Oracle® Linux 8

Enhancing System Security

F22907-03
January 2020

Abstract

Oracle® Linux 8: Enhancing System Security describes features in Oracle Linux 8 that you can use to enhance the security of your systems. The guide also includes guidelines and recommendations for best security practices when working with Oracle Linux.

Document generated on: 2020-01-21 (revision: 9094)

Table of Contents

Preface

Oracle® Linux 8: Enhancing System Security describes features in Oracle Linux 8 that can enhance the security of your systems. The guide also includes guidelines and recommendations for best security practices when working with Oracle Linux.

Audience

This document is intended for administrators who need to configure and administer Oracle Linux networking. It is assumed that readers are familiar with web technologies and have a general understanding of using the Linux operating system, including knowledge of how to use a text editor such as `emacs` or `vim`, essential commands such as `cd`, `chmod`, `chown`, `ls`, `mkdir`, `mv`, `ps`, `pwd`, and `rm`, and using the `man` command to view manual pages.

Document Organization

The document is organized into the following chapters:

- Chapter 1, *Guidelines and Best Practices for Enhancing System Security* provides system security guidelines and best practices.
- Chapter 2, *Planning for a Secure Oracle Linux Environment* provides planning information and guidelines for securing an Oracle Linux environment.
- Chapter 3, *Features and Services for Enhancing System Security* describes Oracle Linux features that are used to enhance security on an Oracle Linux system.
- Chapter 4, *Implementing Additional Security Features and Best Practices* describes features and tasks that you can implement to further enhance security on an Oracle Linux system.
- Chapter 5, *Security Considerations for Developers* provides information for developers about creating secure applications for Oracle Linux.

Related Documents

The documentation for this product is available at:

https://docs.oracle.com/en/operating-systems/linux.html.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit
https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Chapter 1 Guidelines and Best Practices for Enhancing System Security

This chapter provides a brief overview of system security and includes some guidelines and best practices for how you can enhance security on your Oracle Linux systems.

Oracle Linux has evolved into a secure enterprise-class operating system that can provide the performance, data integrity, and application uptime necessary for business-critical production environments.

Thousands of production systems at Oracle run Oracle Linux and numerous internal developers use it as their development platform. Oracle Linux is also at the heart of several Oracle engineered systems, including the Oracle Exadata Database Machine, Oracle Exalytics In-Memory Machine, Oracle Exalogic Elastic Cloud,Oracle Private Cloud Appliance, Oracle Cloud Infrastructure and Oracle Database Appliance.

Oracle On Demand services, which deliver software as a service (SaaS) at a customer's site, via an Oracle data center, or at a partner site, use Oracle Linux at the foundation of their solution architectures. Backed by Oracle support, these mission-critical systems and deployments depend fundamentally on the built-in security and reliability features of the Oracle Linux operating system.

Oracle has been a key participant in the Linux community, contributing code enhancements for kernel development but also on many open-source initiatives, such as Oracle Cluster File System and the Btrfs file system. From a security perspective, having roots in open-source is a significant advantage. The Linux community, which includes many experienced developers and security experts, reviews posted Linux code extensively prior to its testing and release. 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 these tools to provide a solid and secure operating system.

Oracle recommends that you follow some fundamental security principles when using Oracle Linux. These principles are guidelines that should inform your approach to handling security policies.

Minimize and Secure the Software Footprint

Planning a system's purpose, deployment configuration, and software requirements in advance is essential to minimizing attack vectors. During the design phase of your deployment, you should identify any components and services and are not required and ensure that those components and services, including any peripheral functionality or components, are not installed or enabled. Because deployment requirements can vary, you must also have the ability to uninstall or disable any features that are not needed or used in that particular configuration or deployment scenario. In some cases, you might consider using a small core installation method; which by default, only installs the essential components of the operating system.

Another important way in which you can effectively ensure that your Oracle Linux systems are secure is to only install those components that are essential for performing necessary functions. Additional functions and components can increase the security risk and should be removed or uninstalled, if necessary.

Always install software from secure, known and trusted sources. Oracle provides GPG key signing to help you confirm that software is from Oracle and uses TLS to secure communications with software installation and update tools.

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 regularly using the Oracle Linux yum server and the Unbreakable Linux Network (ULN).

Regularly updating the installed software so that it is patched for any vulnerabilities is key in minimizing the attack surface for your systems. Pinning software to a particular release or version or fixing a system to a particular update level of Oracle Linux can result in software becoming increasingly vulnerable and bug prone. Furthermore, it becomes more logistically difficult to update software as the gap between software releases widens, which can make it harder to resolve security vulnerabilities in future.

Consider using Oracle Ksplice along with regular system updates to automatically patch the running kernel and many user-space applications without any requirement for system downtime. For more information about Ksplice, see *Oracle® Linux: Ksplice User's Guide*.

Restrict Network Access to Critical Services

Keep both middle-tier applications and databases behind a firewall. In addition, place a firewall between middle-tier applications and databases if these are hosted on separate servers. The firewalls provide assurance that access to these systems is restricted to a known network route, which can be monitored and restricted, if necessary. As an alternative, a firewall router substitutes for multiple, independent firewalls.

If firewalls cannot be used, restrict access based upon IP address. Restricting database access by IP address often causes application client/server programs to fail for DHCP clients. To resolve this, consider using static IP addresses, a software/hardware VPN or Windows Terminal Services or its equivalent.

See Section 3.4, “About the Packet-Filtering Firewall”, Section 3.5, “About TCP Wrappers” and Section 4.9, “Restricting Access to SSH Connections” for more information on how to restrict and secure network access.

