**ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ**
**«НАЦИОНАЛЬНЫЙ ЦЕНТР ПОДДЕРЖКИ И РАЗРАБОТКИ»**

УТВЕРЖДЕН
ЦАУВ.27001-01 34 01-ЛУ

| | |
|---|---|
| Подпись и дата | |
| Инв. № дубл. | |
| Взам. инв № | |
| Подпись и дата | |
| Инв. № подл. | |

# Операционная система «МСВСфера»

Руководство администратора

**ЦАУВ.27001-01 34 01**

(Листов - 389)

2025

# Содержание

## АННОТАЦИЯ

Настоящее руководство предназначено для администраторов операционной системы МСВСфера 9 Сертифицированная (ФСТЭК). Руководство ориентировано на специалистов, знакомых с операционными системами типа Linux и имеющих минимальный практический опыт работы с ними. Руководство снабжено примерами, сделанными в операционной системе МСВСфера 9 Сертифицированная (ФСТЭК), установленной в конфигурации «Сервер с графическим интерфейсом».

# 1. ОБЩИЕ СВЕДЕНИЯ

## 1.1. Назначение и область применения

Операционная система МСВСфера 9 Сертифицированная (ФСТЭК) — это операционная система на основе ядра Linux, выпускается в следующих редакциях:

- МСВСфера 9 Сервер;
- МСВСфера 9 АРМ.

**МСВСфера 9 Сервер** — серверная операционная система с набором интегрированных служб и приложений, включающим веб-сервер, почтовый сервер, сервер служб сетевой инфраструктуры, серверы файлов и печати, средства резервного копирования и восстановления данных, множество других служб и приложений, а также средства администрирования и защиты информации. Развёрнутую ОС МСВСфера 9 Сервер применяют в качестве программной платформы для использования в корпоративных сетях и серверных окружениях.

**МСВСфера 9 АРМ** (АРМ — автоматизированное рабочее место) — клиентская операционная система с набором интегрированных пользовательских приложений, включающим пакет офисных программ, браузер, почтовую программу, редакторы текстов и графики, проигрыватели аудио и видео, менеджеры файлов и архивов, программу сканирования документов, множество других программ, а также средства администрирования и защиты информации. МСВСфера 9 АРМ представляет собой комплекс решений, предназначенных для организации и оптимизации работы, обладает высокой степенью гибкости и адаптивности.

- ОС МСВСфера 9 включена в Реестр отечественного ПО, запись №16243 от 30.12.2022.

## 1.2. Обеспечение безопасности и требования к администратору

Внедрению и использованию операционной системы должны предшествовать подготовительные процедуры, направленные на обеспечение безопасности при приемке установочного дистрибутива операционной системы от поставщика, на обеспечение безопасной установки, настройки и запуска операционной системы и на создание безопасной среды её функционирования. Реализация подготовительных процедур должна обеспечиваться необходимыми ресурсами и сопровождаться назначением ответственных за их выполнение должностных лиц.

Процедуры безопасной приемки должны предусматривать меры подтверждения подлинности установочного дистрибутива операционной системы, исключающие возможности преднамеренного или непреднамеренного внесения изменений в поставляемую версию, т.е. замены её фальсифицированной или неработоспособной версией.

К таким мерам в общем случае относятся:

- проверка подлинности источника поставки путем визуального контроля наличия и целостности специальных защитных стикеров (наклеек, знаков) на упаковке комплекта поставки, а также целостности самой упаковки;
- проверка комплектности поставки в соответствии с заявкой, договорными материалами и спецификацией, сверка маркировки и номера версии;
- проверка целостности установочного дистрибутива с помощью программного средства контроля целостности путем сравнения с эталонным значением контрольной суммы или с помощью средств электронной подписи.

Процедуры безопасной установки, настройки, запуска операционной системы и создания безопасной среды её функционирования в общем случае должны предусматривать меры, обеспечивающие:

- совместимость операционной системы со средствами вычислительной техники, на которых планируется её установка и использование;
- установку, конфигурирование, настройку, запуск и управление операционной системой в соответствии с эксплуатационной документацией и принятой политикой безопасности;
- защиту от действий, направленных на нарушение физической целостности средств вычислительной техники, на которых она функционирует;
- доверенную загрузку операционной системы, контроль доступа к процессу загрузки, блокирование попыток несанкционированной загрузки, контроль целостности компонентов загружаемой операционной среды;
- наличие ресурсов для выполнения функциональных возможностей безопасности операционной системы, хранения создаваемых резервных копий, а также защищенное хранение данных операционной системы и защищаемой информации;

- ограничение на установку программного обеспечения и его компонентов, не задействованных в технологическом процессе обработки информации;
- доверенный маршрут между операционной системой и пользователями;
- доверенный канал передачи данных между операционной системой и средствами вычислительной техники, на которых происходит обработка информации, а также с которых происходит их администрирование;
- невозможность отключения или обхода компонентов операционной системы и средств защиты информации;
- препятствие несанкционированному копированию информации, содержащейся в операционной системе, на съемные носители информации, в том числе контроль вноса (выноса) в (из) контролируемую зону съемных носителей информации;
- проверку целостности получаемых от поставщика внешних модулей уровня ядра перед их установкой в операционную систему;
- выделение вычислительных ресурсов для процессов в соответствии с их приоритетами;
- профессиональную компетентность и надежность персонала, ответственного за администрирование системы, его способность выполнять свои обязанности в точном соответствии с принятой политикой безопасности и эксплуатационной документацией;
- возможность генерации аутентификационной информации, соответствующей заданной метрике качества;
- недоступность аутентификационной информации для лиц, не уполномоченных на её использование;
- разделение полномочий пользователей и администраторов с назначением им минимально необходимых прав и привилегий;
- исключение в процессе использования системы доступа пользователей к приложениям, выполняющимся с более высокими правами доступа, чем права, предоставленные им согласно матрице доступа;
- завершение администраторами приложений, запущенных ими с административными правами, после окончания работы с ними;
- запрет пользователям на передачу посторонним лицам своей личной идентификационной и аутентификационной информации, а также на регистрацию кого-либо в системе под своим именем и паролем.

Для управления ОС МСВСфера 9 используются командные интерпретаторы (shell). Поэтому администратор должен иметь:

- базовые навыки администрирования ОС семейства Linux;
- навыки конфигурирования и настройки программных продуктов и ОС;
- опыт работы со стандартными элементами графического интерфейса приложений;
- навыки поддержания в работоспособном состоянии технических средств ПК.

# 2. УСТАНОВКА И НАЧАЛЬНАЯ НАСТРОЙКА СИСТЕМЫ

## 2.1. Системные требования

### 2.1.1. Минимальные

Для использования операционной системы требуется компьютер со следующими минимальными характеристиками:

- Процессор:
    - Intel или AMD версии не ниже x86-64-v2 (Intel Nehalem и более поздние, AMD Bulldozer и более поздние).
- 2 Гбайта оперативной памяти.
- 20 Гбайт свободного пространства памяти на жёстком диске в зависимости от используемой конфигурации.

### 2.1.2 Рекомендуемые

Для полнофункционального использования операционной системы рекомендуется использовать компьютер со следующими характеристиками:

- Процессор:
    - Intel или AMD версии не ниже x86-64-v2 (Intel Nehalem и более поздние, AMD Bulldozer и более поздние).
- 8 Гбайт оперативной памяти.
- 40 Гбайт свободного пространства памяти на жёстком диске в зависимости от используемой конфигурации.

Установка ОС МСВСфера 9 может осуществляться различными способами: с оптического диска, с жесткого диска, по сети. В данном документе описывается стандартная установка ОС МСВСфера 9 с загрузочного USB-носителя. См. «*2.3. Установка системы с USB-носителя*».

## 2.2. Создание загрузочного USB-носителя и запись iso-образа дистрибутива

В настоящее время наиболее удобным способом установки операционной системы является использование USB-носителя с записанным на него дистрибутивом. Ниже мы рассмотрим, как создать загрузочный USB-носитель и записать на него iso-образ дистрибутива.

Программное обеспечение, рекомендуемое для создания загрузочного USB-

носителя и записи iso-образа дистрибутива операционной системы:

- Fedora Media Writer — для операционных систем семейства Windows, Linux и macOS;
- balenaEtcher — для операционных систем семейства Windows, Linux и macOS;
- Win32 Disk Imager — для операционных систем семейства Windows;
- Утилита командной строки dd — для операционных систем семейства Linux.

Интерфейс указанного программного обеспечения интуитивно понятный, дополнительные инструкции вы можете найти в документации соответствующего ПО.

В качестве примера рассмотрим процесс создания загрузочного USB-носителя и записи iso-образа дистрибутива операционной системы в программе Fedora Media Writer в операционной системе семейства Windows и с использованием утилиты командной строки dd в операционной системе семейства Linux.

### 2.2.1. Пример создания загрузочного USB-носителя и записи iso-образа дистрибутива операционной системы

1. Скачайте последнюю версию Fedora Media Writer для Windows на ваше устройство.
2. Запустите установочный файл и выполните установку Fedora Media Writer на ваше устройство.
3. Вставьте USB-носитель, на который вы планируете записывать iso-образ дистрибутива. Убедитесь, что на нём достаточно места.
4. Скачайте актуальный iso-образ выбранной вами операционной системы.
5. Запустите Fedora Media Writer.
6. Выберите источник образа — «Выбрать файл iso» и нажмите «Дальше».

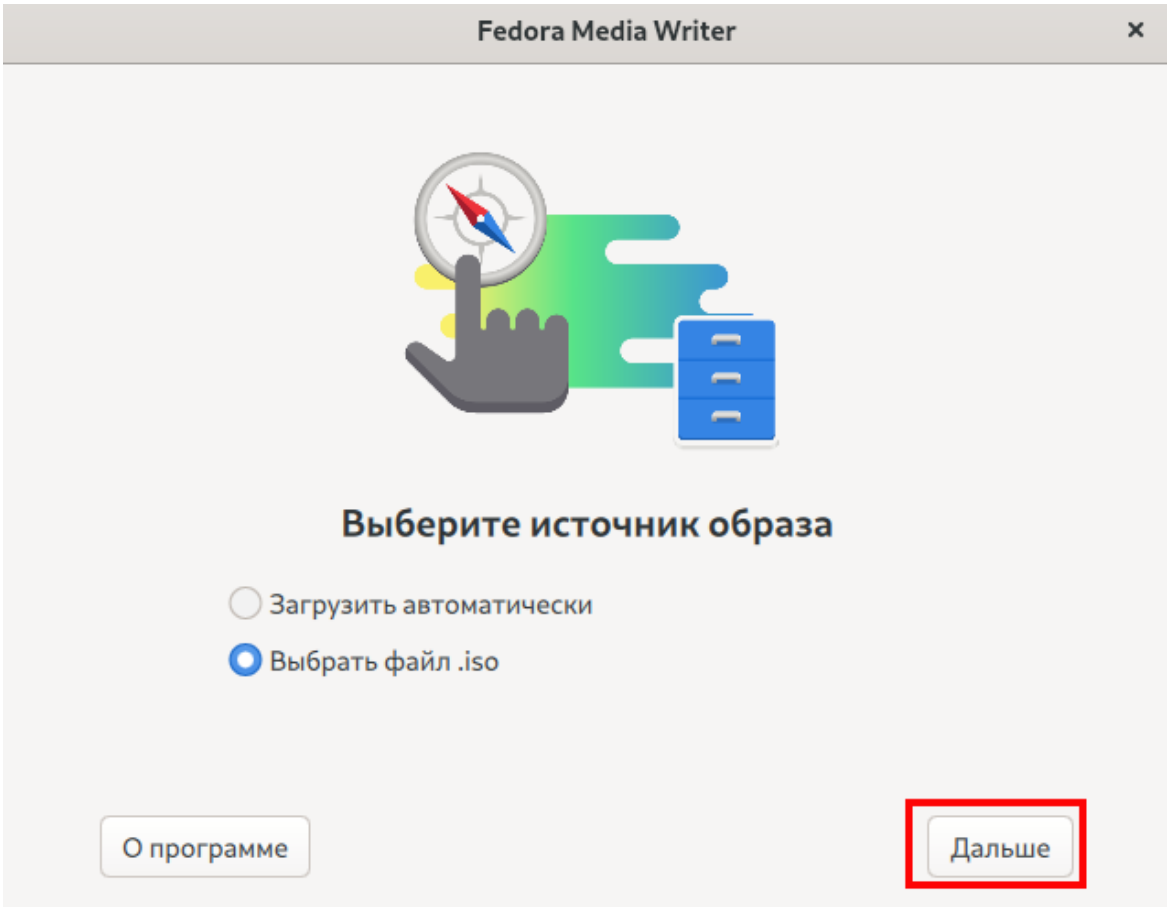



Рис. 1: Выбрать файл iso

7. В окне «Выбрать диск» → «Параметры записи» → «Выбранный файл» нажмите на кнопку «Выбрать» для выбора загруженного ранее iso-образа операционной системы.
8. USB-накопители определяются автоматически. Если у вас подключено несколько USB-носителей, выберите необходимый из списка.
9. После выбора iso-образа нажмите «Запись».
10. При необходимости укажите пароль администратора для подтверждения записи.
11. Начнётся запись iso-образа на USB-носитель. Это может занять некоторое время.
12. После завершения записи нажмите «Готово».
13. Вы успешно создали загрузочный USB-носитель операционной системы!

### 2.2.2. Пример создания загрузочного USB-носителя и записи iso-образа дистрибутива операционной системы с помощью утилиты командной строки dd (Linux)

1. Вставьте USB-носитель, на который вы планируете записывать iso-образ дистрибутива. Убедитесь, что на нём достаточно места.
2. Скачайте актуальный iso-образ выбранной вами операционной системы.
3. Откройте «Терминал».
4. Введите команду для записи iso-образа:

```
$ sudo dd oflag=dsync if=имя_файла_загруженного_образа_oc.
↪iso of=/dev/sdc bs=1M status=progress;sync
```

Измените `имя_файла_загруженного_образа_oc.iso` на имя файла iso-образа дистрибутива опреационной системы, которую вы устанавливаете.

Или

```
$ sudo pv имя_файла_загруженного_образа_oc.iso | dd
↪oflag=dsync of=/dev/sdc bs=1M;sync
```

где `/dev/sdc` — это USB-носитель.

Измените `имя_файла_загруженного_образа_oc.iso` на имя файла iso-образа дистрибутива опреационной системы, которую вы устанавливаете.

## 2.3. Установка системы с USB-носителя

Для установки ОС МСВСфера 9 с USB-носителя необходимо перед началом установки выбрать приоритетную загрузку с USB-носителя в BIOS устройства, либо выбрать загрузку с USB-носителя однократно в процессе инициализации компьютера.

Для установки и загрузки ОС МСВСфера 9 может потребоваться отключить параметр Secure Boot в BIOS устройства, на которое производится установка.

Для начала установки подключите USB-носитель с установочным дистрибутивом к компьютеру.

Рассмотрим пример установки операционной системы МСВСфера 9.5 Сертифицированная (ФСТЭК).

Сначала установка будет проходить в текстовом режиме.

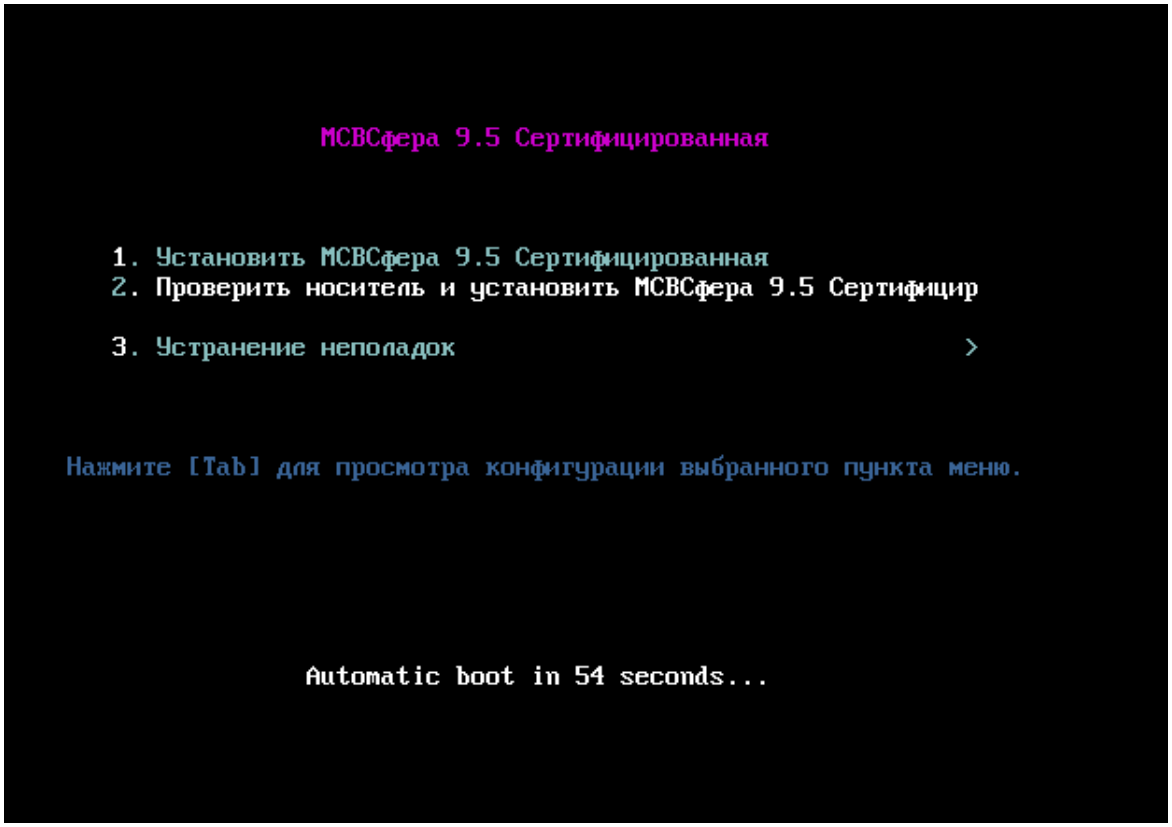



Рис. 2: Текстовый режим установки

Доступны следующие варианты:

- Установить МСВСфера 9 Сертифицированная — начнётся установка МСВСфера 9 на ваше устройство.
- Проверить носитель и установить МСВСфера 9 Сертифицированная — программа установки проверит контрольные суммы образа диска, подтверждая что скачивание образа и запись на загрузочный носитель прошли без ошибок.
- Устранение неполадок — вы сможете перейти в режим восстановления, который представляет собой минимальную среду ОС МСВСфера 9, загружаемую с загрузочного носителя. В этом режиме используются утилиты командной строки, с помощью которых вы можете монтировать или не монтировать файловые системы, заносить в чёрный список и добавлять драйверы, устанавливать и обновлять системные пакеты, а также управлять разделами.

При нажатии на «Установить МСВСфера 9 Сертифицированная» установка продолжится в графическом режиме и на экране монитора компьютера появится

окно с предложением выбрать язык установки.

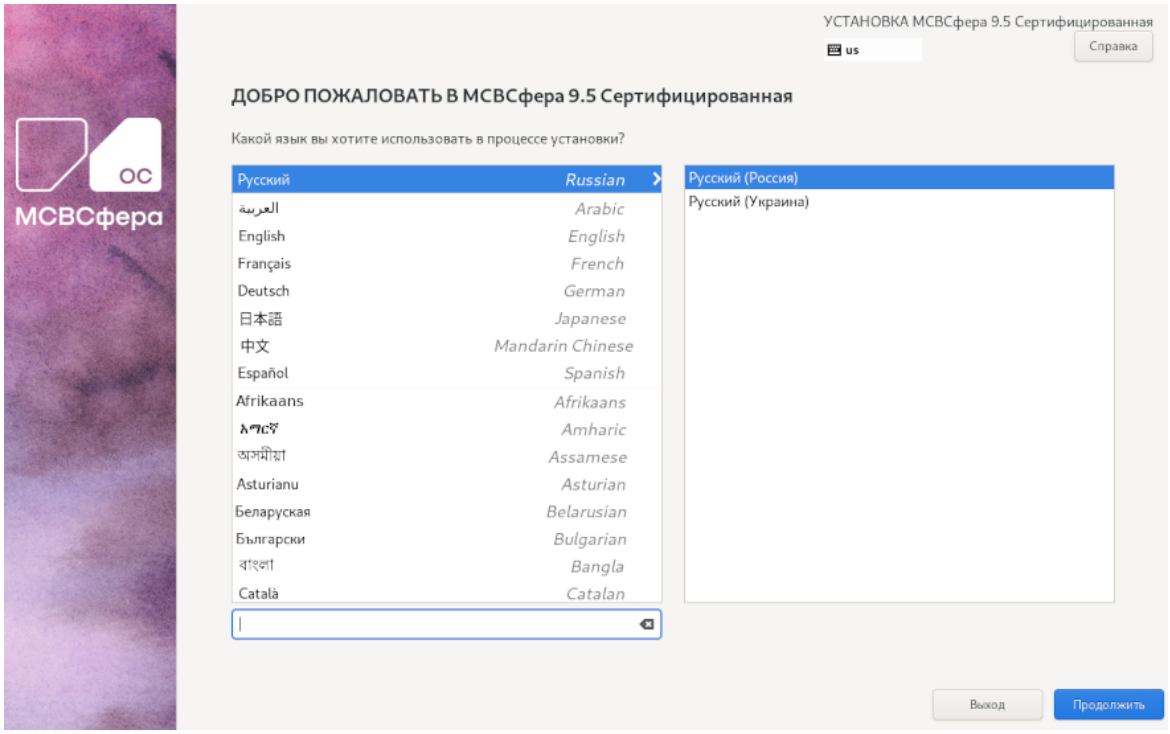



Рис. 3: Выбрать язык установки

Затем появится окно «Обзор установки», с помощью которого, последовательно нажимая кнопку «Готово», можно будет произвести все необходимые настройки. Для начала установки нажмите «Начать установку».

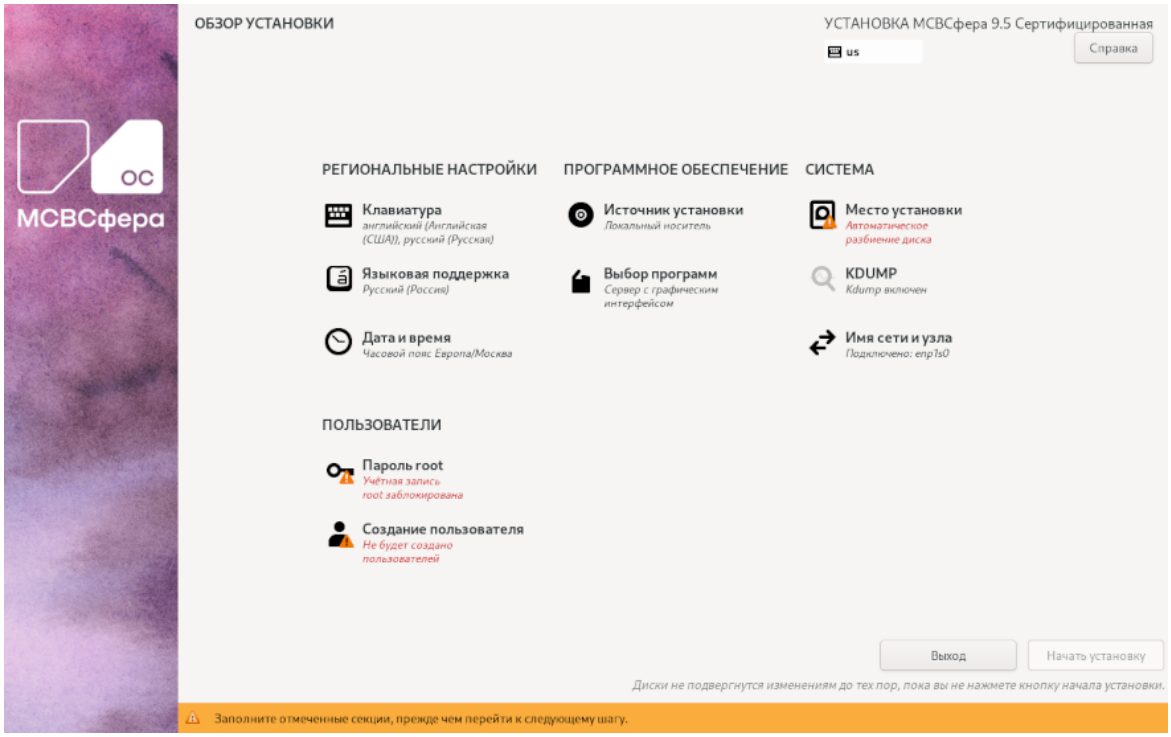



Рис. 4: Обзор установки

Раскладка клавиатуры.

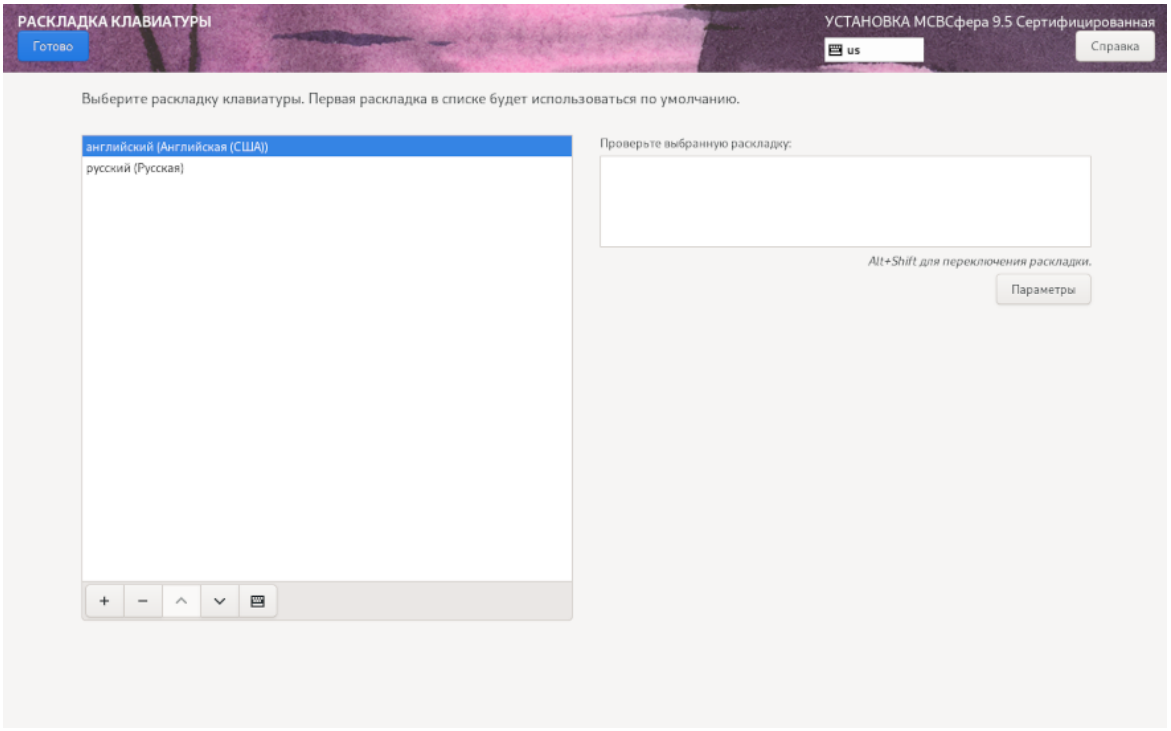



Рис. 5: Раскладка клавиатуры

Языковая поддержка.

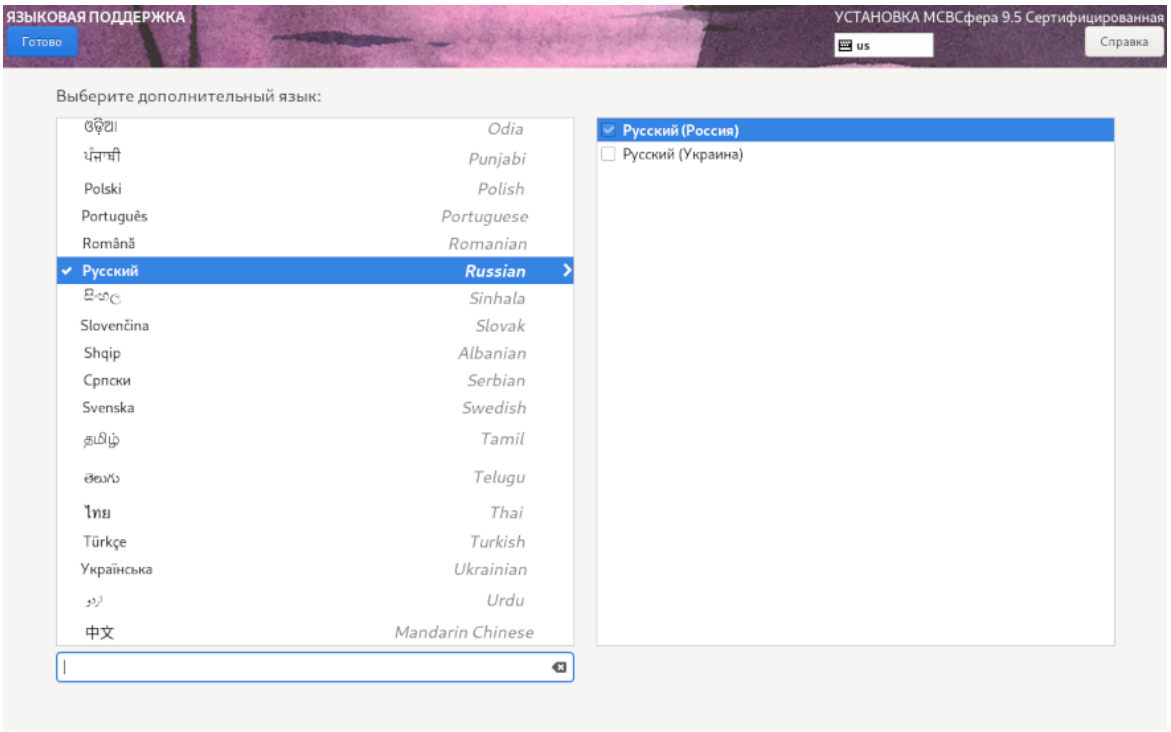



Рис. 6: Языковая поддержка

Дата и время.

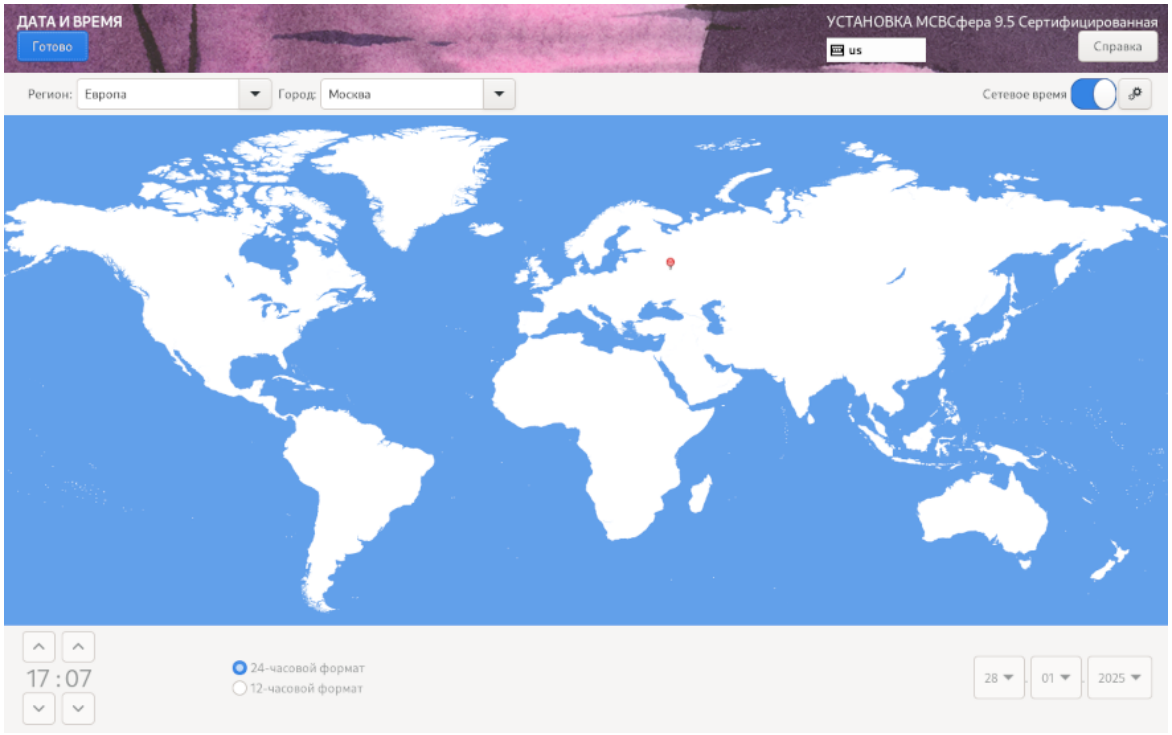



Рис. 7: Дата и время

Источник установки.

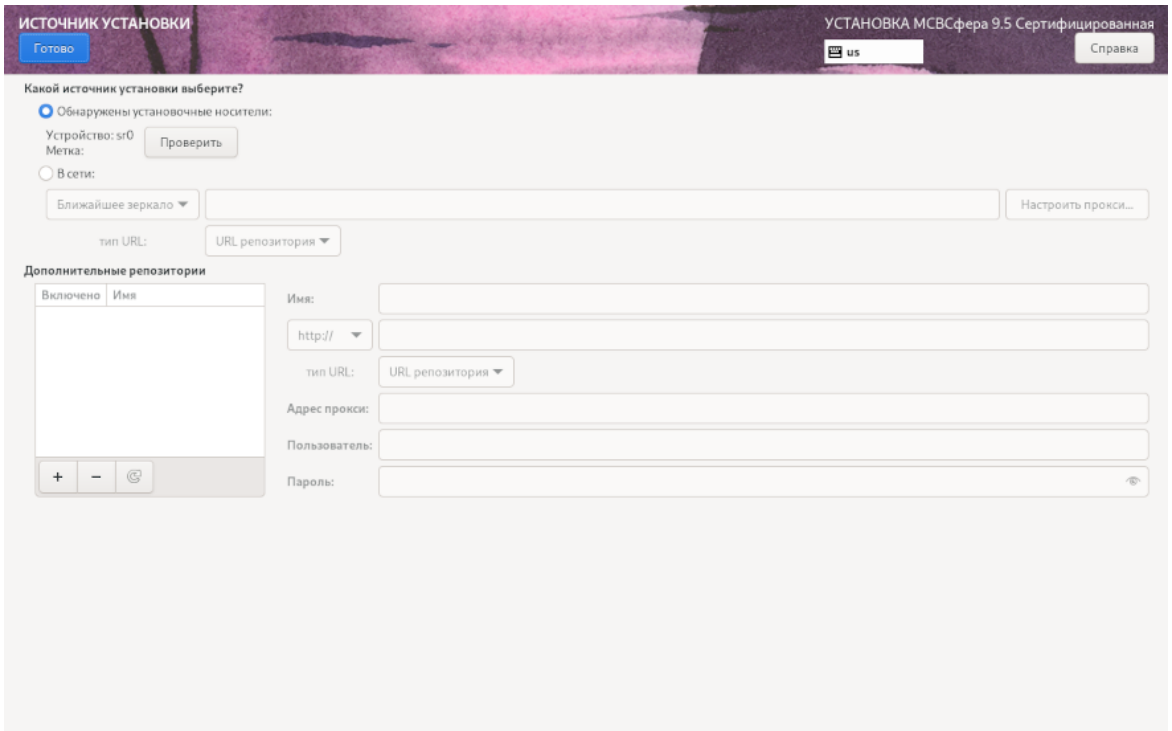



Рис. 8: Источник установки

Выбор программ.

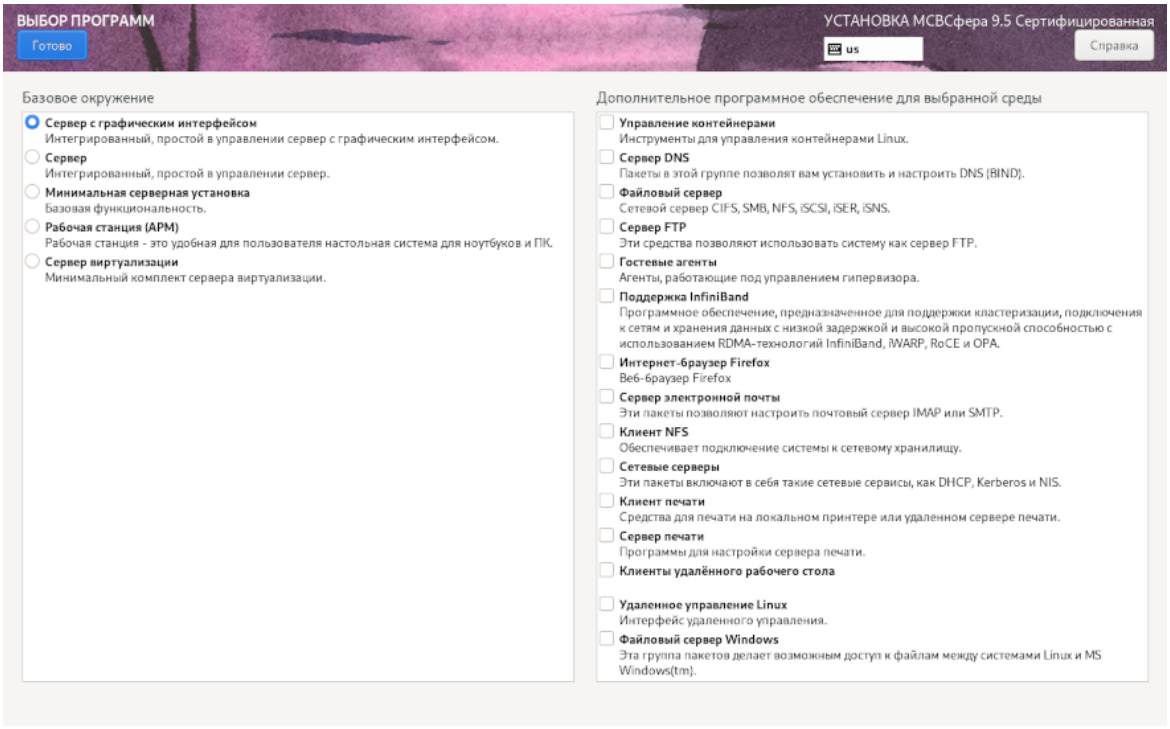



Рис. 9: Выбор программ

Доступны следующие варианты базового окружения:

- **Сервер с графическим интерфейсом** — включает все приложения серверной операционной системы **МСВСфера Сервер** с графическим интерфейсом.
- **Сервер** — включает минимальный набор серверных приложений и текстовый интерфейс.
- **Минимальная серверная установка** — включает минимальный набор пакетов, необходимых для работы операционной системы, без приложений и без графического интерфейса.
- **Рабочая станция (АРМ)** — включает все приложения клиентской операционной системы **МСВСфера АРМ** с графическим интерфейсом.
- **Сервер виртуализации** — включает все приложения, содержащиеся в окружении **Сервер**, а также пакеты для обеспечения виртуализации QEMU/KVM (без графического интерфейса).

Вы также можете добавить дополнительное программное обеспечение для выбранного базового окружения, отметив его в окне справа. Пакеты будут загружены и установлены автоматически.

Место установки.

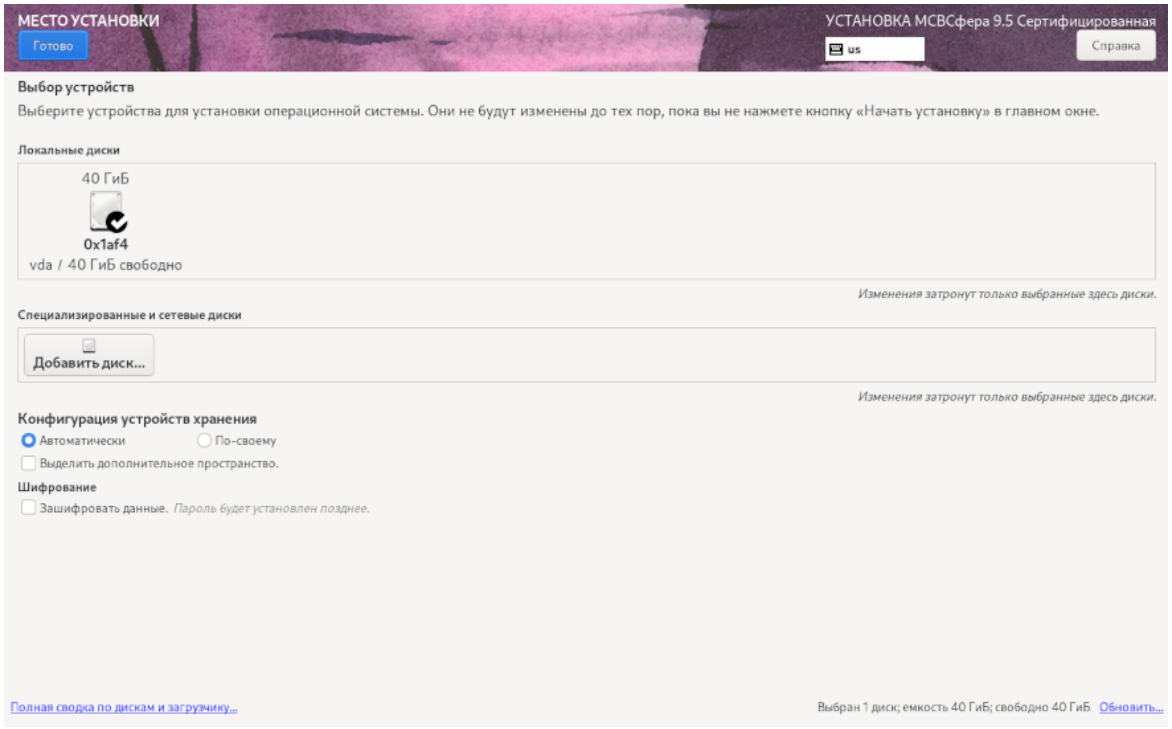



Рис. 10: Место установки

Диагностика сбоев ядра.

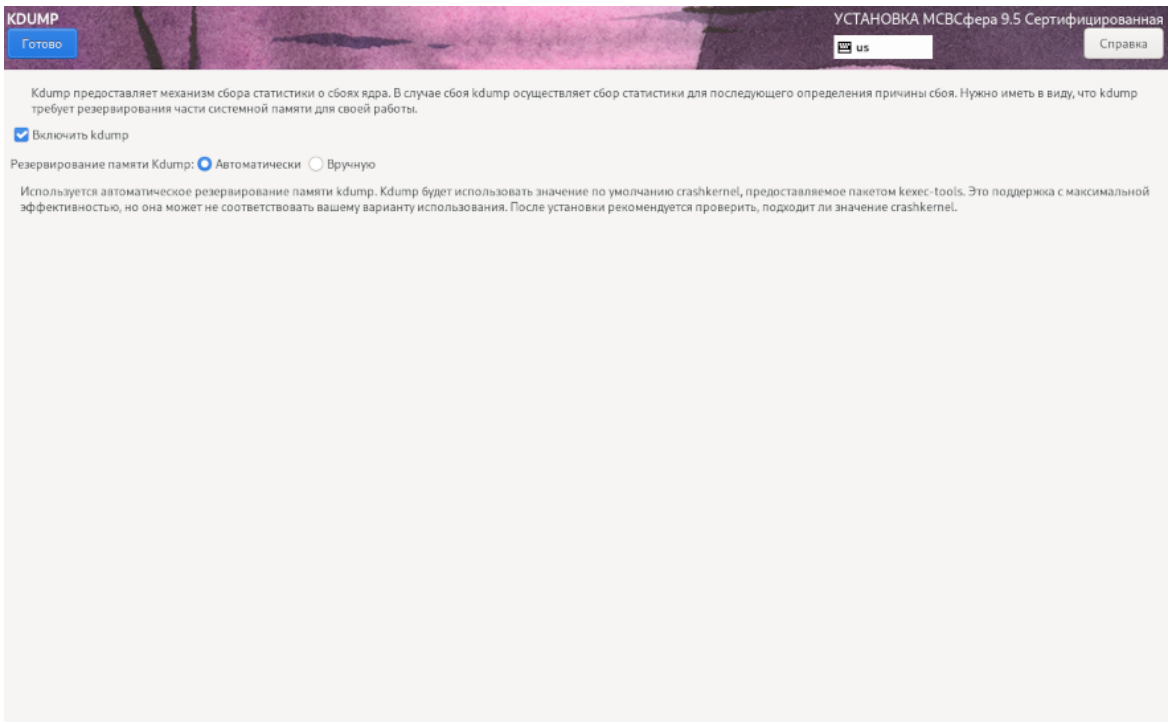



Рис. 11: Диагностика сбоев ядра

Имя сети и узла.

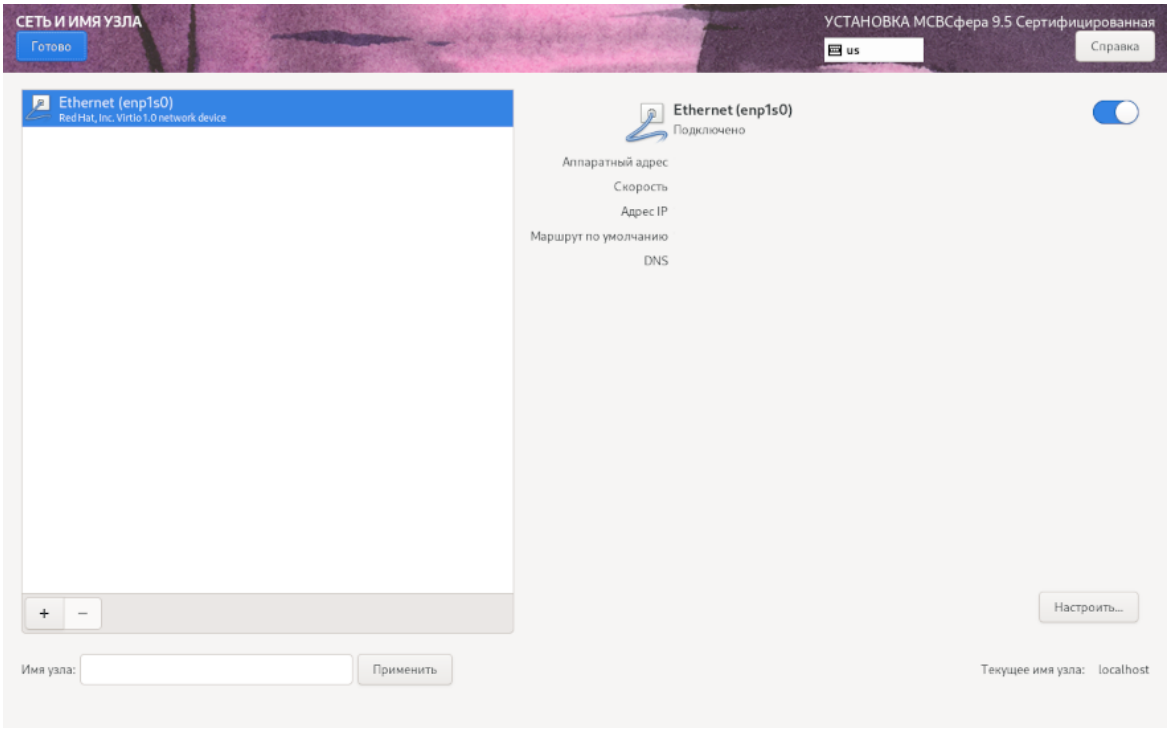



Рис. 12: Имя сети и узла

Задать пароль суперпользователя root.

