APPLICATION ARMOR

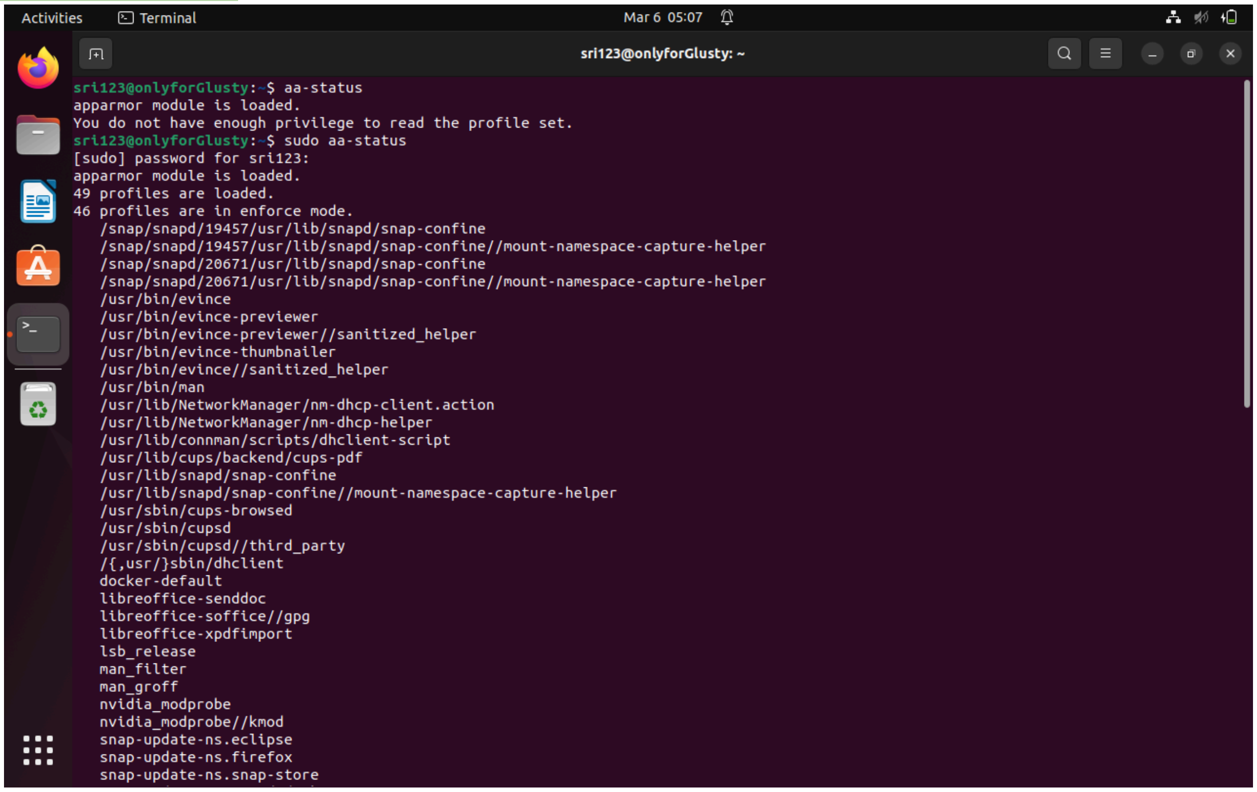
GUMMA SRI SOUGANDHIKA - 121AD0020

INTRODUCTION:

App Armor is a built in application which helps protect apps from other harmful apps and vice versa. It can be run by directly opening a terminal in ubuntu.

MAIN COMMANDS RUN:

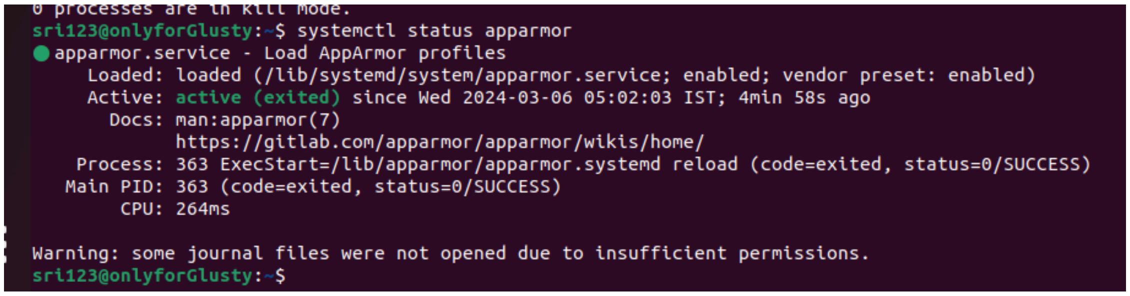
1. ***sudo aa-status***: this gives what app armor is doing right now.



```
sri123@onlyforGlusty:~$ aa-status
apparmor module is loaded.
You do not have enough privilege to read the profile set.
sri123@onlyforGlusty:~$ sudo aa-status
[sudo] password for sri123:
apparmor module is loaded.
49 profiles are loaded.
46 profiles are in enforce mode.
   /snap/snapd/19457/usr/lib/snapd/snap-confine
   /snap/snapd/19457/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /snap/snapd/20671/usr/lib/snapd/snap-confine
   /snap/snapd/20671/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/bin/evince
   /usr/bin/evince-previewer
   /usr/bin/evince-previewer//sanitized_helper
   /usr/bin/evince-thumbnailer
   /usr/bin/evince//sanitized_helper
   /usr/bin/man
   /usr/lib/NetworkManager/nm-dhcp-client.action
   /usr/lib/NetworkManager/nm-dhcp-helper
   /usr/lib/connman/scripts/dhclient-script
   /usr/lib/cups/backend/cups-pdf
   /usr/lib/snapd/snap-confine
   /usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/sbin/cups-browsed
   /usr/sbin/cupsd
   /usr/sbin/cupsd//third_party
   /{,usr/}sbin/dhclient
   docker-default
   libreoffice-senddoc
   libreoffice-soffice//gpg
   libreoffice-xpdfimport
   lsb_release
   man_filter
   man_groff
   nvidia_modprobe
   nvidia_modprobe//kmod
   snap-update-ns.eclipse
   snap-update-ns.firefox
   snap-update-ns.snap-store
```

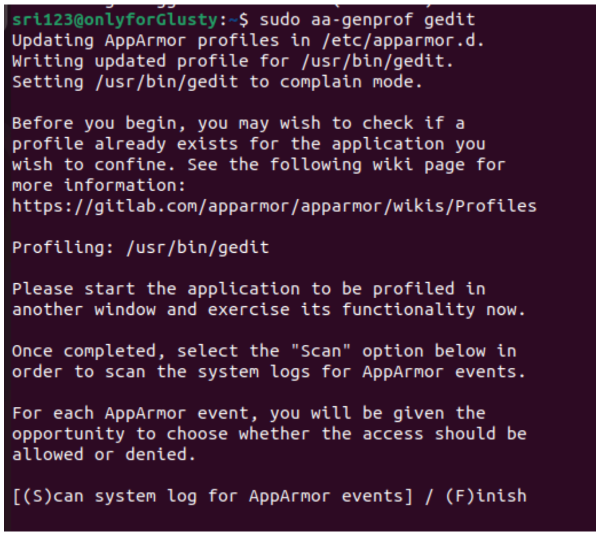
There are three modes in the app armor mainly:
-> Enforce mode: it forces these rules uponthe app.
-> Complaint mode: it allows you to do some things outside of the rules, but it will put a complaint over that action.
-> Unconfined: it freely leaves the app to do whatever it wants.

2. **systemctl status apparmor:** Shows whether the apparmor is running or not



```
0 processes are in kill mode.
sri123@onlyforGlusty:~$ systemctl status apparmor
● apparmor.service - Load AppArmor profiles
     Loaded: loaded (/lib/systemd/system/apparmor.service; enabled; vendor preset: enabled)
     Active: active (exited) since Wed 2024-03-06 05:02:03 IST; 4min 58s ago
       Docs: man:apparmor(7)
             https://gitlab.com/apparmor/apparmor/wikis/home/
    Process: 363 ExecStart=/lib/apparmor/apparmor.systemd reload (code=exited, status=0/SUCCESS)
   Main PID: 363 (code=exited, status=0/SUCCESS)
        CPU: 264ms

Warning: some journal files were not opened due to insufficient permissions.
sri123@onlyforGlusty:~$
```

