COLLABORATORY

CYBER SECURITY HUB AT TU DUBLIN

apparmor.d

Building the largest working set of AppArmor profiles

https://github.com/roddhjav/apparmor.d

Alexandre Pujol Security Research Lead · The Collaboratory @TUDublin, Ireland

✉ alexandre@pujol.io

roddhjav @roddhjav@mamot.fr

November 4, 2023

The Collaboratory @TUDublin

COLLABORATORY
CYBER SECURITY HUB AT TU DUBLIN

Irish, Public *Start Up*

Owned by TUDublin

6 Employees

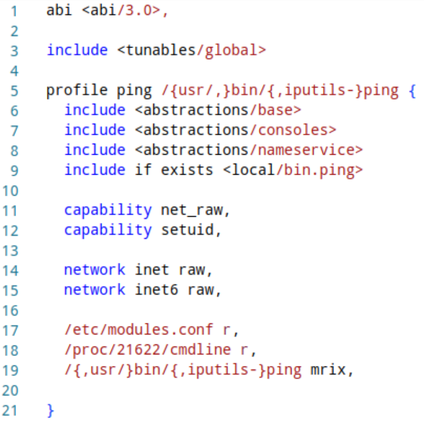
Cyber Security Hub between academic & industry

Physical Security Operation Centre (SOC) for training

Introduction

AppArmor

- Mandatory Access Control (MAC)
- Only what is explicitly required is authorized
- Program confined under profile
- Profiles are path based



```
abi <abi/3.0>,

include <tunables/global>

profile ping /{usr/,}bin/{,iputils-}ping {
  include <abstractions/base>
  include <abstractions/consoles>
  include <abstractions/nameservice>
  include if exists <local/bin.ping>

  capability net_raw,
  capability setuid,

  network inet raw,
  network inet6 raw,

  /etc/modules.conf r,
  /proc/21622/cmdline r,
  /{,usr/}bin/{,iputils-}ping mrix,

}
```

Figure 1: AppArmor profile for *ping*

Introduction

Apparmor is Enabled by default on Ubuntu, however:

- Profiles loaded: 39
- Processes confined: 4

Without profiles AppArmor is useless

Introduction

History

- Focus on server
- Focus on processes that face users or internet
- And...

”The permissions are fine why isn’t this working?

OMG apparmor again

Who turned that on here?

`systemctl disable apparmor`

Problem is solved!”

apparmor.d - *Full set of AppArmor profiles*

There are over 50000 Linux packages and even more applications.

Not possible to write profiles for all of them.

What to confine and why?

Let's go back to the basic...

Architecture

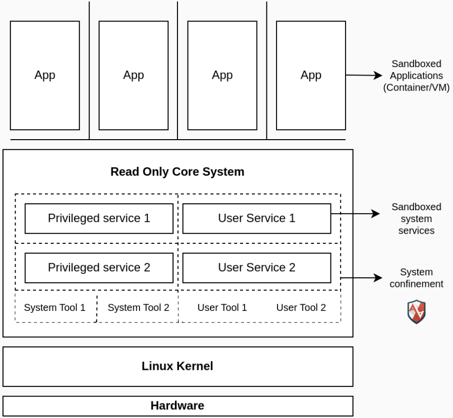
COLLABORATORY
CYBER SECURITY HUB AT TU DUBLIN

Security Model

Do not trust everything that runs on your computer

Implementation

- Secure boot
- Immutable core system
- Sandboxed system/user applications & services
- Full system confinement



Architecture

Confine

- All root processes:

  systemd, *bluetooth*, *dbus*, *polkit*, *NetworkManager*, *OpenVPN*, *tailscale*, *gdm*, *rtkit*, *colord*...

- Desktop environments: Gnome, KDE
- All user services such as *Pipewire*, *Gvfsd*, *dbus*, *xdg*, *xwayland*, *xorg*...
- All *"sandbox managers"*:

  flatpak, *snap*, *docker*, *podman*, *toolbox*, *libvirt*, *steam*...

- Some *"special"* user applications: web browser, file browser...

Architecture

Project Rules

1. **Mandatory Access Control**
   Only what is explicitly required should be authorized.

2. **Do not break a program**
   Should not break a normal usage of the confined software.

3. **Do not confine everything**
   Some programs should not be confined by a MAC policy.

4. **Distribution and devices agnostic**
   No *”It works on my machine”*

Demo

Structure

AppArmor profiles can be written without any strict guidelines.