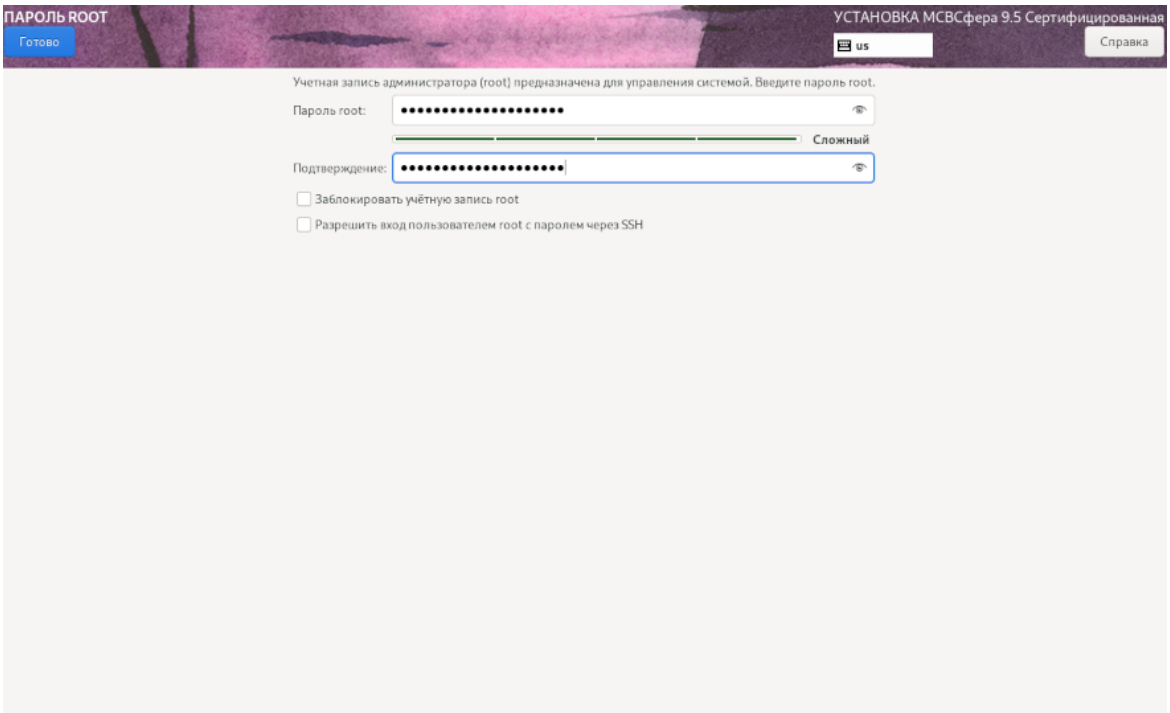



Рис. 13: Пароль суперпользователя root

Вы можете создать одну учётную запись для администрирования. Учётные записи других пользователей можно создать после установки системы.

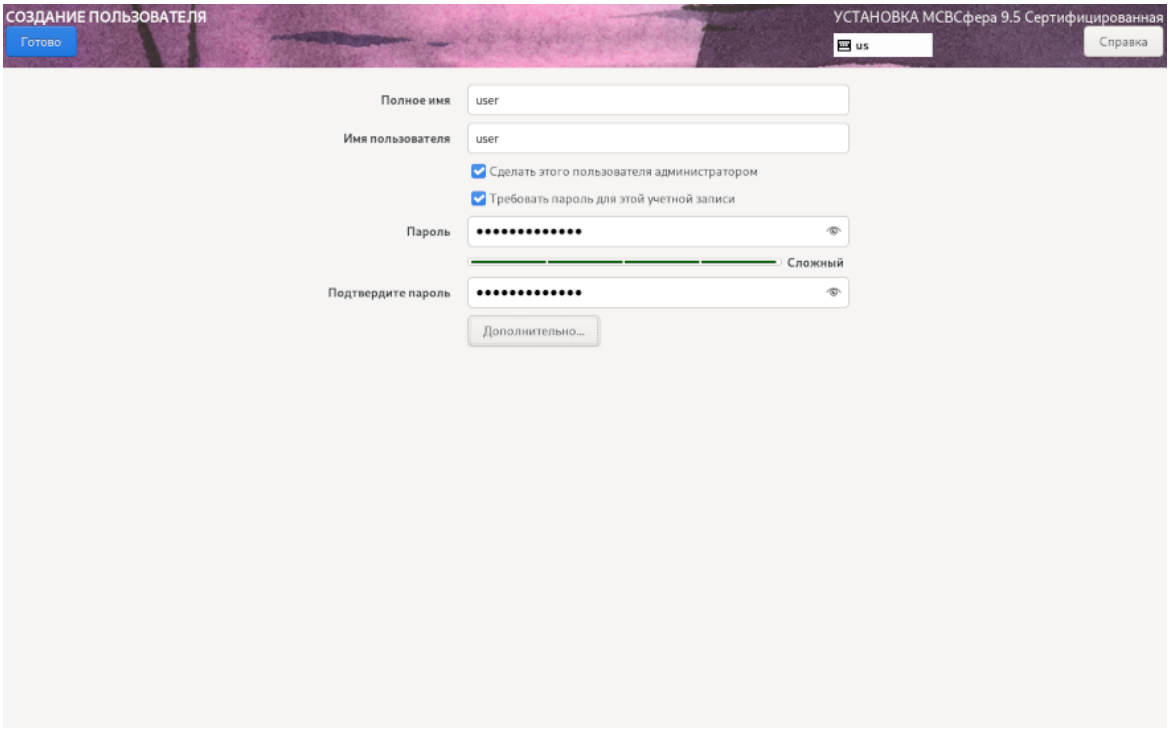



Рис. 14: Новый пользователь

После того, как все необходимые настройки произведены, нажмите «Начать установку».

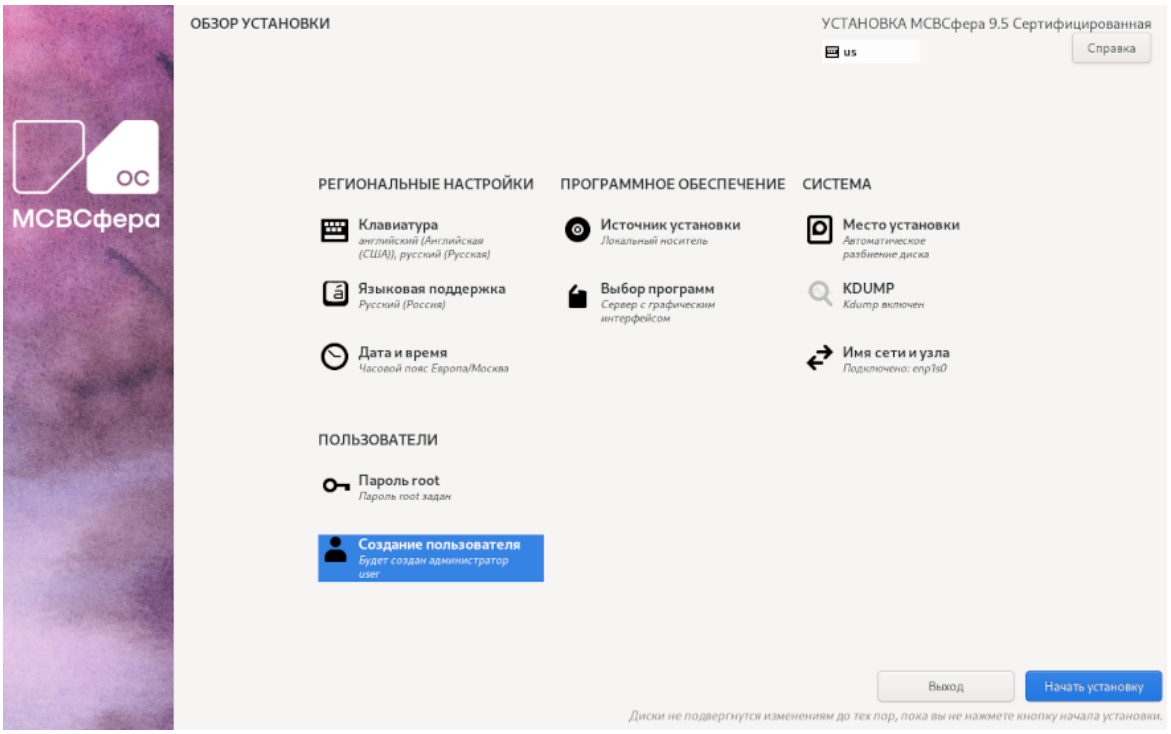



Рис. 15: Начать установку

Продолжительность установки может составить 20-30 минут, в зависимости от быстродействия оборудования и конфигурации программного обеспечения. По завершении установки на экране монитора появится предложение произвести перезагрузку.


Рис. 16: Готово!

После извлечения USB-носителя с установочным дистрибутивом и перезагрузки системы появится окно первой настройки с предложением прочитать и принять лицензионное соглашение. После того как лицензионное соглашение будет принято, нажмите на кнопку «Завершить». После этого на экране появится приглашение войти в систему, пройдя идентификацию и аутентификацию.

# 3. УПРАВЛЕНИЕ ПАКЕТАМИ

## 3.1. Введение и основные понятия

ОС МСВСфера 9 представляет собой комплексную систему, которая обеспечивает стабильную и безопасную работу для пользователей.

Так как ОС МСВСфера 9 собрана на базе ядра Linux, то в ней несколько приложений могут использовать одни и те же библиотеки или, например, одно приложение может использовать другое. С одной стороны это даёт возможность освободить место, занимаемое приложением, и снизить потребление ресурсов, а с другой стороны возникает необходимость обеспечения целостности системы.

Информация о всех необходимых приложению бинарных и конфигурационных файлах, о том, как их следует разместить в файловой системе, а также данные о зависимостях хранится в архиве специального формата, называемом **пакетом**.

В ОС МСВСфера 9 форматом пакета является RPM (рекурсивный акроним RPM Package Manager, ранее Red Hat Package Manager), а сами файлы, содержащие пакеты, имеют расширение `.rpm`.

Как было сказано выше, приложения могут совместно использовать одни и те же библиотеки или даже целые программы, и здесь возникает понятие **зависимости**: в приложении может не хватать чего-то для работы, и ему для этого нужно другое приложение или библиотека. То есть один пакет начинает зависеть от другого. И удалив, например, одну библиотеку можно нарушить работу сразу нескольких приложений.

Для работы с пакетами и обеспечения целостности системы используются программы, называемые **пакетными менеджерами**. Они управляют пакетами: устанавливают, удаляют, обновляют, ведут учёт, выводят информацию, отслеживают версии и зависимости и пр.

В ОС МСВСфера 9 пакетным менеджером является **DNF**.

Так как пакеты зависят друг от друга, то зачастую недостаточно установить только один пакет — нужно устанавливать сразу несколько, поэтому разработчики создают и поддерживают специальные централизованные серверы, называемые **репозиториями**, где хранятся различные пакеты. Пакетный менеджер видит зависимости каждого пакета, сам находит подходящие пакеты в репозитории и предлагает их установить.

**Дистрибутив** МСВСфера 9 Сертифицированная (ФСТЭК) имеет набор

собственных репозиториев для всех поддерживаемых выпусков и архитектур, в которых содержатся приложения и программы.

Обычно некоторые пакеты, которые часто используют вместе, объединены в **группы**. Посмотреть список доступных групп поможет пакетный менеджер DNF.

### 3.2. Пакетный менеджер DNF

Рассмотрим основные операции с пакетами, которые может выполнить пакетный менеджер DNF.

#### 3.2.1. Найти нужный пакет

Для поиска пакета (даже не зная его точного имени) выполните следующую команду:

```
$ dnf search имя_пакета
```

В имени пакета вы можете использовать шаблоны, а также указывать только те буквы из названия, которые помните.

Пример: найдём пакет по первым буквам:

```
$ dnf search *fox
```

Результат работы команды:

```
$ dnf search *fox
==== Имя совпадение: *fox =================
firefox.x86_64 : Mozilla Firefox Web browser
```

#### 3.2.2. Установить нужный пакет

Для установки пакета выполните следующую команду:

```
$ sudo dnf install имя_пакета
```

DNF проверит все зависимости и при обнаружении нужных, но ещё не установленных пакетов, установит их, пользуясь всеми доступными репозиториями.

Пример: установим пакет `firefox.x86_64`:

```
$ sudo dnf install firefox.x86_64
Зависимости разрешены.
================================================================================
Пакет          Архитектура Версия                    Репозиторий
 ↪Размер
================================================================================
Установка:
 firefox       x86_64     128.6.0-1.el9_5.inferit  os          123
 ↪M

Результат транзакции
================================================================================
Установка  1 Пакета

Объем загрузки: 123 M
Объем изменений: 313 M
Продолжить? [д/Н]: д
Загрузка пакетов:
firefox-128.6.0-1.el9_5.inferit.x86_64.rpm  8.1 MB/s | 123 MB
 ↪00:15
--------------------------------------------------------------------------------
 ↪--
Общий размер                                7.9 MB/s | 123 MB
 ↪00:14
...
...
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
 Подготовка       :                                                 1
 Установка        : firefox-128.6.0-1.el9_5.inferit.x86_64          1
 Запуск скриптлета: firefox-128.6.0-1.el9_5.inferit.x86_64          1
 Проверка         : firefox-128.6.0-1.el9_5.inferit.x86_64          1

Установлен:
 firefox-128.6.0-1.el9_5.inferit.x86_64
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Выполнено!
```

### 3.2.3. Обновить установленные пакеты

Для проверки наличия обновлений выполните следующую команду:

```
$ dnf check-upgrade

MSVSphere 9 - OS          1.0 MB/s | 4.2 MB     00:04
MSVSphere 9 - Updates     4.1 MB/s | 1.4 MB     00:00
```

Для обновления всей системы выполните следующую команду:

```
$ sudo dnf upgrade
```

Для обновления определённого пакета (и его зависимостей) выполните следующую команду:

```
$ sudo dnf upgrade имя_пакета
```

### 3.2.4. Удалить установленный пакет

Для удаления пакета выполните следующую команду:

```
$ dnf remove имя_пакета
```

Пример: удалим пакет `firefox.x86_64`:

```
$ sudo dnf remove firefox.x86_64
Зависимости разрешены.
================================================================
Пакет    Архитектура Версия                          Репозиторий
 ↪Размер
================================================================
Удаление:
firefox   x86_64     128.6.0-1.el9_5.inferit.x86_64 @os         313
 ↪M

Результат транзакции
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
================================================================================
Удаление  1 Пакет

Освобожденное место: 313 M
Продолжить? [д/Н]: д
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции
 Подготовка       :                                                        1/1
 Запуск скриптлета: firefox-128.6.0-1.el9_5.inferit.x86_64                 1/1
 Удаление         : firefox-128.6.0-1.el9_5.inferit.x86_64                 1/1
 Запуск скриптлета: firefox-128.6.0-1.el9_5.inferit.x86_64                 1/1
 Проверка         : firefox-128.6.0-1.el9_5.inferit.x86_64                 1/1

Удален:
 firefox-102.9.0-3.el9_1.inferit.3.x86_64
```

Также будут удалены все пакеты, которые зависят от удаляемого (при их наличии).

### 3.2.5. Проверить целостность пакета

Для проверки целостности rpm-пакета выполните следующую команду:

```
$ rpm -V имя_rpm_пакета
```

В результате работы команды будет указана следующая информация:

- размер пакета
- полномочия
- тип
- владелец
- группа
- MD5-сумма
- дата последнего изменения пакета

### 3.2.6. Получить информацию об установленном пакете

Для получения подробной информации об установленном пакете выполните следующую команду:

```
$ dnf info имя_пакета
```

Пример работы команды для пакета `firefox.x86_64`:

```
$ dnf info firefox.x86_64
Установленные пакеты
Имя          : firefox
Версия       : 128.6.0
Выпуск       : 1.el9_5.inferit
Архитектура  : x86_64
Размер       : 313 M
Источник     : firefox-128.6.0-1.el9_5.inferit.src.rpm
Репозиторий  : @System
Из репозитор : os
Краткое опис : Mozilla Firefox Web browser
URL          : https://www.mozilla.org/firefox/
Лицензия     : MPLv1.1 or GPLv2+ or LGPLv2+
Описание     : Mozilla Firefox is an open-source web browser,
 ↪designed for standards
            : compliance, performance and portability.
```

## 3.3. Описание репозиториев ОС МСВСфера 9 Сертифицированная (ФСТЭК)

Рассмотрим репозитории МСВСфера 9.

- **MSVSphere 9 - OS** — содержит базовый набор пакетов, которые прошли сертификацию и изначально поставляются на ISO-образе. По умолчанию включён.
- **MSVSphere 9 - Updates** — содержит пакеты, выпущенные для исправления уязвимостей или ошибок. По умолчанию выключен. Необходимо включить его для получения обновлений.
- **MSVSphere 9 - OS Local** — содержит базовый набор пакетов, которые прошли сертификацию и изначально поставляются на ISO-образе для изолированной среды (без подключения к Интернету). По умолчанию

выключен.

- **MSVSphere 9 - Updates Local** — содержит пакеты, выпущенные для исправления уязвимостей или ошибок для изолированной среды (без подключения к Интернету). По умолчанию выключен.
- **MSVSphere 9 - Testing** — содержит пакеты с обновлениями, которые ещё не попали в репозиторий «Updates».

### 3.3.1. Посмотреть список включённых и доступных репозиториев

Для просмотра списка включённых репозиториев выполните следующую команду:

```
$ dnf repolist
```

Для просмотра списка включённых и отключённых репозиториев выполните следующую команду:

```
$ dnf repolist all
```

Для вывода подробного описания для каждого включённого репозитория выполните следующую команду:

```
$ dnf repolist -v
```

Для вывода списка отключённых репозиториев выполните следующую команду:

```
$ dnf repolist disabled
```

Для получения подробной информации о конкретном репозитории выполните следующую команду:

```
$ dnf repolist название репозитория -v
```

Пример: вывести подробную информацию о репозитории OS:

```
$ dnf repolist os -v
...
...
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
ИД репозитория              : os
Имя репозитория             : MSVSphere 9 - OS
Статус репозитория          : включено
Версия репозитория          : 9.5
Метки дистрибутива          : [cpe:/o:ncsd:msvsphere:9]:  , 9, M, S,
 ↪S, V, e, e, h, p, r
Репозиторий обновлен        : Чт 23 янв 2023 18:50:55
Пакеты репозитория          : 3 357
Пакеты-в-репозитории        : 3 357
Размер-репозитория          : 3.8 G
Зеркала-репозитория         : https://mirrors.inferitos.ru/
 ↪mirrorlist/9/os
Базовый-URL-репозитория     : https://repo1.msvsphere-os.ru/
 ↪certified/msvsphere/9/isos/x86_64/ (0 more)
Истечение срока репозитория: 86 400 секунд(а) (осталось: Ch 29 нв
 ↪2025 15:59:28)
Имя файла репозитория       : /etc/yum.repos.d/msvsphere-baseos.repo
Всего пакетов               : 3 357
```

Здесь мы видим, что репозиторий включён, количество пакетов в репозитории и его размер, а также другие важные параметры.

Зеркала репозитория — это серверы, дублирующие содержимое этого репозитория. Они позволяют снизить нагрузку с основных серверов.

### 3.3.2. Включить или отключить репозиторий

Вы можете по необходимости включать и отключать репозитории, чтобы установить приложение из конкретного репозитория. При этом репозиторий не будет удалён.

Команда включения репозитория:

```
$ sudo dnf config-manager --set-enabled имя_репозитория
```

Команда отключения репозитория:

```
$ sudo dnf config-manager --set-disabled имя_репозитория
```

При необходимости вы можете вывести справку по команде `config-manager`:

```
$ dnf config-manager --help-cmd
```

Вы также можете включать и отключать репозиторий из «Центра приложений» (см. «application-center»).

### 3.4. Включение автоматического обновления пакетов

Если вы хотите получать систематические обновления ОС МСВСфера Сертифицированная (ФСТЭК), вам необходимо включить доступ к репозиторию `Updates`.

Для этого выполните следующую команду:

```
$ sudo dnf config-manager --set-enabled updates
```

Для обновления установленной системы выполните следующую команду:

```
$ sudo dnf update --refresh
```

Для получения автоматических обновлений через «Центр приложений» необходимо выполнить следующие команды:

```
$ gsettings set org.gnome.software allow-updates true
$ gsettings set org.gnome.software download-updates true
```

### 3.5. Обновление системы и приложений в изолированной среде

#### 3.5.1. Введение

Если согласно политикам информационной безопасности вашей компании, все компьютеры находятся в изолированной среде (без подключения к Интернету), то вы всё также можете обновлять систему и приложения, и получать все пакеты, выпущенные для исправления уязвимостей или ошибок.

Рассмотрим два варианта обновления системы и приложений в изолированной среде.

- Создание локального зеркала репозитория ОС МСВСфера.

  В этом случае локальное зеркало будет синхронизироваться с официальными репозиториями ОС МСВСфера, а компьютеры в локальной сети будут получать обновления с него.

- Создание доверенного USB-носителя.

  В этом случае все обновления будут загружаться вами вручную на доверенный USB-носитель, а затем с него на компьютеры в локальной сети.

### 3.5.2. Создание локального зеркала репозитория ОС МСВСфера

#### 3.5.2.1. Требования

Для создания локального зеркала требуется:

- 50 Гбайт свободного пространства памяти на жёстком диске;
- настроенный HTTP-сервер.

Основное зеркало ОС МСВСфера Сертифицированная (ФСТЭК): `rsync://rsync.msvsphere-os.ru/msvsphere-certified`.

#### 3.5.2.2. Настройка зеркала

Для зеркалирования рекомендуется использовать специальный скрипт. Он позволит выполнять синхронизацию только в том случае, если появились новые обновления.

1. В приведённом ниже скрипте замените путь в переменной `_msvsphere_path` на локальный путь до вашего зеркала.

```
#!/bin/sh

# - msvsphere-sync.sh

# Timestamp file to check available updates
_timestamp="TIME"

_tempfile="/tmp/sync-$RANDOM"
_date="$(date +%s)"

# Destination path
_msvsphere_path="локальный_путь_до_вашего_зеркала"

_module="msvsphere-certified/msvsphere/"
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
if [ -f /tmp/.msvsphere-sync ]; then
  exit 0
fi

rsync -d -avSH --timeout=30 rsync://rsync.msvsphere-os.ru/
 ↪msvsphere-full/$_timestamp $_tempfile

if [ "$(diff -u $_tempfile $_msvsphere_path/$_timestamp |
 ↪wc -l)" = "0" ]; then
  exit 0
fi

rm -f $_tempfile

touch /tmp/.msvsphere-sync

false

while [ $? -ne 0 ]; do
  /usr/bin/rsync -d -avSH \
        -f 'R .~tmp~' \
        --timeout=120 \
        --delete-delay \
        --delay-updates \
        rsync://rsync.msvsphere-os.ru/$_module/ \
        $_msvsphere_path/ >> /var/log/msvsphere-sync-$_
 ↪date.log
done

rm -f /tmp/.msvsphere-sync
```

2. Затем сохраните его в удобное место, например, в `/usr/local/bin/msvsphere-sync.sh`.
3. Добавьте его в задачу в планировщик `cron`, например, в файл `/etc/cron.d/msvsphere-sync`.

```
*/5 * * * * root /usr/local/bin/msvsphere-sync.sh
```

4. Далее переключите зеркало на локальный сервер. Для этого в файлах

`/etc/yum.repos.d/msvsphere-os.repo` и `/etc/yum.repos.d/msvsphere-updates.repo` закомментируйте строку `mirrorlist` и раскомментируйте строку `baseurl`:

```
[os]
name=MSVSphere $releasever - OS
# mirrorlist=https://mirrors.inferitos.ru/mirrorlist/
 ↪$releasever/os
baseurl=https://localserver/path/$releasever/OS/$basearch/
 ↪os/
enabled=1
gpgcheck=1
countme=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-MSVSphere-
 ↪Certified-9
metadata_expire=86400
enabled_metadata=1
```

```
[updates]
name=MSVSphere $releasever - Updates
# mirrorlist=https://mirrors.inferitos.ru/mirrorlist/
 ↪$releasever/updates
baseurl=https://localserver/path/$releasever/Updates/
 ↪$basearch/os/
enabled=0
gpgcheck=1
countme=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-MSVSphere-
 ↪Certified-9
metadata_expire=86400
enabled_metadata=1
```

### 3.5.3. Создание доверенного USB-носителя

Вам потребуется доверенный USB-носитель, на который необходимо скопировать содержимое следующих репозиториев:

- https://repo1.msvsphere-os.ru/certified/msvsphere/9/OS/x86_64/os/
- https://repo1.msvsphere-os.ru/certified/msvsphere/9/Updates/x86_64/os/

1. Скопируйте содержимое указанных выше репозиториев на доверенный USB-носитель с помощью утилиты `rsync`:

```
$ rsync -rlDHP --exclude=EFI/ --exclude=images/ --
 ↪exclude=isolinux/ \
  rsync-msvsphere.inferitos.ru::msvsphere-certified/
 ↪msvsphere/9/OS/x86_64/os/ \
  /точка монтирования usb-stick/OS/

$ rsync -rlDHP \
  rsync-msvsphere.inferitos.ru::msvsphere-certified/
 ↪msvsphere/9/Updates/x86_64/os/ \
  /точка монтирования usb-stick/Updates/
```

2. Создайте в файловой системе следующие каталоги:

```
$ sudo mkdir -p /mnt/usb /mnt/repos/os/OS /mnt/repos/
 ↪updates/Updates
```

3. Подключите ваш доверенный USB-носитель к компьютеру с ОС МСВСфера Сертифицированная (ФСТЭК), находящемуся в изолированной среде. Например, если USB-носитель определился как `/dev/sdd1`, то выполните следующую команду:

```
$ sudo mount /dev/sdd1 /mnt/usb
```

   Для определения устройства вы можете использовать утилиту `lsblk`.

4. Для доступа к репозиториям подключите каталоги следующим образом:

```
$ sudo mount /mnt/usb/OS /mnt/repos/os/OS -o bind

$ sudo mount /mnt/usb/Updates /mnt/repos/updates/Updates -o
 ↪bind
```

5. Отключите все репозитории, так как из-за их недоступности могут выводиться ошибки подключения:

```
$ sudo dnf config-manager --set-disabled '*'
```

6. Для обновления выполните следующую команду:

```
$ sudo dnf --disablerepo=os --disablerepo=updates \
      --enablerepo=os-local --enablerepo=updates-local
↪update
```

# 4. ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ

## 4.1. Введение

Средства идентификации и аутентификации предоставляют возможности идентификации объектов доступа, идентификации и проверки подлинности субъектов доступа при входе в систему и при доступе к защищаемым объектам, управления идентификаторами, в том числе их создания, присвоения и уничтожения, управления аутентификационными данными, в том числе их инициализации, защищенного хранения, блокирования и разблокирования, проверки соответствия аутентификационной информации заданной метрике качества, защиты обратной связи при вводе аутентификационной информации, а также другие возможности.

## 4.2. Добавление нового пользователя

Для добавления нового пользователя используется утилита `useradd`. Она позволяет добавить учётную запись нового пользователя. Режимы её работы и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 1 - Опции утилиты `useradd` и их значения

| Опция | Значение |
|---|---|
| `-c, --comment` | Любая текстовая строка. Используется как поле для имени и фамилии пользователя, длина этого поля не должна превосходить 128 символов. |
| `-b, --base-dir` | Базовый системный каталог по умолчанию, если не указан другой каталог. Базовый каталог объединяется с именем учётной записи для определения домашнего каталога. |
| `-d, --home` | Для создаваемого пользователя в качестве начального каталога будет использован базовый каталог. По умолчанию это значение получается объединением имени пользователя с базовым каталогом и используется как имя домашнего каталога. |

| Опция | Значение |
|---|---|
| `-d, --home-dir` | Задать домашний каталог нового пользователя. Если данная опция не используется, то в качестве домашнего каталога выбирается каталог типа `/базовый_системный_каталог/имя_пользователя`. |
| `-D, --defaults` | Вывести значения стандартных опций. |
| `-e, --expiredate` | Дата окончания срока действия учётной записи пользователя. Задаётся в формате `ГГГГ-ММ-ДД`. |
| `--f, --inactive` | Число дней, которые должны пройти после окончания срока действия пароля, чтобы учётная запись заблокировалась. Если указано значение `0`, то учётная запись блокируется сразу после окончания срока действия пароля, а при значении `-1` данная возможность не используется. По умолчанию используется значение `-1`. |
| `-g, --gid` | Название группы нового пользователя или её идентификационный номер. Указываемое название группы или её номер должны существовать в системе. |
| `-G, --groups` | Список дополнительных групп, в которых числится пользователь. Перечисление групп осуществляется через запятую без пробелов. На указанные группы действуют те же ограничения, что и для группы, указанной в опции `-g`. |
| `-m, --create-home` | Создает начальный домашний каталог нового пользователя, если он ещё не существует. Если каталог уже существует, добавляемый пользователь должен иметь права на доступ к указанному каталогу. |
| `-M, --no-create-home` | Позволяет не создавать домашний каталог нового пользователя. |

| Опция | Значение |
|---|---|
| `-K, --key` | Используется для изменения значений по умолчанию для параметров, хранимых в конфигурационном файле `/etc/login.def`. |
| `-N, --no-user-group` | Позволяет добавить нового пользователя в группу, указанную в опции `-g` или заданную по умолчанию в конфигурационном файле `/etc/default/useradd`, не создавая группу, название которой совпадает с именем нового пользователя. Если опции `-g`, `-N`, `-U` не указаны, то настройки групп по умолчанию определяются в конфигурационном файле `/etc/login.defs`. |
| `-o, --non-unique` | Позволяет создать учётную запись с уже имеющимся, не уникальным идентификатором. |
| `-p, --password` | Позволяет задать новый пароль для учётной записи. |
| `-r, --system` | Позволяет создать системную учётную запись. По умолчанию для данной категории учетных записей домашний каталог не создаётся вне зависимости от значения соответствующего параметра конфигурационного файла `/etc/login.defs`. Для создания домашнего каталога системного пользователя необходимо вместе с опцией `-r` задать опцию `-m`. |
| `-s, --shell` | Полный путь к программе, используемой в качестве начального командного интерпретатора для пользователя сразу после регистрации. Длина этого поля не должна превосходить 256 символов. Если задать пустое значение, то будет использоваться оболочка по умолчанию. |

| Опция | Значение |
|---|---|
| `-u, --uid` | Позволяет задать идентификационный номер (численное неотрицательное значение идентификатора) пользователя. Это значение должно быть уникальным, если не задействована опция `-o`. |
| `U, --user-group` | Позволяет создать группу, название которой совпадает с именем пользователя, присоединив данного пользователя к этой группе. |
| `-h, --help` | Показать краткую справку об утилите. |

**Пример**: создадим пользователя с именем `user` и зададим для него основную группу `users` и две дополнительные группы `ftp` и `developers`, к которым он будет приписан.

Для этого выполним следующую команду:

```
$ sudo useradd -g users -G ftp,developers user
```

### 4.3. Изменение уже имеющихся пользовательских записей

Для изменения уже имеющихся пользовательских записей используется утилита `usermod`. Она позволяет изменить данные существующей учётной записи пользователя. Режимы ее работы и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 2 - Опции утилиты `usermod` и их значения

| Опция | Значение |
|---|---|
| `-a, --append` | Добавить пользователя в дополнительную группу. Следует использовать только вместе с параметром `-G`. |
| `-c, --comment` | Новое значение поля комментария. |

| Опция | Значение |
|---|---|
| `d, --home` | Новый домашний каталог учётной записи. Если указан параметр `-m`, то содержимое текущего домашнего каталога будет перемещено в новый домашний каталог, который будет создан, если он ещё не существует. |
| `-e, --expiredate` | Установить дату окончания срока действия учётной записи в формате `ГГГГ-ММ-ДД`. |
| `-f, --inactive` | Установить пароль после окончания срока действия учётной записи в `INACTIVE`. Если указано значение `0`, то учётная запись блокируется сразу после окончания срока действия пароля, а при значении `-1` данная возможность не используется. По умолчанию используется значение `-1`. |
| `-g, --gid` | Принудительно назначить первичную группу. |
| `-G, --groups` | Список дополнительных групп. |
| `-l, --login` | Новое значение учётной записи. |
| `-L, --lock` | Заблокировать пароль пользователя. Это делается помещением символа `!` в начало шифрованного пароля, что приводит к его блокировке. Не следует использовать этот параметр вместе с `-p` или `-U`. |
| `-m, --move-home` | Переместить содержимое домашнего каталога пользователя в новое место. Если новый домашний каталог не существует, то он создаётся автоматически. Данная опция используется только вместе с опцией `-d`. |
| `-o, --non-unique` | При использовании с параметром `-u` этот параметр позволяет указывать не уникальный числовой идентификатор пользователя. |
| `-p, --password` | Задать новый пароль для учётной записи. |
| `-s, --shell` | Задать новую оболочку для учётной записи. |

| Опция | Значение |
|---|---|
| `-u, --uid` | Новый идентификационный номер для учётной записи. |
| `-U, --unlock` | Разблокировать учетную запись. |

**Пример**: изменим срок действия учётной записи пользователя с идентификатором `user6`.

Для этого выполним следующую команду:

```
$ sudo usermod -e 2020-05-01 user6
```

где 2020-05-01 — дата истечения срока действия учётной записи в формате `ГГГГ-ММ-ДД`.

**Пример**: изменим идентификатор (значение учётной записи) пользователя с `user6` на `user7`.

Для этого выполним следующую команду:

```
$ sudo usermod -l user7 user6
```

### 4.4. Удаление пользователей

Для удаления пользователей используется утилита `userdel`. Она позволяет удалить существующую учетную запись пользователя. Режимы ее работы и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 3 - Опции утилиты `userdel` и их значения

| Опция | Значение |
|---|---|
| `-r, --remove` | Файлы в домашнем каталоге пользователя будут удалены вместе с самим домашним каталогом и почтовым ящиком. Пользовательские файлы, расположенные в других файловых системах, нужно искать и удалять вручную. |

| Опция | Значение |
|---|---|
| `-f, --force` | С этой опцией учётная запись будет удалена, даже если пользователь в этот момент работает в системе. Она также заставляет утилиту удалить домашний каталог пользователя и почтовый ящик, даже если другой пользователь использует тот же домашний каталог или если почтовый ящик не принадлежит данному пользователю. **Внимание! Перед использованием этого параметра убедитесь в необходимости этого действия! Этот параметр может привести систему в нерабочее состояние!** |
| `-n` | Задает, сколько месяцев идентификатор пользователя должен устаревать перед повторным использованием. Задайте `-1`, чтобы указать, что идентификатор пользователя никогда не должен использоваться повторно. Задайте `0`, чтобы указать, что идентификатор пользователя можно немедленно использовать повторно. Если опция `-n` не задана, то идентификатор будет устаревать стандартное количество месяцев перед повторным использованием. |
| `-h, --help` | Показать краткую справку. |

**Пример**: удалим пользователя с идентификатором `user7`.

Для этого выполним следующую команду:

```
$ sudo userdel -r user7
```

### 4.5. Добавление группы пользователей

Для добавления группы пользователей используется утилита `groupadd`. Режимы ее работы и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 4 - Опции утилиты `groupadd` и их значения

| Опция | Значение |
|---|---|
| `-f` | Вернуть статус успешного выполнения, если группа уже существует. Если используется вместе с параметром `-g` и указанный идентификатор группы уже существует, то выбирается другой уникальный идентификатор группы, то есть параметр `-g` игнорируется. |
| `-g` | Числовое значение идентификатора группы. Значение должно быть уникальным, если не задан параметр `-o`. Значение должно быть не отрицательным. По умолчанию берётся значение больше 999 и больше идентификатора любой другой группы. Значения от 0 и до 999 обычно зарезервированы под системные группы. |
| `-K` | Изменить значения по умолчанию для параметров, которые хранятся в конфигурационном файле `/etc/login.defsм`. |
| `-o` | Разрешить добавление группы с не уникальным идентификатором. |
| `-r, --system` | Создать системную группу. |
| `-h, --help` | Показать краткую справку. |

**Пример**: создадим группу `group2` с числовым значением идентификатора `8285`.

Для этого выполним следующую команду:

```
$ sudo groupadd group2 -g 8285
```

### 4.6. Изменение существующей группы пользователей

Для изменения существующей группы пользователей используется утилита `groupmod`. Режимы ее работы и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 5 - Опции утилиты `groupmod` и их значения

| Опция | Значение |
|---|---|
| `-g, --gid` | Изменить идентификатор группы. |
| `-n, --new-name` | Изменить имя группы. |
| `-o, --non-unique` | Позволяет использовать не уникальный идентификатор группы. |
| `-p, --password` | Изменить пароль. |
| `-h, --help` | Показать краткую справку. |

**Пример**: изменим идентификатор группы пользователей `users` на `ftp`.

Для этого выполним следующую команду:

```
$ sudo groupmod -g ftp users
```

### 4.7. Удаление существующей группы пользователей

Для удаления существующей группы пользователей используется утилита `groupdel`. Утилита позволяет удалить определение группы из системы путем удаления записи о соответствующей группе из файла `/etc/group`. Однако она не удаляет идентификатор группы из файла паролей. Удаленный идентификатор действует для всех файлов и каталогов, которые его имели.

**Пример**: удалим группу с именем `group3`.

Для этого выполним следующую команду:

```
$ sudo groupdel group3
```

### 4.8. Создание и изменение пароля пользователя

Для создания и изменения пароля пользователя (в том числе для блокировки учётной записи пользователя) используется утилита `passwd`. Обычный пользователь может изменить пароль только своей учётной записи, суперпользователь `root` может изменить пароль любой учётной записи.

При изменении пароля проверяется информация об устаревании пароля, чтобы убедиться, что пользователю разрешено изменять пароль в настоящий момент. Если выяснится, что не разрешено, то утилита не производит изменение пароля и завершает работу.

При изменении пароля пользователь должен будет сначала ввести старый пароль, если он был. Введенное пользователем значение старого пароля зашифровывается и сравнивается со значением зашифрованного текущего пароля. Затем пользователю необходимо будет дважды ввести новый пароль. Значение второго ввода сравнивается с первым, и они должны совпасть. После этого пароль тестируется на сложность подбора, т.е. его значение не должно быть легко угадываемым.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 6 - Опции утилиты `passwd` и их значения

| Опция | Значение |
|---|---|
| `-a, --all` | Эту опцию можно использовать только вместе с `-S` для вывода статуса всех пользователей. |
| `-d, --delete` | Удалить пароль пользователя (сделать его пустым). Это быстрый способ заблокировать пароль учётной записи. |
| `-e, --expire` | Немедленно сделать пароль устаревшим. Это заставит пользователя изменить пароль при следующем входе в систему. |

| Опция | Значение |
|---|---|
| `-i, --inactive` | Эта опция используется для блокировки учётной записи по прошествии заданного числа дней после устаревания пароля. То есть если пароль устарел и прошло больше дней, чем указано, то пользователь больше не сможет использовать свою учётную запись. |
| `-l, --lock` | Заблокировать указанную учётную запись. Эта опция блокирует учётную запись путем изменения значения пароля на такое, которое не может быть ранее указанным зашифрованным паролем. |
| `-m, --mindays` | Задать минимальное количество дней между сменой пароля. Нулевое значение этого поля указывает на то, что пользователь может менять свой пароль тогда, когда захочет. |
| `-S, --status` | Показать состояние учётной записи. Информация о состоянии содержит семь полей. Первое поле содержит имя учётной записи. Второе поле указывает, заблокирована ли учётная запись, она без пароля или у неё есть рабочий пароль. Третье поле хранит дату последнего изменения пароля. В следующих четырёх полях хранятся минимальный срок, максимальный срок, период выдачи предупреждения и период неактивности пароля. Все эти сроки измеряются в днях. |
| `-u, --unlock` | Разблокировать указанную учётную запись. Этот параметр активирует учётную запись путем изменения пароля на прежнее значение, которое было перед использованием параметра `-l`. |
| `-w, --warndays` | Установить число дней выдачи предупреждения, перед тем как потребуется смена пароля. |

| Опция | Значение |
|---|---|
| `-x, --maxdays` | Установить максимальное количество дней, в течение которых пароль остаётся рабочим, после чего его надо будет изменить. |
| `-h, --help` | Показать краткую справку. |

**Пример**: зададим пароль пользователю `user4`. Работа команды `passwd`:

```
$ sudo passwd user4
Изменяется пароль пользователя user4.
Новый пароль :
Повторите ввод нового пароля :
passwd: все данные аутентификации успешно обновлены.
```

**Пример**: посмотрим состояние учётной записи `user4`. Работа команды `passwd`:

```
$ sudo passwd -S user4
user4 PS 2023-07-04 0 99999 7 -1 (Пароль задан, шифр SHA512.)
```

Где:

- `user4` — имя пользователя.
- `PS` — статус пароля.
- `2023-07-04` — отображает время последнего изменения пароля.
- `0` и `99999` — минимальный и максимальный срок действия пароля.
- `7` — срок вывода предупреждения.
- `-1` — срок деактивации пароля.

### 4.9. Изменение срока действия учётной записи и пароля пользователя

Утилита `chage` позволяет установить дату завершения срока действия учётной записи пользователя, минимальный и максимальный срок действия пароля, дату завершения срока действия пароля, а также количество дней, в течение которых пользователю будут выводиться предупреждения о приближении завершения срока действия пароля.

Командой chage может пользоваться только суперпользователь, за исключением использования её с параметром -l, который позволяет непривилегированным пользователям определить время, когда истекает их личный пароль или учетная запись.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 7 - Опции утилиты chage и их значения

| Опция | Значение |
|---|---|
| -m | Меняет значение mindays на минимальное число дней между сменой пароля. Значение 0 в этом поле обозначает, что пользователь может изменять свой пароль когда угодно. |
| -M | Меняет значение maxdays на максимальное число дней, в течение которых пароль будет действителен. Когда сумма maxdays и lastday меньше, чем текущий день, у пользователя будет запрошен новый пароль до начала работы в системе. |
| -d | Меняет значение lastday на день, когда пароль был изменен последний раз (число дней с 1 января 1970). Дата также может быть указана в формате ГГГГ-ММ-ДД. |
| -E | Используется для задания даты, с которой учетная запись пользователя станет недоступной. Дата также может быть указана в формате ГГГГ-ММ-ДД. |
| -I | Используется для задания количества дней «неактивности», то есть дней, когда пользователь вообще не входил в систему, после которых его учетная запись будет заблокирована. Значение 0 отключает этот режим. |

| Опция | Значение |
|---|---|
| `-W` | Используется для задания числа дней, когда пользователю начнет выводиться предупреждение об истечении срока действия его пароля и необходимости его изменения. |
| `-l` | Просмотреть текущую информацию о дате истечения срока действия пароля для пользователя. |

**Пример**: просмотрим текущую информацию о дате истечения срока действия пароля для пользователя `user4`. Работа команды `chage`:

```
$ sudo chage -l user4
Последний раз пароль был изменён: мар 12, 2023
Срок действия пароля истекает: никогда
Пароль будет деактивирован через: никогда
Срок действия учётной записи истекает: никогда
Минимальное количество дней между сменой пароля: 0
Максимальное количество дней между сменой пароля: 99999
Количество дней с предупреждением перед деактивацией пароля: 7
```

### 4.10. Управление политиками паролей

В данном разделе описана процедура управления политиками паролей на локальной системе, не подключённой к LDAP-каталогу пользователей (FreeIPA, Microsoft Active Directory и т.д.). В случае использования LDAP-каталога обратитесь к соответствующему руководству по администрированию.

Для управления политиками паролей в МСВСфера ОС используется системная утилита `authselect`, которая оперирует профилями аутентификации. В первую очередь необходимо убедиться, что выбран профиль аутентификации. Вы можете это сделать с помощью команды `authselect current`:

```
# вывод для систем, подключённых к каталогу пользователей FreeIPA
$ sudo authselect current
Profile ID: sssd
Enabled features:
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
- with-mkhomedir
- with-sudo

# вывод для систем, использующих профиль "minimal"
$ sudo authselect current
Profile ID: minimal
Enabled features: None

# вывод для систем, не использующих профиль authselect
$ sudo authselect current
Конфигурация не обнаружена. / No existing configuration detected.
```

Если система не настроена на использование профиля аутентификации, то необходимо выбрать его, чтобы получить возможность настраивать политики паролей.

