CCCY 324 - Security Architecture and Engineering

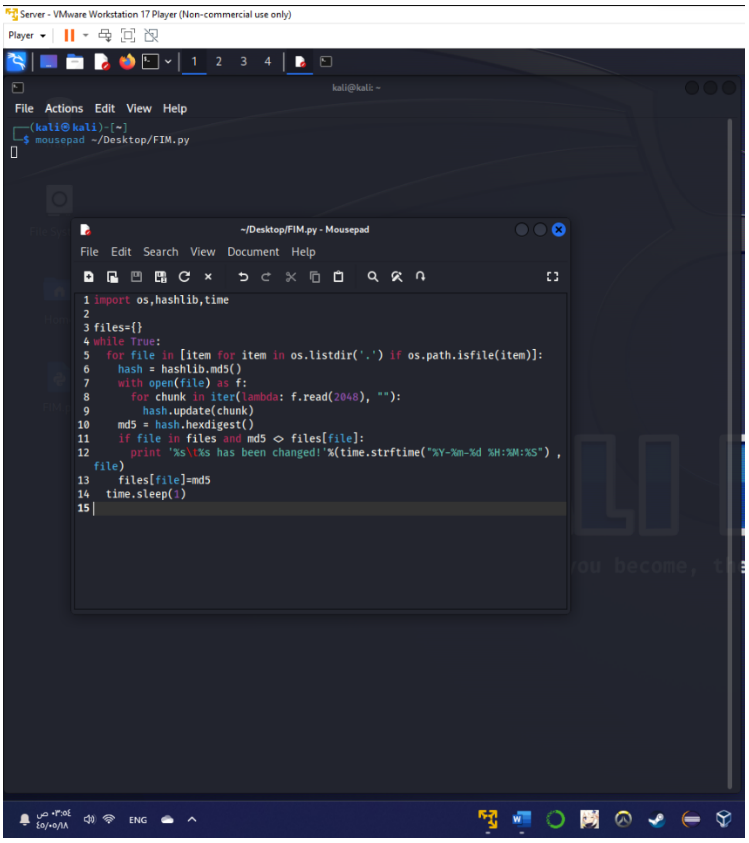
Lab Assignment

Raghad lafe – 2111941

Question 1: Do Part-1 task and include a detailed report with screenshots for each step.

1- Create FIM.py File: Open the Kali terminal and create a new file named FIM.py on the desktop using the mousepad editor.

Copy and paste the provided Python code into the FIM.py file, save it, and close the editor.



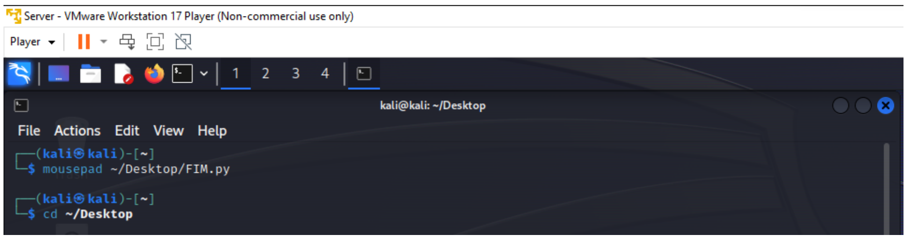
```
┌──(kali㉿kali)-[~]
└─$ mousepad ~/Desktop/FIM.py
```

~/Desktop/FIM.py - Mousepad

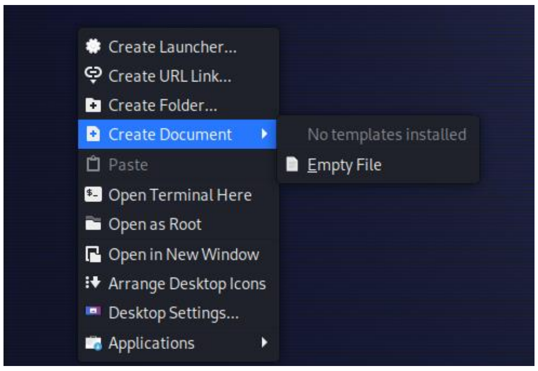
```python
import os,hashlib,time

files={}
while True:
  for file in [item for item in os.listdir('.') if os.path.isfile(item)]:
    hash = hashlib.md5()
    with open(file) as f:
      for chunk in iter(lambda: f.read(2048), ""):
        hash.update(chunk)
    md5 = hash.hexdigest()
    if file in files and md5 <> files[file]:
      print '%s\t%s has been changed!'%(time.strftime("%Y-%m-%d %H:%M:%S") , file)
    files[file]=md5
  time.sleep(1)