Control Authentication Mechanisms and Enforce Password Restrictions

Different authentication mechanisms can be employed to control access to a system. In environments where many systems are involved, consider using a centralized authentication tool so that you are not maintaining accounts across many systems. Also consider the different types of authentication mechanisms at your disposal. While password-level access is often convenient, providing better and more restrictive mechanisms such as key, certificate or token based authentication that often result in 2-factor authentication, can help secure an environment further.

Where using password-style access, ensure that you enforce restrictions to prevent common, short or easily cracked passwords. You may consider the recent NIST 800-63 Digital Identity Guidelines which suggest a deviation from traditional password policy. Rather than forcing complicated passwords with frequent expiry and lockout mechanisms, use long and memorable passwords that are not easy to guess or crack. Most importantly, passwords should be checked against known password dictionaries.

See Section 4.7, “Configuring User Authentication and Password Policies” for more information.

Follow the Principle of Least Privilege

The principle of least privilege states that users should be given the least amount of privilege to perform their jobs. Over ambitious granting of responsibilities, roles, grants, and so on, especially early on in an organization’s life cycle when people are few and work needs to be done quickly, often leaves a system wide open for abuse. User privileges should be reviewed periodically to determine relevance to current job responsibilities.

Only create a single account per user. If a user requires administrator access for a purpose, use `sudo` to grant access for the specific purpose. Avoid distributing the root user password and ensure that this password is lengthy, random and uses a significantly wide number of non-standard characters.

See Section 4.6, “Checking User Accounts and Privileges” for more information.

Monitor System Activity

System security stands on three legs: good security protocols, proper system configuration, and system monitoring. Auditing and reviewing audit records address the third requirement. Each component within a system has some degree of monitoring capability. Follow audit advice in this document and regularly monitor audit records.

See Section 4.10, “Using System Auditing and Monitoring”, Section 4.11, “Using Advanced Intrusion Detection Environment” and Section 4.12, “Implementing System Process Accounting” for more information.

Keep Up to Date on the Latest Security Information

Oracle continually improves its software and documentation. Check regularly on the Oracle Technology Network at https://www.oracle.com/technetwork/server-storage/linux for revisions. For information about common vulnerabilities and exposures (CVE) and errata that are available on the Unbreakable Linux Network, see https://linux.oracle.com/cve and https://linux.oracle.com/errata.

Chapter 2 Planning for a Secure Oracle Linux Environment

Table of Contents

This chapter provides information and guidelines for planning to secure an Oracle Linux environment, based on individual security needs.

To better understand your security needs, ask yourself the following questions:

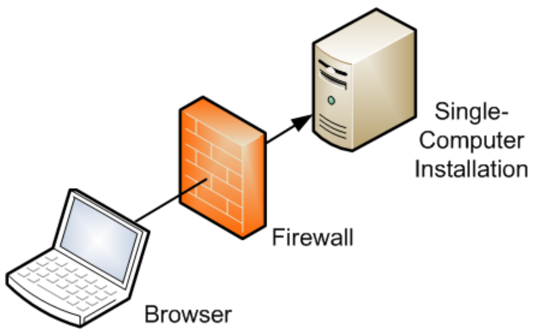
Which resources am I protecting?	Many resources in the production environment can be protected, including information in databases accessed by WebLogic Server and the availability, performance, applications, and the integrity of the website. Consider the resources you want to protect when deciding the level of security you must provide.
From whom am I protecting resources?	For most websites, resources must be protected from everyone on the Internet. But should the website be protected from the employees on the intranet in your enterprise? Should your employees have access to all resources within the WebLogic Server environment? Should the system administrators have access to all WebLogic resources? Should the system administrators be able to access all data? You might consider giving access to highly confidential data or strategic resources to only a few well trusted system administrators. Perhaps it would be best to allow no system administrators access to the data or resources.
What will happen if the protections on strategic resources fail?	In some cases, a fault in your security scheme is easily detected and considered nothing more than an inconvenience. In other cases, a fault might cause great damage to companies or individual clients that use the website. Understanding the security ramifications of each resource will help you protect it properly.

2.1 Recommended Deployment Configurations

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

Figure 2.1, "Simple Firewall Deployment Configuration" shows a simple deployment architecture.

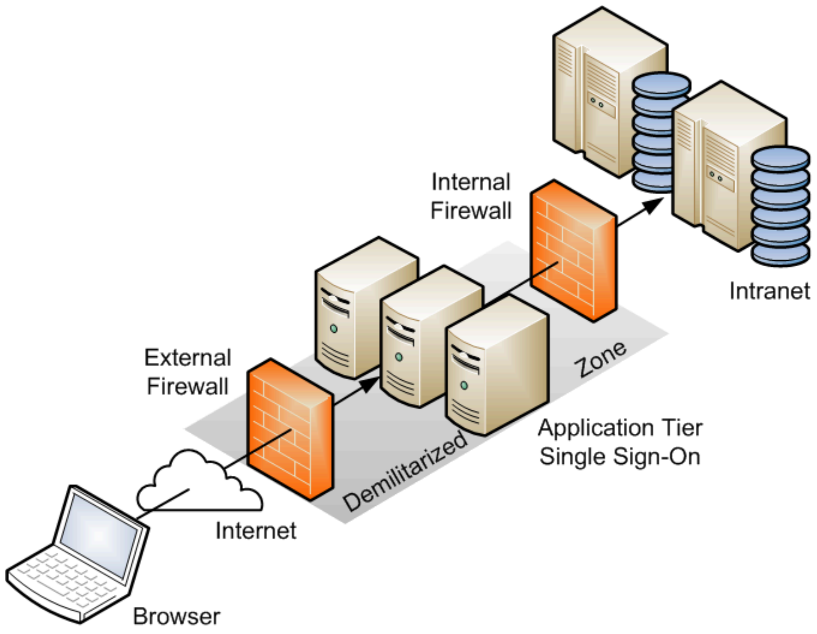
Figure 2.1 Simple Firewall Deployment Configuration



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

Figure 2.2, “DMZ Deployment Configuration” shows the recommended configuration, which uses the well-known and generally accepted Internet-Firewall-DMZ-Firewall-Intranet architecture.