Over 1000 profiles, you need one

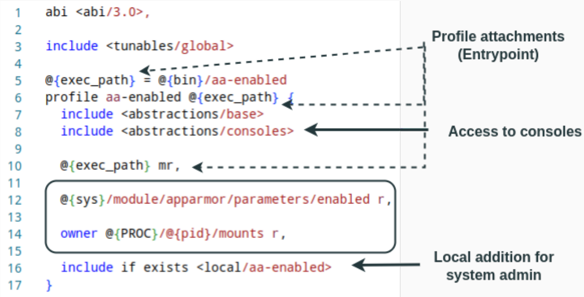
Purpose

- If a rule is present in a profile, it must only be in one location.
- Check if a profile has access to a given resource
- Check if a strict Write xor Execute policy is enforced

Example: aa-enabled

aa-enabled

- **What:** Check if apparmor is enabled
- **How:**
  1. Read module settings
  2. Ensure */sys/kernel/security* is mounted



```
abi <abi/3.0>,

include <tunables/global>

@{exec_path} = @{bin}/aa-enabled
profile aa-enabled @{exec_path} {
  include <abstractions/base>
  include <abstractions/consoles>

  @{exec_path} mr,

  @{sys}/module/apparmor/parameters/enabled r,

  owner @{PROC}/@{pid}/mounts r,

  include if exists <local/aa-enabled>
}
```

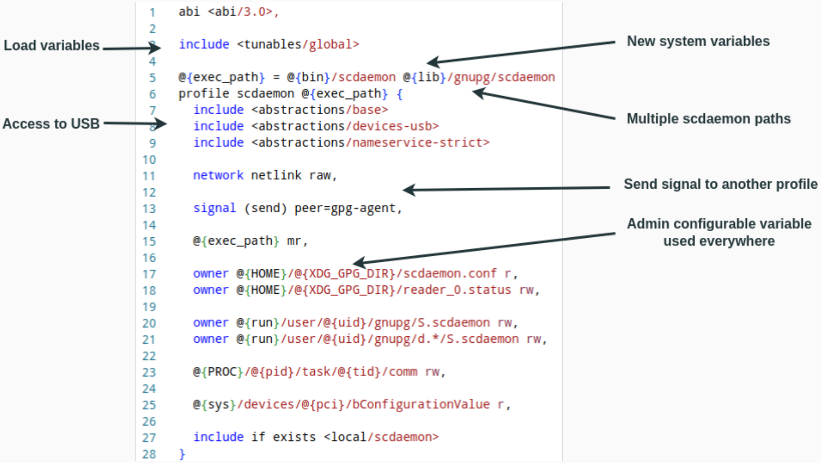
Profile attachments (Entrypoint)

Access to consoles

Local addition for system admin

Figure 2: Profile for *aa-enabled*

Example: scdaemon



```
abi <abi/3.0>,

include <tunables/global>

@{exec_path} = @{bin}/scdaemon @{lib}/gnupg/scdaemon
profile scdaemon @{exec_path} {
  include <abstractions/base>
  include <abstractions/devices-usb>
  include <abstractions/nameservice-strict>

  network netlink raw,

  signal (send) peer=gpg-agent,

  @{exec_path} mr,

  owner @{HOME}/@{XDG_GPG_DIR}/scdaemon.conf r,
  owner @{HOME}/@{XDG_GPG_DIR}/reader_0.status rw,

  owner @{run}/user/@{uid}/gnupg/S.scdaemon rw,
  owner @{run}/user/@{uid}/gnupg/d.*/S.scdaemon rw,

  @{PROC}/@{pid}/task/@{tid}/comm rw,

  @{sys}/devices/@{pci}/bConfigurationValue r,

  include if exists <local/scdaemon>
}
```

Load variables
Access to USB
New system variables
Multiple scdaemon paths
Send signal to another profile
Admin configurable variable used everywhere

Figure 3: Profile for *scdaemon* - Smartcard daemon for GnuPG.

