长擎安全操作系统 24

用户手册 V2.0

北京长擎软件有限公司

二〇二四年十二月

目 录

1 概述

长擎安全操作系统 24（简称 CQOS24）立足“本质安全”理念构建信息系统安全底座，结合“核心安全+最小权限”的原则，以“安全最优，集成创新，按需供给”为行业客户提供高安全和专用化的系统级安全解决方案。

长擎安全操作系统 24 从多个方面提供安全保障，提供了身份认证、细粒度的自主访问控制、安全标记、特权最小化、强制访问控制、禁止客体重用、可信路径、可信信道、三权分立、数据完整性保护、安全审计等安全功能，为用户提供了从内核到应用的全方位安全防护。

2 硬件配置要求

- CPU 架构：x86_64、aarch64、loongarch64、mips64、sw64；
- 内存：2G 及以上；
- 硬盘：4G 及以上。

3 系统登录

任何一个要使用系统资源的用户，必须首先向系统管理员申请账号，并使用该账号登录系统。用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性保护。每个用户账号都拥有一个唯一的用户名和各自的口令。用户在登录时键入正确的用户名和口令后，便可访问系统进入个人主目录。

⚠ **需要注意的是，依据 CQOS24 安全策略要求，用户密码的设置需要满足如下要求：**

- **密码必须至少包含以下三种字符类型中的任意三种：大写字母、小写字母、数字和特殊字符；**
- **密码长度不少于 12 个字符。**

另外，长擎安全操作系统 24 还提供了双因子认证机制。当系统启用了双因子认证机制后，则要求系统管理员为所有创建的用户分配其专用的 Ukey。用户登录时，只有 Ukey 认证和用户口令认证均通过后，才能够正常进入系统和用户的主目录。详细的 Ukey 分发和使用方法，可以参考本文档“10.1.2 双因子认证”的内容。

4 系统基础配置

本章将主要介绍长擎安全操作系统 24 中供系统管理员进行基本服务以及工具的配置使用，这些命令主要用于配置系统基本服务，对系统进行的相应的功能模块的设定等，保证系统可用性。

4.1 语言环境配置

CQOS24 语言环境可以通过 localectl 来设置修改，对应的参数设置保存在 /etc/locale.conf 文件中。这些参数，会在系统启动的早期被 systemd 的守护进程读取。

- 查询当前系统语言环境，使用命令如下：

```
[root@localhost ~]# localectl status
    System Locale: LANG=en_US.UTF-8
        VC Keymap: cn
       X11 Layout: cn
```

- 查询系统可用语言环境，使用命令如下：

```
[root@localhost ~]# localectl list-locales
C.UTF-8
en_AU.UTF-8
en_BW.UTF-8
en_CA.UTF-8
...
```

- 设置系统语言环境，使用命令如下：

```
[root@localhost ~]# localectl set-locale LANG=zh_CN.UTF-8
```

⚠ **注意：此操作需要系统管理员（root）执行，执行成功后需要重新登录或者执行 source /etc/locale.conf，使修改设置生效。**

4.2 键盘布局配置

本节介绍如何通过 localectl 命令配置键盘布局。

- 查询系统当前配置，使用命令如下：

```
[root@localhost ~]# localectl status
    System Locale: LANG=zh_CN.UTF-8
        VC Keymap: cn
       X11 Layout: cn
```

- 查询系统可用键盘布局列表，使用命令如下：

```
[root@localhost ~]# localectl list-keymaps
ANSI-dvorak
af
```

```
af-fa-olpc
af-olpc-ps
af-ps
af-uz
…
```

- 修改系统默认键盘布局，使用命令示例如下：

```
[root@localhost ~]# localectl set-keymap {map}
```

⚠ **注意：该设置只允许系统管理员（root）操作。**

用户可以配置适合的键盘布局标识符以替代｛map｝，如修改为捷克语标识符，捷克语标识符为 cz，则使用命令如下：

```
[root@localhost ~]# localectl set-keymap cz
[root@localhost ~]# localectl status
    System Locale: LANG=zh_CN.UTF-8
        VC Keymap: cz
       X11 Layout: cn
```

4.3 系统日期和时间配置

本节介绍如何通过 timedatectl 命令来设置系统的日期、时间和时区。

- 查看当前系统日期和时间，使用命令如下：

```
[root@localhost ~]# timedatectl
                      Local time: 五 2024-03-29 10:42:27 CST
                  Universal time: 五 2024-03-29 02:42:27 UTC
                        RTC time: 五 2024-03-29 02:42:27
                       Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: yes
                  NTP service: active
              RTC in local TZ: no
```

以上返回结果中 NTP service:active 表示开启了时钟同步，在该模式下无法手动修改系统时间。需关闭后再修改，关闭 NTP service 命令如下：

```
[root@localhost ~]# timedatectl set-ntp no
```

- 修改系统时间，使用命令如下：

```
[root@localhost ~]# timedatectl set-time HH:MM:SS
```

如修改系统时间为 12:00:29，使用命令如下：

```
[root@localhost ~]# timedatectl set-time 12:00:29
```

修改系统日期，使用命令如下：

```
[root@localhost ~]# timedatectl set-time YYYY-MM-DD
```

如修改系统日期为 2023 年 5 月 1 日，命令如下：

```
[root@localhost ~]# timedatectl set-time 2023-05-01
```

开启时钟同步：

```
[root@localhost ~]# timedatectl set-ntp yes
```

- 查看系统当前可用时区，使用命令如下：

```
[root@localhost ~]# timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
...
```

- 修改系统的时区，使用命令如下：

```
[root@localhost ~]# timedatectl set-timezone time_zone
```

如修改系统时区为“亚洲/上海”，命令如下：

```
[root@localhost ~]# timedatectl set-timezone Asia/Shanghai
```

4.4 网络访问配置

4.4.1 动态网络配置

打开终端，以网口 ens160 为例，配置动态网络，修改网卡配置文件：

```
[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-ens160
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens160
DEVICE=ens160
ONBOOT=yes
```

其中 BOOTPROTO 的值改为 dhcp，之后重启网络。

```
[root@localhost ~]# nmcli connection down ens160 && nmcli connection up ens160
```

4.4.2 静态网络配置

打开终端，以网口 ens160 为例，配置静态网络：

```
[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-ens160
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens160
DEVICE=ens160
ONBOOT=yes
IPADDR=192.168.5.230
NETMASK=255.255.255.0
GATEWAY=192.168.5.1
```

其中 BOOTPROTO 的值改为‘static’，ens160 为配置的网口，根据设备的实际情况进行调整；IP、子网掩码、网关根据实际的网络按需配置。

4.4.3 配置 DNS

打开终端，编辑/etc/resolv.conf，设置 nameserver：

```
[root@localhost ~]# vim /etc/resolv.conf
# Generated by NetworkManager
nameserver 114.114.114.114
```

4.5 用户和用户组配置

长擎安全操作系统 24 中用户管理通过系统管理员管理。系统管理员创建用户，并为创建的用户设置密码，被创建的用户才能正常登录使用。

4.5.1 添加用户

系统管理员通过 useradd 命令可以为系统添加新用户信息。

用法：useradd [options] LOGIN

useradd 常见选项说明如下表所示：

序号	选项	说明
1	-D	单独使用该参数打印当前的默认值，配合选项使用为修改默认配置。
2	-c comment	新账号的备注信息。
3	-d home_dir	新账号每次登入时所使用的主目录（home dir），默认值为/home/账户名称。
4	-e expire_date	账号过期日期。日期的指定格式为 MM/DD/YY。
5	-f inactive_days	账号过期几天后永久停用。当值为 0 时账号则立刻被停用；当值为-1 时，则关闭此功能。默认值为-1。
6	-g initial_group	group 名称或一个数字作为用户的起始群组（group）。
7	-G	新账户的附加组列表。
8	-M	不创建用户的默认主目录。
9	-r	建立系统账号。在 CQOS24 中系统账号的 UID 小于 1000。
10	-s shell	用户登录后默认使用的 shell。 如 -s /bin/zsh。
11	-u uid	指定创建用户的 uid，不能与 CQOS24 中已经存在的 uid 一样。

例如新建一个用户 guest，命令如下：

```
[root@localhost ~]# useradd guest
```

以上命令执行之后，没有任何提示，表明用户建立成功。这时并没有设置用户的口令，必须通过系统管理员使用 passwd 命令设置用户的密码，否则创建账号不能登录使用。

4.5.2 修改用户

系统管理员通过 usermod 命令可以修改用户信息。

用法：usermod [options] LOGIN

usermod 的大多数选项与 useradd 命令相同，在此用于修改对应的属性。usermod 特有的选项如下表所示。

序号	选项	说明
1	-a	将用户添加到附加组中。
2	-L	锁定用户账号。

例如修改用户 guest 默认 shell，命令如下：

```
[root@localhost ~]# usermod -s /bin/zsh guest
```

例如修改用户 guest 主目录，命令如下：

```
[root@localhost ~]# usermod -d /home/new guest
```

例如修改用户 guest 的有效期，命令如下：

```
[root@localhost ~]# usermod -e 2027/01/02 guest
```

4.5.3 删除用户

系统管理员通过 userdel 命令可删除现有用户。

userdel 常见选项说明如下表所示。

序号	选项	说明
1	-f	强制删除用户（甚至当用户已经登入 Linux 系统时此选项仍旧生效）。
2	-r	删除用户主目录以及目录中所有文件。

例如删除用户 guest 及主目录，命令如下：

```
[root@localhost ~]# userdel -r    guest
```

4.5.4 添加用户组

CQOS24 中每个用户都有所属用户组，系统可以对一个用户组中的所有用户进行集中管理，CQOS24 中系统管理员通过 groupadd 命令来添加用户组。

用法：groupadd [options] 用户组

常见选项说明如下表所示：

序号	选项	说明
1	-g	GID 指定新用户组的组标识号（GID）。
2	-o	一般与-g 选项同时使用，表示新用户组的 GID 可以与系统已有用户组的 GID 相同。

例如添加一个名为 guest 的用户组，命令如下：

```
[root@localhost ~]# groupadd guest
```

4.5.5 修改用户组

系统管理员使用 groupmod 命令修改用户组的属性。

用法：groupmod [options] 用户组

常见选项说明如下表所示：

序号	选项	说明
1	-g	GID 指定新用户组的组标识号（GID）。
2	-o	一般与-g 选项同时使用，表示新用户组的 GID 可以与系统已有用户组的 GID 相同。
3	-n	新用户组 将用户组的名字改为新名字。

例如将用户组 guest 的 GID 修改为 102，命令如下：

```
[root@localhost ~]# groupmod -g 102 guest
```

例如将名称为 guest 用户组修改名称为 guest2，命令如下：

```
[root@localhost ~]# groupmod -n guest2 guest
```

4.5.6 删除用户组

CQOS24 中系统管理员使用 groupdel 命令删除用户组。

用法：groupdel 用户组

例如删除用户组 guest，命令如下：

```
[root@localhost ~]# groupdel guest2
```

5 基础开发环境支持

长擎安全操作系统 24 为开发者提供的基础开发环境支持，以满足各种开发需求。

⚠ **注意：在软件仓库维护过程中，出于漏洞修复、bug 修正以及其他安全性或功能性改进的目的，以下软件涉及的版本可能会有所更新。所有版本更新都将确保与先前版本保持向后兼容性，以保障现有应用和服务的稳定运行。**

5.1 编译工具

长擎安全操作系统 24 提供多种主流的编译工具，以支持不同编程语言的编译需求。以下是本系统中包含的主要编译工具。

序号	名称	版本	简介
1	GCC	8.5.0	GCC（GNU Compiler Collection）是一个广泛使用的开源编译器套件，支持多种编程语言，包括 C、C++、Fortran 等。GCC 不仅提供了编译功能，还包括链接器、汇编器等一系列工具。
2	LLVM/Clang	15.0.7	LLVM 是一个模块化、可重用的编译器基础设施项目，Clang 则是基于 LLVM 的 C/C++/Objective-C 编译器前端。
3	go	1.18.10	用于编译 Go 语言的编译器。
4	rustc	1.64.0	用于编译 rust 语言的编译器。

5.2 构建工具

构建工具用于管理和自动化软件构建过程。以下是本系统中包含的主要构建工具。

序号	名称	版本	简介
1	Make	4.2.1	传统的构建工具，通过 Makefile 定义项目的构建规则。
2	CMake	3.20.2	跨平台的自动化构建系统，通过 CMakeLists.txt 描述项目构建过程。
3	Autotools	autoconf 2.69 automake 1.16.2	自动化配置和构建工具集合，包括 Autoconf、Automake 等。
4	Ninja	1.8.2	轻量级构建工具，生成的构建文件比 Make 更高效。
5	Maven	3.6.3	Java 项目管理工具，基于 POM 管理项目的构建、依赖和文档。
6	Gradle	4.4.1	Gradle 是一个基于 Apache Ant 和 Apache Maven 概念的项目自动化构建开源工具。
7	Ant	1.10.12	Apache Ant 是一个将软件编译、测试、部署等步骤联系在一起加以自动化的一个工具，大多用于 Java 环境中的软件开发。

5.3 编程语言与运行环境

长擎安全操作系统 24 支持多种编程语言及运行时环境，包括但不限于：

序号	编程语言或运行环境	支持情况	描述
1	Python	Python 3.6.8	提供广泛的库支持，适用于多种应用场景。
2	Java	OpenJDK 1.8 OpenJDK 11	适用于 Java 应用程序开发。
3	PHP	PHP 7.4.33	适用于 Web 开发。
4	Qt	Qt 4.8.7 Qt 5.15.3	适用于 GUI 应用程序开发。
5	Go	Go 1.18.10	适用于高性能网络服务和微服务开发。
6	Perl	Perl 5.26.3	适用于脚本编写和文本处理。
7	Rust	Rust 1.64.0	适用于系统编程和并发安全。
8	Erlang	Erlang 23.3.4.9	适用于分布式系统和实时通信。
9	Ruby	Ruby 2.5.9	适用于 Web 开发和脚本编写。

序号	编程语言或运行环境	支持情况	描述
10	Lua	Lua 5.3.4	适用于嵌入式系统。
11	Node.js	Node.js 18.19.0	适用于后端开发和服务器端 JavaScript。

5.4 系统库支持

长擎安全操作系统 24 提供了广泛的基础库支持，以满足多样化的开发和运行需求。以下是系统中支持的一些常见库及其版本。

序号	库名	版本	描述
1	boost	1.66.0	C++库集合，提供丰富的数据结构、算法和其他功能，增强 C++标准库的功能。
2	brotli	1.0.6	压缩库，用于高效的数据压缩。
3	bzip2	1.0.6	文件压缩库，支持 bz2 格式。
4	c-ares	1.13.0	异步 DNS 解析库。
5	cracklib	2.9.6	密码强度检测库。
6	cyrus-sasl	2.1.27	简单认证和安全层库。
7	duktape	2.6.0	嵌入式 JavaScript 引擎，用于 C/C++程序中执行 JavaScript 代码。
8	glibc	2.28	GNU C Library，提供 C 语言的标准库功能，包括基本的输入输出、字符串处理和内存管理等。
9	gmp	6.2.0	多精度运算库。
10	gnutls	3.6.16	安全套接层（TLS）库，提供加密通信支持。
11	gobject-introspection	1.68.0	用于生成 GObject 库的元数据，简化使用 GObject 的应用开发。
12	gpgme	1.16.0	GPG 邮件加密库。
13	jansson	2.14	JSON 解析库。
14	json-c	0.13.1	JSON 处理库。
15	krb5	1.18.2	Kerberos 认证协议库，提供安全的身份验证机制。
16	libaio	0.3.112	异步 I/O 库。
17	libarchive	3.5.3	用于处理各种存档格式的库。
18	libassuan	2.5.5	GnuPG 辅助库，用于与 GnuPG 交互。
19	libbpf	0.8.1	EBPF（Extended Berkeley Packet Filter）库，用于内核数据过滤。
20	libcap	2.61	POSIX.1e capabilities 库，允许程序设置和获取进程的能力。

序号	库名	版本	描述
21	libcap-ng	0.7.11	新一代的 capabilities 库，提供比 libcap 更简洁的 API。
22	libcomps	0.1.18	DNF 和 YUM 的组件库，用于处理 RPM 包的组件信息。
23	libdb	5.3.28	Berkeley DB 库，提供键值存储功能。
24	libdnf	0.63.0	DNF（Dandified YUM）的库，主要用于管理软件包的安装、升级和删除。
25	libedit	3.1	提供命令行编辑功能的库。
26	libestr	0.1.10	字符串操作库。
27	libevent	2.1.8	事件驱动库。
28	libfastjson	0.99.9	用于 JSON 解析的库。
29	libffi	3.4.2	外部函数接口库，支持调用不同语言的函数。
30	libgcrypt	1.10.2	加密库，支持多种加密算法，常与 GnuPG（GNU Privacy Guard）一起使用。
31	libgpg-error	1.42	GPG 错误处理库。
32	libidn2	2.2.0	国际化域名处理库。
33	libkcapi	1.2.0	Kubernetes 客户端库。
34	libksba	1.3.5	用于数字签名验证的库。
35	libldb	2.6.1	LDAP 数据库。
36	libmnl	1.0.4	Netlink 消息库。
37	libmodulemd	2.13.0	用于管理模块元数据的库。
38	libndp	1.7	Network Discovery Protocol 库，用于发现网络设备。
39	libnetfilter_conntrack	1.0.6	Netfilter 连接跟踪库。
40	libnfnetlink	1.0.1	Netfilter 网络层库。
41	libnftnl	1.1.5	Netfilter 表库。
42	libnl3	3.7.0	Netlink 库，用于与内核交换网络配置信息。
43	libnsl2	1.2.0	网络标准库。
44	libpcap	1.9.1	用于捕获网络数据包的库。
45	libpipeline	1.5.0	管道操作库。
46	libpsl	0.20.2	公共前缀列表库。
47	libpwquality	1.4.4	密码质量检查库。
48	librepo	1.14.2	仓库管理库，用于管理和操作软件包仓库。

序号	库名	版本	描述
49	libseccomp	2.5.2	安全计算模式库。
50	libsecret	0.20.4	一个跨桌面的库，用于通过 Secret Service 安全地存储和检索密码及其他敏感信息。
51	libselinux	3.3	SELinux 策略执行库。
52	libsemanage	3.3	SELinux 策略管理库。
53	libsepol	3.3	SELinux 策略解析库。
54	libsigsegv	2.11	信号处理库。
55	libsolv	0.7.20	解决 DNF 的包依赖关系的库。
56	libssh	0.9.6	SSH 协议库，提供安全的远程访问。
57	libtalloc	2.4.0	内存分配库。
58	libtasn1	4.13	ASN.1 编码库。
59	libtdb	1.4.8	Trivial Database 库，提供简单的键值对存储功能。
60	libtevent	0.14.1	事件处理库。
61	libtirpc	1.3.2	远程过程调用库。
62	libunistring	0.9.9	Unicode 字符串处理库。
63	libusbx	1.0.23	USB 设备库。
64	libuser	0.63	用户管理库。
65	libutempter	1.1.6	用于将用户会话记录到 utmp 和 wtmp 文件的库。
66	libverto	0.3.2	事件驱动的异步 I/O 框架。
67	libxcrypt	4.4.26	加密库，提供加密算法的实现，如 DES、MD5 等
68	libxml2	2.9.7	XML 解析库，处理 XML 文档的读取、解析和生成，广泛应用于各种应用程序。
69	libyaml	0.1.7	YAML 处理库。
70	mpfr	3.1.6	多精度浮点运算库。
71	ncurses	6.1	用于控制终端输出的库。
72	nettle	3.4.1	加密库，提供多种加密算法。
73	newt	0.52.20	用于创建图形用户界面的库。
74	npth	1.5	线程库。
75	openssl	1.1.1m	SSL/TLS 加密库，提供安全通信功能。
76	openssl-pkcs11	0.4.10	PKCS#11 库，用于硬件令牌。

序号	库名	版本	描述
77	pam	1.5.2	Pluggable Authentication Modules（PAM），用于身份验证。
78	pcre2	10.32	正则表达式库。
79	popt	1.18	命令行选项解析库。
80	protobuf-c	1.3.0	Protocol Buffers C 库。
81	slang	2.3.2	用于创建文本用户界面的库。
82	snappy	1.1.8	高效的压缩库。
83	sqlite	3.26.0	轻量级的 SQL 数据库引擎。
84	zlib	1.2.11	文件压缩库。

5.5 文本编辑工具

5.5.1 Emacs 编辑器

- **简介**

Emacs 是一个可扩展、高度可定制的文本编辑器，被誉为“可扩展的自文档编辑器”。它不仅仅是一个编辑器，还是一个集成环境，可以运行各种插件和脚本，用于编程、邮件管理、日历和更多。

Emacs 支持多种编程语言，包括但不限于 C、C++、Java、Python 等。

- **安装与配置**

安装 Emacs，命令如下：

```
[root@localhost ~]# dnf install -y emacs
```

配置 Emacs：

1) Emacs 的配置文件是.emacs 或 init.el，位于用户的主目录下。
2) 使用“emacs -q”命令启动 Emacs，参数“-q”表示不加载任何配置文件。
3) 使用“M-x”（按住 Alt 键再按 x 键）打开配置文件，例如输入 find-file ~/.emacs，根据实际情况更改里面的设置。
4) 使用 Emacs 创建一个新的文件（emacs 文件名称），测试是否成功。

- **缓冲区介绍**

Emacs 常用的缓冲区（Buffer）包括四个：Buffer、Mini-Buffer、Message Buffer 和 Scratch Buffer，下面分别介绍它们的作用和特点。

1) Buffer：打开 Emacs 便会显示编辑文本的缓冲区。

2) Message Buffer：命令缓冲区，用来显示从 Emacs 启动开始相关的所有操作命令信息的缓冲区。

3) Scrathc Buffer：草稿缓冲区，用来临时写东西，不会被保存，除非使用 Ctrl+x Ctrl+W 命令。

4) Mini-Buffer：主命令行。

- **基本读写操作**

1) 创建文档

在 Emacs 中输入 Ctrl+x Ctrl+f，这时会自动切换到 Mini-Buffer 缓冲区中，可以创建文档。

2) 保存与关闭文档

在新建的文档中写入 Python 最经典的入门级代码 print('Hello World!')，使用 Ctrl+x Ctrl+s 保存文档，并在 Mini-Buffer 中输入 y 即可保存文档。

- **常用快捷键汇总**

1) 常用帮助快捷键

序号	选项	说明
1	Ctrl+h + Ctrl+h	重复两次 Ctrl+h 可以打开帮助界面
2	Ctrl+h t	打开官方自带的 tutorial
3	Ctrl+h k	查询快捷键（shortcut key）的帮助信息
4	Ctrl+h v	查找变量（variables）的帮助信息
5	Ctrl+h f	查询快捷键的帮助信息

2) 光标控制快捷键

序号	选项	说明
1	Ctrl+v	进入下一页（view next screen）
2	Alt+v	进入上一页
3	Ctrl+l	在当前页面，将光标放置中心处，如果连续使用则分别放置于最下、中心和最上
4	Ctrl+p	将光标移动到上一行（previous）
5	Ctrl+n	将光标移动到下一行（next）
6	Ctrl+b	将光标移动到后一个位置（backward）
7	Ctrl+f	将光标移动到前一个位置（forward）
8	Alt+b	将光标向后移动一个单词（backward），如果是中文文档，则向前移动一个标点符号的内容
9	Alt+f	将光标向前移动一个单词（forward），如果是中文文档，则向前移动一个标点符号的内容
10	Ctrl+e	将光标移动到行尾（end）
11	Alt+e	将光标移动到段尾/函数尾（end）
12	Ctrl+a	将光标移动到行首（ahead）
13	Alt+e	将光标移动到段首/函数首（ahead）
14	Alt+<	将光标移动到文件开头
15	Alt+>	将光标移动到文件末尾
16	Ctrl+u	指定重复次数，比如 Ctrl+u 8 Ctrl+f 表示向前移动 8 个字符

3) 常规操作快捷键

序号	选项	说明
1	Ctrl+d	删除一个字符
2	Alt+d	删除一个单词
3	Alt+x 输入 linum-mode	显示行号
4	Ctrl+g	终止当前输入的命令
5	Ctrl+x Ctrl+f	打开一个文件，该命令后面总是跟随目录/文件名，如果要打开的文件不存在，则创建新的文件
6	Ctrl+x k	删除缓冲区，执行该命令后，在 Mini-Buffer 处显示 Kill buffer:，输入要删除的缓冲区名即可
7	Ctrl+x Ctrl-w	修改文件名并保存
8	Ctrl+w	剪切
9	Alt+w	复制
10	Ctrl+y	粘贴
11	Ctrl+s	向前搜索
12	Ctrl+r	向后搜索
13	Alt+%	进行替换，输入 y 确定
14	Ctrl+_	撤销操作
15	Ctrl+x Ctrl+s	保存文档
16	Alt+x	运行（run command）
17	Ctrl+x Ctrl+c	退出 Emacs

4) 多窗口快捷键

序号	选项	说明
1	ctrl+x 2	将光标所在的窗口水平分为两个窗口
2	ctrl+x 3	将光标所在的窗口垂直分为两个窗口
3	ctrl+x 0	关闭光标所在的窗口
4	ctrl+x 1	关闭除光标所在窗口以外的所有窗口
5	ctrl+x o	将光标切换到下一个窗口

5.5.2 Vim 编辑器

- **简介**

vim 是一个高度可配置的文本编辑器。它提供了丰富的定制选项以满足个人需求，包括键位、颜色方案及插件的设置。它支持对多种编程语言的语法高亮与代码折叠，并且可以通过脚本实现编辑任务的自动化。

- **安装与配置**

安装 vim 编译器，命令如下：

```
[root@localhost ~]# dnf install -y vim
```

配置 vim 编译器：

vim 编译器是一个“字符型编译器”，在编译器中输入设置的配置命令是一次性的，想永久生效可以把这些配置命令放到.vimrc 文件内，这样 vim 打开时会自动先把.vimrc 文件执行一遍。

1) 把 vim 配置文件复制到 root 文件夹下。这个是一个隐藏文件，需要使用 ls -a 查看。

2) 使用“vim 空格 ~/.vimrc”打开配置文件，按照实际情况更改里面的设置。

3) 使用 vim 创建一个新的文件（vim 空格 文件名称），测试是否成功。

- **使用方法**

1) Vim 打开文件方式

序号	打开方式	命令
1	普通方式	vim 文件名
2	打开并跳到指定行	vim+数字(行号)文件名
3	打开文件并高亮显示指定关键词	vim+/(关键词)文件名
4	打开多个文件	vim 文件名

2) vim 模式

Vim 有四种模式。分别为：命令模式、编辑模式（插入模式）、底行模式、可视模式。

➢ 命令模式

序号	操作	命令
1	页操作	Ctrl+b：上翻页； Ctrl+f：下翻页。
2	行操作	G：将光标移动到末行（最后一行行首位置） shift + g 或者大写锁定 + g； nG：n 为数字，nG 是将光标跳至第 n 行首字符位置； gg：将光标移动到首行（第一行行首）； shift + J，将光标及下一行进行合并。
3	一行内光标操作	Shift+6：将光标跳至光标行 行首位置； ^首 Shift+4：将光标跳至光标行 行尾位置； $尾 n+←：按光标位置向左移动 n（数字）个字符 n+→：按光标位置向右移动 n 个字符
4	复制操作	yy：复制光标当前行； nyy：复制光标当前行及以下 n-1 行。
5	粘贴	p：粘贴到光标行及以下行，P 同理。
6	删除/剪切操作	D/dd：删除光标所在行内容； nD/ndd：删除光标行及以下 n-1 行内容。
7	撤销操作	u：撤销/返回到上一步操作。
8	单词	ciw：删除光标所在单词； B/b：跳至光标前一个单词； W/w：跳至光标后一个单词。

➢ 插入模式 OIAS

序号	选项	说明
1	A	在所在光标行行尾位置进行插入
2	a	在光标位置后紧跟位置进行插入
3	I	在所在光标行行首位置进行插入
4	i	在光标位置前紧跟位置进行插入
5	O	在光标行上一行进行插入
6	o	在光标行下一行进行插入
7	S	删除光标所在行并进行插入

8	s	删除光标所在字符并进行插入

➢ 底行模式

序号	命令	说明
1	:w	保存
2	:w!	强制保存
3	:wq	保存并退出
4	shift+zz	保存并退出
5	:w file	把当前文件另存为路径下文件
6	:files	查看打开的文件
7	:open file	打开指定文件
8	:bn	下一个文件
9	:bp	上一个文件
10	/(关键词)	将本文件中关键词进行高亮显示
11	:s/(原始字符串)/(目标字符串)/g	对于当前光标行首个原始字符串进行替换
12	:s/(原始字符串)/(目标字符串)	对于当前行中全部的原始字符串进行替换为目标字符串
13	:开始行,结束行 s/(原始字符串)/(目标字符串)	对于开始行至结束行中原始字符串进行替换
14	:set nu	显示行号
15	:set nonu	取消显示行号

➢ 可视模式

v：进入可视模式进行选择 可以复制 y。

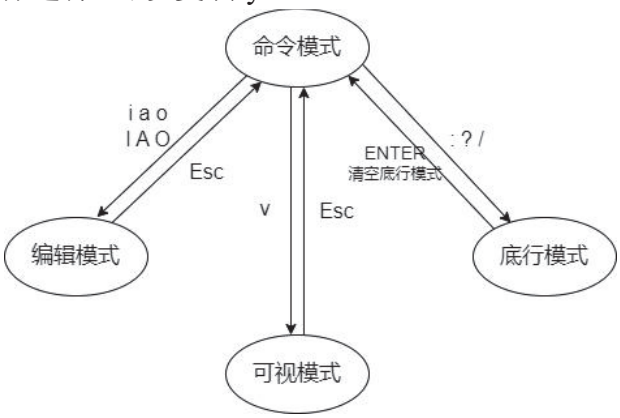

图 5-1 vim 模式关系图

3) 扩展知识

vim -o 用分割屏幕窗口方式同时打开多个文件。

在窗格间切换的方法：

Ctrl+w+方向键——切换到前 / 下 / 上 / 后一个窗格；

Ctrl+ww——依次向后切换到下一个窗格中。

外部数据通过复制命令（ctrl+c），复制至 vim 中需开启插入模式，使用 shift+ctrl+v。

5.5.3 Nano 编辑器

- 简介

Nano 是一个简单易用的文本编辑器，适合初学者。它具有用户友好的界面和相对简单的操作。Nano 提供了内联帮助，可以在编辑时提供指导。

- 安装与配置

安装 Nano：

```
[root@localhost ~]# dnf install -y nano
```

配置 Nano：

(1) Nano的配置文件是~/.nanorc，位于用户的主目录下。

(2) 使用“nano ~.nanorc”打开配置文件，按照实际情况更改里面的设置。

(3) 使用Nano创建一个新的文件（nano 文件名称），测试是否成功。

- Nano 的使用

⚠ **注意：nano 命令中不区分大小写。**

(1) Nano打开文件方式

序号	打开方式	命令
1	普通方式	nano 文件名，例如：nano 1.txt
2	打开并跳到指定行	nano +行号,列号 文件名，例如：nano +3,5 1.txt
3	打开多个文件	nano 文件名 1 文件名 2
4	以只读方式打开文件	nano -v myfile

(2) 基础命令

序号	选项	说明
1	使用箭头键移动光标	移动光标
2	Alt+6	复制当前行
3	Ctrl+U	粘贴文本至光标处
4	Ctrl+K	删除/剪切当前行
5	Alt+U	撤销上一步操作
6	Ctrl+G	显示帮助文本
7	Ctrl+O	保存当前文件
8	Ctrl+R	读取其他文件并插入光标位置
9	Ctrl+Y	跳至上一屏幕
10	Ctrl+C	显示光标位置
11	Ctrl+X	退出编辑文本
12	Ctrl+J	对齐当前段落（以空格为分隔符）
13	Ctrl+W	搜索文本位置
14	Ctrl+V	跳至下一个屏幕
15	Ctrl+T	运行拼写检查

(3) 搜索、查找、替换

搜索命令：^W（^即为 Ctrl）。

查找和替换文本：先 ^W 进入搜索界面，然后 ^R 输入要替换的文本，然后回车即可。

正则搜索：ALT + R（Mac：ESC + R）。

序号	选项	说明
1	CTRL+Q	开始向后搜索
2	CTRL+W	开始向前搜索
3	ALT+Q	向后搜索下一个匹配的文本
4	ALT+W	向前搜索下一个匹配的文本
5	ALT+R	搜索和替换

