Oracle Linux 10

Enhancing System Security

G35582-01
September 2025

Oracle Linux 10 Enhancing System Security,

G35582-01

Contents

Preface

Oracle Linux 10: Enhancing System Security describes features in Oracle Linux 10 that can enhance the security of systems. The guide also includes guidelines and recommendations for best security practices when working with Oracle Linux.

1

Overview of Security Principles

This section provides a brief overview of system security and includes some principles for how to enhance security on Oracle Linux systems.

Oracle Linux is a secure enterprise-class OS that can provide the performance, data integrity, and the application uptime necessary for business-critical production environments.

Thousands of production systems at Oracle run Oracle Linux, and many internal developers use it as their development platform. Oracle Linux is at the heart of Oracle Cloud Infrastructure and several Oracle engineered systems, including the Oracle Exadata Database Machine, Oracle Private Cloud Appliance, and Oracle Database Appliance. Oracle Linux is also used across Oracle cloud, whether it's infrastructure, database services, or other Software-as-a-Service (SaaS).

Backed by Oracle Support, these mission-critical systems, and deployments depend fundamentally on the built-in security and reliability features of Oracle Linux.

Oracle has been a regular participant in the Linux community, contributing code enhancements for the mainline Linux kernel. Oracle also contributes to many open source initiatives, such as Oracle Cluster File System and the Btrfs file system. From a security perspective, having roots within open source is a significant advantage.

The Linux community, which includes many experienced developers and security experts, reviews posted Linux code extensively before it's tested and released. The open source Linux community has supplied many security improvements over time, including access control lists (ACLs), cryptographic libraries, and trusted utilities. Oracle builds on such tools to provide a solid and secure OS.

Oracle recommends that you follow some fundamental security principles when using Oracle Linux. These principles are guidelines that administrators can use to build security policies.

Minimize and Secure the Software Footprint

Planning an Oracle Linux system's purpose, deployment configuration, and software requirements in advance is essential to minimizing attack vectors. During the design phase of a deployment you can uninstall or disable any components and services that aren't needed or used in a particular configuration or deployment scenario, including any peripheral functionality or components. Because deployment requirements can vary over time, you also need processes in place to uninstall and disable any features that aren't needed or used in specific configuration or deployment scenarios. You might also consider using the minimal install base environment that only installs the essential components of the OS by default. If you're using a kickstart configuration file to install Oracle Linux, the minimal install includes the `@base` and `@core` packages.

For more information about these installation options, see Oracle Linux 10: Installing Oracle Linux.

Find more information about the various base environments available by running the `dnf group list -v` command. This command displays the same list of base environments available on the Software Selection screen of the Oracle Linux GUI installer. For example:

```
dnf group list -v
```

```
Last metadata expiration check: 0:05:18 ago on Fri 30 May 2025 13:56:48 GMT.
Available Environment Groups:
   Server with GUI (graphical-server-environment)
   Server (server-product-environment)
   Minimal Install (minimal-environment)
...
Installed Environment Groups:
   Server with GUI (graphical-server-environment)
Installed Groups:
   Container Management (container-management)
...
Available Groups:
   Legacy UNIX Compatibility (legacy-unix)
...
```

To review what the minimal-environment group includes, run the `dnf group info` command. For example:

```
dnf group info minimal-environment
```

```
Last metadata expiration check: 0:05:18 ago on Fri 30 May 2025 13:56:48 GMT.
Environment Group: Minimal Install
 Description: Basic functionality.
 Mandatory Groups:
   Core
 Optional Groups:
   Standard
```

Use the same command to find out what packages are included in the core group. For example:

```
dnf group info core
```

```
Last metadata expiration check: 0:02:37 ago on Fri 30 May 2025 13:52:26 GMT.
Group: Core
 Description: Minimal host installation
 Mandatory Packages:
  audit
  basesystem
  bash
...
Default Packages:
NetworkManager-team
NetworkManager-tui
...
```

```
Optional Packages:
dracut-config-generic
...
```

To review more information about the individual packages, run the dnf info command on the packages. For example:

```
dnf info bash
```

```
Last metadata expiration check: 0:08:25 ago on Fri 30 May 2025 13:52:26 GMT.
Installed Packages
Name         : bash
Version      : 5.2.26
Release      : 6.el10
Architecture : x86_64
Size         : 8.1 M
Source       : bash-5.2.26-6.el10.src.rpm
Repository   : @System
From repo    : anaconda
Summary      : The GNU Bourne Again shell
URL          : https://www.gnu.org/software/bash
License      : GPL-3.0-or-later
Description  : The GNU Bourne Again shell (Bash) is a shell or command
language
             : interpreter that is compatible with the Bourne shell (sh). Bash
             : incorporates useful features from the Korn shell (ksh) and the
C shell
             : (csh). Most sh scripts can be run by bash without modification.
```

Another important way to ensure that Oracle Linux systems are secure is to only install those software packages that are essential for performing necessary functions. Extra functions and components can increase the security risk, so they can be removed or uninstalled as needed.

Installing software from secure, known, and trusted sources is considered good security practice. Oracle uses TLS to secure the networking actions of the software installation and update tools provided with Oracle Linux. Oracle also signs packages with GPG keys so that administrators can confirm the provenance and authenticity of software packages. For more information about verifying downloads with Oracle provided GPG keys, see https://linux.oracle.com/security/gpg/index.html.

Keep Software Up-to-date

One of the principles of good security practice is to keep all software versions and patches up-to-date. Oracle maintains software and releases errata and patch updates using the Oracle Linux yum server and the Unbreakable Linux Network (ULN).

Updating the installed software on Oracle Linux to patch any vulnerabilities and minimize the attack surface as often as possible is considered good security practice. For more information, see Understanding the Importance of Updates.

Also consider using Oracle Ksplice in addition to regular system updates to automatically patch the running kernel and common userspace libraries such as openSSL and glibc without any required system downtime. For more information about Ksplice, see Oracle Linux: Ksplice User's Guide.

Restrict Network Access to Critical Services

Keeping both middle-tier applications and databases behind a firewall restricts access to those systems to a known network route that you can monitor and restrict, or you can use a firewall router as a substitute for several independent firewalls.

If you can't use firewalls, you can access based on IP address. Restricting database access by IP address often causes application client/server programs to fail for DHCP clients. To resolve that problem, consider using static IP addresses, a software/hardware VPN or Windows Terminal Services or similar.

See About the Packet Filtering Firewall and Restricting Access to SSH Connections for more information on how to restrict and secure network access.

Control Authentication Mechanisms and Enforce Password Restrictions

You can select different authentication mechanisms to control access to a system. In environments where many systems are involved, consider using a centralized authentication tool so that you don't need to maintain accounts across many different systems.

Also consider the different types of authentication mechanisms available. While password-level access can be convenient, you can secure an environment further by providing more restrictive mechanisms such as key, certificate, or token based authentication that often use 2-factor authentication.

When using password-style access, you can enforce restrictions to prevent common, short, or easily cracked passwords. Consider the NIST 800-63 Digital Identity Guidelines, which suggest deviating from traditional password policy. Rather than forcing complicated passwords with frequent expiry and forced system lockout, consider requiring passwords that aren't easy to guess or crack and are checked against known password dictionaries.

See Configuring User Authentication and Password Policies for more information.

Follow the Principle of Least Privilege

The principle of least privilege suggests that you grant users the bare minimum privileges required to perform their jobs. The excessive granting of permissions, especially early on in an organization’s lifespan when few employees must complete work within tight deadlines, can leave systems wide open for abuse. Reviewing user privileges periodically to match their current job responsibilities is considered good security practice.

