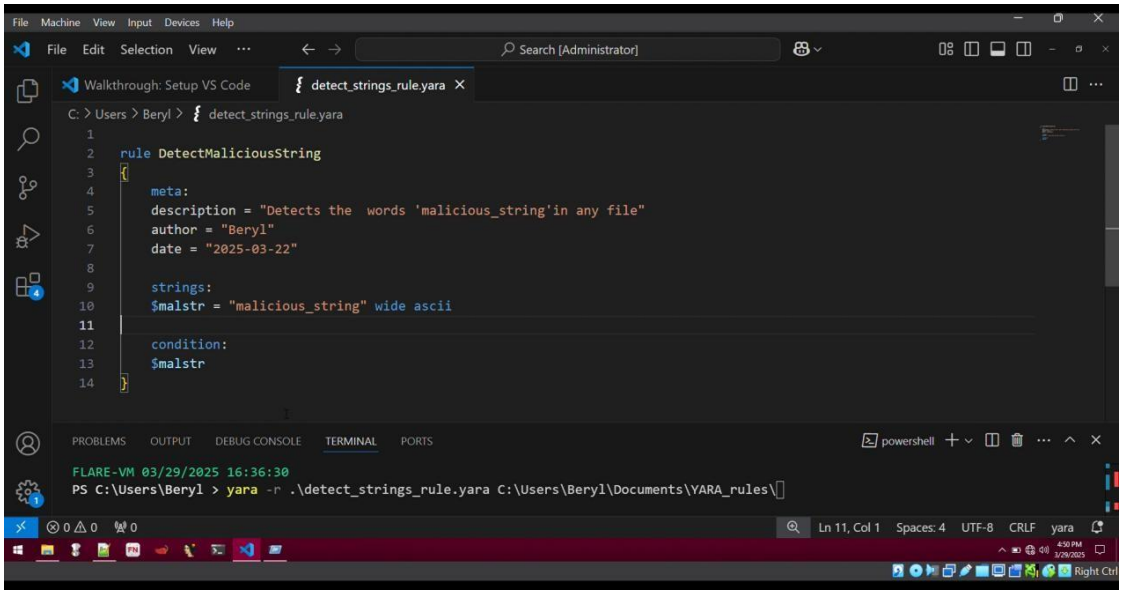
Tools and resources used

- 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.

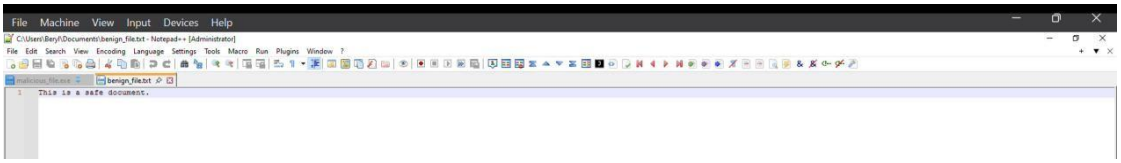
Methodology

Detecting strings in any file.

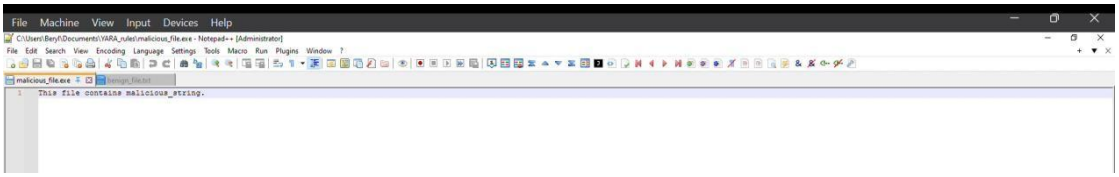
The first task was to create a YARA rule that detects the words ‘malicious_string ‘ in any file. I used vs code to write the rules .