(4) 跳转

序号	选项	说明
1	^T 然后输入行号	跳转指定行
2	CTRL + B	将光标向后移动一个字符
3	CTRL + F	将光标向前移动一个字符
4	CTRL + ←	将光标向后移动一个字
5	CTRL + →	将光标向前移动一个字
6	CTRL + A	将光标移动到该行的起点
7	CTRL + E	将光标移动到行尾
8	CTRL + P	将光标向上移动一行
9	CTRL + N	将光标向下移动一行
10	CTRL + ↑	将光标移动到前一个文本块
11	CTRL + ↓	将光标移动到下一个文本块
12	CTRL + Y	将光标向上移动一整页文字
13	CTRL + V	将光标向下移动一整页文字
14	ALT + \	将光标移动到文件的顶部
15	ALT + /	将光标移动到文件的底部
16	ALT + G	转到指定行
17	ALT +]	转到下一个搜索结果
18	ALT + ↑	向上滚动视口
19	ALT + ↓	向下滚动视口
20	ALT + <	切换到前一个缓冲区
21	ALT + >	切换到下一个缓冲区

5.6 工控协议支持

长擎安全操作系统 24 能支持的工业控制协议如下表所示。

序号	协议名称	协议描述
1	Modbus	Modbus 是一种串行通信协议，是 Modicon 公司（施耐德电气 Schneider Electric）于 1979 年为使用可编程逻辑控制器（PLC）通信而发表。Modbus 已经成为工业领域通信协议的业界标准（De facto），并且现在是工业电子设备之间常用的连接方式。
2	OPC UA	OPC UA 协议主要用于在工业自动化系统中交换信息。它是一种基于服务的通信协议，可以在各种不同的系统和设备之间实现互操作性。
3	CANopenSocket	CANopenSocket 是一种通过 CAN 网络在不同设备之间进行 CANopen 通信的协议。它允许在不同的计算机和设备之间建立 CAN 网络的虚拟连接，从而可以进行远程设备配置和诊断。
4	DNP3	DNP3.0 协议为电力行业常用的一种工控协议，它是在国际电子电工协会（IEC）的 TC57 协议基础上制定的通信规约，它支持 ISO 的 OSI/EPA 模型，这种模型规定了物理层，数据链路层和应用层，然而，为了支持高级的 RTU 功能和大于最大帧长的报文，其数据链路采用一个伪传输层去完成最短报文的组装与分解。
5	CIP	CIP 协议（Common Industrial Protocol，即通用工业协议）是一种应用在

序号	协议名称	协议描述
		工业自动化的通信协定，为开放的现场总线 DeviceNet、ControlNet、Componet、EtherNet/IP 等网络提供了公共的应用层和设备描述。
6	EtherCAT	EtherCAT 是一种高性能、实时性良好的工业以太网通信协议。它是以太网技术的一种扩展和优化，通过实现高速同步和实时通信，成为应用于工业自动化领域的一种通信标准。
7	ProfiBus	ProfiBus 是一个用在自动化技术的现场总线标准，广泛应用于工业自动化、过程控制、制造执行系统（MES）等领域。
8	Fieldbus	Fieldbus（即现场总线）是一种工业数据总线，它主要解决工业现场的智能化仪器仪表、控制器、执行机构等现场设备间的数字通信以及这些现场控制设备和高级控制系统之间的信息传递问题。

⚠ **注意：表中所列协议，为厂商适配验证过的支持情况。实际支持情况不限于上述协议。因为在实际部署场景下，很多工控协议具有实时性要求高、可靠性要求强、领域性明显、协议非开源等特点，所以其他工控协议的支持，还需要根据实际场景下的适配测试情况而定。**

6 安装和管理软件

CQOS24 采用 dnf 进行软件包管理，作为新一代的 RPM 软件包管理器，dnf 能够查询软件包信息、从指定的软件源获取软件包，并自动处理依赖关系，以实现软件包的安装或卸载，同时还能将系统更新至最新的可用版本。dnf 与 yum 完全兼容，提供了与 yum 相同的命令行操作，以及用于扩展和插件的 API。

⚠ **需要注意的是，部分 dnf 命令需要具备管理员权限才能执行。**

6.1 配置 DNF

dnf 的配置文件主要用于控制 dnf 的行为和操作，位于/etc/dnf/dnf.conf 文件中，它通过不同的设置选项来调整软件包管理系统的运行方式。这些选项可以定义缓存管理、日志记录、仓库更新等内容。

dnf.conf 是一个 INI 格式的文件，该文件包含两部分：

- [main]：保存 dnf 的全局设置。
- [repository]：保存软件源的设置，可以有零个或多个“repository”。

dnf.conf 文件的[repository]部分并不是默认包含的，该部分通常用来定义仓库，大多数情况下，dnf 会使用位于/etc/yum.repos.d/中的.repo 文件来管理仓库，而不是在 dnf.conf 中直接定义，dnf.conf 主要用于全局设置。在 CQOS24 中，dnf 使用的就是/etc/yum.repos.d/中的 verything.repo 文件来管理仓库。

6.1.1 main 配置选项

/etc/dnf/dnf.conf 文件包含的[main]部分配置示例如下：

```
[main]
gpgcheck=1
installonly_limit=3
clean_requirements_on_remove=True
best=True
skip_if_unavailable=False
```

[main]部分配置常用选项说明如表 6-1 所示：

表 6-1 main 配置常用参数说明

序号	参数	说明
1	gpgcheck	是否启用 GPG 签名检查，确保从仓库中下载的软件包的真实性和完整性，1 表示启用（默认），0 表示禁用。
2	installonly_limit	设置可以同时安装“installonlypkgs”指令中列出的任何单个软件包的最大版本数，尤其是针对内核包，默认值为 3。
3	clean_requirements_on_remove	用于控制移除软件包时是否自动清理不再需要的依赖包。
4	best	控制 dnf 是否总是尝试安装可用的最新版本的软件包。
5	skip_if_unavailable	用于控制当仓库不可用时是否跳过该仓库继续进行操作。
6	exclude	阻止 dnf 更新或安装指定的软件包。
7	cachedir	定义缓存目录的路径，用于存储 RPM 包和数据库文件。
8	keepcache	指定是否在安装完软件包后保留下载的缓存文件，默认为 0，即安装后删除文件。
9	debuglevel	设置 dnf 生成的 debug 信息，取值范围为[0-10]，数值越大输出的 debug 信息越详细，默认为 2，0 为不输出信息。
10	obsoletes	设置是否允许更新陈旧的 RPM 包，默认值为 1，表示允许更新。
11	plugins	表示启用或禁用 dnf 插件，默认为 1，表示启用 dnf 插件。

6.1.2 repository 配置选项

配置 repository 部分有两种方式，一种是直接配置/etc/dnf/dnf.conf 文件中的[repositort]部分，另外一种是配置/etc/yum.repo.d 目录下的.conf 文件。

CQOS24 中的/etc/yum.repos.d/everything.repo 配置如下：

```
[everything]
name=CQOS-$releasever - everything
baseurl=https://repo.cqsoftware.com.cn/24/everything/$basearch
enabled=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CQOS
```

[repository]部分配置参数说明如表 6-2 所示：

表 6-2 repository 配置参数说明

序号	参数	说明
1	name	软件仓库（repository）描述的字符串。
2	baseurl	软件仓库（repository）的地址。
3	enabled	是否启用该仓库。
4	gpgkey	指定用于验证软件包的 GPG 公钥的 URL。

6.1.3 显示配置信息

显示当前的配置信息：

```
[root@localhost ~]# dnf config-manager --dump
```

它会输出系统中所有 DNF 配置文件的内容，包括各个软件源的设置和其他相关选项。

- 查看某个仓库的配置信息：

```
[root@localhost ~]# dnf config-manager --dump repo_name
```

repo_name 为想要查看仓库的名字，可以用 dnf repolist 进行查询。

- 使用一个全局正则表达式，来显示所有匹配部分的配置：

```
[root@localhost ~]# dnf config-manager --dump glob_expression
```

glob_expression 是一种使用通配符进行模式匹配的表达式。

列出启用的仓库：

```
[root@localhost ~]# dnf repolist
```

这个命令会显示当前已启用的所有 dnf 仓库。

- 列出所有仓库

```
[root@localhost ~]# dnf repolist all
```

该命令显示系统中所有的仓库，包括启用和禁用的。输出结果会增加仓库的状态标识，如 enabled 或 disabled。

6.1.4 创建本地软件源仓库

创建本地软件源仓库可以帮助在没有互联网连接的情况下进行软件安装和更新，或提高安装速度和管理一致性。下面是创建本地 DNF 仓库的步骤：

1.准备软件包，首先将需要的软件包准备好保存到一个目录下，假设将这些软件包放在/mnt/local_repo/目录下。

2. 确保系统上安装了 createrepo 工具，它用于生成仓库元数据，在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf install createrepo
```

3.创建软件源，执行如下命令：

```
[root@localhost ~]# createrepo /mnt/local_repo
```

4.在 /etc/yum.repos.d/ 目录下创建新的 dnf 仓库配置文件，配置文件以 .repo 结尾，例如 local.repo：

```
[local-repo]
name=Local Repository
baseurl=file:///path/to/local-repo
enabled=1
```

```
gpgcheck=0
```

5.更新仓库缓存

```
[root@localhost ~]# sudo dnf makecache
```

6.验证本地仓库，可以使用以下命令验证本地仓库是否正确配置并能正常工作：

```
[root@localhost ~]# dnf repolist
```

6.1.5 添加、启用和禁用软件源

6.1.5.1 添加软件源

要添加一个新的软件源仓库，可以在 /etc/dnf/dnf.conf 文件中添加“repository”部分，但更推荐在/etc/yum.repos.d/目录下添加“.repo”文件的方式，每个软件源都有自己对应的“.repo”文件，就像 6.1.4 创建本地软件源仓库中第 4 步提到的那样的做法。除此之外，也可以在 root 权限下执行如下命令，执行完成之后会在/etc/yum.repos.d/目录下生成对应的 repo 文件。其中 repository_url 为 repo 源地址。

```
[root@localhost ~]# dnf config-manager --add-repo repository_url
```

6.1.5.2 启用软件源

在 root 权限下执行如下命令可以启用指定软件源，其中 repository 为新增.repo 文件中的 repo id（可通过 dnf repolist 查询）：

```
[root@localhost ~]# dnf config-manager --set-enable repository
```

也可以使用一个全局的正则表达式，来启用所有匹配的软件源。其中 glob_expression 是一种使用通配符进行模式匹配的表达式，用于同时匹配多个 repo id：

```
[root@localhost ~]# dnf config-manager --set-enable glob_expression
```

6.1.5.3 禁用软件源

在 root 权限下执行如下命令可以禁用指定软件源，其中 repository 为新增.repo 文件中的 repo id（可通过 dnf repolist 查询）：

```
[root@localhost ~]# dnf config-manager --set-disable repository
```

同样的，也可以使用一个全局正则表达式来禁用所有匹配的软件源：

```
[root@localhost ~]# dnf config-manager --set-disable glob_expression
```

6.2 检查和升级软件包

dnf 能够检查系统中的软件包是否需要更新。可以使用 dnf 列出所有待更新的软件包，并根据需求选择更新所有软件包，或仅更新特定的软件包。

6.2.1 检查软件包更新

检查软件包更新命令示例如下：

```
[root@localhost ~]# dnf check-update
```

```
上次元数据过期检查：0:00:32 前，执行于 2024 年 09 月 20 日 星期五 11 时 21 分 37 秒。
OpenEXR-libs.x86_64                    2.2.0-14.cq24                    everything
abi-dumper.noarch                      1.2-5.cq24                       everything acpid.x86_64
                2.0.32-8.cq24                    everything    apr-util.x86_64
1.6.1-10.cq24                    everything    apr-util-bdb.x86_64                    1.6.1-
10.cq24                          everything
```

dnf check-update 命令可以列出当前系统中所有可用的更新，它不会实际进行更新，只是显示哪些软件包有新版本可用。

6.2.2 升级软件包

如果需要升级单个软件包，在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf update package_name
```

升级 openssh 软件包命令为例：

```
[root@localhost ~]# dnf update openssh
上次元数据过期检查：0:16:33 前，执行于 2024 年 09 月 20 日 星期五 11 时 21 分 37 秒。
依赖关系解决。
================================================================================
软件包                  架构                  版本                  仓库                  大小
================================================================================
升级：
openssh                 x86_64                8.0p1-23.cq24         everything            523 k
openssh-clients         x86_64                8.0p1-23.cq24         everything            669 k
openssh-server          x86_64                8.0p1-23.cq24         everything            493 k
事务概要
================================================================================
升级  3 软件包
总下载：1.6 M
确定吗？[y/N]：
```

dnf 默认会显示升级软件包的基本信息，并提示是否确认安装，用户可以在使用 dnf 命令时添加参数–y，效果等同于出现的“确定吗 [y/N]：”时输入 yes。安装过程中如果出现错误导致安装过程终止，可以使用 dnf history 命令查看详细描述。

如果需要安装一组软件包，可以以 root 用户执行命令：

```
[root@localhost ~]# dnf group update group_name
```

要更新所有的包和它们的依赖，在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf update
```

6.3 管理软件包

dnf 提供了一整套用于管理软件包的命令，支持从检索、查看信息，到安装和删除软件包的多项操作，帮助用户轻松管理系统中的软件包。

6.3.1 搜索软件包

如果不知道想要安装的软件的准确名称时，可以使用 dnf search 命令可以快速搜索系统中的软件包。用户可以通过输入关键字来查找与其匹配的软件包名称或描述。如果搜索结果较多，用户可以通过 grep 命令和正则表达式来进一步过滤结果。示例如下：

```
[root@localhost ~]# dnf search tmux
```

6.3.2 列出软件包清单

dnf 可以列出系统中已安装的、可用的，以及与特定字符匹配的软件包。用户可以通过 dnf list 命令查看所有已安装和可用的软件包，也可以结合特定的搜索模式来筛选结果。例如，可以使用通配符查找与某些字符匹配的软件包。

要列出系统中所有已安装的以及可用的 RPM 包信息，可以使用命令如下：

```
[root@localhost ~]# dnf list all
```

要列出系统中特定的 RPM 包信息，可以通过正则表达式进行匹配：

```
[root@localhost ~]# dnf list glob_expression...
```

示例如下：

```
[root@localhost: libpfm]# dnf list tmux*
上次元数据过期检查：1:16:49 前，执行于 2024 年 09 月 20 日 星期五 12 时 39 分 03 秒。
已安装的软件包
tmux.x86_64                    2.7-3.cq24                    @@commandline tmux-
debuginfo.x86_64               2.7-3.cq24                    @@commandline tmux-
debugsource.x86_64             2.7-3.cq24                    @@commandline
可安装的软件包
tmux-doc.noarch                2.7-3.cq24                    everything
```

显示包括某些字符的已安装软件包列表可以执行以下命令：

```
[root@localhost ~]# dnf list installed glob_expression
```

显示包括某些字符的可安装软件包列表可以执行以下命令：

```
[root@localhost ~]# dnf list available glob_expression
```

6.3.3 查看软件仓库

用户查看系统中当前配置的仓库列表可以通过如下命令：

```
[root@localhost ~]# dnf repolist
```

如果想显示更多信息可以加上-v 选项，或者用 dnf repoinfo 命令输出信息。

```
[root@localhost ~]# dnf repolist -v
[root@localhost ~]# dnf repoinfo
```

如果需要显示所有可用和不可用的仓库，可以使用以下命令：

```
[root@localhost ~]# dnf repolist all
```

6.3.4 显示 RPM 包信息

要显示一个或者多个 RPM 包信息，使用命令如下：

```
[root@localhost ~]# dnf info package_name...
```

可以获得软件包的版本号、大小、架构及其他元数据信息。

6.3.5 安装 RPM 包

要安装一个软件包及其所有未安装的依赖，可以在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf install package_name
```

也可以通过添加软件包名字同时安装多个软件包：

```
[root@localhost ~]# dnf install package_name package_name... --setopt=strict=0
```

“--setopt=strict=0”是可选项，可以让 dnf 在遇到某些问题（比如找不到依赖项）时不会强制中断操作，而是以非严格模式继续进行安装操作。这个选项允许用户忽略一些可能并非关键的依赖检查，从而完成安装。--setopt 选项允许临时更改 dnf 配置参数。--setopt=strict=0 等价于在 dnf 的配置文件/etc/dnf/dnf.conf 中添加 strict=False，它会告诉 dnf 在此次安装过程中忽略一些严格的依赖或其他潜在错误。这是临时性的，只影响当前的命令执行，不会改变配置文件的永久设置。

如果要安装本地软件包，可以执行：

```
[root@localhost ~]# dnf localinstall path
```

6.3.6 下载软件包

如果下载软件包而不安装它们，可以在 root 权限下输入如下命令：

```
[root@localhost ~]# dnf download package_name
```

⚠ **注意：要确保已安装 dnf-plugin-download 插件。**

如果需要同时下载未安装的依赖，则加上--resolve，使用命令如下：

```
[root@localhost ~]# dnf download --resolve package_name
```

示例如下：

```
[root@localhost ~]# dnf download --resolve httpd
```

6.3.7 删除软件包

要卸载软件包以及相关的依赖软件包，在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf remove package_name...
```

示例如下：

```
[root@localhost ~]# dnf remove httpd
```

6.4 管理软件包组

软件包组是指一组具有共同目的或功能的相关联的软件包集合，旨在简化用户在系统中执行特定任务的操作。使用 DNF，用户可以对软件包组进行安装、删除等管理操作，从而更高效地维护和管理系统。通过安装或删除软件包组，用户可以一次性处理多项软件包，而不需要逐一手动安装每个软件包。这提高了操作效率，尤其适合快速配置系统的特定环境。

dnf 软件包管理主要有两个相关命令，分别是 dnf group 和 dnf groups，分别用于管理和查询系统中的软件包组，dnf group 主要用于对特定的软件包组进行操作，如安装、删除、升级或查询详细信息。dnf groups 用于查询系统中所有软件包组的状态，帮助用户查看有哪些可用或已安装的软件包组。它提供的是系统中所有软件包组的总览信息。

6.4.1 列出软件包组清单

dnf groups 使用 summary 参数，可以列出系统中所有已安装软件包组、可用的组、可用的环境组的数量，命令如下：

```
[root@localhost ~]# dnf groups summary
```

示例如下：

```
[root@localhost ~]# dnf groups summary
上次元数据过期检查：2:05:18 前，执行于 2024 年 09 月 20 日 星期五 12 时 39 分 03 秒。
可用组： 6
```

显示某个软件包组的全部信息可以用以下命令：

```
[root@localhost ~]# dnf groups info glob_expression…
```

要列出所有软件包组和它们的组 ID ，命令如下：

```
[root@localhost ~]# dnf group list
```

6.4.2 显示软件包组信息

要查看特定软件包组的详细信息，包括必需包和可选包，可以使用以下命令：

```
[root@localhost ~]# dnf group info "group_name"
```

例如，查看开发工具包组的信息：

```
[root@localhost ~]# dnf group info "Development Tools"
```

6.4.3 安装软件包组

每一个软件包组都有自己的名称以及相应的 id，软件包组的安装可以通过软件包组名称安装，也可通过包组 id 安装。

```
[root@localhost ~]# dnf group install "group name"
[root@localhost ~]# dnf group install groupid
```

也可用通过以下两种命令安装：

```
[root@localhost ~]# dnf install @group
[root@localhost ~]# dnf install @^group
```

下面是 4 种安装开发工具软件分组的示例：

```
[root@localhost ~]# dnf group install "Development Tools"
[root@localhost ~]# dnf group install development
[root@localhost ~]# dnf install @"Development Tools"
[root@localhost ~]# dnf install @development
```

6.4.4 删除软件包组

软件包组的删除同样可以通过软件包组名称和软件包组 id 进行删除。

```
[root@localhost ~]# dnf group remove group_name
[root@localhost ~]# dnf group remove groupid
```

也可用通过以下两种命令删除：

```
[root@localhost ~]# dnf remove @group
[root@localhost ~]# dnf remove @^group
```

以下是具体示例：

```
[root@localhost ~]# dnf group remove "Development Tools"
[root@localhost ~]# dnf group remove development
[root@localhost ~]# dnf remove @"Development Tools"
[root@localhost ~]# dnf remove @development
```

6.5 软件包操作记录管理

软件包操作记录管理是对系统中软件包安装、更新、删除等操作的记录和监控，旨在帮助用户跟踪软件包的状态和历史变化。dnf 可以使用 dnf history 命令进行管理操作。

6.5.1 查看操作

dnf history 命令用于查看系统中所有软件包管理操作的历史记录。执行该命令时，系统将列出最近的事务，包括安装、更新和删除等操作。如果显示内容过多，可以采用以下命令来显示第 1 到第 20 条的操作记录，以 root 权限执行：

```
[root@localhost ~]# dnf history list 1..20
```

dnf history list 输出的内容包括：ID（标识特定记录的编号）、Command line（简要描述操作内容）、Date and time（记录的日期和时间）、Action(s)（描述操作类型）、Altered（操作影响的条目数量）。这些信息帮助用户快速了解软件包管理操作的历史和影响。

下表 6-3 是 Action 的类型说明：

表 6-3 Action 的类型说明

序号	Action	缩写	描述
1	Downgrade	D	下载更新包
2	Erase	E	删除软件包
3	Install	I	安装软件包
4	Obsoleting	O	软件包标注废弃
5	Reinstall	R	软件包重装
6	Update	U	升级软件包

6.5.2 审查操作

需要显示某条操作记录的具体综述信息，可以执行以下命令：

```
[root@localhost ~]# dnf history <transaction_id>
```

其中 transaction_id 是操作的 id。

如果要查看某个特定操作记录的详细信息，可以使用：

```
[root@localhost ~]# dnf history info <transaction_id>
```

如果需要查看指定范围内操作记录的详细信息，通过指定开始和结束的操作 id 来查看：

```
[root@localhost ~]# dnf history info start_id. . end_id
```

示例如下：

```
[root@localhost ~]# dnf history info 4..5
```

6.5.3 恢复与重复操作

如果某次操作导致了问题，可以以 root 权限执行以下命令撤销特定操作，恢复到之前的状态：

```
[root@localhost ~]# dnf history undo <transaction_id>
```

如果需要重新执行某个操作，可以以 root 权限执行以下操作：

```
[root@localhost ~]# dnf history redo <transaction_id>
```

7 系统服务应用

7.1 系统服务管理工具

7.1.1 Systemd 介绍

systemd 是 Linux 下一个与 SysV 和 LSB 初始化脚本兼容的系统和服务管理器。systemd 使用 socket 和 D-Bus 来开启服务，提供基于守护进程的按需启动策略，保留了 Linux

Cgroups 的进程追踪功能，支持快照和系统状态恢复，维护挂载和自挂载点，实现了各服务间基于从属关系的一个更为精细的逻辑控制，拥有前卫的并行性能。systemd 无需经过任何修改便可以替代 sysvinit。

systemd 开启和监督整个系统是基于 unit 的概念。unit 是由一个与配置文件对应的名字和类型组成的（例如：avahi.serviceunit 有一个具有相同名字的配置文件，是守护进程 Avahi 的一个封装单元）。unit 类型如表 7-1 所示：

表 7-1 unit 说明

序号	Unit 类型	文件后缀	描述
1	Service unit	.service	此类 unit 属于系统服务，守护进程的启动、停止、重启和重载为此类型。
2	Target unit	.targe	此类 unit 为其他 unit 进行逻辑分组。
3	Automount unit	.automount	此类 unit 封装系统结构层次中的一个自挂载点。
4	Device unit	.device	此类 unit 封装一个存在于 Linux 设备树中的设备。
5	Mount unit	.mount	此类 unit 封装系统结构层次中的一个挂载点。
6	Path unit	.path	此类 unit 为文件系统中的一个文件或者目录。
7	Scope unit	.scope	外部创建的进程。
8	Slice unit	.slice	此类 unit 为一组管理系统进程的分层 unit。
9	Socket unit	.socket	此类 unit 封装系统和互联网中的一个 socket。
10	Swap unit	.swap	此类 unit 封装 swap 设备或者 swap 文件。
11	Timerunit	.timer	此类 unit 基于 systemd 计时器，封装系统时间。

Systemd unit 可用类型的具体路径及说明如表 7-1 所示：

表 7-2 Systemd unit 可用类型说明

序号	目录	描述
1	/usr/lib/systemd/system/	在 RPM 包安装时生成的 Systemd units。
2	/run/systemd/system/	在运行时创建的 Systemd units，该目录任务会覆盖安装时自带的 Systemd units。
3	/etc/systemd/system/	由系统管理员创建与管理的 Systemd units，该目录任务会覆盖运行时 Systemd units。

7.1.2 管理系统服务操作

Systemd 为用户提供了 systemctl 的系列命令用来运行，关闭、重启、显示、启用、禁用系统服务。

常用 systemctl 相关命令及作用如 7-3 表所示（使用目标服务名，替换下表 NetworkManager.service）：

表 7-3 systemctl 相关命令

序号	**systemd 命令**	**备注**
1	systemctl list-units --type service	显示当前正在运行的服务。
2	systemctl list-units --type service --all	显示所有服务。
3	systemctl start NetworkManager.service	用来启动一个服务（并不会重启现有的）。
4	systemctl stop NetworkManager.service	用来停止一个服务（并不会重启现有的）。

序号	systemd 命令	备注
5	systemctl restart NetworkManager.service	用来停止并启动一个服务。
6	systemctl reload NetworkManager.service	当支持时，重新装载配置文件而不中断等待操作。
7	systemctl condrestart NetworkManager.service	若服务正在运行，则重启此服务。
8	systemctl status NetworkManager.service	检查服务的运行状态。
9	systemctl enable NetworkManager.service	在下次启动时或满足其他触发条件时设置服务为启用。
10	systemctl disable NetworkManager.service	在下次启动时或满足其他触发条件时设置服务为禁用。
11	systemctl is-enabled NetworkManager.service	用来检查一个服务在当前环境下被配置为启用还是禁用。
12	systemctl list-unit-files \-\-type=service	输出在各个运行级别下服务的启用和禁用情况。
13	ls /etc/systemd/system/*.wants/NetworkManager.service	用来列出该服务在哪些运行级别下启用和禁用。
14	systemctl daemon-reload	当创建新服务文件或变更设置时使用。

以下是 systemctl 关键命令的具体说明：

（1）查看服务

如果需要显示当前正在运行的服务，使用命令如下：

```
[root@localhost ~]#systemctl list-units --type service
```

如果需要显示所有的服务（包括未运行的服务），需要添加-all 参数，使用命令如下：

```
[root@localhost ~]#systemctl list-units --type service –all
```

（2）显示服务状态

如果需要显示某个服务的状态，可执行如下命令：

```
[root@localhost ~]#systemctl status xxx.service
```

status 命令返回结果如表 7-4 所示：

表 7-4 status 命令返回结果

序号	参数	描述
1	Loaded	说明服务是否被加载，并显示服务对应的绝对路径以及是否启用。
2	Active	说明服务是否正在运行，并显示时间节点。
3	Main PID	相应的系统服务的 PID 值。
4	Tasks	显示该服务当前使用的任务数量。
5	Memory	显示服务使用的内存量。
6	CGroup	相关控制组（CGroup）的其他信息。

如果需要鉴别某个服务是否运行，可执行如下命令：

```
[root@localhost ~]#systemctl is-active xxx.service
```

is-active 命令的返回结果如表 7-5 所示：

表 7-5 is-active 命令返回结果

序号	状态	含义
1	active(running)	有一个或多个程序正在系统中执行。

序号	状态	含义
2	active(exited)	仅执行一次就正常结束的服务，目前并没有任何程序在系统中执行。举例来说，开机或者是挂载时才会进行一次的 quotaon 功能。
3	active(waiting)	正在执行当中，不过要等待其他的事件才能继续处理。例如：打印的队列相关服务就是这种状态，虽然正在启动中，不过也需要真的有队列进来（打印作业）这样他才会继续唤醒打印机服务来进行下一步打印的功能。
4	inactive	这个服务没有运行。

同样，如果需要判断某个服务是否被启用，可执行如下命令：

```
[root@localhost ~]#systemctl is-enabled xxx.service
```

is-enabled 命令的返回结果如表 7-6 所示：

表 7-6 is-enabled 命令返回结果

序号	状态	含义
1	enabled	已经通过/etc/systemd/system/目录下的 Alias=别名、.wants/或.requires/软链接被永久启用。
2	enabled-runtime	已经通过/run/systemd/system/目录下的 Alias=别名、.wants/或.requires/软链接被临时启用。
3	linked	虽然单元文件本身不在标准单元目录中，但是指向此单元文件的一个或多个软链接已经存在于/etc/systemd/system/永久目录中。
4	linked-runtime	虽然单元文件本身不在标准单元目录中，但是指向此单元文件的一个或多个软链接已经存在于/run/systemd/system/临时目录中。
5	masked	已经被/etc/systemd/system/目录永久屏蔽，因此 start 操作会失败。
6	masked-runtime	已经被/run/systemd/systemd/目录临时屏蔽，因此 start 操作会失败。
7	static	尚未被启用，并且单元文件的“[Install]”段中没有可用于 enable 命令的选项。
8	indirect	尚未被启用，但是单元文件的“[Install]”小节中 Also=选项的值列表非空，也就是列表中的某些单元可能已被启用，或者它拥有一个不在 Also=列表中的其他名称的别名软链接。对于模板单元来说，表示已经启用了一个不同于 DefaultInstance=的实例。
9	disabled	尚未被启用，但是单元文件的“[Install]”小节中存在可用于 enable 命令的选项。
10	generated	单元文件是被单元生成器动态生成的。被生成的单元文件可能并未被直接启用，而是被单元生成器隐含地启用了。
11	transient	单元文件是被运行时 API 动态临时生成的。该临时单元可能并未被启用。
12	bad	单元文件不正确或者出现其他错误。is-enabled 不会返回此状态，而是会显示一条出错信息。list-unit-files 命令有可能会显示此单元。

（3）运行服务

如果需要运行某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl start xxx.service
```

（4）关闭服务

如果需要关闭某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl stop xxx.service
```

（5）重启服务

如果需要关闭某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl restart xxx.service
```

执行命令后，当前服务会被关闭，但马上重新启动。如果指定的服务当前处于关闭状态，执行命令后，服务也会被启动。

（6）启用服务

如果需要在开机时启用某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl enable xxx.service
```

（7）禁用服务

如果需要在开机时禁用某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl disable xxx.service
```

7.1.3 改变运行级别

systemd 用目标（target）替代了运行级别的概念，提供了更大的灵活性，用户可以继承一个已有的目标，并添加其他服务，来创建自己的目标。表 7-7 列举了 systemd 下的目标和常见 runlevel 的对应关系：

表 7-7 运行级别和 systemd 目标

序号	运行级别	Systemd 目标	描述
1	0	runlevel0.target，poweroff.target	关闭系统。
2	1，s，single	runlevel1.target，rescue.target	单用户模式。
3	2，4	runlevel2.target，runlevel4.target，multi-user.target	用户定义/域特定运行级别。默认等同于 3。
4	3	runlevel3.target，multi-user.target	多用户，非图形化。用户可以通过多个控制台或网络登录。
5	5	runlevel5.target，graphical.target	多用户，图形化。通常为所有运行级别 3 的服务外加图形化登录。
6	6	runlevel6.target，reboot.target	重启系统。
7	emergency	emergency.target	紧急 Shell。

（1）查看系统默认启动目标

查看当前系统默认的启动目标，命令如下：

```
[root@localhost ~]#systemctl get-default
```

（2）查看当前系统所有启动目标

查看当前系统所有的启动目标，命令如下：

```
[root@localhost ~]#systemctl list-units --type=target
```

（3）改变默认目标

改变系统默认的目标，在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl set-default xxx.target
```

（4）改变当前目标

改变当前系统的目标，在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl isolate xxx.target
```

（5）切换到救援模式

改变当前系统为救援模式，在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl rescue
```

⚠ **注意：从救援模式进入正常模式，用户需要重启系统。**

（6）切换到紧急模式

改变当前系统为紧急模式，在 root 权限下执行如下命令：

