Incident Response & Digital Forensics

Assignment 1:

Question: Simulate a security breach by creating and executing a malicious script on a Linux system. Create an audit log and track the script's activity, including any file changes.

Simulation of a Security Breach and Audit Log Tracking on Linux

Introduction

In this assignment, a controlled security breach was simulated on a Linux system by creating and executing a benign malicious script. The objective was to demonstrate typical malicious behaviors, such as file creation and permission changes, and to track all such activities using the Linux auditing system (auditd). This simulation enables understanding of attack footprints and effective monitoring.

Scope and Objective

- To create a simple script that mimics malicious behavior by modifying system files and permissions.
- To configure Linux auditd to monitor the execution of the script and resultant file changes.
- To generate and analyze audit logs capturing the breach activity for learning and documentation.

Environment Setup

- A dedicated virtual machine running Ubuntu (or similar Linux distribution) was used for safe and isolated testing.
- The auditd service was installed and initialized to provide system-level audit capabilities.

Malicious Script Creation

Script Purpose

The script performs the following actions to simulate suspicious activity:

- Creates a file /tmp/hack_notice.txt containing a warning message.
- Changes the file permissions to restrict access.
- Logs a timestamp message to /tmp/hack_log.txt.
- Creates a hidden file /tmp/.hidden_file to simulate stealth.
-

Script Content (malicious.sh)

```
#!/bin/bash
echo "You have been hacked" > /tmp/hack_notice.txt
chmod 600 /tmp/hack_notice.txt
echo "Malicious script ran at $(date)" >> /tmp/hack_log.txt
touch /tmp/.hidden_file
```

After creating the script file, it was made executable with:

```
chmod +x malicious.sh
```

Auditd Configuration

Installation and Service Start

```
sudo apt-get install auditd
```

```
sudo systemctl start auditd
```

```
sudo systemctl enable auditd
```

Audit Rules Applied

- Monitored the execution of the script itself:

```
sudo auditctl -w /path/to/malicious.sh -p x -k malicious_script
```

- Monitored file write and attribute changes in /tmp directory:

```
sudo auditctl -w /tmp/ -p wa -k tmp_changes
```

Execution of the Script

The script was executed with:

```
./malicious.sh
```

This triggered the simulated malicious behavior and generated related audit events.

Audit Log Retrieval and Analysis

Commands Used to Retrieve Logs

- To view script execution logs:

```
sudo ausearch -k malicious_script
```

- To view file changes in /tmp:

```
sudo ausearch -k tmp_changes
```

- To get a summary of file-related audit events:

```
sudo aureport -f
```

Observations

- The execution of the script was captured, showing the timestamp and user.
- File creation and permission change events in /tmp were logged with detailed attributes.
- Hidden file creation was recorded, illustrating stealthy actions detected by auditd.

Conclusion

This simulation successfully demonstrated how a simple malicious script can alter files and permissions on a Linux system and how auditd can effectively capture and log these suspicious activities. Proper configuration of auditd rules enables detection and forensic analysis of breaches, proving essential for security monitoring and incident response.