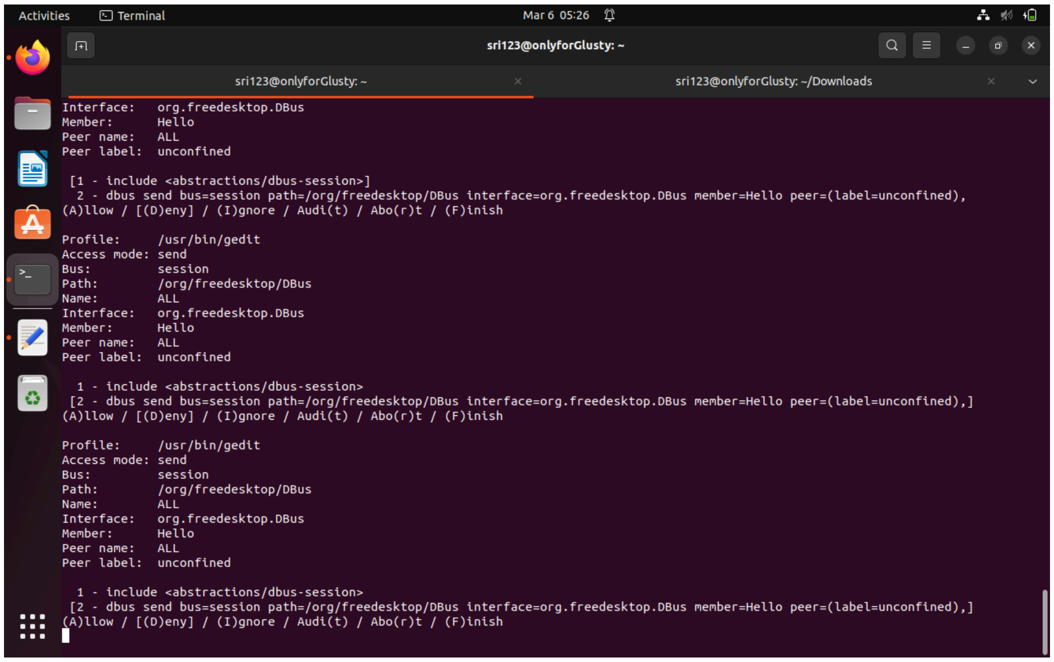
3. **sudo systemctl disable apparmor**: In order to completely disable apparmor.
4. **sudo aa-genprof <scripname.sh or appname>**: It generates a profile for the application.
   But before generating a profile, we should install the utilities of the app armor, else it throws errors.
   sudo apt-get update
   sudo apt-get install apparmor-utils
   Suppose we are trying to create a profile for gedit. Then we can have command: *sudo aa-genprof gedit*



```
sri123@onlyforGlusty:~$ sudo aa-genprof gedit
Updating AppArmor profiles in /etc/apparmor.d.
Writing updated profile for /usr/bin/gedit.
Setting /usr/bin/gedit to complain mode.

Before you begin, you may wish to check if a
profile already exists for the application you
wish to confine. See the following wiki page for
more information:
https://gitlab.com/apparmor/apparmor/wikis/Profiles

Profiling: /usr/bin/gedit

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

[(S)can system log for AppArmor events] / (F)inish
```

This way, the AA shows that a profile for gedit is already created. We can update the settings or finish, using S or F.
When i press S, it shows that:
"Please start the application to be profiled in
another window and exercise its functionality now"
This means that the app needs to be run in order to view the logs.

We can see the log results as given below:



```
Interface:   org.freedesktop.DBus
Member:      Hello
Peer name:   ALL
Peer label:  unconfined

 [1 - include <abstractions/dbus-session>]
  2 - dbus send bus=session path=/org/freedesktop/DBus interface=org.freedesktop.DBus member=Hello peer=(label=unconfined),
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish

Profile:     /usr/bin/gedit
Access mode: send
Bus:         session
Path:        /org/freedesktop/DBus
Name:        ALL
Interface:   org.freedesktop.DBus
Member:      Hello
Peer name:   ALL
Peer label:  unconfined

  1 - include <abstractions/dbus-session>
 [2 - dbus send bus=session path=/org/freedesktop/DBus interface=org.freedesktop.DBus member=Hello peer=(label=unconfined),]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish

Profile:     /usr/bin/gedit
Access mode: send
Bus:         session
Path:        /org/freedesktop/DBus
Name:        ALL
Interface:   org.freedesktop.DBus
Member:      Hello
Peer name:   ALL
Peer label:  unconfined

  1 - include <abstractions/dbus-session>
 [2 - dbus send bus=session path=/org/freedesktop/DBus interface=org.freedesktop.DBus member=Hello peer=(label=unconfined),]
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish
```

When we run aa-genprof and then select "Scan" while interacting with the application, the tool scans system logs (specifically /var/log/syslog in Ubuntu) for AppArmor events related to the application.
In the output shown, aa-genprof has detected an AppArmor event involving gedit.

- **Profile**: the application being profiled.
- **Access mode**: type of operation, here, sending a message(*send*).
- **Bus**: *session* specifies the D-Bus bus type involved
- **Path**: it is the D-Bus path that gedit attempted to access.
- **Interface**: it is the D-Bus interface gedit tried to interact with.
- **Member**: Hello is the method of the D-Bus interface that was called.
- **Peer name**: ALL suggests this applies to all peers, indicating a broad permission if allowed.
- **Peer label**: unconfined means the peer (the other end of the D-Bus communication) is not restricted by AppArmor.