```
[root@localhost ~]#systemctl emergency
```

⚠ **注意：从紧急模式进入正常模式，用户需要重启系统。**

7.1.4 关闭，暂停和休眠系统操作

systemd 通过 systemctl 命令可以对系统进行关机、重启、休眠等一系列操作。建议用户使用 root 权限通过 systemctl 命令进行操作。常用的 systemctl 操作命令如表 7-8 所示：

表 7-8 systemctl 操作命令

序号	Systemctl 命令	描述
1	systemctl poweroff	关闭系统并下电。
2	systemctl halt	关闭系统但不下电机器。
3	systemctl reboot	重启系统。
4	Systemctl --no-wall poweroff/halt/reboot	执行 poweroff/half/reboot 命令会给当前所有的登录用户发送一条提示消息。如果不想让 systemd 发送该消息，可以添加“--no-wall”参数。
5	systemctl suspend	使系统待机。
6	systemctl hibernate	使系统休眠。
7	systemctl hybrid-sleep	使系统待机且处于休眠状态。

7.2 OpenSSH

7.2.1 SSH 介绍

SSH（Secure Shell）是一个基于客户端-服务端架构的协议，用于实现系统间的安全通信，允许用户安全地远程登录到服务端主机。与 FTP 或 Telnet 等传统的远程通信协议不同，SSH 加密了会话数据，防止未授权的访问者通过链接窃取未加密的密码。SSH 被设计为替代像 Telnet 和 rsh 这样的旧有终端应用程序，这些应用程序安全性较低。另一个与 SSH 相关的工具是 scp，它用于取代 rcp 等旧文件传输程序。由于这些老旧程序不对客户端和服务端间的密码进行加密，因此建议优先使用 SSH，以降低客户端和远程主机的安全风险。

7.2.2 配置 OpenSSH

配置文件有两个不同的系列，一系列用于客户端程序（例如 ssh、scp 和 sftp），另外一系列用于服务端（sshd 守护进程）。

系统范围的 SSH 配置信息保存在/etc/ssh/目录下，详见表 7-9 系统范围的配置文件。特定用户的 SSH 配置信息保存在~/.ssh/目录下（该目录在特定用户的家目录里），详见表 7-10 特定用户的配置文件。

表 7-9 系统范围的配置文件

序号	文件	描述
1	/etc/ssh/moduli	包含了 Diffie-Hellman 密钥交换需要使用的 Diffie-Hellman 组，Diffie-Hellman 密钥交换对于构建安全的传输层是关键的。当密钥在一个 SSH 会话的最初阶段被交换的时候，一个共享的密值被创建，该密值无法由任 何单独的一方所确定。这个密值随后被用来提供主机认证。
2	/etc/ssh/ssh_config	默认的 SSH 客户端配置文件。注意，如果~/.ssh/config 存在的话，该文件的配置将被~/.ssh/config 覆盖。
3	/etc/ssh/sshd_config	sshd 守护进程的配置文件。
4	/etc/ssh/ssh_host_ecdsa_key	sshd 守护进程所使用的 ECDSA 私钥。
5	/etc/ssh/ssh_host_ecdsa_key.pub	sshd 守护进程所使用的 ECDSA 公钥。
6	/etc/ssh/ssh_host_ed25518_key	SSH 协议版本 1 的 sshd 守护进程所使用的 RSA 私钥。
7	/etc/ssh/ssh_host_ed25518_key.pub	SSH 协议版本 1 的 sshd 守护进程所使用的 RSA 公钥。
8	/etc/ssh/ssh_host_rsa_key	SSH 协议版本 2 的 sshd 守护进程所使用的 RSA 私钥。
9	/etc/ssh/ssh_host_rsa_key.pub	SSH 协议版本 2 的 sshd 守护进程所使用的 RSA 公钥。
10	/etc/pam.d/sshd	sshd 守护进程的 PAM 配置文件。
11	/etc/sysconfig/sshd	sshd 服务的配置文件。

表 7-10 特定用户的配置文件

序号	文件	描述
1	~/.ssh/authorized_keys	为服务端保留一个授权的公钥列表。当客户端连接 到服务端时，服务端通过检查保存在该文件中的它 的签名公钥来认证客户端。
2	~/.ssh/id_ecdsa	包含用户的 ECDSA 私钥。
3	~/.ssh/id_ecdsa.pub	用户的 ECDSA 公钥。
4	~/.ssh/id_rsa	SSH 协议版本 2 的 ssh 所使用的 RSA 私钥。
5	~/.ssh/id_rsa.pub	SSH 协议版本 2 的 ssh 所使用的 RSA 公钥。
6	~/.ssh/identity	SSH 协议版本 1 的 ssh 所使用的 RSA 私钥。
7	~/.ssh/identity.pub	SSH 协议版本 1 的 ssh 所使用的 RSA 公钥。
8	~/.ssh/known_hosts	包含用户访问的 SSH 服务端的主机密钥。该文件 对于确保 SSH 客户端正在连接正确的 SSH 服务端是很重要的。

7.2.3 启动 OpenSSH 服务端

需安装 openssh-server 包之后才能运行 OpenSSH 服务端。

要在当前会话中启动 sshd 守护进程，请以 root 用户在 shell 命令行提示符下输入以下命令：

```
[root@localhost ~]#systemctl start sshd.service
```

要在当前会话中停止运行 sshd 守护进程，请以 root 用户在 shell 命令行提示符下输入以下命令：

```
[root@localhost ~]#systemctl stop sshd.service
```

如果想在系统启动时自动启动守护进程，请以 root 用户在 shell 命令行提示符下输入以下命令：

```
[root@localhost ~]#systemctl enable sshd.service
Created symlink
/etc/systemd/system/multi-user.target.wants/sshd.service
/usr/lib/systemd/system/sshd.service.
```

⚠ **注意：为了让 SSH 真正发挥作用，应该禁止使用不安全的连接协议。否则，用户的密码可能在一个会话中使用 SSH 时被保护得很好，但是却在之后使用 Telnet 登录时被捕获了。需要禁用的服务包括 telnet、rsh、rlogin 和 vsftpd。**

7.2.4 使用基于密钥的认证

为了更进一步的提高系统安全，可以生成SSH密钥对，然后强制使用基于密钥的认证，并禁用密码认证。要这样做，请在vi或者nano等文本编辑器中打开 /etc/ssh/sshd_config 配置文件，并将PasswordAuthentication选项修改为如下内容：

```
PasswordAuthentication no
```

如果不是在一个新的默认安装的系统中进行操作，请检查配置文件确保没有设置 PubkeyAuthentication no 选项。如果是远程连接上的，而不是使用的控制台访问，建议在禁用密码认证之前先测试基于密钥的登录过程是否可用。

为了能够使用 ssh、scp 或者 sftp 从一个客户端机器连接到服务端，请按照 7.2.5 生成密钥对章节生成一个授权密钥对。注意，这些密钥必须针对每个用户分别生成。

重要说明：

如果以 root 的身份完成了步骤，那么只有 root 用户能够使用这些密钥。

⚠ **注意：重装系统时，若想保留之前生成的密钥对，请备份~/.ssh/目录。在重装后，将其复制回用户家目录。这一过程需要让系统上的所有用户执行，包括 root 用户。**

7.2.5 生成密钥对

以 user 用户为例，要生成 SSH 协议版本 2 的 RSA 密钥对，请按以下步骤操作：

（1）在 shell 命令行提示符下输入以下命令生成 RSA 密钥对：

```
[root@localhost ~]# ssh-keygen-t rsa
Generating public/private rsa key pair.
Enter file in which to save the key
(/home/user/.ssh/id_rsa):
```

（2）按回车键确认新创建的密钥的默认路径，即~/.ssh/id_rsa。

（3）输入一个口令，并且在提示确认的时候再次输入该口令。为了安全起见，请不要使用和登录账户相同的密码。

要生成 SSH 协议版本 2 的 ECDSA 密钥对，请按以下步骤操作：

（1）在 shell 命令行提示符下输入以下命令生成 ECDSA 密钥对：

```
[root@localhost ~]# ssh-keygen-t ecdsa
Generating public/private ecdsa key pair.
Enter file in which to save the key
(/home/user/.ssh/id_ecdsa):
```

（2）按回车键确认新创建的密钥的默认路径，即~/.ssh/id_ecdsa。

（3）输入一个口令，并且在提示确认的时候再次输入该口令。为了安全起见，请不要使用和登录账户相同的密码。

默认情况下，将~/.ssh/目录的权限设置为 rwx------或者以八进制标注表示的 700。这是为了确保只有对应的用户 USER 能够查看其内容。如果有必要，可以使用以下命令来进行确认：

```
[root@localhost ~]# ls-ld ~/.ssh
drwx------. 2 user user 54 Nov 25 16:56 /home/user/.ssh/
```

使用以下格式的命令，将公钥复制到一台远程机器上：

```
[root@localhost ~]# ssh-copy-id user@hostname
```

如果公钥尚未安装的话，该命令会复制最近一次修改的~/.ssh/id*.pub 公钥。可选的，也可以指定公钥的文件名：

```
[root@localhost ~]# ssh-copy-id-i ~/.ssh/id_ecdsa.pub user@hostname
```

该命令会将~/.ssh/id_ecdsa.pub 的内容复制到用户想连接的机器的~/.ssh/authorized_keys 文件中。如果 authorized_keys 文件已经存在了，密钥将会追加到该文件的末尾。

⚠ **注意：请妥善存储私钥，绝不要把它发给其他任何人，这一点是非常重要的。**

7.2.6 OpenSSH 客户端

安装 openssh-clients 包后，才能从客户端机器连接到一个 OpenSSH 服务端。

7.2.7 使用 ssh 工具

ssh 工具可以让用户登录到一台远程机器上，并在上面执行命令。它是对 rlogin、rsh 和 telnet 程序的一个安全替换。和 telnet 命令相似，使用以下命令登录到一台远程机器上：

```
[root@localhost ~]# ssh hostname
```

例如，要登录到一台名为 example.com 的远程主机，可以在 shell 命令行提示符下输入以下命令：

```
[root@localhost ~]# ssh example.com
```

该命令将会以用户当前正在使用的本地机器的用户名登录。如果用户想指定一个不同的用户名，请使用以下命令：

```
[root@localhost ~]# ssh username@hostname
```

例如，以 user 登录到 example.com，请输入以下命令：

```
[root@localhost ~]# ssh user@example.com
```

第一次连接时，将会看到和如下内容相似的信息：

```
The authenticity of host 'example.com' can't be
 established.
 ECDSA key fingerprint is
 SHA256:Ixy64icRYc/h7XSOvUVywS7t7ThtmOsPT1s07wDD5P8.
 Are you sure you want to continue connecting
 (yes/no/[fingerprint])?
```

在回答对话框中的问题之前，用户应该始终检查指印是否正确。用户可以询问服务端的管理以确认密钥是正确的。这应该以一种安全的、事先约定好的方式进行。如果用户可以使用服务端的主机密钥，可以使用以下 ssh-keygen 命令来检查指印：

```
[root@localhost ~]# ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key.pub
256
SHA256:b0gGbI+Xk/l+Ve76j3mqpYSty0n3gR9QiIsd+8oV3GI
no comment (ECDSA)
```

输入 yes 接收密钥并确认连接。将会看到一个有关服务端已经被添加到已知的主机列表中的通告，以及一个输入密码的提示：

```
Warning: Permanently added 'example.com'
(ECDSA) to the list of known hosts.
 user@ example.com's password:
```

⚠ 注意：如果 SSH 服务端的主机密钥改变了，客户端将会通知用户连接不能继续，需将服务端的主机密钥从~/.ssh/known_hosts 文件中删除后重新连接。在进行此操作之前，请联系系统管理员，验证服务端没有受到攻击。

要从~/.ssh/known_hosts 文件中删除一个密钥，可以使用如下命令：

```
[root@localhost ~]#ssh-keygen-R example.com
```

在输入密码之后，用户将会进入远程主机的 shell 命令行提示符下。可选地，ssh 程序可以用来在远程主机上执行一条命令，而不用登录到 shell 命令行提示符下：

```
[root@localhost ~]#ssh username@hostname command
```

例如，/etc/kylin-release 文件提供有关操作系统版本的信息。要查看 example.com 上该文件的内容，输入：

```
[root@localhost ~]#ssh user@example.com cat /etc/cqos-release
Password:
```

在用户输入正确的密码之后，将会显示远程主机的操作系统版本信息，然后将返回到本地的 shell 命令行提示符下。

7.2.8 使用 scp 工具

scp 可以用来在主机之间通过一个安全加密的链接传输文件。

要传输一个本地文件到远程系统中，可以使用如下形式的命令：

```
[root@localhost ~]#scp localfile username@hostname:remotefile
```

例如，以 user 用户，如果想将 taglist.vim 传输到名为 example.com 的远程主机用户家目录上，可以在 shell 命令行提示符下输入以下命令：

```
[root@localhost ~]#scp taglist.vim user@example.com:~
```

一次可以指定多个文件。要传输.vim/plugin/目录下的文件和目录到远程主机 example.com 的相同目录下，可以输入以下命令：

```
[root@localhost ~]#scp -r .vim/plugin/*
 user@example.com:.vim/plugin/
```

要将一个远程的文件传输到本地系统上，可以使用以下语法：

```
[root@localhost ~]#scp username@hostname:remotefile localfile
```

例如，要从远程主机上下载.vimrc 配置文件，可以输入：

```
[root@localhost ~]#scp user@example.com:.vimrc .vimrc
```

7.2.9 使用 sftp 工具

sftp 工具可以用来打开一个安全的、交互式的 FTP 会话。在设计上，它类似于 ftp，不同之处在于 sftp 使用了一个安全的加密链接。

要连接到远程系统中，可以使用如下形式的命令：

```
[root@localhost ~]#sftp username@hostname
```

例如，以 user 用户为例，要使用 user 用户登录到名为 example.com 的远程主机上，可以输入：

```
[root@localhost ~]#sftp user@example.com
 Password:
 Connected to example.com.
 sftp>
```

当用户输入正确的密码之后，将会看到一个 sftp 的命令行提示符。sftp 工具可以使用一系列和 ftp 类似的命令如表 7-11 所示：

表 7-11 sftp 工具命令

序号	命令	描述
1	ls [directory]	列出一个远程目录的内容。如果没有提供任何目录名称，默认使用当前的工作目录。
2	cd directory	切换远程工作目录到指定的目录下。
3	mkdir directory	创建一个远程目录。
4	rmdir path	删除一个远程目录。
5	put localfile [remotefile]	将本地文件传输到远程主机上。
6	get remotefile [localfile]	将远程主机上的文件下载到本地主机上。

7.2.10 X11 转发

1) 配置 X11 转发通常涉及安装必要的软件包、配置 SSH 服务器以允许 X11 转发，并在客户端启用该功能。

安装 X11 转发所需的软件包：

```
[root@localhost ~]# dnf install xorg-x11-xauth xorg-x11-apps -y
```

2) 配置 SSH 服务器以允许 X11 转发

编辑 SSH 服务器的配置文件/etc/ssh/sshd_config，找到并修改以下行（如果它们不存在，则添加它们）：

```
[root@localhost ~]# vim /etc/ssh/sshd_config
```

找到 X11Forwarding 行，并将其设置为 yes：

```
X11Forwarding yes
```

作用：这一行告诉 SSH 服务器允许 X11 转发。

保存并关闭文件，然后重启 SSH 服务以应用更改：

```
[root@localhost ~]# systemctl restart sshd
```

3) 在 SSH 客户端启用 X11 转发

在客户端计算机上，使用-X（或-Y，但-X 更安全）选项连接服务器：

```
[root@localhost ~]# ssh -X username@server_ip_address
```

作用：-X 选项告诉 SSH 客户端启用 X11 转发。username 是你的服务器用户名，server_ip_address 是服务器的 IP 地址。

4) 测试 X11 转发

一旦成功连接到服务器，你可以尝试运行一个图形界面应用程序来测试 X11 转发是否工作。例如，运行 xclock：

```
[root@localhost ~]# xclock
```

作用：xclock 是一个简单的图形时钟应用程序，如果 X11 转发配置正确，它应该在你的本地计算机上显示一个时钟窗口。

7.2.11 端口转发

SSH 可以通过端口转发安全加固其他不安全的 TCP/IP 协议。在使用这一技术时，SSH 服务端成为 SSH 客户端的一个加密通道。

端口转发的工作原理是将客户端的一个本地端口映射到服务端的一个远程端口上。SSH 可以从服务端将任意端口映射到客户端的任意端口上。这一技术并不需要端口号互相匹配。

重要说明：

要设置端口转发监听 1024 以下的端口，需要 root 级别的访问权限。要创建一个监听 localhost 上的连接的 TCP/IP 端口转发通道，可以使用以下形式的命令：

```
[root@localhost ~]# ssh -L local-port:remote-hostname:remote-port
username@hostname
```

例如，要使用 POP3 通过一个加密连接检查名为 mail.example.com 的服务器上的邮件，可以使用以下命令：

```
[root@localhost ~]#ssh -L 1100:mail.example.com:110 mail.example.com
```

一旦客户端机器和邮件服务器之间的端口转发通道准备就绪后，就可以使用一个 POP3 邮件客户端在 localhost 上使用 1100 端口来检查新邮件了。任何在客户端系统上发往 1100 端口的请求，都将被安全地传输到 mail.example.com 服务器。

如果 mail.example.com 没有运行 SSH 服务端，但是同一网络中的另一台机器运行了 SSH 服务端，那么 SSH 仍然可以用来对连接进行安全加固。当然，使用的命令会略有不同：

```
[root@localhost ~]#ssh -L 1100:mail.example.com:110 other.example.com
```

在这一示例中，从客户端机器的 1100 端口发出的 POP3 请求，将通过 22 端口上的 SSH 连接被转发到 SSH 服务端 other.example.com。然后 other.example.com 连接到 mail.example.com 上的 110 端口来检查新邮件。

⚠注意：在使用这一技术时，只有客户端和 other.example.com 服务端之间的连接是安全的。端口转发也可以用来通过网络防火墙安全地获取信息。如果防火墙配置为允许放行使用标准端口（即 22 端口）的 SSH 数据流，但是阻塞了对其他端口的访问，那么要在两台主机之间使用被阻塞端口建立连接仍然是可能的，只要将它们之间的通信通过一条建立好的 SSH 连接进行重定向即可。

重要说明：

以这种方式来使用端口转发技术对连接进行转发，将允许客户端系统上的任何用户都能连接到相应的服务上。如果客户端系统被入侵，攻击者也同样能够访问转发的服务。担心端口转发的系统管理员，可以在服务端将/etc/ssh/sshd_config 文件中AllowTCPForwarding 选项设置为 No，并重启 sshd 服务，来禁用端口转发功能。

7.3 OpenVPN

7.3.1 OpenVPN 介绍

OpenVPN 是一种开源的虚拟专用网络（VPN）解决方案，它基于OpenSSL 库，使用SSL/TLS 协议来提供安全的网络连接。OpenVPN 可以在不同的操作系统平台上运行，包括Linux、Unix、Mac OS 和Windows。它通过创建加密隧道来保护网络数据的隐私和安全，使得用户可以在不安全的公共网络上安全地传输数据。

7.3.2 OpenVPN 服务端搭建

在操作前请确保已经安装 OpenVPN 以及 OpenSSL，再进行下列操作。

（1）安装 OpenVPN

```
[root@localhost ~]# dnf install openvpn -y
```

（2）创建 CA 文件夹

创建一个目录用于存放 CA 的文件和配置：

```
[root@localhost ~]# mkdir -p /etc/myCA/{certs,crl,newcerts,private}
[root@localhost ~]# cd /etc/openvpn/myCA
[root@localhost myCA]# touch index.txt
[root@localhost myCA]# echo 1000 > serial
```

创建 CA 的私钥：

```
[root@localhost myCA]#openssl genpkey -algorithm RSA -out private/ca.key -pkeyopt
rsa_keygen_bits:2048
```

使用私钥创建自签名 CA 证书：

```
[root@localhost myCA]# openssl req -key private/ca.key -new -x509 -days 3650 -out certs/ca.crt
```

在所创建的 CA 目录下创建一个 OpenSSL 配置文件（如 openssl.cnf），

```
[root@localhost myCA]# vim openssl.cnf
```

内容示例：

```
[ ca ]
default_ca = CA_default


[ CA_default ]
dir = ./                        # top dir
database = $dir/index.txt    # database index file
```

```
new_certs_dir = $dir/newcerts    # new certs dir
certificate = $dir/certs/ca.crt    # CA certificate
serial = $dir/serial          # serial no. file
private_key = $dir/private/ca.key # CA private key
default_md = sha256           # Set default message digest algorithm
policy = policy_any           # Set Policy
default_days = 365            # Certificate validity period

[ policy_any ]
countryName = supplied
stateOrProvinceName = supplied
localityName = supplied
organizationName = supplied
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = CN
organizationName = Organization Name
organizationName_default = My CA
commonName = Common Name
commonName_default = My CA
```

可以为需要签名的证书生成 CSR，并根据提示输入相关信息：

```
[root@localhost myCA]# openssl req -new -newkey rsa:2048 -keyout private/server.key -out server.csr
```

使用 CA 签署 CSR，生成证书：

```
[root@localhost myCA]# openssl ca -in server.csr -out certs/server.crt -config openssl.cnf
```

查看生成的证书信息：

```
[root@localhost myCA]# openssl x509 -in certs/server.crt -text -noout
```

成功则会显示证书信息。

（3）为 openvpn 服务器颁发证书

首先生成一个私钥（假设私钥文件名为 server.key）：

```
[root@localhost myCA]# openssl genpkey -algorithm RSA -out server.key -pkeyopt
rsa_keygen_bits:2048
```

接下来生成证书签名请求（CSR），在此示例中，CSR 文件名为 server.csr：

```
[root@localhost myCA]# openssl req -new -key server.key -out server.csr -subj "/CN=server"
```

/CN=server 是设置的公用名称（Common Name），用户可以根据需要更改。

使用以下命令查看生成的 CSR 的内容，确保信息正确：

```
[root@localhost myCA]# openssl req -in server.csr -text -noout
```

签署 CSR 并生成服务器证书（请使用正确的文件路径代替/path）：

```
[root@localhost myCA]# openssl x509 -req -in server.csr -CA certs/ca.crt -CAkey
 private/ca.key -CAcreateserial -out certs/server.crt -days 365 -sha256
```

使用以下命令查看生成的证书内容：

```
[root@localhost myCA]# openssl x509 -in certs/server.crt -text -noout
```

确保已经有了以下文件：

CA 证书（ca.crt），CA 私钥（ca.key），服务器证书申请文件（server.csr）。

使用以下命令来签署服务器证书并生成最终的服务器证书（假设输出的证书文件名为 server.crt）：

```
[root@localhost myCA]# openssl x509 -req -in server.csr -CA certs/ca.crt -CAkey private/ca.key -
CAcreateserial -out certs/server.crt -days 365 -sha256
```

生成证书后，使用以下命令查看证书的详细信息，确保它包含正确的信息：

```
[root@localhost myCA]# openssl x509 -in certs/server.crt -text -noout
```

在某些情况下，用户可能需要验证证书链。可以使用以下命令进行验证：

```
[root@localhost myCA]# openssl verify -CAfile certs/ca.crt certs/server.crt
```

使用以下命令生成 Diffie-Hellman 参数文件（假设生成的文件名为 dh2048.pem，2048 表示密钥的位数）：

```
[root@localhost myCA]# openssl dhparam -out dh2048.pem 2048
```

这个命令会生成一个 2048 位的 Diffie-Hellman 参数，文件 dh2048.pem 中会包含这些参数。

使用以下命令查看生成的 Diffie-Hellman 参数的内容：

```
[root@localhost myCA]# openssl dhparam -in dh2048.pem -text -noout
```

生成一个新的 ta.key 文件：

```
[root@localhost myCA]# openvpn --genkey secret ta.key
```

（4）准备 OpenVPN 服务器配置文件

编辑配置文件（可使用 rpm -ql openvpn 查看配置文件路径）：

```
[root@localhost myCA]# vim /etc/openvpn/server/server.conf
```

修改以下内容，利用正确文件路径替换/path

```
ca /etc/openvpn/myCA/certs/ca.crt
cert /etc/openvpn/myCA/certs/server.crt
key /etc/openvpn/myCA/server.key
dh /etc/openvpn/myCA/dh2048.pem
tls-auth /etc/openvpn/myCA/ta.key 0
```

创建日志所在文件夹

```
[root@localhost ~]# mkdir -p /var/log/openvpn
[root@localhost ~]# chown openvpn.openvpn /var/log/openvpn/
```

开启网卡转发功能

```
[root@localhost ~]# echo net.ipv4.ip_forward =1 >> /etc/sysctl.conf
[root@localhost ~]# sysctl -p
```

使用 openvpn，开启隧道默认的网段就是配置文件里面配置的网段，然后通过 iptables 配置以后将这个网段的地址指向内网地址

```
[root@localhost ~]# echo "iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -j MASQUERADE" >>
/etc/rc.d/rc.local
[root@localhost ~]# bash /etc/rc.d/rc.local
[root@localhost ~]# iptables -vnL -t nat
```

（5）启动 OpenVPN 服务器

```
[root@localhost ~]# systemctl daemon-reload
[root@localhost ~]# systemctl enable --now openvpn-server@server
```

7.3.3 OpenVPN 常用指令

Systemd 为用户提供了 systemctl 的系列命令，其中包括了一些对 OpenVPN 的操作，具体常用操作命令如表 7-12 所示：

表 7-12 OpenVPN 常用命令

序号	命令	描述
1	systemctl start openvpn-server@server	启动服务（如果服务尚未运行）。
2	systemctl stop openvpn-server@server	停止服务（如果服务正在运行）。
3	systemctl reload openvpn-server@server	重新加载服务配置文件，而不重启服务。适用于当用户只更改了配置文件而无需完全重启服务时。
4	systemctl status openvpn-server@server	检查服务的当前状态（是否正在运行）。
5	systemctl enable openvpn-server@server	配置服务在系统启动时自动启动。
6	systemctl disable openvpn-server@server	禁用服务在系统启动时自动启动。
7	systemctl restart openvpn-server@server	重新启动服务，首先停止服务然后再启动它。
8	systemctl try-restart openvpn-server@server	如果服务已经运行，则重新启动它；如果服务没有运行，则什么也不做。

7.4 Apache HTTP Server

7.4.1 Apache HTTP Server 介绍

Apache HTTP Server（简称 httpd）是一个开放源代码的网页服务器，可以在大多数电脑操作系统中运行，具有的跨平台性和安全性，它能够帮助网站管理员托管和服务网站内容，它通过 HTTP 协议将 Web 服务器上的内容传递给浏览器，支持静态和动态网页内容。

7.4.2 Apache HTTP Server 安装

可以通过下面的指令来安装 httpd

```
[root@localhost ~]# dnf install -y httpd
```

然后启动 Apache HTTP

```
[root@localhost ~]# systemctl start httpd.service
```

执行以下的命令可以使得 httpd 开机自启动

```
[root@localhost ~]# systemctl enable httpd.service
```

执行以下命令，可停止 httpd 服务

```
[root@localhost ~]# systemctl stop httpd.service
```

执行以下命令，可以避免 httpd 自启动：

```
[root@localhost ~]# systemctl disable httpd.service
```

重启服务有三种方式，第一种是完全重启服务，要停止运行着的 httpd 服务，然后再重启它。命令如下：

```
[root@localhost ~]# systemctl restart httpd.service
```

第二种是让正在运行的 httpd 服务去重新加载它的配置文件，当前正在处理的请求都会中断。

```
[root@localhost ~]# systemctl reload httpd.service
```

第三种也是让正在运行的 httpd 服务区重新加载配置文件，但是当前正在处理的请求，不会中断，会延续旧的配置继续处理。

```
[root@localhost ~]# apachectl graceful
```

可以通过以下两个命令来查看 httpd 的运行状态

```
[root@localhost ~]# systemctl is-active httpd.service
[root@localhost ~]# systemctl status httpd.service
```

7.4.3 Apache HTTP Server 配置

Apache HTTP Server 的主配置文件是/etc/httpd/conf/httpd.conf。可以编辑此文件来调整服务器的基本设置，例如更改默认的端口、服务器名、文档根目录等。比如可以修改配置文件中的 Listen 来修改监听端口：

```
Listen 8000
```

注：每次更改配置文件，都要重启 httpd。

可以使用 httpd 的虚拟主机特性，来实现基于不同 ip、主机号和端口号来提供不同的网络信息服务。虚拟主机的配置通常在/etc/httpd/conf.d/目录下，可以在该目录下写一个配置文件，来进行虚拟机配置，在配置文件中，添加以下虚拟主机配置

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    DocumentRoot /var/example.com
    ServerName example.com
    ErrorLog /var/log/httpd/example.com-error.log
    CustomLog /var/log/httpd/example.com-access.log combined
</VirtualHost>
```

需要配置防火墙来确保防火墙允许 HTTP 和 HTTPS 流量

```
[root@localhost ~]# firewall-cmd --permanent --add-service=http
[root@localhost ~]# firewall-cmd --permanent --add-service=https
[root@localhost ~]# firewall-cmd --reload
```

7.5 OpenLDAP

7.5.1 OpenLDAP 介绍

OpenLDAP 是轻型目录访问协议（Lightweight Directory Access Protocol，LDAP）的自由和开源的实现，LDAP（轻量级目录访问协议，Lightweight Directory Access Protocol）是一种用于访问分布式目录服务的网络协议。它提供了一种标准化的方法来查询和操作目录中的信息，这些信息通常以树状结构组织，用于存储用户、组织和其他实体的数据。

7.5.2 OpenLDAP 术语

OpenLDAP 术语如表 7-13 所示：

表 7-13 OpenLDAP 术语

序号	术语	介绍
1	DN(Distinguished Name)	唯一标识目录中的条目，包含条目在树形结构中的完整路径。例如：cn=John Doe,ou=users,dc=example,dc=com
2	RDN(Relative Distinguished Name)	条目在其父条目下的唯一标识，通常是 DN 的一部分。例如，cn=John Doe 就是一个 RDN。
3	Base DN	查询或操作的起始位置，通常是整个目录树的根。例如：dc=example,dc=com
4	LDAP Entries	目录中的条目，通常由多个属性组成，每个条目都必须有一个 DN。
5	Attributes	条目的属性，包含有关条目的信息。例如，cn（常用名）、sn（姓）、mail（电子邮件地址）等。
6	Object Class	定义条目可以拥有的属性的集合。条目必须符合至少一个对象类的要求。例如，inetOrgPerson、organizationalUnit。

序号	术语	介绍
7	Search Filter	用于查询条目的条件，常用的过滤器包括：objectClass=*：匹配所有对象类 cn=name：匹配常用名为 name 的条目 mail=*@example.com：匹配所有以@example.com 结尾的邮件地址
8	Scope	查询的范围，主要有三种： Base：仅返回基础条目； One：返回直接子条目； Sub：返回子树中的所有条目。
9	LDAP URLs	用于指向 LDAP 目录的标准格式。例 ldap://hostname:port/dc=example,dc=com
10	Bind	用于连接到 LDAP 服务器的过程。可以是匿名绑定或使用凭证（如用户名和密码）的绑定。
11	ACL (Access Control List)	用于定义谁可以访问和修改 LDAP 条目的权限。
12	Replication	在多个 LDAP 服务器之间同步数据的过程。

7.5.3 OpenLDAP 特性

OpenLDAP 完全支持 LDAPv3，这一版本引入了多项安全改进，以提高协议的安全性和效率。特别是，它增强了对简单认证和安全层（SASL）、传输层安全（TLS）和安全套接层（SSL）等标准协议的支持，确保数据的安全传输。

OpenLDAP 利用进程间通信（IPC）机制，提升了安全性，通过消除了网络通信的需求，减少了潜在的安全风险，从而确保目录服务在本地的安全操作。

OpenLDAP 具备对 IPv6 网络协议的支持，使其能够适应现代网络环境中的新需求，确保在多种网络配置下的可用性和兼容性。

OpenLDAP 支持 LDAP 数据交换格式的 LDIF 版本 1，使用户能够便捷地导入和导出目录信息，简化数据迁移和备份过程。

OpenLDAP 的 C 接口经过更新和优化，增强了程序开发的灵活性和可用性，为开发者提供了更好的工具和库，以构建与 LDAP 交互的应用程序。

7.5.4 OpenLDAP 服务器安装

OpenLDAP 服务器包及其描述如表 7-14 所示：

表 7-14 OpenLDAP 服务器包

序号	包	描述
1	openldap	包含运行 OpenLDAP 服务器和客户端应用所需的基本库。提供了 LDAP 协议的实现，使得应用程序可以与 LDAP 目录进行交互。
2	openldap-clients	包含用于访问和修改 LDAP 服务器的命令行工具，如 ldapsearch、ldapadd、ldapmodify 等。这些工具能够管理 LDAP 数据库和进行目录查询。
3	openldap-servers	包含运行 LDAP 服务器所需的服务和配置程序，主要包括 slapd（LDAP Daemon）。这是 OpenLDAP 的核心组件，负责处理客户端请求和管理目录数据。
4	openldap-devel	开发包，包含了编译和链接 OpenLDAP 应用程序所需的头文件和库。

