Lab 11:

AppArmor

Linux Server Security  
 2024-2025

© Daan Pareit, Henk Brouckxon

## Introduction

# Lab concept

During this lab, we will explore AppArmor.

# Learning goals

* Understanding AppArmor and AppArmor profiles
* Enabling and disabling AppArmor profiles
* Making your own (simpe) AppArmor profile and troubleshooting

# Practicalities and prerequisites

You’ll need the following:

* Your Debian VM and the pfSense VM as installed during previous lab(s)

## AppArmor Basics

On Debian, AppArmor is installed by default. Let’s check if it is working...

* Perform **cat /sys/module/apparmor/parameters/enabled** and consult the output. Is AppArmor enabled by default?

Looks like it is.

A black and white text

Description automatically generated

* Now let’s see if AppArmor is monitoring any of our applications. Execute aa-status. What can you see here? What modes can an application be in?

A screenshot of a computer program

Description automatically generated

**enforce; complain; unconfined** modes

* Where are the AppArmor profile files for these applications located (check the manual page for apparmor and/or lecture slides)? Have a look at the profile for the “man” application. For now, ignore the sections on “groff” and “filter”. Does AppArmor apply any restrictions on file access for “man”? Now check the same for tcpdump (first install tcpdump if necessary).

/etc/apparmor.d

Its either that I am not understanding something, or there is no like clear restriction to write anything regarding “man”

A black background with white text

Description automatically generated

It does have quite some restrictions

A computer screen shot of white text

Description automatically generated

* To modify how AppArmor applies the profiles, we need to install some additional tools. Install the **apparmor-utils** package. Now change the AppArmor mode for tcpdump to “complain”. Check if this worked by running aa-status.

A screen shot of a computer code

Description automatically generated

sudo aa-complain /usr/bin/tcpdump

A screen shot of a computer

Description automatically generated

* Let’s clean up our actions. Re-enable the AppArmor profile for tcpdump.

sudo aa-enforce /usr/bin/tcpdump

A screenshot of a computer

Description automatically generated

## Generating and testing a (simple) AppArmor profile

* Now let’s see if we can make our own AppArmor profile for a new application. To do this, we’ll first make our own script that tries to write some data to the disk. In your working folder, make a new file ”my\_legit\_scipt.sh” with the following contents:

#!/bin/bash  
mkdir -p ./test  
echo "Harmless text" > ./test/textfile.txt  
echo "I'm a friendly program :-)"  
echo "Really, I am!"

sleep 20

A screen shot of a computer code

Description automatically generated

* Modify the properties of this script in such a way that it can execute independently (i.e. without explicitly calling bash) and run it as your default (non-root) user using the command “./my\_legit\_script.sh”.

chmod u+x my\_legit\_script.sh

A computer screen with green and white text

Description automatically generated

A screen shot of a computer

Description automatically generated

* Is this script already being monitored by AppArmor? Does it already have its own profile?

Does not look like it

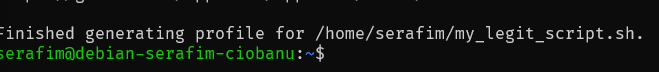
A screenshot of a computer program

Description automatically generated

* Now we’ll make an AppArmor profile for our small script. Run the helper program **aa-genprof** for your program “my\_legit\_script.sh” and follow the instructions it gives you. Read the instructions carefully (e.g. run the script before scanning!). When scanning is done, first option is mostly ok when parsing (inherit/allow/save/…). Then finish the generation.  
  Note: since Debian 12, aa-genprof might complain about a missing syslog file (related to a migration to focus more on systemd journaling). To solve this, install the ‘rsyslog’ package.

I SWEAR TO GOD I COULD NOT READ THE LAST SENTENCE FOR 10 MINUTES!!! HENCE I CREATED MY OWN SYSLOG file…

So I had to start genprof, then run the command, and then I did inherit, inherit, Finish, and save



* Did this generate a new AppArmor profile for your script? What access restrictions does this profile apply? Is the profile already enforced?

A screen shot of a computer

Description automatically generated

It did.

It does not allow to mkdir, sleep, and bash overall?

A screen shot of a computer

Description automatically generated

A screen shot of a computer

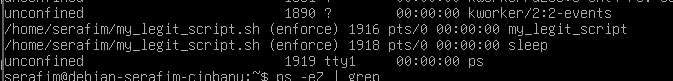
Description automatically generated

It is also enforced.

* Now run the script in one terminal and in another terminal, while the script is still running, execute aa-status once and query the process list with ps -eZ. Do you notice the security context of your running script?

A screenshot of a computer

Description automatically generated



I do not seem to get the security context? All it says is that it is enforced.

* Now let’s modify our script, so it generates some more data. Change the contents of the script to the following:

#!/bin/bash  
mkdir -p ./test  
echo "Harmless text" > ./test/textfile.txt

echo "Harmless text 2" > ./test/textfile2.txt  
echo "I'm a friendly program :-)"

sleep 20

Does the script still run without errors? If not, what changes should you make to the profile to allow it to run error-free? Apply changes to the profile so the script can write to any file in the test directory, re-load the profile using aa-enforce <profile-full-file-name> (or restart the apparmor systemd service) and run the script again. You should not get any errors now.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

It does run with errors.

A computer screen with white text and green text

Description automatically generated

A screenshot of a computer

Description automatically generatedLooks like it works (this is getting mental for my sanity)

* Oh no! a hacker succeeded in modifying our beautiful script, and he’s trying to use it for his own nefarious purposes. Modify the content of the script so it contains the following commands:

#!/bin/bash  
mkdir -p ./test  
echo "Harmless text" > ./test/textfile.txt

echo "Harmless text 2" > ./test/textfile2.txt  
echo "I'm a friendly program :-)"

mkdir -p ./config

echo "Kaboom!" > ./config/nefarious.txt  
echo "Really :-)"

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

Looks like he did not succeed in his nefarious doings.

* Did the hacker succeed in his plans? Run the script and check if you can find his nefarious activity in the logs of the machine.

A screen shot of a computer

Description automatically generated

Can be bypassed by doing

A computer screen shot of white text

Description automatically generated

sh my\_legit\_script.sh

bash my\_legit\_script.sh