This principle requires that users are assigned their own login accounts. If they require administrator access for a purpose, use `sudo` to grant access for that specific purpose.

Distributing the root user password is considered poor security practice. You can enhance the security of the root password by ensuring that it's long, difficult to guess, and contains a wide variety of special characters.

See Checking User Accounts and Privileges for more information.

Monitor System Activity

Robust system security relies on three principles: up-to-date security protocols, correct system configuration, and frequent system monitoring. Auditing and reviewing audit records addresses the third requirement. Each component within a system often has some degree of monitoring capability. You can follow the audit advice in this document and monitor audit records.

See Oracle Linux 10: Auditing the System With Auditd and Rsyslogd and Using Advanced Intrusion Detection Environment for more information.

Also consider using the Ksplice known exploit detection feature with systems that have the Ksplice Enhanced client installed. That feature reports exploitation attempts from known attack vectors. When new Common Vulnerabilities and Exposures (CVEs) are discovered and patched by Ksplice, Oracle might add tripwires to the code that log when an erroneous condition is triggered to ensure that administrators can monitor systems for suspicious activity. For more information about Ksplice, see Oracle Linux: Ksplice User's Guide.

Keep Up-to-date With the Latest Security Information

For information about common vulnerabilities, exposures, and errata, you can use ULN or sign up for one of the Oracle Linux mailing lists. For more information, see Obtaining Errata and CVE Notices. You can also review Oracle's constantly expanding range of documentation, tutorials, and blog posts at https://docs.oracle.com/en/operating-systems/oracle-linux/ and https://blogs.oracle.com/linux/ for the latest information.

2

Planning for a Secure Oracle Linux Environment

This section describes how to plan a secure Oracle Linux environment based on specific security requirements.

To better understand those security requirements, consider the following questions:

Which resources must be protected?
Many resources in the production environment can be protected, such as information in databases accessed by WebLogic Server and the availability, performance, applications, and the integrity of a website. You can evaluate the resources that require protection to decide the level of security to provide for each of them.

From whom must those resources be protected?
For most websites and online services, resources must be protected from everyone on the Internet. You might also consider restricting employee access on a company intranet to only the resources to which they need access, and only granting access for highly confidential data or strategic resources to a few trusted system administrators. In some scenarios it might be better for system administrators to not have direct access to data and resources until they switch to a user account with fewer privileges.

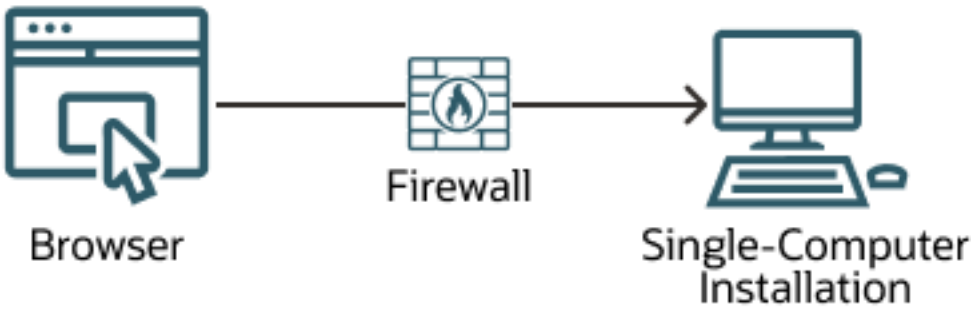
What could happen if the protections on strategic resources fail?
A minor fault in a security scheme could be easily detected and considered nothing more than an inconvenience. In severe cases, a fault might cause significant damage to companies or individual clients that use the website. Understanding the security ramifications of each resource can help you to ensure that they're robustly protected.

Recommended Deployment Configurations

This section describes recommended architectures for deploying Oracle products with secure Internet access.

Figure 2-1 shows a simple deployment architecture.

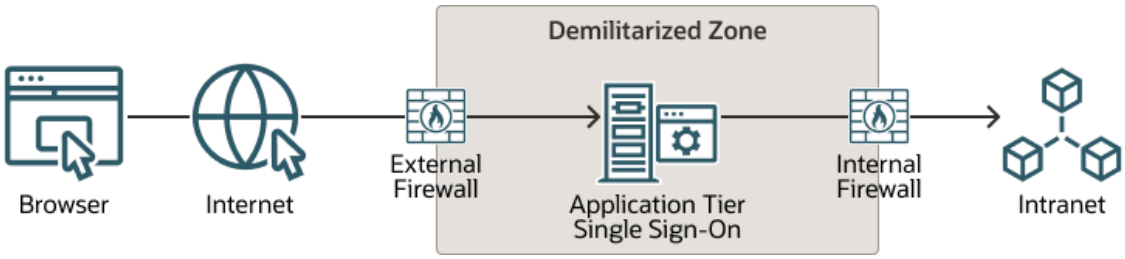
Figure 2-1 Simple Firewall Deployment Configuration



This single-computer deployment can be cost effective for small organizations. However, it can't provide high availability because all components are stored on the same computer.

Figure 2-2 shows a good practice configuration based on an Internet-Firewall-DMZ-Firewall-Intranet architecture.

Figure 2-2 DMZ Deployment Configuration



A "demilitarized zone" (DMZ) refers to a server that's isolated by firewalls from both the Internet and the intranet, and which acts a buffer between them. The firewalls that separate DMZ zones provide two essential functions:

- Blocking any traffic types that aren't allowed.
- Providing intrusion containment if any successful intrusions take over processes or processors.

Component Security

Each application software component often has its own security considerations that you can evaluate independently of those that apply to the OS. See the security guidelines for each component to decide how best to configure it to fit the security requirements for each environment.

3
Managing System Security

This section describes Oracle Linux features that administrators can use to manage system security.

Oracle Linux provides a complete security stack, from network firewall control to access control security policies, and is configured to be secure by default. Oracle Linux includes features that can help you to enhance system security, such as real-time patching, automated software updates, certificate management tools, a built-in firewall, mandatory access controls, and public key cryptography and data encryption tools. This section describes each of these features and discuss where you can find more information.

Consider using a combination of these tools and facilities to manage access that's provisioned on the system and improve the security of the OS, applications that run on the system and network connectivity.

Understanding the Importance of Updates

Keeping system software up-to-date is an important principle of an overall security strategy as described in Overview of Security Principles. Updating Oracle Linux is important to avoid and protect against software vulnerabilities and known attack vectors that malicious hackers can exploit to gain unauthorized access to systems.

Oracle releases important updates to the Oracle Linux and Oracle VM software as individual package updates, known as errata.

How often Oracle Linux systems are updated depends on many factors that each system administrator must assess. One good security practice option to keep systems secure is accepting the latest available updates, because the more out-of-date a system is, the more potential vulnerabilities and attack vectors that could be exposed. The older the vulnerabilities are, the longer that malicious hackers can gather experience and expertise, and the more likely that systems with those vulnerabilities might be hacked or exploited.

Before you configure the frequency and type of updates that are installed, you must decide the level of exposure and risk that you consider to be acceptable and still meet security requirements.

Here are some examples of useful best practices to consider:

- Monitor errata as they're published by Oracle, reviewing errata as they appear and paying careful attention to security errata. Security errata are listed by severity level from critical to low.
- Schedule time to test and deploy security, bug, and enhancement errata. Testing and deploying security and bug errata as soon as possible is considered good security practice.
- Pay attention to minor releases as they often contain a combination of security patches, bug fixes, and feature enhancements. As stated before, you must decide the frequency and timing for testing and deploying the larger set of updates.
- Reduce the number of updates Oracle Linux requires by installing only the minimum software required to suit business needs. Minimizing the package footprint on systems can reduce the amount of time needed to perform updates for software while also reducing

exposure to security attack vectors. For more information, see Minimize and Secure the Software Footprint.

