**Memo – Mitigation Component**

**Group 3 – Fnu Abdullah & Ahmed Hussain**

**Mitigation Overview**

For the mitigation part of this project, we are using AppArmor to limit what the ransomware (or any Python-based script) can do on the system. Basically, AppArmor allows us to control what files and directories a program can access or modify.

Installing the service:

sudo apt update

sudo apt install apparmor apparmor-utils

sudo systemctl enable apparmor

sudo systemctl start apparmor

Then we created a new rule for Python located at /*etc/apparmor.d/usr.bin.python3*

In that file, I wrote a custom policy that blocks python3 from accessing my personal folder /*home/kali/Desktop/personal\_Fa0337* and the virtual environment folder */home/kali/Desktop/venv*

Everything else (like normal read/write in the home directory) is still allowed.

#include <tunables/global>

/usr/bin/python3.13 flags=(attach\_disconnected, audit) {

  # Logging any denied writes anywhere

  audit deny /\*\* w,

  #include <abstractions/base>

  #include <abstractions/python>

  /usr/bin/python3 ix,

  /usr/lib/python3\*/\*\* rix,

  /usr/local/lib/python3\*/\*\* rix,

  # Allow read/write/execute in home directory, except where denied

  owner @{HOME}/\*\* rwix,

  # Explicitly deny access to protected folder and venv

  deny /home/kali/Desktop/personal\_Fa0337/\*\* rwk,

  deny /home/kali/Desktop/venv/\*\* rwk,

}

After creating the rule, we reloaded and enforced it using:

sudo apparmor\_parser -r /etc/apparmor.d/usr.bin.python3

sudo aa-enforce /usr/bin/python3

This means if someone tries to run a malicious Python script that tries to modify files inside those protected folders, AppArmor will deny access. These can later be modified to include more programs and processes.

To stop the AppArmor, we can

sudo aa-teardown

sudo systemctl stop apparmor

and bring it back using:

sudo systemctl start apparmor

sudo apparmor\_parser -r /etc/apparmor.d/\*

We would see if the attack succeeded or got denied in the *setup.log* file which the attack generates (for testing purposes).

Moreover, when the AppArmor blocks something, it gets recorded in the audit log. For example, audit.log denies, we can see in figure below:

A screenshot of a computer program

AI-generated content may be incorrect.

In general, AppArmor denies the action, audit.log records an apparmor=”DENIED” entry and Wazuh detects that entry and sends the alert.

**Defense in Depth**

To add another protection layer (in case AppArmor fails), we are using a **defense-in-depth** approach with **Wazuh Active Response**.

In the Wazuh configuration file /var/ossec/etc/ossec.conf, we enabled the active-response module and linked it to a custom script called ransomware-response.sh.

That script is stored in /var/ossec/active-response/bin/ and it does a few things automatically when ransomware activity is detected:

* It kills the suspicious process using its PID.
* It moves the infected or modified file into a quarantine folder (/tmp/quarantine) so it can’t cause more damage.
* It also tries to send an email alert to the system administrator to let them know what happened.

So, if somehow the ransomware bypasses AppArmor, Wazuh will detect it, kill it, and isolate the affected files. This gives another chance to contain the attack before it spreads. Moreover, Wazuh will generate more alerts like in the following figures:

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Backup and Recovery**

All the important folders are to be backed up to an external drive or a cloud location that is **not constantly connected** to the system. This prevents the ransomware from encrypting the backups too. Incremental backups or snapshot-based systems (like using rsync or Timeshift) help keep older copies safe so we can restore files from before the attack happened. In an advanced technological environment, we can employ daily LVM or Btrfs snapshots, which are then incrementally synced to an off-site, immutable cloud storage bucket (e.g., AWS S3 with Object Lock or a private backup server).

If an attack ever succeeds, we can clean the system, restore the files from the last good backup, and re-enable AppArmor and Wazuh to bring protection back in place.

In detail, we can:

1. Isolate the compromised machine immediately from the network.
2. Access the Wazuh logs and system audit logs to analyze and conduct a full root cause analysis (RCA).
3. Completely re-image the OS from a known trusted source (if the root files are encrypted)
4. Before reconnecting, the OS is fully patched with all the security controls, including the Wazuh and AppArmor, and are verified to be active and in enforce mode.
5. Data for the affected directories is restored from the most recent, verified clean, and off-site backup.
6. We then validate that the data is restored, and applications are functional before the host is fully returned to production status.