Figure 2.2 DMZ Deployment Configuration



A *demilitarized zone* (DMZ) refers to a server that is 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:

- Block any traffic types that are not permitted.
- Provide intrusion containment in the event that successful intrusions take over processes or processors.

2.2 Component Security

Each application software component usually has its own security considerations that you should take into account independently of those that apply to the operating system. Refer to the security guidelines for each component to determine how best to configure it for the requirements of security at your site.

Chapter 3 Features and Services for Enhancing System Security

Table of Contents

This chapter describes Oracle Linux features that you can use to enhance system security.

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

You should consider using a combination of these tools and facilities to improve the security of the operating system, the software and applications that are running on the system and the network connectivity and access that is provisioned for the system.

3.1 About System Software Updates

Keeping system software regularly updated is fundamental to avoiding and protecting against software vulnerabilities and known attack vectors. Oracle Linux 8 provides tools to help you to do this regularly and with minimum interference.

Consider using the `dnf-automatic` package to regularly download updates, alert you to software upgrade options and even to apply them automatically. See *Oracle® Linux 8: Managing Software on Oracle Linux* for more information on how to install and enable this tool.

Oracle Ksplice can also be used to apply patches in real-time to any supported kernel or user space software, so that you can be certain that the latest vulnerabilities are automatically patched without you even having to restart a system. See *Oracle® Linux: Ksplice User's Guide* for more information on how to install and configure the Oracle Ksplice software.

3.2 About Certificate Management

Public-key cryptography enables 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 your organization's intranet, you can use OpenSSL to generate a root certificate and set up a 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 your application configuration. Low-cost domain validation certificate signing is now more obtainable using the IETF standardized Automatic Certificate Management Environment (ACME) protocol as described in RFC 8555, reducing the requirement for costly expenditure around certificate signing and reducing the requirement to run your own CA.

The `openssl` command enables you to generate self-signed certificates that web browsers can use. You can also use the `keytool` command to generate self-signed certificates, but this command's primary purpose is to install and manage JSSE (Java Secure Socket Extension) digital certificates for use with Java applications.

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

3.3 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 is stored or that is 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 is 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 of corporate security policy 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. 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. You can also install RPMs using the Secure Sockets Layer (SSL) protocol, which uses encryption to protect the communications channel.
- 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 supports block device encryption 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. For more information, see the `cryptsetup(8)` manual page.
- Oracle Linux uses encryption to support Virtual Private Networks (VPN) and Secure Shell (`ssh`). These tools can be used to encrypt network traffic from end to end, ensuring that data can be kept safe during transmission. For more information, see *Oracle® Linux: Connecting to Remote Systems With OpenSSH* and *Oracle® Linux 8: Setting Up Networking*.
- Oracle Linux uses encryption to store system passwords. By default, Oracle Linux uses a strong password hashing algorithm (SHA-512) and stores hashed passwords in the `/etc/shadow` file.
- Oracle Linux takes advantage of hardware-accelerated encryption on Intel CPUs that support the Advanced Encryption Standard New Instructions (AES-NI) instruction set, which speeds up the execution of AES algorithms as well as SHA-1 and RC4 algorithms on x86 and x86_64 architectures.

3.4 About the Packet-Filtering Firewall

A firewall filters incoming and outgoing network packets, based on packet header information. You create packet filter rules that determine whether packets are accepted or rejected. If you create a rule to block a port, any request to that port is rejected by the firewall and the request is ignored. Any service that is listening on a blocked port is effectively disabled.

You can configure the Netfilter feature to act as a packet-filtering firewall that uses rules to determine whether network packets are received, dropped, or forwarded. In addition, Netfilter provides Network Address Translation (NAT) to hide IP addresses behind a public IP address, 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 in `/etc/syslog.conf`.

The `nftables` framework is the default network packet filtering framework in Oracle Linux, replacing the `iptables` framework. The `nftables` framework provides improved performance over the `iptables` framework. The `nftables` framework utilizes the building blocks 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.

For more information, see *Oracle® Linux 8: Configuring the Firewall*.

3.5 About TCP Wrappers

The TCP wrappers feature mediates requests from clients to services, and control access based on rules that you define in the `/etc/hosts.deny` and `/etc/hosts.allow` files. You can restrict and permit service access for specific hosts or whole networks. A common way of using TCP wrappers is to detect intrusion attempts. For example, if a known malicious host or network attempts to access a service, you can deny access and send a warning message about the event to a log file or to the system console.

For more information, see *Oracle® Linux 8: Configuring the Firewall*.

3.6 About SELinux

By default, SELinux is enabled when you install an Oracle Linux system.

Traditional Linux security is based on a Discretionary Access Control (DAC) policy, which provides minimal protection from broken software or from malware that is running as a normal user or as `root`. Access to files and devices is based solely on user identity and ownership. Malware or broken 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 in the Linux operating system. The SELinux enhancement to the Linux kernel implements the Mandatory Access Control (MAC) policy, which allows you 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 attempts to open a file, SELinux intercepts the operation in the kernel. If a MAC policy rule allows the operation, it continues; 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 does not 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.

Chapter 4 Implementing Additional Security Features and Best Practices

Table of Contents

This chapter describes additional ways to enhance the security of an Oracle Linux system, as well as information about best practices for securing your environment.

4.1 Disabling Core Dumps

A core dump is a file that is produced when a process terminates unexpectedly. The core dump contains an image of the process's memory. You can use a core dump to debug problems and inspect the stage of the program when it terminated. You can produce core dumps on-demand or they can be created automatically on termination.