- For systems with mission critical applications where system reboots are disruptive, consider using tools such as Ksplice to help you prevent zero day attacks by applying security errata patches with zero downtime in memory without needing a reboot. For more information about Ksplice, see Oracle Linux: Ksplice User's Guide.
- Also consider using software and automation management tools to apply software updates and deployments such as Oracle OS Management Hub, Oracle Linux Automation Manager, Chef, and Puppet.

  Such tools can also run incremental updates to select groups of Oracle Linux systems, which can be useful when testing the impact of an update on real users in a staging environment before rolling out to production. Similarly, you can also use these tools to add RPM packages to base images so that you can tailor Oracle Linux systems and images to user roles rather than deploying the same configuration for everything.

For more recommendations that could suit highly complex production environments, see Oracle Linux: Managing Software on Oracle Linux.

Installing and Updating Errata RPM Packages

You can obtain errata information directly from Oracle Linux systems by using Yum or DNF from any Oracle Linux terminal or from ULN. After assessing the errata information, you can track, install, and update errata using these tools.

Oracle Linux 10 provides tools to help you update the system often and with minimum interference. Consider using the `dnf-automatic` package to download updates on a schedule, alert you to software upgrade options and even to apply them automatically.

Always use DNF to install, update, or remove RPM packages (don't use the `rpm` command). You can use the `dnf update --security` command to update in the following ways:

- Update by CVE number,
- Update by security errata advisory number,
- Update all kernel packages to the latest kernel version that contains security errata,
- Update all security errata by severity level: critical, important, moderate, and low,
- Update all security errata to the latest release available. This option, from a security perspective, is often the best choice.

See Oracle Linux: Managing Software on Oracle Linux for more information on how to use these DNF commands.

Understanding RPM Errata Packages and Cumulative Updates

Oracle Linux is an RPM-based distribution. RPM packages are built cumulatively as Oracle releases updates consisting of security, bug, or enhancements errata. For example, release 2 for an RPM builds on release 1, and if release 1 is installed but you want to update to release 4, then the contents of releases 2 and 3 are also included in release 4.

Errata package RPM binaries include content that you can find by using the `dnf info` *`package`* command. For example:

```
dnf info bash
```

```
Last metadata expiration check: 0:08:25 ago on Fri 30 May 2025 13:52:26 GMT.
Installed Packages
Name         : bash
Version      : 5.2.26
Release      : 6.el10
Architecture : x86_64
Size         : 8.1 M
Source       : bash-5.2.26-6.el10.src.rpm
Repository   : @System
From repo    : anaconda
Summary      : The GNU Bourne Again shell
URL          : https://www.gnu.org/software/bash
License      : GPL-3.0-or-later
Description  : The GNU Bourne Again shell (Bash) is a shell or command
language
             : interpreter that is compatible with the Bourne shell (sh). Bash
             : incorporates useful features from the Korn shell (ksh) and the
C shell
             : (csh). Most sh scripts can be run by bash without modification.

Available Packages
Name         : bash
Version      : 5.2.26
Release      : 6.el10
Architecture : src
Size         : 11 M
Source       : None
Repository   : ol10_baseos_latest
Summary      : The GNU Bourne Again shell
URL          : https://www.gnu.org/software/bash
License      : GPL-3.0-or-later
Description  : The GNU Bourne Again shell (Bash) is a shell or command
language
             : interpreter that is compatible with the Bourne shell (sh). Bash
             : incorporates useful features from the Korn shell (ksh) and the
C shell
             : (csh). Most sh scripts can be run by bash without modification.
```

- **Name, version, and release information**: When you update a package, the release number of the RPM changes and the version number can change if the RPM is rebased (this happens infrequently).

  You can use the `dnf updateinfo --list --installed` *`package`* command to see a list of installed RPMs for a specific package with their version, release, and associated errata introduced. For example, this command lists all the errata applied to the bash package that also lists the name, version, release, and architecture of the package:

  ```
  dnf updateinfo --list --installed bash
  ```

- **Informational metadata**: This metadata includes information such as a summary, description, license, and so on.
- **Cryptographic signature**: All Oracle Linux RPM packages are signed and customers can use this signature to verify the provenance and authenticity of an RPM as it's being downloaded. The person and company who builds the binary provides this signature. Enabling `gpgcheck=1` as a global default in the `/etc/dnf/dnf.conf` file ensures that all RPMs are authentic and have a valid GPG signature before the `dnf` command downloads or installs them. This is an important way to ensure that RPMs come from a trusted source and haven't been changed. All Oracle Linux images have the Oracle GPG keys installed and the `gpgcheck` option enabled by default.

  Ensuring RPM package security is one aspect of an overall data encryption strategy. For more information, see About Data Encryption.
- **Dependency information**: This provides details about RPMs and versions of other packages that an RPM depend upon. For example, you can use the following command to list the direct dependencies for the bash RPM package:

  ```
  dnf deplist bash
  ```
- A series of scripts that can be run at various stages during update or installation.

About Security Errata and CVEs

A security errata is a corrective action intended to address security vulnerabilities identified in one or more Common Vulnerabilities and Exposures (CVEs).

CVE numbers are unique, common identifiers for publicly known information about security vulnerabilities. Oracle uses CVE numbers to identify and track corrective actions driven by a reported security vulnerability. The CVE program is cosponsored by the office of Cybersecurity and Communications at the US Department of Homeland Security and is managed by the MITRE corporation.

About Bug and Enhancement Errata

A bug is a corrective action generated from an issue discovered by a customer or a vendor. Oracle normally provides the bug ID involved in the event in the errata notification. Including bug updates in a maintenance policy is considered good security practice because they can prevent issues that haven't already been planned for or affected the system yet. For example, a bug errata might prevent a problem that isn't exposed until a combination of events occur that destabilizes a system. Therefore it's important to keep Oracle Linux systems updated with bug errata.

Enhancements are incremental new features or updates provided by Oracle.

Obtaining Errata and CVE Notices

To be notified when Oracle releases new errata packages, you can subscribe to the Oracle Linux mailing list at https://oss.oracle.com/mailman/listinfo/el-errata.

If you're signed in to ULN, you can also subscribe to the mailing list by following the **Subscribe to Enterprise Linux Errata mailing list** link that's provided in the Errata tab.

Oracle publishes a complete list of errata made available on ULN at https://linux.oracle.com/errata. You can also see a published listing of Common Vulnerabilities and Exposures (CVEs) and explore their details and status at https://linux.oracle.com/cve.

You can also track updates to Oracle Linux yum server repositories by visiting https://yum.oracle.com/whatsnew.html, where you can see which packages were updated within each repository for the previous six months.

About Certificate Management

Public key cryptography provides secure communication on an insecure public network and verification of the identity of the entity at the other end of a network connection. Public key cryptography is based on establishing asymmetric pairs of secret and public keys.

OpenSSL includes an open source implementation of the TLS and SSL protocols. If a hierarchy of trust is confined to an organization's intranet, you can use OpenSSL to generate a root certificate and set up a Certificate Authority (CA) for that domain. Alternately, you can use OpenSSL to generate a certificate signing request that can be provided to a recognized CA to obtain a signed certificate that you can use in an application configuration. Low-cost domain validation certificate signing is now more obtainable if you use the IETF standardized Automatic Certificate Management Environment (ACME) protocol as described in RFC 8555, reducing the requirement for costly expenditure around certificate signing and running a self-hosted CA.