Structure

Children Profiles

Special profiles with common resources

Example: `xdg-open`

GUI programs that can open resources: link, image...

```
@{bin}/xdg-open rPx -> child-open,
```

Example: `foo | less`

```
@{bin}/less rPx -> child-pager,
@{bin}/more rPx -> child-pager,
@{bin}/pager rPx -> child-pager,
```

Distributions Agnostic

AppArmor is path based

+

Distributions like to have they own things

=

💣

Distributions Agnostic

Firefox attachments

```
/{usr/,}bin/firefox{,.sh,-esr,-bin}
/{usr/,}lib{,32,64}/firefox{,.sh,-esr,-bin}/firefox{,.sh,-esr,-bin}
/opt/firefox{,.sh,-esr,-bin}/firefox{,.sh,-esr,-bin}
```

Colord attachments

```
/{usr/,}lib{,exec}/{,colord/}colord
```

Network Manager attachments

```
/{usr/,}{,s}bin/NetworkManager
```

Distributions Agnostic

Merged /usr

- Not here yet.
- `/bin/foo` and `/usr/bin/foo` are not the same thing

Different Apparmor versions

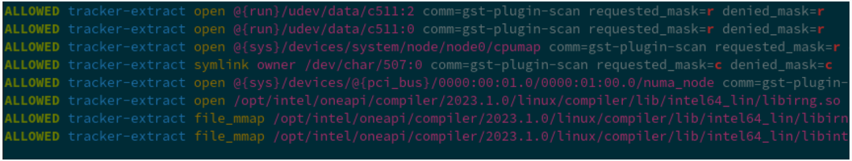
apparmor `2.x`, `3.0`, `3.1` & `4.0` different features set

Different programs versions

Compatibility issues between software versions on Arch, OpenSUSE, Ubuntu & Debian

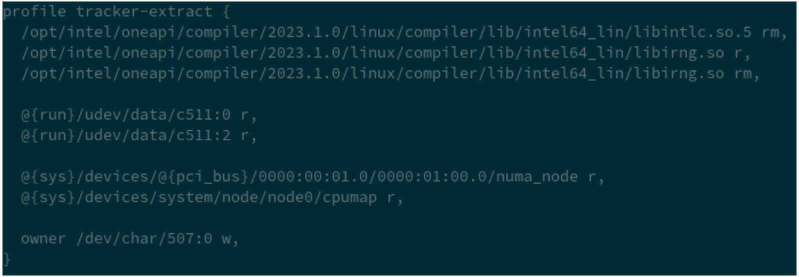
We need maintainer

Development Platform



```
ALLOWED tracker-extract open @{run}/udev/data/c511:2 comm=gst-plugin-scan requested_mask=r denied_mask=r
ALLOWED tracker-extract open @{run}/udev/data/c511:0 comm=gst-plugin-scan requested_mask=r denied_mask=r
ALLOWED tracker-extract open @{sys}/devices/system/node/node0/cpumap comm=gst-plugin-scan requested_mask=r
ALLOWED tracker-extract symlink owner /dev/char/507:0 comm=gst-plugin-scan requested_mask=c denied_mask=c
ALLOWED tracker-extract open @{sys}/devices/@{pci_bus}/0000:00:01.0/0000:01:00.0/numa_node comm=gst-plugin-
ALLOWED tracker-extract open /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64_lin/libirng.so
ALLOWED tracker-extract file_mmap /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64_lin/libirn
ALLOWED tracker-extract file_mmap /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64_lin/libint
```

Figure 4: *aa-log tracker-extract*



```
profile tracker-extract {
  /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64_lin/libintlc.so.5 rm,
  /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64_lin/libirng.so r,
  /opt/intel/oneapi/compiler/2023.1.0/linux/compiler/lib/intel64_lin/libirng.so rm,

  @{run}/udev/data/c511:0 r,
  @{run}/udev/data/c511:2 r,

  @{sys}/devices/@{pci_bus}/0000:00:01.0/0000:01:00.0/numa_node r,
  @{sys}/devices/system/node/node0/cpumap r,

  owner /dev/char/507:0 w,
}
```

Figure 5: *aa-log -r tracker-extract*

Development Platform

Development VM

- Distribution: Archlinux, Ubuntu, OpenSUSE, Debian
- Flavor: Server, Gnome, KDE

Dev stack

- **Packer**, **cloud-init** to generate the VM images,
- **Vagrant** to manage the test VMs

Integration Tests

How to tests AppArmor profiles?

Scope

- Profiles installed by a package manager
- In the context of the target distribution & flavor
- All profiles working together
- Detect update

Concept

- Run a profiled program in a VM and see what goes wrong
- Not testing the program, but the profile

How to run a program? *`foo`*, *`foo --help`*, *`foo --do-something`*

Integration Tests

Need a lot of tests

Stupid idea?