The core file, also referred to a *core*, is created in the current working directory. Note that the writing of core fails if the directory to which it is to be created is non-writable, or if a file exists with the same name in that location, and that file is non-writable or is not a regular file.

Core dumps can contain information that an attacker might be able to exploit. Core dumps can also take up a large amount of disk space. To prevent the system from creating core dumps when it terminates a program due to a segment violation or other unexpected error, add the following line to `/etc/security/limits.conf`:

```
*  hard  core  0
```

You can restrict access to core dumps to certain users or groups. See the `limits.conf(5)` manual page for details.

By default, the system prevents the `setuid` and `setgid` programs, programs that have changed credentials, and programs containing binaries that do not have read permission from dumping core.

To ensure that the setting is permanently recorded, add the following lines to the `/etc/sysctl.conf` file:

```
# Disallow core dumping by setuid and setgid programs
fs.suid_dumpable = 0
```

Run the `sysctl -p` command after adding this information to the file.

> **Note**
>
> A value of 1 permits core dumps that are readable by the owner of the dumping process. A value of 2 permits core dumps that are readable only by `root` for debugging purposes.

4.2 Disable or Restrict the Automatic Bug Reporting Tool

If you are running the Automatic Bug Reporting Tool, abrt, core dumps may continue to be generated for a system, even if you have disabled the core dump facility. On a production system, you may wish to disable or uninstall `abrt` entirely.

To stop and disable the `abrtd` service entirely, run:

```
# systemctl disable --now abrtd
```

If you would prefer to run the service, make sure that you restrict the service so that it only analyzes crashes for binaries installed using signed packages. You might also want to prevent the service from analyzing particular binaries that may reveal sensitive information in a dump, by adding these to a blacklist.

For example, edit `/etc/abrt/abrt-action-save-package-data.conf` to set the following parameters:

```
# Require a GPG signature for a package
OpenGPGCheck = yes

# Add any package names to the Blacklist that contain binaries
# that you want abrt to not store dump data for
BlackList = nspluginwrapper, valgrind, strace, mono-core, bash

# Disable processing of unpackaged binaries
ProcessUnpackaged = no

# Add any paths to the BlackListedPaths that may contain binary
# executables that you want abrt to not store dump data for
BlackListedPaths = /usr/share/doc/*, */example*, /usr/bin/nspluginviewer, \
     /usr/lib*/firefox/plugin-container
```

Note that although the `BlackList` and `BlackListedPaths` options can prevent the service from storing dump data, the dumps are still generated and written to disk briefly before being removed. This action allows `abrtd` to notify system administrators of a crash without using up disk space.

To actually prevent the core dumps from being written to disk and to prevent `abrtd` from detecting crashes in an application, edit `/etc/abrt/plugins/CCpp.conf` and add the absolute path of the binary to be ignored to the `IgnoredPaths` list. For example:

```
IgnoredPaths = /path/to/binary
```

4.3 Configuring a System in FIPS Mode

> **Note**
>
> Currently, there are no FIPS validated cryptographic modules available for Oracle Linux 8. However, you can configure FIPS mode on an Oracle Linux 8 system,

which may be useful because FIPS mode imposes certain limitations when using cryptographic modules.

The following procedures describe how to enable and disable FIPS mode on an Oracle Linux 8 system. Note that the method for enabling and disabling FIPS mode in this release has changed significantly from the method that was used in previous Oracle Linux releases. In particular, the `dracut-fips` package no longer exists, so you do not need to install it to enable FIPS mode on Oracle Linux 8. Also, you no longer need to edit the GRUB configuration file. Instead, you use the `fips-mode-setup` utility to set up and configure FIPS mode, as described in the following procedure.

4.3.1 Enabling FIPS Mode

1. To enable FIPS mode on the system, run the following command:

```
# fips-mode-setup --enable
Setting system policy to FIPS
FIPS mode will be enabled.
Please reboot the system for the setting to take effect.
# reboot
```

   You must reboot the system for the setting to take effect.

   Running the previous command configures FIPS mode implicitly by setting the system-wide cryptographic policy to FIPS. Note that using the `update-crypto-policies` command to set FIPS mode is not sufficient, as shown in the following output:

```
# update-crypto-policies --set FIPS
Warning: Using 'update-crypto-policies --set FIPS' is not sufficient for FIPS compliance.
Use 'fips-mode-setup --enable' command instead.
```

2. Verify that FIPS is enabled by running any of the following commands:

```
# fips-mode-setup --check
FIPS mode is enabled.
```

```
# update-crypto-policies --show
FIPS
```

```
# cat /etc/system-fips
FIPS module installation complete
```

```
# sysctl crypto.fips_enabled
crypto.fips_enabled = 1
```

   For the command output in the last example, a response of `1` indicates that FIPS is enabled.

4.3.2 Disabling FIPS Mode

If you need to disable FIPS mode for any reason, run the following command:

```
# fips-mode-setup --disable
Setting system policy to DEFAULT
Note: System-wide crypto policies are applied on application start-up.
It is recommended to restart the system for the change of policies
to fully take place.
FIPS mode will be disabled.
Please reboot the system for the setting to take effect.
# reboot
```

You must reboot the system for the setting to take effect.

4.4 Configuring and Using Kernel Security Mechanisms

The Linux kernel features some additional 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 executed in non-executable memory.