Просмотреть список доступных профилей `authselect` вы можете следующим образом:

```
$ sudo authselect list
- minimal              Local users only for minimal installations
- sssd                 Enable SSSD for system authentication (also
 ↪for local users only)
- winbind              Enable winbind for system authentication
- custom/minimal_gost  Local users only for minimal installations
 ↪and gost support
- custom/sssd_gost     Enable SSSD with GOST support for system
 ↪authentication (also for local users only)
```

Выбрать профиль вы можете с помощью команды `authselect select`. Для локальной системы рекомендуется использовать профиль `minimal`:

```
$ sudo authselect select minimal --force
[error] File [/etc/authselect/system-auth] is still present
[error] File [/etc/authselect/password-auth] is still present
[error] File [/etc/authselect/fingerprint-auth] is still present
[error] File [/etc/authselect/smartcard-auth] is still present
[error] File [/etc/authselect/postlogin] is still present
[error] File [/etc/authselect/nsswitch.conf] is still present
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
[error] File [/etc/authselect/dconf-db] is still present
[error] File [/etc/authselect/dconf-locks] is still present
[error] Link [/etc/pam.d/system-auth] points to [/etc/authselect/
 ↪system-auth]
[error] Symbolic link [/etc/pam.d/system-auth] to [/etc/authselect/
 ↪system-auth] still exists!
[error] Link [/etc/pam.d/password-auth] points to [/etc/authselect/
 ↪password-auth]
[error] Symbolic link [/etc/pam.d/password-auth] to [/etc/
 ↪authselect/password-auth] still exists!
[error] Link [/etc/pam.d/fingerprint-auth] points to [/etc/
 ↪authselect/fingerprint-auth]
[error] Symbolic link [/etc/pam.d/fingerprint-auth] to [/etc/
 ↪authselect/fingerprint-auth] still exists!
[error] Link [/etc/pam.d/smartcard-auth] points to [/etc/
 ↪authselect/smartcard-auth]
[error] Symbolic link [/etc/pam.d/smartcard-auth] to [/etc/
 ↪authselect/smartcard-auth] still exists!
[error] Link [/etc/pam.d/postlogin] points to [/etc/authselect/
 ↪postlogin]
[error] Symbolic link [/etc/pam.d/postlogin] to [/etc/authselect/
 ↪postlogin] still exists!
[error] Link [/etc/nsswitch.conf] points to [/etc/authselect/
 ↪nsswitch.conf]
[error] Symbolic link [/etc/nsswitch.conf] to [/etc/authselect/
 ↪nsswitch.conf] still exists!
[error] Link [/etc/dconf/db/distro.d/20-authselect] points to [/
 ↪etc/authselect/dconf-db]
[error] Symbolic link [/etc/dconf/db/distro.d/20-authselect] to [/
 ↪etc/authselect/dconf-db] still exists!
[error] Link [/etc/dconf/db/distro.d/locks/20-authselect] points to
 ↪[/etc/authselect/dconf-locks]
[error] Symbolic link [/etc/dconf/db/distro.d/locks/20-authselect]
 ↪to [/etc/authselect/dconf-locks] still exists!
Backup stored at /var/lib/authselect/backups/2024-10-08-16-28-25.
 ↪N8QZyv
Profile "minimal" was selected.
The following nsswitch maps are overwritten by the profile:
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
- aliases
- automount
- ethers
- group
- hosts
- initgroups
- netgroup
- networks
- passwd
- protocols
- publickey
- rpc
- services
- shadow
```

Проверить корректность применения профиля вы можете следующим образом:

```
$ sudo authselect current
Profile ID: minimal
Enabled features: None

$ sudo authselect check
Current configuration is valid.
```

### 4.10.1. Управление требованиями к качеству паролей

По умолчанию МСВСфера ОС предъявляет следующие требования к качеству паролей пользователя:

- пароль должен иметь длину как минимум 8 символов;
- пароль должен отсутствовать в словаре известных паролей программы `cracklib`.

За проверку качества паролей отвечает PAM-модуль `pam_pwquality`, который включён по умолчанию для всех профилей аутентификации.

Модуль настраивается через конфигурационный файл `/etc/security/pwquality.conf`, который по умолчанию имеет следующий вид (для переменных указаны значения по умолчанию, описание параметров переведено на русский

язык и добавлены комментарии):

```
# Количество символов в новом пароле, которые не должны
 ↪присутствовать в старом
# пароле. Значение 0 полностью отключает проверку на пересечение
 ↪символов, за
# исключением попытки использования идентичного пароля.
# difok = 1

# Минимально допустимое количество символов в новом пароле (плюс
 ↪один, если
# использование кредитов не отключено, что является поведением по
 ↪умолчанию).
# Пароль не может быть короче 6 символов.
# minlen = 8

# Максимальное количество кредитов, начисляемое за наличие цифр в
 ↪новом пароле.
# Если значение меньше 0, то это минимальное количество цифр в
 ↪новом пароле.
# dcredit = 0

# Максимальное количество кредитов, начисляемое за наличие
 ↪прописных букв в
# новом пароле. Если значение меньше 0, то это минимальное
 ↪количество прописных
# букв в новом пароле.
# ucredit = 0

# Максимальное количество кредитов, начисляемое за наличие строчных
 ↪букв в
# новом пароле. Если значение меньше 0, то это минимальное
 ↪количество строчных
# букв в новом пароле.
# lcredit = 0

# Максимальное количество кредитов, начисляемое за наличие других
 ↪символов в
# новом пароле. Если значение меньше нуля, то это минимальное
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪количество других
# символов в новом пароле.
# ocredit = 0

# Минимальное количество требуемых классов символов в новом пароле
# (цифры, буквы в нижнем регистре, буквы в верхнем регистре, другие
 ↪символы).
# minclass = 0

# Максимальное количество разрешённых повторяющихся символов в
 ↪новом пароле.
# Проверка отключается, если значение равно 0.
# maxrepeat = 0

# Максимальное количество повторяющихся символов из одного класса,
 ↪разрешённое
# в новом пароле. Проверка отключается, если значение равно 0.
# maxclassrepeat = 0

# Проверять, есть ли слова из поля GECOS пользователя в новом
 ↪пароле. Проверка
# отключается, если значение равно 0.
# gecoscheck = 0

# Проверять наличие пароля в словаре cracklib, если значение не
 ↪равно 0.
# dictcheck = 1

# Проверять наличие имени пользователя в новом пароле. Проверка
 ↪отключается,
# если значение равно 0.
# usercheck = 1

# Длина подстрок из имени пользователя, которую нужно проверить на
 ↪наличие
# в новом пароле. Эта проверка выполняется, если значение больше 0
 ↪и значение
# usercheck равно 1.
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# usersubstr = 0

# Включить принудительную проверку нового пароля пользователя,
 ↪новый пароль,
# не соответствующий требованиям, будет отклонён, если значение не
 ↪равно 0.
# enforcing = 1

# Путь к словарям cracklib. Если не задан, будет использован
 ↪стандартный
# словарь cracklib.
# dictpath =

# Сколько раз запрашивать новый пароль пользователя прежде чем
 ↪выводить ошибку.
# Значение по умолчанию - 1.
# retry = 3

# Применять требования к качеству пароля пользователя root, если
 ↪опция
# раскомментирована.
# enforce_for_root

# Применять требования к качеству пароля только для локальных
 ↪пользователей,
# присутсвующих в файле /etc/passwd, если эта опция
 ↪раскомментирована.
# local_users_only
```

Кредиты в параметрах `dcredit`, `ucredit`, `lcredit` и `ocredit` определяют сколько баллов может быть начислено за определённый тип символов, используемый в новом пароле. Если значение параметра больше `0`, то за каждый такой символ к общей длине пароля добавляется определённое количество кредитов.

Пример: если все четыре параметра имеют значение `1` и минимально допустимая длина пароля составляет 7 символов, то при использовании всех 4 типов символов потребуется ввести пароль всего лишь из 7 символов. За каждый

неиспользованный тип символа к требуемой длине пароля добавляется штраф, указанный в соответствующем параметре. Так, если не использовать цифры и строчные буквы, то минимальная длина пароля уже составит 9 символов.

Использование механизма кредитов позволяет ослабить требования к длине пароля за счёт использования символов из разных групп.

После внесения правок в конфигурационный файл `/etc/security/pwquality.conf` перезапуск каких-либо сервисов не требуется — PAM применит изменения автоматически и новые настройки будут использованы при следующем изменении пароля.

Дополнительную информацию по использованию модуля `pam_pwquality` вы можете найти в соответствующих руководствах:

- `man pam_pwquality`;
- `man pwquality.conf`.

### 4.10.2. Ограничение на повторное использование паролей

В конфигурации по умолчанию операционная система не позволяет повторно использовать только текущий пароль пользователя. Это поведение можно изменить с помощью PAM-модуля `pam_pwhistory`, который позволяет хранить историю паролей для каждого пользователя.

Для включения модуля `pam_pwhistory` выполните следующую команду:

```
$ sudo authselect enable-feature with-pwhistory
```

После этого свойство `with-pwhistory` должно появиться в свойствах текущего профиля аутентификации:

```
$ sudo authselect current
Profile ID: minimal
Enabled features:
- with-faillock
- with-pwhistory
```

Модуль настраивается через конфигурационный файл `/etc/security/pwhistory.conf`, который по умолчанию имеет следующий вид (для переменных указаны значения по умолчанию, описание параметров переведено на русский язык и добавлены комментарии):

```
# Раскомментирование этой опции включает вывод отладочной
 ↪информации.
# debug

# Так же сохранять предыдущие пароли пользователя root, если эта
 ↪опция
# раскомментирована.
# enforce_for_root

# Количество сохраняемых паролей для каждого пользователя.
# remember = 10

# Сколько раз запрашивать новый пароль пользователя прежде чем
 ↪выводить ошибку.
# retry = 1

# Каталог, в котором будут храниться предыдущие пароли
 ↪пользователей.
# file = /etc/security/opasswd
```

Исходя из описания выше, после включения модуля pam_pwhistory, система будет хранить последние 10 паролей для каждого пользователя и выдавать ошибку при попытке использовать сохранённый пароль в качестве нового при смене пароля.

Пример ошибки:

```
# на английском языке
$ passwd
Changing password for user test.
Current password:
New password:
Retype new password:
Password has been already used. Choose another.
passwd: Have exhausted maximum number of retries for service

# на русском языке
$ passwd
Изменение пароля пользователя test.
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Текущий пароль:
Новый пароль:
Повторите ввод нового пароля:
Этот пароль уже был использован. Выберите другой.
```

Дополнительную информацию по использованию модуля pam_pwhistory вы можете найти в соответствующих руководствах:

- man pam_pwhistory;
- man pwhistory.conf.

### 4.10.3. Ограничение количества неуспешных попыток аутентификации

В конфигурации по умолчанию операционная система не ограничивает количество неуспешных попыток аутентификации пользователя. Однако, реализация такого ограничения возможна с помощью PAM-модуля pam_faillock.

Для включения модуля pam_faillock выполните следующую команду:

```
$ sudo authselect enable-feature with-faillock
```

После этого свойство with-faillock должно появиться в свойствах текущего профиля аутентификации:

```
$ sudo authselect current
Profile ID: minimal
Enabled features:
- with-faillock
```

Модуль настраивается через конфигурационный файл /etc/security/faillock.conf, который по умолчанию имеет следующий вид (для переменных указаны значения по умолчанию, описание параметров переведено на русский язык и добавлены комментарии):

```
# Каталог, в котором хранятся файлы с записями об ошибках
 ↪аутентификации
# пользователей.
# Внимание: в случае изменения этого пути потребуется дополнительно
# перенастроить правила SELinux.
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# dir = /var/run/faillock

# Раскомментирование этой опции включает логирование имён
 ↪несуществующих
# пользователей в системный журнал.
# audit

# Раскомментирование этой опции отключает вывод информационных
 ↪сообщений на
# консоль.
# silent

# Раскомментирование этой опции отключает вывод информационных
 ↪сообщений в
# системный журнал.
# no_log_info

# Раскомментирование этой опции включает отслеживание неудачных
 ↪попыток
# аутентификации только для локальных пользователей, указанных в
 ↪файле
# /etc/passwd. В таком случае пользователи из LDAP-каталога будут
# игнорироваться PAM-модулем "faillock". Включение данного
 ↪параметра позволяет
# избежать ситуаций с двойной блокировкой, когда пользователь будет
 ↪заблокирован
# и локально, и на уровне каталога пользователей.
# local_users_only

# Блокировать доступ пользователю, если количество последовальных
 ↪ошибок
# аутентификации за последний промежуток времени превышает N
 ↪попыток.
# deny = 3

# Временной интервал в секундах, в течении которого ошибки
 ↪аутентификации
# пользователя считаются последовательными и приводят к блокировке
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪его учётной
# записи. Значение по умолчанию — 15 минут.
# fail_interval = 900

# Автоматически восстановить доступ к заблокированной учётной
 ↪записи спустя N
# секунд после блокировки. Для перманентной блокировки учётной
 ↪записи задайте
# значение 0 — в таком случае восстановление доступа будет возможно
 ↪только
# вручную через команду "faillock". Значение по умолчанию — 10
 ↪минут.
# unlock_time = 600

# Раскомментирование этой опции также приведёт к блокировке
 ↪пользователя
# root в случае неудачных попыток аутентификации. По умолчанию
 ↪блокируются
# только обычные пользователи.
# even_deny_root

# Автоматически восстанавливать доступ к заблокированной учётной
 ↪записи
# пользователя root спустя N секунд после блокировки. Если значение
 ↪не
# определено, модуль "faillock" будет использовать значение
 ↪параметра "unlock_time".
# root_unlock_time = 900

# Имя группы системных администраторов. Если задано, то для
 ↪участников этой
# группы будут применяться те же правила, что и для пользователя
 ↪root
# (к ним будут применяться параметры "even_deny_root" и "root_
 ↪unlock_time").
# admin_group = <admin_group_name>
```

Исходя из описания выше, при включении PAM-модуля `pam_faillock` будет автоматически применена следующая конфигурация: локальные пользователи

и пользователи из LDAP-каталога будут блокироваться после трёх неудачных попытках входа в течении 15 минут на срок в 10 минут. Пользователь `root` блокироваться не будет.

После внесения правок в конфигурационный файл `/etc/security/faillock.conf` перезапуск каких-либо сервисов не требуется — PAM применит изменения автоматически при обработке следующего запроса на аутентификацию.

Функциональность модуля `pam_faillock` распространяется как на локальную (через графическую или текстовую консоль), так и на сетевую (с помощью SSH) аутентификацию пользователей.

Для просмотра истории неудачных попыток аутентификации пользователя за учётный период `fail_interval` вы можете использовать следующую команду (замените «USERNAME» на имя реального пользователя из вашей системы):

```
$ sudo faillock --user USERNAME
When                Type  Source                                Valid
2024-10-08 21:12:20 RHOST 192.168.1.4                           V
2024-10-08 21:12:25 RHOST 192.168.1.4                           V
2024-10-08 21:12:28 RHOST 192.168.1.4                           V
```

Для снятия блокировки учётной записи пользователя вы можете использовать следуюзую команду:

```
$ sudo faillock --user USERNAME --reset
```

Дополнительную информацию по использованию модуля `pam_faillock` вы можете найти в соответствующих руководствах:

- `man pam_faillock`;
- `man faillock`;
- `man faillock.conf`.

### 4.11. Получение сведений о пользователе

Утилита `id` позволяет получить сведения об указанном пользователе или о текущем пользователе, запустившем данную утилиту, если он не указал явно имя пользователя.

По умолчанию выводятся числовые идентификаторы пользователя и группы, действующие идентификаторы пользователя и группы, а также идентификаторы других групп, в которых состоит пользователь.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 8 - Опции утилиты `id` и их значения

| Опция | Значение |
|---|---|
| `-g, --group` | Выводит только подлинный числовой идентификатор групп. |
| `-G, --groups` | Выводит все подлинные числовые идентификаторы групп, в которых состоит пользователь. |
| `-n, --name` | Выводит действующие имена пользователей или групп. |
| `-r, --real` | Выводит подлинные числовые идентификаторы пользователей или групп. |
| `-u, --user` | Выводит только подлинный числовой идентификатор пользователя. |
| `--version` | Выводит информацию о версии утилиты и завершает работу. |
| `--help` | Выводит справку по этой утилите и завершает работу. |

**Пример**: выведем сведения о текущем пользователе `user`. Работа команды `id`:

```
$ id
uid=1000(user) gid=1000(user) группы=1000(user),100(users)
 ↪контекст=user_u:user_r:user_t:s0
```

### 4.12. Конфигурационный файл /etc/login.defs

Конфигурационный файл `/etc/login.defs` позволяет задавать параметры, определяющие использование пользователями своих паролей.

```
# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

Рис. 17: Фрагмент файла

Параметры перечислены в таблице.

Таблица 9 - Параметры конфигурационного файла `/etc/login.defs` и их описание

| Параметр | Описание |
|---|---|
| `PASS_MAX_DAYS` | Определяет максимальный срок действия пароля, т.е. максимальное число дней, в течение которых действие пароля сохраняется. По истечении этого срока запускается процесс принудительной смены пароля. Если значение параметра не задано, то есть параметр закомментирован символом # или ему присвоено значение `-1`, то данное ограничение не установлено (отменяется). |
| `PASS_MIN_DAYS` | определяет минимальный срок между изменениями пароля, т.е. минимальное число дней между двумя последовательными изменениями пароля. Если значение параметра не задано, то есть параметр закомментирован символом # или ему присвоено значение `-1`, то данное ограничение не установлено (отменяется). |
| `PASS_MIN_LEN` | Определяет минимальную допустимую длину задаваемого пароля. |

| Параметр | Описание |
|---|---|
| PASS_WARN_AGE | Определяет, за сколько дней до истечения срока действия пароля начнётся вывод предупреждения о необходимости его смены. Если значение параметра не задано, то есть параметр закомментирован символом # или ему присвоено значение -1, то данное ограничение не установлено (отменяется). Если значение параметра 0, то предупреждение о необходимости смены пароля будет выведено в день его устаревания. |

**Пример**: выведем на экран текущее заданное значение максимального количества дней действия пароля:

```
$ cat /etc/login.defs | grep PASS_MAX_DAYS
# PASS_MAX_DAYS  Maximum number of days a password may be used.
PASS_MAX_DAYS 30
```

Мы видим, что текущее максимальное количество дней действия пароля — 30 дней.

### 4.13. Конфигурационный файл /etc/pam.d/system-auth

Конфигурационный файл `/etc/pam.d/system-auth` позволяет задавать настройки подключаемых модулей аутентификации.

Каждая строка в нём представляет собой правило, состоящее из трёх обязательных полей и одного опционального. Поля разделены символом пробела. Порядок, в котором указаны правила, определяет очередность их проверки.

```
# Generated by authselect on Fri Jul 14 14:08:55 2023
# Do not modify this file manually.

auth        required                                      pam_env.so
auth        required                                      pam_faildelay.so delay=20000
00
auth        sufficient                                    pam_fprintd.so
auth        [default=1 ignore=ignore success=ok]          pam_usertype.so isregular
auth        [default=1 ignore=ignore success=ok]          pam_localuser.so
auth        sufficient                                    pam_unix.so nullok
auth        [default=1 ignore=ignore success=ok]          pam_usertype.so isregular
auth        sufficient                                    pam_sss.so forward_pass
auth        required                                      pam_deny.so

account     required                                      pam_unix.so
account     sufficient                                    pam_localuser.so
account     sufficient                                    pam_usertype.so issystem
account     [default=bad success=ok user_unknown=ignore] pam_sss.so
account     required                                      pam_permit.so
```

Рис. 18: Фрагмент файла

**Синтаксис правила**:

```
type control module-path [module-arguments]
```

Поле `type` задаёт тип вызываемого модуля и может принимать одно из четырех допустимых значений:

- `auth` — предназначен для аутентификации пользователя путём запроса и проверки его пароля;
- `account` — используется для контроля доступа к сервису/приложению. Например, может быть произведён запрос о том, не истёк ли срок действия аккаунта пользователя, разрешено ли пользователю работать с определённым сервисом в определённое время, хватает ли системных ресурсов для работы;
- `password` — применятся для установки/изменения паролей;
- `session` — управляет действиями пользователя в рамках активной сессии после его успешной аутентификации в системе.

Поле `control` задаёт действие, которое нужно выполнить после вызова модуля. Доступно несколько действий:

- `required` — модуль должен вернуть положительный ответ. Если он возвращает отрицательный ответ, то пользователь будет уведомлен об этом только после того, как все остальные модули данного типа будут проверены;

- `requisite` — требует от модуля положительный ответ. В случае получения отрицательного ответа последовательная проверка выполнения остальных правил моментально прекращается и пользователь получает сообщение об ошибке аутентификации;
- `sufficient` — в случае, если ни один из модулей с действием `required` или `sufficient`, расположенных перед текущим, не вернул отрицательного ответа, текущий модуль вернёт положительный ответ и все последующие модули будут проигнорированы;
- `optional` — результат проверки модуля важен только в том случае, если действие является единственным для данного модуля;
- `include` — предназначается для добавления строк заданного типа из других файлов конфигурации из каталога `/etc/pam.d/` в файл конфигурации `/etc/pam.d/system-auth`. Название файла указывается в качестве аргумента действия.

Поле `module-path` задаёт путь к вызываемому модулю.

Поле `module-arguments` — дополнительные необязательные параметры модуля, необходимые для определения действий некоторых отдельных модулей в случае успешной авторизации. Так, если в конфигурационном файле найти строку, содержащую `pam_pwquality.so`, и добавить в нее `minlen=8`, то будет установлена минимальная длина пароля, равная 8-ми символам.

**Пример**: В качестве примера сделаем блокировку учётной записи пользователя, который совершит определенное количество неудачных попыток входа в систему.

Для этого внесем в файл `/etc/pam.d/system-auth` следующие изменения:

1. Сначала допишем в секцию `auth` строку `auth required pam_tally2.so deny=2 onerr=fail`, т.е. подключим модуль `pam_tally2` и установим блокировку пользователя после двух (значение параметра `deny`) неудачных попыток входа.
2. Затем в секции `account` добавим строку `account required pam_tally2.so` и закомментируем строки вида `auth requisite pam_succeed_if.so uid >= 1000 quiet` и `auth required pam_deny.so`.
3. Потом строку `auth sufficient pam_unix.so nullok try_first_pass` заменим на `auth required pam_unix.so nullok try_first_pass`.

После этого пользователь, допустивший подряд две неверных попытки входа, на третьей получит сообщение о том, что его учетная запись заблокирована. И даже если четвертой попыткой он введет верный пароль, то все равно не получит доступ к системе.

```
$ sudo user2
Пароль:
sudo Сбой при проверке подлинности

$ sudo user2
Пароль:
sudo Сбой при проверке подлинности

$ sudo user2
Пароль:
Учетная запись заблокирована как следствие неудачных попыток входа
 ↪(всего 3)
sudo Сбой при проверке подлинности

$ sudo user2
Пароль:
Учетная запись заблокирована как следствие неудачных попыток входа
 ↪(всего 4)
sudo Сбой при проверке подлинности
```

**Пример**: В качестве другого примера настроим проверку паролей на сложность подбора через `pam_cracklib`.

1. Для этого добавим или изменим следующую строку:

```
password requisite pam_cracklib.so try_first_pass retry=3 type=
 ↪minlen=6 dcredit=-2 ucredit=-3 lcredit=-2 ocredit=-1
```

Это значит следующее:

- после трех неуспешных попыток (`retry=3`) модуль вернет ошибку;
- минимальная длина для пароля — 6 символов (`minlen=6`);
- минимальное количество цифр — 2 (`dcredit=-2`);
- минимальное количество символов верхнего регистра — 3 (`ucredit=-3`);
- минимальное количество символов нижнего регистра — 2 (`lcredit=-2`);
- минимальное количество других символов — 1 (`ocredit=-1`).

2. Удалим или закомментируем следующую строку:

```
password requisite pam_pwquality.so try_first_pass local_users_only
 ↪retry=3 authtok_type=
```

**Результат**

- Выполним команду `passwd` для смены пароля пользователя `user2`.
- Зададим пароль из трех символов и увидим сообщение «НЕУДАЧНЫЙ ПАРОЛЬ: слишком короткий».
- Зададим пароль из четырех символов, система выдаст сообщение «НЕУДАЧНЫЙ ПАРОЛЬ: короткий».
- Зададим пароль из шести символов (букв и цифр), в результате чего получим сообщение «НЕУДАЧНЫЙ ПАРОЛЬ: слишком простой».
- После трех неуспешных попыток модуль вернет ошибку.

```
$ passwd
Изменяется пароль пользователя user2.
Смена пароля для user2.
(текущий) пароль Unix:
Новый пароль:
НЕУДАЧНЫЙ ПАРОЛЬ: слишком короткий
Новый пароль:
НЕУДАЧНЫЙ ПАРОЛЬ: короткий
Новый пароль:
НЕУДАЧНЫЙ ПАРОЛЬ: слишком простой
passwd: Использовано максимальное число попыток, заданное для
 ↪службы
```

- Зададим пароль достаточной длины из одних цифр и получим сообщение «НЕУДАЧНЫЙ ПАРОЛЬ: не содержит достаточное число РАЗЛИЧНЫХ символов».
- Зададим пароль достаточной длины, содержащий все указанные требования, кроме включения в него отличных от алфавита и цифр символов. Например, `2QyfMOb4`. Получим сообщение «НЕУДАЧНЫЙ ПАРОЛЬ: слишком простой».

```
$ passwd
Изменяется пароль пользователя user2.
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Смена пароля для user2.
(текущий) пароль Unix:
Новый пароль:
НЕУДАЧНЫЙ ПАРОЛЬ: не содержит достаточное число РАЗЛИЧНЫХ символов
Новый пароль:
НЕУДАЧНЫЙ ПАРОЛЬ: короткий
Новый пароль:
НЕУДАЧНЫЙ ПАРОЛЬ: слишком простой
passwd: Использовано максимальное число попыток, заданное для
 ↪службы
```

- Зададим пароль, соблюдая все установленные требования. Например, `2QyfM*Ob4`. Пароль будет успешно задан (см. листинг).

```
$ passwd
Изменяется пароль пользователя user2.
Смена пароля для user2.
(текущий) пароль Unix:
Новый пароль:
Повторите ввод нового пароля:
passwd: Все данные аутентификации успешно обновлены.
```

### 4.14. Конфигурационный файл /etc/issue

Конфигурационный файл `/etc/issue` позволяет задать текстовое содержание уведомления пользователю перед началом его идентификации и аутентификации для входа в систему. Например, с предупреждением о том, что в ней реализованы меры защиты информации и о необходимости соблюдения соответствующих правил обработки данных. Традиционно в конфигурационном файле присутствуют опции выдачи сведений об операционной системе и ядре. Дополнительно можно добавить опции выдачи текущих даты и времени, количества работающих пользователей и некоторых других сведений.

### 4.15. Конфигурационный файл /etc/shadow

Конфигурационный файл `/etc/shadow` содержит сведения об учетных записях и паролях пользователей в виде строк со следующей структурой:

```
username:id:salt:hashed:lastchanged:min:max:warn:inactive:expire
```

Структура файла:

- `username` — имя пользователя:
- `id` — алгоритм шифрования: 1 (алгоритм MD5), 5 (SHA-256), 6 (SHA-512);
- `salt` — «соль», добавляемая к паролю строка из 10-20 случайных символов;
- `hashed` — зашифрованный пароль;
- `lastchanged` — дата последнего изменения пароля;
- `min` — минимальное число дней между двумя последовательными сменами паролей;
- `max` — срок действия пароля, т.е. максимальное число дней, в течение которых пароль будет активен;
- `warn` — за какое количество дней до срока истечения действия пароля пользователь будет уведомлен о том, что его необходимо сменить;
- `inactive` — количество дней после истечения срока действия пароля, спустя которое его учётная запись блокируется;
- `expire` — число дней, прошедших с момента блокирования учётной записи.

Если после имени пользователя `username` вместо `id:salt:hashed` стоит символ `*` либо последовательность из двух символов `!!`, то это означает, что попытки входа в систему от имени данного пользователя заблокированы.

# 5. УПРАВЛЕНИЕ ДОСТУПОМ

## 5.1. Введение

Средства управления доступом предоставляют возможности ограничения количества одновременно предоставляемых параллельных сеансов доступа пользователей к системе, блокирования сеанса доступа пользователя в систему после истечения установленного периода времени бездействия или по его запросу, поддержки и сохранения атрибутов безопасности, связанных с информацией в процессе её хранения и обработки, разделения полномочий пользователей и администраторов, обеспечивающих функционирование системы, реализации различных методов управления доступом, типов доступа и правил разграничения доступа, назначения приоритетов для использования субъектами доступа вычислительных ресурсов, квотирования предоставляемых вычислительных ресурсов, а также другие возможности.

## 5.2. Установка и изменение прав доступа к файлам и директориям

Утилита `chmod` позволяет устанавливать и изменять права доступа к файлам и директориям. Она принимает описания прав доступа в двух нотациях: численной и буквенной, описываемой ниже.

В соответствии с буквенной нотацией пользователи, которые могут потенциально работать с файлом, разделяются на владельца (`u`), группу владельцев (`g`) и всех остальных пользователей (`o`), а файл может быть читаемым (`r`), записываемым (`w`) и исполняемым (`x`).

Описание прав доступа начинается с символа, соответствующего типу пользователей. Затем идет символ `+` для установки или символ `-` для снятия прав доступа, после чего описание заканчивается последовательностью символов, соответствующей правам доступа.

Например, для определения прав доступа, позволяющих читать и модифицировать файл `file`, может использоваться следующая команда:

```
$ chmod g+rw file
```

Для удаления всех прав доступа на директорию `/directory` для группы и остальных пользователей может использоваться следующая команда:

```
$ sudo chmod go-rwx /directory.
```

Утилита поддерживает опции, перечисленные в таблице.

Таблица 10 - Опции утилиты chmod и их значения

| Опция | Значение |
|---|---|
| -R, --recursive | Рекурсивное изменение прав доступа для директорий и их содержимого. |
| -c, --changes | Подробно описывать действия для каждого файла, чьи права действительно изменяются. |
| -f, --silent, --quiet | Не выдавать сообщения об ошибке для файлов, чьи права не могут быть изменены. |
| -v, --verbose | Подробно описывать действие или отсутствие действия для каждого файла. |
| --version | Сообщить информацию о версии. |
| --help | Выводит справку по этой утилите и завершает работу. |

**Пример**: сменим права для файла file1 так, чтобы владелец файла имел права на чтение и запись, а группа и остальные пользователи — только на чтение:

```
$ chmod u+rw g-wx o-wx file1
```

### 5.3. Назначение и изменение владельца файла и директории

Утилита chown позволяет назначить или изменить владельца файла или директории.

Утилита поддерживает следующие опции, перечисленные в таблице.

Таблица 11 - Опции утилиты chown и их значения

| Опция | Значение |
|---|---|
| -R, --recursive | Рекурсивное изменение прав доступа для директорий и их содержимого. |
| -c, --changes | Подробно описывать все изменения. |
| -f, --silent, --quiet | Не выдавать сообщения об ошибке. |

| Опция | Значение |
|---|---|
| `-v, --verbose` | Вывести подробное описание действия. |
| `--version` | Сообщить информацию о версии. |
| `--help` | Выводит справку по этой утилите и завершает работу. |

**Пример**: назначим пользователя `user` владельцем файла `file`:

```
$ sudo chown user file
```

**Пример**: выполним рекурсивный обход директории `directory` и назначим пользователя `user` владельцем всех вложенных файлов:

```
$ sudo chown -R user directory
```

### 5.4. Изменение группы-владельца файла или директории

Утилита `chgrp` позволяет изменить группу-владельца файла или директории.

Утилита поддерживает следующие опции, перечисленные в таблице.

Таблица 12 - Опции утилиты `chgrp` и их значения

| Опция | Значение |
|---|---|
| `-R, --recursive` | Рекурсивное изменение группы для каталогов и всего их содержимого. |
| `-c, --changes` | Подробно описывать действия для каждого файла, чья группа действительно меняется. |
| `-f, --silent, --quiet` | Не выдавать сообщения об ошибке для файлов, чья группа не может быть изменена. |
| `-v, --verbose` | Подробно описывать действие или отсутствие действия для каждого файла. |
| `--version` | Сообщить информацию о версии. |
| `--help` | Вывести справку по этой утилите и завершить работу. |

**Пример**: изменим группу-владельца файла `file` на новую группу `new_group`:

```
$ sudo chgrp new_group file
```

## 5.5. Просмотр и изменение списков правил контроля доступа для файлов и директорий

Утилита `setfacl` позволяет просматривать и изменять списки правил контроля доступа для файлов и директорий.

Утилита поддерживает следующие опции, перечисленные в таблице.

Таблица 13 - Опции утилиты `setfacl` и их значения

| Опция | Значение |
|---|---|
| `-d` | Установить правила контроля доступа по умолчанию. |
| `-k` | Удалить правила контроля доступа по умолчанию. |
| `-s` | Заменить правила контроля доступа заданными. |
| `-m` | Модифицировать правила контроля доступа. |
| `-x` | Удалить указанное правило контроля доступа. |
| `-b` | Удалить все правила контроля доступа. |
| `-v` | Вывести версию и выйти. |
| `-h` | Вывести справку об использовании утилиты и выйти. |

**Пример**: удалим все правила контроля доступа к файлу `file`:

```
$ sudo setfacl -b file
```

## 5.6. Просмотр списков контроля доступа

Утилита `getfacl` позволяет просматривать списки контроля доступа.

Утилита поддерживает следующие опции, перечисленные в таблице.

Таблица 14 - Опции утилиты `getfacl` и их значения

| Опция | Значение |
|---|---|
| `-a, --access` | Выводить список контроля доступа к файлам. |
| `-d, --default` | Выводить список контроля доступа по умолчанию. |
| `-c, --omit-header` | Не выводить заголовок с комментариями. |
| `e, --all-effective` | Выводить комментарии с действующими правами доступа для каждого пользователя. |
| `-E, --no-effective` | Не выводить комментарии с действующими правами доступа ни для одного пользователя. |
| `-R, --recursive` | Делать рекурсивный обход директории и выводить списки контроля доступа для каждого файла и директории. |
| `-v, --version` | Вывести версию и выйти. |
| `-h, --help` | Вывести справку об использовании утилиты и выйти. |

**Пример**: просмотрим список контроля доступа для файла `cg.conf`:

```
$ getfacl cg.conf
# file: cg.conf
# owner: user
# group: user
user::rwx
group::r-x
other::r-x
```

**Пример**: зададим дополнительные компоненты списка контроля доступа для пользователя `user` и группы `user` по отношению к файлу `cg.conf`:

```
$ setfacl -m g:user:rxw cg.conf

$ setfacl -m u:user:rxw cg.conf

$ getfacl cg.conf
# file: cg.conf
# owner: user
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# group: user
user::rwx
user:user:rwx
group::r-x
group:user:rwx
mask::rwx
other::r-x
```

**Пример**: от имени администратора модифицируем списки контроля доступа для файлов, владельцем которых он является:

```
$ sudo setfacl -m u:user:rxw ~/file2

$ sudo getfacl ~/file2
getfacl: Removing leading `/` from absolute path names
# file: root/file2
# owner: root
# group: root
user::rw-
user:user:rwx
group::r--
mask::rwx
other::r--
```

**Пример**: от имени администратора модифицируем списки контроля доступа для файлов, владельцем которых он не является:

```
$ sudo setfacl -m u:user:rxw /home/user3/file2

$ sudo setfacl -m u:user:rxw /home/user3/dir2

$ sudo getfacl /home/user3/file2
getfacl: Removing leading `/` from absolute path names
# file: home/user3/file2
# owner: user3
# group: user3
user::rwx
user:user:rwx
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
group::---
mask::rwx
other::---

$ sudo getfacl /home/user3/dir2
getfacl: Removing leading `/` from absolute path names
# file: home/user3/dir2/
# owner: user3
# group: user3
user::rwx
user:user:rwx
group::---
mask::rwx
other::---
```

**Пример**: от имени администратора удалим списки контроля доступа для объектов, владельцем которых он является:

```
$ sudo setfacl -b ~/file2

$ sudo getfacl ~/file2
getfacl: Removing leading `/` from absolute path names
# file: root/file2
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

**Пример**: от имени администратора удалим списки контроля доступа для объектов, владельцем которых он не является:

```
$ sudo setfacl -b /home/user3/file2

$ sudo getfacl /home/user3/file2
getfacl: Removing leading `/` from absolute path names
# file: home/user3/file2
# owner: user3
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# group: user3
user::rwx
group::---
other::---
```

**Важно**

Пользователь, не обладающий полномочиями администратора, не может удалять списки контроля доступа, которые он не создавал.

## 5.7. Редактирование пользовательских квот для файловой системы

Утилита `edquota` позволяет редактировать пользовательские квоты для файловой системы.

Утилита поддерживает следующие опции, перечисленные в таблице.

Таблица 15 - Опции утилиты `edquota` и их значения

| Опция | Значение |
|---|---|
| `-u, --user` | Изменить пользовательскую квоту. |
| `-g, --group` | Изменить групповую квоту. |
| `-p, --prototype = protoname` | Дублировать квоты прототипного пользователя. Это обычный механизм, используемый для инициализации квот для групп пользователей. |
| `-F, --format = имя-формата` | Изменить квоту для указанного формата. |
| `-f, --filesystem` | Выполнять указанные операции только для заданной файловой системы. По умолчанию операция выполняется для всех файловых систем с квотой. |
| `-t, --edit-period` | Редактировать мягкие ограничения по времени для каждой файловой системы. |
| `-T, --edit-times` | Изменить время для пользователя или группы, когда принудительное ограничение установлено. |

### 5.8. Конфигурационный файл /etc/profile

Конфигурационный файл `/etc/profile` используется для задания элементов окружения оболочки пользователя. Например, в нём определяются глобальные переменные:

- `PATH` — переменная среды, используемая для указания оболочке списка каталогов, которые будут просматриваться при поиске исполняемых файлов;
- `USER` — имя пользователя при входе в ОС;
- `LOGNAME` — то же, что и `USER`. Некоторые программы считывают значение этой глобальной переменной вместо `USER`;
- `MAIL` — имя файла, в который записывается локальная почта пользователя, а также его расположение;
- `HOSTNAME` — имя хоста;
- `HISTSIZE` — количество исполненных команд, сохраняемых в истории;
- `HISTCONTROL` — политики в отношении команд, сохраняемых в истории. По умолчанию задано значение `ignoredups`, то есть команда, полностью совпадающая с одной из уже записанных в историю, не сохраняется. Если задать политику `ignorespace`, то будут игнорироваться как дублирующиеся команды, так и те, что начинаются с символа пробела.

Также в конфигурационном файле задаётся маска, используемая для определения конечных прав доступа для пользователя.

**Пример**: определим время бездействия при локальной терминальной сессии равным двум минутам (120 с). Для этого в файле `/etc/profile` после строк

```
HOSTNAME= '/usr/bin/hostname 2>/dev/null'
HISTSIZE=1000
```

Добавим строку `TMOUT=120`. Там же, в строке

```
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE HISTCONTROL
```

Необходимо добавить параметр `TMOUT`:

```
export PATH USER LOGNAME MAIL HOSTNAME HISTSIZE TMOUT HISTCONTROL
```

Для подтверждения вступления изменений в силу надо будет завершить сеанс и зарегистрироваться в системе заново. Тогда появится сообщение, что

после двух минут бездействия время ожидания ввода вышло, в результате чего интерактивный сеанс был закрыт.

```
# /etc/profile

# System wide environment and startup programs, for login setup
# Functions and aliases go in /etc/bashrc

# It's NOT a good idea to change this file unless you know what you
# are doing. It's much better to create a custom.sh shell script in
# /etc/profile.d/ to make custom changes to your environment, as this
# will prevent the need for merging in future updates.

pathmunge () {
    case ":${PATH}:" in
        *:"$1":*)
            ;;
        *)
            if [ "$2" = "after" ] ; then
                PATH=$PATH:$1
            else
                PATH=$1:$PATH
            fi
    esac
}


if [ -x /usr/bin/id ]; then
    if [ -z "$EUID" ]; then
        # ksh workaround
        EUID=`/usr/bin/id -u`
        UID=`/usr/bin/id -ru`
    fi
    USER="`/usr/bin/id -un`"
    LOGNAME=$USER
    MAIL="/var/spool/mail/$USER"
fi

# Path manipulation
if [ "$EUID" = "0" ]; then
    pathmunge /usr/sbin
    pathmunge /usr/local/sbin
else
    pathmunge /usr/local/sbin after
    pathmunge /usr/sbin after
fi
```

Рис. 19: Конфигурационный файл /etc/profile

### 5.9. Конфигурационный файл /etc/security/limits.conf

Конфигурационный файл `/etc/security/limits.conf` может использоваться для задания модулю `pam_limits.so` дополнительных ограничений. Для этого каждая его строка включает четыре группы параметров, которые перечислены и описаны ниже:

**Важно**

По умолчанию все ограничения отключены — все строки закомментированы.

**-<domain>:**

имя пользователя, имя группы с синтаксисом `@group`, подстановочный знак `*` для записи по умолчанию, подстановочный знак `%`, который также может использоваться с синтаксисом `%group` для ограничения `maxlogin`;

**-<type>:**

- `soft` для установки мягких ограничений;
- `hard` для установки жестких ограничений.

**-<item>:**

- `core`: ограничивает размер файла ядра в Кб;
- `data`: максимальный размер данных в Кб;
- `fsize`: максимальный размер файла в Кб;
- `memlock`: максимальное адресное пространство, предустановленное в памяти, в Кб;
- `nofile`: максимальное количество открытых файлов;
- `rss`: максимальный размер резидентного набора в Кб;
- `stack`: максимальный размер стека в Кб;
- `cpu`: максимальное время процессора в MIN;
- `nproc`: максимальное количество процессов;
- `as`: ограничение адресного пространства в Кб;
- `maxlogins`: максимальное количество логинов для этого пользователя;
- `maxsyslogins`: максимальное количество входов в систему;

- `priority`: приоритет процессов пользователя;
- `locks`: максимальное количество блокировок файлов, которое может быть обеспечено пользователем;
- `sigpending`: максимальное количество ожидающих сигналов;
- `msgqueue`: максимальный объем памяти, используемый очередями сообщений POSIX, в байтах;
- `nice`: приоритет для запуска процессов утилитой `nice`;
- `rtprio`: максимальный приоритет в реальном времени.

```
# /etc/security/limits.conf
#
#This file sets the resource limits for the users logged in via PAM.
#It does not affect resource limits of the system services.
#
#Also note that configuration files in /etc/security/limits.d directory,
#which are read in alphabetical order, override the settings in this
#file in case the domain is the same or more specific.
#That means, for example, that setting a limit for wildcard domain here
#can be overridden with a wildcard setting in a config file in the
#subdirectory, but a user specific setting here can be overridden only
#with a user specific setting in the subdirectory.
#
#Each line describes a limit for a user in the form:
#
#<domain>        <type>  <item>  <value>
#
#Where:
#<domain> can be:
#        - a user name
#        - a group name, with @group syntax
#        - the wildcard *, for default entry
#        - the wildcard %, can be also used with %group syntax,
#                 for maxlogin limit
#
#<type> can have the two values:
#        - "soft" for enforcing the soft limits
#        - "hard" for enforcing hard limits
#
#<item> can be one of the following:
#        - core - limits the core file size (KB)
#        - data - max data size (KB)
#        - fsize - maximum filesize (KB)
#        - memlock - max locked-in-memory address space (KB)
#        - nofile - max number of open file descriptors
```

Рис. 20: Конфигурационный файл /etc/security/limits.conf

**Пример**: ограничим число параллельных сеансов доступа для каждой учетной записи пользователя. Для этого добавим в конфигурационный файл строку следующего содержания:

```
username hard maxlogins 2
```

Тогда, при условии, что пользователь `username` открыл локальную сессию (учитывая, что при входе в графический сеанс открываются сразу две сессии пользователя) и попытался зайти в систему через ssh-соединение (потенциально ещё один активный сеанс), ему будет выведено сообщение `Too many logins for 'username'` и это соединение будет заблокировано.

### 5.10. Конфигурационный файл /etc/fstab

Конфигурационный файл `/etc/fstab` используется для настройки параметров монтирования различных блочных устройств, разделов на диске и файловых систем. Он состоит из набора так называемых определений, каждое из которых занимает свою строку и состоит из шести полей, разделённых пробелами или символами табуляции:

```
fs_spec fs_file fs_vfstype fs_mntops fs_freq fs_passno
```

Поля предназначены для задания следующих параметров:

**-fs_spec**

Физическое размещение файловой системы, по которому определяется конкретный раздел или устройство хранения для монтирования. Вместо указания размещения файловой системы явным образом можно воспользоваться её уникальным идентификатором UUID.

**-fs_file**

Точка монтирования, куда монтируется корень файловой системы.

**-fs_vfstype**

Тип файловой системы. Поддерживаются следующие типы: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext2, ext3, ext4, hfs, hpfs, iso9660, jfs, minix, msdos, nvpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs.

**-fs_mntops**

Опции монтирования файловой системы. Основные опции: `defaults`, `noauto`, `user`, `owner`, `comment`, `nofail`.

**`-fs_freq`**

Предназначено для использования утилитой создания резервных копий в файловой системе. Возможные значения: `0` и `1`. Если указано `1`, то утилита создаст резервную копию.

**`-fs_passno`**

Предназначено для использования программой `fsck` при необходимости проверки целостности файловой системы; возможные значения: `0`, `1` и `2`. Значение `1` указывается только для корневой файловой системы (то есть файловой системы с точкой монтирования `/`). Для остальных файловых систем для проверки утилитой `fsck` задаётся значение `2`. При значении `0` — проверка выполняться не будет.

По умолчанию конфигурационный файл включает:

```
/dev/mapper/MSVSphere-root / xfs defaults 0 0
```

Файловая система `/dev/mapper/MSVSphere-root` примонтирована в каталог `/`, тип файловой системы — `xfs`, используемые опции — `defaults`, резервная копия данных не создаётся (`fs_freq=0`), проверка целостности файловой системы не выполняется (`fs_passno=0`).

```
UUID=b1bfe9b0-96ea-4876-883c-a9f1b6c74b /boot ext4 defaults 1 2
```

Файловая система с идентификатором `b1bfe9b0-96ea-4876-883c-a9f1b6c74b` смонтирована в `/boot`, тип файловой системы — `ext4`, используемые опции — `defaults`, резервная копия данных создаётся (`fs_freq=1`), проверка целостности файловой системы выполняется (`fs_passno=2`).

```
/dev/mapper/MSVSphere-swap swap defaults 0 0
```

Файловая система `/dev/mapper/MSVSphere-swap` является разделом подкачки `swap`, используемые опции — `defaults`, резервная копия данных не создаётся (`fs_freq=0`), проверка целостности файловой системы не выполняется (`fs_passno=0`).

```
# /etc/fstab
# Created by anaconda on Tue Jun 20 11:58:05 2023
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/msvsphere-root /                       xfs     defaults        0 0
UUID=8e41c721-164e-455c-bd65-b60ad5ad7cb4 /boot                   xfs     defaults
```

Рис. 21: Конфигурационный файл /etc/fstab

# 6. РЕГИСТРАЦИЯ СОБЫТИЙ БЕЗОПАСНОСТИ

## 6.1. Введение

В операционную систему МСВСфера 9 встроена подсистема аудита, основной задачей которой является регистрация и обработка событий, связанных с нарушением безопасности. Фильтрация сообщений происходит на основе предварительно настроенных правил, отфильтрованные события регистрируются в системном журнале событий безопасности. Системному администратору доступны различные инструменты, упрощающие анализ зарегистрированных событий.