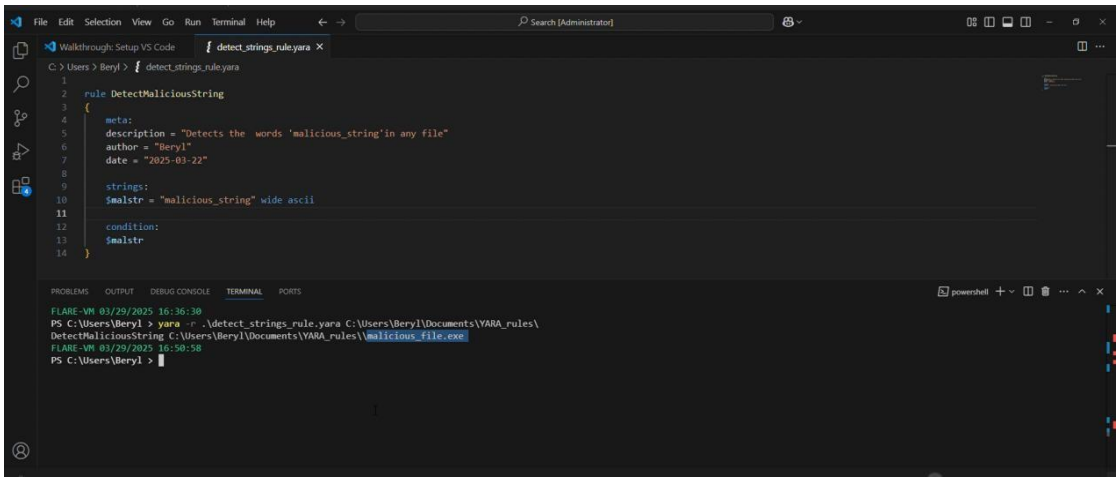
The rules contain strings that detects “malicious_string” in any file. Wide modifier ensures the UTF-16 encoded strings match in the .exe file extension. On the other hand ascii modifier ensure normal ascii text matches. To test the yara rule, I created two files, one benign file that contains the sentence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



and the another file that contains the words “malicious_string”.

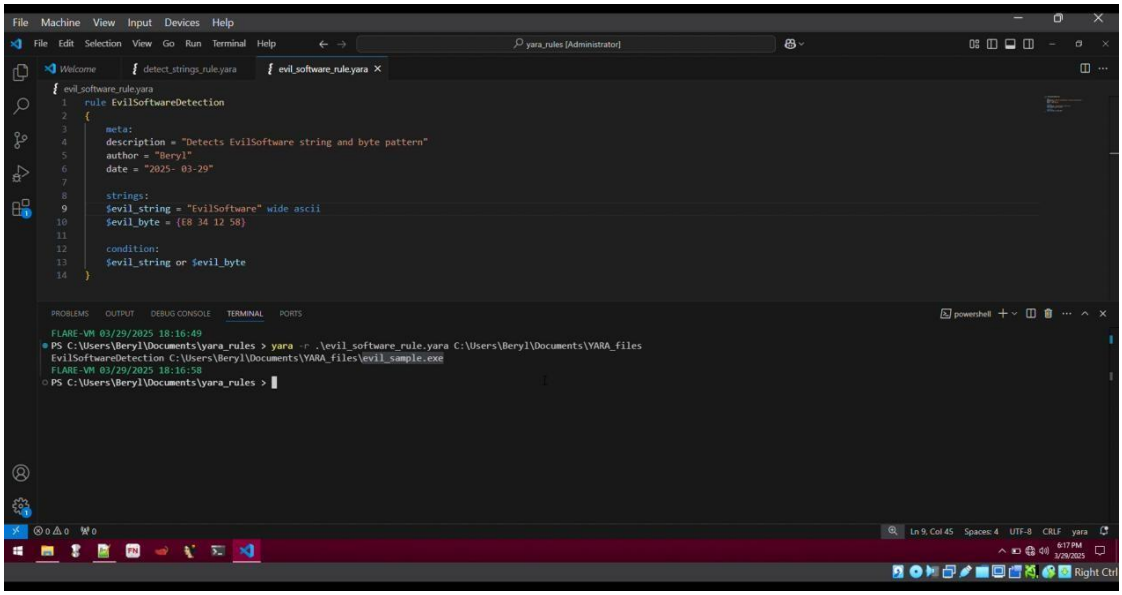


Both files were stored one directory, and running the command “yara -r detect_strings_rule.yara and path to the direcroty” showed the malicious file.