4.4.1 Address Space Layout Randomization

Address Space Layout Randomization (ASLR) can help defeat certain types of buffer overflow attacks. ASLR can locate 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 located 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 > /proc/sys/kernel/randomize_va_space
```

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

```
kernel.randomize_va_space = value
```

Then, run the `sysctl -p` command.

If you change the value of `randomize_va_space`, you should test your application stack to ensure that it is compatible with the new setting.

If necessary, you can disable ASLR for a specific program and its child processes by using the following command:

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

4.4.2 Data Execution Prevention or No eXecute

The Data Execution Prevention (DEP) feature, also known as No eXecute (NX), prevents an application or service from executing code in a non-executable memory region. Hardware-enforced DEP works in conjunction with the NX bit on compatible CPUs to help prevent certain types of buffer overflow attacks. This feature is enabled by default and cannot be disabled. It helps to take advantage hardware capabilities to protect your system.

Oracle Linux does not emulate the NX bit in software for CPUs that do not implement the NX bit in hardware.

4.4.3 Position Independent Executables

The Position Independent Executables (PIE) feature loads executable binaries at random memory addresses so that the kernel can disallow text relocation. This feature allows developers 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 located 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:

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

The command should indicate that the `PIE` flag is set. By default, on Oracle Linux 8 binaries are typically built with this flag set, unless there is a specific reason not to do so, such as a compile issue resulting specifically from setting this option.

4.5 Configuring System Cryptograpic Policies

Oracle Linux 8 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. The command can be used to either relax policy or to harden it further.

The `DEFAULT` policy that is enforced for an initial installation, helps to protect a system by setting sane rules for all cryptographic activity. For example, the TLS 1.0 and 1.1 protocols are disabled along with the DSA, 3DES, and RC4 algorithms. Furthermore, RSA keys and Diffie-Hellman parameters are only accepted if they are at least 2048 bits long.

The system can be further hardened to use a very conservative `FUTURE` policy that prevents SHA-1 in signature algorithms and also requires a 3072 bit key length for RSA and Diffie-Hellman.

A `FIPS` policy level is enforced when FIPS mode is enabled as per Section 4.3, “Configuring a System in FIPS Mode”. This sets the system up to only support the policy requirements specified for FIPS140-2 conformance.

You can check which policy is being enforced by running:

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

You can set the system-wide policy by running the command as follows:

```
# update-crypto-policies --set FUTURE
```

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.

4.6 Checking User Accounts and Privileges

Check the system for unlocked user accounts on a regular basis, for example using a command such as the following:

```
# for u in $(cat /etc/passwd | cut -d: -f1 | sort); do passwd -S "$u"; done
```

```
abrt LK 2012-06-28 0 99999 7 -1 (Password locked.)
adm LK 2011-10-13 0 99999 7 -1 (Alternate authentication scheme in use.)
apache LK 2012-06-28 0 99999 7 -1 (Password locked.)
avahi LK 2012-06-28 0 99999 7 -1 (Password locked.)
avahi-autoipd LK 2012-06-28 0 99999 7 -1 (Password locked.)
bin LK 2011-10-13 0 99999 7 -1 (Alternate authentication scheme in use.)
...
```

In the output from this command, the second field shows if a user account is locked (`LK`), does not have a password (`NP`), or has a valid password (`PS`). 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 additional information about the password's status. The unit of time is days.

Use the `passwd` command to set passwords on any accounts that are not protected.

Use `passwd -l` to lock unused accounts. Alternatively, use `userdel` to remove the accounts entirely.

Caution

System accounts must be preserved. These are any accounts with user IDs that are less than 1000, and especially any user IDs that are less than 100.

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. The default value is 99,999 days.
`PASS_MIN_DAYS`	Minimum number of days that is allowed between password changes. The default value is 0 days.
`PASS_WARN_AGE`	Number of days warning that is given before a password expires. The default value is 7 days.

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 is locked, use the `usermod` command. For example, you would set the inactivity period to 30 days as follows:

```
# usermod -f 30 username
```

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

```
# useradd -D -f 30
```

A value of `-1` specifies that user accounts are not locked due to 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:

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

If you install software that creates a default user account and password, you should change the vendor's default password immediately. Centralized user authentication using an LDAP implementation such as

OpenLDAP can help to simplify user authentication and management tasks, and also reduces the risk arising from unused accounts or accounts without a password.

By default, an Oracle Linux 8 system is configured so that you cannot log in directly as `root`. You must log in as a named user before using either the `su` or `sudo` command to perform tasks as the `root` user. This configuration enables system accounting to trace the original login name of any user who performs a privileged administrative action. If you want to grant certain users authority to be able to perform specific administrative tasks by using the `sudo` command, use the `visudo` command to modify 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 he 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 8: Setting Up System Users and Authentication*.

4.7 Configuring User Authentication and Password Policies

If you follow traditional digital identity policies, the Pluggable Authentication Modules (PAM) feature enables you to enforce strong user authentication and password policies, including rules for 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 opened. It is worth noting that some of these policies are no longer considered particularly helpful for security as they often lead users to implement poor security practice 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 try_first_pass local_users_only retry=3 authtok_type=
password  sufficient  pam_unix.so sha512 shadow nullok try_first_pass use_authtok
password  required    pam_deny.so
```

The line for `pam_pwquality.so` defines that a user gets three attempts to choose a good password. From the module's default settings, the password length must a minimum of six characters, of which three characters cannot 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 password that was previously specified in the stack before prompting for a password; and, if necessary uses the SHA-512 password hashing and the `/etc/shadow` file to allow access. if the existing password is null. Note that `pam_pwquality` will already have performed such checks for users that are defined in the `/etc/passwd` file .

You can modify the control flags and module parameters to change the checks that are performed when a user changes his or her password, for example:

```
password  required  pam_pwquality.so retry=3 minlen=8 difok=5 minclass=-1
password  required  pam_unix.so use_authtok sha512 shadow remember=5
password  required  pam_deny.so
```

The line for `pam_pwquality.so` specifies that a user is allowed three attempts to choose 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 upper-case letter, one lower-case letter, one numeric digit, and one non-alphanumeric character.

The line for `pam_unix.so` specifies that the module does not 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. As `nullok` is not specified, a user cannot change his or her password if the existing password is null.

The omission of the `try_first_pass` keyword means the user is always asked for their existing password, even if the user has entered it for the same module or for a previous module in the stack.