Ниже приведены некоторые сценарии использования подсистемы аудита.

- **Регистрация событий входа в систему**

  Позволяет отслеживать как успешные, так и неудачные попытки входа в систему с использованием различных механизмов аутентификации, таких как локальная база пользователей или LDAP-каталог, SSH, Kerberos и т.п..
- **Отслеживание доступа к файлам**

  Позволяет отслеживать, осуществлялся ли доступ к тому или иному файлу или каталогу, их модификация, изменение атрибутов или запуск.
- **Мониторинг системных вызовов**

  С помощью подсистемы аудита можно отслеживать использование отдельных системных вызовов (`man 2 syscalls`), например, операции монтирования файловых систем, изменение системного времени, открытие сетевого соединения и т.п..
- **Отслеживание событий безопасности средства виртуализации**

  Позволяет отслеживать различные действия пользователей с ресурсами, управляемыми гипервизором `libvirt`.
- **Регистрация событий установки или удаления ПО**

  С помощью подсистемы аудита можно отслеживать установку, удаление или обновление пакетов с использованием различных пакетных менеджеров: dnf/yum, pip, npm, cpan, gem и т.д..

Рассмотрим основные компоненты и архитектуру подсистемы аудита подробнее.

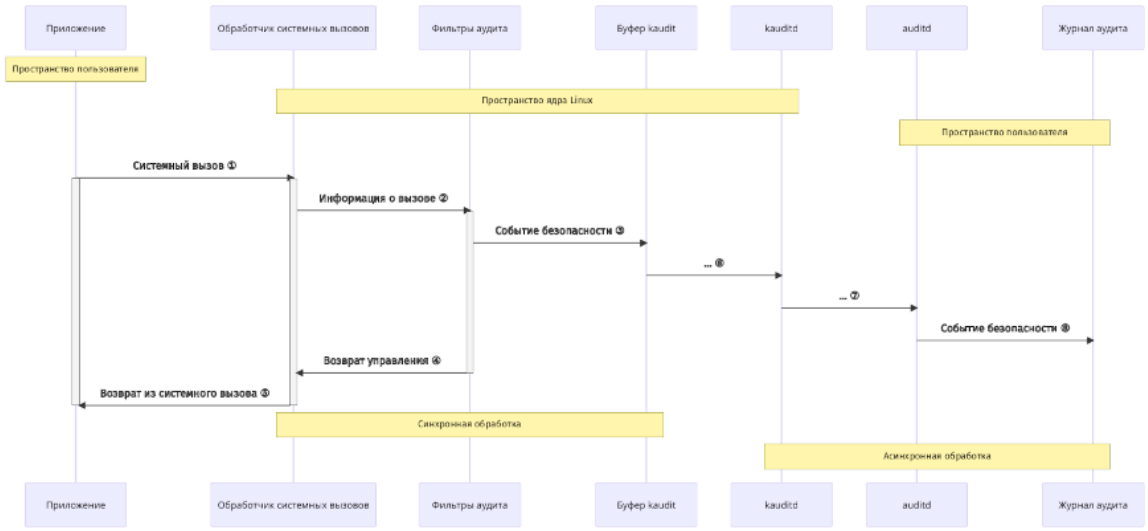



Рис. 22: Основные компоненты и архитектура подсистемы аудита

1. Когда программа выполняет системный вызов, информация об этом попадает в специальный обработчик в ядре Linux — на этом этапе формируется событие безопасности, которое содержит информацию о выполняемом системном вызове, вызвавшем его процессе, идентификатор пользователя и т.д..
2. Из обработчика событие направляется на фильтрацию в соответствующий блок подсистемы аудита, где к этому событию применяются заранее загруженные фильтры.
3. В случае срабатывания одного из фильтров событие направляется в буфер событий компонента `kauditd`, ответственного за регистрацию на стороне ядра.
4. После записи события в буфер `kauditd` осуществляется возврат управления в обработчик системного вызова.
5. Из обработчика системного вызова осуществляется возврат управления в программу.
6. Из буфера событие попадает на обработку в компонент `kauditd`.
7. Компонент `kauditd` передаёт информацию о событии из пространства ядра в сервис `auditd`, работающий в пользовательском пространстве, с помощью стандартного механизма коммуникации `netlink` (`man 7 netlink`).
8. Сервис `auditd` регистрирует событие безопасности в файле системного журнала (по умолчанию в `/var/log/audit/audit.log`).

После этого регистрация события безопасности считается выполненной. Далее возможны следующие варианты обработки события безопасности в пользовательском пространстве.

- Сервис `auditd` может передать информацию о событии безопасности на обработку в другие сервисы/программы с помощью расширений (плагинов). Получателями такой информации могут быть системы централизованного сбора и анализа системных журналов, системы мониторинга, накладные средства защиты информации (СЗИ) и т.п..
- Системный администратор может выполнять анализ и обработку событий безопасности из системного журнала с помощью встроенных средств, таких как `ausearch`, `aureport`, `aulastlog` и других инструментов.

## 6.2. Настройка сервиса auditd

### 6.2.1. Конфигурационный файл /etc/audit/auditd.conf

Настройка сервиса `auditd` осуществляется через конфигурационный файл `/etc/audit/auditd.conf`, использующий стандартный для Unix-подобных систем формат `КЛЮЧ = ЗНАЧЕНИЕ`. Все ключи и значения являются регистронезависимыми.

В таблице ниже приведено описание допустимых параметров и их значения по умолчанию.

Таблица 16 - Параметры сервиса `auditd` и их значения по умолчанию

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| `local_events` | `yes` | Включает (`yes`) или выключает (`no`) регистрацию событий безопасности локальной системы. Если необходимо только регистрировать события, полученные по сети, установите в значение `no`. Обычно это используется при развёртывании `auditd` в контейнере для централизованного сбора информации с нескольких систем. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| log_file | /var/log/audit/audit.log | Полный путь к файлу, в который необходимо записывать журнал событий безопасности. |
| write_logs | yes | Включает (yes) или выключает (no) запись журналов безопасности на диск. |
| log_format | ENRICHED | Определяет, в каком формате будет сохраняться информация в журнал. Допустимые значения параметра перечислены после таблицы. |
| log_group | root | Системная группа, на которую распространяются права на файлы журнала событий безопасности. Допускается использование как идентификатора группы, так и её имени. |
| priority_boost | 4 | Неотрицательное число, определяющее приоритет выполнения службы аудита. Чтобы оставить приоритет без изменений, используйте значение 0. |
| krb5_principal | auditd | Имя Kerberos-принципала сервера auditd. При использовании значения по умолчанию сервер auditd будет искать ключ с именем auditd/HOSTNAME@REALM в файле, заданном директивой krb5_key_file, где HOSTNAME — это каноническое имя сервера, возвращаемое службой DNS по его IP-адресу, а REALM — область (realm) Kerberos. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| freq | 50 | Неотрицательное число, которое определяет сколько событий необходимо записать прежде чем выполнить принудительную синхронизацию данных на диск. Этот параметр применяется только если значение flush установлено в INCRE-MENTAL или INCREMENTAL_ASYNC. |
| name_format | NONE | Определяет как имена компьютеров вставляются в поток событий безопасности. Допустимые значения параметра перечислены после таблицы. |
| name | Не задано | Строка, определяемая системным администратором, для использования в качестве имени системы если параметр name_format установлен в значение USER. |
| max_log_file | 8 | Максимальный размер файла журналов в мегабайтах. При достижении этого лимита будет запускаться действие, настраиваемое с помощью опции max_log_file_action. |
| max_log_file_ac-tion | ROTATE | Указывает какое действие необходимо предпринять когда система обнаружит, что достигнут лимит на максимальный размер файла журнала. Допустимые значения параметра перечислены после таблицы. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| verify_email | yes | Если установлен в yes, то для доменного имени, указанного в почтовом адресе в директиве action_mail_acct, будет выполнена проверка на наличие соответствующей DNS-записи. Этот параметр должен быть расположен перед опцией action_mail_acct в конфигурационном файле, иначе будет использовано значение по умолчанию yes. |
| flush | INCREMENTAL_ASYNC | Определяет стратегию работы с дисковым буфером. Допустимые значения параметра перечислены после таблицы. |
| q_depth | 2000 | Определяет максимальный размер внутренней очереди диспетчера событий auditd. Очередь большего размера позволяет службе лучше справляться с большим потоком событий, но при слишком большой очереди некоторые события могут не успеть быть обработанными при завершении работы сервиса. Если в системном журнале появляются сообщения о том, что некоторые события были удалены, то увеличьте это значение. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| num_logs | 5 | Определяет максимальное количество файлов журналов, которые необходимо сохранять, если параметр max_log_file_action установлен в значение ROTATE. Если значение num_logs меньше 2, то ротация файлов журналов не будет производиться. В качестве значения допустимо любое число в диапазоне от 0 до 999. С увеличением количества сохраняемых файлов может потребоваться поднять лимит на максимальное количество ожидающих запросов к сервису auditd — за это отвечает опция -b в конфигурационном файле /etc/audit/audit.rules. Если настроена ротация журналов, то auditd будет следить за количеством файлов журналов и удалять лишние файлы. Проверка избыточных журналов выполняется только при запуске сервиса и при проверке изменения конфигурации сервиса. |
| space_left_action | SYSLOG | Определяет какое действие необходимо предпринять если на файловой системе начинает заканчиваться свободное пространство. Допустимые значения параметра перечислены после таблицы. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| action_mail_acct | root | Адрес электронной почты или псевдоним (см. /etc/aliases), на который будут отправляться сообщения от auditd. Если адрес не является локальным для данной системы, то необходимо чтобы на ней была настроена почтовая система для отправки, в том числе, требуется наличие программы /usr/lib/sendmail, предоставляемой postfix, exim, sendmail и другими почтовыми системами. |
| space_left | 75 | Если объём свободного места в файловой системе, содержащей log_file, становится меньше указанного значения, то сервис auditd выполнит действие, определённое директивой space_left_action. Если значение указано как целое число, то оно интерпретируется как абсолютный размер в мегабайтах. Если значение указано в виде числа от 1 до 99 со знаком процента (например, 5%), то auditd самостоятельно вычислит соответствующее значение как указанный процент от общего размера файловой системы. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| use_libwrap | yes | Определяет, следует ли использовать механизм tcp_wrappers для фильтрации подключений от других компьютеров. Допустимые значения: yes или no. |
| admin_space_left | 50 | Если объём свободного места в файловой системе, содержащей log_file, становится меньше указанного значения, то сервис auditd выполнит действие, определённое директивой admin_space_left_action. Как и для директивы space_left, значение указывается либо в виде целого числа (абсолютный размер в мегабайтах), либо в процентах от 1% до 99%. В любом случае, значение admin_space_left должно быть меньше space_left — это следует рассматривать как последний шанс что-то предпринять прежде чем дисковое пространство будет полностью заполнено. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| admin_space-_left_action | SUSPEND | Определяет какое действие необходимо предпринять если на файловой системе почти закончилось место. Список допустимых значений и их поведение совпадает с директивой space_left_action: IGNORE, SYSLOG, ROTATE, EMAIL, EXEC /путь-к-программе, SUSPEND, SINGLE и HALT. |
| max_restarts | 10 | Неотрицательное число, которое определяет сколько раз служба auditd может попытаться перезапустить вышедшее из строя расширение (плагин). |
| disk_full_action | SUSPEND | Определяет какое действие необходимо предпринять если на файловой системе закончилось место. Допустимые значения: IGNORE, SYSLOG, ROTATE, EXEC /путь-к-программе, SUSPEND, SINGLE, HALT. Поведение для каждого из значений рассмотрено в описании к директиве space_left_action. |
| disk_error_ac-tion | SUSPEND | Определяет какое действие необходимо предпринять в случае возникновения ошибки при сохранении событий безопасности на диск или при ротации файлов журнала. Допустимые значения параметра перечислены после таблицы. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| `tcp_listen_port` | Не задано | Числовое значение от `1` до `65535`, при указании которого сервис `auditd` будет принимать записи о событиях безопасности от удалённых систем на соответствующем TCP-порту. Для работы с удалённым сервером или клиентами рекомендуется настроить сервис `auditd`, чтобы он запускался после активации сетевых интерфейсов, соответствующая инструкция приведена в файле `/usr/lib/systemd/system/auditd.service`. |
| `tcp_listen_queue` | `5` | Определяет максимально разрешённое количество ожидающих (запрошенных, но ещё не принятых) сетевых подключений к сервису `auditd`. Слишком маленькое значение может привести к тому, что некоторые подключения будут отклонены в случае если множество клиентов будет подключаться одновременно, допустим, после сбоя питания. Эта опция используется только агрегирующими серверами `auditd`, которые обрабатывают события от удалённых систем. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| `end_of-_event_timeout` | `2` | Неотрицательное количество секунд, после которого событие считается завершённым при анализе потока журнала событий пользовательскими утилитами и библиотечными функциями, такими как `aureport` (`man aureport`), `ausearch` (`man ausearch`) и т.д. Если в процессе обработки событий время текущего события превышает `end_of_event_timeout` относительно соседних событий в потоке, то такое событие будет считаться завершённым. |
| `tcp_max_per_addr` | `1` | Числовое значение от `1` до `1024`, которое определяет максимально разрешённое количество одновременных подключений с одного IP-адреса. Установка слишком большого значения может привести к DDoS-атаке на сервер `auditd`. Следует иметь ввиду, что в ядре Linux есть свои собственные лимиты, которые могут ограничить количество подключений даже если настройка сервиса `auditd` позволяет использовать больше соединений. Значение по умолчанию `1` является достаточным для большинства случаев, если только вы не реализовываете самостоятельно какую-то дополнительную надстройку для пересылки ранее неотправленных сообщений. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| tcp_client_ports | Не задано | Определяет с какого исходящего TCP-порта или портов разрешены входящие подключения к сервису auditd. Допустимый диапазон портов: от 1 до 65535. Единичный порт задаётся одним числовым значением, а диапазон — двумя значениями, разделёнными символом «-». Например, чтобы потребовать от клиента использовать привилегированный порт, можно задать значение 1-1023 — это может рассматриваться как дополнительная мера защиты, позволяющая исключить атаки типа «инъекция логов» от имени непривилегированных пользователей. В случае использования этой опции также потребуется установить соответствующее значение директиве local_port в конфигурационном файле /etc/audit/audisp-remote.conf (см. man audisp-remote.conf). В конфигурации по умолчанию никаких ограничений по исходящим портам не применяется. |
| plugin_dir | /etc/audit/plugins.d | Задаёт каталог, в котором auditd будет осуществлять поиск конфигурационных файлов своих расширений (плагинов). |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| tcp_client-_max_idle | 0 | Задаёт время в секундах в течении которого допускается отсутствие каких-либо данных со стороны клиента. Этот параметр используется для закрытия неактивных подключений, если на клиентской системе возникла проблема и она не может самостоятельно завершить подключение. Это глобальная настройка и её значение должно быть выше чем значение heart-beat_timeout (man audisp-remote.conf) на клиентских машинах. Рекомендуется устанавливать значение tcp_client_max_idle в два раза большим чем heartbeat_timeout. Значение по умолчанию 0 отключает эту проверку. |
| transport | TCP | Если установлено в TCP, то данные между клиентом и сервером au-ditd будут передаваться в виде открытого текста без шифрования. Если установлено в KRB5, то протокол Kerberos 5 будет использоваться для аутентификации и шифрования. |
| krb5_key_file | /etc/audit/audit.key | Путь к файлу с ключом для Kerberos-принципала этого клиента. Этот файл должен принадлежать пользователю root и иметь права 0400. |

| Параметр | Значение по умолчанию | Описание |
|---|---|---|
| dis-tribute_network | no | Если установлено в yes, то события, поступающие из сети, будут переданы диспетчеру auditd для обработки расширениями (плагинами), что позволит реализовать их дальнейшую пересылку на другой сервер или в систему мониторинга/анализа событий. Если значение no, то события будут только сохраняться в журнал на диске. |
| overflow_action | SYSLOG | Определяет как служба auditd должна реагировать на переполнение внутренней очереди событий. Когда это происходит, это означает, что в очередь на регистрацию поступает больше событий, чем может быть обработано дочерними процессами auditd. Эта ошибка также означает, что текущее событие, поступившее на обработку, будет потеряно. Допустимые значения: IGNORE, SYSLOG, SUSPEND, SINGLE и HALT. Поведение для каждого из значений рассмотрено в описании к директиве space_left_action. |

**Допустимые значения параметров**

**log_format**:

- RAW — записи о событиях безопасности будут храниться в том формате, в котором их отправляет ядро операционной системы.
- ENRICHED — перед сохранением на диск записи о событиях безопасности будут приведены к более понятному человеку виду путём «разворачивания» информации об идентификаторе пользователя (uid),

идентификаторе группы (gid), системном вызове (syscall), архитектуре и адресе сокета. Это упростит анализ событий, созданных на одной системе, в другой системе.

**flush**:

- NONE — не выполнять какие-либо дополнительные действия по синхронизации буфера с диском со стороны службы регистрации событий.
- INCREMENTAL — выполнять принудительную синхронизацию буфера на диск с частотой, определяемой параметром freq.
- INCREMENTAL_ASYNC — поведение похоже на INCREMENTAL, но синхронизация буфера выполняется в асинхронном режиме для улучшения производительности.
- DATA — незамедлительно синхронизировать данные файла на диск.
- SYNC — незамедлительно сохранять данные и метаданные файла на диск.

**name_format**:

- NONE — имя компьютера не будет вставляться в записи журнала безопасности.
- HOSTNAME — в журнал будет добавляться имя компьютера, получаемое из системного вызова gethostname.
- FQD — система аудита получит имя компьютера и преобразует его в полное имя компьютера (FQDN) с помощью DNS-запроса.
- NUMERIC — поведение похоже на режим FQD, но по имени компьютера будет определяться IP адрес компьютера. Перед использованием этой опции рекомендуется проверить что команды hostname -i или domain-name -i возвращают корректный IP-адрес системы. Использование этой опции не рекомендуется, если для настройки сети используется DHCP так как существует вероятность смены IP-адреса компьютера.
- USER — будет использоваться имя, заданное системным администратором в параметре name.

**max_log_file_action**:

- IGNORE — не контролировать максимальный размер файла.
- SYSLOG — записать соответствующее предупреждение в системный журнал ОС.
- SUSPEND — прекратить записывать журнал аудита на диск, сервис auditd при этом продолжит свою работу.

- ROTATE — выполнить ротацию файла журнала событий безопасности, при необходимости удалить старые файлы журналов. Будет создан новый файл журнала, а текущий будет переименован — к его имени будет добавлен постфикс 1, постфикс более старых файлов журналов будет увеличен на единицу. Такого же поведения придерживается утилита logrotate.
- KEEP_LOGS — поведение аналогично опции ROTATE, но старые файлы журналов не будут удаляться, что предотвратит потерю данных из журнала событий безопасности. Когда на диске закончится место, будет выполненное действие, определённое директивой space_left_action. Этот вариант рекомендуется использовать совместно с системой резервного копирования.

**space_left_action**:

- IGNORE — ничего не предпринимать.
- SYSLOG — записать соответствующее предупреждение в системный журнал.
- ROTATE — осуществить ротацию файлов журналов, удалить самые старые из них чтобы освободить место.
- EMAIL — отправить соответствующее предупреждение на адрес электронной почты, указанный в директиве action_mail_acct и записать предупреждение в системный журнал.
- EXEC /путь-к-программе — приостановить запись событий в журнал и выполнить указанную программу, передача параметров программе не поддерживается. Вызванная программа должна освободить место и подать сигнал SIGUSR2 сервису auditd чтобы он возобновил запись событий. Самый простой способ это сделать — выполнить команду auditctl --signal USR2.
- SUSPEND — прекратить запись данных на диск, при этом сам сервис auditd будет активен.
- SINGLE — перевести компьютер в однопользовательский режим (также известный как режим восстановления), как если бы системный администратор выполнил команду telinit 1 или systemctl isolate runlevel1.target.
- HALT — выключить компьютер. Все действия кроме ROTATE будут выполняться только один раз.

**disk_error_action**:

- `IGNORE` — ничего не предпринимать.
- `SYSLOG` — записать не более пяти последовательных предупреждений в системный журнал.
- `EXEC /путь-к-программе` — приостановить запись событий в журнал и выполнить указанную программу, передача параметров программе не поддерживается. Вызванная программа должна освободить место и подать сигнал `SIGUSR2` сервису `auditd` чтобы он возобновил запись событий. Самый простой способ это сделать — выполнить команду `auditctl --signal USR2`.
- `SUSPEND` — прекратить запись данных на диск, при этом сам сервис `auditd` будет активен.
- `SINGLE` — перевести компьютер в однопользовательский режим (также известный как режим восстановления), как если бы системный администратор выполнил команду `telinit 1` или `systemctl isolate runlevel1.target`.
- `HALT` — выключить компьютер.

### 6.2.2. Применение изменений

После изменения настроек в конфигурационном файле потребуется применить их, выполнив следующую команду:

```
$ sudo auditctl --signal HUP
```

Служба `auditd` перечитает конфигурационный файл и, если не обнаружит синтаксических ошибок, попробует применить запрошенные изменения. В случае успешного завершения операции в журнале событий безопасности `/var/log/audit/audit.log` появится соответствующее сообщение типа `DAEMON_CONFIG`:

```
type=DAEMON_CONFIG msg=audit(1731317470.852:545): op=reconfigure
 ↪state=changed auid=1000 pid=15069 subj=unconfined_u:unconfined_
 ↪r:unconfined_t:s0-s0:c0.c1023 res=success AUID="user"
```

В случае возникновения ошибки, в зависимости от её типа, будет выполнено одно из действий, определённых директивами `space_left_action`, `admin_space_left_action`, `disk_full_action` или `disk_error_action` в

конфигурационном файле.

### 6.2.3. Рекомендации по безопасной настройке

В конфигурации, поставляемой по умолчанию в МСВСфера ОС, служба регистрации событий безопасности имеет сбалансированную с точки зрения производительности и безопасности настройку, которая должна подходить под большинство задач.

Ниже приведены некоторые рекомендации по повышению безопасности и отказоустойчивости службы аудита.

- На сервере рекомендуется настроить почтовую систему (MTA, Mail Transfer Agent) для отправки уведомлений от службы аудита системному администратору. В репозиториях МСВСфера ОС доступны следующие почтовые сервера: `postfix`, `sendmail` и `esmtp`. Если ваша система мониторинга использует протокол SNMP (Simple Network Management Protocol), то в качестве альтернативного способа доставки уведомлений можно реализовать соответствующие скрипты/утилиты для отправки предупреждений в систему мониторинга, а через неё, в свою очередь, отправлять предупреждения системному администратору. В состав пакета `net-snmp-utils` входит утилита `snmptrap`, которую можно использовать для отправки сигналов SNMP-trap. Через разработку собственных утилит можно реализовать отправку уведомлений с использованием других протоколов.
- Каталог, в котором хранятся журналы безопасности (см. описание параметра `log_file`), должен находиться на отдельном разделе/точке монтирования. Это позволит избежать ситуации, когда для журналов безопасности не осталось свободного места по причине того, что оно занято файлами других процессов. Также это позволит службе аудита точнее определять и контролировать оставшееся место.
- Параметрам `max_log_file` и `num_logs` должны быть присвоены такие значения, чтобы служба аудита могла использовать всё доступное место на файловой системе. Следует иметь в виду, что чем больше файлов необходимо ротировать, тем больше времени на это потребуется службе аудита прежде чем приступить к записи событий в новый файл журнала. Соответственно, не рекомендуется устанавливать слишком маленький

размер файла журнала.

- Параметру `max_log_file_action` рекомендуется присвоить значение `KEEP_LOGS`, чтобы старые файлы журналов безопасности не удалялись.
- Параметру `space_left` рекомендуется установить такое значение, которое оставит системному администратору достаточно времени чтобы среагировать на предупреждение и освободить дисковое пространство. Как правило, в таких случаях требуется выполнить процедуру архивирования файлов журналов.
- Для параметра `space_left_action` рекомендуется установить значение `EMAIL`, чтобы отправить соответствующее предупреждение на электронную почту системному администратору. В качестве альтернативы можно использовать значение `EXEC` и вызывать соответствующую утилиту для отправки уведомления с использованием другого протокола, допустим, `SNMP`.
- Параметру `admin_space_left` рекомендуется установить такое значение, при котором у службы аудита останется достаточно свободного пространства для протоколирования действий системного администратора.
- Значение параметра `admin_space_left_action` рекомендуется установить в `SINGLE`, чтобы перевести систему в однопользовательский режим и позволить администратору освободить место на диске.
- Параметру `disk_full_action` рекомендуется установить значение `HALT` для автоматического выключения системы, если на диске кончилось место — это позволит избежать незапротоколированных действий в системе. Если выключение системы неприемлемо, то установите значение `SINGLE`, чтобы перевести систему в однопользовательский режим.
- Параметр `disk_error_action` должен быть установлен в значение `SYSLOG`, `SINGLE` или `HALT`, в зависимости от политики безопасности вашего предприятия в отношении аппаратных сбоев.
- Если у вас отсутствует система резервного питания и потенциальная утеря нескольких (см. описание директивы `freq` в конфигурационном файле) последних записей о событиях безопасности является для вас критичной, то установите значение директивы `flush` в `SYNC`, что обеспечит постоянную синхронизацию данных и метаданных на диск. Однако,

следует отметить, что это приведёт к снижению производительности дисковой подсистемы.

В случае использования сервиса `auditd` для агрегации журналов с других машин по сети рекомендуется:

- Использовать протокол Kerberos для аутентификации и шифрования соединения между компьютерами, за это отвечают директивы `transport`, `krb5_principal` и `krb5_key_file` в конфигурационном файле.
- Разрешить отправку журналов безопасности только с использованием привилегированных портов — директива `tcp_client_ports` в конфигурационном файле. Это поможет избежать атак со стороны непривилегированных пользователей на клиентских машинах.

## 6.3. Управление правилами аудита

Правила фильтрации событий безопасности можно условно разделить по времени их жизни на временные и постоянные. Временные правила в основном используются для отладки, они добавляются в систему вручную, с помощью утилиты `auditctl`, и действуют до перезагрузки или выключения системы. Постоянные правила хранятся в виде файлов в каталоге `/etc/audit/rules.d` и автоматически применяются каждый раз при загрузке системы.

### 6.3.1. Утилита auditctl

Команда `auditctl` служит для управления базовыми функциями подсистемы аудита, а также позволяет задавать правила, определяющие какие события будут регистрироваться в журнале событий безопасности.

Как и для большинства команд в среде GNU/Linux, для вызова утилиты `auditctl` используется стандартный синтаксис:

```
auditctl [аргументы]
```

Аргументы, которые принимает команда, разбиты на три группы по их назначению.

- Конфигурационные опции — отвечают за настройку параметров ядра, связанных с подсистемой аудита.
- Опции состояния — отображают состояние подсистемы аудита, также существует опция для отправки сообщения в поток событий безопасности.

- Опции для управления правилами фильтрации событий безопасности.

Конфигурационные опции перечислены в таблице.

Таблица 17 - Конфигурационные опции `auditctl`

| Аргумент | Описание |
|---|---|
| `-b <количество>` | Устанавливает лимит на максимальное количество ожидающих обработки событий безопасности для подсистемы аудита в ядре. Если лимит будет превышен, то ядром будет выставлен флаг сбоя для дальнейшей обработки. Следует иметь в виду, что очередь необработанных сообщений хранится в оперативной памяти, соответственно, объём потребляемой памяти будет пропорционален количеству записей в очереди. Одна запись может занимать около 10 килобайт. По умолчанию размер буфера в ядре равен `64` записям, но правила, поставляемые с МСВСфера ОС, устанавливают его размер в `8192` записей. |
| `--backlog_wait-_time <время_ожидания>` | Задаёт максимальное время ожидания пока размер полностью заполненной очереди событий, ожидающих обработки, не уменьшится. В случае превышения этого лимита текущее событие, ожидающее обработки, будет утеряно (удалено). При этом в журнал событий будет записана соответствующая ошибка. По умолчанию таймаут в ядре равен `60×HZ`. |
| `--reset_backlog-_wait_time_actual` | Сбрасывает счётчик фактического времени ожидания уменьшения очереди событий, отображаемый статус-командой `auditctl -s`. |
| `-c` | Продолжать загружать правила несмотря на возникающие ошибки. Сохраняет результаты загрузки всех правил и, если хотя бы одно правило не загрузилось, то код возврата будет не нулевой. |
| `-D` | Удалить все правила и точки наблюдения. Может быть скомбинирован с аргументом `-k`. |

| Аргумент | Описание |
|---|---|
| `-e [0..2]` | Управляет состоянием службы аудита. Допустимые значения параметра перечислены после таблицы. |
| `-f [0..2]` | Устанавливает способ обработки возникающих сбоев. Флаг сбоя устанавливается при ошибках передачи данных из пространства ядра в службу `auditd`, работающую в пользовательском пространстве, при превышении максимального времени обработки очереди событий, нехватке памяти и т.п. Допустимые значения параметра перечислены после таблицы. |
| `-h` | Выдать справочную информацию об аргументах командой строки и завершить работу. |
| `-i` | Если используется самостоятельно, то включает режим игнорирования ошибок при чтении правил из файла — в таком случае `auditctl` всегда будет возвращать успешный код возврата. Если же используется в комбинации с аргументом `-s`, то, по возможности, переводит числовые идентификаторы в понятные пользователю слова. |
| `--loginuid-immutable` | Делает UID учётных записей неизменяемым сразу после его установки, что предотвращает возможность выдавать себя за других пользователей. Для изменения UID требуется полномочие `CAP_AUDIT_CONTROL`, поэтому непривилегированный пользователь не может его изменить. Использование этого параметра может вызвать проблемы при использовании контейнеризации. |
| `-t` | «Обрезать» неиспользуемые ветви поддеревьев каталогов после монтирования. |

| Аргумент | Описание |
|---|---|
| `-q <точка_монтирования, поддерево>` | При наличии точки наблюдения за каталогом и объединении или перемещении точки монтирования другого поддерева в наблюдаемое, указывает ядру сделать монтируемое поддерево эквивалентным наблюдаемому каталогу. Если поддерево уже смонтировано во время создания точки наблюдения, то оно автоматически помечается как наблюдаемое. Обратите внимание: значения разделяются запятой, её отсутствие вызовет ошибку. |
| `-r <частота>` | Устанавливает лимит сообщений аудита в секунду. Если значение больше нуля и было превышено, то ядром будет выставлен флаг сбоя для обработки. Значение по умолчанию — `0`, что означает отсутствие ограничений. |
| `--reset-lost` | Сбрасывает счётчик потерянных записей, отображаемых статус-командой `auditctl -s`. |
| `-R <путь_к_файлу>` | Считать и выполнить правила из указанного файла. Правила будут применяться построчно, в том порядке, в котором они определены в файле. Файл должен принадлежать пользователю `root` и быть недоступным для чтения и записи другими пользователями, в противном случае запрос на его обработку будет отклонён. Строки, имеющие в начале символ # считаются комментариями. Каждая строка будет обрабатываться как набор аргументов для команды `auditctl`. Поскольку файл обрабатывается именно командой `auditctl`, а не командной оболочкой `bash`, не экранируйте специальные символы `shell`. Примеры файлов с правилами приведены дальше в этом разделе. |
| `--signal <сигнал>` | Отправляет сигнал (`man 7 signal`) службе аудита, для этого потребуются соответствующие привилегии. Поддерживаемые сигналы перечислены после таблицы. |

**Допустимые значения параметров**

**-f [0..2]**:

- `0` — «тихий» режим, ничего не предпринимать.
- `1` — записать сообщение об ошибке в журнал сообщений ядра с помощью функции `printk()`.
- `2` — отрапортовать о критичной ошибке ядра и перевести его в состояние «kernel panic» заблокировав тем самым дальнейшую работу системы. Значение по умолчанию — `1`, но для защищённых окружений рекомендуется использовать `2`.

**-e [0..2]**:

- `0` — временно отключить службу аудита.
- `1` — включить службу аудита.
- `2` — включить службу аудита и заблокировать последующие изменения её конфигурации. Блокировка конфигурации должна быть последней командой в цепочке правил для службы аудита, после её применения любая попытка изменить конфигурацию службы аудита будет запротоколирована и отклонена. Изменение правил вновь станет возможным только после перезагрузки компьютера.

**Поддерживаемые сигналы для –signal <сигнал>**:

- `TERM` (`stop`) — завершает работу службы аудита.
- `HUP` (`reload`) — при получении данного сигнала служба `auditd` перечитает конфигурационный файл и, если не обнаружит синтаксических ошибок, попробует применить запрошенные изменения. В случае успешного завершения операции в журнале событий безопасности `/var/log/audit/audit.log` появится соответствующее сообщение типа `DAEMON_CONFIG`. В случае возникновения ошибки, в зависимости от её типа, будет выполнено одно из действий, определённых директивами `space_left_action`, `admin_space_left_action`, `disk_full_action` или `disk_error_action` в конфигурационном файле.
- `USR1` (`rotate`) — указывает службе аудита на необходимость прекратить запись в текущий файл журнала, создать новый файл и продолжить вести журнал в нём. Этот сигнал может быть полезным для организации процесса резервного копирования.
- `USR2` (`resume`) — указывает службе аудита на необходимость продолжить

ведение журнала событий безопасности. Обычно это требуется после приостановки ведения журнала или переполнения внутренней очереди подсистемы аудита.

Опции состояния перечислены в таблице.

Таблица 18 - Опции состояния `auditctl`

| Аргумент | Описание |
|---|---|
| `-l` | Вывести список всех правил по одному на строку.<br>В связке с этой опцией можно использовать ещё два параметра:<br>`-k <ключ>` — вывести только правила, соответствующие заданному ключу.<br>`-i` — интерпретировать значения полей от `a0` до `a3` для корректного определения аргументов системных вызовов. |
| `-m <текст>` | Отправить в подсистему аудита сообщение из пользовательского пространства. Это можно сделать только от имени суперпользователя `root` или если у вашего пользователя есть полномочия `CAP_AUDIT_WRITE`. Отправленное сообщение будет иметь тип `USER`. |
| `-s` | Показать отчёт о состоянии подсистемы аудита в ядре. В нём будут указаны значения, которые устанавливаются с помощью опций `-e`, `-f`, `-r` и `-b` команды `auditctl`. Значение `pid` — это идентификатор процесса службы аудита, если оно равно `0`, то служба аудита не запущена. Значение `lost` отображает количество записей, которые были удалены из-за переполнения очереди событий, ожидающих обработки. Значение `backlog` отображает количество записей, которые в данный момент находятся в очереди на обработку. Также к опции `-s` можно добавить параметр `-i` чтобы получить интерпретированное значение некоторых числовых полей. |
| `-v` | Вывести версию утилиты `auditctl`. |

Опции для управления правилами фильтрации перечислены в таблице.

Таблица 19 - Опции для управления правилами фильтрации `auditctl`

| Аргумент | Описание |
|---|---|
| `-a <список, действие \| действие,список>` | Добавить правило с указанным действием в конец списка. Значения должны быть разделены запятой, при этом их порядок не играет роли. Допустимые имена списков и действия для правил перечислены после таблицы. |
| `-A <список, действие>` | Добавить правило с указанным действием в начало списка. Информацию по доступным действиям и спискам смотрите выше в описании к директиве `-a`. |
| `-C <поле=поле \| поле!=поле>` | Создать условие сравнения двух полей для правила. Используется синтаксис первое_поле оператор второе_поле, поддерживаются операторы: = (равно) и != (не равно). В одном правиле может использоваться несколько операций сравнения, каждая должна начинаться с аргумента `-C`. Чтобы правило сработало для события аудита, все его условия, заданные директивами `-C` и `-F` должны быть выполнены.<br>Сравнение можно выполнять для следующих полей:<br>группа `uid`: `auid`, `uid`, `euid`, `suid`, `fsuid` и `obj_uid`.<br>группа `gid`: `gid`, `egid`, `sgid`, `fsgid` и `obj_gid`.<br>Сравнивать между собой можно только поля из одной группы. Поля `obj_uid` и `obj_gid` заполняются данными из объекта, для которого возникло событие — файла, каталога и т.п.. |
| `-d <список, действие>` | Удалить правило с указанным действием из списка. Правило будет удалено только если полностью совпали все поля условия и название системного вызова. |

| Аргумент | Описание |
|---|---|
| -F <поле=значение \| поле!=значение \| поле<значение \| поле>значение \| поле<=значение \| поле>=значение \| поле&значение \| поле&=значение> | Создать условие сравнения для правила. Используется синтаксис поле оператор значение, поддерживаются операторы: = (равно), != (не равно), < (меньше), > (больше), <= (меньше или равно), >= (больше или равно), & (битовая маска) и &= (битовая проверка). В одном правиле может быть до 64 условий, каждое должно начинаться с аргумента -F. Чтобы правило сработало для события аудита, все его условия, заданные директивами -F и -C должны быть выполнены. Для полей, содержащих идентификатор пользователя можно также использовать имя пользователя — программа самостоятельно преобразует имя в идентификатор. То же самое выполняется и для полей с идентификатором группы. Допустимые имена полей перечислены после таблицы. |
| -W <путь> | Удалить точку наблюдения за объектом файловой системы по указанному пути. Требуется полное соответствие правилу, смотрите описание к опции -d. |

| Аргумент | Описание |
|---|---|
| `-k <ключ>` | Устанавливает ключ фильтрации для данного правила. Ключом может быть произвольная текстовая строка длиной не больше 31 байта. Ключ позволяет однозначно идентифицировать записи журнала безопасности, созданные с помощью правила. Обычно используется когда у вас есть несколько правил, которые в совокупности удовлетворяют требованию безопасности. По ключу можно отфильтровать все соответствующие записи с помощью утилиты `ausearch`, не зависимо от того, какое правило сработало. Ключ также можно использовать для удаления определённых правил с помощью аргумента `-D` или для получения списка соответствующих правил с помощью аргумента `-l`. К одному правилу может быть привязано несколько ключей если вы хотите выполнять поиск по событиям несколькими способами или если вы используете собственное расширение `auditd` для анализа записей в журнале. |
| `-p <r\|w\|x\|a>` | Устанавливает фильтр прав доступа к объекту файловой системы: `r` — чтение (read), `w` — запись (write), `x` — исполнение (execute), `a` — изменение атрибутов (attribute change). Не следует путать эти права с обычными правами доступа к файлу, скорее они определяют системные вызовы, которые выполняют данные действия. Обратите внимание, системные вызовы `read` и `write` не включены в этот набор, поскольку они перегрузили бы журнал аудита сообщениями. |

| Аргумент | Описание |
|---|---|
| `-S <имя_или_номер_системного_вызова \| all>` | Название или номер системного вызова, который необходимо отслеживать. Также можно использовать ключевое слово `all`, которое включает обработку всех системных вызовов. В случае если установлен фильтр по другим полям, а системный вызов не указан, то правило будет применяться ко всем системным вызовам. Используя несколько опций `-S` можно указать несколько системных вызовов в одном правиле — это повышает производительность, поскольку потребуется вычислять меньшее количество правил. В качестве альтернативы, вы можете указать несколько системных вызовов через запятую.<br>Если вы работаете с системой, которая поддерживает несколько архитектур, например `x86_64`, то вы должны знать, что `auditctl` просто берёт название системного вызова, находит номер системного вызова в таблице вызовов для «родной архитектуры» (в данном случае, для `b64`) и отправляет это правило ядру. Соответственно, если в правиле не определено поле `arch`, то правило будет применено и к 64-битным системным вызовам, и к 32-битным. Это может привести к нежелательным последствиям, поскольку системные вызова для 32-битных и 64-битных систем могут иметь разные номера. Соответственно, в таком случае рекомендуется создавать два отдельных правила для `b32` и `b64`-архитектур. |

| Аргумент | Описание |
|---|---|
| `-w <путь>` | Добавить точку наблюдения за объектом файловой системы по указанному пути. Вы не можете создать точку наблюдения за корневым каталогом (`/`) поскольку это запрещено ядром. Групповые символы (wild-cards) запрещены — при попытке их использования будет выдано соответствующее предупреждение. Внутри точки наблюдения реализованы методом слежения за номерами индексных дескрипторов (`inodes`). Если вы устанавливаете точку наблюдения за файлом, то это равносильно использованию опции `-F path=значение` в правиле. Если же вы устанавливаете точку наблюдения за каталогом, то это равносильно использованию опции `-F dir=значение` в правиле. Форма записи правил через опцию `-w` предназначена для обратной совместимости, запись через `-F` является более выразительной. В отличии от большинства правил аудита системных вызовов, точки наблюдения за файловой системой не оказывают существенного влияния на производительность в зависимости от количества правил, отправленных ядру. Единственными допустимыми параметрами при использовании опции `-w` являются `-p` и `-k`. Если вам требуется выполнить что-то необычное, например, провести аудит доступа определённого пользователя к файлу, то воспользуйтесь опцией `-F` с аргументами `path` или `dir`. |

**Допустимые имена списков аргумента -a <список,действие | действие,список>**:

- `task` — добавить правило к списку, отвечающему за процессы. Этот список правил используется только во время создания процесса — когда родительский процесс вызывает функции `fork()` или `clone()`. При

использовании этого списка вы должны использовать только те поля, которые известны на момент создания процесса: `uid`, `gid` и т.д..

- `exit` — добавить правило к списку, отвечающему за точки выхода из системных вызовов. Этот список используется чтобы определить, следует ли создавать событие аудита при завершении системного вызова.
- `user` — добавить правило в список, отвечающий за фильтрацию пользовательских сообщений. Этот список используется ядром для фильтрации сообщений, исходящих из пользовательского пространства, перед передачей их службе аудита. При использовании этого списка поддерживаются только следующие поля: `uid`, `auid`, `gid`, `pid`, `subj_user`, `subj_role`, `subj_type`, `subj_sen`, `subj_clr`, `msgtype` и `exe`. Все остальные поля будут считаться не соответствующими условию. Следует понимать, что любое событие, исходящее из пространства пользователя, у которого есть полномочие `CAP_AUDIT_WRITE`, будет записано в журнал событий безопасности. Соответственно, в большинстве случаев этот фильтр будет использоваться с правилами, действие для которых — `never`, поскольку для записи события ничего не нужно делать.
- `exclude` — добавить правило, исключающее определённый тип события. Например, таким образом можно отключить протоколирование событий от SELinux AVC. События можно исключать по следующим полям: `pid`, `uid`, `gid`, `auid`, `msgtype`, `subj_user`, `subj_role`, `subj_type`, `subj_sen`, `subj_clr` и `exe`. Значение параметра `действие` игнорируется и всегда используется значение `never`.
- `filesystem` — добавить правило, которое будет применяться ко всем файловым системам заданного в поле `fstype` типа. Обычно этот фильтр используется для исключения всех событий, связанных со специальными файловыми системами, например `debugfs` или `tracefs`.
- `io_uring` — добавить правило к фильтру системных вызовов подсистемы ядра `io_uring`. Правила для этого фильтра должны указывать системный вызов с помощью аргумента `-S <системный_вызов>`, описанного ниже. Вы также можете использовать аргумент `-k <ключ>` для правила чтобы сгруппировать его с другими правилами, отслеживающими тот же самый системный вызов.

**Допустимые действия для правил аргумента -a <список,действие |**

**действие,список**>:

- never — не генерировать записи аудита для подходящих под правило событий. В общем случае рекомендуется размещать такие правила в начале списка, так как срабатывает первое подходящее правило.
- always — создавать запись аудита, всегда наполнять её данными в точке входа в системный вызов и выдавать на обработку в момент выхода из системного вызова.

**Допустимые имена полей аргумента -F <поле=значение | поле!=значение | поле<значение | поле>значение | поле<=значение | поле>=значение | поле&значение | поле&=значение**>:

- a0, a1, a2, a3 — первые четыре аргумента, переданные системному вызову. Строковые аргументы не поддерживаются поскольку ядро получает указатель на строку вместо самой строки. Соответственно, при использовании этих полей, вам следует использовать только числовые значения. Обычно это используется на платформах, мультиплексирующих сокеты или операции IPC.
- arch — архитектура процессора, на котором выполняется системный вызов. Узнать архитектуру системы можно с помощью команды uname -m. Если вы не знаете архитектуру своего компьютера, но хотите использовать 32-разрядные системные вызовы и ваш компьютер поддерживает 32 бита, вы также можете использовать b32 вместо архитектуры. Таким же образом можно использовать b64 для 64-разрядных системных вызовов. Таким образом можно создавать в некотором смысле независимые от архитектуры правила, поскольку тип семейства будет определяться автоматически. Однако, следует помнить, что системные вызовы могут быть специфичными для определённой архитектуры и то, что доступно для x86_64, может быть недоступно на aarch64. Архитектура должна указываться перед аргументом -S чтобы утилита auditctl могла определить в какой внутренней таблице искать номера системных вызовов.
- auid — исходный идентификатор пользователя, использованный для входа в систему. auid — это сокращение от audit uid, также его иногда называют loginuid. В качестве значения может использовать как идентификатор пользователя, так и его имя.

- devmajor — старший номер устройства (device major number).
- devminor — младший номер устройства (device minor number).
- dir — полный путь к каталогу для наблюдения. Это приведёт к рекурсивному просмотру этого каталога и всего его поддерева. Это поле можно использовать только в списке exit. Также смотрите описание опции -w.
- egid — действующий идентификатор группы. Может использоваться как числовой идентификатор, так и название.
- euid — действующий идентификатор пользователя. Может использоваться как числовой идентификатор, так и имя.
- exe — абсолютный путь к приложению, для которого будет применяться это правило. Поддерживаются только операторы = и !=. Это поле можно проверять только один раз для каждого правила.
- exit — код возврата из системного вызова. Если в качестве кода используется ошибка errno (man 3 errno), то можно использовать её текстовое представление.
- fsgid — идентификатор группы, применяемый к файловой системе.
- fstype — тип файловой системы, используется только в списке правил filesystem. Допустимые значения: debugfs и tracefs.
- fsuid — идентификатор пользователя, применяемый к файловой системе. Можно использовать как числовой идентификатор, так и имя пользователя.
- filetype — тип целевого файла. Допустимые значения: file, dir, socket, link, character, block или fifo.
- gid — идентификатор группы. Можно использовать как числовой идентификатор, так и название группы.
- inode — номер индексного дескриптора (inode).
- key — альтернативный способ установки ключа для фильтрации правил. Смотрите описание опции -k ниже.
- msgtype — тип сообщения, к которому должно применяться правило. Может использоваться только в списках exclude и user.
- obj_uid — идентификатор пользователя, связанный с объектом (файлом или каталогом).
- obj_gid — идентификатор группы, связанный с объектом (файлом или

каталогом).