Detecting malware via strings and byte patterns

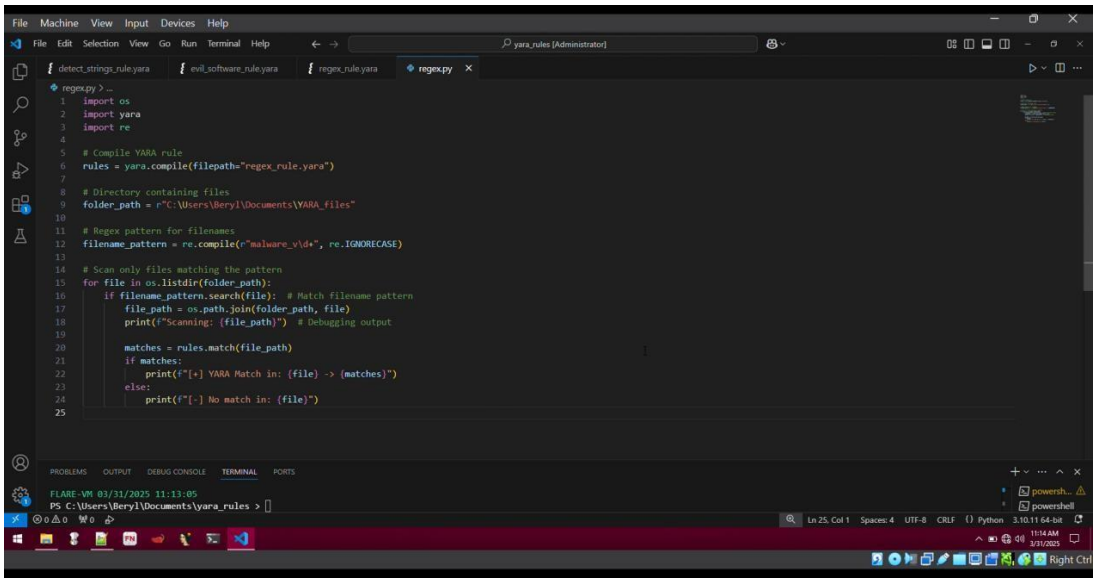
The other task was to write a yara rule to detect malware based on the string "EvilSoftware" and a byte pattern of {E8 34 12 56}



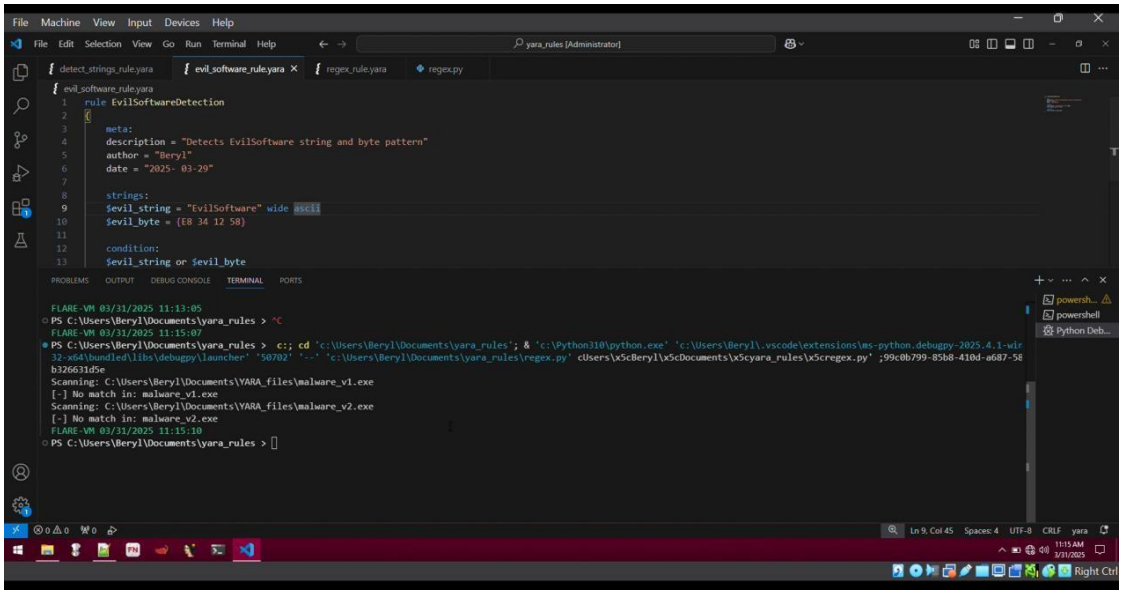
The rule detected the evilsoftware in a file called evil_sample.exe which contained the string EvilSoftware in it.