```

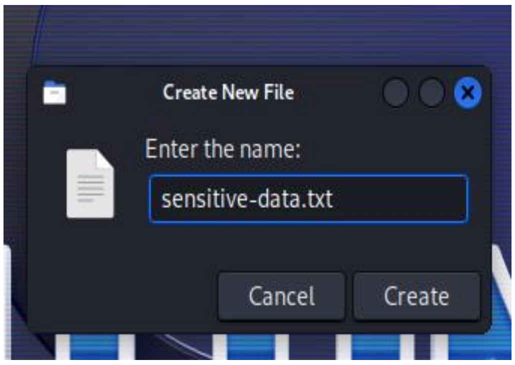
2- Create sensitive-data.txt File: Navigate to the Desktop directory in the terminal.

Right-click on the desktop and create a new file named sensitive-data.txt.

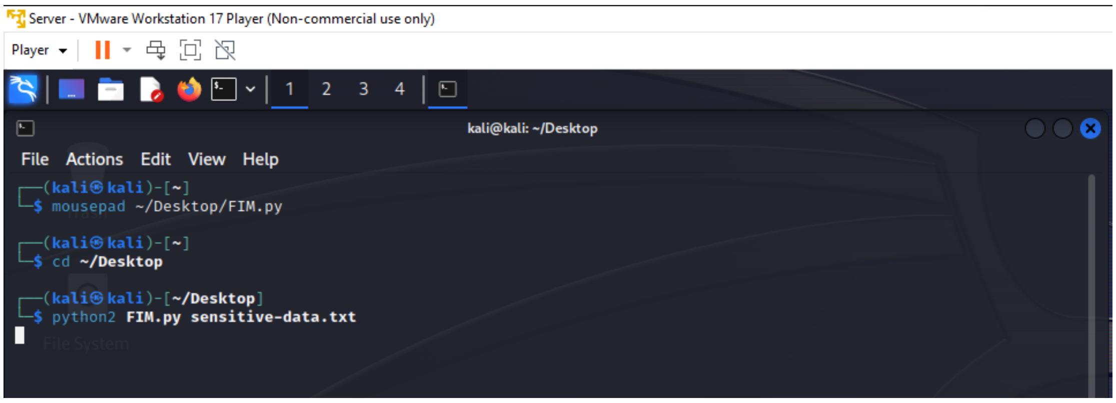


```
┌──(kali㉿kali)-[~]
└─$ mousepad ~/Desktop/FIM.py

┌──(kali㉿kali)-[~]
└─$ cd ~/Desktop
```



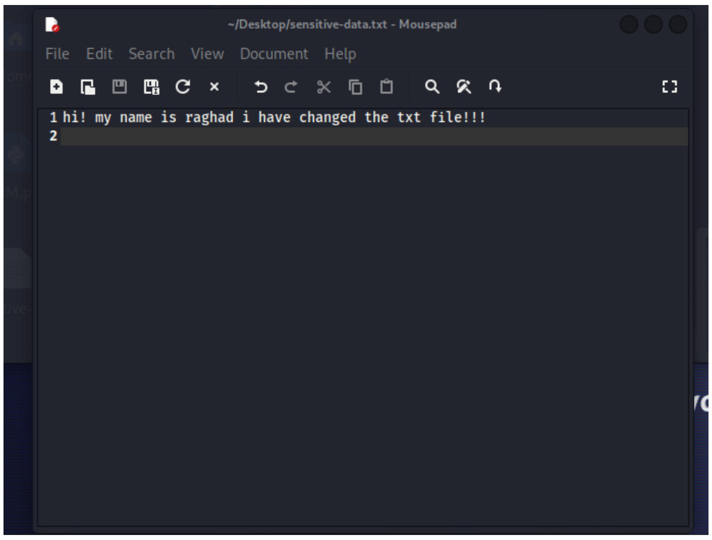


3- Run FIM.py to Monitor Changes: Run the Python program to monitor changes in the sensitive-data.txt file.