For more detailed information, see Oracle Linux: Managing Certificates and Public Key Infrastructure.

About Data Encryption

Cryptographic libraries included with Oracle Linux can be used by software to provide data encryption facilities. You can use data encryption to protect data that's stored or being transmitted. Data on storage devices and media can be at risk of theft or device loss. Data being transmitted over local area networks and the Internet can be intercepted or altered. By encrypting data, you can help protect it while it's in storage or in transmission, thereby providing a safer infrastructure. In addition, data encryption to protect privacy and personal data is increasingly being made a mandatory requirement in corporate security policies and by governmental regulations (for example, HIPAA, GLBA, SOX, and PCI DSS).

Oracle Linux systems provide the following strategies for protecting data:

- When installing systems and application software, only accept RPM packages that have been digitally signed by a trusted source.

  To ensure that downloaded software packages are signed, set `gpgcheck=1` in the repository configuration file and import the GPG key provided by the software supplier. Oracle Linux images normally have this setting enabled as a global default in the `/etc/dnf/dnf.conf` file. You can also install RPMs using the Secure Sockets Layer (SSL) protocol, which uses encryption to protect the communications channel.

  For more information, see Oracle Linux: Managing Software on Oracle Linux.

- To protect against data theft, consider using full-disk encryption, especially on laptops, external hard drives, or removable devices such as USB memory sticks. Oracle Linux provides block device encryption by using the `dm-crypt` kernel module and the Linux Unified Key Setup (LUKS) format. The `cryptsetup` administration command is available in the `cryptsetup` package.

  These technologies encrypt device partitions so that the data is inaccessible when a system is turned off. When the system boots and you supply the appropriate passphrase, the device is decrypted and its data is accessible. See Oracle Linux 10: Managing Storage Devices for more information about encrypting block devices. Also see the `cryptsetup(8)` manual page for general usage instructions.

- Oracle Linux uses encryption to support Virtual Private Networks (VPNs) and Secure Shell (SSH). You can use these tools to encrypt network traffic end-to-end, thereby ensuring that data is kept safe during transmission. For more information, see Oracle Linux: Connecting to Remote Systems With OpenSSH and Configuring Virtual Private Networks in Oracle Linux: Configuring Virtual Private Networks.
- Oracle Linux uses encryption to store system passwords. By default, Oracle Linux 10 uses `yescrypt` to hash passwords and stores hashed passwords in the `/etc/shadow` file.
- Oracle Linux takes advantage of hardware-accelerated encryption on Intel CPUs that use the Advanced Encryption Standard New Instructions (AES-NI) instruction set, which speeds up the execution of AES and RC4 algorithms on the x86_64 architecture.

About the Packet Filtering Firewall

Firewalls filter incoming and outgoing network packets based on their packet header information. You can create packet filter rules that decide whether packets are accepted or rejected. If you create a rule to block a port, any request to that port is automatically rejected by the firewall and the request is ignored. Any service that's listening on a blocked port no longer processes network traffic because it doesn't receive any new packets from that port.

You can configure the Netfilter feature to act as a packet-filtering firewall that uses rules to decide whether network packets are received, dropped, or forwarded. In addition, Netfilter provides Network Address Translation (NAT) and IP masquerading to alter IP header information for routed packets. You can also set rule-based packet logging and define a dedicated log file by changing `/etc/syslog.conf`.

The `nftables` framework is the default stateful network packet filtering framework in Oracle Linux, replacing the `iptables` framework. The `nftables` framework provides improved performance over the `iptables` framework. The `nftables` framework uses components of the Netfilter infrastructure, such as the existing hooks into the networking stack, connection tracking system, the user-space queueing component, and the logging subsystem. In addition `nftables` can also classify packets.

For more information, see Oracle Linux 10: Configuring the Firewall.

About SELinux

By default, SELinux is enabled automatically on new Oracle Linux installations.

Linux security has historically been based on a Discretionary Access Control (DAC) policy, which provides minimal protection from broken software or from malware that's running as a normal user or as `root`. Access to files and devices is based solely on user identity and ownership. Malware or misconfigured software can do anything with files and resources that the user that started the process can do. If the user is `root` or the application is `setuid` or `setgid` to `root`, the process can have `root`-access control over the entire file system.

The National Security Agency created Security Enhanced Linux (SELinux) to provide a finer-grained level of control over files, processes, users, and applications on Linux. The SELinux enhancement to the Linux kernel implements the Mandatory Access Control (MAC) policy, which can be used to define a security policy that provides granular permissions for all users, programs, processes, files, and devices. The kernel's access control decisions are based on all the security relevant information available, and not solely on the authenticated user identity.

When security-relevant access occurs, such as when a process tries to open a file, SELinux intercepts the operation at the kernel level. The operation only continues if a MAC policy rule allows it, otherwise SELinux blocks the operation and returns an error to the process. The

kernel checks and enforces DAC policy rules before MAC rules, so it doesn't check SELinux policy rules if DAC rules have already denied access to a resource.

For more details about SELinux, including task-related information, see Oracle Linux: Administering SELinux.

See also the SELinux Project Wiki and the `selinux(8)` manual page.

4

Implementing Extra Security Features and Best Practices

This section describes more ways to enhance the security of an Oracle Linux system and further information about best practices for securing an environment.

For more information about securely monitoring, auditing, and logging an Oracle Linux 10 system, see Oracle Linux 10: Auditing the System With Auditd and Rsyslogd and Oracle Linux 10: System Management with systemd.

Configuring and Using Kernel Security Mechanisms

The Linux kernel features some extra security mechanisms that enhance the security of a system. These mechanisms randomize the layout of the address space for a process or prevent code from being run in non-executable memory.

Address Space Layout Randomization

Address Space Layout Randomization (ASLR) can help defeat certain types of buffer overflow attacks. ASLR can find the base, libraries, heap, and stack at random positions in a process's address space, which makes it difficult for an attacking program to predict the memory address of the next instruction. ASLR is built into the Linux kernel and is controlled by the parameter `/proc/sys/kernel/randomize_va_space`. The `randomize_va_space` parameter can take the following values:

0
Disable ASLR. This setting is applied if the kernel is booted with the `norandmaps` boot parameter.

1
Randomize the positions of the stack, virtual dynamic shared object (VDSO) page, and shared memory regions. The base address of the data segment is immediately after the end of the executable code segment.

2
Randomize the positions of the stack, VDSO page, shared memory regions, and the data segment. This is the default setting.

You can change the setting temporarily by writing a new value to `/proc/sys/kernel/randomize_va_space`, for example:

```
echo value | sudo tee /proc/sys/kernel/randomize_va_space
```

To change the value permanently, add this setting to `/etc/sysctl.conf`:

```
kernel.randomize_va_space = value
```

Then, run the `sysctl -p` command.

If you change the value of `randomize_va_space`, it's considered good practice to test the application stack to ensure that it's compatible with the new setting.

You can optionally disable ASLR for a specific program and its child processes:

```
setarch `uname -m` -R program [args ...]
```

Position Independent Executables

The Position Independent Executables (PIE) feature loads executable binaries at random memory addresses so that the kernel can disallow text relocation. Developers can use this feature to code applications that load at different memory addresses each time the application loads, making it more difficult for an attacker to predict where the application is stored in memory, thereby helping to protect against memory-related exploits.

To generate a position-independent binary:

- Specify the `-fpie` option to `gcc` when compiling.
- Specify the `-pie` option to `ld` when linking.