Tldr Pages - https://tldr.sh
Simplified and community-driven man pages

$_

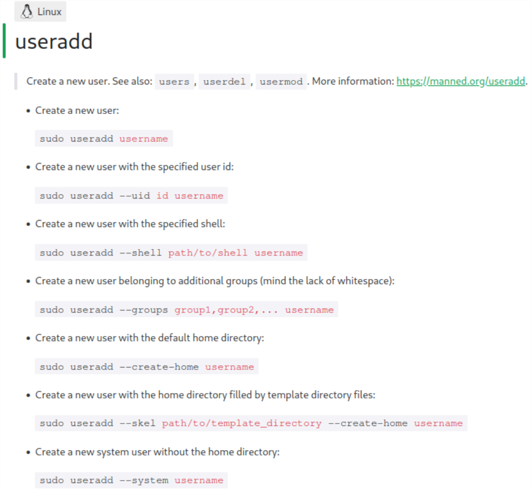
tldr-pages

COLLABORATIVE CHEATSHEETS FOR CONSOLE COMMANDS

Integration Tests

Not so stupid

- Give tests for hundred of commands
- Can be fully automated
- More complex than *`foo --help`*
- Can be used as base to manage hundred of tests: generated and manual



Linux

useradd

> Create a new user. See also: `users`, `userdel`, `usermod`. More information: https://manned.org/useradd.

- Create a new user:

```
sudo useradd username
```

- Create a new user with the specified user id:

```
sudo useradd --uid id username
```

- Create a new user with the specified shell:

```
sudo useradd --shell path/to/shell username
```

- Create a new user belonging to additional groups (mind the lack of whitespace):

```
sudo useradd --groups group1,group2,... username
```

- Create a new user with the default home directory:

```
sudo useradd --create-home username
```

- Create a new user with the home directory filled by template directory files:

```
sudo useradd --skel path/to/template_directory --create-home username
```

- Create a new system user without the home directory:

```
sudo useradd --system username
```

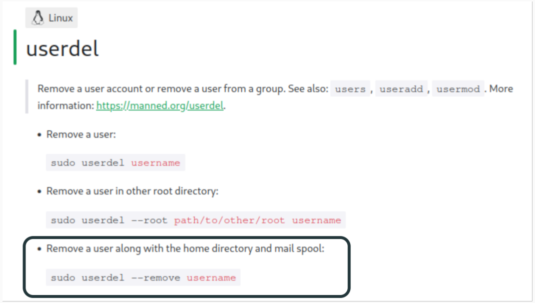
More stupid than you think

- What are?

  username

  path/to/other/root

- Interactive program?
- Not enough for complex GUI such as Gnome and KDE
- Better on a VM



Linux

userdel

> Remove a user account or remove a user from a group. See also: `users`, `useradd`, `usermod`. More information: https://manned.org/userdel.

- Remove a user:

```
sudo userdel username
```

- Remove a user in other root directory:

```
sudo userdel --root path/to/other/root username
```

- Remove a user along with the home directory and mail spool:

```
sudo userdel --remove username
```

Integration Tests

Apparmor profile test suite

- Use TLDR to bootstrap scenarios
- Work in Progress
  - Over 150 profiles tested
  - Over 350 tests

Results

- Early profiles tested raised a few apparmor logs
- Kind of logs raised

Useless	Missing consoles access
Useful	Missing execution & transition

```yaml
- name: acpi
  profiled: true
  root: false
  require: []
  arguments: {}
  tests:
    - dsc: Show battery information
      cmd: acpi
      stdin: []
    - dsc: Show thermal information
      cmd: acpi -t
      stdin: []
    - dsc: Show cooling device information
      cmd: acpi -c
      stdin: []
    - dsc: Show thermal information in Fahrenheit
      cmd: acpi -tf
      stdin: []
    - dsc: Show all information
      cmd: acpi -V
      stdin: []
    - dsc: Extract information from `/proc` instead of `/sys`
      cmd: acpi -p
      stdin: []
```

Future work

1. Upstream integration
2. Rewrite Dbus rules
3. Full system policy
4. Tests, Tests, Tests

Help is welcome

Code	*github.com/roddhjav/apparmor.d*
Docs	*apparmor.pujol.io*

Questions?

Code	*github.com/roddhjav/apparmor.d*
Docs	*apparmor.pujol.io*