Server - VMware Workstation 17 Player (Non-commercial use only)
Player
kali@kali: ~/Desktop
File Actions Edit View Help

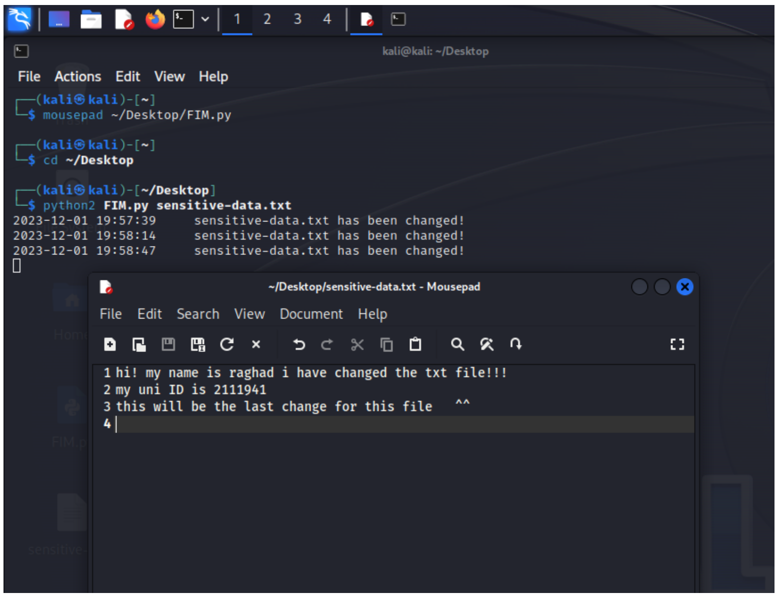
```
┌──(kali㉿kali)-[~]
└─$ mousepad ~/Desktop/FIM.py

┌──(kali㉿kali)-[~]
└─$ cd ~/Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ python2 FIM.py sensitive-data.txt
```

4- Edit sensitive-data.txt: Open sensitive-data.txt on the Kali desktop, write some text, and save the file.


~/Desktop/sensitive-data.txt - Mousepad
File Edit Search View Document Help

```
hi! my name is raghad i have changed the txt file!!!
```

5- Repeat the Process: Repeat the process of editing and saving sensitive-data.txt multiple times to see how FIM.py responds to changes.


kali@kali: ~/Desktop
File Actions Edit View Help

```
┌──(kali㉿kali)-[~]
└─$ mousepad ~/Desktop/FIM.py

┌──(kali㉿kali)-[~]
└─$ cd ~/Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ python2 FIM.py sensitive-data.txt
2023-12-01 19:57:39     sensitive-data.txt has been changed!
2023-12-01 19:58:14     sensitive-data.txt has been changed!
2023-12-01 19:58:47     sensitive-data.txt has been changed!
```

~/Desktop/sensitive-data.txt - Mousepad
File Edit Search View Document Help

```
hi! my name is raghad i have changed the txt file!!!
my uni ID is 2111941
this will be the last change for this file    ^^
```

Question 2: Explain the code in FIM.py and how it works?

The FIM script is created to identify any modifications made to files in the current directory. It uses an infinite loop to continuously monitor the files and calculates the MD5 hash for each file, which is a unique fingerprint based on the file's content. If the file has been previously recorded, the script compares the current MD5 hash with the stored one. If there is a difference, indicating a change in the file's content, it prints a timestamped message to the console, alerting the user about the alteration. The script uses the hashlib library for MD5 hashing, the os library for file-related operations, and time for timestamping and introducing a brief delay between iterations. This simple file monitoring mechanism provides a basic level of integrity checking, which is suitable for detecting unauthorized modifications or potential security incidents in a specified directory.

Question 3: Can this program be enhanced to send an email message (alert) to the data owner/administrator if any change happened?

Yes, the script only displays a message on the console when a change occurs. To enable email alerts, we will integrate an email-sending mechanism into the script. This can be done by utilizing a Python library like (smtplib) to send emails. The email notification can contain important information such as the timestamp of the change and the name of the modified file. It is crucial to securely include the necessary email configurations, such as the sender's email address, password, and the recipient's email address, within the script. By implementing email alerts, the script becomes more proactive in informing the relevant individuals about potential unauthorized changes, ensuring prompt action and resolution.