附加包及其描述如表 7-15 所示：

表 7-15 OpenLDAP 服务器附加包

序号	包	描述
1	nss-pam-ldapd	包含 nslcd，用于通过 LDAP 查询用户和组信息，使系统能够在身份验证和信息查询时使用 LDAP 目录。
2	mod_ldap	Apache HTTP 服务器的 LDAP 验证模块，允许使用 LDAP 目录进行用户凭证验证，实现基于 LDAP 的访问控制。

可以通过在 root 用户下使用 dnf 安装所需包，例如

```
[root@localhost ~]# dnf install openldap openldap-clients openldap-servers
```

7.5.5 OpenLDAP 服务器配置

OpenLDAP 的配置文件在目录/etc/openldap 下。该目录中重要文件和目录的描述如表 7-16 所示：

表 7-16 OpenLDAP 配置文件描述

序号	文件	描述
1	ldap.conf	是客户端应用（如 ldapadd、ldapsearch 和其他使用 OpenLDAP 库函数的应用）的配置文件，包含连接到 LDAP 服务器所需的信息。
2	slapd.d	是 OpenLDAP 服务器 slapd 的配置目录，包含动态加载的配置文件，用于管理 LDAP 服务器的设置和架构。

LDAP 服务器的全局配置存储在/etc/openldap/slapd.d/cn=config.ldif 文件中。可以使用 olcAllows 指令来指定可以使用的特性。特性及其描述如表 7-17 所示：

表 7-17 特性及描述

序号	特性	描述
1	bind_v2	允许使用 LDAP v2 版本的绑定请求（默认设置）
2	bind_anon_cred	允许在 DN 为空时进行匿名绑定。
3	bind_anon_dn	允许在 DN 非空时进行匿名绑定。
4	update_anon	允许匿名升级操作。
5	proxy_authz_anon	允许匿名代理控制。

也可以使用 olcDisallows 指令用于定义不允许使用的特性，如表 7-18 所示：

表 7-18 特性及描述

序号	特性	描述
1	bind_anon	禁止接收匿名绑定请求。
2	bind_simple	禁止简单的绑定验证机制。
3	tls_2_anon	在接收到 STARTTLS 命令时禁止增强匿名会话。
4	tls_authc	在已验证后禁止使用 STARTTLS 命令。

还有一些其他指令如表 7-19 所示：

表 7-19 特性及描述

序号	指令	描述
1	olcConnMaxPending	用于设置匿名会话的最大请求等待数量。
2	olcConnMaxPendingAuth	允许指定已验证会话的最大请求等待数量（默认值是 1000）。
3	olcIdleTimeout	允许设置关闭空闲连接时的等待时间(默认值是 0,表示不使用)。
4	olcLogFile	指定日志信息的记录文件（日志信息默认使用错误标准输入格式）。
5	olcReferral	用于指定一个服务器的 URL，以处理特定请求(默认情况下不启用）。
6	olcWriteTimeout	允许设置关闭未完成写请求的连接时的最大等待时间（默认值为 0，表示不限制等待时间）。

7.5.6 建立安全连接

OpenLDAP 客户端和服务器端可以通过使用 Transport Layer Security （TLS）框架来确保通信安全。TLS 是一种加密协议，用于在网络通信中提供加密和数据完整性保护。为建立 TLS/SSL 安全连接，服务器端至少需要配置 CA 证书、服务器证书和私钥。客户端则需要配置包含受信任的 CA 证书的文件，以确保能够连接到安全的服务器。通过正确配置，OpenLDAP 客户端和服务器端能够通过 TLS 实现安全的加密连接，确保通信的机密性和完整性。

首先确保 OpenLDAP 服务器和客户端工具安装完毕

1) 生成自签名证书

创建证书存放目录并设置权限

```
[root@localhost ~]# mkdir -p /etc/openldap/certs
[root@localhost ~]# chown ldap:ldap /etc/openldap/certs
[root@localhost ~]# chmod 700 /etc/openldap/certs
```

2) 生成证书和私钥

注意：生成证书过程 Common Name (e.g. server FQDN or YOUR name) []: localhost（实际主机名）

```
[root@localhost ~]# openssl req -new -x509 -nodes -out /etc/openldap/certs/slapd.crt -keyout
/etc/openldap/certs/slapd.k
```

3) 设置权限

```
[root@localhost~]#chown ldap:ldap /etc/openldap/certs/slapd.crt /etc/openldap/certs/slapd.key
[root@localhost ~]# chmod 600 /etc/openldap/certs/slapd.key
[root@localhost ~]# chmod 644 /etc/openldap/certs/slapd.crt
```

4) 应用 TLS 配置，创建配置文件 enable-tls.ldif，内容如下

```
[root@localhost ~]# cd  /etc/openldap
[root@localhost openldap]# vim enable-tls.ldif
dn: cn=config
```

```
changetype: modify
replace: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/certs/slapd.crt
-
replace: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/certs/slapd.key
-
replace: olcTLSCACertificatePath
olcTLSCACertificatePath: /etc/openldap/certs
```

5) 通过 ldapmodify 将配置导入

```
[root@localhost ~]# systemctl start slapd.service
[root@localhost ~]# ldapmodify -Y EXTERNAL -H ldapi:/// -f enable-tls.ldif
```

```
重启 OpenLDAP 使配置生效
[root@localhost ~]# systemctl restart slapd.service
```

6) 添加自签名证书到信任存储

```
复制服务器证书到系统信任存储路径
[root@localhost ~]# cp /etc/openldap/certs/slapd.crt /etc/pki/ca-trust/source/anchors/
```

```
更新系统信任
[root@localhost ~]# update-ca-trust
```

7) 设置 OpenLDAP 的管理员密码

```
[root@localhost ~]# slappasswd -s  [PASSWORD]
```

设置完会生成一个{SSHA}gmyq5L5Y1RlHcXWtSRCScBPbyIyLepeW 类似于这样的信息，记住这个信息

(1) vim /etc/openldap/slapd.d/cn\=config/olcDatabase\=\{2\}mdb.ldif

在该配置文件的 olcRootDN 一行中将 cn=Manager 修改为 cn=admin，在最后一行中添加 olcRootPW:{SSHA}gmyq5L5Y1RlHcXWtSRCScBPbyIyLepeW（设置密码时的信息）

(2)修改 /etc/openldap/slapd.d/cn\=config/olcDatabase\=\{1\}monitor.ldif，将 olcAccess 行中的 dn.base 中的 cn=Manager 修改为 cn=admin

(3)注意：（cn=admin 中的 admin 表示 OpenLDAP 管理员的用户名，dc 为 ldap 的服务器域名，olcRootPW 表示 OpenLDAP 管理员的密码）

通过 slaptest -u 验证 LDAP 的基本配置（出现 succeeded 就表示成功了）

```
重启 OpenLDAP 使配置生效
[root@localhost ~]# systemctl restart slapd
```

8) 创建 LDAP 数据库的条目结构

```
[root@localhost ~]#    cd /etc/openldap/
[root@localhost openldap]# vim base.ldif
dn: dc=my-domain,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: My Domain Organization
dc: my-domain
```

9) ldapadd 命令添加到 ldap 数据库中

```
[root@localhost ~]# ldapadd -x -H ldaps://localhost -D "cn=admin,dc=my-domain,dc=com" -W -f /etc/openldap/base.ldif
```

10) 测试

当使用 LDAP 客户端命令（如 ldapsearch）时，通过 ldaps://来连接 SSL，默认端口为 636。

```
[root@localhost ~]# ldapsearch -H ldaps://localhost -b "dc=my-domain,dc=com" -D "cn=admin,dc=my-domain,dc=com" -W
```

7.6 Samba

7.6.1 Samba 介绍

Samba 是一个开放源代码的套件，允许 Windows 和 Linux/Unix 之间的文件、打印机、用户权限等资源共享。它实现了 SMB/CIFS（Server Message Block/Common Internet File System）协议，允许 Linux 和其他 Unix 系统像 Windows 一样提供文件和打印共享服务。

Samba 通常由三个主要的守护进程组成，分别是 smbd、nmbd 和 winbindd。每个守护进程都有不同的职责和功能：

smbd 守护进程负责处理文件和打印服务，管理客户端对共享文件的读写请求、用户认证、权限控制，并提供对共享打印机的访问，确保 Linux 和 Windows 客户端能够无缝进行资源共享。smbd 是整个 Samba 服务的核心。

nmbd 守护进程负责 NetBIOS 名称解析和网络浏览服务。允许 Samba 服务器在 Windows 网络中被发现，并通过 NetBIOS 协议处理名称解析和工作组浏览器的功能，使得 Windows 客户端可以在资源管理器中查看 Samba 服务器及其共享资源。

winbindd 守护进程用于将 Samba 服务器与 Windows 域或 Active Directory 集成，允许 Linux 系统使用 Windows 域中的用户和组信息进行身份认证。它简化了跨平台的用户管理，使得域用户能够无缝地访问 Samba 提供的资源，无需在 Linux 上单独创建用户账户。

7.6.2 配置 Samba

（1）安装 Samba

```
[root@localhost ~]# dnf install samba
```

（2）修改配置文件

```
[root@localhost ~]# vim /etc/samba/smb.conf
#在文件中添加以下内容
[share]
comment = this is samba dir
path = /home/lighthouse
public = yes
valid users = root
force user = root
force group = root
read only = no
create mask = 0755
directory mask=0755
available = yes
writable = yes
browseable = yes
security=share
```

在修改配置文件之后，需要重启 samba 才能够生效

```
[root@localhost ~]# systemctl restart smb.service
```

samba 可以加密密码，可以创建一个使用加密密码的用户

```
[root@localhost ~]# smbpasswd -a username
```

7.6.3 使用 Samba

samba 开机自启动，命令如下

```
[root@localhost ~]# systemctl enable smb.service
```

启动 samba 命令如下

```
[root@localhost ~]# systemctl start smb.service
```

关闭 samba 命令如下

```
[root@localhost ~]# systemctl stop smb.service
```

condrestart（条件重启）仅在服务当前正在运行时才会重新启动 smb 服务。这个选项在脚本中非常有用，因为如果守护进程未运行，服务将不会被启动。

```
[root@localhost ~]# systemctl try-restart smb.service
```

要重新加载/etc/samba/smb.conf 配置文件，而无需完全重启服务，可以使用以下命令：

```
[root@localhost ~]# systemctl reload smb.service
```

7.6.4 Samba 安全模式

Samba 通过多种安全模式来控制用户访问共享资源的方式，每种模式都有不同的认证机制，适用于从简单的家庭网络到复杂的企业网络。了解这些安全模式的特性，可以帮助管理员根据需求选择最合适的方案，确保资源共享的安全性和灵活性。

Share（共享模式，已被弃用）：这种模式允许用户无需输入用户名和密码即可访问共享资源，适用于简单的家庭或小型局域网环境。在共享模式下，每个共享资源可以独立设置访问权限，这意味着用户访问某个共享资源时只需提供共享资源的密码，而不需要账户认证。然而，由于该模式缺乏对用户身份的严格控制，安全性较低，且无法针对不同用户设置细致的权限管理。因此，这种模式已经过时并被弃用，现代系统不推荐使用。

User（用户模式）：这是 Samba 中最常用的安全模式。在该模式下，每个用户都需要提供用户名和密码才能访问共享资源，系统会验证用户的凭据，确保只有授权用户能够访问共享文件或打印机资源。这种模式非常适合大多数工作组或局域网环境，因为它允许管理员为不同的用户设置不同的访问权限，支持更精细的控制。然而，这种模式要求管理员在 Samba 服务器上维护用户账户和密码数据库，这对大规模用户管理可能带来一些负担。

Domain（域模式）：域模式允许 Samba 服务器作为 Windows 域中的一部分工作，用户的身份验证由域控制器完成，而不需要在 Samba 服务器上单独创建账户。该模式非常适合已经部署了 Windows 域的企业网络环境，可以简化用户管理和认证过程。然而，域模式的缺点是依赖于外部的域控制器，如果域控制器不可用或出现问题，Samba 服务器可能无法完成用户的身份验证。

Server（服务器模式）：在该模式下，Samba 服务器会将用户认证请求转发给另一台 Samba 服务器或 Windows 服务器。这种模式的优点是，Samba 服务器无需自己处理认证，可以依赖远程服务器的用户数据库。然而，它的安全性较差，并且复杂度较高，因此不再被推荐使用。

7.7 FTP

7.7.1 介绍 FTP 和 vsftpd

FTP 是一种在互联网中进行文件传输的协议，基于客户端/服务器模式，默认使用 20、21 号端口，其中端口 20（数据端口）用于进行数据传输，端口 21（命令端口）用于接收客户端发出的相关 FTP 命令与参数。

FTP 协议有下面两种工作模式：

主动模式：FTP 服务器主动向客户端发起连接请求。

被动模式：FTP 服务器等待客户端发起连接请求（FTP 的默认工作模式）。

vsftpd（very secure ftp daemon，非常安全的 FTP 守护进程）是一款运行在 Linux 操作系统上的 FTP 服务程序，不仅完全开源而且免费，此外，还具有很高的安全性、传输速度，以及支持虚拟用户验证等其他 FTP 服务程序不具备的特点。

vsftpd 允许用户以三种认证模式登录到 FTP 服务器上：

匿名开放模式：是一种最不安全的认证模式，任何人都可以无需密码验证而直接登录到 FTP 服务器。

本地用户模式：是通过 Linux 系统本地的账户密码信息进行认证的模式，相较于匿名开放模式更安全，而且配置起来也很简单。但是如果被黑客破解了账户的信息，就可以畅通无阻地登录 FTP 服务器，从而完全控制整台服务器。

虚拟用户模式：是这三种模式中最安全的一种认证模式，它需要为 FTP 服务单独建立用户数据库文件，虚拟出用来进行口令验证的账户信息，而这些账户信息在服务器系统中实际上是不存在的，仅供 FTP 服务程序进行认证使用。这样，即使黑客破解了账户信息也无法登录服务器，从而有效降低了破坏范围和影响。

7.7.2 配置 FTP

安装 vsftpd 服务程序

```
[root@localhost ~]# dnf install vsftpd -y
```

添加防火墙配置

```
[root@localhost ~]# firewall-cmd --permanent --zone=public --add-service=ftp
[root@localhost ~]# firewall-cmd --reload
```

修改vsftpd服务程序的主配置文件（etc/vsftpd/vsftpd.conf）

表 7-1 中罗列了 vsftpd 服务程序主配置文件中常用的参数以及作用

表7-1 vsftpd 服务程序常用的参数以及作用

序号	参数	作用
1	anonymous_enable=[YES|NO]	是否允许匿名用户访问
2	anon_upload_enable=[YES|NO]	是否允许匿名用户上传文件
3	anon_umask=022	匿名用户上传文件的 umask 值
4	anon_root=/var/ftp	匿名用户的 FTP 根目录
5	anon_mkdir_write_enable=[YES|NO]	是否允许匿名用户创建目录
6	anon_other_write_enable=[YES|NO]	是否开放匿名用户的其他写入权限（包括重命名、删除等操作权限）
7	anon_max_rate=0	匿名用户的最大传输速率（字节/秒），0 为不限制
8	local_enable=[YES|NO]	是否允许本地用户登录 FTP

序号	参数	作用
9	local_umask=022	本地用户上传文件的 umask 值
10	local_root=/var/ftp	本地用户的 FTP 根目录
11	local_max_rate=0	本地用户最大传输速率（字节/秒），0 为不限制
12	write_enable=[YES|NO]	是否允许登录用户有写权限。
13	guest_enable=[YES|NO]	开启虚拟用户模式
14	guest_username=virtual	指定虚拟用户账户为 virtual
15	pam_service_name=vsftpd.vu	指定 PAM 文件为 vsftpd.vu
16	allow_writeable_chroot=[YES|NO]	允许对禁锢的 FTP 根目录执行写入操作，而且不拒绝用户的登录请求
17	dirmessage_enable=[YES|NO]	当用户进入某个目录时，是否显示该目录需要注意的内容。
18	xferlog_enable=[YES|NO]	是否记录使用者上传与下载文件的操作
19	connect_from_port_20=[YES|NO]	Port 模式进行数据传输是否使用端口 20
20	xferlog_std_format=[YES|NO]	传输日志文件是否以标准 xferlog 格式书写
21	listen=[YES|NO]	是否以独立运行的方式监听服务
22	listen_ipv6=[YES|NO]	是否侦听 IPv6 的 FTP 请求，listen 和 listen_ipv6 不能同时开启。
23	download_enable=[YES|NO]	是否允许下载文件
24	userlist_enable=[YES|NO]	是否支持/etc/vsftpd/user_list 文件内的账号登录控制

客户端上安装 FTP 工具，FTP 是 Linux 系统中以命令行界面的方式来管理 FTP 传输服务的客户端工具：

```
[root@localhost ~]# dnf install ftp
```

7.7.3 使用 FTP

1) 匿名开放模式

修改服务器/etc/vsftpd/vsftpd.conf 文件

```
[root@localhost ~]# vim /etc/vsftpd/vsftpd.conf
```

在文件中修改以下内容：

```
anonymous_enable=YES
anon_umask=022
anon_upload_enable=YES
anon_mkdir_write_enable=YES
anon_other_write_enable=YES
```

重启 vsftpd 服务程序，并加入开机启动项中：

```
[root@localhost ~]# systemctl restart vsftpd
[root@localhost ~]# systemctl enable vsftpd
```

客户端执行 ftp 命令连接到远程的 FTP 服务器。在 vsftpd 服务程序的匿名开放认证模式下，其账户统一为 anonymous，密码为空。而且在连接到 FTP 服务器后，默认访问的是/var/ftp 目录。在此处假设服务器的 IP 地址是 192.168.16.137。

```
# 记住账号是 anonymous，密码为空因此直接回车
[root@localhost ~]# ftp 192.168.16.137
```

登录之后，如果在客户端无法创建目录和文件，请修改服务器上/var/ftp 目录的权限

```
[root@localhost ~]# chmod o+w /var/ftp
```

2) 本地用户模式

不需要修改服务器/etc/vsftpd/vsftpd.conf 文件，CQOS24 默认开启本地用户模式。

重启 vsftpd 服务程序，并加入开机启动项中：

```
[root@localhost ~]# systemctl restart vsftpd
[root@localhost ~]# systemctl enable vsftpd
```

客户端执行 ftp 命令连接到远程的 FTP 服务器。注意在采用本地用户模式登录 FTP 服务器时，账号和密码都是本地用户的账号和密码，且登录成功默认访问的是该用户的家目录。此处假设服务器的 IP 地址是 192.168.16.137。

```
[root@localhost ~]# ftp 192.168.16.137
```

如果使用 root 用户登录失败，可以查看服务器上/etc/vsftpd/目录下的 ftpusers 和 user_list 文件，在这两份文件中的用户都不能登录 ftp 服务器，这是为了保证服务器的安全性而默认禁止了 root 管理员和大多数系统用户的登录行为，这样可以有效地避免黑客通过 FTP 服务对 root 管理员密码进行暴力破解。如果确认在生产环境中使用 root 管理员，只需在这两份文件中删除掉 root 用户名即可。

3) 虚拟用户模式

虚拟用户模式是这三种模式中最安全的一种认证模式，当然，因为安全性较之于前面两种模式有了提升，所以配置流程也会稍微复杂一些。

第一步，创建用于进行 FTP 认证的用户数据库文件 login.pag：

```
[root@localhost ~]# cd /etc/vsftpd/
[root@localhost vsftpd ]# gdbmtool login.pag store [userenam] [password]
#其中[userenam]是用户名，[password]是该用户的密码，如果想要创建其他的用户，只要对应着修改就可以
[root@localhost ~]# gdbmtool login list
#上面那条命令可以查看在 login.pag 数据库文件中的用户名和密码
```

第二步，创建 vsftpd 服务程序用于存储文件的根目录以及虚拟用户映射的系统本地用户。FTP 服务用于存储文件的根目录指的是，当虚拟用户登录后所访问的默认位置。由于 Linux 系统中的每一个文件都有所有者、所属组属性，例如使用虚拟账户“张三”新建了一个文件，但是系统中找不到账户“张三”，就会导致这个文件的权限出现错误。为此，需要再创建一个可以映射到虚拟用户的系统本地用户。为了方便管理 FTP 服务器上的数据，可以把这个系统本地用户的家目录设置为/var 目录（该目录用来存放经常发生改变的数据）。并且为了安全起见，我们将这个系统本地用户设置为不允许登录 FTP 服务器，这不会影响虚拟用户登录，而且还可以避免黑客通过这个系统本地用户进行登录。

```
[root@localhost ~]# useradd -d /home/ftproot -s /sbin/nologin virtual
[root@localhost ~]# chmod -Rf 755 /home/ftproot/
```

第三步，建立用于支持虚拟用户的 PAM 文件，其中 PAM 文件内的“db=”参数为使用 db_load 命令生成的账户密码数据库文件的路径，但不用写数据库文件的后缀：

```
[root@localhost ~]# vim /etc/pam.d/vsftpd.vu
```

在 vsftpd.vu 文件中写入以下内容，auth 和 account 都是 PAM 中的管理模块，required 表示这个模块是必需的：

```
auth required pam_userdb.so db=/etc/vsftpd/login
account required pam_userdb.so db=/etc/vsftpd/login
```

第四步，在 vsftpd 服务程序的主配置文件中通过 pam_service_name 参数将 PAM 认证文件的名称修改为 vsftpd.vu：

```
[root@localhost ~]# vim /etc/vsftpd/vsftpd.conf
pam_service_name=vsftpd.vu
```

并在文件中写入以下内容：

```
guest_enable=YES
guest_username=virtual
allow_writeable_chroot=YES
```

第五步，重启 vsftpd 服务并加入开机自启中：

```
[root@localhost ~]# systemctl restart vsftpd
[root@localhost ~]# systemctl enable vsftpd
```

第六步，客户端连接服务器，此处假设服务器的 IP 地址是 192.168.16.137：

```
[root@localhost ~]# ftp 192.168.16.137
```

接着输入虚拟用户的账号名“lisi”以及密码“123456”。

7.8 NTP

7.8.1 介绍 NTP

NTP（Network Time Protocol）是一种用于网络时间同步的协议。它旨在确保计算机和其他网络设备具有准确的时间，并通过与时间服务器进行通信来同步其时钟。如果没有 NTP 或其他类似的网络时间同步协议，那么随着时间的推移，由于计算机硬件时钟本身的精度限制（例如，计算机内部时钟晶振的频率可能存在一定偏差），系统时间会逐渐产生偏移。这可能导致在一段时间后，系统时间与实际时间相差较大。这对于一些依赖准确时间的应用程序，如数据库系统（数据库事务的时间戳准确性对于数据完整性和恢复很重要）、邮件服务器（邮件的收发时间排序可能错乱）、备份任务（备份计划可能基于准确的时间触发）等，时间的不准确会影响它们的正常运行。

NTP 服务器是提供时间服务的特定服务器，它们通过网络向客户端提供准确的时间信息。这些服务器通常与原子钟或其他高精度时间源同步，以确保提供高度准确的时间。

NTP 客户端会向 NTP 服务器发送时间请求消息，NTP 服务器根据自己的时钟信息和与其他时钟源的同步情况，计算出一个准确的时间值并返回给客户端。客户端收到服务器的响应后，会根据网络延迟等因素对时间进行调整，逐步将自身时钟与服务器时钟同步。这个过程会持续进行，以不断纠正时钟偏差。

Chrony 是网络时间协议 NTP 的实现，类 Unix 系统上 NTP 客户端和服务器的替代品。它可以通过 NTP 服务或者类似 GPS 时钟接收器的硬件级参考时钟来同步系统时钟，具有更好的时钟准确度，并且对于那些间歇性互联网连接的系统很有帮助。

Chrony 有两个核心组件：一个是 chronyd 守护进程，主要用于调整内核中运行的系统时间和时间服务器同步。它确定计算机增减时间的比率，并对此进行调整补偿。另一个是 chronyc，它提供一个用户界面，用于监控性能并进行多样化的配置。chronyc 可以在 chronyd 实例控制的计算机上工作，也可以在一台不同的远程计算机上工作。

7.8.2 配置 Chrony

在 CQOS24 中默认安装了 Chrony 并已设置成开机自启，它的默认配置文件为 /etc/chrony.conf，下面将介绍一些常用的配置项：

```
server hostname [option]
```

server 指令用于指定要同步的 NTP 服务器。例如 ntp1.aliyun.com iburst，其中的 ntp1.aliyun.com 是 NTP 服务器的地址。

iburst 是参数，一般用此参数即可。该参数的含义是在头四次 NTP 请求以 2s 或者更短的间隔，而不是以 minpoll x 指定的最小间隔，这样的设置可以让 chronyd 启动时快速

进行一次同步。其他的参数有 minpoll x 默认值是 6，代表 64s。maxpoll x 默认值是 9，代表 512s。

```
driftfile file
```

Chrony 会根据实际时间计算修正值，并将补偿参数记录在该指令指定的文件里，默认为 driftfile /var/lib/chrony/drift。与 ntpd 或者 ntpdate 最大的区别就是，Chrony 的修正是连续的，通过减慢时钟或者加快时钟的方式连续地修正。而 ntpd 或者 ntpdate 搭配 Crontab 的校时工具是直接调整时间，会出现间断，并且相同时间可能会出现两次。因此，请放弃使用 ntpd、ntpdate 来校时。

```
makestep threshold limit
```

此指令使 Chrony 根据需要通过加速或减慢时钟来逐渐校正任何时间偏移。例如：makestep 1.0 3，就表示当头三次校时，如果时间相差 1.0s，则跳跃式校时。

```
rtcsync
```

启用内核时间与 RTC 时间同步（自动写回硬件）。

```
logdir
```

该参数用于指定 Chrony 日志文件的路径。

```
stratumweight
```

该参数用于设置当 chronyd 从可用源中选择同步源时，每个层应该添加多少距离到同步距离。默认情况下设置为 0，让 chronyd 在选择源时忽略源的层级。

7.8.3 管理 Chrony

使用 chronyc 来管理 Chrony，例如：

检查 Chrony 是否实际同步：

```
chronyc tracking
```

该命令输出中主要关注 Update interval 这个参数，它说明最后两次更新的时间间隔。

显示所有 NTP 源服务器的信息：

```
chronyc sources -v
```

查看 NTP 服务器的在线和离线状态：

```
chronyc activity
```

查看 Chrony 服务的日志：

```
journalctl -u chronyd
```

手动添加一台新的 NTP 服务器：

```
chronyc add server NTP 地址
```

手动移除 NTP 服务器或对等服务器：

```
chronyc delete NTP 地址
```

7.9 REAR

7.9.1 介绍 REAR

Relax＆Recover（简称 ReaR）是一种迁移和灾难恢复工具。REAR 为正在运行的 Linux 机器创建一个可引导映像，并在需要时使用相同的可引导映像来恢复系统。要使用 Rear，至少需要两个相同的系统：运行生产环境的计算机，以及相同的测试计算机。举例来说，这里所说的“相同”是指用户可以将一块网卡替换为使用相同内核驱动程序的另一块网卡。注意如果某个硬件组件使用的驱动程序不同于生产环境中所用的驱动程序，则 Rear 不会将该组件视为相同。

REAR 保留了 OS 的最新状态，包括其分区，引导加载程序配置，所有系统数据等。它是 mkcdrc 的后继产品，其设计简单易行且无需维护。其简单的设置和特性使其成为灾难恢复和迁移的理想解决方案，实际上，它是领先的开源灾难恢复解决方案。

7.9.2 配置和使用 REAR

假设本地计算机的 IP 是 192.168.16.128，我们使用 nfs 网络文件挂载系统来存储本地系统可引导映像，nfs 服务器的 IP 是 192.168.16.137。

第一步，配置 NFS 网络文件挂载系统：

本地计算机和 nfs 服务器安装 nfs：

```
[root@localhost ~]# dnf install -y nfs-utils
```

放行防火墙 nfs 服务：

```
[root@localhost ~]# firewall-cmd --zone=public --add-service=nfs --permanent
[root@localhost ~]# firewall-cmd --reload
```

在 NFS 服务器上建立用于 NFS 文件共享的目录，并设置足够的权限确保其他人也有写入权限。

```
[root@localhost ~]# mkdir /backup
[root@localhost ~]# chmod -Rf 777 /backup
```

修改 NFS 服务程序的配置文件为/etc/exports（默认情况下里面没有任何内容）：

```
[root@localhost ~]# vim /etc/exports
# 在文件中按照“共享目录的路径 允许访问的 NFS 客户端（共享权限参数）”的格式添加以下内容：
/backup 192.168.16.*(rw,sync,root_squash)
```

启动和启用 NFS 服务程序。由于在使用 NFS 服务进行文件共享之前，需要使用 RPC（Remote Procedure Call，远程过程调用）服务将 NFS 服务器的 IP 地址和端口号等信息发送给客户端。

```
[root@localhost ~]# systemctl restart rpcbind
```

```
[root@localhost ~]# systemctl enable rpcbind
[root@localhost ~]# systemctl start nfs-server
[root@localhost ~]# systemctl enable nfs-server
```

在本地计算机上创建挂载目录并使用 mount 命令进行挂载：

```
[root@localhost ~]# mkdir /nfsfire
[root@localhost ~]# mount -t nfs 192.168.16.137:/backup /nfsfile
```

第二步，安装 rear：

```
[root@localhost ~]# dnf install -y rear genisoimage syslinux
```

第三步，修改配置文件/etc/rear/local.conf：

```
[root@localhost ~]# vim /etc/rear/local.conf
# 在配置文件中添加以下内容：
OUTPUT=ISO
BACKUP=NETFS
BACKUP_URL=nfs://192.168.16.137/backup
```

第四步，创建系统的可引导映像并生成备份文件：

```
[root@localhost ~]# rear -d -v mkbackup
```

REAR 将检查并收集所需的必要文件，例如磁盘布局，引导加载程序信息等，然后将在目录/var/lib/rear/output 下为 OS 创建一个 ISO 映像。之后，所有文件以及备份 ISO 映像将被传输到 NFS 共享。注意：此过程可能需要一些时间，具体取决于操作系统的大小。

第五步，恢复测试：

将可引导 ISO 映像刻录到 CD 或 DVD 等恢复介质上（软碟通可以刻录）；

从恢复介质引导测试计算机；

系统启动图像后，我们将看到一个 REAR 菜单，选择第一个选项，即“Recover localhost”（“localhost”是系统的主机名）并按 Enter。如果恢复目标机器配置类似也可选择第二个选项进行自动恢复；

此后，我们将获得登录菜单，使用 root 用户登录而无需密码，注意：我们还可以选择第二个选项“自动恢复本地主机”，但是我们需要确保恢复期间不会出现任何错误。自动恢复期间发生的一些常见错误是由于网络更改、磁盘布局更改等引起的。因此，如果要将操作系统还原到同一系统，则可以尝试使用“自动恢复”来自动执行恢复过程，否则请选择“手动恢复”。

执行以下命令将备份恢复到此测试服务器：

```
[root@localhost ~]# rear -d -v recover
```

该命令将执行以下步骤：

1) 恢复磁盘布局（分区、文件系统和安装点）。

2) 从备份中恢复系统和用户文件。

3) 恢复引导加载程序。

4) 恢复过程完成后，检查是否已成功创建系统，并且该系统可在生产环境中替代原始系统运作。

7.10 Docker

7.10.1 Docker 介绍

Docker 是一个用于开发、传送和运行应用程序的开放平台。Docker 能够将应用程序与基础设施分开，以便用户快速交付软件。使用 Docker，用户可以像管理应用程序一样管理基础设施。

7.10.2 配置 Docker

- 安装 docker

```
[root@localhost ~]# dnf install docker -y
```

- 启动 docker

```
[root@localhost ~]# systemctl start docker.service
```

- 配置开机启动

```
[root@localhost ~]# systemctl enable docker.service
```

- 查看状态

```
[root@localhost ~]# systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
     Active: active (running) since Mon 2024-10-21 17:11:23 CST; 7s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1879 (dockerd)
      Tasks: 9
     Memory: 102.3M
     CGroup: /system.slice/docker.service
             └─ 1879 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
.....
```

7.10.3 常用操作命令

- 查看 docker 所有命令

```
[root@localhost ~]# docker command --help
```

- 查看当前 docker 版本

```
[root@localhost ~]# docker --version
```

- 查看已下载本地镜像

```
[root@localhost ~]# docker images
```

- 下载镜像

```
[root@localhost ~]# docker pull  镜像名称:TAG
```

- 删除镜像

```
[root@localhost ~]# docker rmi  镜像 ID 或镜像名称:TAG
```

- 运行一个容器

```
[root@localhost ~]# docker run --name  容器名  -i -t -p  主机端口:容器端口  -d -v  主机目录:容器目录:ro  镜像 ID 或镜像名:TAG
# --name  指定容器名，可自定义，不指定自动命名
# -i  以交互模式运行容器
# -t  分配一个伪终端，即命令行，通常-it 组合来使用
# -p  指定映射端口，将主机端口映射到容器内的端口
# -d  后台运行容器
# -v  指定挂载主机目录到容器目录，默认为 rw 读写模式，ro 表示只读
```

