

Securing Linux Applications: A Deep Dive into SELinux & AppArmor

Check GitHub for helpful DevOps tools:

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

 <https://github.com/MichaelRobotics>



Ask Personal AI Document assistant to learn
interactively (FASTER)!

1


Download PDF

1

<https://github.com/MichaelRobotics/DevOpsTools/blob/main/LinuxAppSecurity.pdf>

2

Go to website

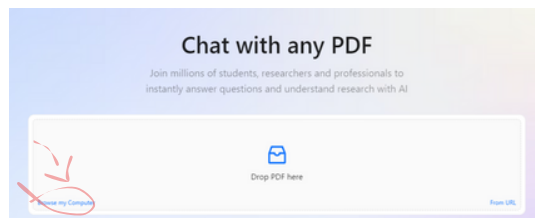
 | Click there to go to ChatPdf website

2

3

Browse file

3



4

Chat with Document

Ask questions about document!

4

Completly new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

 <https://www.hackthebox.com/>



What is SELinux & AppArmor?

SELinux (Security Enhanced Linux) is a security module that enforces strict access controls based on policies, restricting processes to the least privilege needed. AppArmor uses text-based profiles to define application permissions, limiting their interactions with files, directories, and networks.

How it Securing Linux Applications is done?

SELinux and AppArmor secure Linux applications by applying Mandatory Access Control (MAC) rules that restrict applications to specified actions, thus preventing unauthorized access. Both tools allow system administrators to switch between strict enforcement and permissive logging modes to test configurations.

Securing Linux Applications: Why and When

Security tools like SELinux and AppArmor are essential to minimize the risk of unauthorized access by limiting what applications can do on a system. They are particularly useful in server environments to ensure services are running with only the permissions they need.

Example: If an administrator wants to run an SSH service on a non-standard port, SELinux may block this action by default. Configuring SELinux to permit the new port ensures security without disabling SELinux entirely.

System Requirements

- 1 GB RAM (minimum, 2 GB recommended for better performance)
- 10 GB free storage (more may be required depending on the RAID setup and data size)
- Ubuntu 22.04

Securing Linux Applications: Main components & packages

- **SELinux** (Security Enhanced Linux) – A kernel module enforcing Mandatory Access Control (MAC) policies, restricting processes to defined permissions to enhance security.
- **AppArmor** – A security framework using text-based profiles to limit application actions, controlling file and network access in enforce or complain modes.
- **policycoreutils-python-utils** – Utilities for managing SELinux, including semanage for modifying policies, ports, and file contexts.

Linux App Security: Apparmor

1) Apparmor intro:

AppArmor operates using plain text profiles that define permissions and access controls for applications. Some profiles come pre-installed with the OS, while others can be added automatically by applications or manually by administrators.

2) Main Apparmor modes:

Similarity as in SELinux

- **Enforce mode:** Grants only necessary permissions to applications.
- **Complain mode:** Allows restricted actions but logs potential issues

Apparmor status can be shown:

```
sudo apparmor_status
```

Screenshot indicates that profiles like: /usr/bin/service are in **enforce mode**

```
laptopdev@laptopdev2:~$ sudo apparmor_status
apparmor module is loaded.
62 profiles are loaded.
59 profiles are in enforce mode.
 /snap/snapd/21465/usr/lib/snapd/snap-confine
 /snap/snapd/21465/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /snap/snapd/21759/usr/lib/snapd/snap-confine
 /snap/snapd/21759/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
 /usr/bin/evince
 /usr/bin/evince-previewer
 /usr/bin/evince-previewer//sanitized_helper
```

And profiles like: snap.kubeadm.kubeadm are in complain mode

```
virt-aa-helper
3 profiles are in complain mode.
 snap.code.code
 snap.code.url-handler
 snap.kubeadm.kubeadm
0 profiles are in kill mode.
```

To switch a profile currently in **enforce mode** to **complain mode**:

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

vice versa:

```
$ sudo aa-enforce /path/to/file
```

Wildcards are allowed in the above cases. For example,

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

To entirely disable a profile, create a symbolic link in the `/etc/apparmor.d/disabled` directory:

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

3) Apparmor profile config

An AppArmor profile is defined as a set of rules that specify the permissions and restrictions for a particular application or process. These rules determine what resources (files, directories, network access, etc.) the application can access, and what actions it can perform. The profile is typically written in a plain-text file and is stored in the `/etc/apparmor.d/` directory on the system.