Here are the basic steps:

1- Add the Needed Libraries like (smtplib), (MIMEText)
2- Set up Email field: Create variables to store all the necessary information for your email configuration.
3- We need to create a function that can send emails.
4- Send Email in the Change Detection Section.

Question 4: In your own words, explain in detail what does the following Snort rule do?

The Snort rule provided is created to generate an alert whenever it identifies a potential Cross-Site Scripting (XSS) attempt. And it will display this msg to the user "XSS attempt: script injection detected ". First the (SID) stand for the unique identifier assigned to this Snort rule which is 1000007. This rule is set to trigger the alert for TCP traffic originating from any IP address on the external network ($EXTERNAL_NET) and directed towards any IP address on the home network ($HOME_NET). The connection must be in an established state for the alert to be generated. the rule searches for the content "Alert(document.cookie)" in the payload, which signifies an attempt to execute a JavaScript alert function that exposes the contents of the document's cookie. The classification of this alert tags it as an attempted admin attack, suggesting that the observed activity may be linked to compromising administrative functionality.

A-Technique 1: Command and Control Channel (ATT&CK technique T1041)[1]

This technique involves establishing a communication channel (C2 channel) between external adversaries and the compromised network. Malware within the victim's network connects to external servers to receive instructions and transfer exfiltrated data. The C2 traffic often blends with regular user traffic to avoid detection.

B-Examples (Tools):

Attor:

Attor is a sophisticated remote access trojan (RAT) known for its use in targeted attacks.

Usage: It establishes a covert C2 channel, enabling adversaries to issue commands and exfiltrate data.

Kessel:

Kessel is a malware strain associated with advanced persistent threats (APTs).

Usage: It communicates with external servers, facilitating data exfiltration and remote control.

C-Mitigation:

Employ network traffic analysis to detect unusual patterns indicative of C2 communication.

Utilize intrusion detection and prevention systems to identify and block C2 traffic.

Regularly update and patch systems to address vulnerabilities exploited by C2 malware.

D-Detection:

Monitor for anomalies in network traffic, such as unexpected connections to external servers.

Implement signature-based detection for known C2 malware.

Use behavior analysis tools to identify patterns indicative of C2 activity.

A-Technique 2: Exfiltration through Protocols (ATT&CK technique T1048)

Attackers leverage common internet protocols (DNS, HTTP, HTTPS, FTP) to exfiltrate data. They may embed data within protocol packets, exploiting the inherent functionality of these protocols.

B-Examples (Tools):

cURL (Command-line Tool):

cURL is a command-line tool for transferring data with URL syntax.

Usage: Attackers can use cURL to directly send data through HTTP POST requests, facilitating exfiltration.

DNS Tunneling:

DNS tunneling involves using DNS queries to carry data, often segmented, compressed, and encrypted.

Usage: Malware within the victim's network sends DNS queries containing exfiltrated data to a rogue DNS server.

C-Mitigation:

Monitor and filter outbound DNS traffic for anomalies and large data payloads.

Implement deep packet inspection to detect unusual patterns in HTTP, HTTPS, FTP traffic.

Consider blocking connections with newly registered domains.

D-Detection:

Analyze DNS traffic for irregular patterns, especially in terms of query types and response sizes.

Monitor for unusual HTTP/HTTPS/FTP traffic, such as large data transfers.

Employ anomaly-based detection to identify deviations from normal protocol usage.

A-Technique 3: Exfiltration to Devices in Proximity (ATT&CK technique T1011)

This technique involves an insider, often an employee, copying files to removable storage devices or using radio frequency channels like Bluetooth to transfer data to their own devices.

B-Examples (Tools):

USB Storage Devices:

Employees copy sensitive data to USB drives or external hard disks.

Usage: Convenient for employees with physical access to systems, posing an insider threat.

Bluetooth File Transfer:

Using Bluetooth to transfer files between devices in short range.

Usage: Enables data exfiltration without reliance on the organization's network.

C-Mitigation:

Restrict the use of removable storage devices in sensitive environments.

Monitor and control Bluetooth usage within the organization.

Implement data loss prevention (DLP) policies to block unauthorized transfers.

D-Detection:

Monitor file copy activities, especially to external storage devices.

Employ endpoint security solutions to detect and block Bluetooth-based file transfers.

Implement user behavior analytics to identify suspicious patterns of data copying.

Reference

1- https://www.manageengine.com/products/eventlog/cyber-security/data-exfiltration.html