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.


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
regex_rule.yara
regex.py
regex.py > ...
import os
import yara
import re
# Compile YARA rule
rules = yara.compile(filepath="regex_rule.yara")
# Directory containing files
folder_path = r"C:\Users\Beryl\Documents\YARA_files"
# Regex pattern for filenames
filename_pattern = re.compile(r"malware_v\d+", re.IGNORECASE)
# Scan only files matching the pattern
for file in os.listdir(folder_path):
if filename_pattern.search(file): # Match filename pattern
file_path = os.path.join(folder_path, file)
print(f"Scanning: {file_path}") # Debugging output
matches = rules.match(file_path)
if matches:
print(f"[+] YARA Match in: {file} -> {matches}")
else:
print(f"[-] No match in: {file}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
FLARE-VM 03/31/2025 11:13:05
PS C:\Users\Beryl\Documents\yara_rules >
powersh...
powershell
Ln 25, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.11 64-bit
11:14 AM
3/31/2025
Right Ctrl

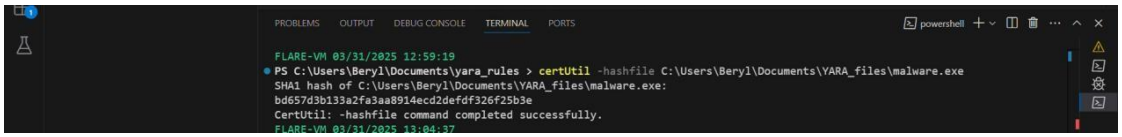
The code read only two files starting with malware_v



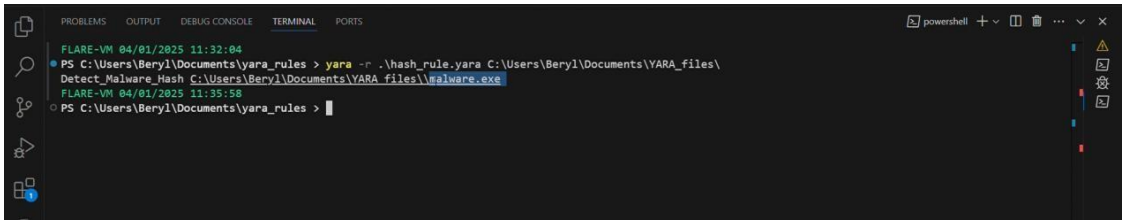
The rule outputs the matching file which is malware_1.exe which had the sentense ‘This is a malicious file with malicious_behavior’ excluding two file in the same directory benign_file1.txt and malware_2.exe which had the content ‘This is a safe file. Success’ and ‘This file contsins both malicious_behavior and success’ consecutively.

Hash based detection

In a case of detecting a file hash. I generated a hash from my file uisng certutil . Yara uses has 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

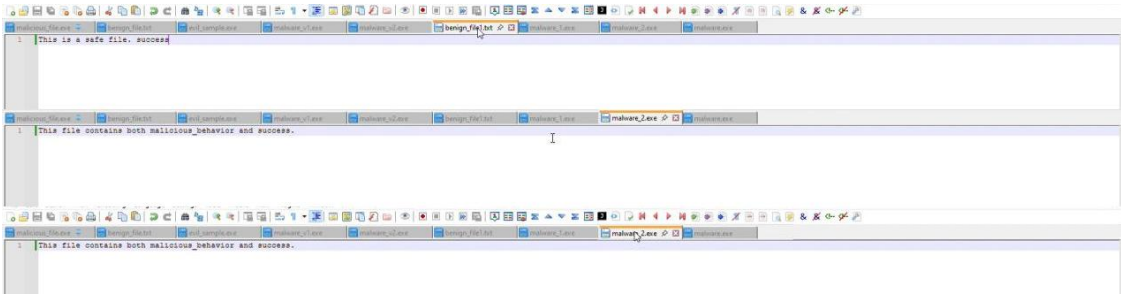


First I imported the hash module before writing the rule.