- `obj_user` — имя SELinux пользователя, владеющего ресурсом.
- `obj_role` — SELinux роль ресурса.
- `obj_type` — SELinux тип ресурса.
- `obj_lev_low` — нижний уровень ресурса в контексте SELinux.
- `obj_lev_high` — верхний уровень ресурса в контексте SELinux.
- `path` — полный путь к файлу для точки наблюдения, используется только в списке правил `exit`.
- `perm` — фильтр прав доступа для файловых операций. Подробное описание доступно в справке по опции `-p`. Этот фильтр используется только в списке правил `exit`. Его также можно использовать без указания системного вызова — ядро само подберёт системные вызовы, которые удовлетворяют запрашиваемым разрешениям.
- `pers` — персональный номер операционной системы.
- `pid` — идентификатор процесса.
- `ppid` — идентификатор родительского процесса.
- `saddr_fam` — номер семейства протоколов, который указан в файле `/usr/include/bits/socket.h`. Например, IPv4 будет иметь номер 2, а IPv6 — 10.
- `sessionid` — идентификатор сеанса пользователя.
- `subj_user` — имя пользователя-владельца процесса в контексте SELinux.
- `subj_role` — SELinux роль процесса.
- `subj_type` — SELinux тип процесса.
- `subj_sen` — SELinux чувствительность процесса (Linux Sensitivity).
- `subj_clr` — SELinux допуск процесса (SELinux Clearance).
- `sgid` — установленный идентификатор группы (см. `man getresgid`).
- `success` — если код возврата системного вызова больше либо равен нулю, то данное поле будет иметь значение `true/yes`, иначе — `false/no`. При создании правила используйте 1 для `true/yes` и 0 для `false/no`.
- `suid` — установленный идентификатор пользователя (см. `man getresuid`).
- `uid` — идентификатор пользователя, можно использовать как числовой идентификатор, так и имя пользователя.

### 6.3.2. Создание правил аудита

В этом разделе приведены примеры решения некоторых типовых задач, связанных с регистрацией событий безопасности. В большинстве примеров используются временные правила для подсистемы аудита, процедура создания постоянных правил описана в следующем разделе.

#### 6.3.2.1. Контроль загрузки и выгрузки модулей ядра

Добавление и/или удаление модулей ядра может быть использовано для изменения поведения ядра и внедрения вредоносного кода в пространство ядра. Приведённый ниже набор правил позволит отслеживать такие операции:

```
# регистрировать любой запуск команды /usr/bin/kmod. В МСВСфера ОС
 ↪9 команды
# /usr/sbin/insmod, /usr/sbin/rmmod и /usr/sbin/modprobe являются
 ↪символическими
# ссылками на /usr/bin/kmod.
$ sudo auditctl -a always,exit -F path=/usr/bin/kmod -F perm=x \
  -F key=kernel_modules

# предыдущее правило так же можно переписать с использованием опции
 ↪`-w`,
# с точки зрения подсистемы аудита оба правила эквивалентны:
# sudo auditctl -w /usr/bin/kmod -p x -k kernel_modules

# отслеживать системные вызовы, выполняющие загрузку и выгрузку
 ↪модулей ядра
# на 64-битной архитектуре
$ sudo auditctl -a always,exit -F arch=b64 \
  -S init_module,finit_module,delete_module -F key=kernel_modules
```

Для всех подпадающих под правило записей в журнале событий безопасности будет устанавливаться ключ `kernel_modules`. Соответственно, для их просмотра можно использовать следующую команду:

```
$ sudo ausearch -k kernel_modules
```

#### 6.3.2.2. Отслеживание изменений в конфигурации sudo

Sudo — это утилита, позволяющая пользователю выполнять программы с привилегиями другой учётной записи, в том числе и от имени суперпользователя. Настройка sudo выполняется либо через редактирование основного конфигурационного файла /etc/sudoers, либо через создание дополнительных конфигурационных файлов в каталоге /etc/sudoers.d.

Следующий набор правил позволит отслеживать изменения в конфигурации sudo:

```
# отслеживать любые изменения файла /etc/sudoers
$ sudo auditctl -a always,exit -F path=/etc/sudoers -F perm=wa \
  -F key=sudo_config

# отслеживать любые изменения в каталоге /etc/sudoers.d
$ sudo auditctl -a always,exit -F dir=/etc/sudoers.d -F perm=wa \
  -F key=sudo_config
```

Эти правила также можно записать с использованием опции -w:

```
# отслеживать любые изменения файла /etc/sudoers
$ sudo auditctl -w /etc/sudoers -p wa -k sudo_config

# отслеживать любые изменения в каталоге /etc/sudoers.d
$ sudo auditctl -w /etc/sudoers.d -p wa -k sudo_config
```

Для просмотра записей можно будет использовать использовать следующую команду:

```
$ sudo ausearch -k sudo_config
```

#### 6.3.2.3. Отслеживание установки или обновления RPM-пакетов

Для отслеживания установки или обновления RPM-пакетов подсистемой аудита установите из репозиториев МСВСфера ОС пакет rpm-plugin-audit, который содержит необходимое расширение для пакетного менеджера RPM:

```
$ sudo dnf install -y rpm-plugin-audit
```

Больше никаких действий по настройке не требуется — при каждой

установке или обновлении RPM-пакета в журнал событий безопасности будет добавляться соответствующая запись с типом `SOFTWARE_UPDATE`.

Пример сообщения об обновлении пакета `osinfo-db`:

```
type=SOFTWARE_UPDATE msg=audit(1731661320.537:3549): pid=22903
 ↪uid=0 auid=1000 ses=4 subj=unconfined_u:unconfined_r:unconfined_
 ↪t:s0-s0:c0.c1023 msg='op=update sw="osinfo-db-20231215-1.el9.
 ↪inferit.noarch" sw_type=rpm key_enforce=0 gpg_res=1 root_dir="/"
 ↪comm="dnf" exe="/usr/bin/python3.9" hostname=msvsphere-94-arm.
 ↪msvsphere.test addr=? terminal=pts/2 res=success'
```

#### 6.3.2.4. Отслеживание установки модулей для Lua, NodeJS, Perl, Python, Ruby

Многие современные языки программирования имеют собственный пакетный менеджер для установки дополнительных библиотек или утилит из централизованных репозиториев. В набор правил подсистемы аудита МСВСфера ОС входят правила для отслеживания запуска следующих пакетных менеджеров:

- `/usr/bin/pip` — установщик Python-модулей
- `/usr/bin/npm` — установщик NodeJS-модулей
- `/usr/bin/cpan` — установщик Perl-модулей
- `/usr/bin/gem` — установщик Ruby-модулей
- `/usr/bin/luarocks` — установщик Lua-модулей
- `/usr/bin/dnf-3` — пакетный менеджер DNF
- `/usr/bin/yum` — пакетный менеджер Yum (в настоящее время является ссылкой на DNF чтобы обеспечить совместимость)

Данные правила находятся в файле `/usr/share/audit/sample-rules/44-installers.rules` и не включены по умолчанию. Для их активации выполните следующие действия.

1. Скопируйте файл с правилами в каталог постоянных правил подсистемы аудита `/etc/audit/rules.d`:

   ```
   $ sudo cp /usr/share/audit/sample-rules/44-installers.rules
    ↪/etc/audit/rules.d/
   ```

2. Загрузите обновлённый набор правил в подсистему аудита:

```
$ sudo augenrules --load
```

3. Убедитесь, что правила были загружены, с помощью команды:

```
$ sudo auditctl -l | grep software-installer
-p x-w /usr/bin/dnf-3 -k software-installer
-p x-w /usr/bin/yum -k software-installer
-p x-w /usr/bin/pip -k software-installer
-p x-w /usr/bin/npm -k software-installer
-p x-w /usr/bin/cpan -k software-installer
-p x-w /usr/bin/gem -k software-installer
-p x-w /usr/bin/luarocks -k software-installer
```

Теперь при запуске любого из отслеживаемых пакетных менеджеров в журнал подсистемы аудита будет добавлена соответствующая запись. Отфильтровать такие события можно по ключу `software-installer`. Ниже приведён пример события, которое было запротоколировано при выполнении команды `pip install markdown2`:

```
$ ausearch -k software-installer
----
time->Fri Nov 15 13:40:38 2024
type=PROCTITLE msg=audit(1731667238.489:3725):
 ↪proctitle=2F7573722F62696E2F707974686F6E33002F62696E2F70\
697000696E7374616C6C006D61726B646F776E32
type=PATH msg=audit(1731667238.489:3725): item=2 name="/lib64/ld-
 ↪linux-x86-64.so.2" inode=1980964 dev=fd:00 mode=0100755 ouid=0
 ↪ogid=0 rdev=00:00 obj=system_u:object_r:ld_so_t:s0
 ↪nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_
 ↪frootid=0
type=PATH msg=audit(1731667238.489:3725): item=1 name="/usr/bin/
 ↪python3" inode=1993087 dev=fd:00 mode=0100755 ouid=0 ogid=0
 ↪rdev=00:00 obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_
 ↪fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=PATH msg=audit(1731667238.489:3725): item=0 name="/bin/pip"
 ↪inode=1966587 dev=fd:00 mode=0100755 ouid=0 ogid=0 rdev=00:00
 ↪obj=system_u:object_r:bin_t:s0 nametype=NORMAL cap_fp=0 cap_fi=0
 ↪cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1731667238.489:3725): cwd="/home/user"
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
type=EXECVE msg=audit(1731667238.489:3725): argc=4 a0="/usr/bin/
 ↪python3" a1="/bin/pip" a2="install" a3="markdown2"
type=SYSCALL msg=audit(1731667238.489:3725): arch=c000003e
 ↪syscall=59 success=yes exit=0 a0=560b1acfe030 a1=560b1aceae30
 ↪a2=560b1abd46e0 a3=8 items=3 ppid=14961 pid=23067 auid=1000 uid=0
 ↪gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=4
 ↪comm="pip" exe="/usr/bin/python3.9" subj=unconfined_u:unconfined_
 ↪r:unconfined_t:s0-s0:c0.c1023 key="software-installer"
```

Более подробно работа с постоянными правилами подсистемы аудита и утилитой `ausearch` описана в следующих разделах.

#### 6.3.2.5. Рекомендации по оптимизации правил

Правила, анализирующие системные вызовы, проверяются для каждого системного вызова каждой выполняемой программы. Соответственно, если у вас десять правил для системных вызовов, то каждая программа во время выполнения любого системного вызова будет приостановлена пока подсистема аудита применяет каждое из этих правил. По возможности старайтесь комбинировать как можно больше системных вызовов в одном правиле, если фильтры, действия и ключ для этих системных вызовов совпадают.

В качестве примера рассмотрим следующие три правила, которые отслеживают системные вызовы, выполняющие загрузку и выгрузку модулей ядра:

```
$ sudo auditctl -a always,exit -F arch=b64 -S init_module -F
 ↪key=kernel_modules
$ sudo auditctl -a always,exit -F arch=b64 -S finit_module -F
 ↪key=kernel_modules
$ sudo auditctl -a always,exit -F arch=b64 -S delete_module -F
 ↪key=kernel_modules
```

Поскольку все остальные поля правила совпадают, их легко можно объединить в одно правило, которое уже приводилось в примере ранее:

```
$ sudo auditctl -a always,exit -F arch=b64 \
  -S init_module,finit_module,delete_module -F key=kernel_modules
```

Также можно указывать каждый системный вызов с помощью отдельного

аргумента `-S` — с точки зрения подсистемы аудита оба варианта эквивалентны:

```
$ sudo auditctl -a always,exit -F arch=b64 \
  -S init_module -S finit_module -S delete_module -F key=kernel_
 ↪modules
```

Также постарайтесь использовать точки наблюдения за файловой системой там, где это возможно. Это значительно повышает производительность, поскольку правила будут применяться только для операций с указанными файлами и каталогами, а не ко всем системным вызовам.

В качестве примера предположим, что вы бы хотели фиксировать все неудачные операции открытия и усечения (truncate) файлов. Тогда вы могли бы написать следующее правило:

```
$ auditctl -a always,exit -F arch=b64 -S openat -S truncate -F
 ↪success=0
```

Такое правило применялось бы ко всем системным вызовам всех программ, независимо от того, какие файлы они изменяют. Будут отслеживаться и изменения временных файлов в каталоге `/tmp`, и изменения файлов в домашних каталогах пользователя и любые другие операции.

Но, вероятнее всего, вместо любых операций вы хотели бы отслеживать изменения конкретных файлов, допустим, общесистемных файлов конфигурации в каталоге `/etc`. Тогда правило будет иметь следующий вид:

```
$ auditctl -a always,exit -F dir=/etc -F arch=b64 -S openat,
 ↪truncate -F success=0
```

И будет применяться только к файлам внутри каталога `/etc` или в его подкаталогах, что значительно сократит количество проверяемых системных вызовов.

### 6.3.3. Создание постоянных правил

Постоянные правила подсистемы аудита хранятся в виде файлов в каталоге `/etc/audit/rules.d/` и автоматически применяются во время запуска службы `auditd`.

Каждый файл с правилами должен иметь расширение `.rules`, например, `50-mail-server.rules`. Файлы с правилами обрабатываются и загружаются последовательно, в порядке естественной сортировки («natural sort order»). Общепринятой, хотя и не обязательной, является следующая схема именования файлов: `приоритет-описание.rules` где `приоритет` — целое число, задающее порядок загрузки файла, а `описание` — краткое описание назначения загружаемых правил.

При разработке правил для МСВСфера ОС предлагается придерживаться следующей схемы:

- `10` — правила для настройки ядра и подсистемы аудита;
- `20` — правила, переопределяющие настройки ядра и подсистемы аудита, поставляемые с операционной системой;
- `30` — основные правила;
- `40` — дополнительные/опциональные правила;
- `50` — правила, специфичные для группы серверов;
- `70` — правила, специфичные для локальной системы;
- `90` — правило, блокирующее дальнейшее изменение списка правил.

Файл с правилами подсистемы аудита является обычным текстовым файлом, в котором используется следующий формат:

- пустые строки и весь текст после символа # игнорируются;
- каждая непустая строка считается правилом и оформляется в виде списка аргументов для команды `auditctl` (см. раздел *6.3.2. Создание правил аудита*), при этом сама команда `auditctl` не указывается — она будет автоматически вызвана системой с заданными аргументами при обработке файла с правилами. На одной строке должно объявляться только одно правило;
- файл должен оканчиваться пустой строкой.

Пример файла с правилами, отслеживающими изменения в конфигурационных файлах системы виртуализации на базе гипервизора `libvirt`:

```
-w /etc/libvirt/libvirt-admin.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/libvirt.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/libvirtd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/qemu.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/qemu-lockd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtinterfaced.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtlockd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtlogd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtnetworkd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtnodedevd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtnwfilterd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtproxyd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtqemud.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtsecretd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtstoraged.conf -p wa -k libvirt-config-changes
-w /usr/share/polkit-1/actions/org.libvirt.api.policy -p wa -k
 ↪libvirt-polkit-changes
-w /usr/share/polkit-1/actions/org.libvirt.unix.policy -p wa -k
 ↪libvirt-polkit-changes
-w /usr/share/polkit-1/rules.d/50-libvirt.rules -p wa -k libvirt-
 ↪polkit-changes
```

С точки зрения безопасности рекомендуется назначать владельцем файлов с правилами пользователя `root`, группу `root` и разрешать доступ на чтение и запись только пользователю `root`:

```
$ sudo chown root:root /etc/audit/rules.d/50-mail-server.rules
$ sudo chmod 600 /etc/audit/rules.d/50-mail-server.rules
```

Как уже упоминалось ранее, правила из файлов загружаются автоматически при запуске службы `auditd`. Однако, в случае необходимости, можно загрузить новые правила с помощью следующей команды:

```
$ sudo augenrules --load
```

## 6.4. Работа с журналом событий безопасности

### 6.4.1. Формат файла журнала событий безопасности

В конфигурации по умолчанию подсистема аудита хранит текущий журнал событий безопасности в файле `/var/log/audit/audit.log`. Если ротация файлов журналов включена, то предыдущие файлы журнала будут находиться в том же каталоге.

Рассмотрим формат записей журнала на примере, для этого добавим в подсистему аудита следующее правило, которое будет регистрировать операции чтения файла `/etc/fstab`:

```
$ sudo auditctl -a always,exit -F path=/etc/fstab -F perm=r -F
 ↪key=fstab_read
```

Теперь прочитаем файл какой-нибудь командой, допустим `cat`:

```
$ cat /etc/fstab
```

В журнале `/var/log/audit/audit.log` появятся следующие записи о событии:

```
type=SYSCALL msg=audit(1731576783.677:1528): arch=c000003e
 ↪syscall=257 success=yes exit=3 a0=ffffff9c a1=7ffd91e5b610 a2=0
 ↪a3=0 items=1 ppid=19085 pid=20721 auid=1667 uid=1667 gid=1667
 ↪euid=1667 suid=1667 fsuid=1667 egid=1667 sgid=1667 fsgid=1667
 ↪tty=pts6 ses=7 comm="cat" exe="/usr/bin/cat" subj=unconfined_
 ↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="fstab_read"

type=CWD msg=audit(1731576783.677:1528): cwd="/home/user"

type=PATH msg=audit(1731576783.677:1528): item=0 name="/etc/fstab"
 ↪inode=262528 dev=fd:00 mode=0100644 ouid=0 ogid=0 rdev=00:00
 ↪obj=unconfined_u:object_r:etc_t:s0 nametype=NORMAL cap_fp=0 cap_
 ↪fi=0 cap_fe=0 cap_fver=0 cap_frootid=0 OUID="root" OGID="root"

type=PROCTITLE msg=audit(1731576783.677:1528):
 ↪proctitle=636174002F6574632F6673746162
```

Все четыре записи имеют одинаковое значение поля `msg`, что определяет их

принадлежность к одному событию. Записи журнала аудита всегда начинаются с поля `type`, каждая запись состоит из нескольких пар `поле=значение`, разделённых пробелом, в некоторых случаях может использоваться запятая.

Теперь рассмотрим каждую из записей подробнее.

Для первой записи установлены следующие поля.

- `type=SYSCALL` — поле `type` указывает на тип записи, в данном случае значение `SYSCALL` означает, что эта запись вызвана системным вызовом ядра.
- `msg=audit(1731576783.677:1528)` — поле `msg` содержит следующие данные.
    - Временную отметку и уникальный идентификатор события в формате `audit(временная_отметка:ID)`. Несколько записей в журнале аудита могут иметь одинаковую отметку и идентификатор если они были сгенерированы в рамках обработки одного события безопасности. Для временной отметки используется стандартный для Unix-подобных систем формат — количество секунд, прошедших с 00:00:00 по UTC 1 января 1970 года. Существует множество способов преобразования такой временной отметки в понятный человеку формат, самый простой из них — утилита `date`:

      ```
      $ date -d @1731576783.677
      Чт 14 ноя 2024 12:33:03 MSK
      ```

    - Различные специфичные для события пары `поле=значение`, полученные из ядра или из пользовательского пространства. Формально, все данные, которые находятся после `msg=audit(временная_отметка:ID): ...` являются значением поля `msg`, так что не удивляйтесь если в некоторых случаях вы даже увидите два поля `msg` в одной записи — такие записи встречаются, например, в событиях от гипервизора `libvirt`.
- `arch=c000003e` — информация об архитектуре системы, закодированная в шестнадцатеричной системе. При поиске записей с помощью утилиты ausearch используйте аргумент `-i / --interpret` чтобы автоматически преобразовывать закодированные значения в понятный человеку формат.

Значение `c000003e` будет преобразовано в `x86_64`.

- `syscall=257` — тип (номер) системного вызова, который был отправлен ядру. Для 64-битной архитектуры понятное пользователю значение можно получить из файла `/usr/include/asm/unistd_64.h`. В данном случае, `257` — это системный вызов `openat`. Для преобразования номера системного вызова в его название можно использовать утилиту `ausyscall`:

```
$ ausyscall 257
openat
```

  С помощью команды `ausyscall --dump` можно получить список номеров всех системных вызовов с их именами.
- `success=yes` — указывает на то, был ли системный вызов завершён успешно или возникла какая-то ошибка. В этом примере вызов был успешным.
- `exit=3` — код возврата, который вернул системный вызов. Это значение может отличаться для разных системных вызовов.
- `a0=ffffff9c a1=7ffd91e5b610 a2=0 a3=0` — в полях от `a0` до `a3` содержатся первые четыре аргумента системного вызова, закодированные в шестнадцатеричной системе счисления. Значения этих аргументов зависят от системного вызова и могут быть декодированы с помощью утилиты `ausearch`.
- `items=1` — количество вспомогательных записей типа `PATH`, которые следуют в журнале за данной записью системного вызова.
- `ppid=19085` — идентификатор родительского процесса (parent PID).
- `pid=20721` — идентификатор анализируемого процесса (PID).
- `auid=1667` — исходный идентификатор пользователя, использованный для входа в систему (audit id). Этот идентификатор наследуется каждым процессом, даже если идентификатор пользователя был изменён, например, при переключении учётных записей с помощью команды `su -`.
- `uid=1667` — идентификатор пользователя, запустившего анализируемый процесс.
- `gid=1667` — идентификатор группы пользователя, запустившего анализируемый процесс.

- euid=1667 — действующий идентификатор пользователя, запустившего процесс. Определяет текущие полномочия процесса.
- suid=1667 — установленный идентификатор пользователя, запустившего процесс. Используется для хранения начального значения EUID, задаваемого при запуске файла с установленным битом set-user-ID.
- fsuid=1667 — идентификатор пользователя файловой системы, запустившего процесс.
- egid=1667 — действующий идентификатор группы пользователя, запустившего процесс. Определяет текущие полномочия процесса.
- sgid=1667 — установленный идентификатор группы пользователя, запустившего процесс. Используется для хранения начального значения EGID, задаваемого при запуске файла с установленным битом set-group-ID.
- fsgid=1667 — идентификатор группы пользователя файловой системы, запустившего процесс.
- tty=pts6 — терминал, с которого был запущен процесс. Типовые форматы значений для этого поля:
    - ttyN — физический терминал.
    - pts/N — терминал, который эмулируется какой-то программой, допустим, сервером SSH. Командные оболочки, запущенные в графической среде, будут являться pts-терминалами.
    - (none) — терминал отсутствует, обычно такое значение устанавливается для системных вызовов, выполняемых системными процессами.
- ses=7 — идентификатор сессии, из которой был запущен процесс.
- comm="cat" — название команды, которая использовалась для запуска процесса.
- exe="/usr/bin/cat" — путь к исполняемому файлу, который использовался для запуска процесса.
- subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 — SELinux контекст анализируемого процесса.
- key="fstab_read" — определённый администратором ключ фильтрации, связанный с правилом, которое сгенерировало событие безопасности.

Для второй записи установлены следующие поля (описание поля msg

пропущено, так как полностью совпадает с первой записью):

- `type=CWD` — тип `CWD` используется для записи о текущем рабочем каталоге, из которого был запущен анализируемый процесс, вызвавший системный вызов из первой записи. Цель этой записи состоит в том, чтобы зафиксировать местоположение текущего процесса на случай, если в запись с типом `PATH` будет записан относительный путь. Таким образом, можно будет восстановить абсолютный путь к выполненной команде.
- `cwd="/home/user"` — путь к каталогу, из которого был сделан системный вызов.

Для третьей записи установлены следующие поля (описание поля `msg` пропущено, так как полностью совпадает с первой записью):

- `type=PATH` — тип `PATH` используется для записей о каждом пути, который был передан системному вызову в качестве аргумента. В нашем случае, в системный вызов openat передавался только один путь — `/etc/fstab`. Для системных вызовов, принимающих несколько путей в качестве аргументов, подсистема аудита создала бы соответствующее количество записей данного типа.
- `item=0` — порядковый номер этой записи среди общего числа записей типа `PATH` для данного события безопасности. Нумерация начинается с нуля, соответственно, в этом примере рассматривается первая запись.
- `name="/etc/fstab"` — путь к файлу или каталогу, который был передан системному вызову в качестве аргумента.
- `inode=262528` — номер индексного дескриптора (`inode`), связанного с этим файлом или каталогом. Зная этот номер, можно определить путь к файлу или каталогу с помощью следующей команды:

```
$ find / -inum  262528 -print
/etc/fstab
```

- `dev=fd:00` — младший (minor) и старший (major) номера устройства, которое содержит файл или каталог, записанный в этом событии.
- `mode=0100644` — права доступа к файлу или каталогу, записанные в числовой форме, которые возвращаются функцией `stat` (`man 2 stat`) в поле `st_mode`. В данном случае `0100644` можно интерпретировать как `-rw-r--r--`, что означает, что пользователь `root` имеет право на чтение и

запись, а остальные пользователи только на чтение.

- `ouid=0` — идентификатор пользователя, который владеет заданным файлом или каталогом.
- `ogid=0` — идентификатор группы, которая владеет заданным файлом или каталогом.
- `rdev=00:00` — младший (minor) и старший (major) номера устройства для специальных файлов. В данном случае он не используется, потому что `/etc/fstab` является обычным файлом.
- `obj=unconfined_u:object_r:etc_t:s0` — SELinux контекст файла или каталога на момент выполнения системного вызова.
- `nametype=NORMAL` — обозначает тип файлового объекта внутри контекста события. Подсистемой аудита используются следующие типы:
    - `UNKNOWN` — файловый объект не известен системе, например, его не существует.
    - `NORMAL` — обычный файловый объект. Как правило, это исполняемый файл или файл, у которого меняются атрибуты.
    - `PARENT` — файловый объект является родительским для одного из объектов, представленного в событии безопасности.
    - `DELETE` — файловый объект, удаляемый во время выполнения системного вызова.
    - `CREATE` — файловый объект, создаваемый во время выполнения системного вызова.
- `cap_fp=0` — данные, относящиеся к настройке разрешённых возможностей файловой системы для файлового объекта.
- `cap_fi=0` — данные, относящиеся к настройке унаследованных возможностей файловой системы для файлового объекта.
- `cap_fe=0` — установка эффективного бита возможностей файловой системы для файлового объекта.
- `cap_fver=0` — версия возможностей файловой системы для файлового объекта.

Для четвёртой записи установлены следующие поля (описание поля `msg` пропущено, так как полностью совпадает с первой записью):

- `type=PROCTITLE` — тип `PROCTITLE` указывает на то, что данная запись содержит полную команду запуска процесса, вызвавшего данное событие

безопасности.

- `proctitle=636174002F6574632F6673746162` — закодированная в шестнадцатеричной системе счисления полная команда запуска процесса. Утилита `ausearch`, запущенная с аргументом `-i / --interpret` автоматически преобразует закодированный текст в понятную пользователю строку, в нашем примере — в `cat /etc/fstab`.

Записи подсистемы аудита не ограничиваются перечисленными выше четырьмя типами, полный список типов доступен в конце этой главы.

Вероятнее всего вам никогда не потребуется работать с файлом журнала событий безопасности напрямую поскольку подсистема аудита включает в себя различные утилиты для поиска и обработки событий. Две основные, `aureport` и `ausearch`, будут рассмотрены в следующих разделах.

### 6.4.2. Утилита aureport

Команда `aureport` генерирует итоговые отчёты на основе зарегистрированных в журнале безопасности событий. Также `aureport` может принимать данные, поступающие на стандартный ввод (`stdin`), до тех пор пока на входе будут необработанные события аудита. За исключением основного сводного отчёта, все остальные отчёты содержат номер события в журнале аудита. Зная этот номер, вы можете посмотреть все данные по этому событию, используя команду `ausearch -a <НОМЕР_СОБЫТИЯ>`. Также существует возможность задать временной интервал, за который необходимо сгенерировать отчёт.

Опции командной строки утилиты `aureport` перечислены в таблице.

Таблица 20 - Опции командной строки утилиты `aureport`

| Аргумент | Описание |
|---|---|
| `-au, --auth` | Сгенерировать отчёт о попытках аутентификации. |
| `-a, --avc` | Сгенерировать отчёт о предоставлении разрешений SELinux AVC (Access Vector Cache). |
| `--comm` | Сгенерировать отчёт о выполнении команд. |
| `-c, --config` | Сгенерировать отчёт об изменениях конфигурации. |
| `-cr, --crypto` | Сгенерировать отчёт о событиях, связанных с криптографией. |
| `--debug` | Выводить искажённые события, которые были пропущены, в стандартный поток ошибок (`stderr`). |

| Аргумент | Описание |
|---|---|
| `--eoe-timeout <секунды>` | Устанавливает время ожидания разбора событий. Подробности доступны в описании директивы `end_of_event_timeout` конфигурационного файла `auditd.conf`. Значение, переданное в этом аргументе, имеет больший приоритет, чем значение, указанное в `auditd.conf`. |
| `-e, --event` | Сгенерировать отчёт о событиях. |
| `--escape <режим>` | Эта опция определяет, требуется ли экранирование вывода чтобы сделать его более безопасным для некоторых сценариев. Доступны следующие режимы экранирования: `raw`, `tty`, `shell` и `shell_quote`. Каждый режим включает правила экранирования из предыдущего режима и экранирует всё больше символов. Например, `shell` экранирует всё то, что экранируется в `tty` и добавляет экранирование дополнительных символов. Режим по умолчанию — `tty`. |
| `-f, --file` | Сгенерировать отчёт об операциях с файлами и Unix-сокетами. |
| `--failed` | Строить отчёт только на основе неудачных событий. По умолчанию показываются все события независимо от их статуса. |
| `-h, --host` | Сгенерировать отчёт о системе, который, среди прочего, включает события обновления ПО, аутентификации и т.п. |
| `--help` | Выдать справочную информацию об аргументах командой строки и завершить работу. |
| `-i, --interpret` | Включает преобразование числовых значений в текст. Например, идентификатор пользователя будет преобразован в его имя. Преобразование выполняется с использованием ресурсов текущего компьютера, на котором запущена команда `aureport`. Если вы переименовывали учётные записи или анализируете данные с другой системы, то вы можете получить ошибочные результаты. |

| Аргумент | Описание |
|---|---|
| `-if, --input < файл \| каталог >` | Использовать указанный файл или каталог вместо системного журнала для построения отчёта. Это может быть полезно в случае анализа журналов на другом компьютере или если сохранилась только часть журнала. |
| `--input-logs` | Получить путь к журналу аудита для анализа из конфигурационного файла `audit.conf`. Применяется при автоматическом формировании отчётов через `cron`. |
| `--integrity` | Сгенерировать отчёт о событиях целостности. |
| `-k, --key` | Сгенерировать отчёт по ключевым словам в правилах аудита. |
| `-l, --login` | Сгенерировать отчёт о попытках входа в систему. |
| `-m, --mods` | Сгенерировать отчёт об изменениях учётных записей. |
| `-ma, --mac` | Сгенерировать отчёт об изменениях в системе мандатного доступа SELinux. |
| `-n, --anomaly` | Сгенерировать отчёт об аномальных событиях, сюда, в том числе, входят события о переходе сетевой карты в «неразборчивый» режим (promiscuous mode) и ошибки сегментирования (segmentation fault). |
| `--node <имя-узла>` | Использовать для выборки только события, поступившие с определённого узла. По умолчанию отчёт генерируется по всем узлам. Допускается указание имён нескольких узлов. |
| `-nc, --no-config` | Не включать в отчёт события с типом `CONFIG_CHANGE`. Это может быть полезным для отчёта по ключевым словам, поскольку многие правила аудита их используют — указание этой опции поможет избежать ложных срабатываний. |
| `-p, --pid` | Сгенерировать отчёт о процессах. |
| `-r, --response` | Сгенерировать отчёт о реагировании на аномальные события. |
| `-s, --syscall` | Сгенерировать отчёт о системных вызовах. |
| `--success` | Строить отчёт только на основе событий, завершившихся успешно. |

| Аргумент | Описание |
|---|---|
| `-t, --log` | Сгенерировать отчёт о временных периодах каждого файла журнала подсистемы аудита. |
| `--tty` | Сгенерировать отчёт о нажатиях клавиш в терминале. |
| `-te, --end [дата] [время]` | Учитывать только события, которые произошли раньше или во время указанной временной отметки. Формат даты зависит от ваших региональных настроек (см. описание переменной окружения `LC_TIME`). Если дата не указана, то используется значение `today`. Если время не указано, то используется значение `now`. Для указания времени используется 24-часовой формат. Ключевые слова перечислены после таблицы. |
| `-u, --user` | Сгенерировать отчёт о пользователях. |
| `-v, --version` | Вывести версию программы aureport и завершить работу. |
| `--virt` | Сгенерировать отчёт о событиях системы виртуализации. |
| `-x, --executable` | Сгенерировать отчёт об исполняемых файлах. |

**Ключевые слова агумента -te, –end [дата] [время]**:

- `now` — сейчас.
- `recent` — 10 минут назад.
- `boot` — время за секунду до последней загрузки системы.
- `today` — сегодня.
- `yesterday` — 1 секунда после полуночи вчерашнего дня.
- `this-week` — 1 секунда после полуночи 0 (первого) дня текущей недели (определяется вашими региональными настройками).
- `week-ago` — 1 секунда после полуночи ровно 7 дней назад.
- `this-month` — 1 секунда после полуночи первого дня текущего месяца.
- `this-year` — 1 секунда после полуночи первого января текущего года.

#### 6.4.2.1. Примеры использования утилиты aureport

##### Отчёт об изменениях учётных записей

Сгенерировать отчёт о событиях безопасности, связанных с изменением пользовательских учётных записей, за последние сутки:

```
$ sudo aureport --mods --start yesterday

Account Modifications Report
================================================
# date time auid addr term exe acct success event
================================================
1. 12/09/2024 18:11:56 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd devuser yes 2199
2. 12/09/2024 18:11:56 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd devuser yes 2200
3. 12/09/2024 18:11:56 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd devuser yes 2201
4. 12/09/2024 18:11:56 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd devuser yes 2202
5. 12/09/2024 18:11:56 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd ? yes 2203
6. 12/09/2024 18:12:04 1000 libvirt.msvsphere.test pts/11 /usr/bin/
 ↪passwd devuser yes 2204
7. 12/09/2024 18:12:53 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/groupadd ? yes 2205
8. 12/09/2024 18:12:53 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/groupadd ? yes 2206
9. 12/09/2024 18:13:48 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd qauser yes 2207
10. 12/09/2024 18:13:48 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd qauser yes 2208
11. 12/09/2024 18:13:48 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd qauser yes 2209
12. 12/09/2024 18:13:48 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd qauser yes 2210
13. 12/09/2024 18:13:48 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/useradd ? yes 2211
14. 12/09/2024 18:13:59 1000 libvirt.msvsphere.test pts/11 /usr/
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪bin/passwd qauser yes 2212
15. 12/09/2024 18:14:14 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/groupadd ? yes 2213
16. 12/09/2024 18:14:14 1000 libvirt.msvsphere.test pts/11 /usr/
 ↪sbin/groupadd ? yes 2214
```

Результатом работы команды является таблица, в которой на каждой строке представлена информация о событии безопасности, а в столбцах следующая информация:

- `#` — порядковый номер строки в таблице;
- `date` — дата события;
- `time` — время события;
- `auid` — исходный идентификатор пользователя, инициировавшего событие. Вы также можете использовать аргумент `-i` для автоматического преобразования идентификатора в имя пользователя;
- `addr` — имя узла или его IP адрес;
- `term` — терминал, с которого была запущена команда, вызвавшая событие;
- `exe` — запущенная команда;
- `acct` — название учётной записи, для которой проводилось изменение;
- `success` — статус операции;
- `event` — идентификатор события.

Зная идентификатор события, можно посмотреть всю доступную информацию о нём с помощью команды `ausearch`.

```
$ sudo ausearch -a 2212
----
time->Mon Dec  9 18:13:59 2024
type=USER_CHAUTHTOK msg=audit(1733757239.466:2212): pid=589141
 ↪uid=0 auid=1000 ses=4 subj=unconfined_u:unconfined_r:passwd_t:s0-
 ↪s0:c0.c1023 msg='op=PAM:chauthtok grantors=pam_pwquality,pam_unix
 ↪acct="qauser" exe="/usr/bin/passwd" hostname=libvirt.msvsphere.
 ↪test addr=? terminal=pts/11 res=success'
```

В данном примере пользователь с идентификатором `1000` успешно изменил пароль пользователю `qauser` с помощью команды `passwd`.

### Итоговый отчёт о входе пользователей в систему

Для некоторых данных поддерживается формирование итоговых отчётов с помощью аргумента `--summary`. Например, следующая команда генерирует итоговый отчёт о входе пользователей в систему за последние 7 дней.

```
$ sudo aureport -u -i --summary --start yesterday

User Summary Report
===========================
total  auid
===========================
1185  unset
36  virtadmin
27  gdm
24  vmuser
21  devuser
9  vmdev
7  qauser
1  root
```

В первом столбце указано суммарное количество зарегистрированных событий входа в систему, а во втором — исходный идентификатор пользователя, преобразованный в его имя с помощью аргумента `-i`.

### 6.4.3. Утилита ausearch

Утилита `ausearch` — это инструмент для поиска событий в журнале службы аудита на основе различных критерий. Опции командной строки утилиты `ausearch` перечислены в таблице.

Таблица 21 - Опции командной строки утилиты `ausearch`

| Аргумент | Описание |
|---|---|
| `-c, --comm <название_команды>` | Искать события, связанные с указанным именем команды. |

| Аргумент | Описание |
|---|---|
| `--arch <архитектура>` | Выполнить поиск событий для указанной архитектуры процессора. Вместо явного указания архитектуры вы также можете использовать константы `b32` для 32-битных архитектур и `b64` для 64-битных. Узнать архитектуру вашей системы можно с помощью команды `uname -m`. |
| `-a, --event <идентифика-тор_события>` | Выполнить поиск события с заданным идентификатором. Все сообщения от подсистемы аудита имеют поле вида `msg=audit(1731576783.677:1528): ...`, идентификатор события — это число после символа `:`, в данном примере — `1528`. Все записи, связанные с одним системным вызовом приложения, имеют одинаковый идентификатор. Следующий системный вызов, выполненный тем же приложением, уже будет иметь другой идентификатор, таким образом обеспечивается уникальность. |
| `--debug` | Выводить искажённые события, которые были пропущены, в стандартный поток ошибок (stderr). |
| `-w, --word <слово>` | Выполнить поиск событий, у которых значение поля полностью совпадает с указанным словом. Поддерживаются следующие поля: имя файла, имя компьютера, терминал, SELinux контекст. |
| `-x, --executable <программа>` | Выполнить поиск событий с заданным именем исполняемой программы. |
| `-vm, --vm-name <название_госте-вой_системы>` | Выполнить поиск событий, связанных с заданным названием гостевой системы (виртуальной машины). |

| Аргумент | Описание |
|---|---|
| `--checkpoint <файл_контроль-ной_точки>` | Использовать файл контрольной точки для сохранения состояния, чтобы при последующих вызовах `ausearch` выводились только события, которые не отображались ранее.<br>События подсистемы аудита могут состоять из одной или нескольких записей. При их обработке `ausearch` определяет событие как завершённое или незавершённое. Завершённое — это либо событие с одной записью, либо событие, которое произошло на две секунды (см. описание опции `--eoe-timeout`) раньше по сравнению с текущим обрабатываемым событием (см. описание директивы `end_of_event_timeout` в конфигурационном файле `auditd.conf`).<br>При использовании опции `--checkpoint` в `файл_контрольной_точки` записывается последнее завершённое событие, а также номер устройства и номер индексного дескриптора (`inode`) файла журнала, из которого было получено это событие. При следующем вызове `ausearch` загрузит эти данные из файла контрольной точки и будет игнорировать все события из журналов до тех пор, пока не будет обнаружено совпадение с контрольной точкой. После этого утилита начнёт выводить завершённые события.<br>Если указанный в контрольной точке файл или последнее событие не будут найдены, то `ausearch` завершит свою работу с ошибкой (см. таблицу кодов возврата ниже). |

| Аргумент | Описание |
|---|---|
| `--eoe-timeout <секунды>` | Устанавливает количество секунд, после которого событие считается завершённым при анализе потока журнала событий. Подробности смотрите в описании директивы `end_of_event_timeout` конфигурационного файла `auditd.conf`. Установка этой опции переопределит значение, указанное в файле `auditd.conf`. |
| `-e, --exit <код>` | Выполнить поиск событий на основе указанного кода возврата из системного вызова или кода ошибки errno (`man errno`). |
| `--escape <режим>` | Эта опция определяет, требуется ли экранирование вывода чтобы сделать его более безопасным для некоторых сценариев. Доступны следующие режимы экранирования: `raw`, `tty`, `shell` и `shell_quote`. Каждый режим включает правила экранирования из предыдущего режима и экранирует всё больше символов. Например, `shell` экранирует всё то, что экранируется в `tty` и добавляет экранирование дополнительных символов. Режим по умолчанию — `tty`. |
| `--extra-keys` | Если `format` установлен в `csv`, эта опция добавит последний столбец с информацией о ключе, если он задан для события. Это будет применимо только к записям типа `SYSCALL`, которые были записаны в результате обработки правила аудита, определяющего ключ. |
| `--extra-labels` | Если `format` установлен в `csv`, эта опция добавит столбцы с SELinux метками субъекта и объекта если они определены. |

| Аргумент | Описание |
|---|---|
| `--extra-obj2` | Если `format` установлен в `csv`, эта опция добавит информацию о втором объекте, если он указан в записи о событии. Второй объект иногда является частью записи, например, при переименовывании файла или монтировании устройства. |
| `--extra-time` | Если `format` установлен в `csv`, эта опция добавит дополнительные столбцы с разобранным по полям временем и датой: `YEAR`, `MONTH`, `DAY`, `WEEKDAY`, `HOUR`, `MILLI` и `GMT_OFFSET`. |
| `-f, --file <имя_файла>` | Выполнить поиск событий, связанных с указанным именем файла. Эта опция будет применима как к обычным файлам, так и к Unix-сокетами (AF_UNIX/AF_LOCAL). |
| `--format <формат>` | Определяет формат вывода сообщений, соответствующих критериям поиска. Допустимые значения параметра перечислены после таблицы. |
| `-ga, --gid-all <идентифика-тор_группы>` | Выполнить поиск событий, у которых либо действующий идентификатор группы, либо идентификатор группы совпадает с заданным `идентификатором группы`. |
| `-ge, --gid-effective <идентифика-тор_группы>` | Выполнить поиск событий, у которых действующий идентификатор группы соответствует заданному `идентификатору группы`. Также в качестве параметра можно использовать название группы. |
| `-gi, --gid <идентифика-тор_группы>` | Выполнить поиск событий, у которых идентификатор группы соответствует заданному `идентификатору группы`. Также в качестве параметра можно использовать название группы. |
| `--help` | Выдать справочную информацию об аргументах командой строки и завершить работу. |

| Аргумент | Описание |
|---|---|
| -hn, --host <имя_узла> | Выполнить поиск событий, связанных с заданным именем узла. Имя узла может быть именем компьютера, полным именем компьютера (FQDN) или IP-адресом. Преобразование IP-адресов в доменные имена не выполняется. Обычно этот критерий поиска коррелирует с полями записей addr или host. Также смотрите описание опции --node, которая выполняет поиск по полю node. |
| -i, --interpret | Включает преобразование числовых значений в текст. Например, идентификатор пользователя будет преобразован в его имя. Преобразование выполняется с использованием ресурсов текущего компьютера, на котором запущена команда aureport. Если вы переименовывали учётные записи или анализируете данные с другой системы, то вы можете получить ошибочные результаты. |
| -if, --input <файл \| каталог> | Использовать указанный файл или каталог вместо системного журнала для построения отчёта. Это может быть полезно в случае анализа журналов на другом компьютере или если сохранилась только часть журнала. |
| --input-logs | Получить путь к журналу аудита для анализа из конфигурационного файла audit.conf. Применяется при автоматическом формировании отчётов через cron. |
| --just-one | Остановить поиск после выдачи первого события, соответствующего критериям поиска. |
| -k, --key <ключ> | Выполнить поиск событий, связанных с указанным ключом. Подробности доступны в описании опции -k команды auditctl. |

| Аргумент | Описание |
|---|---|
| -l, --line-buffered | Сбрасывать буфер вывода после каждой строки. Обычно используется когда стандартный вывод перенаправляется через канал и буферизация вывода является нежелательной. Использование этой опции может привести к снижению производительности. |
| -m, --message <тип \| список_типов> | Выполнить поиск записей о событиях с указанным типом. Поддерживается указание нескольких типов через запятую без пробелов или с помощью отдельных параметров -m. Вы также можете использовать не существующий реально тип ALL, который включает в себя события всех типов.<br>Вы можете посмотреть список всех поддерживаемых типов, если запустите команду ausearch -m без указания типа. |
| -n, --node <имя_узла \| список_имён_узлов> | Выполнить поиск событий, исходящих от определённого компьютера (узла). Допускается перечисление нескольких узлов через запятую, для вывода события достаточно совпадения с одним из перечисленных имён. Поиск выполняется по полю записи node. Также смотрите описание директивы --host, которая выполняет поиск событий, связанных с именем или IP-адресом узла. |
| -o, --object <контекст_SELinux> | Выполнить поиск событий, связанных с объектами, которым присвоен указанный контекст SELinux. Поиск выполняется по полю obj. |
| -p, --pid <pid> | Выполнить поиск событий, связанных с заданным идентификатором процесса (pid). |
| -pp, --ppid <pid_родитель-ского_процесса> | Выполнить поиск событий, связанных с заданным идентификатором родительского процесса (parent pid). |

| Аргумент | Описание |
|---|---|
| `-r, --raw` | Не применять какое-либо форматирование к выводу. Это полезно для извлечения найденных записей в файл, с которым смогут продолжать работать команды подсистемы аудита. |
| `-sc, --syscall <системный_вызов>` | Выполнить поиск событий, связанных с указанным системным вызовом. Вы можете указать либо название, либо номер системного вызова. Если вы указываете название, то `ausearch` определит номер вызова на основании таблицы системных вызовов для архитектуры компьютера, на котором запущена команда. |
| `-se, --context <контекст_SELinux>` | Выполнить поиск событий, связанных с объектами или субъектами, которым присвоен указанный контекст SELinux. Поиск выполняется по полям `obj` и `subj`. |
| `--session <сессия>` | Выполнить поиск событий, связанных с заданным идентификатором пользовательской сессии. Этот атрибут устанавливается при входе пользователя в систему и позволяет связать процесс с определённым сеансом пользователя. |
| `-su, --subject <контекст_SELinux>` | Выполнить поиск событий, связанных с субъектами, которым присвоен указанный контекст SELinux. Поиск выполняется по полю `subj`. |
| `-sv, --success <статус>` | Выполнить поиск событий, завершившихся с заданным статусом. Допустимые значения: `yes` (успешно) и `no` (неудачно). |
| `-ul, --loginuid <идентифика-тор_пользователя>` | Выполнить поиск событий, у которых исходный идентификатор пользователя (`auid`) соответствует указанному идентификатору пользователя. |

| Аргумент | Описание |
|---|---|
| -te, --end [дата] [время] | Учитывать только события, которые произошли раньше или во время указанной временной отметки. Формат даты зависит от ваших региональных настроек (см. описание переменной окружения LC_TIME). Если дата не указана, то используется значение today. Если время не указано, то используется значение now. Для указания времени используется 24-часовой формат. Ключевые слова для параметра перечислены после таблицы. |
| -ts, --start [дата] [время] | Учитывать только события, которые произошли позже или во время указанной временной отметки. Формат даты зависит от ваших региональных настроек (см. описание переменной окружения LC_TIME). Если дата не указана, то используется значение today. Если время не указано, то используется значение now. Для указания времени используется 24-часовой формат. Ключевые слова для параметра перечислены после таблицы. |
| -tm, --terminal <терминал> | Выполнить поиск событий, связанных с заданным терминалом. Некоторые службы, такие как cron и atd, используют имя службы как имя терминала. |
| -ua, --uid-all <идентифика-тор_пользователя> | Выполнить поиск событий, у которых либо идентификатор пользователя (uid), либо действующий идентификатор пользователя (euid), либо исходный идентификатор пользователя (auid) соответствуют указанному идентификатору пользователя. |
| -ue, --uid-effective <идентифика-тор_пользователя> | Выполнить поиск событий, у которых действующий идентификатор пользователя (euid) соответствует указанному идентификатору пользователя. |

| Аргумент | Описание |
|---|---|
| -ui, --uid <идентифика-тор_пользователя> | Выполнить поиск событий, у которых идентификатор пользователя (uid) соответствует указанному идентификатору пользователя. |
| -uu, --uuid <идентифика-тор_госте-вой_системы> | Выполнить поиск событий, связанных с заданным идентификатором гостевой системы (UUID). |
| -v, --version | Вывести версию утилиты ausearch и завершить работу. |

**Допустимые значения параметра –format <формат>**:

- raw — смотрите описание к опции --raw.
- default — формат, в котором вы получаете вывод если опция --format не задана: сначала идёт строка-разделитель, затем — временная отметка события, после — все записи, связанные с этим событием.
- interpret — смотрите описание к опции --interpret.
- csv — выводит результаты поиска в виде нормализованных событий в формате значений, разделённых запятой (CSV) — этот формат подходит для импорта в аналитические программы.
- text — превращает каждое событие в предложение на английском языке, которое легче понять, но при этом теряются различные детали. В большинстве случаев это считается нормальным, поскольку исходное событие по прежнему содержит полный объём информации.

