

# Introduction to Computer Security

## Detection-Phase 5

### Team Members:

Pooja Peechara - 11663837

Rachana Reddy Sunki - 11709719

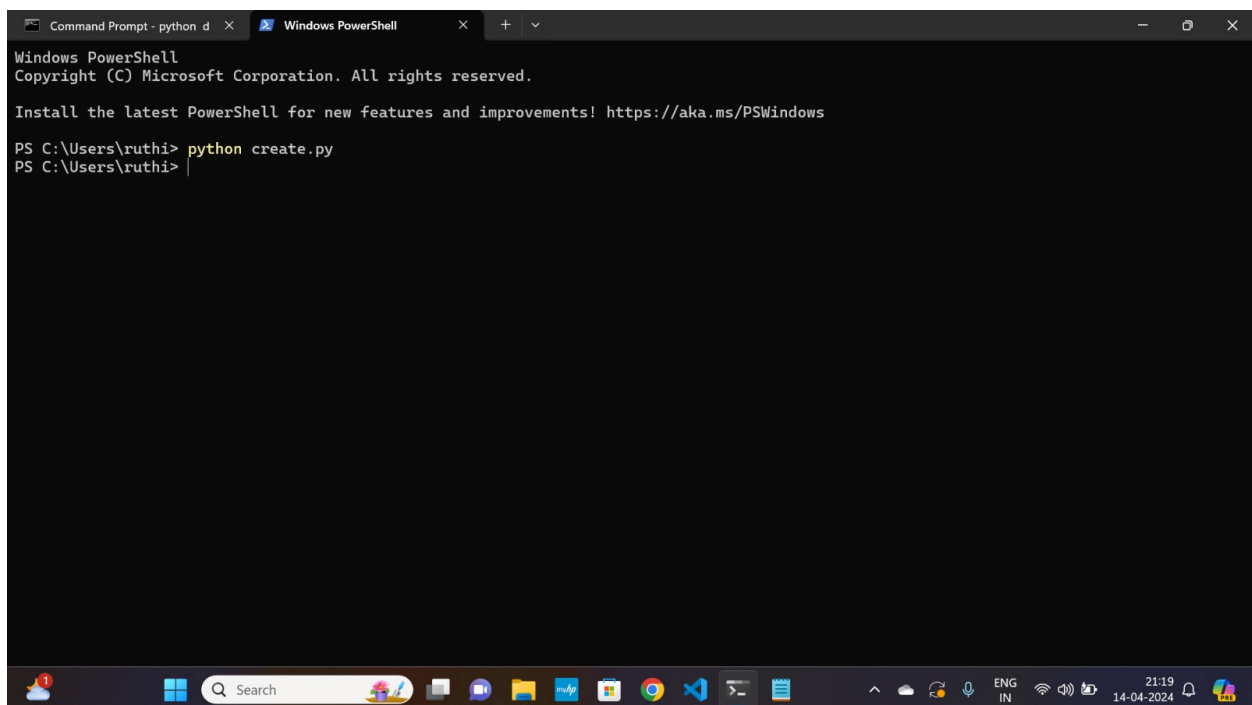
Mounika Nuchu - 11653658

Lagadapati Kondarao - 11661743

Yenugu Ruthiksha Reddy - 11714976

**Introduction:** In the previous phases we have monitored the logs by using the Python Library Watchdog. Here, We will be detecting the unusual activities based on the investigation performed on the files.

To create an unusual activity from different sources, we have created a file named create.py



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ruthi> python create.py
PS C:\Users\ruthi>
```

The screenshot shows a Windows PowerShell terminal window. The title bar indicates it's a 'Windows PowerShell' window. The terminal displays the standard PowerShell startup text, including the copyright notice and a link to download the latest version. The user has entered the command 'python create.py' at the prompt, and the cursor is now on the next line, ready for further input. The taskbar at the bottom shows various application icons and the system clock indicating 21:19 on 14-04-2024.

## Code:-

```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Detection_2.py 3 create.py Release Notes: 1.88.0
C:\Users> ruthi > Downloads > Detection_2.py > ...