- 查看容器列表

```
[root@localhost ~]# docker ps -a -q
# docker ps 查看正在运行的容器
# -a  查看所有容器（运行中、未运行）
# -q  只查看容器的 ID
```

- 启动、停止、重启、删除容器

```
[root@localhost ~]# docker start  容器 ID 或容器名
[root@localhost ~]# docker stop  容器 ID 或容器名
[root@localhost ~]# docker restart  容器 ID 或容器名
[root@localhost ~]# docker rm  容器 ID 或容器名
```

- 查看容器日志

```
[root@localhost ~]# docker logs  容器 ID 或容器名
```

⚠ **注意：Docker 在启动时会自动在 iptables 中创建一系列的规则，用于管理容器间和主机之间的网络通信。而 firewalld 重启时，它会将 iptables 上的规则全部清空，并将 firewalld 上的防火墙策略规则写入到 iptables 中。使用 docker 时，若用户手动操作重启 firewalld，请在重启 firewalld 之后重启 docker 服务。**

8 系统监控

8.1 系统监控工具

系统监控工具能够实时的监控收集软件进程在系统的运行情况，包括进程 PID 号，占用了多少 CPU 和内存资源，针对网络资源监控和磁盘管理本节也有描述。为了方便用户的日常管理与维护，下面介绍几个实用、好用的监控工具。

8.1.1 查看 CPU

top 命令提供了系统运行的动态实时视图。显示运行时间、系统负载、线程运行情况、CPU 占用率以及内存使用情况。

```
[root@localhost ~]# top
top - 10:04:56 up 13 min,  3 users,  load average: 0.07, 0.35, 0.23
Tasks: 142 total,   1 running, 141 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.1 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3724.7 total,   1933.4 free,    199.3 used,   1592.0 buff/cache
MiB Swap:   4052.0 total,   4052.0 free,      0.0 used.   3280.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
    231 root      20   0       0      0      0 I   0.3   0.0   0:00.89 kworker/1:3-events
   7247 root      20   0  193848  12052   9716 S   0.3   0.3   0:00.09 sssd_be
  29069 root      20   0   53692   4004   3428 R   0.3   0.1   0:00.02 top
      1 root      20   0   79388  16024   9868 S   0.0   0.4   0:04.05 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
```

8.1.2 查看系统进程

ps ax 命令可以列出系统中的所有进程，包括进程 PID、启动进程的终端、进程的状态、进程占用 CPU 的时间以及可执行文件的名字。

可以在 shell 终端中使用如下命令，列出系统中的所有进程。

```
[root@localhost ~]# ps ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:04 /usr/lib/systemd/systemd --system --deserialize 21
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
      4 ?        I<     0:00 [rcu_par_gp]
      6 ?        I<     0:00 [kworker/0:0H-events_highpri]
```

```
      8 ?        I<     0:00 [mm_percpu_wq]
      9 ?        S      0:00 [rcu_tasks_kthre]
     10 ?        S      0:00 [rcu_tasks_rude_]
     11 ?        S      0:00 [rcu_tasks_trace]
     12 ?        S      0:00 [ksoftirqd/0]
     13 ?        I      0:00 [rcu_sched]
     14 ?        S      0:00 [migration/0]
     16 ?        S      0:00 [cpuhp/0]
     17 ?        S      0:00 [cpuhp/1]
```

8.1.3 查看内存

free -h 命令可以查看系统的内存和交换分区使用的总量，以及内核使用的缓冲区和缓存。

```
[root@localhost ~]# free -h
               total        used        free      shared  buff/cache   available
Mem:           3.6Gi       198Mi       1.9Gi       8.0Mi       1.6Gi       3.2Gi
Swap:          4.0Gi          0B       4.0Gi
```

8.1.4 查看硬盘、分区和文件系统

8.1.4.1 lsblk 命令

lsblk 命令可以查看硬盘大小和 lvm 动态逻辑磁盘。

```
[root@localhost ~]# lsblk
NAME                MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda                   8:0    0   100G  0 disk
├─sda1                8:1    0   600M  0 part /boot/efi
├─sda2                8:2    0     1G  0 part /boot
└─sda3                8:3    0  98.4G  0 part
  ├─cslo-root 253:0    0  63.5G  0 lvm  /
  ├─cslo-swap 253:1    0     4G  0 lvm  [SWAP]
  └─cslo-home 253:2    0    31G  0 lvm  /home
sr0                  11:0    1   2.6G  0 rom
```

8.1.4.2 df -HT 命令

df -HT 命令可以查看磁盘内的文件系统和分区使用情况。

```
[root@localhost ~]# df -HT
```

8.1.4.3 blkid 命令

blkid 命令可以显示可用块设备底层信息，包含块设备路径、UUID 和文件系统类型。

```
[root@localhost ~]# blkid
```

/dev/mapper/cslo-root: UUID="e8064980-2c01-490c-80c5-2382d4fd2005" BLOCK_SIZE="512" TYPE="xfs"

/dev/sda3: UUID="ief4yB-IJMZ-xID2-0exY-rd71-c6T4-lCAFzb" TYPE="LVM2_member" PARTUUID="f6e5bf92-c4ef-42fc-92d4-8d6b6fb0b46c"

/dev/mapper/cslo-swap: UUID="b91761a0-3ae3-46e0-933f-e5eec76746d4" TYPE="swap"

/dev/sr0: BLOCK_SIZE="2048" UUID="2024-07-24-17-06-21-00" LABEL="CQ-Security" TYPE="iso9660"

/dev/mapper/cslo-home: UUID="8870f01a-ec81-4ffb-8136-3e940357618e" BLOCK_SIZE="512" TYPE="xfs"

/dev/sda2: UUID="24095ce6-a01f-4c49-ba7c-5a94f2febe64" BLOCK_SIZE="512" TYPE="xfs" PARTUUID="2ac381b6-c148-4823-9004-9d99c8568b36"

/dev/sda1: UUID="CC89-E85C" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="EFI System Partition" PARTUUID="cc98cc5a-6edf-4e61-a98a-26b034123098"

8.1.5 查看硬件信息

8.1.5.1 lspci 命令

lspci 命令显示了有关系统 PCI 总线和所连接设备信息，可以使用 lspci -vvx 显示详细信息，如下所示。

```
[root@localhost ~]# lspci -vvx
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
        Subsystem: Virtio: Qemu virtual machine
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap- 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
00: 86 80 37 12 07 00 00 00 02 00 00 06 00 00 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 f4 1a 00 11
30: 00 00 00 00 00 00 00 00 00 00 00 00 ff 00 00 00

00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
        Subsystem: Virtio: Qemu virtual machine
        Physical Slot: 1
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap- 66MHz- UDF- FastB2B- ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
```

```
        Latency: 0
```

8.1.5.2 lsusb 命令

lsusb 命令显示了有关系统 USB 总线和所连接设备信息。

```
[root@localhost ~]# lsusb
Bus 001 Device 002: ID 0627:0001 Adomax Technology Co., Ltd
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

8.1.5.3 lscpu 命令

lscpu 命令显示了当前系统的 CPU 信息，包含了 CPU 架构、CPU 核数目、厂商名称、型号名称等信息。

```
[root@localhost ~]# lscpu
```

8.2 查看和管理日志文件

日志文件包括系统运行日志、服务日志、应用程序日志等，日志文件在管理系统时不可或缺，它们提供了对系统运行状态及应用程序的运行信息，有助于排查诊断问题、监控系统行为及维护安全性。

在系统中日志文件主要由 rsyslogd 和 journald 守护进程控制管理，rsyslogd 是 syslogd 的增强替代品，提供扩展过滤、加密保护消息转发、各种配置选项、输入和输出模块，支持通过 TCP 或 UDP 协议转发或接收日志文件。journald 守护进程捕获 Syslog 消息、内核日志消息、初始 RAM 磁盘和早期启动消息，以及写入到所有服务的标准输出和标准错误输出的消息，对消息进行索引并供用户使用。本地日志文件是结构化、索引化的二进制文件，改进了搜索及操作速度，另外还存储了时间戳或用户 ID 等元数据信息。journald 生成的日志文件并不是永久的，它保存在内存中或/run/log/journal/目录中的小型环形缓冲区。记录数据量取决系统可用内存，当文件容量限制时，会删除最旧的记录。此设置可以更改，具体方法请参考“8.4.5 持久存储”章节。

默认情况下，两个日志记录工具在系统中共存。journald 守护进程是故障排除的主要工具，它还提供了创建结构化日志消息所需的其他数据。journald 获取的数据将转发到 /run/systemd/journal/syslog 套接字，供 rsyslogd 用于进一步处理数据。但是 rsyslog 默认通过 imjournal 输入模块进行实际集成，从而避免使用套接字。同时还可以使用 omjournal 模块，将数据以反方向从 rsyslogd 传输到 journald。

8.2.1 日志文件位置

rsyslogd 的主配置文件为/etc/rsyslog.conf，文件中配置了由它维护日志的保存位置，大多文件保存在系统/var/log/目录下，如/var/log/messages 存放全局系统运行日志，同时会

有一些应用程序会在此目录下生成自己的保存目录，如 nginx 服务日志会存放在/var/log/nginx 目录中。

同时可能会在/var/log 目录中存在多个日志文件名字后缀数字不同，这是因为日志文件轮替保存时间戳不同，由 Logrotate 来实现日志轮替并通过/etc/logrotate.conf 及/etc/logrotate.d/目录下文件来配置。

8.2.2 Rsyslog 的基本配置

Rsyslog 的主配置文件是/etc/rsyslog.conf，可以在此文件中指定由过滤和操作部分组成的全局指令、模块和规则。

8.2.2.1 过滤器

规则通过过滤器来设定，过滤器选择系统日志消息的子集和行动部分，指定了对消息处理动作。通过/etc/rsyslog.conf 配置文件来定义规则，需要在一行中同时定义过滤器和动作，并用空格或制表符来分隔。

rsyslog 提供了根据属性过滤日志的多种方式。主要有基于设置/基于优先级、基于属性及基于正则表达式的过滤器。下面介绍一种最常用的过滤器：基于设置/基于优先级的过滤器。

基于设置/优先级的过滤器是最常用的方式，它会根据两个条件过滤 syslog 消息：由点号分分隔设备和优先级。需要创建选择器，语法如下：

```
FACILITY.PRIORITY
```

FACILITY 指定一个子系统生成 syslog 消息。如 authpriv 子系统处理与认证相关的所有 syslog 消息。FACILITY 由以下几种关键字（或数字代码）表示：Kern(0)、用户(1)、mail(2)、daemon(3)、auth(4)、syslog(5)、lpr(6)、news(7)、cron(8)、authpriv(9)、ftp(10)及local0-local7(16-23)。

PRIORITY 指定系统日志消息的优先级。PRIORITY 由以下几种关键字来表示（或按编号表示）：debug(7)、info(6)、notice(5)、warning(4)、err(3)、crit(2)、alert(1)和 emerg(0)。

上述语法选择优先级为已定义或更高的系统日志消息。通过比较等号（=）两边优先级，预先指定的优先级系统日志消息被选中，同时其他优先级会被忽略。如果优先级关键字前有(!)将会选择除预先设置优先级消息之外的所有系统日志消息。

除了上面的关键字外，也可以使用星号(*)定义所有设备或优先级（具体取决于星号位置）。关键字 none 适用于没有指定优先级的设备，设备和优先级条件不区分大小写。

要定义多个设备和优先级，需要用逗号分隔(,)。如果要在一行中定义多个选择器，用分号(;)分隔。要注意的是选择器字段可以覆盖前面的选择器，这可以从模式中排除一些优先级。

实例：设备/基于优先级的过滤器

下面是在/etc/rsyslog.conf 中指定设施/基于优先级的过滤器的几个示例。如果选择所有优先级的所有邮件子系统的日志消息，需添加如下配置：

```
mail.*
```

要选择所有的 cron 子系统除了优先级为 info 和 debug 的日志消息，配置如下：

```
cron.!info,!debug
```

8.2.2.2 行为

行为是指对已经定义选择器过滤的消息要进行的操作。以下是可以在规则中定义的一些操作：

保存 syslog 消息到日志文件

行为会指定将 syslog 消息保存到指定日志文件中。通过在已定义的选择器后指定文件路径来实现。

```
FILTER PATH
```

FILTER 代表用户指定选择器，PATH 指定用户定义目标文件的保存路径。实例如下：

```
authpriv.*                  /var/log/secure
```

此实例说明规则会选择所有的 authpriv 日志消息并将其保存到系统/var/log/secure 文件中。

默认情况下，每次生成 syslog 消息时都会同步日志文件。使用标记(-)来指定的文件路径前缀，以忽略同步操作。

⚠ **注意：这种设置可能会丢失日志信息，如果在写入之前系统终止，就可能会丢失信息。但是它可以提高性能，尤其在运行生成非常详细的日志消息的程序。**

除了指定静态文件路径外还可以指定动态文件路径，静态文件由固定文件路径表示，如上实例所示。动态文件路径可能会根据收到的消息而有所不同。动态文件路径由模板和一个问号(?)前缀表示：

```
FILTER ?DynamicFile
```

DynamicFile 是预先定义的模板名称，其用来修改输出路径。同时可以使用符号（-）来禁用同步，也可以使用一个冒号（;）分隔的多个模板。

如果指定的文件是存在的终端或/dev/console 设备，系统日志消息会在使用 X Window 系统时发送到标准输出（使用特殊的终端处理）或控制台（使用特殊的/dev/console-处理）。

通过网络发送 syslog 消息

rsyslog 可以通过网络发送和接收系统日志消息。它通过一台主机管理多台主机的系统日志。如果要将 syslog 信息转发到远程机器，参考语法如下：

```
@[(zNUMBER)]HOST:[PORT]
```

其中：

- "@"符号表示发送 syslog 消息时使用 UDP 协议；
- "@@"符号表示发送 syslog 消息时使用 TCP 协议；
- zNUMBER 是可选选项，此选项可以启动 zlib 压缩，NUMBER 属性用来指定压缩级别（由低到高为 1-9）。rsyslogd 自动检查压缩增益，只有存在压缩增益且消息大小不小于 60 字节时才会压缩。
- HOST 属性指定接收日志消息的主机
- PORT 属性指定主机端口
- 当主机地址为 IPV6 时，地址需用[]括号括起来。

实例：通过网络发送 syslog 消息

下面选择器实例是通过网络发送 syslog 消息的配置，需要注意所有的操作前面需定义选择器，如果通过 UDP 协议发送到 192.168.10.1，则如下：

```
*.* @192.168.0.1
```

如果使用 TCP 协议 9876 端口发送日志消息到 192.168.5.1，则如下：

```
*.* @@192.168.5.1:9876
```

如果要使用 zlib 压缩日志信息并使用 UDP 协议发送到 192.168.5.1，则如下：

```
*.* @(z9)192.168.5.1:9876
```

8.2.3 全局指令

全局指令是用于 rsyslogd 守护进程的配置选项，它们会影响 rsyslogd 守护进程的行为或规则的特定预定义变量的值，全局指令必须在全局配置块中。如下是全局指令的一个示例，其指定了 rsyslogd 工作目录：

```
global(workDirectory="/var/lib/rsyslog")
```

指令可以在/etc/rsyslog.conf 文件中配置多个，一个指令会影响到所有配置选项的行为，直到同一指令的另一事件被检测到。全局指令可以用来配置行为、队列和调试。

8.2.4 日志轮替

日志轮替是指定期对日志文件进行处理的过程，目的是防止日志文件无限制地增长导致占用过多磁盘文件。通过日志轮替可以将旧的日志文件归档、压缩、删除，同时创建新的日志文件继续记录新的日志信息，系统中日志轮替由 logrotate 工具完成。

logrotate 的主配置文件默认为/etc/logrorate.conf，系统默认 logrotate.conf 配置文件内容如下：

```
# see "man logrotate" for details
weekly     #每周对日志文件进行一次轮替
rotate 4   #轮替日志文件最多保存 4 周
create     #在日志轮替时，自动创建新的日志文件
dateext    #以日期作为日志文件后缀命名
#compress #日志文件是否压缩

# RPM packages drop log rotation information into this directory
include /etc/logrotate.d    #包含子目录/etc/logrorate.d 中的所有配置文件

# system-specific logs may be also be configured here.
```

其中所有行定义了全局选项。适用于每个日志文件，文件定义日志文件每周轮替，轮替的日志文件保留 4 周且以日期为文件名称后缀命名，所有轮替日志文件不压缩。

此外还可以在/etc/lograte.d/目录下放置多个配置文件来针对不同服务的日志文件进行配置，如 redis 日志文件配置如下：

```
/var/log/redis/*.log {
        weekly     #每周轮替
        rotate 10     #轮替文件最多保存 10 周
        copytruncate #在创建新日志文件之前，先复制当前的日志文件，然后截断原文件。
delaycompress   #延迟压缩操作直到下一个轮替周期。
        compress     #日志文件压缩
        notifempty   #如日志为空，也要进行轮替
        missingok    #如日志不存在，则忽略该日志的告警信息
}
```

此文件中的配置只为 redis 日志文件生效，定义日志文件每周轮替，轮替日志保存 10 周，所有的轮替日志文件由 gzip 压缩的成.gz 的格式。

8.2.5 使用 rsyslog 模块

由于其模块化设计，rsyslog 提供多种模块，可以提供许多额外功能。但是模块可以由第三方编写。大多数模块提供额外的输入或输出，其他模块提供不同的功能。加载模块后需要提供额外的配置指令。如要载入模块，语法如下：

```
module(load=”MODULE”)
```

其中 module 指定所需要的模块，例如要加载可让 rsyslog 将标准文本文件转换为 syslog 信息的文本输入模块（imfile），需在/etc/rsyslog.conf 配置如下：

```
module(load=”imfile”)
```

rsyslog 提供较多模块，主要分为以下几类：

- 输入模块-输入模块可以收集各种来源消息。输入模块的名称以 im 前缀开头，如 imfile 和 imjournal。
- 输出模块-输出模块提供发送消息到各种目标功能，如通过网络发送、存储在数据库或加密等目标。输出模块的名称以 om 前缀（如 omsnmp、omrelp 等）开头。
- 解析器模块-这些模块可用于创建自定义解析规则或解析错误的消息。此类模块名称以 pm 前缀开头，如 pmrfc5424、pmrfc3164 等。
- 消息修改模块-消息修改模块用来更改 syslog 消息的内容。这些模块的名称以 mm 前缀开头。消息修改模块（如 mmanon、MMnormalize 或 mmjsonparse）用于消息的匿名化或规范化。
- 字符串生成模块-此类模块基于消息内容生成字符串，并与 rsyslog 提供的模板功能高度协作。字符串生成器模块的名称以 sm 前缀开头，如 smfile 或 smtradfile。
- 库模块-库模块用来提供其他可加载模块。在需要时，rsyslog 会自动加载这些模块，用户无法配置这些模块。

8.2.6 Syslogd 服务和日志的交互

如前面章节所述，系统中存在 rsyslog 和 journal 两个日志记录工具，每个工具有不同功能适合特定的场景。同时在特定情景下可以使两者结合使用，如在创建结构化的信息时需将它们保存在一个文件数据库中。此时就需要两者合作，由 rsyslog 提供输入输出模块及 journal 提供通信 socket。

默认情况下 rsyslogd 会使用 imjournal 模块作为日志文件的输入模式，此模块可以导入消息及导入 journald 提供的结构化数据。此外可以将 rsyslogd 配置为读取由 journal 提供的 socket 来作为 syslog 应用的输出，该 socket 的路径为/run/systemd/journal/syslog。如

果想维护 rsyslog 日志时可以使用此选项，它比 imjournal 模块有更多的功能。如果要绑定规则集或过滤器就要通过 socket 来导入日志数据，需要在/ect/rsyslog.conf 中配置如下：

```
module(load="imuxsock"
       SysSock.Use="on"
       SysSock.Name="/run/systemd/journal/syslog")
```

此外还可以通过 omjournal 模块输出来自 Rsyslog 到 journal 的日志信息，需要配置如下：

```
module(load="omjournal")
action(type="omjournal")
```

如以下配置将 TCP 端口 7654 上的所有接收的信息转发到日志：

```
module(load="imtcp")
module(load="omjournal")
ruleset(name="remote") {
       action(type="omjournal")
}
input(type="imtcp" port="7654" ruleset="remote")
```

8.3 调试 Rsyslog

调试 rsyslog 是确保日志记录系统正常运行的关键步骤，在 debug 模式运行 rsyslogd，使用如下命令，产生调试信息并打印到标准输出。

```
[root@localhost ~]# rsyslogd -dn
```

在启动 rsyslogd 之前，在命令行上执行以下操作：

```
export RSYSLOG_DEBUGLOG="path"
export RSYSLOG_DEBUG="Debug"
```

"path"指定记录调试信息文件的位置。对于可用于 RSYSLOG_DEBUG 变量选项的完整列表。

使用 rsyslog 自带的语法检查工具，来检查配置文件的语法：

```
[root@localhost ~]# rsyslogd -N 1
```

- -N：调试选项，会读取配置文件并模拟启动过程，但不会实际启动服务。
- 1：调试级别，0 表示关闭，1 表示最低级别的调试信息，7 表示最高级别的调试信息。

8.4 使用 journal

journal 是 systemd 的一个组件，它负责日志文件的查看与管理任务。该组件旨在作为对旧有日志守护进程（如 rsyslogd）的并行运行选择或替代方案，其开发初衷是为了解决

与旧的日志记录有关的问题。它深度集成了系统的各个部分，支持多样化的日志记录技术和访问管理日志文件。

8.4.1 查看日志文件

journal 管理的日志可以使用 journalctl 命令进行查看，journalctl 是查看和管理 systemd-journald 日志的主要工具。例如，要查看所有日志（包括系统组件和用户产生的消息），使用 root 用户运行命令：

```
[root@localhost ~]# journalctl
```

要查看特定服务的日志，可以使用-u 选项：

```
[root@localhost ~]# journalctl -u sshd.service
```

8.4.2 访问控制

访问 journald 日志通常需要具有相应的系统权限。例如，普通用户可能无法查看所有服务的日志，只能查看当前用户所产生的日志，而只有 root 用户或具有 sudo 权限的用户才能查看所有日志。若要授予普通用户查看完整日志文件的权限，使用 root 用户输入：

```
[root@localhost ~]# usermod -a -G adm username
```

username 表示普通用户的用户名，重新登录后就能看见所有日志了。

8.4.3 使用 Live view

journal 支持实时查看日志更新，类似于 tail -f。

- 实时查看日志：

使用-f 或--follow 选项来实时查看日志更新，使用命令来开启 journalctl 的 Live view 模式：

```
[root@localhost ~]# journalctl -f
```

上述命令返回十个最新的日志行消息。journalctl 工具将保持运行，等待新的消息并立即显示他们。

8.4.4 过滤消息

journal 提供了强大的过滤功能，允许用户根据各种条件（如优先级、时间、服务名等）来过滤日志消息。

- 基于优先级的过滤：

使用-p 或--priority 选项来过滤特定优先级的日志。例如，仅查看错误和警告消息：

```
[root@localhost ~]# journalctl -p err..warning
```

也可以使用下列关键字或优先级（数字）来替代 err..warning 进行过滤：：debug（7），info（6），notice（5），warning（4），err（3），crit（2），alert（1），emerg（0）。

- 基于时间的过滤：

查看过去一小时的日志：

```
[root@localhost ~]# journalctl -S -1H
```

查看今天的日志：

```
[root@localhost ~]# journalctl -S today
```

限制时间范围输出：可以通过添加时间范围等参数来限制 journalctl 的输出。例如，查看时间范围内的日志：

```
[root@localhost ~]# journalctl --since="StartTime" --until="EndTime"
```

StartTime 表示想要查询日志的开始时间，EndTime 则表示结束时间，时间格式："yyyy-MM-dd HH:mm:ss"。

- 基于字段的过滤：

可以使用_SYSTEMD_UNIT、_COMM 等字段来过滤特定服务或进程的日志。例如，查看所有由 sshd 生成的日志：

```
[root@localhost ~]# journalctl _COMM=sshd
```

8.4.5 持久存储

默认情况下，journal 只在内存中或者小环形缓冲区中存储日志文件，不会永久保存。若要使系统日志能够持久化地保存在磁盘上，则需要配置持久化存储，即使系统重启后也能保留这些日志记录。

配置持久化存储的步骤：

1) 创建目录：

如果/var/log/journal 目录不存在，则创建它。

```
[root@localhost ~]# mkdir -p /var/log/journal
```

2) 修改 journald 配置：

编辑/etc/systemd/journald.conf 文件，找到 Storage=行，并将其设置为 persistent 或 auto。persistent 表示始终将日志存储在磁盘上，而 auto 则会在磁盘空间不足时回退到易失性存储。

3) 重启 journald 服务：

```
[root@localhost ~]# systemctl restart systemd-journald
```

9 外设管控

9.1 外设管控概述

长擎安全操作系统 24 提供 devadm 工具进行外设管控。devadm 工具能够对外设进行统一的管控。

devadm 工具可以实现对 USB，无线，串口等多种外设的控制，devadm 工具会根据当前的 USB 设备自动生成配置文件，防止接入不受信任的 USB 设备，并支持自定义 USB 规则以确保设备的可信性。同时，devadm 工具还提供对无线和串口设备的管理控制。

9.2 外设管控工具使用

CQOS24 提供的 devadm 外设管控工具需要使用 root 用户进行设置，使用方法如下：

9.2.1 列出 usb 设备列表

```
[root@localhost ~]# devadm -l
```

9.2.2 列出 usb 规则列表

```
[root@localhost ~]# devadm -r
```

9.2.3 控制 usb 设备的使用权限

1) 列出 usb 设备列表，获取设备 id

```
[root@localhost ~]# devadm -l
```

2) 将默认允许的规则添加到规则文件中

```
[root@localhost ~]# devadm -a “allow id”
```

3) 查看规则文件列表（列表中其他的规则为初始规则）

```
[root@localhost ~]# devadm -r
```

4) 通过行号删除对应行的规则

```
[root@localhost ~]# devadm -m 行号
```

5) 再次查看规则文件

```
[root@localhost ~]# devadm -r
```

9.2.4 为指定 usb 设备添加或删除指定规则

1) 列出 usb 设备列表，获取设备 id

```
[root@localhost ~]# devadm -l
```

2) 关闭鼠标使用权限

```
[root@localhost ~]# devadm -c id
```

3) 查看设备列表

```
[root@localhost ~]# devadm -l
```

4) 开启鼠标使用权限

```
[root@localhost ~]# devadm -s id
```

5) 查看设备列表

```
[root@localhost ~]# devadm -l
```

9.2.5 列出无线设备列表

```
[root@localhost ~]# devadm -w
```

9.2.6 控制无线设备的开启关闭

1) 查看无线设备列表，并获取设备 id

```
[root@localhost ~]# devadm -w
```

2) 关闭蓝牙

```
[root@localhost ~]# devadm -b id
```

3) 或者执行如下命令：

```
[root@localhost ~]# devadm -b bluetooth
```

4) 查看无线设备列表

```
[root@localhost ~]# devadm -w
```

5) 开启蓝牙

```
[root@localhost ~]# devadm -u id
```

6) 或者执行如下命令：

```
[root@localhost ~]# devadm -u bluetooth
```

7) 查看无线设备列表

```
[root@localhost ~]# devadm -w
```

9.2.7 控制串口的开启关闭

1) 关闭串口的使用权限

```
[root@localhost ~]# devadm -d serial -t off
```

2) 开启串口的使用权限

```
[root@localhost ~]# devadm -d serial -t on
```

3) 重启系统，查看窗口权限

```
[root@localhost ~]# ll /dev/ttyUSB*
```

10 系统安全

10.1 启动安全

10.1.1 可信引导

长擎安全操作系统 24 提供可信引导功能，利用 TPM2.0 可信平台模块对加载的内核镜像、根文件系统镜像以及设备描述信息进行完整性度量验证，保证操作系统引导过程中的安全可信。

可信引导模块可以防止因计算机启动相关文件被篡改而导致的系统破坏，提供对启动过程中涉及的引导程序和引导文件进行度量验证，防止使用不可信的配置程序或文件启动。可信引导模块提供初始化、更新启动过程相关的预期度量值功能，允许用户关闭可信引导模块。

10.1.1.1 开启可信引导

CQOS24 提供命令行工具开启可信引导功能，其操作员必须为系统管理员 root 身份：

```
[root@localhost ~]# trust_measure    -e
```

⚠ **注意：开启可信引导时必须确保主机中存在 TPM2.0 可信平台模块。**

10.1.1.2 关闭可信引导

CQOS24 提供命令行工具关闭可信引导功能，其操作员必须为系统管理员 root 身份：

```
[root@localhost ~]# trust_measure    -d
```

⚠ **注意：开启可信引导时必须确保主机中存在 TPM2.0 可信平台模块。**

10.1.1.3 更新可信引导

CQOS24 提供命令行工具更新可信引导中启动相关文件的度量值，其操作员必须为系统管理员 root 身份：

```
[root@localhost ~]# trust_measure    -u
```

⚠ **注意：开启可信引导时必须确保主机中存在 TPM2.0 可信平台模块。**

10.1.2 双因子认证

双因子认证系统是一套高可靠的可灵活定制的用户权限认证系统。功能分两部分：

1) 在用户指定情况下可以生成指定用户的双因子认证数据，并完成相关环境配置。

2) 对指定了使用双因子认证的用户在登录时进行用户权限验证。

双因子的权限管理是遵照三权分立系统规定的权限原则执行的，与传统的 Linux 中用户权限管理有一些差异。安全管理员可以操作任意非管理员用户的所有身份鉴别信息；非

安全管理员（包括系统管理员和审计管理员）可以完全自由操作自己的身份鉴别信息。普通用户可以在授权范围内操作自己的身份鉴别信息。

10.1.2.1 帮助信息查询

命令：

```
[root@localhost ~]# secureid -h/--help
```

可显示命令帮助信息，单独执行 secureid 也会显示帮助信息，详细信息如下。

序号	选项	说明
1	-i/--init	初始化设备，默认的用户 PIN 码为 12345678
2	-a/--assign	安全管理员分配 ukey 设备给用户使用
3	-e/--enable	开启双因子认证
4	-d/--disable	关闭双因子认证
5	-m/--modify <userPIN>/<adminPIN>/<devPasswd>	修改用户 PIN，管理员 PIN 以及设备密码
6	-s/--status	查看双因子认证开启状态
7	-u/--unlock	安全管理员解锁已锁定的用户 PIN
8	-r/--recovery	安全管理员回收分配给用户的 ukey 设备
9	--export <filepath>	导出文件到 ukey 设备中
10	--import <filepath>	从 ukey 设备中导入文件到系统中
11	-h/--help	提示帮助信息

10.1.2.2 初始化设备信息

命令：

```
[root@localhost ~]# secureid -i/--init
```

操作流程：

1) 输入命令。
2) 弹出警告，提示是否初始化 ukey 设备，选择 y 则继续，选择 n 则退出。
3) 若插入多个 ukey 设备，会显示多个可操作选项，选项格式为“【标号】ukey 设备标签-设备序列号”，需选择要操作的 ukey 设备，输入项为标号，若此时只有一个 ukey，则默认操作当前设备。
4) 输入原设备密钥，长度为 16 位，校验成功则继续。
5) 输入新的设备密钥，长度为 16 位，需输入两次且一致。
6) 输入新的管理员 PIN，长度大于等于 8 位，小于等于 256 位，需输入两次且一致。
7) 初始化成功会显示 Success！

使用说明：

1) 输入新的管理员密码或新的设备密码均需输入两次，两次相同才可继续。
2) 设备密钥长度为固定 16 位，管理员密码为大于等于 8 位，小于等于 256 位。
3) 初始化的用户密码为 12345678。
4) 初始化后的 ukey 默认标签名为 Auth-Key，在插入多个 ukey 设备时，选项格式为“【标号】ukey 设备标签-设备序列号”。
5) 只有系统管理员可以执行。

10.1.2.3 分配 ukey 设备

命令：

```
[root@localhost ~]# secureid -a/--assign
```

操作流程：

1) 输入命令。
2) 若插入多个 ukey 设备，会显示多个可操作选项，选项格式为“【标号】ukey 设备标签-设备序列号”，需选择要操作的 ukey 设备，输入项为标号，若此时只有一个 ukey，则默认操作当前设备。
3) 输入 ukey 设备的用户 PIN，校验成功则继续。
4) 输入需分配的用户名称，输入的用户名称需在系统中存在。
5) 设置 ukey 设备标签为分配的用户名，分配 ukey 成功会显示 Assign success !

使用说明：

1) 只有系统管理员可以执行。

10.1.2.4 开启双因子认证

命令：