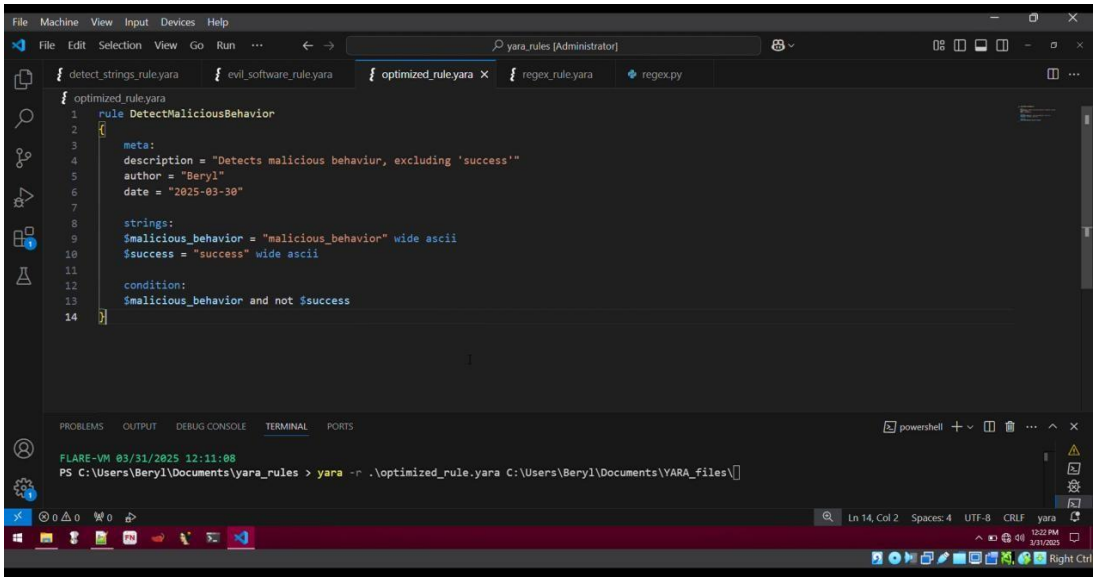


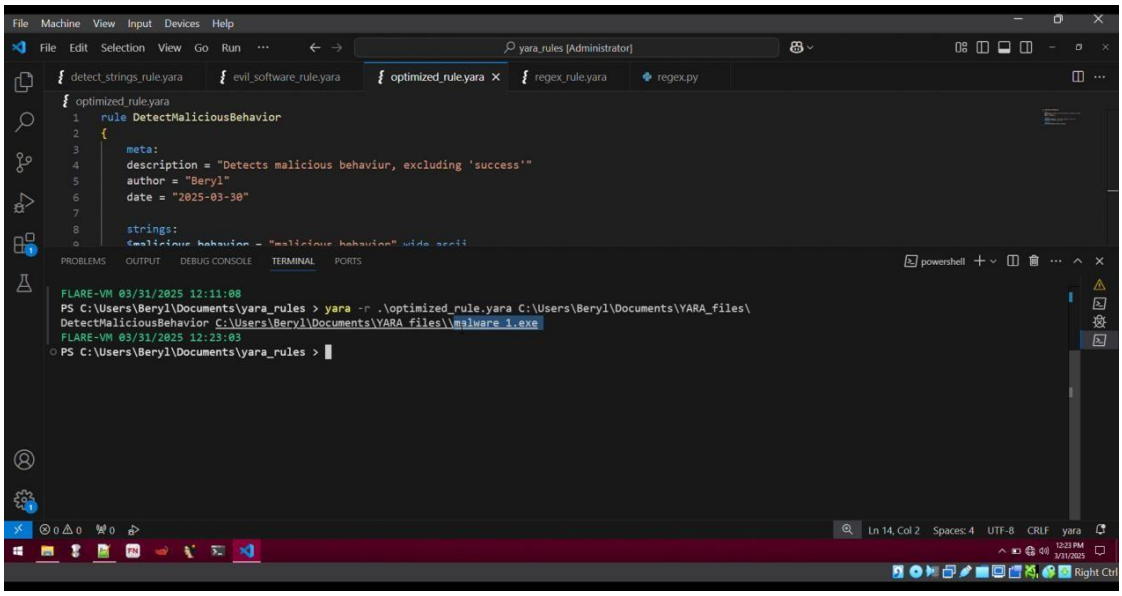
Avoiding false positive

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_behaviour.