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

4.8 Configuring File System Mounts, File Permissions, and File Ownerships

Use separate disk partitions for operating system and user data to prevent a *file system full* issue from impacting the operation of a server. For example, you might create separate partitions for `/home`, `/tmp`, `/oracle`, and so on.

Establish disk quotas to prevent a user from accidentally or intentionally filling up a file system and denying access to other users.

To prevent the operating system files and utilities from being altered during an attack, mount the `/usr` file system read-only. 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, 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, specify the `-o noexec, nosuid, nodev` options to `mount`. These option prevent the execution of binaries (but not scripts), prevent the `setuid` bit from having any effect, and prevent the use of device files.

Use the `find` command to check for unowned files and directories on each file system, for example:

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

Unowned files and directories might be associated with a deleted user account, they might indicate an error with software installation or deleting, or they might a sign of an intrusion on the system. Correct the permissions and ownership of the files and directories that you find, or remove them. If possible, investigate and correct the problem that led to their creation.

Use the `find` command to check for world-writable directories on each file system, for example:

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

Investigate any world-writable directory that is owned by a user other than a system user. The user can remove or change any file that other users write to the directory. Correct the permissions and ownership of the directories that you find, or remove them.

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

```
# find path -type f \( -perm -4000 -o -perm -2000 \) -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 exploit such executables to run unauthorized code with the rights of the exploited process.

4.9 Restricting Access to SSH Connections

The Secure Shell (SSH) allows protected, encrypted communication with other systems. As SSH is an entry point into the system, disable it if it is not required, or alternatively, edit the `/etc/ssh/sshd_config` file to restrict its use.

You can restrict access to the `root` user, as well as remote access to certain users and groups by specifying settings in this file. 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.

It is common practice to also disable password-based authentication for SSH and to require public key authentication instead. By doing this, you can limit access to users who possess a private key. Often this can help to further limit SSH access on systems.

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.

4.10 Using System Auditing and Monitoring

Auditing collects data at the kernel level that you can analyze to identify unauthorized activity. Auditing collects more data in greater detail than system logging, but most audited events are uninteresting and insignificant. The process of examining audit trails to locate events of interest can be a significant challenge that you will probably need to automate.

The audit configuration file, `/etc/audit/auditd.conf`, defines the data retention policy, the maximum size of the audit volume, the action to take if the capacity of the audit volume is exceeded, and the locations of local and remote audit trail volumes. The default audit trail volume is `/var/log/audit/audit.log`. For more information, see the `auditd.conf(5)` manual page.

By default, auditing captures specific events such as system logins, modifications to accounts, and `sudo` actions. You can also configure auditing to capture detailed system call activity or modifications to certain files. The kernel audit daemon (`auditd`) records the events that you configure, including the event type, a time stamp, the associated user ID, and success or failure of the system call.

The entries in the audit rules file, `/etc/audit/audit.rules`, determine which events are audited. Each rule is a command-line option that is passed to the `auditctl` command. You should typically configure this file to match your site's security policy.

The following are examples of rules that you might set in the `/etc/audit/audit.rules` file.

Record all unsuccessful exits from `open` and `truncate` system calls for files in the `/etc` directory hierarchy.

```
-a exit,always -S open -S truncate -F /etc -F success=0
```

Record all files opened by a user with UID 10.

```
-a exit,always -S open -F uid=10
```

Record all files that have been written to or that have their attributes changed by any user who originally logged in with a UID of 500 or greater.

```
-a exit,always -S open -F auid>=500 -F perm=wa
```

Record requests for write or file attribute change access to `/etc/sudoers`, and tag such record with the string `sudoers-change`.

```
-w /etc/sudoers -p wa -k sudoers-change
```

Record requests for write and file attribute change access to the `/etc` directory hierarchy.

```
-w /etc/ -p wa
```

Require a reboot after changing the audit configuration. If specified, this rule should appear at the end of the `/etc/audit/audit.rules` file.

```
-e 2
```

You can find more examples of audit rules in `/usr/share/doc/audit-`*`version`*`/stig.rules`, and in the `auditctl(8)` and `audit.rules(7)` manual pages.

Stringent auditing requirements can impose a significant performance overhead and generate large amounts of audit data. Some site security policies stipulate that a system must shut down if events cannot be recorded because the audit volumes have exceeded their capacity. As a general rule, you should direct audit data to separate file systems in rotation to prevent overspill and to facilitate backups.

You can use the `-k` option to tag audit records so that you can locate them more easily in an audit volume with the `ausearch` command. For example, to examine records tagged with the string `sudoers-change`, you would enter:

```
# ausearch -k sudoers-change
```

The `aureport` command generates summaries of audit data. You can set up `cron` jobs that run `aureport` periodically to generate reports of interest. For example, the following command generates a reports that shows every login event from 1 second after midnight on the previous day until the current time:

```
# aureport -l -i -ts yesterday -te now
```

For more information, see the `ausearch(8)` and `aureport(8)` manual pages.

4.11 Using Advanced Intrusion Detection Environment

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

This tool is installed as follows:

```
# dnf install aide
```

When installed, you can modify the configuration in `/etc/aide.conf`. The configuration controls which files and directories are monitored by AIDE and also how logging and output should be handled.

AIDE stores its current information about a system's configuration state in a database located at `/var/lib/aide/aide.db`. Ideally you should store a copy of this database file at an external location so that you can replace it with a known safe state when you want to peform an audit. If the file does not yet exist, you can create one for the currect system state by running:

```
# aide --init
# cp /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

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

```
# 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!!
```