```
[root@localhost ~]# secureid -e/--enable
```

操作流程：

1) 输入命令。
2) 输入分配给系统管理员 ukey 设备的用户 PIN，校验成功则继续。
3) 开启双因子认证成功会显示 Success !

使用说明：

1) 只有系统管理员可以执行。
2) 需先分配 ukey 设备给系统管理员，若未分配给系统管理员则会执行失败。

10.1.2.5 关闭双因子认证

命令：

```
[root@localhost ~]# secureid -d/--disable
```

操作流程：

1) 输入命令。
2) 输入分配给系统管理员 ukey 设备的用户 PIN，校验成功则继续。
3) 关闭双因子认证成功会显示 Success !

使用说明：

1) 只有系统管理员可以执行。

10.1.2.6 修改用户 PIN

命令：

```
[root@localhost ~]# secureid -m userPIN
```

操作流程：

1) 输入命令。
2) 输入分配给当前用户 ukey 设备的用户 PIN，校验成功则继续。
3) 输入新的用户 PIN，长度大于 8 位，小于等于 256 位，需输入两次且一致。
4) 修改成功会显示 Success !

使用说明：

1) 管理员与普通用户均可执行,但只可以修改分配给当前用户 ukey 设备的用户 PIN

10.1.2.7 修改管理员 PIN

命令：

```
[root@localhost ~]# secureid -m adminPIN
```

操作流程：

1) 输入命令。
2) 输入分配给系统管理员 ukey 设备的用户 PIN，长度大于 8 位，小于等于 256 位，校验成功则继续。
3) 输入管理员 PIN，长度大于 8 位，小于等于 256 位，校验成功则继续。
4) 输入新的管理员 PIN，长度大于 8 位，小于等于 256 位，需输入两次且一致。
5) 修改成功会显示 Success !

使用说明：

1) 只有系统管理员可以执行。

10.1.2.8 修改 ukey 设备密码

命令：

```
[root@localhost ~]# secureid -m DevPasswd
```

操作流程：

1) 输入命令。
2) 输入原设备密钥，长度为 16 位，校验成功则继续。
3) 输入新的设备密钥，长度为 16 位，需输入两次且一致。
4) 修改成功会显示 Success !

使用说明：

1) 只有系统管理员可以执行。
2) 设备密钥长度为 16 位。

10.1.2.9 查看双因子认证状态

命令：

```
[root@localhost ~]# secureid -s/--status
```

操作流程：

1) 输入命令。
2) 读取双因子认证开启状态。

使用说明：

1) 关闭显示为 Ukey check off !
2) 开启显示为 Ukey check on !

10.1.2.10 解锁已锁定的用户 PIN

命令：

```
[root@localhost ~]# secureid -u/--unlock
```

操作流程：

1) 输入命令。
2) 若插入多个 ukey 设备，会显示多个可操作选项，选项格式为“【标号】ukey 设备标签-设备序列号”，需选择要操作的 ukey 设备，输入项为标号，若此时只有一个 ukey，则默认操作当前设备。
3) 查看用户 PIN 是否被锁住，若锁住则继续校验管理员 PIN，校验成功则继续。若未锁住则返回剩余重试次数。
4) 输入新的用户 PIN，长度大于 8 位，小于等于 256 位，需输入两次且一致。
5) 修改成功会显示 Success !

使用说明：

1) 只有系统管理员可以执行。

10.1.2.11 回收分配的 ukey 设备

命令：

```
[root@localhost ~]# secureid -r/--recovery
```

操作流程：

1) 输入命令。
2) 若插入多个 ukey 设备，会显示多个可操作选项，选项格式为“【标号】ukey 设备标签-设备序列号”，需选择要操作的 ukey 设备，输入项为标号，若此时只有一个 ukey，则默认操作当前设备。
3) 校验管理员 PIN，校验成功则继续。
4) 校验用户 PIN，校验成功则继续。
5) 回收成功会显示 Recovery success !

使用说明：

1) 只有系统管理员可以执行。
2) 恢复 ukey 默认标签名为 Auth-Key。

10.1.2.12 导出文件到 ukey 设备中

命令：

```
[root@localhost ~]# secureid --export <filepath>
```

操作流程：

1) 输入命令。
2) 检查文件是否存在，文件存在则继续。
3) 校验用户 PIN，校验成功则继续。
4) 读取文件数据到 ukey 中。
5) 导出成功会显示 Success !

使用说明：

1) 管理员与普通用户均可执行。
2) 命令需带有文件路径。
3) 文件数据大小不超过 30KB。

10.1.2.13 ukey 设备导入文件到主机中

命令：

```
[root@localhost ~]# secureid --import <filepath>
```

操作流程：

1) 输入命令。
2) 校验用户 PIN，校验成功则继续。
3) 读取文件到主机文件中。
4) 导入成功会显示 Success !

使用：

1) 管理员与普通用户均可执行。
2) 命令需带有文件路径。

10.1.3 三权分立

三权分立是一种安全管理机制，旨在通过分离系统管理员、安全管理员和审计管理员的权限，提高系统的安全性和可管理性。以下是对长擎安全操作系统 24 三权分立的介绍。

- 系统管理员（root）

主要负责系统的日常运维工作，如创建用户账号、分配权限、安装软件等。在系统中，系统管理员拥有较高的操作权限，但仅限于其职责范围内的操作。系统管理员的权限受到安全策略和审计机制的监督。

- 安全管理员（secadm）

负责系统的安全策略配置和管理，如配置强访策略、可信度量设置等。安全管理员的权限专注于安全相关的操作。安全管理员的操作同样受到严格的控制和审计。

- 审计管理员（auditadm）

负责对系统操作进行审计，记录并分析系统日志，以发现潜在的安全威胁或违规行为。审计管理员的权限仅限于添加审计规则、查看及分析日志，不具备修改系统配置或用户数据的权限。

10.2 内核安全

10.2.1 核内安全功能及配置

长擎安全操作系统 24 提供对系统中加载的驱动进行签名验证的功能，可以确保驱动程序的完整性和来源可信性，只有经过签名验证的驱动程序才能在系统上加载，这样可以防止未授权的驱动程序对系统造成损害。

CQOS24 默认关闭了内核驱动签名验证功能，需要通过以下步骤开启此安全功能：

1) 登录 root 用户；
2) 编辑/etc/default/grub 配置文件，在“GRUB_CMDLINE_LINUX”配置项后面加入“module.sig_enforce”字符串。示例如下图所示：

```
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto resume=/dev/mapper/cslo-swap rd.lvm.lv=cslo/root rd.lvm.lv=cslo/swap rd.shell=0 module.sig_enforce"
GRUB_DISABLE_RECOVERY="true"
```

3) 更新系统启动时的配置项：

基于传统 BIOS 的机器，使用如下命令：

```
[root@localhost ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

基于 UEFI 的机器，使用如下命令：

```
[root@localhost ~]# grub2-mkconfig -o /boot/efi/EFI/cqos/grub.cfg
```

4) 重启系统。

10.3 网络安全

10.3.1 网络防火墙

防火墙是系统的主要的安全工具，可以提供基本的安全防护。firewalld 支持网络/防火墙区域(zone)定义网络链接、是接口安全等级的动态防火墙管理工具。它支持 IPv4,IPv6 防火墙设置以及以太网桥接，并且拥有运行时配置和永久配置选项。它也支持允许服务或者应用程序直接添加防火墙规则的接口且可以动态管理防火墙。

10.3.1.1 主要功能

- 实现动态管理，对于规则的更改不再需要重新创建整个防火墙；
- 提供 firewall-cmd 命令行界面进行管理及配置工作；
- 实现系统全局及用户进程的防火墙规则配置管理；
- 区域支持。

10.3.1.2 基本命令

- 安装命令：

```
[root@localhost ~]# dnf   install firewalld
```

- firewall 服务在系统中状态：

用法：systemctl [选项] firewalld.service

常见选项说明如下表所示：

序号	选项	说明
1	status	查看 firewall 服务状态
2	start	启动 firewall 服务
3	stop	关闭 firewall 服务
4	enable	设置系统启动时 firewall 服务自动启动
5	disable	设置系统启动时 firewall 服务不启动

- 重新加载 firewalld 防火墙的配置：

```
[root@localhost ~]# firewall-cmd --reload
```

10.3.1.3 区域管理

- 查看 firewalld 中所有可用的区域：

```
[root@localhost ~]# firewall-cmd --get-zones
block dmz drop external home internal nm-shared public trusted work
```

- 看默认区域：

```
[root@localhost ~]# firewall-cmd --get-default-zone
home
```

- 设置默认区域：

用法：firewall-cmd --set-default-zone=<zone>

```
[root@localhost ~]# firewall-cmd --set-default-zone=internal
success
```

- 查看指定区域的详细信息，不指定区域查看的是默认区域：

用法：firewall-cmd [--zone=<zone>] --list-all

```
[root@localhost ~]# firewall-cmd --zone=home --list-all
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

- 查看当前活动区域：

```
[root@localhost ~]# firewall-cmd --get-active-zones
internal
  interfaces: ens33
```

- 临时为指定区域添加网络接口：

用法：

```
firewall-cmd [--zone=<zone>] --add-interface=<interface>
```

如果网络接口已经被其他区域占有，则会报错。

```
[root@localhost ~]# firewall-cmd --zone=home --add-interface=ens33
Error: ZONE_CONFLICT: 'ens33' already bound to a zone
```

将其他区域占有的网络接口移除，再次添加，成功。

```
[root@localhost ~]# firewall-cmd --zone=home --add-interface=ens33
success
```

- 临时更改指定区域的网络接口：

用法：

```
firewall-cmd [--zone=<zone>] --change-interface=<interface>
```

更改会直接更改网络接口，不管网络接口是否被占有。

```
[root@localhost ~]# firewall-cmd --zone=home --change-interface=ens33
success
```

- 临时删除指定区域的网络接口：

用法：

```
firewall-cmd [--zone=<zone>] --remove-interface=<interface>
```

如果网络接口已经没被区域占有，则会报错。

```
[root@localhost ~]# firewall-cmd --zone=internal --remove-interface=ens33
Error: ZONE_CONFLICT: remove_interface(internal, ens33): zoi='home'
[root@localhost ~]# firewall-cmd --zone=home --remove-interface=ens33
success
```

⚠ **注：若要永久保存更改，请添加 --permanent 选项。**

10.3.1.4 服务管理

- 查看 firewall 提供的服务：

```
[root@localhost ~]# firewall-cmd --get-services
amanda-client amanda-k5-client amqp amqps apcupsd audit bacula bacula-client bb bgp bitcoin bitcoin-rpc bitcoin-testnet bitcoin-testnet-rpc bittorrent-lsd ceph ceph-mon cfengine cockpit collectd condor-collector ctdb dhcp dhcpv6 dhcpv6-client distcc dns dns-over-tls docker-registry docker-swarm dropbox-lansync elasticsearch etcd-client etcd-server finger foreman foreman-proxy freeipa-4 freeipa-ldap freeipa-ldaps freeipa-replication freeipa-trust ftp galera ganglia-client ganglia-master git grafana gre high-availability http https imap imaps ipp ipp-client ipsec irc ircs iscsi-target isns jenkins kadmin kdeconnect kerberos kibana klogin kpasswd kprop kshell kube-apiserver ldap ldaps libvirt libvirt-tls lightning-network llmnr managesieve matrix mdns memcache minidlna mongodb mosh mountd mqtt mqtt-tls ms-wbt mssql murmur mysql nbd nfs nfs3 nmea-0183 nrpe ntp nut openvpn ovirt-imageio ovirt-storageconsole ovirt-vmconsole plex pmcd pmproxy pmwebapi pmwebapis pop3 pop3s postgresql privoxy prometheus proxy-dhcp ptp pulseaudio puppetmaster quassel radius rdp redis redis-sentinel rpc-bind rquotad rsh rsyncd rtmp rtsp salt-master samba samba-client samba-dc sane sip sips slp smtp smtp-submission smtps snmp snmptrap spideroak-lansync spotify-sync squid ssdp ssh steam-streaming svdrp svn syncthing syncthing-gui synergy syslog syslog-tls
```

```
telnet tentacle tftp tftp-client tile38 tinc tor-socks transmission-client upnp-client vdsm vnc-server wbem-http wbem-https wsman wsmans xdmcp xmpp-bosh xmpp-client xmpp-local xmpp-server zabbix-agent zabbix-server
```

- 查看区域允许访问的服务，不指定区域则查看默认区域：

用法：

```
firewall-cmd [--zone=<zone>] --list-services
```

```
[root@localhost ~]# firewall-cmd --zone=home --list-services
cockpit dhcpv6-client mdns samba-client ssh
```

- 区域的服务：

用法：

```
firewall-cmd [--zone=<zone>] --[选项]-service=<service>
```

常见选项说明如下表所示：

序号	选项	说明
1	add	添加服务
2	remove	移除服务

```
[root@localhost ~]# firewall-cmd --zone=home --add-service=http
success
[root@localhost ~]# firewall-cmd --zone=home --remove-service=http
success
```

⚠ **注：若要永久保存更改，请添加 --permanent 选项。**

10.3.1.5 端口管理

- 区域的端口：

用法：firewall-cmd [--zone=<zone>] --add-port=<port>

常见选项说明如下表所示：

序号	选项	说明
1	add	添加端口
2	remove	移除端口

⚠ **注：若要永久保存更改，请添加 --permanent 选项。**

```
[root@localhost ~]# firewall-cmd --zone=home --add-port=443/tcp
success
[root@localhost ~]# firewall-cmd --zone=home --remove-port=443/tcp
success
```

- 地址伪装：

开启：

用法：firewall-cmd [--zone=<zone>] --[选项]-masquerade

常见选项说明如下表所示：

序号	选项	说明
1	add	开启端口转发
2	remove	关闭端口转发

⚠ 注：若要永久保存更改，请添加 --permanent 选项。

```
[root@localhost ~]# firewall-cmd --zone=home --add-masquerade
success
[root@localhost ~]# firewall-cmd --zone=home --remove--masquerade
usage: see firewall-cmd man page
firewall-cmd: error: unrecognized arguments: --remove--masquerade
[root@localhost ~]# firewall-cmd --zone=home --remove-masquerade
success
```

- 端口转发：

用法：firewall-cmd --zone=<zone> --[选项]-forward-port=[port]

常见选项说明如下表所示：

序号	选项	说明
1	add	开启端口转发
2	remove	关闭端口转发

⚠ **注：若要永久保存更改，请添加 --permanent 选项。**

例如：

将把 home 区域的 22 端口转发到 3777。

```
[root@localhost ~]# firewall-cmd --zone=home --add-forward-port=port=22:proto=tcp:toport=3777
success
[root@localhost ~]# firewall-cmd --zone=home --list-all
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
        port=22:proto=tcp:toport=3777:toaddr=
  source-ports:
```

```
  icmp-blocks:
  rich rules:
```

将把 home 区域的 22 端口转发至目的 IP：192.168.2.10 端口 3777。

```
[root@localhost ~]# firewall-cmd --zone=home --add-forward-
port=port=22:proto=tcp:toaddr=192.168.2.10:toaddr=192.168.2.10:toport=3777
success
[root@localhost ~]# firewall-cmd --zone=home --list-all
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
        port=22:proto=tcp:toport=3777:toaddr=
        port=22:proto=tcp:toport=3777:toaddr=192.168.2.10
  source-ports:
  icmp-blocks:
  rich rules:
```

10.3.1.6 直接接口

firewalld 提供了一种被称为“直接接口”（direct interface）的功能，它可以直接通过 iptables、ip6tables 和 ebtables 的规则。它适用于应用程序，而不是用户。firewalld 保持对所增加项目的追踪，所以它还能质询 firewalld 和发现由使用直接端口模式的程序造成的更改。直接端口由增加--direct 选项使用。直接端口模式适用于服务或者程序，以便在运行时间内增加特定的防火墙规则。这些规则不是永久性的。

例如：

- 添加一个名为 blacklist 的新的 ipv4 原始防火墙链：

```
[root@localhost ~]# firewall-cmd --direct --add-chain ipv4 raw blacklist
success
[root@localhost ~]# iptables -t raw -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
```

```
target     prot opt source               destination
Chain blacklist (0 references)
target     prot opt source               destination
# Warning: iptables-legacy tables present, use iptables-legacy to see them
```

删除一个名为 blacklist 的新的 ipv4 原始防火墙链：

```
[root@localhost ~]# firewall-cmd --direct --remove-chain ipv4 raw blacklist
success
[root@localhost ~]# iptables -t raw -nL
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
# Warning: iptables-legacy tables present, use iptables-legacy to see them
```

- 添加端口 tcp9000 端口：

```
[root@localhost ~]# firewall-cmd --direct --add-rule ipv4 filter INPUT 0 -p tcp --dport 9000 -j ACCEPT
success
[root@localhost ~]# iptables -t filter -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           tcp dpt:9000
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
# Warning: iptables-legacy tables present, use iptables-legacy to see them
```

- 删除端口 tcp9000 端口：

```
[root@localhost ~]# firewall-cmd --direct --remove-rule ipv4 filter INPUT 0 -p tcp --dport 9000 -j
ACCEPT
success
[root@localhost ~]# iptables -t filter -nL
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
# Warning: iptables-legacy tables present, use iptables-legacy to see them
```

10.3.1.7 富规则

通过“rich language”语法，可以用比直接接口方式更易理解的方法建立复杂防火墙规则。此外还能永久保留设置。这种语言可以用来配置分区，也仍然支持现行的配置方式。所有命令都必须以 root 用户身份运行。

- 添加一项规则：

```
[root@localhost ~]# firewall-cmd [--zone=<zone>] --add-rich-rule='rule'
```

- 移除一项规则：

```
[root@localhost ~]# firewall-cmd [--zone=<zone>] --remove-rich-rule='rule'
```

- 检查一项规则是否存在：

```
[root@localhost ~]# firewall-cmd [--zone=<zone>] --query-rich-rule='rule'
```

例如：

- 允许 http 服务：

```
[root@localhost ~]# firewall-cmd --zone=home --add-rich-rule='rule family="ipv4" source address="192.168.0.0/24" service name="http" accept'
```

- 允许来自相同网段的主机访问 TCP 端口范围在 7000 到 8000 的服务：

```
[root@localhost ~]# firewall-cmd --zone=home --add-rich-rule="rule family='ipv4' source address='192.168.10.0/24' port port='7000-8000' protocol='tcp' accept"
success
[root@localhost ~]# firewall-cmd --zone=home --list-all
home
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpv6-client mdns samba-client ssh
  ports:
  protocols:
  forward: no
  masquerade: no
  forward-ports:
        port=22:proto=tcp:toport=3777:toaddr=
        port=22:proto=tcp:toport=3777:toaddr=192.168.2.10
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="192.168.10.0/24" port port="7000-8000" protocol="tcp" accept
```

10.3.1.8 创建服务

首先，假设需要建立的服务是 RTMP（RealTimeMessagingProtocol，实时消息传输协议，该协议基于 TCP）端口号 1935。

在/etc/firewalld/services/目录中，利用现有的配置文件如 nfs.xml 作为模板。

```
[root@localhost ~]# cd /etc/firewalld/services/
```

说明：该目录中存放的是定义好的网络服务和端口参数，只用于参考，不能修改。这个目录中只定义了一部分通用网络服务。在该目录中没有定义的网络服务，也不必再增加相关 xml 定义，后续通过管理命令可以直接增加。

```
[root@localhost ~]# cp /usr/lib/firewalld/services/nfs.xml /etc/firewalld/services/
```

说明：从上面目录中将需要使用的服务的 xml 文件拷至这个目录中，如果端口有变化则可以修改文件中的数值。

下面修改 nfs.xml 为 rtmp.xml。

```
[root@localhost ~]# mv nfs.xml rtmp.xml
```

下面使用 vi 编辑器修改 rtmp.xml 文件为如下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<service>
        <short>rtmp </short>
        <description>RTMP Stream </description>
        <port protocol="tcp" port="1935"/>
    </service>
```

每一个服务定义都需要一个简短的名字、描述和端口网络用于指定需要使用的协议、端口和模块名。然后把此服务加入防火墙规则中。

最后重新加载 firewall。

```
[root@localhost ~]# firewall-cmd--reload
```

查看 firewall 服务，可以找到自己创建的服务。

```
[root@localhost ~]# firewall-cmd --get-services
```

10.4 数据安全

10.4.1 文件机密性保护

10.4.1.1 用户层的文件及文件夹加解密

长擎安全操作系统 24 提供用户层的文件及文件夹的加密服务，加密程序 filesecurity 通过调用系统加密库 openssl 的 API 实现对文件及文件夹使用对称加密算法进行加解密。目前提供支持的对称加密算法包括 SM4、AES128、aes256。用户可以通过命令 filesecurity

完成对文件及文件夹的加密，其中对文件夹的加密是通过扫描文件夹中的文件并对文件进行逐个加密。

- 软件的安装

```
[root@localhost ~]# dnf install filesecurity
```

- 查看使用帮助

```
[root@localhost ~]# filesecurity -h
usage
filesecurity [options] <the parameters required for the command>
this command is used to implement the use of SM4 and aes algorithm
to encrypt and decrypt files

options
-e <file or dir> [encrypting program]                    =encrypt the file with the encrypting program(sm4、aes128、aes256)!
-d <file or dir> [decrypting program]                    =decrypt the file with this decrypting program(sm4、aes128、aes256)
-h                                                       =show usage help
tips: user password length should be greater than or equal to 6    .The default algorithm is SM4
```

帮助参数说明：

序号	参数	说明
1	-e	对文件或者文件夹进行加密操作跟文件或文件夹的路径，加密类型为可选参数，默认使用 SM4 进行加密
2	-d	对文件或者文件夹进行解密操作跟文件或文件夹的路径，解密类型为可选参数，默认使用 SM4 进行解密
3	-h	查看帮助信息

- 对文件进行加密操作

创建示例文件 filetext.txt，填入内容"hello"并查看内容，然后使用命令 filesecurity 对文件进行加密，默认使用 SM4 进行加密，需要重复 2 次密码输入进行密码确认，输入的密码不会回显到屏幕，便于密码的保护，加密后查看文件内容。

```
[root@localhost ~]# touch file.txt
[root@localhost ~]# echo "hello" >file.txt
[root@localhost ~]# cat file.txt
hello
[root@localhost ~]# filesecurity -e file.txt
please input user key(len>=6):
please input user key again(len>=6):
File encryption successful:file.txt
```

```
[root@localhost ~]# cat file.txt
▒
`▒▒▒▒#▒▒▒3▒▒
```

- 对文件进行解密操作

使用命令对文件进行解密操作，输入加密时所使用的密码，解密完成后查看解密后的结果，解密密码同样不会显到屏幕。

```
[root@localhost ~]# filesecurity -d file.txt
please input user key(len>=6):
File decryption successful:file.txt
[root@localhost ~]# cat file.txt
hello
```

10.4.1.2 可信文件加密系统

可信文件加密系统，利用 ecryptfs 文件系统实现，利用 TPM2.0 可信芯片和国密算法，提供密码合规以及安全可信的透明加密文件系统功能。

可信文件加密系统使用 Linux 内核的密钥环服务、国密内核加密 API 以及可信平台模块 TPM2.0，给终端用户提供无缝的认证特征管理。可信文件加密系统应用于企业环境，它不需要对内核做任何改动，只是作为一个内核模块，利用用户空间工具执行密钥管理的功能。

在使用可信文件加密系统确保主机中存在 TPM2.0 可信平台模块。

1) root 用户登录系统，创建国密 SM2 可信存储根密钥 SRK

```
[root@localhost ~]# tpm2_createsrk
```

2) root 用户登录系统，创建国密 SM4 用户级加解密密钥

```
[root@localhost ~]# tpm2_create_user_key
```

3) root 用户登录系统，创建需要加密的目录

比如在/root/目录下创建 crypt 目录：

```
[root@localhost ~]# mkdir crypt
```

4) 挂载可信加密文件系统到需要加密的目录下

比如挂载可信加密文件系统到/root/crypt 目录下：

```
[root@localhost ~]# mount -t ecryptfs -o key=passphrase:tpm2_user_key /root/crypt/ /root/crypt/
```

执行完上述命令后，可以看到多种密码算法组合，选择国密算法加密组合的数字选项，此测试用例选择 sm4 算法，即输入“7”后按回车键。

5) 执行完以上命令，即可以在需要加密的目录下进行文件的操作，内核会透明地对此目录下的文件进行加密保护。

10.4.2 程序可信完整性保护

wlm 是一款针对系统文件进行完整性校验的工具，能有力防范非法用户对系统文件和数据进行任何形式的篡改、增删等操作。wlm 工具遵循策略文件内预设的规则，针对目标文件重新计算其哈希值，并将新生成的哈希值与存储在内核中的原始哈希值进行比对核实。当二者精确吻合时，则可确认文件未遭受任何变更；反之，若发现两者存在不一致，则表明文件已被非法修改。如果启用可信恢复且将目标文件加入白名单中，则会将文件恢复到未修改之前。

⚠ **注意：长擎安全操作系统 24 中，程序可信度量管理由安全管理员负责，wlm 配置和相关命令仅限安全管理员使用。**

10.4.2.1 查看 wlm 的状态

查看 wlm 状态的命令如下：

```
[secadm@localhost ~]$ wlm -s
wlm status:       enabled
wlm audit:        deny
wlm exec:         permissive
wlm lib:          permissive
wlm file:         permissive
wlm module:       permissive
wlm recovery:     enabled
```

10.4.2.2 开启和关闭

开启 wlm 功能的命令如下：

```
[secadm@localhost ~]$ wlm enable
```

关闭 wlm 功能的命令如下：

```
[secadm@localhost ~]$ wlm disable
```

10.4.2.3 加载策略或配置

使用 wlm load 命令加载策略或配置

用法：wlm load {options}

常见选项说明如下表所示：

序号	选项	说明
1	-c	加载/etc/wlm/config 配置文件
2	-p	加载已度量的文件，通常使用 label 时已经 load 过，load 通常只在开机时服务自动调用加载

例如，修改过/etc/wlm/config 下配置文件后，重新载入配置文件，命令如下：

```
[secadm@localhost ~]$ wlm load -c
```

10.4.2.4 度量

使用 wlm label 度量文件，可以将文件加入可信白名单中

用法：wlm label {options}

常见选项说明如下表所示：

序号	选项	说明
1	-a	将指定文件添加到白名单中
2	-d	将指定文件从白名单中移除
3	-C	清空所有已添加的项目

例如，添加一个文件到白名单当中，命令如下：

```
[secadm@localhost ~]$ wlm label -a ./a
```

例如，从白名单中删除一个文件，命令如下：

```
[secadm@localhost ~]$ wlm label -d ./a
```

例如，清空白名单，命令如下

```
[secadm@localhost ~]$ wlm label -C
```

10.4.2.5 检索数据

使用 wlm search 检索通过 wlm label 添加进白名单的项目

用法：wlm search {options}

常见选项说明如下表所示：

序号	选项	说明
1	-n	检索指定的文件
2	-e	筛选出可执行文件
3	-l	筛选出库文件
4	-m	筛选出模块文件
5	-s	筛选出脚本

例如，从白名单中检索 1.txt，命令如下：

```
[secadm@localhost ~]$ wlm search -n 1.txt
```

例如，列出白名单中 exe 类型的项目，命令如下：

```
[secadm@localhost ~]$ wlm search -e
```

10.4.2.6 容器管理

使用 wlm_podman 和 wlm_docker 命令实现对 podman 和 docker 容器的管理功能

- wlm_podman 命令

  用法：wlm_podman {commands}

常见命令说明如下表所示：

序号	命令	说明
1	load	将 podman 镜像策略加载到安全内核
2	label	将 podman 的镜像策略度量到中，并加载到内核中
3	search	检索 podmna 镜像策略标记到数据库中的文件
4	status	列出导入到安全内核的镜像策略
5	setmax	设置度量镜像策略的最大值

(1) 加载

wlm_podman load 能将生成的规则库导入系统内核中，不过使用 wlm_podman label 后会自动导入，无需重复 load，通常在开机时由服务自动导入。

用法：wlm_podman load

(2)容器度量

wlm_podman label 命令能导入镜像策略，将镜像策略中的文件导入白名单数据库。

用法：wlm_podman label {options}

常见选项如下表所示：

序号	选项	说明
1	-a	将指定的已度量的 podman 镜像中的指定文件或目录添加到白名单中
2	-A	度量 podman 的所有的镜像策略，并加载到内核中
3	-n	度量指定 podman 的镜像策略，并加载到内核中
4	-d	将指定的已度量的 podman 镜像策略的指定文件从白名单中移除
5	-C	删除指定的已度量 podman 镜像策略

例如将一个 docker.io/library/busybox:latest 镜像策略导入到内核中，命令如下：

```
[secadm@localhost ~]$ wlm_podman label -n docker.io/library/busybox:latest
```

例如将 docker.io/library/busybox:latest 中的/bin/xz 文件导入白名单中，命令如下：

```
[secadm@localhost ~]$ wlm_podman label -a /bin/xz -n docker.io/library/busybox:latest
```

例如将 docker.io/library/busybox:latest 中的/bin/xz 文件从白名单移除，命令如下：

```
[secadm@localhost ~]$ wlm_podman label -d /bin/xz -n docker.io/library/busybox:latest
```

例如将 podman 的所有镜像导入内核中，命令如下：

```
[secadm@localhost ~]$ wlm_podman label -A
```

例如删除所有指定的 podman 镜像策略，命令如下：

```
[secadm@localhost ~]$ wlm_podman label -C -n docker.io/library/busybox:latest
```

(3)检索数据

wlm_podman search 命令能检索出指定已度量 podman 镜像策略添加进白名单的文件

用法：wlm_podman search {options}

常用选项如下表：

序号	选项	说明
1	-n	指定要检索的已度量 podman 镜像策略
2	-e	筛选出可执行文件
3	-l	筛选出库文件
4	-m	筛选出模块文件
5	-s	筛选出脚本

例如检索 docker.io/library/busybox:latest 中添加进白名单的文件，命令如下：

```
[secadm@localhost ~]$ wlm_podman search -n docker.io/library/busybox:latest
```

例如检索 docker.io/library/busybox:latest 中添加进白名单的可执行文件，命令如下：

```
[secadm@localhost ~]$ wlm_podman search -n docker.io/library/busybox:latest -e
```

(4) 查看已度量的 podman 镜像策略

使用 wlm_podman status 命令可以查看所有已度量的镜像策略，docker 的已度量镜像策略也会列出

方法：wlm_podman status

例如查看所有已度量的镜像策略，命令如下：

```
[secadm@localhost ~]$ wlm_podman status
```

(5) 设置度量镜像策略的最大值

使用 wlm_podman setmax 命令能够设置可度量的镜像策略的最大值

方法：wlm_podman setmax -n

例如将可度量的镜像策略的最大值设为 500，命令如下：

```
[secadm@localhost ~]$ wlm_podman setmax -n 500
```

wlm_docker 命令

用法：wlm_docker {commands}

常见命令如下表：

序号	命令	说明
1	load	将 docker 镜像策略加载到安全内核
2	label	将 docker 的镜像策略度量到中，并加载到内核中

3	search	检索 docekr 镜像策略标记到数据库中的文件
4	status	列出导入到安全内核的镜像策略
5	setmax	设置度量镜像策略的最大值

wlm_docker 命令和选项与 wlm_podman 命令相同，请参考 wlm_podman 命令的使用。

10.4.3 强制访问控制

在计算机安全领域，强制访问控制（Mandatory Access Control, MAC）是一种由操作系统内核强制施行的访问控制机制。其主要目标是限制主体（例如进程或线程）对客体（例如文件、目录、端口、共享内存段或 I/O 设备等资源）执行操作的能力，确保系统在任何条件下能够根据预先定义的策略规则实施访问控制。

1) 在长擎安全操作系统 24 中，主体和客体都拥有一组安全属性，每当主体尝试访问客体时，操作系统内核会强制检查这些属性并应用预定义的授权规则，以决定是否允许访问。任何主体对客体的操作都会根据系统中定义的授权策略（通常称为安全策略）进行判断，这种访问控制机制能够防止未经授权的操作。

- 主体：指发起访问操作的实体，通常是一个进程或线程。
- 客体：指被访问的资源，可能包括文件、目录、设备端口、进程以及其他系统资源。长擎安全操作系统 24 通过细粒度的策略规则来管理这些客体的访问权限。