To test whether a binary or library has been built with PIE enabled, run the following command:

```
sudo readelf -d elfname | grep -i flags
```

The command often indicates whether the `PIE` flag is set. By default, on Oracle Linux 10 binaries are typically built with this flag set, unless there's a specific reason not to do so, such as a compile issue resulting from setting this option.

Restricting Access to Kernel Ring Buffer Messages

The kernel uses a ring buffer to surface messages for troubleshooting purposes, and these messages can be viewed by running the `dmesg` command.

Threat actors can use those messages to find exploits in the system, so it's considered good security practice to ensure that only users with root permissions can run the `dmesg` command.

On Oracle Linux 10 systems, access is restricted by default regardless of whether the system is running the Red Hat Compatible Kernel (RHCK) or the Unbreakable Enterprise Kernel (UEK) 8.

Configuring System Cryptograpic Policies

Since Oracle Linux 8, Oracle Linux provides a facility to set a system-wide cryptographic policy. Many applications implement cryptographic protocols to secure communications or to encrypt data. Historically, applications have maintained their own configuration of cryptographic policies in various ways, which meant that changing cryptographic policy across an entire system needed to be performed for each application and often the configuration method differed from application to application.

The ability to define a system-wide cryptographic policy that applications can hook into often reduces administrative overhead and simplifies the process. An administrator can configure the system-wide cryptographic policy and have confidence that most applications can use the same policy, by default.

Policies enable an administrator to configure:

- TLS/SSL (and DTLS) versions that are accepted
- Ciphersuites that are accepted and the preferred order
- Parameters that are accepted for certificates and key exchange, including:
  - the minimum acceptable size of parameters (DH,ECDH,RSA,DSA,ECDSA),
  - the acceptable elliptic curves (ECDH,ECDSA),
  - the acceptable signature hash functions.
- Other TLS options including safe-renegotiation

Most of the major cryptographic software on Oracle Linux is already configured to use the system-wide cryptographic policy by default. Applications that are configured to behave in this manner include important applications such as OpenSSH and bind, in addition to any applications that use the OpenSSL, GnuTLS, NSS, and libkrb5 libraries.

Configuring system-wide policy doesn't enforce behavior across the system. That policy provides a common configuration across a wide variety of applications. Any application that's not designed to use the system-wide policy continues to function according to the different policy configuration that it uses. Many applications also provide options to override the system-wide cryptographic policy if required. For example, OpenSSH provides options to set different cryptographic policies on the server and client applications, and commands such as `wget` and `curl` provide options to define a custom cipher selection and order by using the `--ciphers` option, effectively overriding the system-wide policy.

The system-wide policy defines the default cryptographic behavior within applications so that you can harden a system and remove insecure protocols to match your security requirements.

Oracle Linux includes the `update-crypto-policies` command that can be used to configure which cryptographic algorithms, ciphers, and protocols are enabled on a system for use by applications and services. That command can be used to either relax policy or to harden it further.

For more information on this tool and the applications that are affected by it, see the `crypto-policies(7)` and `update-crypto-policies(8)` manual pages.

About Predefined Policies

Oracle Linux provides four different built-in predefined cryptographic policies:

LEGACY
Configures certain legacy protocols to maximize compatibility with legacy systems. It includes enabling TLSv1.2, and TLSv1.3. It also sets a 2048 bit minimum parameter size for DH and RSA. Protocols and values specified in this policy aren't considered highly secure but aren't easily exploitable.

DEFAULT
Configures standard modern protocols including TLSv1.2 and TLSv1.3, IKEv2 and SSH2. It sets a 2048 bit minimum parameter size for DH and RSA.

FIPS
Configures the system to meet FIPS 140-3 requirements for cryptographic policies. This policy is enabled when Oracle Linux 10 is installed with FIPS mode enabled. See Configuring FIPS Mode in Oracle Linux 10 for more information on using this policy.

FUTURE
A conservative policy level that disables SHA-1 and CBC and sets a 3072 bit minimum parameter size for DH and RSA. This policy can disable communications with many older systems but is worth exploring to decide what actions you can perform in future to ensure that applications continue to function securely.

Restrictions in these policies can change over time as new secure default values are decided.

You can use the `update-crypto-policies` tool to view the current system policy and to change which policy is applied to the system.

Reviewing the Current System-Wide Policy

Any user can review the current system-wide cryptographic policy by running:

```
update-crypto-policies --show
```

Setting the System-Wide Policy

Switching between cryptographic policies on Oracle Linux can be achieved using the `update-crypto-policies --set` command with the name of the policy. For example, to switch to the `LEGACY` policy, run:

```
sudo update-crypto-policies --set LEGACY
```

The policy is updated immediately and any applications that are enabled to use the system-wide cryptographic policy work with the new policy immediately when they're run or restarted. Because some applications might already be running using a custom policy it's good practice to reboot the system after changing policy to ensure that all applications are using the correct policy.

To switch back to the `DEFAULT` policy, run:

```
sudo update-crypto-policies --set DEFAULT
```

Extending a Policy By Using Modules

You can customize the system-wide policy by creating a policy module or a subpolicy. You can fine-tune a policy without needing to create an entire policy from scratch by creating a module. For example, if you intended to use the `DEFAULT` system policy and also disable the weaker SHA-1 hash functionality in all applications, rather than rewriting the entire `DEFAULT` system policy, you can apply a module by setting the `DEFAULT` policy with an appended module for example:

```
sudo update-crypto-policies --set DEFAULT:NO-SHA1
```

Oracle Linux provides some extra modules that have already been configured and can be used immediately in the `/usr/share/crypto-policies/policies/modules/` directory.

You can create custom modules in the `/etc/crypto-policies/policies/modules/` directory. Modules must be named in uppercase and have a lowercase `.pmod` extension. For example,

you can create a module named `/etc/crypto-policies/policies/modules/NO-AES-128.pmod` to add this content to the file to disable the AES-128 cipher entirely:

```
# Disable the AES-128 cipher
cipher = -AES-128-*
```

Note that to disable the cipher, you must prefixed it with a `-` character. To enable a functionality, specify it without a prefix. In the example, the `*` character is also used to specify a wildcard so that the rule matches all modes of the AES-128 cipher.

You can also chain modules together when you set the system-wide cryptograpic policy:

```
sudo update-crypto-policies --set DEFAULT:NO-SHA1:NO-AES-128
```

For more information about the syntax for policy definition files, see the `crypto-policies(7)` manual pages.

Creating a New System-Wide Cryptographic Policy

You can create a custom cryptographic policy from scratch instead of using any of the predefined policies provided with Oracle Linux. Policies can be defined in the `/etc/crypto-policies/policies/` directory. Policy file names must be uppercase and end in the lowercase suffix `.pol`. Policy files use the INI file format with standard `key = value` entries.

The predefined policies provided with Oracle Linux are stored in the `/usr/share/crypto-policies/policies/` directory. To define a custom policy, you can copy an existing policy and then configure it as you need. For example:

```
sudo cp /usr/share/crypto-policies/policies/DEFAULT.pol /etc/crypto-policies/
policies/MYPOLICY.pol
```

See the section titled "CRYPTO POLICY DEFINITON FORMAT" in the `crypto-policies(7)` manual page for more information about the file format and structure.

When you have finished editing the custom policy, you can enable it with this command:

```
sudo update-crypto-policies --set MYPOLICY
```

Remember to reboot the system after enabling a custom system-wide policy so that it's enabled for all running services.