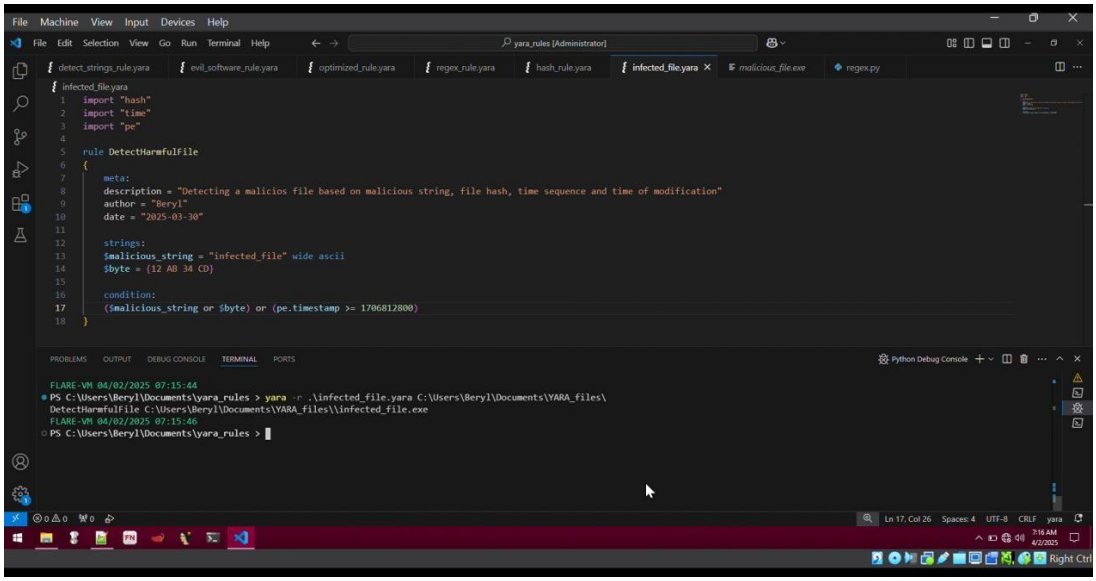




Comprehensive malware detection rule

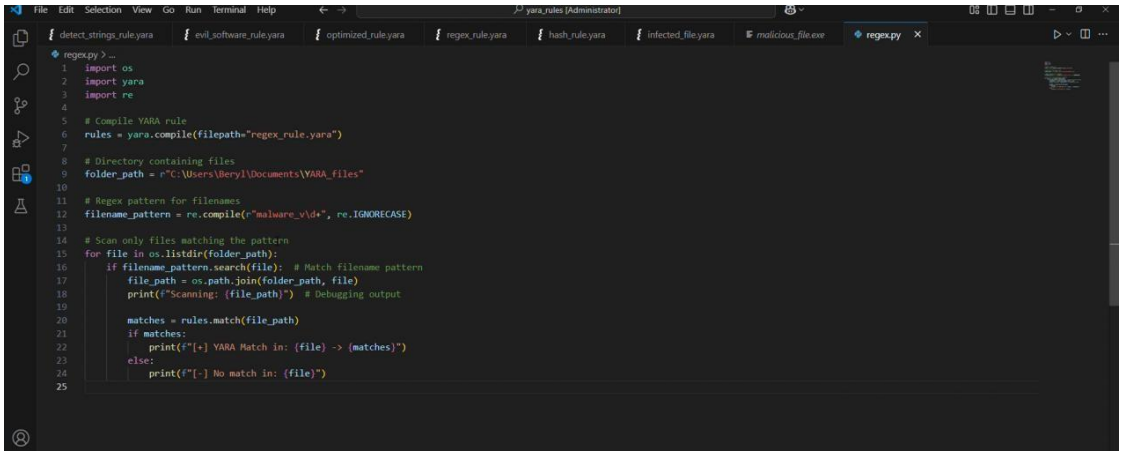
Finally a rule can have multiple conitions such as string, byte and hashe sto be excluded. In this case we wrote a rule that includes:

- 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.