**Ключевые слова для параметра -te, –end [дата] [время]**:

- now — сейчас.
- recent — 10 минут назад.
- boot — время за секунду до последней загрузки системы.
- today — сегодня.
- yesterday — 1 секунда после полуночи вчерашнего дня.
- this-week — 1 секунда после полуночи 0 (первого) дня текущей недели (определяется вашими региональными настройками).
- week-ago — 1 секунда после полуночи ровно 7 дней назад.

- `this-month` — 1 секунда после полуночи первого дня текущего месяца.
- `this-year` — секунда после полуночи первого января текущего года.

**Ключевые слова для параметра -ts, –start [дата] [время]**:

- `now` — сейчас.
- `recent` — 10 минут назад.
- `boot` — время за секунду до последней загрузки системы.
- `today` — сегодня.
- `yesterday` — 1 секунда после полуночи вчерашнего дня.
- `this-week` — 1 секунда после полуночи 0 (первого) дня текущей недели (определяется вашими региональными настройками).
- `week-ago` — 1 секунда после полуночи ровно 7 дней назад.
- `this-month` — 1 секунда после полуночи первого дня текущего месяца.
- `this-year` — 1 секунда после полуночи первого января текущего года.
- `checkpoint` — `ausearch` будет использовать временную отметку из файла контрольной точки игнорируя при этом идентификатор последнего завершённого события, номер устройства и номер индексного дескриптора (`inode`) файла журнала (см. описание опции `--checkpoint`). По сути это действие для восстановления, если вызов `ausearch --checkpoint` завершился с кодом возврата `10`, `11` или `12` (см. таблицу кодов возврата ниже).

  Эту опцию можно использовать в сценариях командной оболочки следующим образом:

```
ausearch --checkpoint /etc/audit/auditd_checkpoint.txt -i
_au_status=$?
if test ${_au_status} eq 10 -o ${_au_status} eq 11 -o ${_au_
 ↪status} eq 12
then
  ausearch --checkpoint /etc/audit/auditd_checkpoint.txt --
 ↪start checkpoint -i
fi
```

Коды возврата утилиты `ausearch` перечислены в таблице.

Таблица 22 - Коды возврата утилиты `ausearch`

| Код возврата | Описание |
|---|---|
| 0 | Операция выполнена успешно. |
| 1 | Если не найдено событий, соответствующих условию, или возникла ошибка при обработке опций командной строки, или возникли незначительные ошибки доступа/чтения файлов. |
| 10 | В файле контрольной точки обнаружены неверные данные. |
| 11 | Ошибка обработки контрольной точки. |
| 12 | Событие из контрольной точки не найдено в соответствующем файле журнала. |

#### 6.4.3.1. Примеры использования утилиты ausearch

**Поиск по типу события**

Отобразить события безопасности за последние сутки, связанные с входом пользователя в систему:

```
$ sudo ausearch -m USER_LOGIN,LOGIN --start yesterday
----
time->Tue Dec 17 14:38:08 2024
type=LOGIN msg=audit(1734435488.883:195): pid=2132 uid=0
 ↪subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 old-auid=4294967295
 ↪auid=1004 tty=(none) old-ses=4294967295 ses=4 res=1
----
time->Tue Dec 17 14:38:08 2024
type=LOGIN msg=audit(1734435488.907:201): pid=2146 uid=0
 ↪subj=system_u:system_r:init_t:s0 old-auid=4294967295 auid=1004
 ↪tty=(none) old-ses=4294967295 ses=5 res=1
----
time->Wed Dec 18 15:49:57 2024
type=LOGIN msg=audit(1734526197.764:621): pid=256120 uid=0
 ↪subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 old-auid=4294967295
 ↪auid=1000 tty=(none) old-ses=4294967295 ses=6 res=1
----
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
time->Wed Dec 18 15:49:57 2024
type=USER_LOGIN msg=audit(1734526197.821:626): pid=256120 uid=0
 ↪auid=1000 ses=6 subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 msg=
 ↪'op=login id=1000 exe="/usr/sbin/sshd" hostname=? addr=192.168.1.
 ↪4 terminal=/dev/pts/3 res=success'
```

С помощью аргумента `-i` / `--interpret` можно преобразовать числовые идентификаторы пользователей в имена:

```
$ sudo ausearch -m USER_LOGIN,LOGIN --start yesterday -i
----
type=LOGIN msg=audit(12/17/2024 14:38:08.883:195) : pid=2132
 ↪uid=root subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 old-
 ↪auid=unset auid=virtadmin tty=(none) old-ses=4294967295 ses=4
 ↪res=yes
----
type=LOGIN msg=audit(12/17/2024 14:38:08.907:201) : pid=2146
 ↪uid=root subj=system_u:system_r:init_t:s0 old-auid=unset
 ↪auid=virtadmin tty=(none) old-ses=4294967295 ses=5 res=yes
----
type=LOGIN msg=audit(12/18/2024 15:49:57.764:621) : pid=256120
 ↪uid=root subj=system_u:system_r:sshd_t:s0-s0:c0.c1023 old-
 ↪auid=unset auid=virtuser tty=(none) old-ses=4294967295 ses=6
 ↪res=yes
----
type=USER_LOGIN msg=audit(12/18/2024 15:49:57.821:626) : pid=256120
 ↪uid=root auid=virtuser ses=6 subj=system_u:system_r:sshd_t:s0-
 ↪s0:c0.c1023 msg='op=login id=virtuser exe=/usr/sbin/sshd
 ↪hostname=? addr=192.168.1.4 terminal=/dev/pts/3 res=success'
----
type=LOGIN msg=audit(12/18/2024 15:50:36.490:657) : pid=256436
 ↪uid=root subj=system_u:system_r:init_t:s0 old-auid=unset auid=gdm
 ↪tty=(none) old-ses=4294967295 ses=7 res=yes
```

### Поиск по идентификатору пользователя

Отобразить события безопасности за сегодняшний день, связанные с определённым идентификатором или именем пользователя:

```
$ sudo ausearch --uid-all virtadmin --start today
----
time->Wed Dec 18 15:50:16 2024
type=USER_AUTH msg=audit(1734526216.077:641): pid=256200 uid=0
 ↪auid=1004 ses=4 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 msg=
 ↪'op=PAM:authentication grantors=pam_usertype,pam_localuser,pam_
 ↪unix,pam_gnome_keyring acct="virtadmin" exe="/usr/libexec/gdm-
 ↪session-worker" hostname=libvirt.msvsphere.test addr=? terminal=/
 ↪dev/tty1 res=success'
----
time->Wed Dec 18 15:50:16 2024
type=USER_ACCT msg=audit(1734526216.079:642): pid=256200 uid=0
 ↪auid=1004 ses=4 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 msg=
 ↪'op=PAM:accounting grantors=pam_unix,pam_localuser acct=
 ↪"virtadmin" exe="/usr/libexec/gdm-session-worker"
 ↪hostname=libvirt.msvsphere.test addr=? terminal=/dev/tty1
 ↪res=success'
----
time->Wed Dec 18 15:50:16 2024
type=CRED_REFR msg=audit(1734526216.080:643): pid=256200 uid=0
 ↪auid=1004 ses=4 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 msg=
 ↪'op=PAM:setcred grantors=pam_localuser,pam_unix,pam_gnome_keyring
 ↪acct="virtadmin" exe="/usr/libexec/gdm-session-worker"
 ↪hostname=libvirt.msvsphere.test addr=? terminal=/dev/tty1
 ↪res=success'
----
time->Wed Dec 18 15:50:36 2024
type=USER_END msg=audit(1734526236.347:648): pid=2132 uid=0
 ↪auid=1004 ses=4 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 msg=
 ↪'op=PAM:session_close grantors=pam_selinux,pam_loginuid,pam_
 ↪selinux,pam_keyinit,pam_namespace,pam_keyinit,pam_limits,pam_
 ↪systemd,pam_unix,pam_gnome_keyring,pam_umask acct="virtadmin"
 ↪exe="/usr/libexec/gdm-session-worker" hostname=libvirt.msvsphere.
 ↪test addr=? terminal=/dev/tty2 res=success'
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
----
time->Wed Dec 18 15:50:36 2024
type=CRED_DISP msg=audit(1734526236.347:649): pid=2132 uid=0
 ↪auid=1004 ses=4 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 msg=
 ↪'op=PAM:setcred grantors=pam_localuser,pam_unix,pam_gnome_keyring
 ↪acct="virtadmin" exe="/usr/libexec/gdm-session-worker"
 ↪hostname=libvirt.msvsphere.test addr=? terminal=/dev/tty2
 ↪res=success'
----
time->Wed Dec 18 15:50:46 2024
type=LOGIN msg=audit(1734526246.330:677): pid=256830 uid=0
 ↪subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 old-auid=4294967295
 ↪auid=1004 tty=(none) old-ses=4294967295 ses=8 res=1
----
time->Wed Dec 18 15:50:46 2024
type=PROCTITLE msg=audit(1734526246.330:677):
 ↪proctitle=67646D2D73657373696F6E2D776F726B6572205B70616\
D2F67646D2D70617373776F72645D
type=SYSCALL msg=audit(1734526246.330:677): arch=c000003e syscall=1
 ↪success=yes exit=4 a0=9 a1=7ffd6aa865a0 a2=4 a3=3ec items=0
 ↪ppid=1020 pid=256830 auid=1004 uid=0 gid=1004 euid=0 suid=0
 ↪fsuid=0 egid=1004 sgid=1004 fsgid=1004 tty=(none) ses=8 comm=
 ↪"gdm-session-wor" exe="/usr/libexec/gdm-session-worker"
 ↪subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 key=(null)
----
time->Wed Dec 18 15:50:46 2024
type=USER_ROLE_CHANGE msg=audit(1734526246.330:678): pid=256830
 ↪uid=0 auid=1004 ses=8 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023
 ↪msg='op=pam_selinux default-context=unconfined_u:unconfined_
 ↪r:unconfined_t:s0-s0:c0.c1023 selected-context=unconfined_
 ↪u:unconfined_r:unconfined_t:s0-s0:c0.c1023 exe="/usr/libexec/gdm-
 ↪session-worker" hostname=libvirt.msvsphere.test addr=? terminal=/
 ↪dev/tty2 res=success'
----
time->Wed Dec 18 15:50:46 2024
type=USER_START msg=audit(1734526246.352:679): pid=256830 uid=0
 ↪auid=1004 ses=8 subj=system_u:system_r:xdm_t:s0-s0:c0.c1023 msg=
 ↪'op=PAM:session_open grantors=pam_selinux,pam_loginuid,pam_
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪selinux,pam_keyinit,pam_namespace,pam_keyinit,pam_limits,pam_
 ↪systemd,pam_unix,pam_gnome_keyring,pam_umask acct="virtadmin"
 ↪exe="/usr/libexec/gdm-session-worker" hostname=libvirt.msvsphere.
 ↪test addr=? terminal=/dev/tty2 res=success'
```

### Выгрузка данных из журнала

Утилиту `ausearch` можно использовать для выгрузки данных из журнала безопасности для их последующей обработки.

Например, можно экспортировать результаты в формате CSV:

```
$ sudo ausearch --uid-all virtadmin --start today --format csv
NODE,EVENT,DATE,TIME,SERIAL_NUM,EVENT_KIND,SESSION,SUBJ_PRIME,SUBJ_
 ↪SEC,SUBJ_KIND,ACTION,RESULT,OBJ_PRIME,OBJ_SEC,OBJ_KIND,HOW
,USER_AUTH,12/18/2024,15:50:16,641,user-login,4,virtadmin,root,
 ↪privileged-acct,authenticated,success,virtadmin,,user-session,/
 ↪usr/libexec/gdm-session-worker
,USER_ACCT,12/18/2024,15:50:16,642,user-login,4,virtadmin,root,
 ↪privileged-acct,was-authorized,success,virtadmin,,user-session,/
 ↪usr/libexec/gdm-session-worker
,CRED_REFR,12/18/2024,15:50:16,643,user-login,4,virtadmin,root,
 ↪privileged-acct,refreshed-credentials,success,virtadmin,,user-
 ↪session,/usr/libexec/gdm-session-worker
,USER_END,12/18/2024,15:50:36,648,user-login,4,virtadmin,root,
 ↪privileged-acct,ended-session,success,/dev/tty2,,user-session,/
 ↪usr/libexec/gdm-session-worker
,CRED_DISP,12/18/2024,15:50:36,649,user-login,4,virtadmin,root,
 ↪privileged-acct,disposed-credentials,success,virtadmin,,user-
 ↪session,/usr/libexec/gdm-session-worker
,LOGIN,12/18/2024,15:50:46,677,user-login,8,system,root,privileged-
 ↪acct,changed-login-id-to,success,virtadmin,,user-session,
,SYSCALL,12/18/2024,15:50:46,677,audit-rule,8,virtadmin,root,
 ↪privileged-acct,triggered-unknown-audit-rule,success,,,admin-
 ↪defined-rule,/usr/libexec/gdm-session-worker
,USER_ROLE_CHANGE,12/18/2024,15:50:46,678,mac,8,virtadmin,root,
 ↪privileged-acct,changed-role-to,success,unconfined_u:unconfined_
 ↪r:unconfined_t:s0-s0:c0.c1023,,user-session,/usr/libexec/gdm-
 ↪session-worker
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
,USER_START,12/18/2024,15:50:46,679,user-login,8,virtadmin,root,
 ↪privileged-acct,started-session,success,/dev/tty2,,user-session,/
 ↪usr/libexec/gdm-session-worker
```

Также можно экспортировать фрагменты журнала в «сыром» (raw) формате для их последующей обработки с помощью `ausearch` или других утилит для работы с журналами событий безопасности.

Например, следующая команда экспортирует все события входа пользователей в систему за последнюю неделю в файл `weekly-logins.log`:

```
$ sudo ausearch -m LOGIN,USER_LOGIN --start week-ago --raw >
 ↪weekly-logins.log
```

С полученным файлом можно работать как с обычным журналом службы аудита, например выполнить фильтрацию по определённому пользователю и преобразовать вывод в CSV формат:

```
$ ausearch --uid-all 1004 --format csv --input weekly-logins.log
NODE,EVENT,DATE,TIME,SERIAL_NUM,EVENT_KIND,SESSION,SUBJ_PRIME,SUBJ_
 ↪SEC,SUBJ_KIND,ACTION,RESULT,OBJ_PRIME,OBJ_SEC,OBJ_KIND,HOW
,LOGIN,12/16/2024,12:03:51,1269,user-login,10,system,root,
 ↪privileged-acct,changed-login-id-to,success,virtadmin,,user-
 ↪session,
,LOGIN,12/16/2024,12:03:51,1275,user-login,11,system,root,
 ↪privileged-acct,changed-login-id-to,success,virtadmin,,user-
 ↪session,
,LOGIN,12/16/2024,21:11:17,173,user-login,2,system,root,privileged-
 ↪acct,changed-login-id-to,success,virtadmin,,user-session,
,LOGIN,12/16/2024,21:11:17,179,user-login,3,system,root,privileged-
 ↪acct,changed-login-id-to,success,virtadmin,,user-session,
,LOGIN,12/17/2024,14:38:08,195,user-login,4,system,root,privileged-
 ↪acct,changed-login-id-to,success,virtadmin,,user-session,
,LOGIN,12/17/2024,14:38:08,201,user-login,5,system,root,privileged-
 ↪acct,changed-login-id-to,success,virtadmin,,user-session,
,LOGIN,12/18/2024,15:50:46,677,user-login,8,system,root,privileged-
 ↪acct,changed-login-id-to,success,virtadmin,,user-session,
```

### 6.4.4. Резервное копирование журналов событий безопасности

Поскольку журналы событий безопасности хранятся в виде файлов в каталоге `/var/log/audit`, для создания их резервной копии может применяться любой инструмент резервного копирования, поддерживающий работу с файлами. В состав операционной системы включены утилиты `tar`, `rsync` и система резервного копирования `bacula`.

Простой пример создания локальной резервной копии с помощью команды `tar`:

```
$ tar -cjpvf "auditd-logs.$(date --iso-8601).tar.bz2" /var/log/
 ↪audit/audit.log*
```

В результате выполнения команды в текущем каталоге будет создан файл `auditd-logs.YYYY-MM-DD.tar.bz2` где `YYYY` — год, `MM` — месяц и `DD` — сегодняшнее число. В архив будут помещены все файлы из каталога `/var/log/audit`, соответствующие шаблону `audit.log*`.

Однако, при таком подходе есть некоторый риск получить неконсистентную копию файла текущего журнала поскольку, теоретически, в момент копирования в этот файл может происходить запись.

Более надёжным решением будет подать службе аудита сигнал на выполнение принудительной ротации файлов журнала. Получив такой сигнал, служба аудита выполнит следующие действия:

- переименует все имеющиеся копии журнала увеличив число в расширении файла на единицу: `audit.log.1` будет переименован в `audit.log.2`, `audit.log.2` — в `audit.log.3` и т.д.;
- остановит запись в текущий файл журнала `audit.log`;
- переименует текущий файл журнала в `audit.log.1`;
- создаст новый файл журнала `audit.log` и продолжит запись событий уже в него.

Таким образом будет обеспечена консистентность текущего файла журнала при его резервном копировании.

С учётом вышеизложенного, более правильным будет использовать следующий набор команд, оформленный в виде сценария командной оболочки, для создания резервной копии:

```
#!/bin/bash

# завершить работу программы в случае возникновения ошибки
set -e

# сменить текущий каталог на /srv/backup
cd /srv/backup

# выполнить принудительную ротацию журналов службы auditd
auditctl --signal rotate

# создать архив со всеми файлами, соответствующими шаблону
tar -cjpvf "auditd-logs.$(date --iso-8601).tar.bz2" /var/log/audit/
 ↪audit.log.*
```

Автоматизировать создание резервных копий можно с помощью службы периодического выполнения заданий `cron` или таймеров `systemd`. Например, для создания ежедневных архивов вы можете сохранить указанный выше сценарий в файл в каталоге `/etc/cron.daily` (например, `audit-logs-backup.sh`) и сделать его исполняемым:

```
$ sudo chmod +x /etc/cron.daily/audit-logs-backup.sh
```

После этого служба cron будет автоматически выполнять сценарий ежедневно.

Для восстановления можно использовать следующую команду (замените `auditd-logs.2024-12-28.tar.bz2` на реальное имя файла):

```
$ tar -C / -xjpvf auditd-logs.2024-12-28.tar.bz2
```

### 6.4.5. Контроль целостности сведений о событиях безопасности

Целостность сведений о событиях безопасности обеспечивается на уровне ограничения прав доступа в операционной системе: каталог с журналами безопасности `/var/log/audit` и текущий файл журнала `/var/log/audit/audit.log` доступны для чтения и записи только привилегированному пользователю `root` и службе `auditd`. В свою очередь, предыдущие файлы журналов доступны

только для чтения привилегированному пользователю `root`.

Для повышения защищённости системы рекомендуется использовать дополнительные средства контроля целостности, например, утилиту `aide`, которая входит в состав дистрибутива МСВСфера ОС. Документация по установке и настройке `aide` доступна в главе «*9. КОНТРОЛЬ ЦЕЛОСТНОСТИ*».

При изменении прав доступа, владельца и других атрибутов каталога `/var/log/audit` или файла журнала событий безопасности `/var/log/audit/audit.log` команда `aide` выдаст соответствующее предупреждение о нарушении целостности в отчёте:

```
aide --check
Start timestamp: 2025-01-20 06:48:56 +0000 (AIDE 0.16)
AIDE found differences between database and filesystem!!

Summary:
  Total number of entries:      37822
  Added entries:                0
  Removed entries:              0
  Changed entries:              2

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

d   p..      A.. : /var/log/audit
f   ..g      ... : /var/log/audit/audit.log

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

Directory: /var/log/audit
  Perm     : drwx------                        | drwxr-xr-x
  ACL      : A: user::rwx                      | A: user::rwx
           A: group::---                     | A: group::r-x
           A: other::---                     | A: other::r-x

File: /var/log/audit/audit.log
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
  Gid      : 0                                    | 1000


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

/var/lib/aide/aide.db.gz
  MD5      : AABMsEbRMfWShAvA8Yl8kg==
  SHA1     : FssoJmeKJvo7VMc79ZuV1bGgNwI=
  RMD160   : OXPu+xyPaeV6I2c8AOrxwZx8EKA=
  TIGER    : Cez1vaIx/3koN5MbQOwktp/D247COpTa
  SHA256   : B8eWuoZeZ/9PsRrFJIV7lmrXBHo5DPbD
             qaBEO4iea1E=
  SHA512   : xmC1vT9hx9jXmX8NZDbzwUpsaldBbUPj
             F95IEPWxaJn8I3PQnR4G2fZZHnz6mG9G
             OW222CS6V3s2u25O5SqiTQ==


End timestamp: 2025-01-20 06:48:58 +0000 (run time: 0m 2s)
```

### 6.4.6. Оповещение о событиях безопасности

Основной задачей подсистемы аудита является регистрация событий безопасности — автоматическое отслеживаниие таких событий и занесение их в системный журнал.

Оповещение о возникновении событий безопасности и, при необходимости, автоматизированное реагирование на такие события реализуется с помощью дополнительных инструментов. В данном разделе рассмотрены несколько вариантов решения данной задачи.

### 6.4.6.1. Разработка расширений для службы аудита

Служба `auditd` имеет встроенный механизм расширений (плагинов), за счёт которого можно значительно расширить функциональность системы.

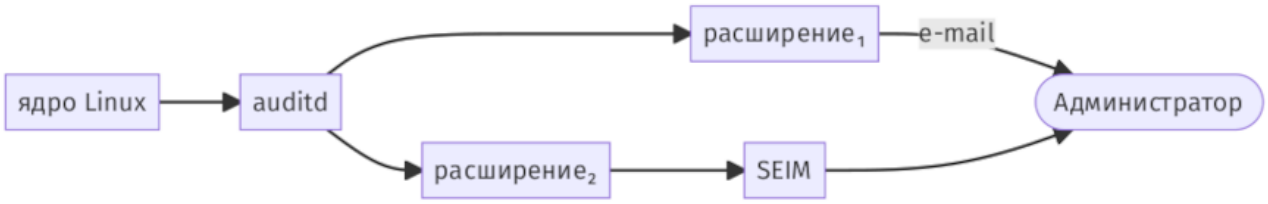



Рис. 23: Механизм расширений (плагинов) службы `auditd`

Например, с помощью расширений можно в реальном времени фильтровать определённые события, оповещать администраторов, загружать информацию о событиях безопасности в другие системы и предпринимать любые другие запрограммированные действия.

Расширение службы аудита представляет собой программу, реализованную на любом языке программирования, которая соответствует следующим требованиям:

- непрерывно получает поток событий на стандартный поток ввода (`stdin`), формат получаемых данных идентичен тому, в котором события записываются в системный журнал событий безопасности;
- обрабатывает системные сигналы `HUP` (обновить настройки из конфигурационного файла) и `TERM` (завершить работу);
- обрабатывает события максимально быстро, не допуская блокировок и переполнения очереди событий, ожидающих отправки со стороны службы `auditd`;
- предоставляет конфигурационный файл, который информирует службу аудита о способе запуска программы-расширения.

В состав операционной системы МСВСфера 9 сертифицированная редакция входят библиотеки для языков программирования Си (пакет *audit-libs-devel*) и Python (пакет *python3-audit*), которые предоставляют набор готовых функций для разбора поступающих данных. В случае с остальными языками программирования вам потребуется разработать соответствующий парсер самостоятельно.

Рассмотрим подход к созданию расширений для службы аудита на примере

создания простого плагина на языке программирования Python, отправляющего уведомления о входе пользователя в систему по электронной почте.

Исходный код расширения с пояснениями:

```
#!/usr/bin/python3 -I
# аргумент -I указывает на необходимость запуска в изолированном
 ↪режиме: не
# выполняется поиск модулей в каталоге с программой и в
 ↪пользовательском
# домашнем каталоге, так же игнорируются все переменные окружения
 ↪PYTHON*.

# импорт необходимых модулей из стандартной библиотеки Python
import datetime
import email.message
import pprint
import signal
import smtplib
import sys

# модуль auparse поставляется в RPM пакете python3-audit и
 ↪предоставляет функции
# для разбора событий, поступающих от службы аудита
import auparse

# исходящий почтовый адрес для отправляемых уведомлений
FROM_EMAIL = 'audit-plugin-notify@localhost'
# список почтовых адресов, на которые необходимо отправлять
 ↪уведомления
ADMIN_EMAILS = ['secadmin@localhost']


def reload_config():
    """
    Функция-заглушка, внутри которой может быть реализована логика
 ↪обработки
    сигнала HUP - обновление настроек из конфигурационного файла.
    """
    pass
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
def notify_user_login(event: dict):
    """
    Функция, которая отправляет электронное письмо с уведомлением о
↪входе
    пользователя в систему.
    """
    # поле "subj" содержит информацию от подсистемы SELinux,
↪благодаря типу
    # можно определить каким образом пользователь вошёл в систему
    se_subj = event['fields'].get('subj', '').split(':')
    if len(se_subj) < 3:
        return
    se_type = se_subj[2]
    if se_type == 'sshd_t':
        # удалённый вход в систему по протоколу SSH
        login_type = 'SSH'
    elif se_type == 'local_login_t':
        # локальный вход в систему через текстовый терминал
        login_type = 'terminal'
    elif se_type == 'xdm_t':
        # вход в систему в графическом режиме
        login_type = 'graphical console'
    else:
        # игнорировать события остального типа, как правило это
↪системные
        # события типа "init_t", которые отображают активность
↪внутренних
        # компонентов системы.
        return
    # сформировать почтовое сообщение и отправить его через
↪локальный сервер
    # электронной почты. В данном случае локальный сервис принимает
↪почту от
    # локальных пользователей без дополнительной аутентификации.
↪Библиотека
    # smptlib также поддерживает аутентификацию и подключение по
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
↪защищённому
    # протоколу SMTPs, так что при необходимости код можно
↪модифицировать для
    # использования любого другого SMTP сервиса для отправки.
    msg_subj = f'User {event["fields"]["auid"]} logged in via
↪{login_type}'
    with smtplib.SMTP('localhost') as smtp:
        msg = email.message.EmailMessage()
        msg['Subject'] = msg_subj
        msg['To'] = ', '.join(ADMIN_EMAILS)
        msg['From'] = FROM_EMAIL
        msg.set_content(pprint.pformat(event))
        smtp.send_message(msg)


def parse_record(parser: auparse.AuParser):
    """
    Функция, которая разбирает отдельную запись события
↪безопасности.
    """
    # метод parser.get_timestamp возвращает объект, который
↪содержит временную
    # метку записи, имя узла и идентификатор события
    event = parser.get_timestamp()
    # преобразовать данные из записи в словарь (хеш-таблицу) чтобы
↪облегчить
    # их последующую обработку
    data = {
        'type': parser.get_type_name(),
        'event': {
            'host': event.host,
            'ts': datetime.datetime.fromtimestamp(event.sec),
            'serial': event.serial
        },
        'fields': {}
    }
    # выполнить обход всех полей записи и записать информацию в
↪словарь
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    # data["fields"]. Метод parser.first_field "передвигает курсор"
↪на первое
    # поле записи
    parser.first_field()
    while True:
        data['fields'][parser.get_field_name()] = parser.interpret_
↪field()
        # метод parser.next_field "передвигает курсор" на следующее
↪поле записи,
        # если полей больше нет, будет возвращено значение False
        if not parser.next_field():
            break
    # вызвать функцию notify_user_login если тип записи - LOGIN
↪(вход
    # пользователя в систему)
    if data['type'] == 'LOGIN':
        notify_user_login(data)


def parse_input_line(parser: auparse.AuParser):
    """
    Функция, которая разбирает загруженную в парсер строку на
↪отдельные события
    и записи, а затем вызывает обработчик для каждой записи
    """
    if not parser.first_record():
        # на вход была получена пустая строка или строка, не
↪содержащая
        # информацию о событиях безопасности - выйти из функции
        return
    while True:
        while True:
            # вызывать функцию parse_record для каждой отдельной
↪записи
            parse_record(parser)
            # метод parser.next_record "передвигает курсор" на
↪следующую запись
            # события, если записей нет, будет возвращено значение
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
↪False - в
            # таком случае необходимо переходить к обработке
↪следующего события
            if not parser.next_record():
                break
        # метод parser.parse_next_event "передвигает курсор" на
↪следующее
        # событие, если событий нет, будет возвращено значение
↪False - в таком
        # случае необходимо завершить обработку текущей строки
        if not parser.parse_next_event():
            break


def main():
    """
    Функция, которая является точкой входа в программу.
    """
    # настройка обработчиков сигналов HUP (перечитать
↪конфигурационный файл) и
    # TERM (завершить работу программы)
    hup_flag = False
    term_flag = False

    def sighup_handler(signal, frame):
        nonlocal hup_flag
        hup_flag = True

    def sigterm_handler(signal, frame):
        nonlocal term_flag
        term_flag = True

    signal.signal(signal.SIGHUP, sighup_handler)
    signal.signal(signal.SIGTERM, sigterm_handler)

    # войти в бесконечный цикл обработки событий
    while True:
        if hup_flag:
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
            # вызвать функцию для обновления настроек программы
↪если получен
            # сигнал HUP
            reload_config()
            hup_flag = False
        elif term_flag:
            # завершить работу программы если получен сигнал TERM
            sys.exit(0)
        else:
            # считать строку, содержащую информацию об одном или
↪нескольких
            # событиях аудита, из потока стандартного ввода stdin
            for line in sys.stdin:
                # инициализировать парсер записей журнала аудита
↪AuParser,
                # поставляемый с библиотекой auparse.
                parser = auparse.AuParser(auparse.AUSOURCE_BUFFER,
↪line)
                # вызвать функцию для разбора строки с информацией
↪о событии
                parse_input_line(parser)


if __name__ == '__main__':
    sys.exit(main())
```

Чтобы служба аудита могла запустить расширение, файл с исполняемым кодом необходимо разместить в каталоге, доступном для чтения службой и сделать этот файл исполняемым. Обычно для этих целей используется каталог `/usr/local/bin`.

В нашем примере код на языке программирования Python не требует компиляции, так что достаточно будет просто сохранить его в файл `/usr/local/bin/audit-plugin-notify`, сделать его исполняемым и установить безопасные права:

```
$ sudo chown root:root /usr/local/bin/audit-plugin-notify
$ sudo chmod 755 /usr/local/bin/audit-plugin-notify
```

Для тестирования и/или отладки расширения нет необходимости сразу подключать его к службе аудита. Поскольку расширение работает с данными в том же формате, в котором они записываются в системный журнал событий безопасности, вы можете использовать для отладки вывод команды `ausearch --raw`:

```
$ sudo ausearch -m LOGIN --raw  | /usr/local/bin/audit-plugin-
 ↪notify
```

Либо перенаправлять последние записи из журнала непосредственно на ввод расширения с помощью команды `tail -F`:

```
$ sudo tail -F /var/log/audit/audit.log | /usr/local/bin/audit-
 ↪plugin-notify
```

Для отправки электронных писем из плагина вам понадобится локальная почтовая служба, например, `postfix`:

```
$ sudo dnf install -y postfix
$ sudo systemctl enable --now postfix
```

Для целей тестирования дополнительная конфигурация почтового сервиса не требуется — достаточно чтобы он был запущен и пользователь, на имя которого будет отправляться электронная почта, существовал (не забудьте внести соответствующие изменения в значение переменной `ADMIN_EMAILS` в исходном коде расширения). Полученная почта будет сохраняться в текстовый файл очереди `/var/spool/mail/ИМЯ_ПОЛЬЗОВАТЕЛЯ`, с которым может работать как почтовый клиент Evolution (тип сервера «Стандартная для Unix очередь типа mbox»), так и любая программа для просмотра текстовых файлов (`less`, `cat` и т.п.). Для получения информации о безопасной настройке почтовой службы для реальных условий вам необходимо обратиться к специализированному руководству.

После того как разработка расширения завершена, его можно подключать к службе аудита. Для этого необходимо создать соответствующий конфигурационный файл плагина в каталоге `/etc/audit/plugins.d`, имя файла может быть любым, но файл должен иметь расширение `.conf`. В нашем примере создадим файл `/etc/audit/plugins.d/audit-plugin-notify.conf` следующего содержания:

```
# включает (yes) или выключает расширение (no)
active = yes

# направление, в котором работает расширение. В настоящий момент
 ↪значение всегда
# должно быть out.
direction = out

# путь к исполняемому файлу расширения.
path = /usr/local/bin/audit-plugin-notify

# для пользовательских расширений, не входящих в поставку службы
 ↪аудита,
# значение type всегда должно быть always.
type = always

# опционально, вы можете передать через службу аудита до двух
 ↪аргументов
# командной строки для запуска расширения. Например, путь к файлу
 ↪или
# идентификатор пользователя. В нашем примере дополнительные
 ↪аргументы не
# требуются.
# args = 1004

# служба аудита может подавать данные на ввод расширения в двух
 ↪форматах:
# в своём внутреннем бинарном (binary) и в текстовом (string).
 ↪Используемый
# нами модуль auparse поддерживает только текстовый формат.
format = string
```

Полная информация по всем опциям конфигурационного файла расширения доступна на странице руководства `man auditd-plugins`.

После создания файла установите для него корректные права:

```
$ sudo chown root:root /etc/audit/plugins.d/audit-plugin-notify.
 ↪conf
$ sudo chmod 644 /etc/audit/plugins.d/audit-plugin-notify.conf
```

Для того чтобы расширение, запущенное службой аудита, могло отправлять почту, потребуется добавить соответствующее разрешение в политики SELinux. Создайте файл `audit-plugin-notify.te` следующего содержания:

```
module audit-plugin-notify 1.0;

require {
        type smtp_port_t;
        type sysctl_net_t;
        type auditd_t;
        class tcp_socket name_connect;
        class dir search;
        class file getattr;
        class file read;
        class file open;
}

#============= auditd_t ==============
allow auditd_t smtp_port_t:tcp_socket name_connect;
allow auditd_t sysctl_net_t:dir search;
allow auditd_t sysctl_net_t:file { getattr open read };
```

Затем скомпилируйте его и создайте SELinux модуль с политикой:

```
$ checkmodule -M -m -o audit-plugin-notify.mod audit-plugin-notify.
 ↪te
$ semodule_package -o audit-plugin-notify.pp -m audit-plugin-
 ↪notify.mod
```

После этого, используйте следующую команду для загрузки SELinux модуля:

```
$ sudo semodule -i audit-plugin-notify.pp
```

Подготовка к запуску расширения службы аудита завершена. Теперь достаточно подать службе сигнал `HUP` чтобы она перечитала свои конфигурационные файлы:

```
$ sudo auditctl --signal reload
```

В случае успешного запуска вы увидите, что служба аудита запустила плагин

в дереве процессов:

```
$ ps axf | grep audit
    45 ?        S      0:00  \_ [kauditd]
   826 ?        S<sl   0:00 /sbin/auditd
 10342 ?        S<     0:00  \_ /usr/bin/python3 -I /usr/local/
↪bin/audit-plugin-notify
```

Если процесс не запустился, то для диагностики вам необходимо будет просмотреть последние записи в журнале сервиса `auditd`:

```
$ sudo journalctl -u auditd
```

Теперь, когда расширение запущено, вы можете войти в систему от имени какого-либо пользователя и убедиться, что вы получили уведомление на почтовый адрес, указанный в переменной `ADMIN_EMAILS`.

Используя механизм расширений, вы можете реализовать любую логику обработки событий безопасности в реальном времени: автоматизировать действия, являющиеся реакцией на определённые события безопасности, отправлять уведомления через корпоративные каналы связи, реализовать интеграцию с системами, которые изначально не поддерживают данные в формате службы `auditd` и т.д.

#### 6.4.6.2. SEIM-системы

SEIM (Security information and event management) система — это комплексная система управления информационной безопасностью. Как правило, такое программное обеспечение имеет клиент-серверную архитектуру и предназначено для централизованного решения широкого круза задач:

- сбор данных о безопасности из различных источников в компьютерной сети: серверов и рабочих станций, сетевых устройств и различного программного обеспечения;
- организация эффективного хранения полученных данных для последующей обработки;
- анализ событий безопасности, выявление потенциальных угроз и связей между различными событиями;
- автоматическая либо автоматизированная реакция на инциденты, связанные с безопасностью: отправка уведомлений, блокировка доступа

и т.п.;

- аудит и отчётность на основе собранных данных.

С точки зрения службы `auditd` интеграция с SEIM-системами обычно реализуется одним из следующих способов:

- настройка подсистемы `auditd` на отправку журналов безопасности в SEIM-систему;
- SEIM-система предоставляет расширение для службы `auditd`, реализующее передачу данных;
- на клиентскую машину устанавливается специальный агент SEIM-системы, который отслеживает появление событий безопасности в системном журнале и отправляет их в SEIM-систему самостоятельно.

Операционная система МСВСфера может выполнять действия, являющиеся реакцией на события безопасности с применением сертифицированных средств защиты информации класса SIEM и систем обнаружения вторжений.

### 6.4.7. Типы записей подсистемы аудита

В таблице представлены некоторые типы записей, генерируемые подсистемой аудита в МСВСфера ОС.

Таблица 23 - Типы записей, генерируемые подсистемой аудита в МСВСфера ОС

| Тип записи | Источник | Описание |
|---|---|---|
| `ACCT_LOCK` | `user` | Пользовательская учётная запись была заблокирована администратором. |
| `ADD_GROUP` | `user` | Добавлена новая пользовательская группа. |
| `ADD_USER` | `user` | Добавлена новая пользовательская учётная запись. |
| `ANOM_ABEND` | `kernel` | Процесс завершился аварийно (segmentation fault и т.п.). |
| `AVC` | `kernel` | Отказ или предоставление разрешений SELinux AVC (Access Vector Cache). |
| `BPF` | `kernel` | Загрузка или выгрузка BPF (Berkeley Packet Filter). |

| Тип записи | Источник | Описание |
|---|---|---|
| CONFIG_CHANGE | user | Конфигурация подсистемы аудита была изменена. |
| CRED_ACQ | user | Пользовательские учётные данные загружены в пользовательское пространство. См. описание функции pam_setcred модуля PAM. |
| CRED_DISP | user | Пользовательские учётные данные выгружены из пользовательского пространства. См. описание функции pam_setcred модуля PAM. |
| CRED_REFR | user | Пользовательские учётные данные были обновлены в пользовательском пространстве. См. описание функции pam_setcred модуля PAM. |
| CRYPTO_KEY_USER | user | Криптографический ключ был использован в криптографических целях. |
| CRYPTO_SESSION | user | Содержит параметры, использованные во время установления TLS-сессии. |
| CWD | kernel | Запись о текущем рабочем каталоге, из которого был запущен процесс, выполнивший системный вызов. |
| DAEMON_CONFIG | user | Конфигурация службы (сервиса) аудита была изменена. |
| DAEMON_START | user | Служба (сервис) аудита была запущена. |
| DEL_GROUP | user | Пользовательская группа была удалена. |
| DEL_USER | user | Пользовательская учётная запись удалена. |
| EXECVE | kernel | Содержит команду запуска процесса, присутствует только в событиях, связанных с системным вызовом execve(2). |
| GRP_MGMT | user | Изменены атрибуты пользовательской группы. |

| Тип записи | Источник | Описание |
|---|---|---|
| KERN_MODULE | kernel | Модуль ядра был загружен или выгружен. |
| LOGIN | user | Содержит информацию о входе пользователя в систему. |
| MAC_CONFIG_CHANGE | user | Был изменён логический переключатель SELinux (SELinux boolean). |
| PATH | kernel | Информация о пути, который был передан системному вызову в качестве аргумента. |
| PROCTITLE | kernel | Содержит полную команду запуска процесса, вызвавшего данное событие безопасности. |
| SERVICE_START | user | Запущена служба (сервис). |
| SERVICE_STOP | user | Остановлена служба (сервис). |
| SOCKADDR | kernel | Содержит информацию о сокете, присутствует только в событиях, связанных с этим сокетом. |
| SOFTWARE_UPDATE | user | Информация об обновлении программного обеспечения. |
| SYSCALL | kernel | Информация о выполненном системном вызове. |
| SYSTEM_BOOT | user | Система была загружена. |
| SYSTEM_RUNLEVEL | user | Изменение уровня выполнения системы (например, через telinit). |
| SYSTEM_SHUTDOWN | user | Система была остановлена. |
| USER_ACCT | user | Обнаружена попытка авторизации пользователя. |
| USER_AUTH | user | Обнаружена попытка аутентификации пользователя. |
| USER_CHAUTHTOK | user | Пароль пользователя был изменён. |

| Тип записи | Источник | Описание |
|---|---|---|
| USER_CMD | user | Команда была запущена из пользовательского пространства. В конфигурации по умолчанию протоколирует только запуск sudo. |
| USER_END | user | Пользовательская сессия была завершена. |
| USER_ERR | user | Ошибка состояния учётной записи пользователя. |
| USER_LOGIN | user | Пользователь вошёл в систему. |
| USER_LOGOUT | user | Пользователь вышел из системы. |
| USER_MGMT | user | Изменение атрибутов пользовательской учётной записи. |
| USER_ROLE_CHANGE | user | SELinux-роль пользователя изменилась. |
| USER_START | user | Запущена пользовательская сессия. |
| VIRT_CONTROL | user | Изменено состояние виртуальной машины (запущена, остановлена и т.д.). |
| VIRT_MACHINE_ID | user | Виртуальной машине назначен контекст безопасности SELinux. |
| VIRT_RESOURCE | user | Виртуальной машине был назначен (выделен) какой-либо ресурс. |

В столбце «Источник» применяются следующие сокращения:

- user — источником события является пользовательское пространство;
- kernel — источником события является пространство ядра.

# 7. ОГРАНИЧЕНИЕ ПРОГРАММНОЙ СРЕДЫ

## 7.1. Введение

Средства ограничения программной среды предоставляют возможности установки программного обеспечения доверенным образом; применения типовых наборов различных программных конфигураций; управления запуском программного обеспечения, в том числе определения запускаемых программ, настройки параметров запуска и контроля за их запуском; реагирования на попытки запуска, произведенные в нарушение установленных правил, а также другие возможности.

