AppArmor

AppArmor is a **Mandatory Access Control** (MAC) system, implemented upon the **Linux Security Modules** (LSM).

Related articles

Security

SELinux

TOMOYO Linux

AppArmor, like most other LSMs, supplements rather than replaces the default Discretionary Access Control (DAC). As such it's impossible to grant a process more privileges than it had in the first place.

Ubuntu, SUSE and a number of other distributions use it by default. RHEL (and it's variants) use SELinux which requires good userspace integration to work properly. SELinux attaches labels to all files, processes and objects and is therefore very flexible. However configuring SELinux is considered to be very complicated and requires a supported filesystem. AppArmor on the other hand works using file paths and its configuration can be easily adapted.

AppArmor proactively protects the operating system and applications from external or internal threats and even zero-day attacks by enforcing a specific rule set on a per application basis. Security policies completely define what system resources individual applications can access, and with what privileges. Access is denied by default if no profile says otherwise. A

few default policies are included with AppArmor and using a combination of advanced static analysis and learning-based tools, AppArmor policies for even very complex applications can be deployed successfully in a matter of hours.

Every breach of policy triggers a message in the system log, and AppArmor can be configured to notify users with real-time violation warnings popping up on the desktop.

Contents

- 1 Installation
  - 1.1 Kernel
  - 1.2 Userspace Tools
  - 1.3 Testing
- 2 Disabling
- 3 Configuration
  - 3.1 Auditing and generating profiles
  - 3.2 Understanding profiles
  - 3.3 Parsing profiles
- 4 Security considerations
  - 4.1 Preventing circumvention of path-based MAC via links
- 5 Tips and tricks
  - 5.1 Get desktop notification on DENIED actions

- 5.2 Cache profiles
- 6 See also

Installation

Kernel

Install `linux-hardened-apparmor` (https://aur.archlinux.org/packages/linux-hardened-apparmor/)[AUR] or compile the kernel yourself.

When **compiling the kernel**, it is required to at least set the following options:

```
CONFIG_SECURITY_APPARMOR=y
CONFIG_SECURITY_APPARMOR_BOOTPARAM_VALUE=1
CONFIG_DEFAULT_SECURITY_APPARMOR=y
CONFIG_AUDIT=y
```

For those new or altered variables to not get overridden, place them at the bottom of the config file or adjust the previous invocations accordingly.

Instead of setting `CONFIG_SECURITY_APPARMOR_BOOTPARAM_VALUE` and `CONFIG_DEFAULT_SECURITY_APPARMOR`, you can also set **kernel boot parameters**: `apparmor=1 security=apparmor`.

Userspace Tools

Note: Since AppArmor builds and installs a kernel module it must be rebuilt against the current kernel on each update

The userspace tools and libraries to control AppArmor are supplied by the **apparmor (https://aur.archlinux.org/packages/apparmor/)**[AUR] package.

The package is a split package which consists of following sub-packages:

- **apparmor (https://aur.archlinux.org/packages/apparmor/)**[AUR] (meta package)
- **apparmor-libapparmor (https://aur.archlinux.org/packages/apparmor-libapparmor/)**[AUR]
- **apparmor-utils (https://aur.archlinux.org/packages/apparmor-utils/)**[AUR]
- **apparmor-parser (https://aur.archlinux.org/packages/apparmor-parser/)**[AUR]
- **apparmor-profiles (https://aur.archlinux.org/packages/apparmor-profiles/)**[AUR]
- **apparmor-pam (https://aur.archlinux.org/packages/apparmor-pam/)**[AUR]
- **apparmor-vim (https://aur.archlinux.org/packages/apparmor-vim/)**[AUR]

To load all AppArmor profiles on startup, **enable** `apparmor.service`.

Testing

After a reboot you can test if AppArmor is really enabled using this command as root:

```
# cat /sys/module/apparmor/parameters/enabled
Y
```

(Y=enabled, N=disabled, no such file = module not in kernel)

Disabling

To disable AppArmor for the current session, **stop** `apparmor.service`, or **disable** it to prevent it from starting at the next boot.

Alternatively you may choose to disable the kernel modules required by AppArmor by appending `apparmor=0 security=""` to the **kernel boot parameters**.

Configuration

Auditing and generating profiles

To create new profiles using `aa-genprof`, `auditd.service` from the package **audit (https://www.archlinux.org/packages/?name=audit)** must be running. This is because Arch Linux adopted systemd and doesn't do kernel logging to file by default. Apparmor can grab

kernel audit logs from the userspace auditd daemon, allowing you to build a profile. To get kernel audit logs, you'll need to have rules in place to monitor the desired application. Most often a basic rule configured with **auditctl(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/auditctl.8)** will suffice:

```
# auditctl -a exit,always -F arch=b64 -S all -F path=/usr/bin/chromium -F key=MonitorChromium
```

but be sure to read **Audit framework#Adding rules** if this is unfamiliar to you.

Note: Remember to stop the service afterwards (and maybe clear `/var/log/audit/audit.log`) because it may cause overhead depending on your rules.