2) 长擎安全操作系统 24 的强制访问控制使用安全上下文来标记每个主体和客体。这些安全上下文信息通过标记机制保存在系统中的所有对象上，确保每次访问请求都能通过安全上下文匹配进行验证。安全上下文包括以下几个组成部分：

- 用户：指操作系统中的安全身份。
- 角色：定义用户所扮演的特定角色，并限制该角色能够执行的操作。
- 类型：对应于特定的进程或资源类型，决定哪些主体能够访问哪些客体。
- 安全级别（如果使用 MLS 策略）：多级安全模型中的安全级别，定义了不同资源的敏感级别、类型级别和完整性级别。

3) 域转换是长擎安全操作系统 24 强制访问控制中的一个关键概念，它控制着进程在不同安全域之间的切换能力。每个进程在系统中都运行于特定的安全域，这个域决定了进程能够访问哪些资源。当一个进程执行某些操作或被触发执行新的任务时，它可能需要从当前域转换到另一个具有不同权限的域，这个过程就是域转换。通过域转换，系统可以确保进程在不同安全上下文之间运行时不会违反访问控制策略。域转换的过程必须满足以下三个条件：

- 当前域的转换权限：当前进程所在的域必须具备转换到目标域的权限。
- 目标域的可访问性：目标域必须符合系统策略的规定，即系统策略必须明确允许该进程切换到目标域。
- 执行的合法性：进行域转换的操作（如执行某个文件）必须被视为安全且合法，即文件或操作本身必须具备适当的安全标签和属性。

10.4.3.1 SELinux 基础策略

长擎安全操作系统 24 提供了三种主要的 SELinux 安全策略模型：MLS（多级安全模型）、Targeted（目标安全模型）、Minimum（最小安全模型）。这三种策略是根据不同的需求，提供了针对性的访问控制机制，以确保操作系统在不同安全场景下的安全性。

- MLS（多级安全模型）是针对高度安全敏感环境设计的，特别适用于需要保护多层次敏感数据的系统，并且通过对资源打上多个类别标签和完整集属性，提供了灵活的访问控制机制。该模型主要通过对主体（进程）和客体（文件、目录等）进行严格的安全级别划分来控制信息流动。
- Minimum（最小安全模型）是长擎安全操作系统 24 的可选安全策略模型，只针对用户需求添加最小安全策略，不影响其他进程的访问控制限制。
- Targeted（目标安全模型）是长擎安全操作系统 24 的可选安全策略模型，它专注通过对主体（进程）和客体（文件、目录等）进行严格的安全级别划分来控制信息流动。

10.4.3.2 安全策略模式切换

长擎安全操作系统 24 默认在启动时启用强制模式（Enforcing），为系统提供严格的安全控制，确保未授权的操作被阻止。为了满足不同安全需求，系统支持通过安全管理员查询和修改安全策略模式，并允许通过修改引导配置来禁用安全策略。

1) 查询当前安全策略模式只能由安全管理员 secadm 用户执行。secadm 用户可通过以下命令来查看系统的安全状态：

- cyse status 或 sestatus 命令：在此命令的输出中，可以看到系统是否启用了安全控制模块、当前的安全策略模式（如 enforcing 强制模式），以及系统正在使用的策略类型（如 MLS）等。

```
[secadm@localhost ~]$ cyse status 或 sestatus
cyse status:                    enabled
SELinuxfs mount:                /sys/fs/selinux
SE root directory:              /etc/cyse
Loaded policy name:             mls
```

```
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     denied
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

- getenforce 命令：安全管理员使用该命令可简单显示当前的安全模式状态，输出为 Enforcing 时表示系统处于强制模式，输出为 Permissive 时表示系统处于宽容模式。

```
[secadm@localhost ~]$ getenforce
Enforcing
[secadm@localhost ~]$ getenforce
Permissive
```

2) 修改系统的安全策略模式只能由安全管理员 secadm 用户操作。有以下两种方法可修改系统安全策略模式：

- setenforce 或 cyse setmode 命令：安全管理员使用该命令可以临时修改当前系统的安全模式，setenforce 0 或 cyse setmode --permissiv 表示临时设置安全模式为宽容模式，setenforce 1 或 cyse setmode --enforce 表示临时设置安全模式为强制模式。当系统重启时恢复至默认安全模式。

3) 设置成宽容模式：

```
[secadm@localhost ~]$ setenforce 0
[secadm@localhost ~]$ cyse setmode --permissive
```

4) 设置成严格模式：

```
[secadm@localhost ~]$ setenforce 1
[secadm@localhost ~]$ cyse setmode --enforce
```

- 修改配置文件/etc/cyse/config：安全管理员可以修改 CYSE=enforcing、permissive 来配置系统默认安全模式为强制模式、宽容模式：

```
# This file controls the state of CYSE on the system.
# CYSE= can take one of these three values:
#       enforcing - CYSE security policy is enforced.
#       permissive - CYSE prints warnings instead of enforcing.
#       disabled - No CYSE policy is loaded.
CYSE=enforcing
# CYSETYPE= can take one of these three values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
```

```
#       mls - Multi Level Security protection.
CYSETYPE=mls
```

5) 禁用安全策略：禁用安全策略有以下两种方法：

- 修改配置文件/etc/cyse/config：安全管理员可以修改 CYSE=disabled 来配置系统默认安全模式为禁用模式，重启后生效：

```
# This file controls the state of CYSE on the system.
# CYSE= can take one of these three values:
#       enforcing - CYSE security policy is enforced.
#       permissive - CYSE prints warnings instead of enforcing.
#       disabled - No CYSE policy is loaded.
CYSE=disabled
# CYSETYPE= can take one of these three values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
CYSETYPE=mls
```

- 修改内核启动参数：系统启动时进入 grub 菜单，输入 grub 账号密码后，修改启动参数：selinux=0 或者 cyse=0 时表示禁用安全策略：

```
load_video
set gfx_payload=keep
insmod gzio
linux ($root)/vmlinuz-5.10.92-24.xc.cy8.x86_64 root=/dev/mapper/cslo-root ro c\
      rashkernel=auto resume=/dev/mapper/cslo-swap rd.lvm.lv=cslo/root rd.lvm.lv=csl\
o/swap rd.shell=0 loglevel=0 cyse=0
initrd    ($root)/initramfs-5.10.92-24.xc.cy8.x86_64.img


          Press Ctrl-x to start, Ctrl-c for a command pormpt or Escape to
          discard edits and return to the memu. Pressing Tab lists possible
Completions.
```

⚠ **注意：selinux=1 或者 cyse=1 或者默认时表示启动安全策略。**

10.4.3.3 策略配置命令

为了方便用户对安全策略进行配置，长擎安全操作系统 24 提供了以下命令来用于灵活调整当前的安全策略：

10.4.3.3.1 查看安全上下文命令

在系统中，常用命令的 -Z 参数提供了查询文件或进程安全上下文的功能。例如，使用 ls -Z、id -Z 和 ps -Z 等命令，配合 -Z 选项后可以显示相关文件或进程的安全上下文信息：

- 查看/usr/sbin/getenforce 文件的安全上下文：

```
[secadm@localhost ~]$ ls -Z /usr/sbin/getenforce
system_u:object_r:cyse_exec_t:s0:iU /usr/sbin/getenforce
```

- 查看安全管理员 secadm 的安全上下文：

```
[secadm@localhost ~]$ id -Z
secadm_u:secadm_r:secadm_t:s0-s15:c0.c1023:iU
```

- 查看 systemd-logind 进程的安全上下文：

```
[secadm@localhost ~]$ ps -Z aux|grep logind|grep -v "grep"
system_u:system_r:systemd_logind_t:s0-s15:c0.c1023:iU root 840 0.0  0.2 73948 8456 ? Ss 09:48 0:00 /usr/lib/systemd/systemd-logind
```

10.4.3.3.2 修改安全上下文命令

安全管理员可以使用以下命令修改文件的安全上下文标记，包括 chcon、setfiles 和 restorecon 等命令。

- 使用 chcon 修改文件安全上下文：

```
[secadm@localhost ~]$ ll -Z /usr/bin/teamd
-rwxr-xr-x. 1 root root system_u:object_r:NetworkManager_exec_t:s0:iU 164080 5 月  9 15:10 /usr/bin/teamd
[secadm@localhost ~]$ chcon -u secadm_u -t bin_t -l s1:c1:iL /usr/bin/teamd
[secadm@localhost ~]$ ll -Z /usr/bin/teamd
-rwxr-xr-x. 1 root root secadm_u:object_r:bin_t:s1:c1:iL 164080 5 月  9 15:10 /usr/bin/teamd
```

chcon 用法：

序号	选项	说明
1	-u, --user=USER	设置目标安全上下文的用户标签
2	-r, --role=ROLE	设置目标安全上下文的角色标签
3	-t, --type=TYPE	设置目标安全上下文的类型标签
4	-l, --range=RANGE	设置目标安全上下文的范围
5	--no-preserve-root	不要特别对待 '/'（默认）
6	--preserve-root	递归操作 '/' 时失败
7	--reference=RFILE	使用参考文件的安全上下文，而不是指定 CONTEXT

8	-R, --recursive	递归操作文件和目录
9	-v, --verbose	为处理的每个文件输出诊断
10	-L	遍历遇到的每个指向目录的符号链接
11	-P	不遍历任何符号链接（默认）
12	--help	显示此帮助信息并退出
13	--version	显示版本信息并退出

⚠ **注意：chcon 修改文件的安全上下文是临时的，系统重启会失效**

- setfiles 命令用于根据 SELinux 安全策略中的文件上下文（file contexts）为系统中的文件或目录设置安全上下文。它可以通过预定义的策略文件来批量标记文件和目录的安全上下文，例如使用/etc/cyse/mls/contexts/files/file_contexts 内容重新设置/usr/bin 的安全上下文：

```
[secadm@localhost ~]$ setfiles /etc/cyse/mls/contexts/files/file_contexts /usr/bin/
```

- restorecon 命令用于恢复文件或目录的安全上下文到默认值。它根据当前系统的文件上下文策略来恢复某个文件或目录的安全上下文，例如恢复/usr/bin/目录的安全上下文：

```
[secadm@localhost ~]$ restorecon -RvF /usr/bin/
Relabeled /usr/bin from system_u:object_r:bin_t:s0:iU to system_u:object_r:bin_t:s0
Relabeled /usr/bin/idiag-socket-details from system_u:object_r:bin_t:s0:iU to
system_u:object_r:bin_t:s0
      ……
```

10.4.3.3.3 配置默认安全上下文命令

安全管理员可以使用 cyse 命令配置文件的安全上下文。

- 使用 cyse fcontext 可以设置默认安全上下文，例如设置文件夹（/www）及其子文件及文件夹的默认安全上下文，设置完成后查看已设置的规则：

```
[secadm@localhost ~]$ cyse fcontext -a -t httpd_sys_content_t -r s0:c3:iL "/www(/.*)?"
[secadm@localhost ~]$ cyse fcontext -l -C
SEfcontext              类型              上下文
/www(/.*)?              all files         system_u:object_r:httpd_sys_content_t:s0:c3:iL
```

- 使用 cyse fcontext 可以修改默认安全上下文，例如修改之前设置的文件夹（/www）及其子文件及文件夹的级别 c 属性和完整集 i 属性，修改完成后查看已设置的规则：

```
[secadm@localhost ~]$ cyse fcontext -m -t httpd_sys_content_t -r s0:c4:iM "/www(/.*)?"
[secadm@localhost ~]$ cyse fcontext -l -C
```

```
SEfcontext              类型            上下文
/www(/.*)?              all files       system_u:object_r:httpd_sys_content_t:s0:c4:iM
```

10.4.3.3.4 用户、角色和域的关系

在长擎安全操作系统 24 中，用户、角色和域的管理遵循类似的安全模型，以确保系统的高安全性和精细化访问控制。该模型通过安全标识符和策略定义不同用户在系统中的行为权限。

- 用户、角色和安全策略映射：

在长擎安全操作系统 24 中，登录过程（例如通过 login 或 sshd）会将系统用户映射到安全策略中定义的安全用户。当一个系统用户登录时，若该用户的标识符与安全策略中的某个安全用户标识符完全匹配，则此标识符被用于设置登录进程的安全上下文。安全管理员可以使用 cyse 命令来配置这种映射关系。例如用户 user 会作为安全策略用户 user_u 登录。

```
[secadm@localhost ~]$ cyse login -a -s user_u user
```

通过运行 cyse login -l 命令，可以查看当前系统中用户的安全策略映射关系。输出结果显示系统中每个登录名与对应的 SELinux 用户之间的关联关系，以及该用户在系统中的多级安全（MLS）和多类别安全（MCS）范围及适用的服务：

```
[secadm@localhost ~]$ cyse login -l
登录名                  SE User         MLS/MCS 范围            服务
__default__             user_u          s0-s0                   *
auditadm                auditadm_u      s0-s15:c0.c1023         *
root                    sysadm_u        s0-s15:c0.c1023         *
secadm                  secadm_u        s0-s15:c0.c1023         *
user                    user_u          s0:iU                   *
```

- 配置用户与角色：在长擎安全操作系统 24 中，每个角色代表一组域或类型的集合，从而对用户在系统中的权限和操作范围进行定义和控制。安全管理员可以通过 cyse 命令为某个用户分配多角色。例如创建一个可以作为角色 user_r 和 system_r 的安全策略用户 test1_u，并设置用户默认安全级别为 s0，指定用户有效安全范围为 s0-s12:c0.c1023:iL：

```
[secadm@localhost ~]$ cyse user -a -R user_r -R system_r -L s0 -r s0-s12:c0.c1023:iL test1_u
[secadm@localhost ~]$ cyse user -l
                        标记中          MLS/            MLS/
SE User         前缀    MCS             级别            MCS 范围                SE Roles
auditadm_u      user    s0:iU           s0-s15:c0.c1023:iU                      auditadm_r
secadm_u        user    s0:iU           s0-s15:c0.c1023:iU                      secadm_r
```

```
staff_u          user          s0:iU          s0-s15:c0.c1023:iU                auditadm_r secadm_r staff_r
sysadm_r system_r
sysadm_u         user          s0:iU          s0-s15:c0.c1023:iU                sysadm_r
system_u         user          s0:iU          s0-s15:c0.c1023:iU                system_r
test1_u          user          s0:iU          s0-s12:c0.c1023:iL                user_r system_r
unconfined_u     user          s0:iU          s0-s15:c0.c1023:iU                auditadm_r secadm_r
staff_r sysadm_r system_r unconfined_r
user_u           user          s0:iU          s0:iU                             user_r
```

- 删除用户与角色配置：安全管理员也可通过 cyse 命令删除具体用户新增的 selinux 用户：

```
[secadm@localhost ~]$ cyse user -d test1_u
[secadm@localhost ~]$ cyse user -l
                          标记中          MLS/          MLS/
SE User          前缀          MCS  级别          MCS  范围          SE Roles

auditadm_u       user          s0:iU          s0-s15:c0.c1023:iU                auditadm_r
secadm_u         user          s0:iU          s0-s15:c0.c1023:iU                secadm_r
staff_u          user          s0:iU          s0-s15:c0.c1023:iU                auditadm_r secadm_r staff_r
sysadm_r system_r
sysadm_u         user          s0:iU          s0-s15:c0.c1023:iU                sysadm_r
system_u         user          s0:iU          s0-s15:c0.c1023:iU                system_r
unconfined_u     user          s0:iU          s0-s15:c0.c1023:iU                auditadm_r secadm_r
staff_r sysadm_r system_r unconfined_r
user_u           user          s0:iU          s0:iU                             user_r
```

- 域与访问控制

在长擎安全操作系统 24 中，角色只是域和类型集合的抽象化，用于简化安全策略的管理和用户角色的分配。系统的实际访问控制是通过进程的域（类型）和对象的域（类型）根据制定的规则来实现的。每个域类型定义了特定的权限范围，允许或限制该域中的进程对特定资源的访问。用户通过切换角色间接控制进程所属的域，从而控制其在系统中的行为和权限。

10.4.3.3.5 端口配置

长擎安全操作系统 24 中的所有端口也都被设置了安全上下文类型标签，安全管理员可使用 cyse 命令配置与查看。

- 查看端口协议以及对应的安全标签：

```
[secadm@localhost ~]$ cyse port -l
SE Port Type                      协议          端口号
```

```
afs3_callback_port_t          tcp        7001
afs3_callback_port_t          udp        7001
afs_bos_port_t                udp        7007
afs_fs_port_t                 tcp        2040
afs_fs_port_t                 udp        7000, 7005
afs_vl_port_t                 udp        7003
agentx_port_t                 tcp        705
agentx_port_t                 udp        705
amanda_port_t                 tcp        10080-10083
amanda_port_t                 udp        10080-10082
amavisd_recv_port_t           tcp        10024
amavisd_send_port_t           tcp        10025
amqp_port_t                   tcp        15672, 5671-5672
……
```

- 设置 tcp 协议 11180-11188 端口号的安全上下文标签为 http_port_thttp_port_t：

```
[secadm@localhost ~]$ cyse port -a -t http_port_t -p tcp 11180-11188
[secadm@localhost ~]$ cyse port -l -C
SE Port Type                  协议        端口号
http_port_t                   tcp        11180-11188
```

- 删除设置的端口安全上下文标签：

```
[secadm@localhost ~]$ cyse port -d -p tcp 11180-11188
```

10.4.3.3.6 网络接口配置

在长擎安全操作系统 24 中，网络接口同样被分配了安全上下文类型标签，安全管理员可以使用 cyse 命令来配置和查看网络接口的安全上下文类型。每个网络接口都具有相应的安全标签，以确保网络流量和访问权限的严格控制。

- 查看当前系统中所有网络接口的安全上下文类型：

```
[secadm@localhost ~]$ cyse interface -l
SE Interface                  上下文
lo                            system_u:object_r:lo_netif_t:s0-s15:c0.c1023:iU
设置 ens160 网络接口安全上下文标签为 netif_t：
[secadm@localhost ~]$ cyse interface -a -t netif_t -r s0-s15:c0.c1023:iH ens160
[secadm@localhost ~]$ cyse interface -l -C
SE Interface                  上下文
ens160                        system_u:object_r:netif_t:s0-s15:c0.c1023:iH
```

- 删除设置的网络接口安全上下文标签：

```
[secadm@localhost ~]$ cyse interface -d ens160
```

10.4.4 国密算法支持

密码技术已经是国家网络信息安全重要的“护城河”，而密码国产化，也就成为具有战略意义的大事。特别是在关系到国计民生的工业控制领域，采用国密算法或行业自主密码体制已经逐渐成为一种趋势，它对于维护国家主权安全、维护客户利益、保护数据安全、防止各种针对性网络攻击，推动我国信息安全产业发展具有十分重要的意义。

为实现密码算法的自主可控，防止采用国际公开算法而导致的安全风险，CQOS24 通过将国密算法默认集成到操作系统中，为应用提供安全可靠的国密算法支撑。

10.4.4.1 内核层的国密算法支撑

CQOS24 内核层提供 SM2、SM3 和 SM4 国密算法，并为用户提供两种机制进行算法调用：

1) 内核提供内核级的 crypto 接口，方便内核模块进行国密算法的功能开发；
2) CQOS24 提供 libkcapi 用户层接口，方便用户层的应用调用内核层提供的国密算法进行国密算法的功能开发。

10.4.4.2 用户层的国密算法支撑

CQOS24 用户层通过用户登录认证以及 OpenSSL 算法库提供国密算法支撑。

1) 用户登录认证支持基于国密算法 SM3 的口令认证，以及基于 Ukey 国密算法 SM2 公私密钥对身份认证技术的双因子认证；
2) OpenSSL 算法库支持的算法接口包括：SM2 国密非对称算法，包括加解密以及签名验签；SM3 国密哈希算法；SM4 国密对称算法。

10.5 应用安全

10.5.1 安全加固

安全加固是对系统进行检测与配置，降低系统受到攻击的可能性。OpenSCAP 提供了一套开源工具，可帮助用户根据预定义的安全基线对系统进行评估，并自动修复发现的问题。长擎安全操作系统 24 安全加固是基于 OpenSCAP 开源工具实现的，并实现了一系列适用于本系统的安全检查项，支持 oscap 命令行和 scap-workbench 图形用户界面（GUI）这两种工具实施安全加固。

10.5.1.1 安全加固工具

可以使用以下工具对系统执行自动化的安全加固。

- OpenSCAP

OpenSCAP 提供了 oscap 命令行工具，oscap 命令行用于对本地系统上执行配置和漏洞扫描，并且支持生成评估报告。

- SCAP 工作台

与 oscap 命令行功能相似，scap-workbench 是一种图形化工具。可以在安装了图形化界面的系统中使用 scap-workbench 执行扫描和加固。

- SCAP 安全指南

scap-security-guide 软件包为 Linux 系统提供安全策略的集合，长擎安全操作系统 24 安全基线已集成到 scap-security-guide，该软件包提供了执行安全加固所需的数据流文件。

- 脚本检查引擎（SCE）

SCE 是 SCAP 协议的扩展，其本身不是 SCAP 标准的一部分。SCE 支持使用脚本语言（如 Bash、Python 和 Ruby）编写安全加固检查项，长擎安全操作系统 24 安全基线的部分检查项使用脚本实现，实施安全加固前应安装 openscap-engine-sce 软件包。

10.5.1.2 扫描系统漏洞

oscap 命令行实用程序能够扫描本地系统，验证配置合规性内容，并根据这些扫描和评估生成报告和指南。此工具充当 OpenSCAP 库的前端，并根据它所处理的 SCAP 内容类型将其功能分组到模块（子命令）。

前提条件

- 安装 openscap-scanner、scap-security-guide 和 openscap-engine-sce

```
[root@localhost ~]# dnf install openscap-scanner scap-security-guide openscap-engine-sce -y
```

扫描本地系统以确定是否符合所选的配置文件，并将扫描结果保存到文件中：

```
[root@localhost ~]# oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_standard --report <file> /usr/share/xml/scap/ssg/content/ssg-cq24-ds.xml
```

可以生成的报告格式如下：

- 需要详细分析结果：选择 XCCDF Results（XML）。 --results <file>
- 需要长期存档或与其他工具共享：选择 ARF。 --results-arf <file>
- 需要快速、直观地查看结果：选择 HTML Report。 --report <file>
- 需要满足 DISA STIG 合规要求：选择 STIG Viewer 格式。 --stig-viewer <file>

10.5.1.3 扫描远程系统的漏洞

可以通过 SSH 协议使用 oscap-ssh 工具，通过 OpenSCAP 扫描程序检查远程系统上的漏洞。

1) 前提条件

- scap-security-guide 软件包已安装在用于扫描的系统中。
- openscap-scanner 软件包已安装在远程系统上。
- SSH 服务器在远程系统上运行。

2) 扫描远程系统上的漏洞，并将结果保存到文件中：

```
[root@localhost ~]# oscap-ssh <username>@<hostname> <port> oval eval --profile xccdf_org.ssgproject.content_profile_standard --report report_file.html /usr/share/xml/scap/ssg/content/ssg-cq24-ds.xml
```

3) 替换：

- 带有用户名和远程系统主机名的<username>@<hostname>。
- 可通过端口号<port>访问远程系统，例如 22。
- 使用 oscap 保存扫描结果的文件名<scan-report.html>。

10.5.1.4 扫描的可能结果

根据应用到 OpenSCAP 扫描的数据流和配置文件，以及系统的各种属性，每个规则可能会产生一个特定的结果。以下是可能的结果，以及其含义的简要解释：

序号	扫描结果	说明
1	Pass	扫描没有发现与此规则有任何冲突。
2	Fail	扫描发现与此规则有冲突。
3	Not checked	没有对此规则执行评估。
4	Not applicable	此规则不适用于当前配置。
5	Not selected	此规则不是配置文件的一部分。OpenSCAP 不评估此规则，也不会在结果中显示这些规则。
6	Error	扫描遇到了错误。要获得更多信息，可以输入带有--verbose DEVEL 选项的 oscap 命令。
7	Unknown	扫描遇到了意外情况。要获得更多信息，可以输入带有--verbose DEVEL 选项的 oscap 命令。

10.5.1.5 查看配置文件

在决定使用配置文件进行扫描或补救前，可以使用 oscap info 子命令列出它们并检查其详细描述。

1) 前提条件

openscap-scanner 和 scap-security-guide 软件包已安装。

2) 流程

- 列出 SCAP 安全指南项目所提供的带有安全合规配置文件的所有可用文件：

```
[root@localhost ~]# ls /usr/share/xml/scap/ssg/content/
...
```

```
ssg-cq24-ds.xml
...
```

- 使用 oscap info 子命令显示有关所选数据流的详细信息。包含数据流的 XML 文件由其名称中的 -ds 字符串表示。在 Profiles 部分，可以找到可用的配置文件及其 ID 列表：

```
[root@localhost ~]# oscap info /usr/share/xml/scap/ssg/content/ssg-cq24-ds.xml
...
        Profiles:
            Title: Standard System Security Profile for cq24
                Id: xccdf_org.ssgproject.content_profile_standard
...
```

- 从数据流文件中选择一个配置文件，并显示所选配置文件的更多详情。为此，可使用带有 --profile 选项的 oscap info ，后跟上一命令输出中显示的 ID 的最后一部分。例如，cq24 配置文件的 ID 是 xccdf_org.ssgproject.content_profile_standard，--profile 选项的值为 standard

```
[root@localhost ~]# oscap info --profile standard /usr/share/xml/scap/ssg/content/ssg-cq24-ds.xml
Document type: Source Data Stream
Imported: 2024-09-05T17:51:03

Stream: scap_org.open-scap_datastream_from_xccdf_ssg-cq24-xccdf.xml
Generated: (null)
Version: 1.3
Profile
    Title: Standard System Security Profile for cq24
    Id: xccdf_org.ssgproject.content_profile_standard

    Description: This profile contains rules to ensure standard security baseline of a cq24 system.
```

10.5.1.6 修复系统

用户可以修复系统，以与特定基准保持一致。

1) 前提条件

openscap-scanner 和 scap-security-guide 软件包已安装。

2) 步骤

- 使用带有 --remediate 选项的 oscap 命令修复系统：

```
[root@localhost ~]# oscap xccdf eval --remediate --profile
xccdf_org.ssgproject.content_profile_standard --report report.html /usr/share/xml/scap/ssg/content/ssg-cq24-
ds.xml
```

若修复远程系统：

```
[root@localhost ~]# oscap-ssh <username>@<hostname> <port> oval eval --remediate --profile xccdf_org.ssgproject.content_profile_standard --report report_file.html /usr/share/xml/scap/ssg/content/ssg-cq24-ds.xml
```

Result 为 fixed 代表该检查项修复成功，如：

```
Title    Check if superusers is set in /boot/grub2/grub.cfg or /boot/grub2/user.cfg
Rule     xccdf_org.ssgproject.content_rule_check_boot_grub_cfg_or_user_cfg_exist_superusers
Result   fixed