The following example shows changes that indicate a user was added to the system:

```
# aide --check
Start timestamp: 2019-12-04 05:55:13 -0800 (AIDE 0.16)
AIDE found differences between database and filesystem!!

Summary:
  Total number of entries:      122738
  Added entries:                2
  Removed entries:              0
  Changed entries:              6

---------------------------------------------------
Added entries:
---------------------------------------------------

f++++++++++++++++: /etc/subgid-
f++++++++++++++++: /etc/subuid-

---------------------------------------------------
Changed entries:
---------------------------------------------------

f   ...    .C... : /etc/group
f   ...    .C... : /etc/gshadow
f   ...    .C... : /etc/passwd
f   ...    .C... : /etc/shadow
f   ...    .C... : /etc/subgid
f   ...    .C... : /etc/subuid

---------------------------------------------------
Detailed information about changes:
---------------------------------------------------

File: /etc/group
  SHA512   : W5tzAJAAXppdWLhwTrfLbj1hybSHwyxL | eiDsbAMeN8+3RGBwbvzO9Z2cSCirA19N
             Dta+VSq1EJ5nGnji/nmcrT+a2BAhZBVj | /0qxQqnR00PpAQxV0o0PDWVmSpoVQZtt
             eD5aT3AONeqXFAmSEbjpeA==         | 4Kv7CJ2JY+xAEn1ZLJuA1w==

File: /etc/gshadow
  SHA512   : IkHK5bWCT+rvJD5nnjWqaMtSs9O1IK/1 | 2/dZW5sMeVNeEn6L2u3DMk7U9M6LSNUO
             cbDZ3KjxCQBnGiWZF71i96Z4lsiF6HhD | BVQgMWmDJ1owZgCg2kRRR35F1QscD9vi
             XtSIl15nnLPgrOOOI4m5aA==         | 9AdxB37el4n2mZefr5eyuw==

File: /etc/passwd
  SHA512   : IDuW4odlc9sTUheajcI4e7DUCKIy4IIN | utGe+XyYkQev2tb7C0N8hKrWvM98R/f+
             6SDvQ559VPt571F3Ufy9nezUbzAiyvEF | JT5XT0YJmUf6xLw1DdX6KgNuRAwQD3xW
             RTZOdvMDrzwg2XeC4eXFpQ==         | J5Otz/NqogLRNMQY5ID+HQ==

File: /etc/shadow
  SHA512   : moPRI01xc9NR6dq4hhu+X7eG7JlYs+sa | 7WTuSitxpbvli7SoRqwcj+9puKzYeXaI
             nkjYz+3WMqOD7fmb0D7HC3kyieJyxExB | M3G8ChKgkFIRk6ASA+jBhBP6Pb7yVScK
             b1tPATwN4Ehq2sNjM272tQ==         | fr08ajH2WrKaoc/+Y5dM0w==

File: /etc/subgid
  SHA512   : z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXc | 1ZlaLZw6se3D5tHuTLq1G1V1QLyyHq+S
             g/SpIdNs6c5H0NE8XYXysP+DGNKHfuwv | tACG1j0SmYle1z7NoHtJ8MgCo9He5P/2
             Y7kxvUdBeoGlODJ6+SfaPg==         | wEstUtZ5DAnriW9LTL1khQ==

File: /etc/subuid
  SHA512   : z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXc | 1ZlaLZw6se3D5tHuTLq1G1V1QLyyHq+S
```

```
             g/SpIdNs6c5H0NE8XYXysP+DGNKHfuwv | tACG1j0SmYle1z7NoHtJ8MgCo9He5P/2
             Y7kxvUdBeoGlODJ6+SfaPg==         | wEstUtZ5DAnriW9LTL1khQ==


---------------------------------------------------
The attributes of the (uncompressed) database(s):
---------------------------------------------------

/var/lib/aide/aide.db.gz
  MD5      : /EcraAZYQoOY4MYKitwgqw==
  SHA1     : phqMw8phaa6SE8FevlkRmsXb3uU=
  RMD160   : DO63MxEGCVm3rWo6nRHePhtd/6o=
  TIGER    : G+z6vQRwH+QbTrRw3ottzUT9gTJvcD/A
  SHA256   : acDo9Rswl/pTdtwKiYEyzpb0sR1f655r
             1ZXAGDrcARI=
  SHA512   : insPGTVN8n966ccZWn2VvE21nLwJGnNt
             zv7zE5Y8/iLW9/tFxQT8pTeQrTCmomsD
             mRtj7BgkNvSZPYLSDfj6iQ==


End timestamp: 2019-12-04 05:55:54 -0800 (run time: 0m 41s)
```

If you configure this tool to run on a cron, it is possible to get regular reporting to indicate changes to system configuration and state that may help with early intrusion detection.

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

4.12 Implementing System Process Accounting

The `psacct` package implements the process accounting service in addition to the following utilities that you can use to monitor process activities:

`ac`	Displays connection times in hours for a user as recorded in the `wtmp` file (by default, `/var/log/wtmp`).
`accton`	Turns on process accounting to the specified file. If you do not specify a file name argument, process accounting is stopped. The default system accounting file is `/var/account/pacct`.
`lastcomm`	Displays information about previously executed commands as recorded in the system accounting file.
`sa`	Summarizes information about previously executed commands as recorded in the system accounting file.

Note

As for any logging activity, ensure that the file system has enough space to store the system accounting and `wtmp` files. Monitor the size of the files and, if necessary, truncate them.

For more information, see the `ac(1)`, `accton(8)`, `lastcomm(1)`, and `sa(8)` manual pages.

4.13 Protecting the Root Directory Using chroot Jails

A `chroot` operation changes the apparent root directory for a running process and its children. It allows you to run a program with a root directory other than `/`. The program cannot see or access files outside the designated directory tree. Such an artificial root directory is called a *chroot jail*, and its purpose is to limit the directory access of a potential attacker. The chroot jail locks down a given process and any user ID that it is using so that all they see is the directory in which the process is running. To the process, it appears that the directory in which it is running is the root directory.