1 import time
2 import pandas as pd
3 from watchdog.observers import Observer
4 from watchdog.events import FileSystemEventHandler
5 from collections import defaultdict, deque
6 import datetime
7
8 class Detection(FileSystemEventHandler):
9     def __init__(self):
10         # Using defaultdict to automatically handle any folder not previously accessed
11         self.folder_access_log = defaultdict(lambda: deque(maxlen=100))
12
13     def on_modified(self, event):
14         if not event.is_directory:
15             self.log(event.src_path, 'modified')
16
17     def on_created(self, event):
18         if not event.is_directory:
19             self.log(event.src_path, 'created')
20
21     def log(self, path, event_type):
22         # Extract folder path from file path
23         folder_path = '/'.join(path.split('/')[:-1])
24         event_timestamp = pd.Timestamp.now()
25
26         # Log file access event with the current timestamp
27         new_event = {'timestamp': event_timestamp, 'file_path': path, 'event_type': event_type}
28         self.folder_access_log[folder_path].append(new_event)
```

2)

```
File Edit Selection View Go Run ... Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More

Detection_2.py 3 create.py Release Notes: 1.88.0
C:\Users> ruthi > Downloads > Detection_2.py > ...

8 class Detection(FileSystemEventHandler):
9     def on_created(self, event):
10         self.log(event.src_path, 'created')
11
12     def log(self, path, event_type):
13         # Extract folder path from file path
14         folder_path = '/'.join(path.split('/')[:-1])
15         event_timestamp = pd.Timestamp.now()
16
17         # Log file access event with the current timestamp
18         new_event = {'timestamp': event_timestamp, 'file_path': path, 'event_type': event_type}
19         self.folder_access_log[folder_path].append(new_event)
20
21         # Check for anomaly after logging the event
22         self.anomaly(folder_path)
23
24     def anomaly(self, folder_path):
25         # Analyze recent events within the last 10 seconds for the specified folder
26         current_time = pd.Timestamp.now()
27         recent_events = [event for event in self.folder_access_log[folder_path] if event['timestamp'] > current_time - pd.Timedelta(seconds=10)]
28
29         # Detect high frequency of file operations
30         if len(recent_events) > 5:
31             print(f"Threat detected in {folder_path}: high frequency of file operations detected.")
32             self.log_for_investigation(folder_path, recent_events)
33
34     def log_for_investigation(self, folder_path, events):
35         # Log details to a file or a monitoring system for further investigation
36         with open(f"event_log_{folder_path}.txt", "a") as file:
```

3)

The screenshot shows a Windows 11 desktop. In the foreground, a Windows Security notification is displayed, stating: "Windows Security blocked an app from accessing your files. The app is named 'Detection\_2.py' and is located in the folder 'Downloads'. It is a Python script. To help protect your privacy, Outlook prevented automatic download of some pictures in this message." The notification includes a "More info" link and a "Turn on" button.

In the background, a VS Code window is open, showing a Python script named "Detection\_2.py". The script is designed to monitor file system events in a specified directory. It uses the "FileSystemEventHandler" class from the "watchdog.observers" module. The script defines a function "anomaly" that checks if the number of recent events exceeds a threshold (5). If it does, it prints a message and logs the event. It also defines a "log\_for\_investigation" function that writes details to a log file. The script then sets up an observer to monitor the directory "C:\Users\ruthi\OneDrive\Desktop\Critical\_1" and starts the observation.

```

8 class Detection(FileSystemEventHandler):
33     def anomaly(self, folder_path):
39         if len(recent_events) > 5:
40             print(f"Threat detected in {folder_path}: high frequency of file operations detected.")
41             self.log_for_investigation(folder_path, recent_events)
42
43     def log_for_investigation(self, folder_path, events):
44         # Log details to a file or a monitoring system for further investigation
45         with open("suspicious_activity_log.txt", "a") as file:
46             file.write(f"Unusual activity detected on {datetime.datetime.now()} for folder {folder_path}:\n")
47             for event in events:
48                 file.write(f'{event["timestamp"]} - {event["event_type"]} - {event["file_path"]}\n')
49             file.write("\n")
50
51 path_to_monitor = "C:\\Users\\ruthi\\OneDrive\\Desktop\\Critical_1"
52 event_handler = Detection()
53 observer = Observer()
54 observer.schedule(event_handler, path_to_monitor, recursive=True)
55
56 observer.start()
57 try:
58     while True:
59         time.sleep(1)
60 except KeyboardInterrupt:
61     observer.stop()
62 observer.join()
63

```

In our code, we have imported the watchdog library along with creation of a new class called as `Detection` from the `FileSystemEventHandler` present in the `watchdog` module.

A screenshot of a Windows Command Prompt window titled "Command Prompt - python". The window shows the output of running a Python script named "detection\_2.py". The output consists of 20 identical lines, each stating "Threat detected in : high frequency of file operations detected.". The command prompt's title bar includes standard Windows icons for maximize, minimize, and close. At the bottom of the screen, the Windows taskbar is visible, showing various application icons and the system clock indicating 21:19 on 14-04-2024.





**Conclusion:-** By implementing this phase, we have developed script which displays the logs of the suspicious threats by detecting the high frequency of the files modifications/changes. The log file has the logs of each detected events until the code is run again.