## 7.2. Включение программ в автозагрузку

Утилита `chkconfig` позволяет включать программы в автозагрузку с целью их автоматического запуска при старте операционной системы. Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 24 - Опции утилиты `chkconfig` и их значения

| Опция | Значение |
|---|---|
| `--level levels` | Определяет уровни, на которых соответствующая программа должна выполняться. Уровни указываются на месте параметра `levels` в качестве строки целочисленных значений в диапазоне от 0 до 6. Так, например, при передаче опции `--level 35` утилите будет передано указание на уровни 3 и 5 соответственно. |
| `--add name` | Добавляет новую службу для управления утилитой `chkconfig`. Имя службы указывается на месте параметра `name`. |
| `--override name` | Производит переопределение настроек службы, имя которой указывается на месте параметра `name`, вместо базовых настроек. |

| Опция | Значение |
|---|---|
| `--no-redirect` | Если утилита запущена в системе, использующей утилиту `systemd` в качестве системы инициализации, то `chkconfig` будет перенаправлять команды в `systemd`, если у данной службы существует соответствующий файл, предназначенный для таких обращений. Данная опция отключает процесс перенаправления утилите `systemd` и обеспечивает работу только с символьными ссылками в директориях `/etc/rc[0-6].d`. |
| `--del name` | Удаляет службу, имя которой указывается на месте параметра `name`, из-под управления утилитой `chkconfig`. Также из директорий `/etc/rc[0-6].d` удаляются любые символьные ссылки, указывающие на удаляемую службу. |
| `--list name` | Выводит все службы, доступные для `chkconfig`, а также показывает их статус на каждом уровне (вкл/выкл). Если опции передать аргументом имя некоторой службы, которое указывается на месте параметра `name`, то будет выведена информация только об указанной службе. |

### 7.3. Управление системными службами

Утилита `systemctl` позволяет управлять системными службами. Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 25 - Опции утилиты `systemctl` и их значения

| Опция | Значение |
|---|---|
| `start [имя сервиса]` | Запускает работу сервиса с указанным именем. |
| `stop [имя сервиса]` | Останавливает работу сервиса с указанным именем. |

| Опция | Значение |
|---|---|
| `-t, --type` | Указывает на тип так называемого юнита (службы, сокета, устройства и т.п.). Может быть в виде списка наименований типов, разделенных запятой, если требуется указать более, чем на один тип. |
| `-a, --all` | При выведении списка юнитов вывести абсолютно все загруженные юниты вне зависимости от их статуса, включая те из них, которые являются неактивными. |
| `reload [имя сервиса]` | Перезагружает конфигурацию сервиса с указанным именем. |
| `restart [имя сервиса]` | Перезапускает сервис с указанным именем. |
| `try-restart [имя сервиса]` | Перезапускает сервис с указанным именем, если данный сервис уже работает на момент запуска утилиты. |
| `reload-or-restart [имя сервиса]` | Перезагрузить конфигурацию сервиса с указанным именем, если сервис поддерживает такую команду, или выполнить перезапуск службы. Если на момент запуска утилиты указанная служба не была запущена, то она запустится после успешного выполнения команды. |
| `reload-or-try-restart [имя сервиса]` | Перезагрузить конфигурацию сервиса с указанным именем, если сервис поддерживает такую команду, или выполнить перезапуск службы. Если на момент запуска утилиты указанная служба не была запущена, то указанная команда не произведет никаких действий. |
| `kill [имя сервиса]` | Осуществить принудительную остановку работы службы с указанным именем. |

| Опция | Значение |
|---|---|
| `is-active [имя сервиса]` | Осуществляет проверку, активна ли на момент запуска утилиты служба с указанным именем. Если служба активна, или хотя бы одна из служб, переданных в качестве аргумента данной команде, активна (в случае, если были переданы наименования более, чем одной службы), выведется нулевое значение. В противном случае — ненулевое. |
| `is-failed [имя сервиса]` | Проверяет, были ли проблемы при запуске указанной службы или служб. Если хотя бы у одной из служб возникали проблемы, будет выведено нулевое значение. |
| `enable [имя сервиса]` | Добавляет указанный сервис (или их множество) в автозапуск. |
| `disable [имя сервиса]` | Убирает указанный сервис (или их множество) из автозапуска. |
| `is-enabled [имя сервиса]` | Проверяет, находится ли указанная служба (или службы, в случае, если в качестве аргумента был передан список наименований) в автозапуске. Если хотя бы одна из указанных служб находится в автозапуске, будет выведено нулевое значение. |
| `--version` | Вывести информацию о версии утилиты. |
| `-h, --help` | Вывести справочную информацию об утилите. |

**Пример**: проверим статус сервера печати.

Для этого выполним следующую команду:

```
$ sudo systemctl status cups
cups.service - CUPS Printing Service
  Loaded: loaded (/usr/lib/systemd/system/cups.service; disabled;
 ↪vendor preset: enabled)
  Active: inactive (dead)
```

**Пример**: разрешим автоматический запуск сервера печати CUPS при

загрузке системы.

Для этого выполним следующую команду:

```
$ sudo systemctl enable cups
Created symlink from /etc/systemd/system/multi-user.target.wants/
 ↪cups.service to /usr/lib/systemd/system/cups.service.
Created symlink from /etc/systemd/system/printer.target.wants/cups.
 ↪service to /usr/lib/systemd/system/cups.service.
Created symlink from /etc/systemd/system/sockets.target.wants/cups.
 ↪service to /usr/lib/systemd/system/cups.socket.
Created symlink from /etc/systemd/system/multi-user.target.wants/
 ↪cups.path to /usr/lib/systemd/system/cups.path.
```

### 7.4. Настройка запуска программ по расписанию

Утилита `crontab` позволяет настраивать запуск программ по расписанию.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 26 - Опции утилиты `crontab` и их значения

| Опция | Значение |
|---|---|
| `-u` | Указывает пользователя, чье расписание должно редактироваться. |
| `-l` | Вывод текущего файла расписания. |
| `-r` | Удаление текущего файла расписания. |
| `-e` | Редактирование файла расписания. |

Таблица расписания состоит из шести колонок, разделяемых пробелами или символами табуляции. Первые пять колонок задают время выполнения (минута, час, день, месяц, день недели). В них может находиться число, список чисел, разделённых запятыми, диапазон чисел, разделённых дефисом, символы `*` или `/`. После полей времени указывается пользователь, от которого запускается программа. Все остальные символы в строке интерпретируются как выполняемая программа с её параметрами.

**Пример**: установим с помощью утилиты `crontab` ограничения на доступ к системе по времени, с 10:28 до 10:30. Команда `passwd -l user2` блокирует

возможность авторизации, дописывая символ восклицательного знака к строке пароля в файле `/etc/shadow`. Команда `passwd -u user2` производит обратную операцию, снимая тем самым блокировку. Заполним файл расписания и выполним команду `service crond restart`:

```
$ sudo crontab -e
crontab: Installing new crontab

$ sudo service crond restart
Redirecting to /bin/systemctl restart crond.service

$ sudo crontab -l
28 10 * * * /usr/bin/passwd -l user2
30 10 * * * /usr/bin/passwd -u user2
```

### 7.5. Управление программными пакетами

Утилита `rpm` позволяет управлять так называемыми программными пакетами, т.е. управлять их установкой, обновлением, проверкой и удалением.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 27 - Опции утилиты `rpm` и их значения

| Опция | Значение |
|---|---|
| `-i, --install` | Установка нового пакета. |
| `-u, --upgrade` | Установка или обновление уже установленного пакета до новой версии. При этом после установки пакета все другие версии удаляются. |
| `-f, --freshen` | Обновление пакета, но только если предыдущая версия уже установлена. |
| `--nodeps` | Не выполнять проверку зависимостей перед установкой или обновлением пакета. |
| `--nosuggest` | Не предлагать пакет(ы) для разрешения отсутствующих зависимостей. |

| Опция | Значение |
|---|---|
| `--noorder` | Не выполнять переупорядочивание пакетов для установки. Список пакетов обычно переупорядочивается для удовлетворения зависимостей. |
| `--oldpackage` | Разрешает обновить или заменить пакет более старой версией. |
| `--replacefiles` | Установить пакеты, даже если они заменяют файлы от других установленных пакетов. |
| `--replacepkgs` | Установить пакеты, даже если они уже установлены в систему. |
| `--includedocs` | Устанавливать файлы с документацией. |
| `--excludedocs` | Не устанавливать файлы с документацией. |
| `-e, --erase` | Удалить заданный пакет. |
| `--allmatches` | Удалить все версии пакета. |
| `--nodeps` | Не проверять зависимости перед удалением пакетов. |
| `--test` | Выполнить только проверку установки пакета. |
| `-q, --query` | Вывести информацию о пакете. |
| `-a, --all` | Выполняет запрос ко всем установленным пакетам. |
| `--changelog` | Вывести информацию об изменениях в пакете. |
| `-l, --list` | Вывести список файлов в пакете. |
| `-P, --provides` | Вывести функциональность, предоставляемую пакетом. |
| `-R, --requires` | Вывести пакеты, от которых зависит этот пакет. |
| `-v, --verify` | Выполнить проверку метаданных пакета и его контрольной суммы. |
| `--version` | Вывести номер версии утилиты. |
| `--help` | Вывести справку об использовании утилиты. |

### 7.6. Установка последней версии пакета/группы пакетов

Утилита `dnf` используется для установки последней версии пакета или группы пакетов с учетом существующих зависимостей.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 28 - Опции утилиты `dnf` и их значения

| Опция | Значение |
|---|---|
| `install` | Используется для установки последней версии пакета с учетом существующих зависимостей. |
| `reinstall` | Используется для переустановки пакета с идентичной версией. |
| `update` | Используется для обновления всех пакетов в системе. |
| `download` | Используется для загрузки пакета из репозитория. |
| `downgrade` | Используется для понижения версии пакета с версии, установленной на данный момент, до предыдущей самой высокой версии или указанной версии. |
| `remove` | Используется для удаления указанных пакетов из системы, а также для удаления пакетов, зависящих от удаляемых пакетов. |
| `info` | Используется для вывода описаний и общей информации о доступных пакетах. |
| `search` | Используется для поиска пакетов. |
| `list` | Используется для вывода различной информации о доступных пакетах. |
| `repolist all` | Используется для вывода списка всех репозиториев. |
| `clean` | Используется для удаления различных данных, накапливающихся со временем в кэше утилиты. |
| `history` | Используется для вывода истории использования утилиты. |

| Опция | Значение |
|---|---|
| `groupinstall` | Используется для установки последней версии всех пакетов из группы с учетом существующих зависимостей. |
| `groupupdate` | Используется для обновления всех пакетов из группы. |
| `groupremove` | Используется для удаления всех пакетов из группы. |
| `groupinfo` | Используется для вывода списка пакетов, относящихся к группе. |
| `grouplist` | Используется для вывода имен всех существующих групп пакетов. |
| `provides` | Используется, чтобы выяснить, какой пакет предоставляет тот или иной файл. |
| `repoquery --requires` | Вывести зависимости неустановленного пакета. |
| `repoquery --requires --resolve` | Вывести список пакетов, которые необходимы для удовлетворения зависимостей. |
| `-v, --verbose` | Запустить с большим количеством отладочной информации. |
| `-d, --debuglevel` | Устанавливает уровень отладки. |
| `-h, --help` | Вывести справку и выйти. |

# 8. СТИРАНИЕ ДАННЫХ

## 8.1. Введение

Средства стирания данных предоставляют возможности безвозвратного удаления ставших ненужными данных и обеспечения недоступности остаточной информации путем многократной перезаписи использованных мест памяти специальными последовательностям.

## 8.2. Заполнение случайными числами места, занятого файлами

Утилита `shred` позволяет заполнять случайными числами место, занятое файлами.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 29 - Опции утилиты `shred` и их значения

| Опция | Значение |
|---|---|
| `-f, --force` | Изменить права для разрешения записи, если это необходимо. |
| `-n, --iterations=N` | Перезаписать файл N раз вместо 3-х по умолчанию. |
| `--random-source=FILE` | Перезаписать файл случайными данными, взятыми из файла с именем `FILE`. |
| `-s, --size=N` | Перезаписать только N байт. Можно использовать суффиксы `K`, `M`, `G` для указания размерности: килобайт, мегабайт, гигабайт. |
| `-u, --remove` | Обрезать и удалить файл после перезаписи. По умолчанию файлы не удаляются. |
| `-v, --verbose` | Показывать ход выполнения. |
| `-x, --exact` | Не округлять размер файла до следующего целого блока. |
| `-z, --zero` | На последней итерации перезаписать файл нулями. |
| `--version` | Показать версию утилиты и выйти. |
| `--help` | Показать справку и выйти. |

**Пример**: заполним место, занятое файлом `filename`, с последующим

удалением файла.

Для этого выполним следующую команду:

```
$ sudo shred -u -z filename
```

### 8.3. Стирание данных в свободном пространстве раздела, в котором находится директория

Утилита `sfill` позволяет стирать данные в свободном пространстве раздела, в котором находится заданная директория. Стирание производится в четыре шага:

1. Однократная перезапись числами 255 (0xFF).
2. Пятикратная перезапись случайными числами.
3. Двадцатисемикратная перезапись специальными числами.
4. И еще один раз пятикратная перезапись случайными числами.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 30 - Опции утилиты `sfill` и их значения

| Опция | Значение |
|---|---|
| `-f` | Выполнение более быстрым образом за счет пропуска второго и четвертого шагов перезаписи случайными числами. |
| `-i` | Очистка свободного пространства только индексного дескриптора, но не свободного пространства жесткого диска. |
| `-I` | Очистка свободного пространства только жесткого диска без затрагивания свободного пространства индексного дескриптора. |
| `-l` | Выполнение более быстрым образом за счет пропуска третьего и четвертого шагов или путем выполнения только одного шага перезаписи данных нулевыми значениями, если эту опцию задать дважды (например, `sdmem -l -l`). |

| Опция | Значение |
|---|---|
| -v | Работа будет сопровождаться выводом динамической строки, показывающей прогресс её выполнения. |
| -z | На четвертом шаге вместо перезаписи случайными числами выполнять перезапись нулями. |

**Пример**: выполним очистку свободного пространства.

Для этого выполним следующую команду:

```
$ sudo sfill -vz /mnt/
Using /dev/urandom for random input.
Wipe mode is secure (38 special passes)
Wiping now ...
Creating /mnt/00000000.000 ... ************************** Wiping
 ↪inodes ...
Done ... Finished
```

### 8.4. Стирание данных в разделах подкачки

Утилита `sswap` позволяет стирать данные в разделах подкачки. Алгоритм стирания данных абсолютно такой же, как и у утилиты `sfill`.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 31 - Опции утилиты `sswap` и их значения

| Опция | Значение |
|---|---|
| -f | Выполнение более быстрым образом за счет пропуска второго и четвертого шагов перезаписи случайными числами. |

| Опция | Значение |
|---|---|
| `-l` | Выполнение более быстрым образом за счет пропуска третьего и четвертого шагов или путем выполнения только одного шага перезаписи данных нулевыми значениями, если эту опцию задать дважды. |
| `-v` | Работа будет сопровождаться выводом динамической строки, показывающей прогресс её выполнения. |
| `-z` | На четвертом шаге вместо перезаписи случайными числами выполнять перезапись нулями. |

## 8.5. Стирание данных в оперативной памяти

Утилита `sdmem` позволяет стирать данные в оперативной памяти. Алгоритм стирания данных почти такой же, как и у утилиты `sfill`, но с тем отличием, что на первом шаге однократная перезапись производится числами 0 (0x00).

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 32 - Опции утилиты `sswap` и их значения

| Опция | Значение |
|---|---|
| `-f` | Выполнение более быстрым образом за счет пропуска второго и четвертого шагов перезаписи случайными числами. |
| `-l` | Выполнение более быстрым образом за счет пропуска третьего и четвертого шагов или путем выполнения только одного шага перезаписи данных нулевыми значениями, если эту опцию задать дважды. |
| `-v` | Работа будет сопровождаться выводом динамической строки, показывающей прогресс её выполнения. |

# 9. КОНТРОЛЬ ЦЕЛОСТНОСТИ

## 9.1. Контроль целостности установленных RPM-пакетов

В процессе установки RPM-пакетов пакетный менеджер сохраняет в свою внутреннюю базу данных различную информацию о файлах и каталогах, которые входят в состав пакета: права доступа, сведения о владельце, размер, контрольную сумму и т.д.

Используя эти данные, команда `rpm --verify` позволяет проверить целостность установленных RPM-пакетов. Для вызова команды используется следующий синтаксис:

```
rpm {--verify|-V} <пакет | параметры_выборки> [параметры_проверки]
```

Где:

- опция `--verify` или её краткая форма `-V` переводит пакетный менеджер `rpm` в режим проверки целостности;
- `пакет` — название RPM-пакета, также можно использовать подробный формат `название[-версия[-релиз]][.архитектура]` (примеры: `bash-5.1.8`, `bash-5.1.8-9.el9`, `bash-5.1.8-9.el9.x86_64`).

  В качестве альтернативы указанию имени пакета можно использовать большинство опций `rpm`, предназначенных для поиска/фильтрации пакетов. Полный список поддерживаемых опций доступен в соответствующей документации (`man 8 rpm`), а в данном руководстве рассмотрим две опции, которые представляют наибольший интерес в контексте верификации пакетов.

  - `-a, --all [условие]` — выполнить проверку всех установленных RPM-пакетов, либо всех пакетов, соответствующих `условию`, если оно определено. `Условие` определяется в формате `тег=шаблон`, например можно использовать команду `rpm -V -a name=krb5*` для проверки всех пакетов, имя которых начинается с `krb5`. Существует возможность задать несколько условий, используя несколько аргументов `-a` — в таком случае будут обработаны только те пакеты, которые соответствуют всем заданным критериям.
  - `-f, --file <файл>` — выполнить проверку пакета, которому принадлежит указанный файл.

- по умолчанию команда `rpm --verify` выполняет все проверки пакета, но с помощью следующих параметров проверки можно отключить те или иные тесты:
    - `--nodeps` — не выполнять проверку зависимостей пакетов;
    - `--nodigest` — не выполнять проверку контрольных сумм пакета и/или его заголовков;
    - `--nofiles` — отключить проверку атрибутов файлов;
    - `--noscripts` — не выполнять секцию `%verifyscript` RPM-пакета если она определена;
    - `--nosignature` — не проверять подписи пакета и его заголовков;
    - `--nolinkto` — не проверять атрибуты ссылок;
    - `--nofiledigest` — не проверять контрольные суммы файлов пакета;
    - `--nosize` — не проверять размер файла;
    - `--nouser` — не выполнять проверку на изменение пользователя владельца файла;
    - `--nogroup` — не выполнять проверку на изменение группы владельца файла;
    - `--nomtime` — не проверять время последнего изменения файла;
    - `--nomode` — не проверять права доступа к файлу;
    - `--nordev` — не проверять атрибут `rdev` (тип устройства) для файлов устройств;
    - `--nocaps` — не проверять разрешения (capabilities) файла.

По умолчанию команда `rpm --verify` выводит на консоль построчный список файлов, для которых как минимум одна из проверок завершилась неудачно. Используется следующий формат вывода:

```
.........   [тип] путь_к_файлу
```

Строка начинается с девяти ячеек, каждая из которых отображает статус определённой проверки. Ячейка может принимать значение одного из следующих типов:

- `.` — означает, что проверка пройдена успешно;
- `?` — означает, что проверку не удалось выполнить по каким-то причинам (например, отсутствуют права на чтение файла);

- один из указанных ниже символов, в таком случае это означает, что соответствующая проверка завершилась неудачей:
    - `S` — размер файла отличается;
    - `M` — права доступа или тип файла отличаются;
    - `5` — контрольная сумма файла не соответствует эталонной;
    - `D` — старший (major) или младший (minor) номер устройства отличается;
    - `L` — путь, на который ссылается ссылка, не соответствует ожидаемому;
    - `U` — отличается пользователь — владелец файла;
    - `G` — отличается группа — владелец файла;
    - `T` — отличается время последнего изменения файла;
    - `P` — разрешения (capabilities) файла не соответствуют ожидаемым.

В случае если проверяемый файл является специальным с точки зрения пакетного менеджера RPM, после девяти ячеек со статусом будет указан тип файла.

- `c` — конфигурационный файл (перечислен в блоке `%config` spec-файла RPM-пакета);
- `d` — файл с документацией (перечислен в блоке `%doc` spec-файла);
- `g` — так называемый «призрачный» файл (перечислен в блоке `%ghost` spec-файла), это означает что содержимое файла не является частью данного пакета;
- `l` — файл с лицензионным соглашением (перечислен в блоке `%license` spec-файла);
- `r` — файл README (перечислен в блоке `%readme` spec-файла).

Для обычных файлов или каталогов тип не указывается.

В конце строки вывода находится путь к файлу, который не прошёл проверку.

Если все пакеты и включённые в них файлы прошли проверку, команда `rpm --verify` вернёт код возврата `0`, в противном случае код возврата будет ненулевым.

Далее, рассмотрим несколько реальных примеров работы с утилитой.

- Успешная проверка пакета `bash`:

```
$ sudo rpm -V bash

$ echo $?
0
```

Изменений не обнаружено, код возврата — `0`.

- Проверка всех пакетов, имена которых начинаются с `krb5`, обнаруживает изменения в файле `/etc/krb5.conf`:

```
$ sudo rpm -V -a krb5*
S.5......  c /etc/krb5.conf
S.5......  c /etc/krb5.conf

$ echo $?
1
```

Статус `S` в первой ячейке означает, что фактический размер файла отличается от ожидаемого, а статус `5` в третьей ячейке — что контрольная сумма файла отличается от эталонной. Данный файл является конфигурационным (статус `c` перед именем файла) и обнаруженное несоответствие считается нормальным, если вы изменяли файл в процессе настройки системы. Поскольку один из файлов был изменён, код возврата ненулевой. Проверка выполняется последовательно для каждого отдельного пакета, подходящего под условие. В этом примере файл `/etc/krb5.conf` принадлежит как 32-битному, так и 64-битному варианту пакета `krb5-libs` и, соответственно, информация об этом файле выводится два раза:

```
$ rpm -qf /etc/krb5.conf
krb5-libs-1.21.1-4.el9_5.x86_64
krb5-libs-1.21.1-4.el9_5.i686
```

## 9.2. Программа для контроля целостности AIDE

`aide` (Advanced Intrusion Detection Environment) — это программа для проверки целостности файлов.

Принцип работы данного инструмента заключается в следующем: по требованию администратора *aide* создаёт базу данных, которая содержит различную информацию о файлах в системе и при последующих запусках утилиты выполняется проверка текущего состояния отслеживаемых файлов на предмет соответствия эталонному. В случае выявления отклонений генерируется соответствующий отчёт.

Утилита `aide` обладает следующими возможностями:

- поддерживает хранение в БД и отслеживание изменения различных атрибутов файлов: тип файла, права доступа, номер индексного дескриптора (inode), владельца и группу файла, размер, время последнего изменения файла (mtime), время последнего изменения его метаданных/содержимого (ctime), время последнего доступа к файлу (atime), количество ссылок на файл и т.п.
- создаёт и проверяет контрольные суммы с использованием различных хеш-функций: SHA256, SHA512, SHA1, MD5 и т.д.
- отслеживает изменение расширенных атрибутов файлов: Posix ACL, SELinux, xattr, e2fsattrs.
- автоматический запуск через `systemd` таймеры или `cron` с отправкой уведомлений по электронной почте.

### 9.2.1. Установка и первичная настройка aide

Для установки `aide` выполните следующую команду:

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

В конфигурации по умолчанию `aide` проверяет лишь некоторый набор файлов и каталогов, определённый в конфигурационном файле `/etc/aide.conf`. Для отслеживания изменения в других файлах и каталогах вам потребуется внести соответствующие правки в конфигурационный файл перед инициализацией базы данных `aide`. Полная информация о настройке программы находится на соответствующей странице документации `man aide.conf`.

Для инициализации базы данных выполните следующую команду:

```
$ sudo aide --init
Start timestamp: 2025-01-16 12:54:34 +0300 (AIDE 0.16)
AIDE initialized database at /var/lib/aide/aide.db.new.gz

Number of entries:      204419

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

/var/lib/aide/aide.db.new.gz
  MD5      : bJRt0P9rQ3lR7Wq4SZJlsw==
  SHA1     : /nM1abzKvpoUSt22HiuKAsx7zE4=
  RMD160   : Fw7M3O5xq55CKfXXuJgeQ9bn8aE=
  TIGER    : cJJxmyGjwuecrEIUNhJgsrRApqQmHNv6
  SHA256   : iUpCg8pVKy6a34FdCNurMUL04BtV65sD
             8k9pgX1mSr0=
  SHA512   : /+DHSwBp2kcMHvO5HXk3FQvLbvZUO1xj
             6+YDcT4kXyUcAekf46zoEKdDVd89AX8S
             H3Xh7CRxF2uQ4wFTrFv8Gg==


End timestamp: 2025-01-16 12:55:16 +0300 (run time: 0m 42s)
```

При запуске с ключом `--init` утилита `aide` создаёт новую базу данных в файле `/var/lib/aide/aide.db.new.gz`, заданном директивой `database_out` в конфигурационном файле. Но для проверки целостности системы используется база из файла `/var/lib/aide/aide.db.gz`, путь к которому задан директивой `database`. Соответственно, для использования созданной базы данных необходимо переименовать файл:

```
$ sudo mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

### 9.2.2. Проверка целостности системы с помощью aide

Команда `aide --check` выполняет проверку целостности системы и генерирует соответствующий отчёт:

```
$ aide --check
Start timestamp: 2025-01-17 00:16:58 +0300 (AIDE 0.16)
AIDE found differences between database and filesystem!!

Summary:
  Total number of entries:      204420
  Added entries:                1
  Removed entries:              0
  Changed entries:              2

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

d++++++++++++++++: /root/.config/procps

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

f   ...    .C... : /etc/cups/subscriptions.conf
f   ...    .C... : /etc/cups/subscriptions.conf.O

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

File: /etc/cups/subscriptions.conf
  SHA512   : iY//ZXGfIOCw5w+dsglTJ5uanAu6ycGo |
 ↪gw0CO7RN6yjIbRHvTVjthdZyfI/igk7K
             SU5HoonBNSbGDb2sNVZLX/olLvPHyPDo |
 ↪ZGoICuAK4CEZHcoTQTMPo0RSNRUqWXz0
             qGGLZHHx35ce7yFxVxmjdQ==         |
 ↪el6dW9hvpSltkjJxywX3RQ==
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
File: /etc/cups/subscriptions.conf.O
  SHA512   : HwYU5Ddhxs+MTd+7h67ToWOc1njts3eu |
 ↪MZOiOFpYIbRYnZECCA+Dq9UZCoBlTO/n
             Lro+IZrhTR+yVA0W85Ji0cKP/77ZdqFS | 0ELsf8rnHmL/
 ↪pp6uz2GgJHURnrHclo0h
             7FRt4bCEB1L1T8SNe8Nonw==         |
 ↪l4JpCwIjfXhWANs3mkkzzg==


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

/var/lib/aide/aide.db.gz
  MD5      : bJRt0P9rQ3lR7Wq4SZJlsw==
  SHA1     : /nM1abzKvpoUSt22HiuKAsx7zE4=
  RMD160   : Fw7M3O5xq55CKfXXuJgeQ9bn8aE=
  TIGER    : cJJxmyGjwuecrEIUNhJgsrRApqQmHNv6
  SHA256   : iUpCg8pVKy6a34FdCNurMUL04BtV65sD
             8k9pgX1mSr0=
  SHA512   : /+DHSwBp2kcMHvO5HXk3FQvLbvZUO1xj
             6+YDcT4kXyUcAekf46zoEKdDVd89AX8S
             H3Xh7CRxF2uQ4wFTrFv8Gg==


End timestamp: 2025-01-17 00:17:18 +0300 (run time: 0m 20s)
```

В данном примере, в системе были обнаружены следующие изменения относительно эталонного состояния из базы данных:

- был создан каталог `/root/.config/procps`;
- был изменён файл `/etc/cups/subscriptions.conf`;
- был изменён файл `/etc/cups/subscriptions.conf.O`.

### 9.2.3. Обновление базы данных aide

Для обновления базы данных `aide` после внесения изменений в систему используйте команду `aide --update`:

```
$ sudo aide --update
Start timestamp: 2025-01-17 00:27:30 +0300 (AIDE 0.16)
AIDE found differences between database and filesystem!!
New AIDE database written to /var/lib/aide/aide.db.new.gz

Summary:
  Total number of entries:      204420
  Added entries:                1
  Removed entries:              0
  Changed entries:              2

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

d++++++++++++++++: /root/.config/procps

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

f   ...    .C... : /etc/cups/subscriptions.conf
f   ...    .C... : /etc/cups/subscriptions.conf.O

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

File: /etc/cups/subscriptions.conf
  SHA512   : iY//ZXGfIOCw5w+dsglTJ5uanAu6ycGo |
 ↪gw0CO7RN6yjIbRHvTVjthdZyfI/igk7K
             SU5HoonBNSbGDb2sNVZLX/olLvPHyPDo |
 ↪ZGoICuAK4CEZHcoTQTMPo0RSNRUqWXz0
             qGGLZHHx35ce7yFxVxmjdQ==         |
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪el6dW9hvpSltkjJxywX3RQ==

File: /etc/cups/subscriptions.conf.O
  SHA512   : HwYU5Ddhxs+MTd+7h67ToWOc1njts3eu |
 ↪MZOiOFpYIbRYnZECCA+Dq9UZCoBlTO/n
             Lro+IZrhTR+yVA0W85Ji0cKP/77ZdqFS | 0ELsf8rnHmL/
 ↪pp6uz2GgJHURnrHclo0h
             7FRt4bCEB1L1T8SNe8Nonw==         |
 ↪l4JpCwIjfXhWANs3mkkzzg==


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

/var/lib/aide/aide.db.gz
  MD5      : bJRt0P9rQ3lR7Wq4SZJlsw==
  SHA1     : /nM1abzKvpoUSt22HiuKAsx7zE4=
  RMD160   : Fw7M3O5xq55CKfXXuJgeQ9bn8aE=
  TIGER    : cJJxmyGjwuecrEIUNhJgsrRApqQmHNv6
  SHA256   : iUpCg8pVKy6a34FdCNurMUL04BtV65sD
             8k9pgX1mSr0=
  SHA512   : /+DHSwBp2kcMHvO5HXk3FQvLbvZUO1xj
             6+YDcT4kXyUcAekf46zoEKdDVd89AX8S
             H3Xh7CRxF2uQ4wFTrFv8Gg==

/var/lib/aide/aide.db.new.gz
  MD5      : 9XQcGyeUGCo4jQFqKNWCSw==
  SHA1     : JT98QJxegb+XsjzCHB5sbaFpsoQ=
  RMD160   : t8pAZIn/4MsB+YiYi7wr1uie4iY=
  TIGER    : oddP+JHtsDFFr+9GeCymlZ7meJV0K5uI
  SHA256   : SB9BYNaOzq8f5QYMbkNzTZ78yMWyp0lF
             UUbNH8Q04Ig=
  SHA512   : WeutFJZKQQydQkBSGvsuyC/ASE54vOcP
             NnrJlj7PjjMXINFpQvIhgQvE+LwtDFjq
             Kj9/MeYhjHQchYHNwhmotw==
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
End timestamp: 2025-01-17 00:27:53 +0300 (run time: 0m 23s)
```

Как и в случае с `aide --init`, данная команда создаст новый файл базы данных `/var/lib/aide/aide.db.new.gz`, а также сгенерирует отчёт об изменениях в системе относительно предыдущего состояния базы данных.

Поскольку на данном этапе у вас есть и новое, и старое состояние базы данных, вы всё ещё можете запустить `aide --check` для старой базы данных и провести сравнительный анализ отчётов в случае возникновения такой необходимости.

Также утилита `aide` поддерживает сравнение новой версии базы данных со старой — за это отвечает аргумент командной строки `--compare`. Для этого вам необходимо либо добавить в конфигурационный файл `/etc/aide.conf` директиву `database_new`, указывающую на путь к файлу с новой базой данных:

```
database_new=file:@@{DBDIR}/aide.db.new.gz
```

Либо вы можете определить значение директивы с помощью аргументов командной строки `--before` или `--after` при вызове `aide --compare`:

```
$ aide --compare --after='database_new=file:@@{DBDIR}/aide.db.new.
↪gz'
Start timestamp: 2025-01-17 11:19:47 +0300 (AIDE 0.16)
AIDE found differences between the two databases!!

Summary:
  Total number of entries:      204420
  Added entries:                1
  Removed entries:              0
  Changed entries:              2

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

d++++++++++++++++: /root/.config/procps

---------------------------------------------------
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

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

f   ...    .C... : /etc/cups/subscriptions.conf
f   ...    .C... : /etc/cups/subscriptions.conf.O

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

File: /etc/cups/subscriptions.conf
  SHA512   : iY//ZXGfIOCw5w+dsglTJ5uanAu6ycGo |
 ↪gw0CO7RN6yjIbRHvTVjthdZyfI/igk7K
             SU5HoonBNSbGDb2sNVZLX/olLvPHyPDo |
 ↪ZGoICuAK4CEZHcoTQTMPo0RSNRUqWXz0
             qGGLZHHx35ce7yFxVxmjdQ==         |
 ↪el6dW9hvpSltkjJxywX3RQ==

File: /etc/cups/subscriptions.conf.O
  SHA512   : HwYU5Ddhxs+MTd+7h67ToWOc1njts3eu |
 ↪MZOiOFpYIbRYnZECCA+Dq9UZCoBlTO/n
             Lro+IZrhTR+yVA0W85Ji0cKP/77ZdqFS | 0ELsf8rnHmL/
 ↪pp6uz2GgJHURnrHclo0h
             7FRt4bCEB1L1T8SNe8Nonw==         |
 ↪l4JpCwIjfXhWANs3mkkzzg==



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

/var/lib/aide/aide.db.gz
  MD5      : bJRt0P9rQ3lR7Wq4SZJlsw==
  SHA1     : /nM1abzKvpoUSt22HiuKAsx7zE4=
  RMD160   : Fw7M3O5xq55CKfXXuJgeQ9bn8aE=
  TIGER    : cJJxmyGjwuecrEIUNhJgsrRApqQmHNv6
  SHA256   : iUpCg8pVKy6a34FdCNurMUL04BtV65sD
             8k9pgX1mSr0=
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
  SHA512   : /+DHSwBp2kcMHvO5HXk3FQvLbvZUO1xj
             6+YDcT4kXyUcAekf46zoEKdDVd89AX8S
             H3Xh7CRxF2uQ4wFTrFv8Gg==

/var/lib/aide/aide.db.new.gz
  MD5      : 9XQcGyeUGCo4jQFqKNWCSw==
  SHA1     : JT98QJxegb+XsjzCHB5sbaFpsoQ=
  RMD160   : t8pAZIn/4MsB+YiYi7wr1uie4iY=
  TIGER    : oddP+JHtsDFFr+9GeCymlZ7meJV0K5uI
  SHA256   : SB9BYNaOzq8f5QYMbkNzTZ78yMWyp0lF
             UUbNH8Q04Ig=
  SHA512   : WeutFJZKQQydQkBSGvsuyC/ASE54vOcP
             NnrJlj7PjjMXINFpQvIhgQvE+LwtDFjq
             Kj9/MeYhjHQchYHNwhmotw==


End timestamp: 2025-01-17 11:19:57 +0300 (run time: 0m 10s)
```

После обновления базы данных и завершения работы с отчётами переименуйте файл, чтобы утилита `aide` начала использовать новую базу для последующих проверок:

```
$ sudo mv /var/lib/aide/aide.db.new.gz /var/lib/aide/aide.db.gz
```

### 9.2.5. Оповещение о нарушении целостности объектов контроля

Для автоматической проверки целостности системы утилитой `aide` с заданной периодичностью можно использовать службу `cron` или таймеры `systemd`, а для отправки уведомлений — локальный почтовый сервер или любое другое решение, которое можно вызвать из сценария командной строки.

#### 9.2.5.1. Сценарий для автоматического запуска aide

Ниже приведён пример сценария командной строки, который запускает утилиту `aide` и отправляет её отчёт электронным письмом, если код возврата был ненулевым. Такой код возврата означает, что во время проверки были обнаружены расхождения с базой данных или возникли ошибки.

```
#!/bin/bash

# e-mail адрес, на который необходимо отправить отчёт aide
ADMIN_EMAIL='admin@localhost'

# исходящий e-mail адрес
FROM_EMAIL='aide@localhost'

# путь к файлу отчёта утилиты aide, задаётся директивой report_url
 ↪в
# конфигурационном файле
AIDE_LOG='/var/log/aide/aide.log'

# тема письма
SUBJECT='AIDE: integrity check failed'

# запустить aide и отправить файл отчёта если код возврата не равен
 ↪0
/sbin/aide --check &>/dev/null || /sbin/sendmail -i -t <<EOF
To: ${ADMIN_EMAIL}
From: ${FROM_EMAIL}
Subject: ${SUBJECT}

$(cat ${AIDE_LOG})
EOF
```

Сохраните сценарий в файл `/usr/local/bin/aide-report.sh`, установите корректные права доступа и сделайте его исполняемым:

```
$ sudo chown root:root /usr/local/bin/aide-report.sh
$ sudo chmod 755 /usr/local/bin/aide-report.sh
```

#### 9.2.5.2. Настройка локального почтового сервера

В данном руководстве в качестве почтовой службы предлагается использовать `postfix`, для его установки и включения выполните следующие команды:

```
$ sudo dnf install -y postfix
$ sudo systemctl enable --now postfix
```

Для целей тестирования дополнительная конфигурация почтового сервиса не требуется — достаточно чтобы он был запущен и пользователь, на имя которого будет отправляться электронная почта, существовал в системе. Полученная почта будет сохраняться в текстовый файл очереди `/var/spool/mail/ИМЯ_ПОЛЬЗОВАТЕЛЯ`, с которым может работать как почтовый клиент Evolution (тип сервера «Стандартная для Unix очередь типа mbox»), так и любая программа для просмотра текстовых файлов (`less`, `cat` и т.п.). Для получения информации о безопасной настройке почтовой службы для реальных условий вам необходимо обратиться к специализированному руководству.

#### 9.2.5.3. Периодический запуск aide

##### Периодический запуск aide с помощью cron

Для периодического запуска утилиты `aide` с помощью службы `cron` добавьте соответствующую запись в файл `crontab` пользователя `root`. Например, для проверки целостности системы каждые три часа, запустите редактор `crontab` командой `crontab -e`, добавьте в конец файла следующую запись и сохраните изменения:

```
0 */3 * * * /usr/local/bin/aide-report.sh
```

##### Периодический запуск aide с помощью таймера systemd

В качестве альтернативы службе `cron` вы также можете использовать таймеры `systemd` для запуска периодических задач.

В первую очередь создайте сервисный файл `/etc/systemd/system/aide-report.service` следующего содержания:

```
[Unit]
Description=AIDE periodic scan

[Service]
Type=simple
# команда для запуска
ExecStart=/usr/local/bin/aide-report.sh
# запускать команду от имени пользователя root
User=root
```

Затем создайте файл `/etc/systemd/system/aide-report.timer` с описанием таймера:

```
[Unit]
Description=Run AIDE scan every 3 hours

[Timer]
# запускать таймер каждые 3 часа
OnUnitActiveSec=3h
# запустить таймер через 5 минут после загрузки системы
OnBootSec=5min

[Install]
WantedBy=timers.target
```

Установите правильные права доступа и владельца для созданных файлов:

```
$ sudo chown root:root /etc/systemd/system/aide-report.{service,
 ↪timer}
$ sudo chmod 644 /etc/systemd/system/aide-report.{service,timer}
```

Обновите конфигурацию `systemd` и запустите таймер:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable --now aide-report.timer
Created symlink /etc/systemd/system/timers.target.wants/aide-
 ↪report.timer → /etc/systemd/system/aide-report.timer.
```

После этого таймер активируется и проверка системы будет выполняться каждые три часа.

Статус таймера можно посмотреть с помощью следующей команды:

```
$ sudo systemctl status aide-report.timer
● aide-report.timer - Run AIDE scan every 3 hours
     Loaded: loaded (/etc/systemd/system/aide-report.timer;
 ↪enabled; preset: disabled)
     Active: active (waiting) since Tue 2025-01-28 18:52:27 MSK;
 ↪4min 41s ago
      Until: Tue 2025-01-28 18:52:27 MSK; 4min 41s ago
    Trigger: Tue 2025-01-28 21:52:27 MSK; 2h 55min left
   Triggers: ● aide-report.service

Jan 28 18:52:27 msvsphere.localdomain systemd[1]: Started Run AIDE
 ↪scan every 3 hours.
```

### 9.2.6. Опции командной строки утилиты aide

Таблица 33 - Опции командной строки утилиты aide

| Аргумент | Описание |
|---|---|
| `--check, -C` | Выполняет проверку системы на предмет нарушения целостности относительно состояния, зафиксированного в базе данных `aide`. База данных должна быть инициализирована и находиться по пути, определённом директивой `database` в конфигурационном файле. Этот режим работы используется по умолчанию если утилита запущена без каких-либо аргументов. |
| `--init, -i` | Инициализирует базу данных для хранения состояния системы. После инициализации вам потребуется переименовать файл чтобы команда `--check` могла с ним работать. |

| Аргумент | Описание |
|---|---|
| `--update, -u` | Обновляет базу данных `aide` чтобы она соответствовала текущему состоянию системы. Входная (`database` в конфигурационном файле) и выходная (`database_out` в конфигурационном файле) базы данных должны отличаться. |
| `--compare, -E` | Сравнивает две базы данных `aide`. Пути к ним должны быть определены директивами `database` и `database_new` в конфигурационном файле. |
| `--config-check, -D` | Останавливает работу `aide` после чтения конфигурационного файла, пользователь будет уведомлён о всех обнаруженных в нём ошибках. |
| `--config=<путь>, -c <путь>` | Задаёт путь к конфигурационному файлу `aide`. Значение по умолчанию — `/etc/aide.conf`. Используйте `-` для чтения конфигурационного файла со стандартного ввода (stdin). |
| `--limit=<рег_выр> , -l <рег_выр>` | Ограничить проверку и обновление БД только файлами и каталогами, соответствующими заданному регулярному выражению. Обратите внимание, что регулярное выражение совпадает только с началом строки. Пример запуска `aide` для проверки только объектов, путь которых начинается с `/etc`: `aide --update --limit /etc`. Все остальные объекты будут проигнорированы. |
| `--before=<параметры>, -B <параметры>` | Позволяет задать конфигурационные параметры, которые будут применены перед чтением конфигурационного файла. С полным списком доступных параметров вы можете ознакомиться на странице документации `man aide.conf`. |

| Аргумент | Описание |
|---|---|
| `--after=<параметры>, -A <параметры>` | Позволяет задать конфигурационные параметры, которые будут применены после чтения конфигурационного файла. С полным списком доступных параметров вы можете ознакомиться на странице документации `man aide.conf`. |
| `--verbose=<уровень>, -V<уровень>` | Определяет степень детальности вывода `aide`, допустимые значения находятся в пределах от `0` до `255`. Значение по умолчанию — `5`, при указании аргумента без уровня будет использовано значение `20`. Указанное через командную строку значение имеет больший приоритет чем значение, определённое в конфигурационном файле. |
| `--report=<URI>, -r <URI>` | Указывает `aide` куда отправлять отчёты. Список поддерживаемых значений вы можете посмотреть в секции `URLS` страницы документации `man aide.conf`. |
| `--version, -v` | Вывести версию и параметры сборки `aide` на экран и завершить работу. |
| `--help, -h` | Выдать справочную информацию об аргументах командой строки и завершить работу. |

### 9.3. Замкнутая программная среда (IMA/EVM)

В состав операционной системы МСВСфера входят компоненты IMA (Integrity Measurement Architecture) и EVM (Extended Verification Module), предназначенные для обеспечения целостности системы и защиты от несанкционированных изменений.

IMA — это подсистема ядра Linux, которая контролирует целостность файлов в системе:

- при обращении к файлам вычисляет их хеш-суммы с помощью функции криптографического хеширования. Используемые ключи хранятся в подсистеме связки ключей ядра Linux;

- сохраняет вычисленные хеш-суммы в расширенных атрибутах файловой системы;
- обеспечивает проверку текущего содержимого файлов на соответствие ранее сохранённым эталонным значениям. Запрещает выполнение каких-либо операций с файлом если текущее значение хеш-суммы не совпадает с ожидаемым;
- сохраняет результаты проверок в системный журнал для их последующего анализа.

EVM — это расширение IMA, которое контролирует целостность метаданных файлов в системе:

- подписывает метаданные файлов (хеш-суммы IMA, атрибуты SELinux, права доступа, информацию о владельце/группе и т.п.) с помощью функции криптографического хеширования. Используемые ключи хранятся в подсистеме связки ключей ядра Linux;
- проверяет подлинность метаданных файлов чтобы предотвратить их несанкционированное изменение.

Вместе технологии IMA и EVM обеспечивают комплексную систему защиты, которая помогает предотвратить запуск или использование изменённых или поддельных файлов. Система контроля целостности активируется на раннем этапе загрузки операционной системы, непосредственно после монтирования корневой файловой системы и перед запуском `systemd`, и остаётся активной пока компьютер не будет выключен.

### 9.3.1. Настройка подсистемы EVM

#### 9.3.1.1. Создание ключей

Подсистема EVM использует симметричные ключи переменной длины для подписи и верификации расширенных атрибутов файлов.

Поддерживаются два типа ключей:

- доверенные (trusted) — ключи, которые создаются и шифруются с использованием аппаратного модуля TPM (Trusted Platform Module), что обеспечивает дополнительный уровень безопасности;
- программные — ключи, которые создаются с использованием программного генератора случайных чисел ядра Linux, наличие модуля TPM при этом не требуется. Такие ключи считаются менее защищёнными,

но при этом обеспечивают лучшую производительность, что может быть иметь значение для высоконагруженных систем.

Необходимо решить, какой тип ключей будет использоваться в вашей системе и следовать инструкциям, предложенным в соответствующем разделе данного руководства.

Для генерации ключей и последующей настройки подсистемы IMA/EVM установите следующие пакеты:

```
$ dnf install attr keyutils ima-evm-utils
```

**Предупреждение**

**Все команды, перечисленные в данном разделе, необходимо выполнять от имени пользователя root в рамках одной сессии** поскольку в процессе работы ключи хранятся только в хранилище пользовательских ключей текущей сессии.

### Создание доверенных ключей (TPM)

#### Предварительные требования

Операционная система МСВСфера 9 поддерживает работу только с модулями TPM версии 2.0. Проверить версию модуля можно следующим образом:

```
$ cat /sys/class/tpm/tpm0/tpm_version_major
2
```

Модули TPM 1.2 не поддерживаются.

Вам также потребуется установить пакет `tpm2-tools`, предоставляющий утилиты для работы с TPM модулем:

```
$ dnf install tpm2-tools
```

### Создание корневого ключа хранилища TPM

Для создания ключей подсистемы EVM с помощью модуля TPM в первую очередь необходимо создать корневой ключ хранилища TPM (Storage Root Key, SRK):

```
$ tpm2_createprimary --hierarchy=o --key-algorithm=rsa2048 \
    --hash-algorithm=sha256 --key-context=srk-key.ctxt
name-alg:
 value: sha256
 raw: 0xb
attributes:
 value: fixedtpm|fixedparent|sensitivedataorigin|userwithauth| \
 restricted|decrypt
 raw: 0x30072
type:
 value: rsa
 raw: 0x1
exponent: 65537
bits: 2048
scheme:
 value: null
 raw: 0x10
scheme-halg:
 value: (null)
 raw: 0x0
sym-alg:
 value: aes
 raw: 0x6
sym-mode:
 value: cfb
 raw: 0x43
sym-keybits: 128
rsa: a7d3ff17a593a18fb3c0e85184dc2cec3ae322cb4df59a \
553ac8457fc675a942b1e79c16e25fcefeb707078364abfd5061cf29edc51396 \
68dd938546322f76d5faff726d64783ef500f088810e3e4b55d7d479bcc371097 \
1a698ae101ab6c18d0635cf2beea81d07c18f80c87a68f483a2 \
2ca8751291f5b87f7681f44143237baaab6df6f9f3563c3a0ccd30e172a11b3ed \
c9e1f234ac7c5574db10b9b4c16422d82e9875e526ebe22b36e \
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
3dff941b48d324913829903dd9dc87f8d37fd9381dfb24a425c2e2ec1befc2606d
 ↪\
3724a71adae6815ecdbe69eacdba50e3e9c3d8562b170b45525 \
2929edbf518826887add46c86e28dd30d3229f1c7d2249b3f9a2d
```

Где аргументы имеют следующий смысл (см. подробности в `man 1 tpm2_createprimary`):

- `--hierarchy=o` — иерархия платформы TPM, в которой необходимо создать ключ, o — это `TPM_RH_OWNER`;
- `--key-algorithm=rsa2048` — использовать алгоритм RSA2048 для генерации ключа;
- `--hash-algorithm=sha256` — использовать SHA256 в качестве алгоритма хеширования;
- `--key-context=srk-key.ctxt` — файл, куда следует сохранить контекст объекта TPM (условно, это некая ссылка/ключ поиска объекта в TPM).