Challenges and Learnings

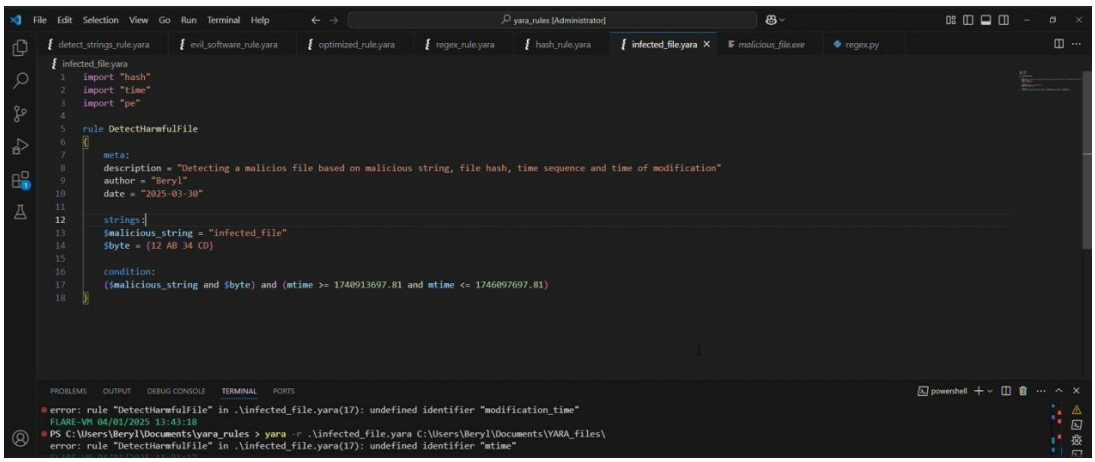
Since yara alone does not detect filenames, it only reads the contents of the file. I had to write a python code that detects the regex.



```python
import os
import yara
import re

# Compile YARA rule
rules = yara.compile(filepath="regex_rule.yara")

# Directory containing files
folder_path = r"C:\Users\Beryl\Documents\YARA_files"

# Regex pattern for filenames
filename_pattern = re.compile(r"malware_v\d+", re.IGNORECASE)

# Scan only files matching the pattern
for file in os.listdir(folder_path):
    if filename_pattern.search(file):  # Match filename pattern
        file_path = os.path.join(folder_path, file)
        print(f"Scanning: {file_path}")  # Debugging output

        matches = rules.match(file_path)
        if matches:
            print(f"[+] YARA Match in: {file} -> {matches}")
        else:
            print(f"[-] No match in: {file}")
```

```
FLARE-VM 03/31/2025 11:15:10
PS C:\Users\Beryl\Documents\yara_rules > ^C
FLARE-VM 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\bundled\libs\debugpy\launcher' '60403' '--' 'c:\Users\Beryl\Documents\yara_rules\regex.py'
cUsers\x5cBeryl\x5cDocuments\x5cyara_rules\x5cregex.py' ;99c0b799-85b8-410d-a687-58b326631d5e
Scanning: C:\Users\Beryl\Documents\YARA_files\malware_v1.exe
[-] No match in: malware_v1.exe
Scanning: C:\Users\Beryl\Documents\YARA_files\malware_v2.exe
[-] No match in: malware_v2.exe
FLARE-VM 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.



```
import "hash"
import "time"
import "pe"

rule DetectHarmfulFile
{
    meta:
    description = "Detecting a malicios file based on malicious string, file hash, time sequence and time of modification"
    author = "Beryl"
    date = "2025-03-30"

    strings:
    $malicious_string = "infected_file"
    $byte = {12 AB 34 CD}

    condition:
    ($malicious_string and $byte) and (mtime >= 1740913697.81 and mtime <= 1746097697.81)
}
```

```
error: rule "DetectHarmfulFile" in .\infected_file.yara(17): undefined identifier "modification_time"
FLARE-VM 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.