Note

The `chroot` mechanism cannot 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 break out 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, you should ensure that a chroot jail does not contain any `setuid` or `setgid` executables that are owned by `root`.

For a `chroot` process to be able 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.

4.13.1 Running DNS and FTP Services in a Chroot Jail

If the DNS name service daemon (`named`) runs in a chroot jail, any hacker that enters your system via 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.

4.13.2 Creating a Chroot Jail

To create a chroot jail:

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

```
# mkdir /home/oracle/jail
```

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

```
# ldd /usr/bin/bash
 linux-vdso.so.1 =>  (0x00007fffdedfe000)
 libtinfo.so.5 => /lib64/libtinfo.so.5 (0x0000003877000000)
 libdl.so.2 => /lib64/libdl.so.2 (0x0000003861c00000)
 libc.so.6 => /lib64/libc.so.6 (0x0000003861800000)
 /lib64/ld-linux-x86-64.so.2 (0x0000003861000000)
```

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 recreate 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 have to the real root directory, for example:

```
# mkdir -p /home/oracle/jail/usr/bin
# 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.

```
# ln -s /home/oracle/jail/usr/bin /home/oracle/jail/bin
# 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:

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

4.13.3 Using a Chroot Jail

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

```
# chroot chroot_jail command
```

If you do not specify a command argument, chroot runs the value of the SHELL environment variable or /usr/bin/sh if SHELL is not set.

For example, to run /usr/bin/bash in a chroot jail (having previously set it up as described in (xref to set up procedure?)):

```
# chroot /home/oracle/jail
bash-4.2# pwd
/
bash-4.2# ls
bash: ls: command not found
bash-4.2# exit
exit
#
```

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.

Chapter 5 Security Considerations for Developers

Table of Contents

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

5.1 Design Principles for Secure Coding

The following well-established design principles are recommended for secure coding:

Least privilege	A process or user should be given only those privileges that are necessary to complete a task. User privileges should be assigned according to their role, but not otherwise. To create a minimal protection domain, assign rights when a process or thread requires them and remove them afterwards. This principle limits the potential damage that can result from attacks and user errors.
Economy of mechanism	Keep the design simple. There is less to go wrong, fewer inconsistencies are possible, and the code is easier to understand and debug.
Complete mediation	Check every attempt to access to a resource, not just the first. For example, Linux checks access permissions when a process opens a file but not thereafter. If a file's permissions change while a process has the file open, unauthorized access can result. Ideally, one could argue that the permissions should be checked whenever an open file is accessed. In practise, such checking is considered to be an unnecessary overhead given the circumstances under which access was first obtained.
Open design	Security should not depend on the secrecy of the code's design or implementation, sometimes referred to as *security through obscurity*. For example, an open back door to a system is only as secure as the knowledge of its existence. Of course, this principle does not apply to information such as passwords or cryptographic keys, knowledge of which should also be shared among as few people as possible. For this 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. Typically, multiple privileges should be required 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 thus gains minimal access to the system.

Least common mechanism	A system should isolate users and their activities from each other. Users should not share processes or threads and information channels should not be shared between users.
Fail-safe defaults	The default action should be to deny access to an operation. Should an attempt to perform an operation be denied, the system is as secure as it was before the operation started.
Accountability	Log the user and their privileges for each action that he or she attempts to perform. Any logs should be capable of being rotated and archived to avoid filling up a file system.
Psychological acceptability	Security mechanisms should be easy to install, configure, and use so that a user is less tempted to try to bypass them.

5.2 General Guidelines for Secure Coding

The following coding practices are recommended:

- 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 XML and 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, do not use files in world-writable directories, verify that a file being written to is not actually 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.
- Do not 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 necessary.
- Perform assert checking on variables that can take a finite set of values.
- Log information about privileged actions and error conditions. Do not allow the program to dump a core file on an end-user system.
- Do not echo passwords to the screen, or transmit or store them as clear text. Before transmitting 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 your program uses a pseudo-random number generating routine, verify that the numbers that it generates are sufficiently random for your requirements. You should also use a good random seed that a potential attacker should not be able to predict. See RFC 4086, *Randomness Requirements for Security*, for more information.
- It is recommended that Address Space Layout Randomization (ASLR) is enabled on the host system as this feature can help defeat certain types of buffer overflow attacks. See Section 4.4.1, “Address Space Layout Randomization”.

- When compiling and linking your program, use the Position Independent Executables (PIE) feature to generate a position-independent binary. See Section 4.4.3, “Position Independent Executables”.
- Consider using `chroot()` to confine the operating boundary of your program to a specified location within a file system.
- Do not execute a shell command by calling `popen()` or `syscall()` from within a program, especially from a `setuid` or `setgid` program.

The following guidelines apply if your program has its `setuid` or `setgid` bit set so that it can perform privileged actions on behalf of a user who does not possess those privileges:

- Do not set the `setuid` or `setgid` bit on shell scripts. However, if you use Perl scripts that are `setuid` or `setgid`, `perl` runs in *taint mode*, which is claimed to be more secure than using the equivalent 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 a separate, closely-monitored thread or process.
- Do not allow a `setuid` or `setgid` program to execute child processes using `execlp()` or `execvp()`, which use the `PATH` environment variable.

5.3 General Guidelines for Network Programs

The following coding practices are recommended for 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. The two IP addresses should be identical.
- Protect a service against Denial of Service (DoS) attacks by allowing it to stop processing 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 does not conform to what the program expects.
- Use certificates or preshared keys to authenticate the local and remote ends of the network connection.
- Use a well-established technology such as TLS or SSL to encrypt data sent over the network connection.
- Wherever possible, use existing networking protocols and technologies whose security characteristics are well known.
- Log information about successful and unsuccessful connection attempts, data reception and transmission errors, and changes to the service state.