Title    Check if logging is done for logging - LASTLOG_ENAB
Rule     xccdf_org.ssgproject.content_rule_check_etc_login_defs_exist_LASTLOG_ENAB_yes
Result   fixed
```

- 重启系统。
- 验证，重新扫描系统漏洞。

10.5.1.7 使用 scap-workbench 图形用户界面进行安全加固

SCAP Workbench 是一个基于图形用户界面的工具，用于管理和执行 SCAP（Security Content Automation Protocol）内容的评估和修复任务。它是 OpenSCAP 工具套件的一部分，旨在简化安全配置和漏洞评估过程。通过用户友好的界面，用户可以轻松选择 SCAP 内容、配置评估参数并执行安全检查。使用 scap-workbench 对系统进行评估的过程如下：

1) 选择数据流文件

软件内置了进行评估需要的数据流文件，在运行 scap-workbench 之后选择 CQ24，之后点击 Load Content 即可。

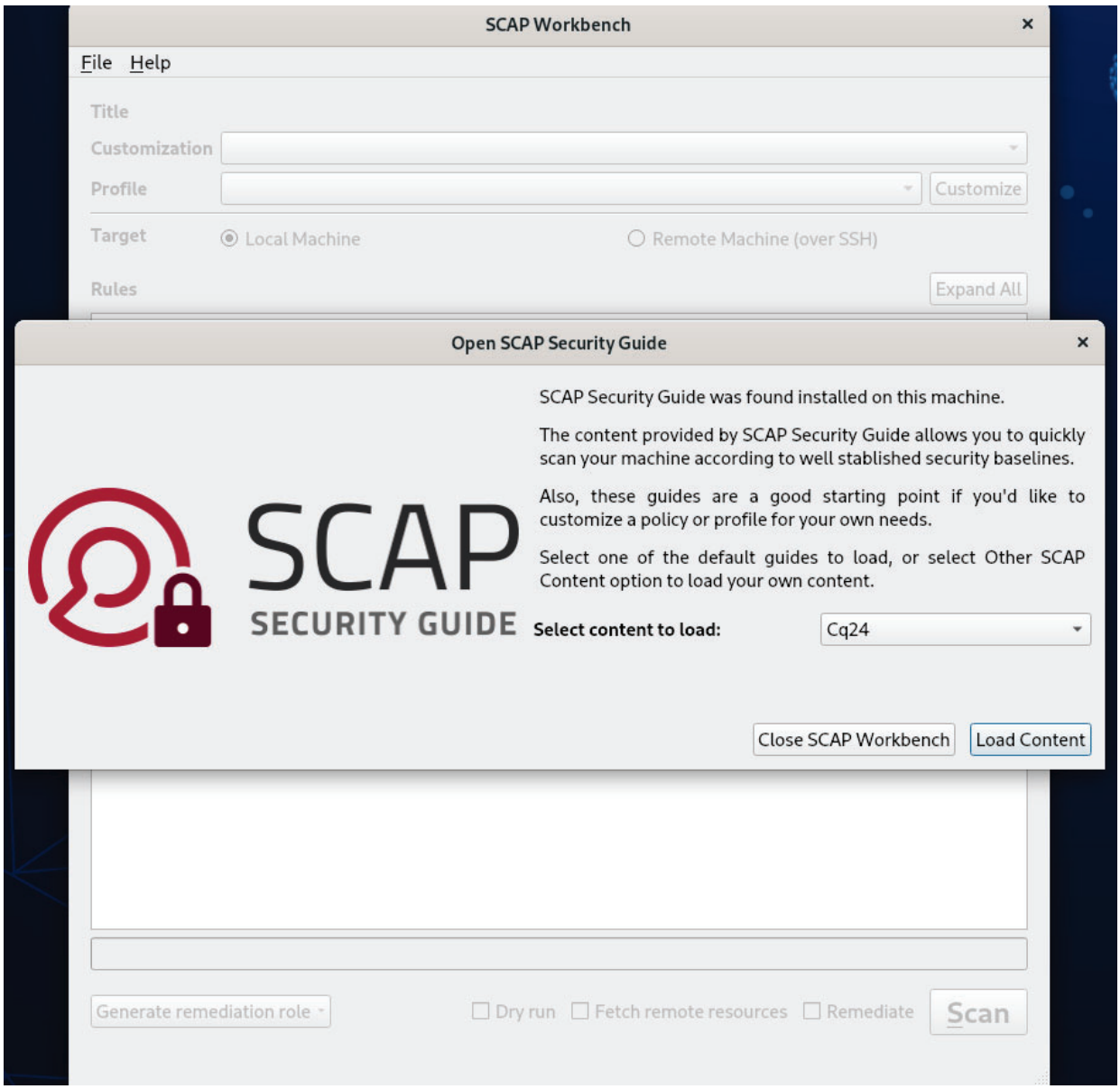

图 10-1 SCAP Workbench 启动界面

2) 评估系统

进入主界面，可以看到，需要的数据流文件已经被正确加载，在界面上方 Target 一栏可以选择在本机（Local Machine）或者远程机器（Remote Machine，通过 SSH 连接）来进行系统评估。点击右下角的 Scan 按钮，即可开始系统评估，评估结果大体如下图所示，对于通过的条目显示 pass，失败的条目显示 fail，而对于一些缺失需要的软件包的项，一般显示 notchecked，如果需要回到未进行评估的状态，点击下方 Clear 按钮。需要注意的是评估过程需要 root 身份，请以 root 身份运行程序。

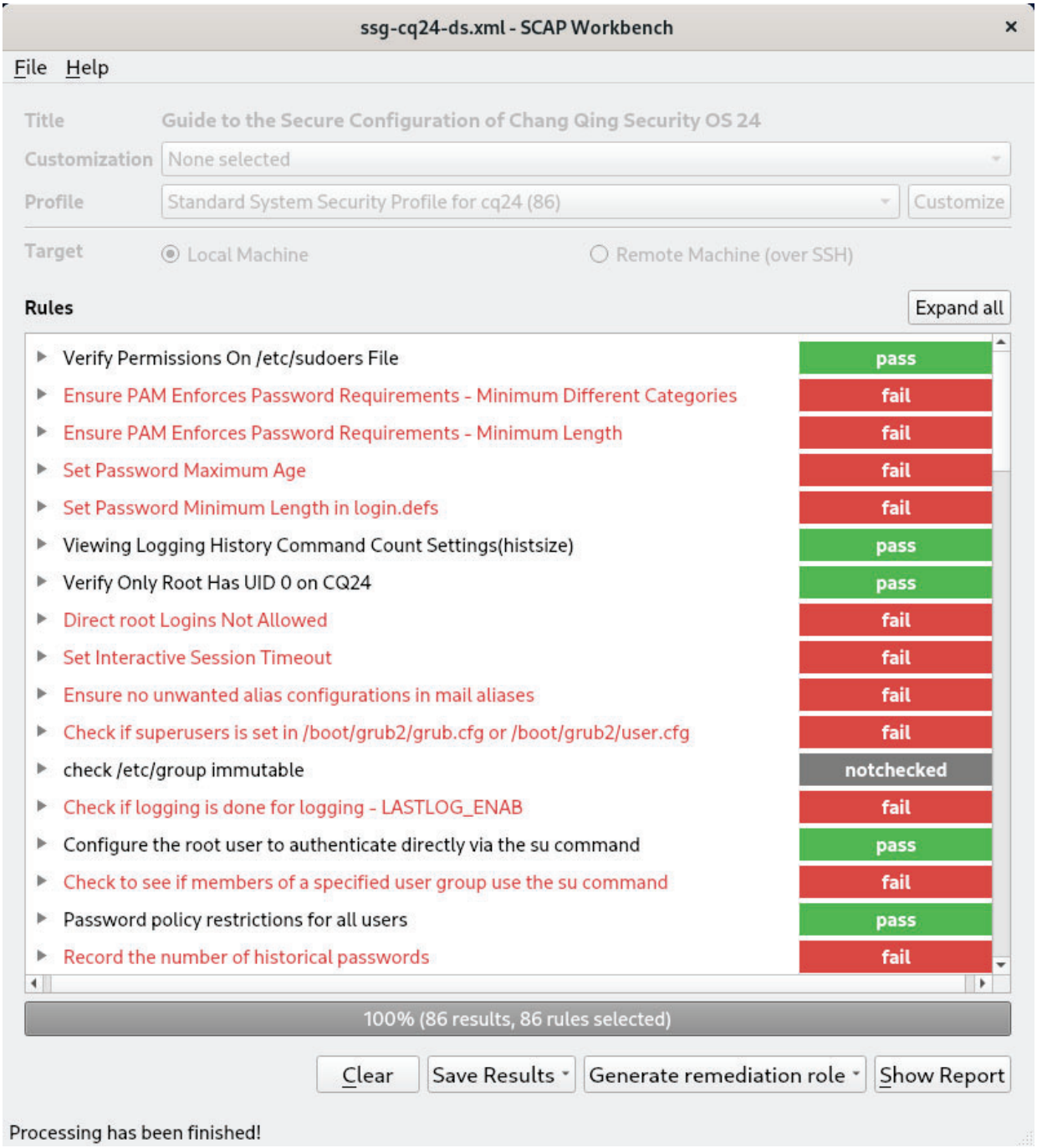

图 10-2 评估结果

3) 修复系统

同样的，使用 scap-workbench 也可以对系统评估过程中失败的条目进行修复。在点击 Scan 按钮之前，勾选左边的 Remediate 复选框，就可以对系统进行修复，基本过程是先进行系统评估，然后对评估失败的条目进行修复。这里被修复的条目右边显示 fixed。同样的，修复过程也需要以 root 身份进行，否则部分修复操作会无法进行。

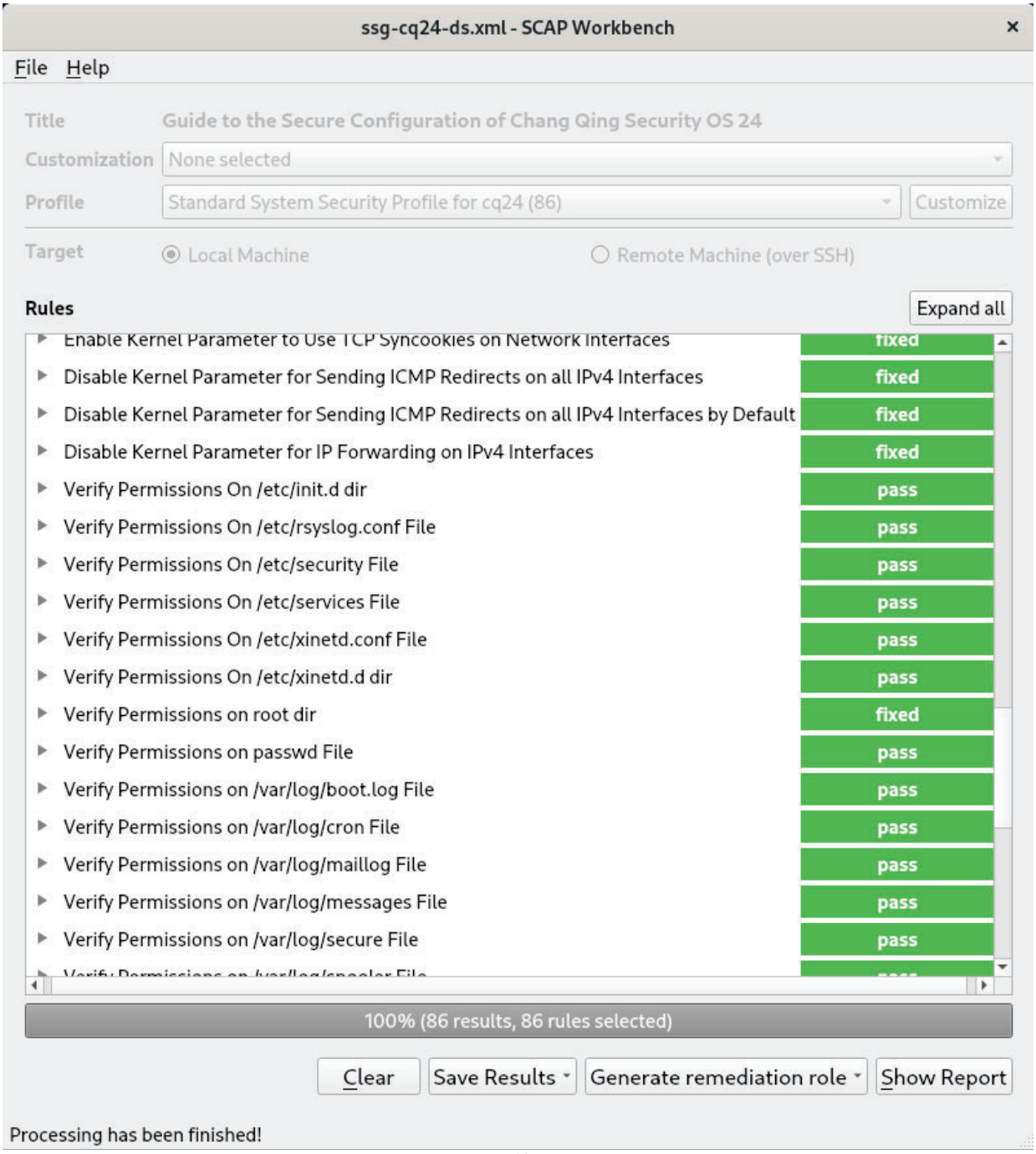

图 10-3 修复结果

对于生成评估结果的格式，可以点击下方 Save Results 按钮，有 XCCDF，ARF，HTML 三种格式，XCCDF 和 ARF 格式适用于进一步处理，HTML 格式为人类可读的格式。这里可以重复选择三个选项。如果想要将基于结果的补救（Remediation）导出到文件，请点击 Generate remediation role 按钮进行选择。

4) 使用 scap-workbench 自定义安全配置文件

使用 scap-workbench 可以自定义安全配置文件，可以更改某些规则中的参数（如最小密码长度）、选择额外的规则或删除不需要的规则来自定义安全配置文件。还可以保存它，以保证 oscap 也能够使用。具体流程如下：

- 运行 scap-workbench，选择 CQ24 安全配置文件；
- 点击右上方的 Customize 按钮，在弹出的界面中选择新的 Profile ID，点击 OK；
- 在新的窗口中点击复选框来包含和排除规则，或者点击规则本身，在右侧修改规则中的值；
- 点击 OK，左上角可以点击 File 按钮，可以选择 Save Customization Only，也可以选择 Save All 来保存自定义安全配置文件。

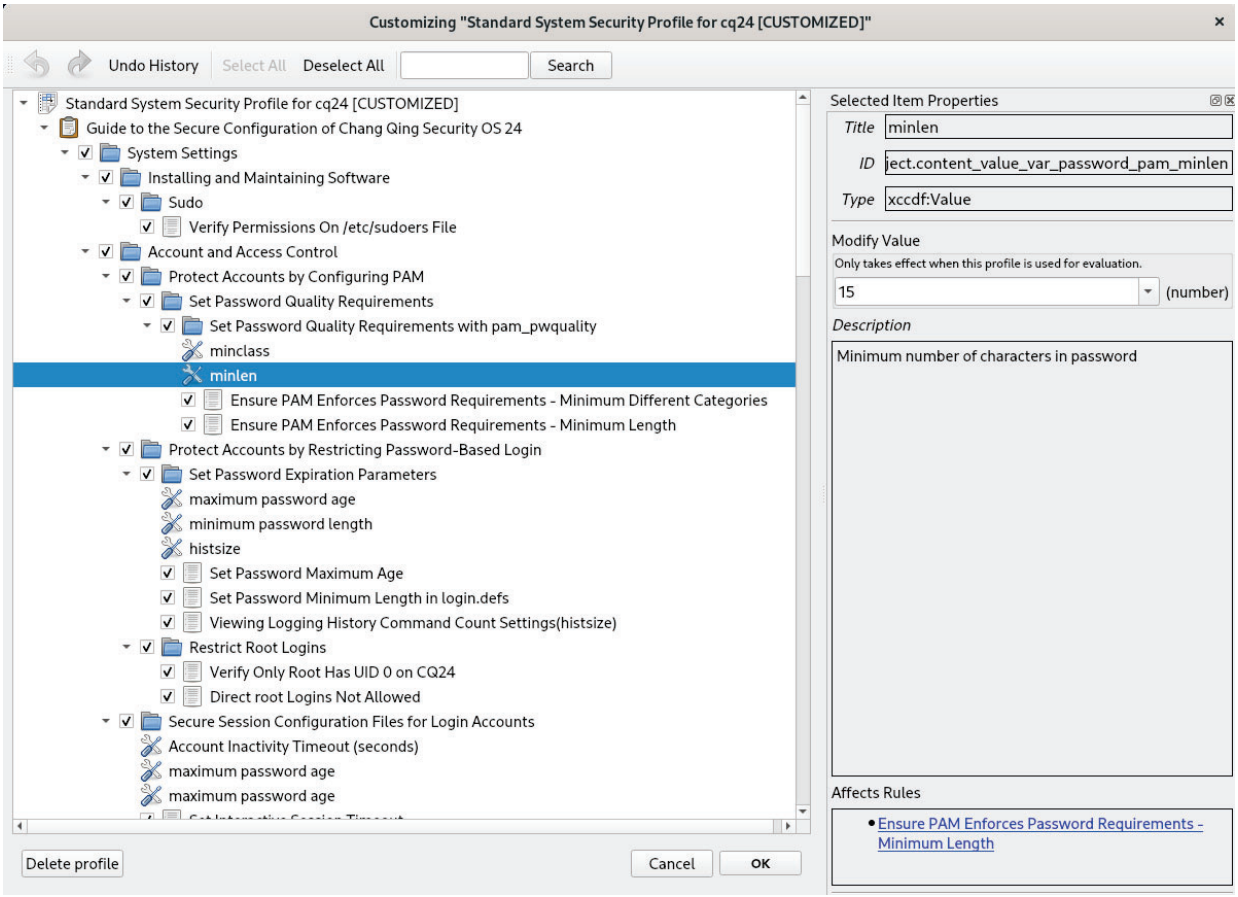

图 10-4 自定义配置界面

如果在保存时选择了 Into a directory 选项，scap-workbench 会将数据流文件保存到指定目录下，如果选择了 As RPM 选项，scap-workbench 将会创建包含数据流文件和自定义安全配置文件的 RPM 软件包，这在某些场景下很有用，比如对于向无法远程扫描的系统分发安全内容，以及为进一步处理交付内容都非常有用。

10.6 审计安全

10.6.1 安全审计简介

安全审计提供了一种记录系统安全信息的方法，可以记录系统内部发生的事件信息，并根据用户的需求，提供不同的报表功能，从而实现对系统信息的追踪、审查、统计和报告等功能。审计系统的审计信息包括可被审计的事件名称、事件状态（成功或失败）和安全信息等。

⚠ **注意：长擎安全操作系统 24 中，安全审计由审计管理员负责，审计配置和相关命令仅限审计管理员使用。**

10.6.2 安全审计系统架构

长擎安全操作系统 24 的安全审计系统基于 Linux 审计子系统框架实现，具体结构如下图所示。审计系统分为用户空间审计系统和内核空间审计系统，用户空间审计系统用来设置规则和审计系统状态、将内核审计系统传来的审计消息写入 log 文件。内核审计系统用于产生和过滤内核的各种审计消息。

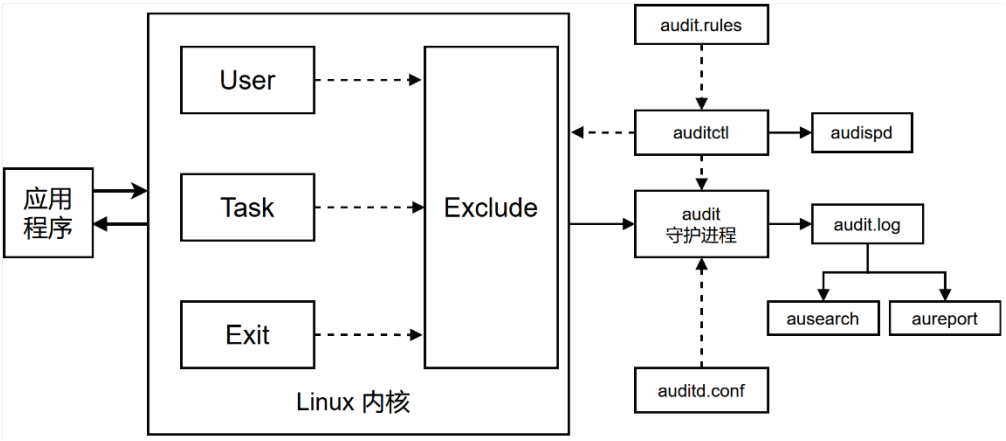

图 10-5 审计系统架构

内核组件接收用户空间应用程序的系统调用，并通过以下过滤器对其进行过滤：user、task、filesystem 或 exit。一旦系统调用通过这些过滤器中的其中一个，它将经过 exclude 过滤器，exclude 过滤器根据审计规则的配置，将其发送到审计守护进程以进行进一步处理。

用户空间审计守护进程从内核收集信息，并在日志文件中创建条目。其他审计用户空间工具与审计守护进程、内核审计组件或审计日志文件进行交互：

- auditctl 命令可以对内核中的审计系统进行控制，可以用来获取 audit 状态和添加/删除 audit 规则。

- 其余的审计工具将审计日志文件的内容作为输入，并根据用户的要求生成报告。例如，aureport 工具生成所有记录的事件的报告。

审计分发守护进程（audisp）的功能已经集成到了用户空间审计守护进程（auditd）。

10.6.3 安装 audit 软件包

要使用审计系统，必须在系统中安装了 audit 软件包。audit 软件包默认安装在长擎安全操作系统 24 中。如果没有安装这些软件包，请以 root 用户身份执行以下命令来安装审计和依赖项：

```
[root@localhost ~]# dnf install audit -y
```

10.6.4 配置审计服务

默认的 auditd 配置适用于大多数环境。如果需要修改审计服务的配置，审计守护进程的配置位于/etc/audit/auditd.conf 文件中，常用配置选项：

序号	选项	说明
1	log_file	包含审计日志文件的目录（通常为/var/log/audit/）应位于单独的挂载点上。这可以防止其他进程消耗此目录的空间，并为审计守护进程提供准确的剩余空间检测。
2	max_log_file	指定单个审计日志文件的最大大小，必须设置为充分利用保存审计日志文件的分区上的可用空间。 max_log_file参数指定最大文件大小（以MB为单位）。给出的值必须是数字。
3	max_log_file_action	在达到max_log_file中设置的限制后，决定要采取什么操作，应设置为keep_logs以防止审计日志文件被覆盖。
4	space_left	指定磁盘上剩余的可用空间量，该磁盘中触发space_left_action参数中设置的操作。必须设置一个数字，让管理员有足够的时间来响应，并释放磁盘空间。space_left值取决于生成审计日志文件的速率。 如果space_left的值指定为整数，它将解释为绝对大小（MiB）。如果该值指定为 1 到 99 之间的数字，后跟一个百分比符号（例如 5%），则审计守护进程会根据包含log_file的文件系统的大小计算绝对大小（以MB为单位）。
5	space_left_action	建议使用适当的通知方法将space_left_action参数设置为email或exec。
6	admin_space_left	指定触发admin_space_left_action参数中设置的操作的绝对最小可用空间量，必须将其设置为一个值，以便有足够的空间来记录管理员执行的操作。 此参数的数字值应小于space_left的数字值。还可以在数字后面附加一个百分比符号（例如 1%），以便审计守护进程根据磁盘分区计算数值。
7	admin_space_left_action	应设置为single来将系统置于单用户模式，并允许管理员释放一些磁盘空间。
8	disk_full_action	指定当保存审计日志文件的分区上没有可用空间时触发的操作，必须设置为halt或single。当审计无法记录事件时，这可确保系统关闭或以单用户模式运行。

序号	选项	说明
9	disk_error_action	指定当在包含审计日志文件的分区上检测到错误时触发的操作，必须设置为syslog、single或halt，具体取决于处理硬件故障的本地安全策略。
10	flush	应设置为incremental_async。它与freq参数结合使用，该参数决定了在强制与硬盘进行硬盘同步前可以将多少条记录发送到磁盘。freq参数应设置为 100。这些参数可确保审计事件数据与磁盘上的日志文件同步，同时保持良好的活动性能。

10.6.5 启动和控制审计服务

配置 auditd 后，启动审计服务收集审计信息存储到日志文件。运行下面的命令启动auditd：

```
[auditadm@localhost ~]$ auservice start
```

停止 auditd：

```
[auditadm@localhost ~]$ auservice stop
```

查看 auditd 状态：

```
[auditadm@localhost ~]$ auservice status
```

重新从/etc/audit/auditd.conf 加载 auditd 配置：

```
[auditadm@localhost ~]$ auservice reload
```

通过 auditctl -e 0 命令可以临时禁用 auditd，auditctl -e 1 重新启用 auditd。

10.6.6 定义审计规则

审计系统根据一组规则运行，这些规则定义了哪些日志被捕获到日志文件。可以使用auditctl 在命令行中设置审计规则，也可以在/etc/audit/rules.d/目录中设置审计规则。

auditctl 命令让用户能够控制审计系统的基本功能，并定义审计规则决定哪些审计事件被记录。

auditctl 命令格式如下：

```
auditctl [选项]
```

常用选项：

序号	选项	含义
1	-e<启用标志>	设置启用标志。禁用或启用内核审计功能，使用 0 是禁用，1 是启用。
2	-a<列表名称，动作>	向列表末尾添加规则。
3	-A	向列表开头添加规则。
4	-c	处理规则遇到错误时继续执行。
5	-S<系统调用名|系统调用号|all>	生成规则：系统调用的名称或编号。
6	-F<字段名>	建立一个规则字段：名称、操作、值。
7	-w<路径>	在指定路径插入一个记录器。
8	-p<权限>	设置文件权限，r表示读取，w表示写入，x表示执行，a表示更改属性。

序号	选项	含义
9	-k<关键词>	设置审计规则上的过滤关键词，关键词是不超过 32 字节长的任意字符串。
10	-D	删除所有的规则。
11	-s	报告内核的审计子系统状态。
12	-l	列出所有的规则。
13	-R<文件>	从文件中读取规则。
14	-m<文本>	发送用户空间消息到审计系统。
15	-i	从文件中读取规则时忽略错误。
16	-f<故障标志位>	设置故障标志位，有 0、1、2 三个值。 0：审计系统在遇到内部错误时将保持沉默，不会向内核日志输出任何信息。 1：审计系统在遇到内部错误时会将错误信息输出到内核日志（通过 printk 函数）。 2：审计系统在遇到内部错误时会触发内核紧急情况（kernel panic）。
17	-r<发送率>	设置每秒消息发送率。
18	-b<缓冲区最大值>	设置aud：it缓冲区最大值。
19	-d	删除规则。

audit 审计规则有 3 种类型：控制规则（control）、文件监控规则（file）和系统调用规则（syscall）。

1) 控制规则：用于控制 audit 系统的行为，例如设置内核中 audit 缓冲空间的最大值。这些规则通过 auditctl 命令在命令行中临时设置，或者通过配置 /etc/audit/audit.rules 文件来持久化设置。

2) 文件监控规则：文件监控规则用于审计特定文件或目录的访问。如果规则中给定的路径是目录，则使用的规则将递归到目录树的底部，不包括任何可能是挂载点的目录。这些规则的语法通常遵循以下格式：

```
-w path-to-file -p permissions -k keyname
```

其中，权限为以下任一项：

- r 读取文件
- w 写入文件
- x 执行文件
- a 更改文件的属性

3) 系统调用规则：系统调用规则被加载到一个匹配引擎中，该引擎会拦截系统上所有程序发出的每个 syscall。建议仅在必要时使用 syscall 规则，因为这些规则会影响性能。规则越多，性能受到的影响就越大。可以通过尽可能将 syscall 合并到一个规则中来帮助提高性能。

文件系统规则示例：

定义一个规则记录/etc/passwd 文件的所有的写入访问和属性变更

```
[auditadm@localhost ~]$ auditctl -w /etc/passwd -p wa -k passwd_changes
```

定义一个规则记录/etc/pam.d 文件夹的所有的写入访问和属性变更

```
[auditadm@localhost ~]$ auditctl -w /etc/pam.d/ -p wa -k pam_changes
```

定义规则审计目录只会对目录本身的属性进行审计，如果想审计目录下的文件需要单独指定。

系统调用规则示例：

- 定义一个规则记录 sethostname 系统调用被程序使用。

```
[auditadm@localhost ~]$ auditctl -a always,exit -F arch=b64 -S sethostname -k hostname_changes
```

可执行文件规则示例：

- 定义一个规则记录/bin/id 程序的执行，执行下面的命令：

```
[auditadm@localhost ~]$ auditctl -a always,exit -F exe=/bin/id -F arch=b64 -S execve -k
execution_bin_id
```

10.6.7 定义持久化的审计规则

通过命令行定义的审计规则在重启后就会丢失，要想重启后仍然持久的审计规则，必须直接将他们放到/etc/audit/rules.d/audit.rules 文件，或者使用 augenrules 加载位于/etc/audit/rules.d 目录下的规则。

每当 auditd 服务启动时都会生成/etc/audit/audit.rules 文件。/etc/audit/rules.d 目录下的文件使用和 auditctl 命令行相同的语法指定规则。空行和#开头的行将被忽略。

此外，可以使用 auditctl 命令的-R 选项从指定文件读取规则，例如：

```
[auditadm@localhost ~]$auditctl -R /usr/share/audit/sample-rules/71-networking.rules
```

10.6.8 了解审计日志文件

默认审计系统存储日志在/var/log/audit/audit.log 文件，如果启用了日志轮转，轮转的 audit.log 文件也保存在/var/log/audit/目录下。

添加以下审计规则以记录每次读取或修改/etc/ssh/sshd_config 文件的尝试：

```
[auditadm@localhost ~]$ auditctl -w /etc/ssh/sshd_config -p warx -k sshd_config
```

如果 auditd 守护程序正在运行，使用以下命令会在审计日志文件中创建新事件：

```
[auditadm@localhost ~]$ cat /etc/ssh/sshd_config
```

audit.log 文件中的此事件如下所示：

```
type=PROCTITLE msg=audit(1726794739.713:194):
proctitle=636174002F6574632F7373682F737368645F636F6E666967
```

type=PATH msg=audit(1726794739.713:194): item=0 name="/etc/ssh/sshd_config" inode=134801666 dev=fd:00 mode=0100600 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:etc_t:s0:iU nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0

type=CWD msg=audit(1726794739.713:194): cwd="/home/auditadm"

type=SYSCALL msg=audit(1726794739.713:194): arch=c000003e syscall=257 success=yes exit=3 a0=ffffff9c a1=7ffd7ce7dc4e a2=0 a3=0 items=1 ppid=2168 pid=2510 auid=1001 uid=1001 gid=1001 euid=1001 suid=1001 fsuid=1001 egid=1001 sgid=1001 fsgid=1001 tty=pts1 ses=3 comm="cat" exe="/usr/bin/cat" subj=auditadm_u:auditadm_r:auditadm_t:s0-s15:c0.c1023:iU key="sshd_config"

上述事件由四条记录组成，它们具有相同的时间戳和序列号。记录始终以 type=关键字开头。每条记录由多个 name=value 对组成，用空格或逗号分隔。以上事件的详细分析如下：

- 第一条记录

序号	记录	说明
1	type=PROCTITLE	type字段包含记录的类型，这条记录给出触发审计事件的完整命令行。
2	proctitle=636174002F6574632F7373682F737368645F636F6E666967	proctile字段为完整命令行的ASCII码十六进制表示。使用ausearch命令搜索审计记录时使用-i选项自动转换十六进制值为字符串。

- 第二条记录

序号	记录	说明
1	type=PATH	这条记录包含传递给系统调用作为参数的每个路径
2	msg=audit(1726794739.713:194)	时间戳
3	item=0	item表明当前是SYSCALL类型记录中的第 0 项
4	name="/etc/ssh/sshd_config"	作为参数传递给系统调用的文件路径
5	inode=134801666	文件或目录关联的 inode 号
6	dev=fd:00	包含这个文件的设备的主ID和次ID
7	mode=0100600	文件或目录的权限，0100600 可以解释为 -rw-------
8	ouid=0	对象所有者ID
9	ogid=0	对象所有者组ID
10	rdev=00:00	rdev 字段仅包含特殊文件的记录设备标识符
11	obj=system_u:object_r:etc_t:s0:iU	记录的文件的SELinux上下文
12	nametype=NORMAL	每个路径记录的操作意图，常规操作
13	cap_fp=0	基于文件系统的权能设置的相关数据
14	cap_fi=0	继承的基于文件系统权能设置的相关数据
15	cap_fe=0	对象的权能的有效位
16	cap_fver=0	权能版本
17	cap_frootid=0	执行文件操作时所涉及的根文件的标识符

- 第三条记录

序号	记录	说明
1	type=CWD	触发系统调用的进程所在的工作目录
2	cwd="/home/auditadm"	审计管理员的家目录

- 第四条记录

序号	记录	说明
1	type=SYSCALL	记录因为系统调用触发
2	arch=c000003ev	c000003ev是x86_64 的ASCII码 16 进制表示
3	syscall=257	257 是openat。使用ausyscall 将系统调用号转系统调用名
4	success=yes	调用成功
5	exit=3	系统调用返回值 3，这里 3 是新打开文件的fd。
6	a0=ffffff9c a1=7ffd7ce7dc4e a2=0 a3=0	系统调用的前 4 个参数，可以使用ausearch解释
7	ppid=2168	父进程ID
8	pid=2510	进程ID
9	auid=1001	登录用户ID即loginid。1001 是审计管理员的用户ID
10	uid=1001	启动这个被分析的进程的用户ID
11	gid=1001	启动这个被分析的进程的用户组ID
12	euid=1001	启动这个被分析的进程的用户的有效用户ID
13	suid=1001	启动这个被分析的进程的用户的设置用户ID
14	fsuid=1001	启动这个被分析的进程的用户的文件系统用户ID
15	egid=1001	启动这个被分析的进程的用户的有效组ID
16	sgid=1001	启动这个被分析的进程的用户的设置组ID
17	fsgid=1001	启动这个被分析的进程的用户的文件系统组ID
18	tty=pts1	记录进程从哪个终端启动
19	ses=3	会话ID
20	comm="cat"	命令的命令行名字
21	exe="/usr/bin/cat"	可执行文件的路径
22	subj=auditadm_u:auditadm_r:auditadm_t:s0-s15:c0.c1023:iU	进程的安全上下文
23	key="sshd_config"	管理员定义的事件key

10.6.9 搜索审计日志文件

ausearch 允许用户在审计日志中搜索指定的事件。默认 ausearch 会搜索/var/log/audit/audit.log 文件。可以通过-if 选项指定不同的文件。

运行下面的命令，在审核日志中搜索 USER_LOGIN 消息类型，可以找出用户登录的时间：

```
[auditadm@localhost ~]$ ausearch -m USER_LOGIN -ts `date -d '2024/9/20' '+%x %X'` -sv yes
```

这里使用-ts 选项指定日期和时间。-ts 后的日期或时间格式必须和本地日期和时间格式一致，这里使用 date 命令来输出符合本地化日期和时间格式的日期和时间，%x 输出。-sv yes 找出登录成功的日志。

要在/var/log/audit/audit.log 文件中搜索失败的登录尝试，请使用以下命令：

```
[auditadm@localhost ~]$ ausearch --message USER_LOGIN --success no --interpret
```

要搜索所有账户、组和角色更改，请使用以下命令：

```
[auditadm@localhost ~]$ ausearch -m ADD_USER -m DEL_USER -m ADD_GROUP -m
USER_CHAUTHTOK -m DEL_GROUP -m CHGRP_ID -m ROLE_ASSIGN -m ROLE_REMOVE -i
```

要使用用户的登录 ID（auid）搜索特定用户执行的所有记录操作，请使用以下命令：

```
[auditadm@localhost ~]$ ausearch -ua 1002 -i
```

要搜索从昨天到现在的所有失败的系统调用，请使用以下命令：

```
[auditadm@localhost ~]$ ausearch --start yesterday --end now -m SYSCALL -sv no -i
```

搜索与特定 pid 相关的事件使用-p 参数：

```
[auditadm@localhost ~]$ ausearch -p 1
```

该命令将显示根据与 PID 1（即 systemd）相关的规则记录的所有事件。

审核框架的一大功能是它具有 keys 管理事件的能力，建议使用这种用法。可以-k 在规则中使用该选项，以便能够轻松找到相关事件：

```
[auditadm@localhost ~]$ auditctl -w /etc/passwd -p rwxa -k key_pwd
```

如果使用关键搜索事件，ausearch 将仅显示日志记录中存在 key_pwd 的记录：

```
[auditadm@localhost ~]$ ausearch -k key_pwd | less
```

10.6.10 创建审计报告

aureport 实用程序允许审计管理员生成有关审计日志文件中记录的事件的摘要和列示报告。默认情况下，将查询/var/log/audit/目录中的所有 audit.log 文件以创建报告。可以使用 aureport options -if file_name 命令指定其他文件来运行报告。

aureport 命令格式如下：

```
aureport [参数]
```

常用参数：

序号	参数	说明
1	-t	日志时间范围报表
2	-r	关于异常事件响应的报表
3	-x	关于可执行文件的报表

下面是 aureport 使用示例。

要为过去三天（不包括当前示例日）的记录事件生成报告，请使用以下命令：

```
[auditadm@localhost ~]$ aureport --start $(date -d '2024/09/20' '+%x %X' ) --end $(date -d '2024/09/23' '+%x %X')
```

要生成所有可执行文件事件的报告，请使用以下命令：

```
[auditadm@localhost ~]$ aureport -x
```

要生成上述可执行文件事件报告的摘要，请使用以下命令：

```
[auditadm@localhost ~]$ aureport -x --summary
```

要为所有用户生成失败事件的摘要报告，请使用以下命令：

```
[auditadm@localhost ~]$ aureport -u --failed --summary -i
```

要生成每个系统用户所有登录尝试失败的摘要报告，请使用以下命令：

```
[auditadm@localhost ~]$ aureport --login --summary --failed -i
```

要从 ausearch 查询生成搜索用户 1002 的所有文件访问事件的报告，请使用以下命令：

```
[auditadm@localhost ~]$ ausearch --loginuid 1002 --raw | aureport -f --summary
```

要生成查询的所有审计文件及其包含的事件的时间范围的报告，请使用以下命令：

```
[auditadm@localhost ~]$ aureport -t
```