15.9.3 Application Armor

Click one of the buttons to take you to that part of the video.

AppArmor 0:00-0:23

In this demonstration, we're going to cover a few commands that will help you manage AppArmor.

First, I'll show you how to determine whether AppArmor is running, how to start AppArmor, and how to automatically load AppArmor when the system boots.

Then I'll show you how to manage the modes AppArmor profiles can run in.

And finally, I'll show you how you can fine-tune AppArmor without editing a profile.

Enabling AppArmor 0:24-2:12

On several distributions, such as OpenSUSE and Ubunto, AppArmor is installed and enabled by default. But you should definitely verify that installed and running correctly.

To do this, the first thing I'm going to do is switch to the root account. So, from my terminal, I'll type ‘sudo -i' and then enter my password. I need to do this because most of the commands we'll be using today require root permissions.

There are several ways to determine whether AppArmor has been installed.

One method is to check to see if the AppArmor profiles have been added to the system. An AppArmor profiles is a set of rules that determine how an application can interact with the computer.

To see if these profiles are installed, let's use the cd command to enter the /etc/AppArmor.d directory and list its contents using the ls command. The fact that this directory exists and contains profiles tells me that AppArmor has been installed.

To see if AppArmor is running, we'll use the 'systemctl status AppArmor' command.

With this command, we not only see that AppArmor is installed, but if we look at the output closely, we see that the AppArmor service is inactive, meaning that the service isn't running. We also see that the load state is set to Disabled. This means that even if the services were running, the next time I boot this computer, AppArmor won't start automatically.

To fix these issues, I'll enter ‘systemctl start AppArmor'. If we look at the status again, we see that AppArmor is now running.

Since we don't want to have to remember to start AppArmor each time the computer is rebooted, we can force AppArmor to load automatically by typing ‘systemctl enable AppArmor'.

If we look at the status again, we see that this service is now enabled and will start automatically each time the system is restarted.

AppArmor Modes-Status 2:13-2:39

The AppArmor profiles, which control what our applications can and can't do, can be configured in one of three modes.

The first mode, which is the default, is the Enforce mode. This mode simply means that the rules contained in the profile will be enforced.

Another mode is called Complain. When a profile is configured as Complain, the profile rules aren't enforced, but AppArmor tracks any violation of those rules and saves them in your log files.

Installing AppArmor Utilities 2:40-3:31

If you run aa-status and display the results one page at a time using the | less option, you see that we have 40 profiles loaded. And of those 40, we have 38 that are being enforced, while two are configured to work in Complain mode.

The last mode available is Disabled mode, meaning that the profile is not running in any form. You may want to switch to this mode if an application being protected by AppArmor is causing issues.

To switch between these different modes, we'll use a series of AppArmor commands.

Before I can do that, however, I need to install the AppArmor utilities.

To do this, I'll run ‘apt install AppArmor-utils'.

AppArmor Modes-Enable 3:32-4:09

Okay. With those installed, let's run aa-status again. Notice that the Firefox profile is not being enforced.

If this were an application I'd like to protect, I would run ls from the /etc/AppArmor.d directory to find out the name of the profile and then type ‘aa-enforce usr.bin.firefox'.

Running aa-status again, I now see that Firefox is being enforced.

AppArmor Modes-Disable 4:10-4:41

But let's suppose that users start complaining that Firefox isn't working properly now that it's being protected.

If you wanted to get things back to normal quickly, you could disable this profile by running 'aa-disable usr.bin.firefox'. This not only disables the profile, but also moves the profile into the Disabled directory.

Another quick look at the status, and we see that Firefox is no longer being protected.

AppArmor Modes-Complain 4:42-5:27

Although disabling a profile may be required at times, instead, you may want to simply change the profile to Complain mode. This way, you can look at the log files and see what changes would let the application run as intended.

Let's switch to Complain mode outside of the AppArmor.d directory and try this.

So, I'll type ‘cd' and then ‘pwd', and you can see that I'm now in the /root directory.

To change Firefox to Complain mode, I'll run aa-complain. But this time, I'll need to specify the full path to the profile, so I'll enter 'etc/AppArmor.d/usr.bin.firefox'.

Now, if we look at the status, we see that Firefox is running in Complain mode.

AppArmor Unconfined 5:28-7:10

Now let's shift gears a bit and look at another useful command called

aa-unconfined.

If we look at the man pages for this command, it says this command will "Output a list of processes with tcp or udp ports that do not have AppArmor profiles loaded."

We can also see that there are three options available for this command.

The first option, --paranoid, displays all of the processes from the /proc filesystem with tcp or udp ports that do not have AppArmor profiles loaded.

The second option is --with-ss. This is the default option that will be used if no options are explicitly entered. It works with the ss command to show you just the processes listening on the network sockets.

The third options is --with-netstat. This options gives you the same results as the previous one, but does so using the deprecated netstat command.

To run this command, type aa-unconfined followed by the desired options. Let's start with --with-ss. Now the processes listening on the network sockets are shown.

Let's run the command again. But this time, let's use the -netstat option. As you can see, the results are identical. The only difference was that netstat was used to find the results instead of the ss command.

And now, as I run the command again using the --paranoid option, I see page after page of results.

An important thing to keep in mind with this command is that it's still a little unreliable. In other words, you wouldn't want to use this program for your forensics; but you would want to use it as an aid in profiling all of the network-assessable processes in a lab environment.

Tunables 7:11-8:09

If we use cd to go back to the /etc/AppArmor.d directory, we'll see we also have a subdirectory named tunalbes. And as I move to this directory, we have several text-based files.

These tunable files give you a way to tune AppArmor without having to adjust your profiles.

For example, one common item often tuned is the location of the user's home directory, and with AppArmor, you can alter this using the tunables file named home.

If I open this file in my text editor, I see that the home directory is defined as /home/.

However, if some accounts use different location for their home directors, such as /exports/home, all I need to do is add a space and the desired path to the end of the existing path, save my file, and I'm done.

Summary 8:10-8:32

That's it for this demonstration.

In this demonstration, we covered how to determine whether AppArmor is running using the systemctl commands and how to start and enable AppArmor. We also talked about how to manage the different modes in which an AppArmor profile can run using the aa commands, such as aa-disable, aa-complain, and aa-enforce. We also introduced the concept of tunables and discussed how to configure these files.