Profile Declaration

The profile starts with the keyword `profile`, followed by the path to the executable file that the profile is meant to protect

```
profile /usr/bin/myapp {  
    # Profile rules go here  
}
```

File Permissions

The rules specify which files and directories the application can read, write, or execute

```
/usr/bin/myapp r,  
/var/log/myapp.log rw,  
/tmp/ r,
```

Network Access

AppArmor can restrict or allow network access. For example

```
network inet tcp,
```

Capabilities

You can specify capabilities (special permissions) that the application may need. For instance:

```
capability net_bind_service,
```

Entry and Exit Rules

Profiles can specify what happens when a program enters or exits a particular state:

```
/usr/bin/myapp flags=(attach_disconnected,complain) { # Rules }
```

Profiles are usually located in the `/etc/apparmor.d/` directory. For example:

- **`/etc/apparmor.d/usr.bin.myapp`**

After creating or editing a profile, you need to reload AppArmor to apply the changes:

```
sudo apparmor_parser -r /etc/apparmor.d/usr.bin.myapp
```

To enable or disable a profile:

```
sudo aa-enforce /etc/apparmor.d/usr.bin.myapp # Enforce mode
```

or

```
sudo aa-complain /etc/apparmor.d/usr.bin.myapp # Complain mode
```

4) auto profile generation

Use aa-logprof to read through the AppArmor logs and interactively update the profile:

```
sudo aa-logprof
```

You can also use the aa-genprof tool to guide you through creating a profile while observing the application's behavior:

```
sudo aa-genprof /usr/bin/myapp
```

Linux App Security: SELinux

1) Intro

To enhance security beyond standard ugo/rwx permissions and access control lists, the NSA developed SELinux (Security Enhanced Linux), a flexible Mandatory Access Control (MAC) system. It restricts processes' access to system objects (files, directories, network ports, etc.) to the minimum necessary, while allowing for later adjustments to the security model.

2) SELinux modes

1. **Enforcing:** Denies access based on policy rules.
2. **Permissive:** Logs actions that would be denied in enforcing mode without blocking them.

SELinux can be disabled, though it's better to learn to use it.

To check the current mode, use

```
getenforce
```

Switch modes with:

```
sudo setenforce 1 #Switches SELinux to enforcing mode.
```

or

```
sudo setenforce 0 #Switches SELinux to permissive mode
```


This change isn't persistent; to make it last, edit `/etc/selinux/config` and set SELINUX:

```
SELINUX=enforcing
```

or

```
SELINUX=permissive
```

or

```
SELINUX=disable
```

If `getenforce` shows Disabled, change the config file and reboot. Switching modes is often used for troubleshooting to identify SELinux permission issues.

3) SSH example

System administrators often change the default SSH port (22) to a non-standard port, like 8888, to reduce the risk of automated attacks and port scans.

However You might see a message indicating that SELinux is preventing sshd from binding to port 8888 because it is reserved for another service

There are 2 solutions: 1) You can configure SELinux to Allow SSH on the New Port through updating its rules 2) disable SELinux.

In case 2) first, ensure you have the `polycoreutils-python` package installed:

```
sudo apt install polycoreutils-python-utils
```

Assign 8888 port to ssh service if you are sure that actually there is no service bound to this port:

```
semanage port -m -t ssh_port_t -p tcp 8888
```

common troubleshooting

1) Application Denied Access by SELinux

Check `/var/log/audit/audit.log` for SELinux denials. Use `audit2why` to understand the cause and `audit2allow` to create a policy if needed.

2) AppArmor Blocking Application Functionality

Switch the profile to complain mode using `aa-complain` to allow the application to run and log issues. Use `aa-logprof` to analyze logs and update the profile.

3) Service Fails to Start Due to SELinux Permissions

Confirm the SELinux context of files using `ls -Z` and update with `chcon` or `restorecon`. Adjust the policy with `semanage` if needed.

4) Application Fails After Moving to New Directory (SELinux)


Use `restorecon -R /path/to/directory` to reset file contexts. Ensure the new directory has the correct SELinux labels.

Learn more about SELinux & Apparmor & Linux

Check TecMint - great docs!

Implementing Mandatory Access Control

Introduction to AppArmor & SELinux

 <https://www.tecmint.com/mandatory-access-control-with-selinux-or-apparmor-linux/2/>



Linux File System Explained

The concept of boot loading

 <https://www.tecmint.com/linux-file-system-explained/>



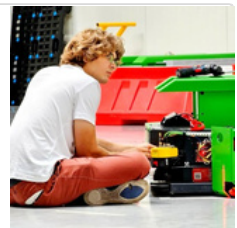
Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

<https://github.com/MichaelRobotics>



PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!