Understanding profiles

Profiles are human readable text files residing under `/etc/apparmor.d/` describing how binaries should be treated when executed. A basic profile looks similar to this:

/etc/apparmor.d/usr.bin.test

```
#include <tunables/global>

profile test /usr/lib/test/test_binary {
    #include <abstractions/base>

    # Main libraries and plugins
    /usr/share/TEST/** r,
    /usr/lib/TEST/** rm,

    # Configuration files and logs
    @{HOME}/.config/ r,
```

```
    @{HOME}/.config/TEST/** rw,
}
```

Strings preceded by a '@' symbol are variables defined by abstractions (`/etc/apparmor.d/abstractions/`), tunables (`/etc/apparmor.d/tunables/`) or by the profile itself. `#include` includes other profile-files directly. Paths followed by a set of characters are **access permissions (http://wiki.apparmor.net/index.php/AppArmor_Core_Policy_Reference#File_access_rules)**. Pattern matching is done using **AppArmor's globbing syntax (http://wiki.apparmor.net/index.php/AppArmor_Core_Policy_Reference#AppArmor_globbing_syntax)**.

Most common use cases are covered by the following statements:

- `r` — read: read data
- `w` — write: create, delete, write to a file and extend it
- `m` — memory map executable: memory map a file executable
- `x` — execute: execute file; needs to be preceded by a **qualifier (http://wiki.apparmor.net/index.php/AppArmor_Core_Policy_Reference#Execute_rules)**

Remember that those permission do not allow binaries to exceed the permission dictated by the Discretionary Access Control (DAC).

This is merely a short overview, for a more detailed guide be sure to have a look at the **documentation (http://wiki.apparmor.net/index.php/AppArmor_Core_Policy_Reference)**.

Parsing profiles

To load (enforce or complain), unload, reload, cache and stat profiles use `apparmor_parser`. The default action (`-a`) is to load a new profile in enforce mode, loading it in complain mode is possible using the `-C` switch, in order to overwrite an existing profile use the `-r` option and to remove a profile use `-R`. Each action may also apply to multiple profiles. Refer to **apparmor_parser(8) (http://man.cx/apparmor_parser(8))** man page for more information.

Security considerations

Preventing circumvention of path-based MAC via links

This section refers back to outdated kernels and should be considered obsolete. Previously AppArmor can be circumvented via hardlinks in the standard POSIX security model. However, the kernel **included (https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=800179c9b8a1e796e441674776d11cd4c05d61d7)** the ability to prevent this vulnerability via the following settings:

```
/usr/lib/sysctl.d/50-default.conf
```

```
...
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
```

Kernel patches distributed by Ubuntu as workarounds are not needed.

Tips and tricks

Get desktop notification on DENIED actions

The notify daemon displays desktop notifications whenever AppArmor denies a program access. The script must be started at each boot and needs a few additional parameters:

```
# aa-notify -p -f /var/log/audit/audit.log --display $DISPLAY
```

The daemon relies on the auditing events being logged to a text file which can be specified using `-f`. To circumvent **systemd** not logging to a file it is necessary to **enable** `auditd.service` and pass its log file to `aa-notify`. No special auditing rules are necessary for this to work, therefore the overhead is not as significant as it was when **#Auditing and generating profiles**.

Cache profiles

Since AppArmor has to translate the configured profiles into a binary format it may take some time to load them. Besides being bothersome for the user, it may also increases the boot time significantly!

To circumvent some of those problems AppArmor can cache profiles in `/etc/apparmor.d/cache/`. However this behaviour is disabled by default therefore it must be done manually with `apparmor_parser`. In order to write to the cache use `-W` (overwrite existing profiles with `-T`) and reload the profiles using `-r`. Refer to **#Parsing profiles** for a brief overview of additional arguments.

See also

- **AppArmor wiki (http://wiki.apparmor.net/)**
- **AppArmor Core Policy Reference (http://wiki.apparmor.net/index.php/AppArmor_Core_Policy_Reference)** — Detailed description of available options in a profile
- **Ubuntu Tutorial (http://ubuntuforums.org/showthread.php?t=1008906)** — General overview of available utilities and profile creation
- **Ubuntu Wiki (https://help.ubuntu.com/community/AppArmor)** — Basic command overview
- **AppArmor Versions (http://wiki.apparmor.net/index.php/AppArmor_versions)** — Version overview and links to the respective release notes
- `apparmor.d(5) (http://manpages.ubuntu.com/manpages/artful/man5/apparmor.d.5.html)` — Structure of the AppArmor configuration directory
- `apparmor_parse(8) (http://manpages.ubuntu.com/manpages/artful/man8/apparmor_parser.8.html)` — The most fundamental AppArmor utility to load, unload, cache and stat profiles

- **Kernel Interfaces (http://wiki.apparmor.net/index.php/Kernel_interfaces)** — Low level interfaces to the AppArmor kernel module
- **Apparmor Profile Migration (https://wiki.ubuntu.com/ApparmorProfileMigration)** — Emergence of profiles
- **wikipedia:Linux Security Modules** — Linux kernel module on which basis AppArmor is build upon
- **Launchpad Project Page (https://launchpad.net/apparmor)**
- **FS#21406 (https://bugs.archlinux.org/task/21406)** — Initial discussion about the introduction of AppArmor