STEP 6: RANSOMWARE MITIGATION AND RESPONSE PLAN

Group#8

- **Rashmika Adusumilli – 11770425**
- **Esther Eze - 11886695**
- **Mounika Chavagani - 11737704**
- **Sai Teja Uppu – 11691244**

Mitigation

Following our successful file monitoring and ransomware detection alert based on policy violations, the next critical step is rapid response through mitigation measures. Upon detecting suspicious ransomware behavior (as identified through file extension monitoring, ransom note detection, and burst file modifications), the system automatically initiates mitigation actions to minimize damage. We will implement a defense-in-depth approach, alongside backup and recovery recommendations to ensure resilience even if one layer fails

Mitigation Strategy

Our system uses a two-pronged immediate mitigation approach:

- ✓ Killing Suspicious Processes

The system inspects all running processes. Any process that has written more than 10MB to disk (considered a suspicious behavior for ransomware encryption) is forcefully terminated using **psutil.** This prevents further file encryption or malicious activity.

- ✓ Isolating the System from the Network

The system issues a network disconnection command (ipconfig /release on Windows). This helps contain the ransomware spread across other systems, especially if it attempts to move laterally through the network.

These are the actions that are logged and displayed to inform administrators that active mitigation has been executed.

Defense in Depth Approach

To further strengthen our system beyond immediate mitigation we will look to add additional defense:

- Immediate Response Layer: kill suspicious processes and isolate the network immediately upon detection.
- Manual Intervention Layer: notify administrators to inspect logs and confirm further actions.

In the event that automatic mitigation fails, the system will immediately alert our system administrator for manual intervention, with manual system shutdown or physically unplugging the network as the final containment measures.

Backup and Recovery Strategy

To stay safe from ransomware, we would save copies of important files every day or week, and store them somewhere safe like a cloud account that cannot be changed or deleted easily. We will also consider writing clear steps on how to restore the files if needed. It's important to practice restoring the files often to make sure it works. If ransomware attacks, we should not try to fix infected files but erase everything and load back clean copies from the backup.

Code:

```
import json
import time
from collections import defaultdict
```

```
import os
import psutil


LOG_FILE = "integrity_log.json"
SUSPICIOUS_EXTENSIONS = [".enc", ".locked", ".crypted"]
RANSOM_KEYWORDS = ["your files have been encrypted", "decrypt", "bitcoin"]
def load_logs():
with open(LOG_FILE, "r") as f:
return json.load(f)
def check_file_extension(path):
return any(path.lower().endswith(ext) for ext in SUSPICIOUS_EXTENSIONS)
def check_ransom_note_content(path):
try:
with open(path, "r", errors="ignore") as f:
content = f.read().lower()
print(content)
return any(word in content for word in RANSOM_KEYWORDS)
except:
return False
def detect_policy_violations(logs):
file_mod_times = defaultdict(list)
ransomware_detected = False
for log in logs:
event = log["event"]
path = log["file"]
```

