Vulnerabilities, Patching, and Security

SELinux, AppArmor, CVE

- SELinux (Security Enhanced)
- AppArmor
- CVE Vulnerabilities
- Rootkit Scans
- Malware Scans

SELinux (Security Enhanced)

==========

What is SELinux?

SELinux is a kernel-level access control system. SELinux acts like a gatekeeper, enforcing rules about what users, programs, and services can access on a system. SELinux is a complex but effective security tool. While it might seem like overkill for some users, it offers a strong layer of defense for those who need to seriously tighten up system security.

SELinux Enforcement Modes

SELinux comes pre-installed on most new RHEL systems (most likely not enabled, or set into an inactive mode).

Check SELinux status

```
sestatus
```

SELinux has 3 modes:

enforcing	the strictest security setting. When enabled, SELinux actively enforces the security policies it has been configured with.
permissive	SELinux logs attempted violations of the security policy but doesn't block them. This can be useful for troubleshooting purposes or when initially configuring SELinux policies for new applications.
disabled	SELinux is disabled and it is not having any impact.

Changing SELinux mode

```
setenforce chosenmode
```

Check SELinux enforcement mode

```
getenforce
```

Access Levels

In SELinux, every process and system resource has a security label called a context. This context is like an ID card that defines the security properties of that process or resource. The SELinux policy uses these contexts along with a set of rules to dictate how processes can interact with each other and access system resources.

Here's a breakdown of the key aspects of access levels for processes in SELinux:

- **SELinux Context:** This context contains multiple fields, including user, role, type, and a security level.
  - **SELinux Type:** This is a crucial part of the context, often ending in "_t". For instance, a web server process might have a type of "httpd_t". SELinux policy rules primarily rely on these types to define allowed interactions between processes and resources.

- **SELinux Policy Rules:** These rules define what a process with a certain type is allowed to do with other processes and resources based on their types. By default, all interaction is denied unless a rule explicitly grants permission. DAC (Discretionary Access Controls - traditional file permission/ownership) rules are checked first, and SELinux rules only come into play if DAC allows access.

Check context of a process:

```
ps axfuZ | grep -i processname
```

Show context of a file

```
ls -lZ
```

Changing context of a file

```
chcon --type=servicetype_t /path/to/change
```

Ports

List all ports being monitored by SELinux

```
semanage port -l
```

Change port management

```
semanage -a -t portname_t -p TCP portnumber
```

example:

```
semanage -a -t http_port_t -p TCP 8080
```

In this context, we're wanting to enable Apache to access port 8080. Apache has a context specifically for setting port access. So this command is adding http_port_t to the allow configuration on port 8080.

-a	add
-d	delete
-t	define SELinux type
-p	protocol ie TCP or udp

Logging

SELinux logs all activity that it detects into the audit log (`/var/log/audit/audit.log`) when in enforcing or permissive mode.

AppArmor

==========

AppArmor is a high-level security system, primarily designed for use on Debian-based systems.

AppArmor itself is installed on most new Debian systems, however, to customise the configuration you'll need to ensure that the apparmor-utils package installed.

The primary difference between AppArmor and SELinux is that AppArmor bases its security policies off inode location, whereas SELinux uses a contextual system.

Check APPArmor status:

```
apparmor_status
```

Check app armor version:

```
apt policy apparmor
```

AppArmor profiles are stored within `/etc/apparmor.d`

CVE Vulnerabilities

Common Vulnerabilities and Exposures (CVE) is a system that provides a reference-method for publicly known information-security vulnerabilities and exposures.

A CVE-ID follows the format "CVE-YYYY-NNNN", where "YYYY" is the year the CVE-ID was assigned or published and "NNNN" is a unique number.

Checking CVE patching (RHEL)

```
rpm -q --changelog <package_name> | grep -i CVENUMBER
```

Checking CVE patching (Debian)

```
apt-get changelog <package_name> | grep -i CVENUMBER
```

Rootkit Scans

A rootkit is a collection of software tools that enable an attacker to gain root or administrative-level access to a computer or network and maintain this access covertly.

==========

chkrootkit

`chkrootkit` (Check Rootkit) is an open-source security tool used to detect rootkits and other malicious software on Linux systems.

To use chkrootkit, you'll need to install the `chkrootkit` package.

Running `chkrootkit`

To perform a basic scan, you simply run:

```
chkrootkit
```

Additional Options

-v	verbose output
-r /path/to/scan	Specify a specific path to scan
-q	suppress warnings
> /path/to/log	Specify log file for output

==========

RKHunter

`rkhunter` (Rootkit Hunter) is another popular open-source security tool designed to detect rootkits, backdoors, and other possible signs of compromise on Linux systems.

To use rkhunter, you'll need to install the `rkhunter` package.

Running rkhunter

A basic rootkit scan can be run using the below:

```
rkhunter --check
```

Additional Options

--update	Update rkhunter's database of known rootkits
--verbose	Verbose output
--logfile /path/to/log	Specify a log file for rkunter output

Understanding the Output

The output of `rkhunter` includes various sections and categories:

- **[OK]**: Indicates that the item being checked is within expected parameters.
- **[Warning]**: Highlights potential security issues or suspicious findings that should be investigated further.
- **[Suspicious]**: Flags items that may require attention due to unusual or unexpected behavior.
- **[Not Found]**: Indicates that an expected file or configuration item was not found.

---

rkhunter baseline

rkhunter includes the ability to create a 'baseline'. This essentially means that a scan of the system is run, and then future scans will compare against the existing baseline for any changes.

> If you suspect your system is compromised or infected with malware (including rootkits), refrain from using `rkhunter --propupd`. Running this command in such cases can potentially embed the infection into the baseline, compromising `rkhunter`'s ability to accurately detect the malware.

Create a baseline

```
rkhunter --propupd
```

==========

Malware Scans

==========

ClamAV

ClamAV is a widely used open-source antivirus engine designed for detecting viruses, malware, and other threats on Linux systems.

ClamAV Usage

```
clamscan [options] /path/to/scan
```

> Depending on the size of the path you're scanning, the scan can take a while. It would be worth running the scan in a screen to ensure that the process isn't interrupted or killed when your session closes.
>
> Start a new screen: `screen`
> Show screens: `screen -ls`
> Connect to existing screen: `screen -r name`

ClamAV Options

-r	recursive
-i	only print infected files.
-l /path/to/log	specify log file for clamscan output
--move=/path/to/dir	Move infected files to a specified directory

==========