AppArmor

AppArmor is a Linux Security Module implementation of name-based mandatory access controls. AppArmor confines individual programs to a set of listed files and posix 1003.1e draft capabilities.

AppArmor is installed and loaded by default. It uses *profiles* of an application to determine what files and permissions the application requires. Some packages will install their own profiles, and additional profiles can be found in the **apparmor-profiles** package.

To install the **apparmor-profiles** package from a terminal prompt:

```
sudo apt-get install apparmor-profiles
```

AppArmor profiles have two modes of execution:

- Complaining/Learning: profile violations are permitted and logged. Useful for testing and developing new profiles.

- Enforced/Confined: enforces profile policy as well as logging the violation.

Using AppArmor

The **apparmor-utils** package contains command line utilities that you can use to change the **AppArmor** execution mode, find the status of a profile, create new profiles, etc.

- **apparmor_status** is used to view the current status of AppArmor profiles.

  ```
  sudo apparmor_status
  ```

- **aa-complain** places a profile into *complain* mode.

  ```
  sudo aa-complain /path/to/bin
  ```

- **aa-enforce** places a profile into *enforce* mode.

  ```
  sudo aa-enforce /path/to/bin
  ```

- The `/etc/apparmor.d` directory is where the AppArmor profiles are located. It can be used to manipulate the *mode* of all profiles.

  Enter the following to place all profiles into complain mode:

  ```
  sudo aa-complain /etc/apparmor.d/*
  ```

  To place all profiles in enforce mode:

  ```
  sudo aa-enforce /etc/apparmor.d/*
  ```

- **apparmor_parser** is used to load a profile into the kernel. It can also be used to reload a currently loaded profile using the *-r* option. To load a profile:

```
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -a
```

To reload a profile:

```
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -r
```

- `/etc/init.d/apparmor` can be used to *reload* all profiles:

```
sudo /etc/init.d/apparmor reload
```

- The `/etc/apparmor.d/disable` directory can be used along with the **apparmor_parser -R** option to *disable* a profile.

```
sudo ln -s /etc/apparmor.d/profile.name /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/profile.name
```

To *re-enable* a disabled profile remove the symbolic link to the profile in `/etc/apparmor.d/disable/`. Then load the profile using the *-a* option.

```
sudo rm /etc/apparmor.d/disable/profile.name
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -a
```

- **AppArmor** can be disabled, and the kernel module unloaded by entering the following:

```
sudo /etc/init.d/apparmor stop
sudo update-rc.d -f apparmor remove
```

- To re-enable **AppArmor** enter:

```
sudo /etc/init.d/apparmor start
sudo update-rc.d apparmor defaults
```

Replace *profile.name* with the name of the profile you want to manipulate. Also, replace `/path/to/bin/` with the actual executable file path. For example for the **ping** command use `/bin/ping`

Profiles

AppArmor profiles are simple text files located in `/etc/apparmor.d/`. The files are named after the full path to the executable they profile replacing the "/" with ".". For example `/etc/apparmor.d/bin.ping` is the AppArmor profile for the `/bin/ping` command.

There are two main type of rules used in profiles:

- *Path entries:* which detail which files an application can access in the file system.
- *Capability entries:* determine what privileges a confined process is allowed to use.

As an example take a look at `/etc/apparmor.d/bin.ping`:

```
#include <tunables/global>
/bin/ping flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/consoles>
  #include <abstractions/nameservice>

  capability net_raw,
  capability setuid,
  network inet raw,

  /bin/ping mixr,
  /etc/modules.conf r,
}
```

- *#include <tunables/global>:* include statements from other files. This allows statements pertaining to multiple applications to be placed in a common file.
- */bin/ping flags=(complain):* path to the profiled program, also setting the mode to *complain*.
- *capability net_raw,:* allows the application access to the CAP_NET_RAW Posix.1e capability.
- */bin/ping mixr,:* allows the application read and execute access to the file.

After editing a profile file the profile must be reloaded. See the section called “Using AppArmor” for details.

Creating a Profile

- *Design a test plan:* Try to think about how the application should be exercised. The test plan should be divided into small test cases. Each test case should have a small description and list the steps to follow.

  Some standard test cases are:

  - Starting the program.
  - Stopping the program.
  - Reloading the program.
  - Testing all the commands supported by the init script.

- *Generate the new profile:* Use **aa-genprof** to generate a new profile. From a terminal:

  ```
  sudo aa-genprof executable
  ```

  For example:

  ```
  sudo aa-genprof slapd
  ```

- To get your new profile included in the **apparmor-profiles** package, file a bug in *Launchpad* against the AppArmorpackage:

  - Include your test plan and test cases.
  - Attach your new profile to the bug.

Updating Profiles

When the program is misbehaving, audit messages are sent to the log files. The program **aa-logprof** can be used to scan log files for **AppArmor** audit messages, review them and update the profiles. From a terminal:

```
sudo aa-logprof
```

References

- See the AppArmor Administration Guide for advanced configuration options.
- For details using AppArmor with other Ubuntu releases see the AppArmor Community Wiki page.
- The OpenSUSE AppArmor page is another introduction to AppArmor.
- A great place to ask for **AppArmor** assistance, and get involved with the Ubuntu Server community, is the *#ubuntu-server* IRC channel on freenode.