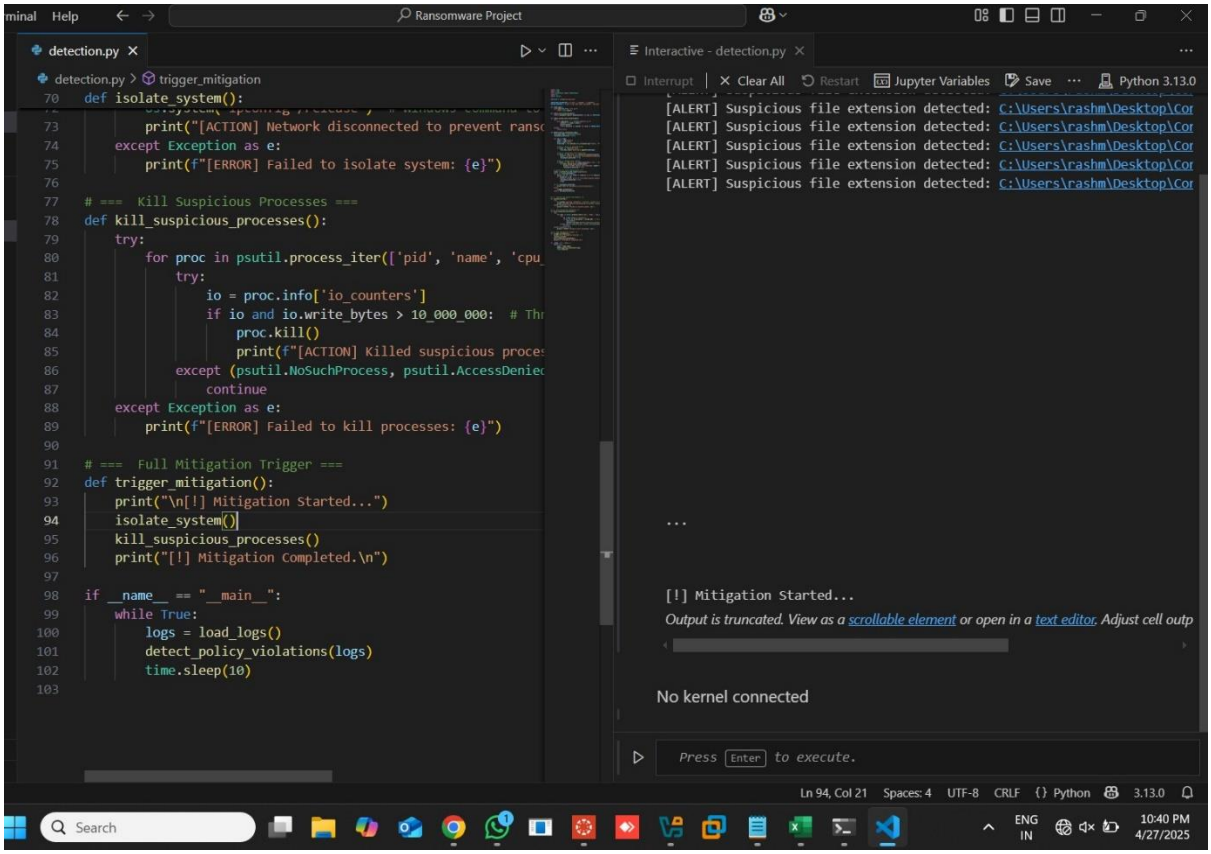
```
timestamp = time.mktime(time.strptime(log["time"], "%Y-%m-%d %H:%M:%S"))

# Rule 1: Burst of modified files
if event == "MODIFIED FILE":
file_mod_times["modified"].append(timestamp)

# Rule 2: Suspicious file extensions
if event == "NEW FILE" and check_file_extension(path):
print(f"[ALERT] Suspicious file extension detected: {path}")
ransomware_detected = True

# Rule 3: Ransom note content
if event == "NEW FILE" and path.endswith((".txt", ".html")):
if check_ransom_note_content(path):
print(f"[ALERT] Ransom note detected: {path}")
ransomware_detected = True

# Analyze for burst modification rule
times = sorted(file_mod_times["modified"])
for i in range(len(times)):
burst = [t for t in times if times[i] <= t <= times[i] + 30]
if len(burst) > 20:
print(f"[ALERT] Burst of file modifications detected: {len(burst)} in 30s")
ransomware_detected = True
break
```

```
if not ransomware_detected:
print("[OK] No ransomware activity detected.")
else:
trigger_mitigation()
return ransomware_detected

# === Isolate the System from Network ===
def isolate_system():
try:
os.system("ipconfig /release") # Windows command to disconnect network
print("[ACTION] Network disconnected to prevent ransomware spread.")
except Exception as e:
print(f"[ERROR] Failed to isolate system: {e}")

# === Kill Suspicious Processes ===
def kill_suspicious_processes():
try:
for proc in psutil.process_iter(['pid', 'name', 'cpu_percent', 'io_counters']):
try:
io = proc.info['io_counters']
if io and io.write_bytes > 10_000_000: # Threshold: >10MB written = suspicious
proc.kill()
print(f"[ACTION] Killed suspicious process: {proc.info['name']} (PID {proc.info['pid']})")
except (psutil.NoSuchProcess, psutil.AccessDenied):
```

```
continue
except Exception as e:
print(f"[ERROR] Failed to kill processes: {e}")

# === Full Mitigation Trigger ===
def trigger_mitigation():
print("\n[!] Mitigation Started...")
kill_suspicious_processes()
isolate_system()
print("[!] Mitigation Completed.\n")
if __name__ == "__main__":
while True:
logs = load_logs()
detect_policy_violations(logs)
time.sleep(10)
```

Output:

When a ransomware threat was detected, the mitigation script successfully triggered and automatically disconnected the system from the Wi-Fi network to prevent the ransomware from spreading or contacting its control server. This containment action was confirmed during testing, as seen in the screenshot where the Wi-Fi got disconnected immediately after mitigation started.


Ransomware Project
detection.py
detection.py > trigger_mitigation
def isolate_system():
print("[ACTION] Network disconnected to prevent ransc
except Exception as e:
print(f"[ERROR] Failed to isolate system: {e}")
# === Kill Suspicious Processes ===
def kill_suspicious_processes():
try:
for proc in psutil.process_iter(['pid', 'name', 'cpu_
try:
io = proc.info['io_counters']
if io and io.write_bytes > 10_000_000: # Thr
proc.kill()
print(f"[ACTION] Killed suspicious proces
except (psutil.NoSuchProcess, psutil.AccessDenied
continue
except Exception as e:
print(f"[ERROR] Failed to kill processes: {e}")
# === Full Mitigation Trigger ===
def trigger_mitigation():
print("\n[!] Mitigation Started...")
isolate_system()
kill_suspicious_processes()
print("[!] Mitigation Completed.\n")
if __name__ == "__main__":
while True:
logs = load_logs()
detect_policy_violations(logs)
time.sleep(10)
Interactive - detection.py
Interrupt
Clear All
Restart
Jupyter Variables
Save
Python 3.13.0
[ALERT] Suspicious file extension detected: C:\Users\rashm\Desktop\Cor
[ALERT] Suspicious file extension detected: C:\Users\rashm\Desktop\Cor
[ALERT] Suspicious file extension detected: C:\Users\rashm\Desktop\Cor
[ALERT] Suspicious file extension detected: C:\Users\rashm\Desktop\Cor
[ALERT] Suspicious file extension detected: C:\Users\rashm\Desktop\Cor
...
[!] Mitigation Started...
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell outp
No kernel connected
Press Enter to execute.
Ln 94, Col 21
Spaces: 4
UTF-8
CRLF
{} Python
3.13.0
Search
ENG
IN
10:40 PM
4/27/2025