> **Note**
>
> Consider whether you can achieve what you need to do by extending an existing policy using a module. Maintaining a custom system-wide cryptographic policy requires that you consistently monitor new security standards and research, so by extending the predefined policies to meet security requirements you can avoid needing to maintaining an entire policy by yourself.

Checking User Accounts and Privileges

Checking the system for unlocked user accounts often is considered good security practice, for example by using this command:

```
for u in $(awk -F: '{print $1}' /etc/passwd;); do sudo passwd -S "$u"; done |
sort
```

The following output is displayed:

```
adm L 2025-01-29 0 99999 7 -1
bin L 2025-01-29 0 99999 7 -1
chrony L 2025-06-13 -1 -1 -1 -1
clevis L 2025-08-28 -1 -1 -1 -1
...
```

In the output from this command, the second field shows if a user account is locked (`L`), doesn't have a password (`NP`), or has a valid password (`P`). The third field shows the date on which the user last changed their password. The remaining fields show the minimum age, maximum age, warning period, and inactivity period for the password and extra information about the password's status. The unit of time is days.

You can use the `passwd` command to set passwords on any accounts that aren't protected.

To lock unused accounts, use the `passwd -l` command. You can also use the `userdel` command to remove the accounts entirely.

> ⚠ **Caution**
>
> System accounts must be preserved. These are any accounts with user IDs that are less than 1000.

For more information, see the `passwd(1)` and `userdel(8)` manual pages.

To specify how users' passwords are aged, edit the settings in the `/etc/login.defs` file that are described in the following table.

Setting	Description
`PASS_MAX_DAYS`	Maximum number of days for which a password can be used before it must be changed. If no value is specified, the default behavior is to set the value to `-1` and disable the restriction.
`PASS_MIN_DAYS`	Minimum number of days that's allowed between password changes. If no value is specified, the default behavior is to set the value to `0` and disable the restriction.

Setting	Description
PASS_WARN_AGE	Number of days' warning that's provided before a password expires. A value of 0 only warns on the day of expiration, and -1 provides no warning. If no value is specified, the default behavior is to provide no warning.

For more information, see the login.defs(5) manual page.

To change the length of time a user's account can be inactive before it's locked, use the usermod command. For example, you would set the inactivity period to 30 days as follows:

```
sudo usermod -f 30 username
```

To change the default inactivity period for new user accounts, use the useradd command:

```
sudo useradd -D -f 30
```

A value of -1 specifies that user accounts are never locked because of inactivity.

For more information, see the useradd(8) and usermod(8) manual pages.

To verify that no user accounts other than root have a user ID of 0, you would use the following command:

```
sudo awk -F":" '$3 == 0 { print $1 }' /etc/passwd
```

The following is the output of the previous command:

```
root
```

If you install software that creates a default user account and password, it's considered good security practice to change the vendor's default password immediately. Centralized user authentication using an LDAP implementation such as OpenLDAP can centralize user authentication and management tasks, and also reduce the risks arising from unused accounts or accounts without a password.

By default, an Oracle Linux 10 system is configured to prevent users from signing in directly as root. If a root user hasn't been created during the initial system installation, sign in as a named user and then use either the su or sudo command to perform tasks as the root user so that system accounting can trace the original username of any user who performs a privileged administrative action. To grant certain users authority to perform specific administrative tasks by using the sudo command, use the visudo command to configure the /etc/sudoers file.

For example, the following entry grants the user user1 the same privileges as root when using the sudo command, but defines a limited set of privileges to user2 so that they can run commands such as systemctl, rpm, and dnf:

```
user1           ALL=(ALL)       ALL
user2           ALL= SERVICES, SOFTWARE
```

For more information about setting up user accounts and authentication, see Oracle Linux 10: Setting Up System Users and Authentication.

Configuring User Authentication and Password Policies

If you follow traditional digital identity policies, the Pluggable Authentication Modules (PAM) feature can be used to enforce strong user authentication and password policies, including rules that decide password complexity, length, age, expiration, and the reuse of previous passwords. You can configure PAM to block user access after too many failed login attempts, after normal working hours, or if too many concurrent sessions are open. Note that some of these policies are no longer considered helpful for security as they can lead users to implement their own poor security practices when storing passwords or when renewing. See https://pages.nist.gov/800-63-3/sp800-63-3.html for more information.

PAM is highly customizable by its use of different modules with customizable parameters. For example, the default password integrity checking module `pam_pwquality.so` tests password strength. The PAM configuration file (`/etc/pam.d/system-auth`) contains the following default entries for testing a password's strength:

```
password    requisite    pam_pwquality.so local_users_only retry=3
authtok_type= enforce_for_root
password    requisite    pam_pwhistory.so use_authtok enforce_for_root
remember=4
password    sufficient   pam_unix.so sha512 shadow use_authtok
enforce_for_root remember=4
password    sufficient   pam_sss.so use_authtok
password    required     pam_deny.so
```

The line for `pam_pwquality.so` defines that a user gets three tries to choose a good password. From the module's default settings, the password length must a minimum of six characters, of which three characters can't be the same as a previous password. The module only tests the quality of passwords for users who are defined in the `/etc/passwd` file.

The line for `pam_unix.so` specifies that the module tests the old password that was specified in the stack before prompting for a new password and uses the SHA-512 password hashing and the `/etc/shadow` file to decide access. Note that `pam_pwquality` will have performed such checks for users that have been defined in the `/etc/passwd` file.

The line for `pam_pwquality.so` specifies that a user is allowed three tries to select a good password, with a minimum of eight characters, of which five characters must be different from the previous password, and which must contain at least one uppercase letter, one lowercase letter, one numeric digit, and one special character.

The line for `pam_unix.so` specifies that the module doesn't perform password checking, uses SHA-512 password hashing and the `/etc/shadow` file, and saves information about the previous five passwords for each user in the `/etc/security/opasswd` file.

For more information, see the `pam_deny(8)`, `pam_pwquality(8)`, and `pam_unix(8)` manual pages.

The `authselect` command can be used to switch between system authentication profiles. It automatically changes the `/etc/nsswitch.conf` configuration file, and configuration files in the `/etc/pam.d/` and `/etc/dconf/db/distro.d/` directories, as needed. For more information, see the `authselect(8)` and `authselect-migration(7)` manual pages.

Configuring File System Mounts, File Permissions, and File Ownerships

Using separate disk partitions for OS and user data can prevent a "file system full" error from impacting the operation of a server. For example, you can create separate partitions for `/home`, `/tmp`, `/oracle`, and so on.

Establishing disk quotas can prevent a user from filling up a file system (intentionally or not) and therefore denying access to other users.

To prevent the OS files and utilities from being altered during an intrusion, you can mount the `/usr` file system with read-only permissions. If you need to update any RPMs on the file system, use the `-o remount,rw` option with the `mount` command to remount `/usr` for both read and write access. After performing the update, you can use the `-o remount,ro` option to return the `/usr` file system to read-only mode.

To limit user access to non-`root` local file systems such as `/tmp` or removable storage partitions, you can specify the `-o noexec, nosuid, nodev` options to `mount`. These options prevent the execution of binaries (but not scripts), prevent the `setuid` bit from having any effect, and prevent the use of device files.

Unowned files and directories can be associated with a deleted user account, and that might indicate an error with software installation or removal, or they might a sign of an intrusion on the system. You can correct the permissions and ownership of the files and directories that you find, or remove them. Investigating and correcting the problem that led to their creation is considered good security practice.