The prompt then offers choices for handling this event:

- **[1 - include <abstractions/dbus-session>]**: Suggests adding a collection of rules that allows standard D-Bus session communication.
- **2 - dbus send bus=session…**: Represents a specific rule to allow the detected D-Bus operation. This option is more granular and restricts the permission to just this operation.

Then we have other choices: (A)llow, (D)eny, (I)gnore, Audi(t), Abo(r)t, or (F)inish.

5. **sudo aa-logprof:** it updates any of the profiles existing on our systems. After we create a profile, then update the app, then the profile needs to be updated also, hence this manages all of these.

A SIMPLE EXAMPLE OF HOW APPARMOR WORKS - NANO!

We will try to create a profile for the well-known application “nano”.

In order to do so, just follow these steps:

1. sudo aa-genprof nano: This generates a profile.
2. Before clicking for “Scan”, open another terminal and using nano, open any sample text file.
3. Now click Shift+S in the first terminal, which then scans the actions of the nano file.
4. Keep on pressing Shift+A, so as to allow nano to access the path where you have opened the sample text file.
5. After all “allows”, it asks for the (S)ave changes. So click Shift+S.
6. Now go back to the 2nd terminal, close the previous file, go to another directory and try opening another file. We are now getting permission denied.

Even if you use sudo, it is of no use. You can no longer view any data except for the first directory you allowed!

7. In order to fix this, we can change the contents of the script for the profile! Yes, it is present as */etc/apparmor.d/usr.bin.appname*. But hold on. We cannot access any **other directories, so we cannot modify them.**
8. In order to do so, just open the above file. Switch back to the 1st terminal, open the profile again: sudo aa-genprof nano. Now Shift+S to scan. After this again give all “Allows”, by Shift+A.
9. After this, go to the 2nd terminal and reopen the script file for the nano profile. Now you can view the contents! We can change them according to our requirements.
10. Finally, if we do not want these rules, we just want nano as it was before, let’s revert back by deleting the profile for the nano. Execute these commands:

```
sudo aa-status
sudo apparmor_parser -R /etc/apparmor.d/usr.bin.nano
sudo aa-status
sudo rm /etc/apparmor.d/usr.bin.nano
sudo systemctl reload apparmor
sudo aa-status
```

This finally gives us back what we had, an unconfined nano application that works just as good as it did before.

KEY TERMINOLOGIES TO PONDER OVER

The options used in AppArmor profile generation when aa-genprof or aa-logprof detects an access violation or a request for access in a specific context are as below. They allow you to specify how AppArmor should handle the detected access attempt. Here's what each option does:

1. **(I)nherit**: Permissions will be the same as what's specified in the parent profile.
2. **(C)hild**:t means that the access is allowed only if it matches the child profile.
3. **(N)amed**: This option allows you to specify a specific named profile for the access.
4. **(U)nconfined**: This option indicates that the access should be allowed without any AppArmor confinement, or simply disables AA for it.
5. **(D)eny**: This option denies the access attempt.
6. **Abo(r)t**: This option allows you to abort the profiling process. The profiling process will be stopped, and no further changes will be made to the AppArmor profile.
7. **(F)inish**: This option indicates that you've finished making changes for the current access attempt.

File system access: *sudo cat /etc/apparmor.d/<profile_name>*

We can clearly see the files or directories to which the app has access. In the ending lines, like *<u>/etc/host.conf</u>* with options such as <u>r</u>, <u>rw</u> or <u>w</u>.

Process access and root level permissions: *sudo cat /etc/apparmor.d/<profile_name>*

In the same file, we can find the process access and root level permissions:

- px, ux, Px, Ux: These tokens govern executed processes. For example, px allows the execution of a process and transitions to the profile of the executed process, while ux allows execution unconfined.
- Capabilities: Lines that start with capability define what Linux kernel capabilities the lynx process can have. For example, capability net_raw allows the process to use raw sockets.

Network-access permissions:

Directly managing network access is typically outside AppArmor's direct scope, as AppArmor focuses more on filesystem and process capabilities. However, AppArmor can restrict access to network-related files (e.g., /etc/resolv.conf, socket files), indirectly affecting how an application interacts with the network.

```
# Allow basic networking (DNS resolution, etc.)
   network inet stream,
   network inet6 stream,
   network inet dgram,
   network inet6 dgram,
# Deny raw and packet socket creation
   deny network raw,
   deny network packet,
```