To check for unowned files and directories on each file system, use the `find` command:

```
sudo find mount_point -mount -type f -nouser -o -nogroup -exec ls -l {} \;
```

Investigating any world-writable directory that's owned by a user other than a system user is considered good security practice. If the user can remove or change any file that other users write to the directory, you can correct the permissions and ownership of any directories that you find or remove them.

To check for world-writable directories on each file system, use the `find` command:

```
sudo find mount_point -mount -type d -perm /o+w -exec ls -l {} \;
```

If the `setuid` and `setgid` bits are set, an executable can perform a task that requires other rights, such as `root` privileges. However, buffer overrun attacks can still exploit those executables to run unauthorized code with the rights of the exploited process.

You can also use the `find` command to check for `setuid` and `setgid` executables.

```
sudo find path -type f \( -perm -4000 -o -perm -2000 \) -exec ls -l {} \;
```

Restricting Access to SSH Connections

The Secure Shell (SSH) provides protected, encrypted communication with other systems. As SSH is an entry point into the system, it's considered good security practice to disable it if it's not required.

You can edit the `/etc/ssh/sshd_config` file to restrict local access to the `root` user and remote access to certain users and groups by configuring the settings. You can also configure settings in the `/etc/ssh/sshd_config` file so that the SSH client automatically times out after of period of inactivity.

Disabling password-based authentication for SSH and to requiring public key authentication instead is considered good security practice. By doing this, you can limit access to users who own an authorized private key.

After making any changes to the configuration file, you must restart the `sshd` service for the changes to take effect.

For more information, see Oracle Linux: Connecting to Remote Systems With OpenSSH and the `sshd_config(5)` manual page.

Using Advanced Intrusion Detection Environment

Advanced Intrusion Detection Environment (AIDE) is an application that uses various tools to detect changes to particular files on a system and report on them so that you can maintain baseline file integrity and detect unauthorized changes and potential tootkits.

This tool is installed as follows:

```
sudo dnf install -y aide
```

When AIDE is installed, you can change the configuration in `/etc/aide.conf`. The configuration file is used to decide which files and directories are monitored by AIDE and also how logging and output are handled.

AIDE stores its current information about a system's configuration state in a database stored in the `/var/lib/aide/aide.db`. If you store a copy of this database file at an external location then you can replace it with a known safe state for AIDE when you perform an audit. If the file doesn't yet exist, you can create one for the current system state by running:

```
sudo aide --init
```

```
sudo cp /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

When you have created a database, you can check file integrity at any time by running:

```
sudo aide --check
```

If no differences are found, AIDE returns the results with the following message:

```
AIDE found NO differences between database and filesystem. Looks okay!!
```

If you configure this tool to run as an automated `cron` job, then you can get regular reports to indicate changes to system configuration and state that could help with early intrusion detection.

See the `aide(1)` and `aide.conf(5)` manual pages for more information.

Protecting the Root Directory by Using chroot Jails

A `chroot` command changes the visible root directory for running processes and their children, so it can be used to run a program with a root directory other than `/`. The program can't see or access files outside of the configured directory tree. Such an artificial root directory is called a "chroot jail", and its purpose is to limit the directory access of malicious processes and hackers. The chroot jail locks down each process and any user ID that's using it so that all they can access is the directory in which the process is running. The process is also tricked into thinking that the directory in which it's running is the root directory.

> **Note**
>
> The `chroot` mechanism can't defend against intentional tampering or low-level access to system devices by privileged users. For example, a `chroot` `root` user could create device nodes and mount file systems on them. A program can also gain access to resources outside of a chroot jail if it can gain `root` privilege and use `chroot()` to change its current working directory to the real `root` directory. For this reason, it's considered good security practice to ensure that a chroot jail doesn't contain any `setuid` or `setgid` executables inside it that are owned by `root`.

For a `chroot` process to start successfully, you must populate the `chroot` directory with all required program files, configuration files, device nodes, and shared libraries at their expected locations relative to the level of the `chroot` directory.

Running DNS and FTP Services in a Chroot Jail

If the DNS name service daemon (`named`) runs in a chroot jail, any hacker that accesses a system by using a BIND exploit is isolated to the files under the chroot jail directory. Installing the `bind-chroot` package creates the `/var/named/chroot` directory, which becomes the chroot jail for all BIND files.

You can configure the `vsftpd` FTP server to automatically start chroot jails for clients. By default, anonymous users are placed in a chroot jail. However, local users that access an `vsftpd` FTP server are placed in their home directory. Specify the `chroot_local_user=YES` option in the `/etc/vsftpd/vsftpd.conf` file to place local users in a chroot jail based on their home directory.

Creating a Chroot Jail

To create a chroot jail:

1. Create the directory that becomes the `root` directory of the chroot jail, for example:

   ```
   sudo mkdir /home/oracle/jail
   ```

2. Use the `ldd` command to decide which libraries are required by the command that you intend to run in the chroot jail, for example `/usr/bin/bash`:

   ```
   sudo ldd /usr/bin/bash
   ```

   The following output is displayed:

   ```
   linux-vdso.so.1 (0x00007fffa5726000)
   libtinfo.so.6 => /lib64/libtinfo.so.6 (0x00007f29127fa000)
   libc.so.6 => /lib64/libc.so.6 (0x00007f29125f1000)
   /lib64/ld-linux-x86-64.so.2 (0x00007f291298c000)
   ```

   > **Note**
   >
   > Although the path is displayed as `/lib64`, the actual path is `/usr/lib64` because `/lib64` is a symbolic link to `/usr/lib64`. Similarly, `/bin` is a symbolic link to `/usr/bin`. You need to re-create such symbolic links within the chroot jail.

3. Create subdirectories of the chroot jail's root directory that have the same relative paths as the command binary and its required libraries in the real root directory, for example:

   ```
   sudo mkdir -p /home/oracle/jail/usr/bin
   ```

   ```
   sudo mkdir -p /home/oracle/jail/usr/lib64
   ```

4. Create the symbolic links that link to the binary and library directories in the same manner as the symbolic links that exists in the real root directory, for example:

   ```
   sudo ln -s /home/oracle/jail/usr/bin /home/oracle/jail/bin
   ```

   ```
   sudo ln -s /home/oracle/jail/usr/lib64 /home/oracle/jail/lib64
   ```

5. Copy the binary and the shared libraries to the directories under the chroot jail's root directory, for example:

   ```
   sudo cp /usr/bin/bash /home/oracle/jail/usr/bin
   ```

   ```
   sudo cp /usr/lib64/{libtinfo.so.5,libdl.so.2,libc.so.6,ld-linux-
   x86-64.so.2} /home/oracle/jail/usr/lib64
   ```

Using a Chroot Jail

To run a command in a chroot jail within an existing directory (*chroot_jail*), use the following command:

```
sudo chroot chroot_jail command
```

If you don't specify a command argument, `chroot` runs with the value of the `SHELL` environment variable, or `/usr/bin/sh` if `SHELL` isn't set.

For example, you could run the `/usr/bin/bash` command in a chroot jail as follows:

```
sudo chroot /home/oracle/jail
```

Note that you can run built-in shell commands such as `pwd` in this shell, but not other commands unless you have copied their binaries and any required shared libraries to the chroot jail.

For more information, see the `chroot(1)` manual page.

5

FIPS 140-3 Compliance in Oracle Linux 10

Oracle Linux provides a set of cryptographic libraries, services, and user-level cryptographic applications that are compliant with the Federal Information Processing Standard (FIPS) Publication 140-3.

FIPS Publication 140-3, Security Requirements for Cryptographic Modules, specifies the security requirements that must be satisfied by a cryptographic module that's used within a security system to protect sensitive, but unclassified information. The NIST/CSE Cryptographic Module Validation Program (CMVP) validates cryptographic modules to FIPS 140-3. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

Configuring FIPS Mode in Oracle Linux 10

FIPS mode can be configured during the initial installation of Oracle Linux 10, as described in the following sections.

Installing Oracle Linux 10 in FIPS Mode

Add `fips=1` to the kernel command line during system installation to automatically configure a new Oracle Linux 10 system to run in FIPS mode from the first boot.

The main benefit of setting FIPS mode during the installation stage is that Oracle Linux 10 enforces the use of strong cryptographic algorithms that are used to secure application data.

To verify that FIPS mode is enabled, run the following command after Oracle Linux 10 has been installed:

```
cat /proc/sys/crypto/fips_enabled
```

If the value returned is `1`, then FIPS mode is enabled on the system.

> **Note**
>
> FIPS mode can't be enabled or disabled on existing Oracle Linux 10 installations. The `fips-mode-setup` tool has been deprecated and removed. To disable FIPS mode, reinstall Oracle Linux 10 without FIPS mode enabled.

FIPS 140-3 Validated Modules in Oracle Linux 10

The following sections describe how to review FIPS certifications and install FIPS 140-3 validated cryptographic modules in Oracle Linux 10.

Information About Modules That Have Received FIPS 140-3 Validation

The Oracle FIPS Certifications website provides the following information for each module:

- Name and description of the module.
- Status of the FIPS 140-3 validation process.

> **Important**
>
> To achieve compliance with FIPS Publication 140-3, you must use the package version that the Security Policy document specifies for each respective module only.

- Package version for the module.
- Certificate number for the module.

> **Note**
>
> Although compliance with FIPS requires that you must use the package version that was validated for each respective cryptographic module, it's considered good security practice to enable FIPS software update channels so that security patches can continue to be applied to those package versions.

6

Security Considerations for Developers

This section provides information for developers about how they can create secure applications for Oracle Linux, and how to extend Oracle Linux to access external systems without compromising security.

Design Principles for Secure Coding

Follow these design principles to enhance security at the source code level:

Least privilege
A process or user is granted only those privileges that are necessary to complete a task. User privileges are assigned according to their role, and all others are denied. To create a minimal protection domain, assign permissions when a process or thread requires them and removed after. This principle limits the potential damage that can result from attacks and user errors.

Economy of mechanism
Keep the design straightforward so that fewer things can go wrong, fewer inconsistencies can emerge, and the code is easier to understand and debug.

Complete mediation
Check every try to access to a resource, rather than only the first. For example, Linux checks access permissions when a process opens a file but not after. If a file's permissions change while a process has the file open, this can result in unauthorized access. Permissions could be checked whenever an open file is accessed but in practice, such checking is considered to be an unnecessary overhead because of the circumstances under which access was first obtained.

Open design
Share the code's design or implementation. An open back door to a system is only as secure as the knowledge of its existence, so empowering others to help you find those back doors and close them before they're discovered is considered good security practice. This principle doesn't apply to information such as passwords or cryptographic keys, knowledge of which is best shared among as few people as possible. For that reason, many secure authentication schemes also rely on biometric identification or the possession of a physical artifact such a hardware token or smart card, in addition to knowledge of a PIN code or password.

Separation of privilege
Divide the code into modules, where each module requires a specific, limited set of privileges to perform a specific task. You can require several privileges to grant access to a sensitive operation. This principle ensures separation of duty and provides defense in depth. For example, a main thread that has no privileges can generate a privileged thread to perform a task. A successful attack against the main thread would only gain minimal access to the system.

Least common mechanism
Isolate users and their activities from each other. Users sharing processes or threads and information channels wouldn't be considered good security practice.

Fail-safe defaults
Deny access to an operation by default. If a user without the necessary privileges tries to perform an operation it's denied and the system is still as secure as it was before the operation started.

Accountability
Log the user and their privileges for each action that they try to perform. You can rotate and archive logs to avoid filling up a file system.

Psychological acceptability
Select security mechanisms that are straightforward to install, configure, and use so that a user isn't tempted to try to bypass them.

General Guidelines for Secure Coding

Follow these coding practices to enhance security at the source code level:

- Check that input data is what the program expects by performing type, length, and bound checking. Inputs include command line arguments and environment variables in addition to data that a user enters.
- Check input data for the inclusion of constructs such as shell commands, SQL statements, and any XML or HTML code that might be used in an injection attack.
- Check the type, length, and bounds of arguments to system calls and library routines. If possible, use library routines that guard against buffer overflows.
- Use full pathnames for file-name arguments, don't use files in world-writable directories, verify that a file being written to isn't a symbolic link, and protect against the unintended overwriting of existing files.
- Check the type, length, and bounds of values returned by system calls and library routines. Make the code respond appropriately to any error codes that system calls and library functions set or return.
- Don't assume the state of the shell environment. Check any settings that a program inherits from the shell, such as the user file-creation mask, signal handling, file descriptors, current working directory, and environment variables, especially `PATH` and `IFS` . Reset the settings if needed.
- Perform assert checking on variables that can take a finite set of values.
- Log any information about privileged actions and error conditions. Don't let the program dump a core file on an end-user system.
- Don't echo passwords to the screen or send or store them as clear text. Before sending or storing a password, combine it with a salt value and use a secure one-way algorithm such as SHA-512 to create a hash.
- If the program uses a pseudo random number generating routine, verify that the numbers that it generates are sufficiently random to match security requirements. Also use a good random seed that a potential attacker can't be expected to predict. See RFC 4086, Randomness Requirements for Security, for more information.
- Enable Address Space Layout Randomization (ASLR) on the host system as this feature can help defeat certain types of buffer overflow attacks. See Address Space Layout Randomization.
- When compiling and linking a program, use the Position Independent Executables (PIE) feature to generate a position-independent binary. See Position Independent Executables.

- Consider using `chroot()` to confine the operating boundary of a program to a specified location within a file system.
- Don't run a shell command by calling `popen()` or `syscall()` from within a program, especially from a `setuid` or `setgid` program.

The following guidelines apply if a program has its `setuid` or `setgid` bit set so that it can perform privileged actions on behalf of a user who haven't been granted those privileges:

- Don't set the `setuid` or `setgid` bit on shell scripts. However, if you use Perl scripts that are `setuid` or `setgid`, you can run `perl` in "taint mode," which can be more secure than using similar C code. See the `perlsec(1)` manual page for details.
- Restrict the use of the privilege that `setuid` or `setgid` grants to the functionality that requires it, and then return the effective UID or GID to that of the user. If possible, perform the privileged functionality in its own monitored thread or process.
- Don't run a `setuid` or `setgid` program inside a child processes using `execlp()` or `execvp()`, which use the `PATH` environment variable.

General Guidelines for Network Programs

Follow these guidelines to enhance the security of network programs:

- Perform a reverse lookup on an IP address to obtain the fully qualified domain name, and then use that domain name look up the IP address. Ensure that both IP addresses are identical.
- Protect a service against Denial of Service (DoS) attacks by pausing the processing of requests if it becomes overloaded.
- Set timeouts on read and write requests over the network.
- Check the content, bounds, value, and type of data received over the network, and reject any data that doesn't conform to what the program expects.
- Use certificates or preshared keys to authenticate the local and remote ends of the network connection.
- Use encryption technology such as TLS or SSL to secure data sent over the network connection.
- Wherever possible, use existing networking protocols and technologies whose security characteristics are widely understood.
- Log any information about successful and unsuccessful connection attempts, data reception, and transmission errors, and changes to the service state.