Для всех вышеприведённых аргументов кроме `--key-context` были указаны их значения по умолчанию, соответственно, при реальном использовании их можно опустить.

Далее, необходимо сделать ключ хранилища персистентным — записать его в постоянный дескриптор (persistent handle):

```
$ tpm2_evictcontrol -c srk-key.ctxt 0x81000001
persistent-handle: 0x81000001
action: persisted
```

В данном примере используется дескриптор `0x81000001`, что является общепринятой практикой, но доступные постоянные дескрипторы могут отличаться у разных TPM-модулей. Посмотреть их список можно следующим образом:

```
$ tpm2_getcap handles-persistent
- 0x81000001
- 0x81000002
- 0x81010001
- 0x81800000
- 0x81800001
```

Однако при использовании эмулируемых TPM-устройств, например, в виртуальных машинах, это не всегда работает.

### Создание доверенного ключа EVM

Следующим этапом является генерация доверенного (trusted) ключа, с помощью которого затем будет создан защищённый ключ для EVM:

```
$ keyctl add trusted kmk-trusted "new 32 keyhandle=0x81000001" @u
534881893
```

в результате будет сгенерирован 256-битный (32 байта) доверенный ключ `kmk-trusted` в пользовательской связке ключей ядра (@u). Обратите внимание на аргумент `keyhandle=0x81000001` — его значение должно совпадать с дескриптором, в который был записан корневой ключ хранилища в на предыдущем этапе. Команда `keyctl add` возвращает серийный номер созданного ключа, в данном примере — `534881893`.

Теперь необходимо экспортировать доверенный ключ в файл чтобы сделать возможной его загрузку после перезагрузки системы:

```
# связать ключи ядра пользовательского пространства (@u) с
 ↪пространством
# сессии (@s) чтобы команда keyctl pipe могла его экспортировать
$ keyctl link @u @s

# экспортировать доверенный ключ в файл /etc/keys/kmk-trusted.blob
$ keyctl pipe "$(keyctl search @u trusted kmk-trusted)" > /etc/
 ↪keys/kmk-trusted.blob

# разрешить чтение и изменение файла только владельцу, в данном
 ↪случае -
# пользователю root
$ chmod 600 /etc/keys/kmk-trusted.blob
```

### Создание защищённого ключа EVM

Защищённый ключ подсистемы EVM используется для подписывания метаданных файлов. Он генерируется с помощью доверенного ключа следующей командой:

```
$ keyctl add encrypted evm-key "new trusted:kmk-trusted 32" @u
695962815
```

В данном примере `695962815` — это серийный номер созданного ключа.

Экспортируйте созданный ключ в файл `/etc/keys/evm-trusted.blob` чтобы его можно было загружать в ядро после перезагрузки системы:

```
$ keyctl pipe "$(keyctl search @u encrypted evm-key)" > /etc/keys/
 ↪evm-trusted.blob
$ chmod 600 /etc/keys/evm-trusted.blob
```

### Автоматическая загрузка доверенного ключа EVM (TPM)

Ядро операционной системы работает с ключами, которые находятся в памяти. Соответственно, после перезагрузки компьютера эти ключи необходимо загружать обратно в память ядра.

Вручную доверенный ключ EVM можно загрузить следующей командой:

```
$ keyctl add trusted kmk-trusted "load `cat /etc/keys/kmk-trusted.
 ↪blob`" @u
559687686
```

Однако имеет смысл автоматизировать процесс с помощью dracut-модуля `masterkey`. Для этого создайте конфигурационный файл `/etc/sysconfig/masterkey` следующего содержания:

```
MULTIKERNELMODE="NO"
MASTERKEYTYPE="trusted"
MASTERKEY="/etc/keys/kmk-trusted.blob"
```

И добавьте модуль `masterkey` в initramfs:

```
$ echo 'add_dracutmodules+=" masterkey "' > /etc/dracut.conf.d/97-
 ↪masterkey.conf
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
$ dracut --regenerate-all --force --verbose
...
dracut: *** Including module: masterkey ***
...
dracut: *** Creating image file '/boot/initramfs-6.1.123-4.lvc12.
 ↪el9.inferit.x86_64.img' ***
dracut: dracut: using auto-determined compression method 'pigz'
dracut: *** Creating initramfs image file '/boot/initramfs-6.1.123-
 ↪4.lvc12.el9.inferit.x86_64.img' done ***
```

### Создание программных ключей EVM

Программные ключи являются полностью программной альтернативой доверенным ключам — для их создания и использования не требуется наличие TPM модуля. Такие ключи считаются менее безопасными, но при этом они обеспечивают лучшую производительность по сравнению с ключами, использующими TPM-модуль.

### Создание корневого (KMK) ключа

В первую очередь необходимо создать KMK (Kernel Master Key) — корневой ключ, с помощью которого затем будет сгенерирован защищённый ключ для подсистемы EVM.

Для генерации ключа выполните следующую команду:

```
$ keyctl add user kmk-user "$(dd if=/dev/urandom bs=1 count=32 2> /
 ↪dev/null)" @u
26358515
```

в результате будет сгенерирован 256-битный ключ (32 байта) `kmk` в пользовательской связке ключей ядра (`@u`). Команда `keyctl add` возвращает серийный номер созданного ключа, в данном примере — `26358515`.

Теперь необходимо экспортировать доверенный ключ в файл чтобы сделать возможной его загрузку после перезагрузки системы:

```
# связать ключи ядра пользовательского пространства (@u) с
 ↪пространством
# сессии (@s) чтобы команда keyctl pipe могла его экспортировать
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
$ keyctl link @u @s

# экспортировать ключ в файл /etc/keys/kmk-user.blob
$ keyctl pipe "$(keyctl search @u user kmk-user)" > /etc/keys/kmk-
 ↪user.blob

# разрешить чтение и изменение файла только владельцу, в данном
 ↪случае –
# пользователю root
$ chmod 600 /etc/keys/kmk-user.blob
```

### Создание защищённого ключа EVM

Защищённый ключ подсистемы EVM используется для подписывания метаданных файлов. Он генерируется с помощью корневого (KMK) ключа следующей командой:

```
$ keyctl add encrypted evm-key "new user:kmk-user 64" @u
441992374
```

в результате будет создан 512-битный (64 байта) ключ `evm-key` в пользовательской связке ключей ядра (@u), выводом команды является серийный номер этого ключа.

Далее необходимо экспортировать созданный ключ в файл для загрузки в память ядра после перезагрузки системы и установить корректные права доступа:

```
$ keyctl pipe "$(keyctl search @u encrypted evm-key)" > /etc/keys/
 ↪evm-trusted.blob
$ chmod 600 /etc/keys/evm-trusted.blob
```

### Автоматическая загрузка корневого (KMK) ключа

Ядро операционной системы работает с ключами, которые находятся в памяти. Соответственно, после перезагрузки компьютера эти ключи необходимо загружать обратно в память ядра.

Вручную корневой (KMK) ключ можно загрузить следующей командой:

```
$ keyctl add user kmk "$(cat /etc/keys/kmk-user.blob)" @u
456548873
```

Однако имеет смысл автоматизировать процесс с помощью dracut модуля `masterkey`. Для этого создайте конфигурационный файл `/etc/sysconfig/masterkey` следующего содержания:

```
MULTIKERNELMODE="NO"
MASTERKEYTYPE="user"
MASTERKEY="/etc/keys/kmk-user.blob"
```

И добавьте модуль `masterkey` в initramfs:

```
$ echo 'add_dracutmodules+=" masterkey "' > /etc/dracut.conf.d/97-
 ↪masterkey.conf
$ dracut --regenerate-all --force --verbose
...
dracut: *** Including module: masterkey ***
...
dracut: *** Creating image file '/boot/initramfs-6.1.123-4.lvc12.
 ↪el9.inferit.x86_64.img' ***
dracut: dracut: using auto-determined compression method 'pigz'
dracut: *** Creating initramfs image file '/boot/initramfs-6.1.123-
 ↪4.lvc12.el9.inferit.x86_64.img' done ***
```

### Автоматическая загрузка защищённого ключа EVM

Как и в случае с доверенным ключом, защищённый ключ EVM можно загружать в память ядра вручную:

```
$ keyctl link @u @s
$ keyctl add encrypted evm-key "load $(cat /etc/keys/evm-trusted.
 ↪blob)" @u
560076073
```

Однако для автоматического включения подсистемы EVM во время загрузки системы требуется настроить автоматическую загрузку защищённого ключа. Для этого служит dracut-модуль `integrity`.

Необходимо создать конфигурационный файл `/etc/sysconfig/evm` следующего содержания:

```
EVMKEY="/etc/keys/evm-trusted.blob"
```

И добавить модуль `integrity` в initramfs:

```
$ echo 'add_dracutmodules+=" integrity "' > /etc/dracut.conf.d/98-
 ↪integrity.conf
$ dracut --regenerate-all --force --verbose
...
dracut: *** Including module: integrity ***
...
dracut: *** Creating image file '/boot/initramfs-6.1.123-4.lvc12.
 ↪el9.inferit.x86_64.img' ***
dracut: dracut: using auto-determined compression method 'pigz'
dracut: *** Creating initramfs image file '/boot/initramfs-6.1.123-
 ↪4.lvc12.el9.inferit.x86_64.img' done ***
```

### 9.3.2. Настройка подсистемы IMA

#### 9.3.2.1. Выпуск ЭЦП для подсистемы IMA

Для построения замкнутой программной среды (ЗПС) вам потребуется электронно-цифровая подпись (ЭЦП), выпущенная на вашу организацию и заверенная производителем операционной системы МСВСфера.

Создайте конфигурационный файл `ima-sign.conf` следующего содержания:

```
[req]
default_bits = 2048
distinguished_name = req_distinguished_name
prompt = no
string_mask = utf8only
x509_extensions = sign_extensions

[req_distinguished_name]
# название сертификата вашей организации
CN = IMA test signing key
# название вашей организации
O = IMA test
# контактный e-mail адрес вашей организации
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
emailAddress = ima-test@example.com

[sign_extensions]
basicConstraints=critical, CA:false
keyUsage=critical, digitalSignature
extendedKeyUsage = critical, codeSigning
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid
```

Скорректируйте значения полей `CN` (Common Name), `O` (Organization) и `emailAddress` чтобы они соответствовали вашей организации.

Выполните следующую команду для генерации приватного ключа и запроса на выпуск сертификата ЭЦП:

```
$ openssl req -new -batch -config ima-sign.conf -out ima-sign.csr \
      -keyout ima-sign.key
```

В процессе выполнения команды потребуется ввести пароль для создаваемого приватного ключа. Для генерации ключа без пароля необходимо добавить аргумент `-nodes`, но это ухудшает безопасность и не рекомендуется для промышленного использования — в случае утечки приватного ключа злоумышленник сразу сможет им воспользоваться.

В результате выполнения команды в текущем каталоге будут созданы следующие файлы:

- `ima-sign.key` — приватный ключ будущего сертификата;
- `ima-sign.csr` — CSR-запрос (Certificate Signing Request) на выпуск сертификата.

Поместите файл `ima-sign.key` в безопасное место, недоступное другим пользователям, допустим, в каталог `/root/ima`.

Просмотреть информацию, которая содержится в запросе на выпуск сертификата, можно с помощью команды:

```
$ openssl req -text -noout -verify -in ima-sign.csr
```

Далее, вам необходимо отправить файл `ima-sign.csr` производителю ОС МСВСфера для выпуска публичного сертификата. Производитель вернёт вам файл `ima-sign.der` с заверенным публичным сертификатом, данный

файл необходимо поместить в каталог `/etc/keys/ima` и установить для него безопасные права доступа:

```
$ mkdir -p /etc/keys/ima
$ mv ima-sign.der /etc/keys/ima/
$ chown root:root /etc/keys/ima/ima-sign.der
$ chmod 644 /etc/keys/ima/ima-sign.der
```

Просмотреть информацию, которая содержится в публичном сертификате, можно с помощью команды:

```
$ openssl x509 -noout -text -in /etc/keys/ima/ima-sign.der
```

#### 9.3.2.2. Запуск IMA/EVM в режиме первичной настройки

Для продолжения настройки необходимо перезагрузить систему в режим первичной настройки IMA/EVM:

```
$ grubby --update-kernel=/boot/vmlinuz-$(uname -r) \
      --args="ima_policy=appraise_tcb ima_appraise=fix evm=fix"
$ reboot
```

После перезагрузки необходимо убедиться, что подсистема IMA/EVM корректно инициализирована и готова к настройке:

```
$ cat /sys/kernel/security/evm
1
```

`1` — подсистема EVM включена, `0` — выключена.

```
$ dmesg | grep 'evm: key'
[    2.815612] evm: key initialized
```

Ключ для подсистемы EVM был успешно загружен.

#### 9.3.2.3. Настройка замкнутой программной среды

Создайте каталог для хранения политик IMA:

```
$ mkdir /etc/ima
```

И файл политики IMA `/etc/ima/ima-policy`, в качестве стартовой точки можно использовать следующий набор правил:

```
# игнорировать файловую систему procfs: /proc
dont_appraise fsmagic=0x9fa0
# игнорировать файловую систему sysfs: /sys
dont_appraise fsmagic=0x62656572
# игнорировать файловую систему debugfs: /sys/kernel/debug
dont_appraise fsmagic=0x64626720
# игнорировать файловую систему tmpfs
dont_appraise fsmagic=0x1021994
# игнорировать файловую систему ramfs
dont_appraise fsmagic=0x858458f6
# игнорировать файловую систему devpts: /dev/pts
dont_appraise fsmagic=0x1cd1
# игнорировать файловую систему binfmt: /proc/sys/fs/binfmt_misc
dont_appraise fsmagic=0x42494e4d
# игнорировать файловую систему securityfs: /sys/kernel/security
dont_appraise fsmagic=0x73636673
# игнорировать файловую систему selinuxfs: /sys/fs/selinux
dont_appraise fsmagic=0xf97cff8c
# игнорировать файловую систему smackfs: /sys/fs/smackfs
dont_appraise fsmagic=0x43415d53
# игнорировать файловую систему nsfs: /proc/*/ns/*
dont_appraise fsmagic=0x6e736673
# игнорировать файловую систему efivarfs: /sys/firmware/efi/efivars
dont_appraise fsmagic=0xde5e81e4
# игнорировать файловую систему cgroup: /sys/fs/cgroup
dont_appraise fsmagic=0x27e0eb
# игнорировать файловую систему cgroup2: /sys/fs/cgroup
dont_appraise fsmagic=0x63677270
# игнорировать файловую систему FAT32
dont_appraise fsmagic=0x4d44
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# не выполнять проверку файлов с меткой SELinux var_log_t. Этим
 ↪типом
# маркируются файлы системных журналов, которые постоянно
 ↪изменяются.
# Соответственно, проверка их ЭЦП не имеет смысла.
dont_measure obj_type=var_log_t
dont_appraise obj_type=var_log_t
# не выполнять проверку файлов с меткой SELinux audit_log_t - это
 ↪файлы
# системного журнала событий безопасности службы auditd, которые
 ↪так же
# постоянно изменяются.
dont_measure obj_type=auditd_log_t
dont_appraise obj_type=auditd_log_t
# не выполнять проверку файлов с меткой SELinux (s)virt_image_t -
 ↪это файлы
# образов виртуальных машин, которые так же постоянно изменяются
dont_measure obj_type=svirt_image_t
dont_appraise obj_type=svirt_image_t
dont_measure obj_type=virt_image_t
dont_appraise obj_type=virt_image_t

# проверять ЭЦП для файлов с политиками IMA
appraise func=POLICY_CHECK appraise_type=imasig
# проверять ЭЦП запускаемых файлов
appraise func=BPRM_CHECK appraise_type=imasig
# проверять ЭЦП для отображаемых в память файлов (системный вызов
 ↪mmap)
appraise func=FILE_MMAP mask=MAY_EXEC appraise_type=imasig
# проверять ЭЦП для загружаемых ядром прошивок
appraise func=FIRMWARE_CHECK appraise_type=imasig
# проверять IMA хеши (но не ЭЦП) для всех файлов, которыми владеет
 ↪root
appraise fowner=0
```

Установите для файла политик корректные права доступа:

```
$ chmod 600 /etc/ima/ima-policy
$ chown root:root /etc/ima/ima-policy
$ restorecon /etc/ima/ima-policy
```

И подпишите его вашей ЭЦП, заверенной производителем ОС:

```
$ evmctl ima_sign --key=/root/ima/ima-sign.key /etc/ima/ima-policy
hash(sha256):
 ↪ca9fe00713e1280f2162fd3a2af7c8296f3a5aa7e9136150df10ad7414656622
Enter PEM pass phrase:
evm/ima signature: 264 bytes
0302042a5b2625010063d3f76227571772377b52a4bb8d0c6d9b1ee26a2cee2aa0
 ↪\
1ffa2507599e1926f08d71c261eb48ebdd26e5d2420b7701a164ab1c5f61524147c
 ↪\
254f0737d8425120d3df40a9cfbc3094fefbcd91f3bf3eeb8b6785665212ffb8539
 ↪\
166a0071552ce476af57b6e97d4fc96bd00ad3cf01d6a5c254127e9a7b443b85198
 ↪\
2b9137e50383524df6e6e8715ea585251dcd721c5877a632dc6e2aaca4010fdb80
 ↪\
c9d1299f4b1ebf8e2a073647919f1d3cb0f2040b2033b4effd9ccbfd010978a748
 ↪\
c1ce9d5c9af71d9f1e8d89f94f4634dd44770c95a8996157dadab3bf2ac5144c3d
 ↪\
7a9c4bf8bc7c3f20b0db1c855c9b017b5f7a5e1bbc24cb9b02ef4e3664d68a1a7
```

Задайте корректный путь к файлу привтного ключа вашей ЭЦП, в данном примере используется `/root/ima/ima-sign.key`. При следующей загрузке системы политика IMA будет автоматически загружена в ядро подсистемой `systemd`.

Далее необходимо будет подписать вашей ЭЦП все файлы, заданные политикой IMA.

Следующая команда подпишет все исполняемые файлы, владельцем которых является пользователь `root`:

```
$ /usr/bin/find / \( -fstype ext4 -o -fstype xfs \) -type f -
 ↪executable -uid 0 \
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
      -exec evmctl --hashalgo=sha256 ima_sign --pass=PASSWORD \
      --key=/root/ima/ima-sign.key '{}' \;
```

Здесь и в последующих листингах замените PASSWORD на пароль от приватного ключа вашей ЭЦП IMA.

Для подписи разделяемых библиотек выполните:

```
$ /usr/bin/find /lib /lib64 /usr/lib /usr/lib64 /usr/libexec/ \
      \( -fstype ext4 -o -fstype xfs \) -\! -executable -type f -
↪name "*.so*" \
      -uid 0 -exec evmctl --hashalgo=sha256 \
      ima_sign --pass=PASSWORD --key=/root/ima/ima-sign.key '{}' \;
```

Следующая команда подпишет модули ядра:

```
$ /usr/bin/find /lib/modules \( -fstype ext4 -o -fstype xfs \) \
      -type f -name "*.ko" -uid 0 -exec evmctl --hashalgo=sha256 \
      ima_sign --pass=PASSWORD --key=/root/ima/ima-sign.key '{}' \;
```

Для подписи прошивок драйверов выполните:

```
$ /usr/bin/find /lib/firmware \( -fstype ext4 -o -fstype xfs \) \
      -type f -uid 0 -exec evmctl --hashalgo=sha256 \
      ima_sign --pass=PASSWORD --key=/root/ima/ima-sign.key '{}' \;
```

Для завершения настройки подсистемы IMA необходимо обновить хеш-суммы всех файлов для подсистемы EVM:

```
$ find / \( -fstype ext4 -o -fstype xfs \) -type f -uid 0 \
      -exec dd if='{}' of=/dev/null count=0 status=none \;
```

Выполнение последней команды может занять достаточно продолжительное время (несколько минут), поскольку выполняется маркировка всех файлов в системе.

Теперь необходимо перевести систему в основной режим работы IMA/EVM и перезагрузить компьютер:

```
$ grubby --update-kernel=/boot/vmlinuz-$(uname -r) \
      --args="ima_policy=appraise_tcb ima_appraise=enforce" \
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
      --remove-args="evm=fix"
$ reboot
```

### 9.3.3. События безопасности IMA/EVM

В случае обнаружения нарушения целостности отслеживаемых файлов подсистема IMA/EVM создаёт в журнале службы аудита `/var/log/audit/audit.log` запись об инциденте с типом `INTEGRITY_DATA`.

В качестве эксперимента создайте копию исполняемого файла `/usr/bin/cp`:

```
$ sudo cp /usr/bin/cp /root
```

И попробуйте его запустить:

```
$ sudo /root/cp --version
-bash: /root/cp: Отказано в доступе
```

В это время системном журнале событий безопасности появится следующая запись:

```
$ sudo ausearch -m INTEGRITY_DATA -ts recent
----
time->Wed Feb  5 19:38:49 2025
type=PROCTITLE msg=audit(1738773529.602:286): proctitle="-bash"
type=SYSCALL msg=audit(1738773529.602:286): arch=c000003e
 ↪syscall=59 success=no exit=-13 a0=55996be5ce10 a1=55996bd40c00
 ↪a2=55996be4b570 a3=8 items=0 ppid=2680 pid=2776 auid=1000 uid=0
 ↪gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=1
 ↪comm="bash" exe="/usr/bin/bash" subj=unconfined_u:unconfined_
 ↪r:unconfined_t:s0-s0:c0.c1023 key=(null)
type=INTEGRITY_DATA msg=audit(1738773529.602:286): pid=2776 uid=0
 ↪auid=1000 ses=1 subj=unconfined_u:unconfined_r:unconfined_t:s0-
 ↪s0:c0.c1023 op=appraise_data cause=IMA-signature-required comm=
 ↪"bash" name="/root/cp" dev="dm-0" ino=67975531 res=0 errno=0
```

При этом исходный файл, подписанный вашей ЭЦП, может быть запущен без проблем:

```
$ /usr/bin/cp --version
cp (GNU coreutils) 8.32
```

### 9.3.4. Рекомендации по отладке и диагностике IMA/EVM

Основные сообщения об ошибках IMA/EVM появляются в `dmesg` по ключам `ima`, `evm` или `dract-pre-pivot`, для их просмотра можно использовать следующую команду:

```
$ dmesg | grep -i -e evm -e ima -e dracut-pre
[    0.000000] Command line: BOOT_IMAGE=(hd0,gpt2)/vmlinuz-6.1.123-
 ↪4.lvc12.el9.inferit.1.x86_64 root=/dev/mapper/mvg-root ro
 ↪resume=/dev/mapper/mvg-swap rd.lvm.lv=mvg/root rd.lvm.lv=mvg/swap
 ↪rhgb quiet ima_policy=appraise_tcb ima_appraise=enforce
[    0.039996] mtrr_cleanup: can not find optimal value
[    0.081870] Kernel command line: BOOT_IMAGE=(hd0,gpt2)/vmlinuz-
 ↪6.1.123-4.lvc12.el9.inferit.1.x86_64 root=/dev/mapper/mvg-root ro
 ↪resume=/dev/mapper/mvg-swap rd.lvm.lv=mvg/root rd.lvm.lv=mvg/swap
 ↪rhgb quiet ima_policy=appraise_tcb ima_appraise=enforce
[    0.081943] Unknown kernel command line parameters "rhgb BOOT_
 ↪IMAGE=(hd0,gpt2)/vmlinuz-6.1.123-4.lvc12.el9.inferit.1.x86_64",
 ↪will be passed to user space.
[    0.418423] Trying to unpack rootfs image as initramfs...
[    1.003913] Loaded X.509 cert 'NCSD LLC: MSVSphere IMA CA:
 ↪77006f8de02bdc27ead8dbf4c1da12adbf6e05ec'
[    1.014173] ima: Allocated hash algorithm: sha256
[    1.024341] ima: No architecture policies found
[    1.024368] evm: Initialising EVM extended attributes:
[    1.024368] evm: security.selinux
[    1.024369] evm: security.SMACK64 (disabled)
[    1.024369] evm: security.SMACK64EXEC (disabled)
[    1.024369] evm: security.SMACK64TRANSMUTE (disabled)
[    1.024370] evm: security.SMACK64MMAP (disabled)
[    1.024370] evm: security.apparmor (disabled)
[    1.024370] evm: security.ima
[    1.024371] evm: security.capability
[    1.024371] evm: HMAC attrs: 0x1
[    1.133410] Freeing unused kernel image (initmem) memory: 3116K
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
[    1.133878] Freeing unused kernel image (text/rodata gap)
 ↪memory: 2040K
[    1.133948] Freeing unused kernel image (rodata/data gap)
 ↪memory: 4K
[    1.140578]     BOOT_IMAGE=(hd0,gpt2)/vmlinuz-6.1.123-4.lvc12.
 ↪el9.inferit.1.x86_64
[    1.156567] systemd[1]: systemd 252-46.el9_5.2.inferit.1 running
 ↪in system mode (+PAM +AUDIT +SELINUX -APPARMOR +IMA +SMACK
 ↪+SECCOMP +GCRYPT +GNUTLS +OPENSSL +ACL +BLKID +CURL +ELFUTILS
 ↪+FIDO2 +IDN2 -IDN -IPTC +KMOD +LIBCRYPTSETUP +LIBFDISK +PCRE2 -
 ↪PWQUALITY +P11KIT -QRENCODE +TPM2 +BZIP2 +LZ4 +XZ +ZLIB +ZSTD -
 ↪BPF_FRAMEWORK +XKBCOMMON +UTMP +SYSVINIT default-
 ↪hierarchy=unified)
[    1.156691] systemd[1]: Hostname set to <ima-test>.
[    1.326797] device-mapper: core: CONFIG_IMA_DISABLE_HTABLE is
 ↪disabled. Duplicate IMA measurements will not be recorded in the
 ↪IMA log.
[    2.077542] fbcon: vmwgfxdrmfb (fb0) is primary device
[    2.605704] PM: Image not found (code -22)
[    2.890032] evm: key initialized
[    3.376525] systemd[1]: Successfully loaded the IMA custom
 ↪policy /etc/ima/ima-policy.
[    3.376527] ima: policy update completed
[    3.437773] systemd[1]: systemd 252-46.el9_5.2.inferit.1 running
 ↪in system mode (+PAM +AUDIT +SELINUX -APPARMOR +IMA +SMACK
 ↪+SECCOMP +GCRYPT +GNUTLS +OPENSSL +ACL +BLKID +CURL +ELFUTILS
 ↪+FIDO2 +IDN2 -IDN -IPTC +KMOD +LIBCRYPTSETUP +LIBFDISK +PCRE2 -
 ↪PWQUALITY +P11KIT -QRENCODE +TPM2 +BZIP2 +LZ4 +XZ +ZLIB +ZSTD -
 ↪BPF_FRAMEWORK +XKBCOMMON +UTMP +SYSVINIT default-
 ↪hierarchy=unified)
[    3.719750] systemd[1]: TPM2 PCR Machine ID Measurement was
 ↪skipped because of an unmet condition check
 ↪(ConditionPathExists=/sys/firmware/efi/efivars/
 ↪StubPcrKernelImage-4a67b082-0a4c-41cf-b6c7-440b29bb8c4f).
```

Особое внимание следует обратить на следующие сообщения:

- `Kernel command line: ... ima_policy=appraise_tcb ima_appraise=enforce` — IMA/EVM запущены в основном режиме;

- `evm: key initialized` — ключ EVM был успешно загружен в ядро;
- `systemd[1]: Successfully loaded the IMA custom policy /etc/ima/ima-policy` и `ima: policy update completed` — политика IMA была успешно загружена в ядро.

Для просмотра расширенных атрибутов файла используйте команду `getfattr`:

```
$ getfattr -d -m - /usr/bin/bash
getfattr: Removing leading '/' from absolute path names
# file: usr/bin/bash
security.evm=0sAmb9oe4fRqe+0QHweWVpFF6FJ6JM
security.ima=0sAwIEKlsmJQEAbpXlqB4Xg3QISGrJ \
FX4aJb2Y4LrMb2rvQlrOTQ+GgCM42+9cFnwYGyfOP8ua9 \
9R//ZukguXEgY1VWc5jdFyKmbfJHkmJRebYHa15Q9LGQ7lr \
9mfMxaNU8fU7VvdKHDhtPDA3mU2VbPamLI+eB210kT6MCGGs \
2bsou512HImpKE7jQR/X0nmOoOAJTeuinT9qeB2QKtuF2pth3H \
g03iU85xdAhFi+ZWHAKj2i3F4frI/pTXwQE1XrwS67ULpME4MRT \
LJWpyC4g8rZnBk50/2KK0QY4Rv6d2isIKA+OiccGu/jdKVxrGn \
5zI1f28gGeKw+Amx44rKnmR9eSRz25MtFog==
security.selinux="system_u:object_r:shell_exec_t:s0"
```

Для проверки IMA-подписи файла испольуется утилита `evmctl` из состава пакета `ima-evm-utils`:

```
$ evmctl ima_verify --key=/etc/keys/ima/ima-sign.der /usr/bin/bash
key 1: 2a5b2625 /etc/keys/ima/ima-sign.der
/usr/bin/bash: verification is OK
```

В качестве значения аргумент `--key` принимает путь к публичному сертификату IMA-подписи.

Для просмотра публичных ключей IMA в связке ключей ядра используйте следующую команду:

```
$ keyctl show %keyring:.ima
Keyring
 332912505 ---lswrv      0     0  keyring: .ima
1043633425 ---lswrv      0     0   \_ asymmetric: NCSD LLC:
 ↪MSVSphere kernel signing key:
 ↪d4353662476ed06d8c1d78c4dcf488fb698a44a1
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
279154357 --als--v      0     0   \_ asymmetric: NCSD LLC:
↪MSVSphere 9 IMA release key:
↪90881827430f8032f8ab35acde286d3bb9f555e0
432155330 --als--v      0     0   \_ asymmetric: IMA test signing
↪key: d1bea3d33a7fe2795470ac3dd046c45f2a5b2625
```

Первые два ключа принадлежат ЭЦП производителя операционной системы МСВСфера, а последний ключ должен совпадать с идентификатором вашего IMA-сертификата, который можно получить с помощью команды:

```
$ openssl x509 -noout -text -in /etc/keys/ima/ima-sign.der | grep -
↪A1 "Subject Key Identifier"
            X509v3 Subject Key Identifier:

↪D1:BE:A3:D3:3A:7F:E2:79:54:70:AC:3D:D0:46:C4:5F:2A:5B:26:25
```

Если ваша система не загружается по причине неверной конфигурации IMA/EVM, то вы можете вручную добавить в список опций ядра параметры `evm=fix ima_appraise=fix` через меню загрузчика Grub — компьютер загрузится в режим первичной настройки IMA/EVM и вы сможете устранить проблему.

### 9.4. Проверка контрольных сумм неизменяемых компонентов

Для выполнения проверки контрольных сумм неизменяемых компонентов операционной системы МСВСфера 9 (ФСТЭК) вам необходимо скачать утилиту `fstec-checksum-validator.py` (https://docs.msvsphere-os.ru/_static/integrity-monitoring/fstec-checksum-validator.py) и файл `msvsphere-fstec-checksums.txt` с контрольными суммами с официальной страницы Продукта (https://msvsphere-os.ru/downloads/).

Пометьте скачанный файл `fstec-checksum-validator.py` как исполняемый:

```
$ chmod +x fstec-checksum-validator.py
```

Затем, запустите следующую команду для выполнения проверки:

```
$ sudo ./fstec-checksum-validator.py verify -c msvsphere-fstec-
 ↪checksums.txt
```

Программа завершит работу с нулевым кодом возврата, если контрольные суммы всех неизменяемых компонентов соответствуют эталонным:

```
$ sudo ./fstec-checksum-validator.py verify -c msvsphere-fstec-
 ↪checksums.txt
0 failed, 123531 passed

$ echo $?
0
```

В случае обнаружения несоответствий программа завершит работу с ненулевым кодом возврата и выведет на экран список файлов, контрольные суммы которых отличаются от эталонных:

```
$ sudo ./fstec-checksum-validator.py verify -c msvsphere-fstec-
 ↪checksums.txt
/usr/libexec/git-core/git-http-backend checksum
 ↪b8f115ad8618a6a37af4c02f6a4558c9f256ca3af4eb6bbfd59f55a0ae49d6ff
 ↪does not match expected
/usr/libexec/git-core/git-http-fetch checksum
 ↪b7d5446ef3c34a3d633260a5ef59f7ade0cd7ed3fe953e65c512b9026ef4ef41
 ↪does not match expected
/usr/libexec/git-core/git-http-push checksum
 ↪93a9b24b51cdb92af75710dbd00c242c0894ef7fc6a19f319d72a0eaf0996dd2
 ↪does not match expected
/usr/libexec/git-core/git-imap-send checksum
 ↪6e8e3c34822ccc824ba08154c7d1da3315d8f6f28bec413d7423c04e07932306
 ↪does not match expected
/usr/libexec/git-core/git-remote-http checksum
 ↪95301168ea86f0ec01dd11aec1692e4a706d3ff39ada97a8888e18dadbee682c
 ↪does not match expected
/usr/libexec/git-core/git-sh-i18n--envsubst checksum
 ↪6fe4f71da651a23eb2721727c02fede3100c282f932eef885b71514e20f3861c
 ↪does not match expected
6 failed, 123525 passed
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
$ echo $?
1
```

# 10. ЗАЩИТА ПАМЯТИ

## 10.1. Защита оперативной памяти в ОС МСВСфера

Операционная система МСВСфера ОС основана на базе ядра GNU/Linux, соответственно, защита оперативной памяти и ограничение прав доступа к страницам памяти реализовано на базе стандартных механизмов ядра, компилятора GCC и функций аппаратного обеспечения.

## 10.2. Аппаратная защита от переполнения буфера

Начиная с версии ядра GNU/Linux 2.6.8, выпущенной в 2004 году, на центральных процессорах архитектуры x86 поддерживается аппаратная защита от переполнения буфера путём выполнения кода в страницах памяти, помеченных как данные — NX-Bit (No Execute Bit) в терминологии AMD или XD-Bit (Execute Disable Bit) в терминологии Intel. Эта технология так же реализована в современных ARM-процессорах.

Проверить, задействован ли этот механизм можно с помощью одной из следующих команд:

```
$ dmesg | grep 'Execute Disable'
[    0.000000] NX (Execute Disable) protection: active
```

Или

```
$ sudo journalctl | grep "protection: active"
Oct 08 10:10:25 localhost kernel: NX (Execute Disable) protection:
 ↪active
```

Если защита не активна, то, возможно, в настройках BIOS/UEFI вашего оборудования есть настройка для её включения.

## 10.3. Программная защита от переполнения буфера

Использование современных компиляторов позволяет применять в МСВСфера ОС различные методы защиты от переполнения буфера.

Уже классическим методом защиты стека от переполнения является «канарейка» (canary stack protection), названный так в честь птиц, которых когда-то шахтёры использовали в шахтах в качестве примитивных газоанализаторов: если птица умирала, значит находиться в шахте было небезопасно. Суть

метода заключается в том, что при каждом запуске программы генерируется некое секретное число, которое затем записывается в память перед адресом возврата из функции и проверяется при выходе из функции. Если оно не соответствует ожидаемому, то программа немедленно завершает свою работу. При переполнении буфера данное значение, соответственно, затирается, что и приводит к срабатыванию защиты. В пакеты МСВСфера ОС данная защита добавляется автоматически через использование опции компилятора `-fstack-protector-strong`.

Ещё одной возможностью компилятора, применяемой для сборки пакетов, является `FORTIFY_SOURCE`, которая добавляет проверку на переполнение буфера для различных функций, выполняющих операции с памятью и со строками.

Также при сборке пакетов используется опция компилятора `-fstack-clash-protection`, которая реализует защиту от атак типа «stack clash». Соответствующая защита реализована в ядре ОС и в библиотеке glibc. Смысл такой атаки заключается в том, чтобы вызвать выполнение вредоносного кода или повысить привилегии одним из следующих способов:

- пересечение (clashing) — вызвать пересечение стека с другой областью памяти путём выделения памяти, пока стек не достигнет другой области памяти или пока другая область памяти не достигнет стека.
- прыжок (jumping) — позволяет переместить указатель стека в другую область памяти, не затрагивая сторожевую страницу памяти.
- разбиение (smashing) — выполнить перезапись стека содержимым другой области памяти или выполнить перезапись другой области памяти содержимым стека.

ASLR (address space layout randomization) или рандомизация размещения адресного пространства — ещё одна технология защиты памяти, применяемая в МСВСфера ОС. Суть защиты ASLR сводится к использованию случайных адресов для размещения сегментов кода и данных в адресном пространстве процесса, что усложняет эксплуатацию различных уязвимостей, связанных с переполнением буфера, так как атакующему сначала потребуется «угадать» по каким адресам расположены те или иные структуры данных процесса (стек, куча и т.п.).

Защита ASLR включена по умолчанию и какие-либо дополнительные действия по её настройке не требуются. Проверить статус можно с помощью следующей команды:

```
$ sudo sysctl -a | grep kernel.randomize_va_space
kernel.randomize_va_space = 2
```

Опция `kernel.randomize_va_space` может принимать следующие значения:

- 0 — защита ASLR отключена, рандомизация адресного пространства не происходит;
- 1 — защита ASLR включена, случайные адреса используются для разделяемых библиотек, стека, VDSO и системного вызова `mmap()`;
- 2 — (по умолчанию в МСВСфера ОС) защита ASLR включена, в дополнение к предыдущим пунктам случайные адреса также будут использоваться для кучи и системного вызова `brk()`.

Также в данном разделе стоит упомянуть технологию KASLR (Kernel Address Space Layout Randomization), которая затрудняет реализацию некоторых атак путём размещения структур данных ядра по случайному адресу при каждой загрузке операционной системы. Эта защита включена по умолчанию и не требует какой-либо дополнительной настройки.

### 10.4. Принудительная очистка оперативной памяти

В ядре МСВСфера ОС доступна функция принудительной очистки (перезаписи нулями) оперативной памяти во время её выделения и/или освобождения, что может значительно осложнить атаки, связанные с утечкой информации при повторном использовании памяти.

Эта функция отключена по умолчанию, поскольку приводит к некоторому замедлению операций, связанных с управлению страницами памяти, однако её включение может иметь смысл для повышения безопасности многопользовательских или критически важных систем.

Для включения принудительной очистки памяти используются следующие опции ядра:

- `init_on_alloc=1` — заполнять выделяемые страницы памяти и объекты кучи нулями.
- `init_on_free=1` — заполнять освобождаемые страницы памяти и объекты кучи нулями.

В своих рекомендациях по безопасной настройке операционных систем

Linux ФСТЭК рекомендует использовать только опцию `init_on_alloc=1`, но технически возможно использование обеих опций одновременно или по отдельности.

Для включения опции `init_on_alloc=1` для всех установленных в системе ядер выполните следующую команду:

```
$ sudo grubby --update-kernel=ALL --args="init_on_alloc=1"
```

И перезагрузите компьютер.

После перезагрузки вы можете проверить содержимое файла `/proc/cmdline`, чтобы убедиться в том, что функция очистки памяти активна:

```
$ grep -oP 'init_on_alloc\S+' /proc/cmdline
init_on_alloc=1
```

Чтобы отключить функцию принудительной очистки памяти для всех ядер используйте следующую команду:

```
$ sudo grubby --update-kernel=ALL --remove-args="init_on_alloc=1"
```

После которой также потребуется перезагрузить компьютер для применения изменений.

# 11. ОБЕСПЕЧЕНИЕ НАДЁЖНОГО ФУНКЦИОНИРОВАНИЯ

## 11.1 Введение

Средства обеспечения надёжного функционирования предоставляют возможности резервного копирования и восстановления данных и программного обеспечения при сбоях и отказах, а также возможности функционирования отдельных экземпляров системы на нескольких технических средствах в отказоустойчивом режиме, обеспечивающем доступность сервисов и данных при выходе из строя одного из технических средств или при исчерпании вычислительных ресурсов.

## 11.2. Архивация файлов и директорий

Утилита `tar` позволяет архивировать файлы и директории со всеми их поддиректориями и файлами, а затем восстанавливать их из архива, т.е. является удобным средством для создания резервных копий.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленным в таблице.

Таблица 34 - Опции утилиты `tar` и их значения

| Опция | Значение |
|---|---|
| `-c, --create` | Создать новый архив. |
| `-r, --append` | Присоединить файлы к концу архива. |
| `--delete` | Удалить файл из архива. |
| `-t, --list` | Вывести список содержимого архива. |
| `-A, --catenate, --concatenate` | Присоединить существующий архив к другому архиву. |
| `-x, --extract, -get` | Извлечь файлы из архива. |
| `-u, --update` | Добавить в архив более новые версии файлов. |
| `-C, --directory=DIR` | Сменить директорию перед выполнением операции на `DIR`. |
| `--f, --file=ARCHIVE` | Вывести результат в архивный файл или в устройство `ARCHIVE`. |
| `-d, --diff` | Осуществить проверку на наличие различий между архивом и некоторой файловой системой. |

| Опция | Значение |
|---|---|
| `-v, --verbose` | Выводить подробную информацию о процессе выполнения команды. |

**Пример**: в примере директория `mydir` и все её поддиректории сначала сохраняются в файле `myarch.tar`:

```
$ tar cf myarch.tar mydir
```

а затем извлекаются из архива:

```
$ tar xf myarch.tar
```

А этот скрипт организует хранение четырех последних резервных копий директории `/var/www` в директории `/opt/backup/www-backup`. Первая версия будет всегда иметь номер 0, последняя — номер 3. При создании новых версий старые будут удаляться. Сами резервные копии хранятся в сжатом виде.

```
#! /bin/bash
cd /opt/backup/www-backup
rm www-dump-3.tar.gz
cp www-dump-2.tar.gz www-dump-3.tar.gz
cp www-dump-1.tar.gz www-dump-2.tar.gz
cp www-dump-0.tar.gz www-dump-1.tar.gz
tar --selinux --acls --xattrs --czf www-dump-o.tar.gz /var/www
```

### 11.3. Создание архивов и извлечение файлов из них

Утилита `cpio` используется для создания архивов и извлечения файлов из них, а также для копирования файлов в целях их переноса из текущей директории в другую. Поддерживает множество различных архивных форматов. При извлечении файлов из архива утилита автоматически распознает, каким типом обладает архив, с которым она взаимодействует.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 35 - Опции утилиты `cpio` и их значения

| Опция | Значение |
|---|---|
| `-o, --create` | Копировать файлы в архив. |
| `-A, --append` | Добавить файлы в архив. Может быть использована только в связке с опцией `-o`. |
| `-i, --extract` | Копирует файлы из архива или выводит список содержимого некоторого архива. |
| `-p, --pass-through` | Копирует файлы из одной файловой структуры в другую, комбинируя при этом режимы работы, использующиеся при передаче опций `-i` и `-o`, но не используя при этом архивы. |
| `-a, --reset-access-time` | Сбрасывает времена обращения к входным файлам после их копирования, так что при использовании данной опции будет нельзя распознать, что файлы были скопированы. |

**Пример**: в примере сначала флеш-носитель монтируется как устройство `/mnt`:

```
$ mount /dev/sdb4 /mnt
```

Затем создается и записывается на флеш-носитель резервная копия директории `/lib`:

```
$ find /lib/ | cpio -o > /mnt/2/backup.cpi
```

Для того чтобы восстановить все файлы в директорию `/lib` из созданной ранее архивной копии, необходимо выполнить следующую команду:

```
$ cpio -ivmd /lib/\* < /mnt/2/backup.cpio
```

## 11.4. Резервное копирование данных

Утилита amanda обладает возможностью резервного копирования данных, хранящихся на множестве компьютеров в вычислительной сети. Она реализует клиент-серверную модель и использует следующие утилиты:

**-клиентская утилита amandad, взаимодействующая с сервером системы.**

**Во время своего выполнения вызывает другие утилиты:**

- selfcheck (проверка конфигурации клиента);
- sendsize (оценка объема резервной копии);
- sendbackup (выполнение операции резервного копирования);
- amcheck (проверка конфигурации системы).

**-серверная утилита amdump, инициирующая все операции резервного копирования.**

**Во время своего выполнения использует другие утилиты и контролирует их выполнение:**

- planner (определение того, что надо копировать);
- driver (интерфейс к внешнему устройству);
- dumper (связывается с клиентским процессом amandad);
- taper (запись данных на внешнее устройство);
- amreport (подготовка сообщения о выполненном копировании).

**-административные утилиты:**

- amcheck (проверка готовности системы к работе);
- amlabel (записать метку на сменный носитель перед использованием в системе);
- amcleanup (очистить систему после неплановой перезагрузки сервера или после неплановго завершения операции резервного копирования);
- amflush (переписать данные из дискового кэша на внешний носитель);
- amadmin (выполнение большого количества различных административных операций).

**-утилиты восстановления данных:**

- `amrestore` (восстановление данных с носителей, на которых записаны резервные копии, выполненные системой);
- `amrecover` (программа для интерактивного восстановления данных с резервных копий).

### 11.5. Создание дисковых RAID-массивов

Утилита `mdadm` позволяет создавать так называемые дисковые RAID-массивы с использованием технологии распределения данных по нескольким дискам с целью достижения избыточности, отказоустойчивости, сокращения задержек и/или увеличения скорости чтения и записи, а также для улучшения возможностей восстановления данных в случае отказа.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 36 - Опции утилиты `mdadm` и их значения

| Опция | Значение |
|---|---|
| `-A, --assemble` | Режим сборки ранее созданного массива и его активации. |
| `-B, --build` | Режим сборки массива без суперблоков. |
| `-C, --creat` | Режим сборки нового массива. |
| `-F, --follow, --monitor` | Режим слежения за состоянием устройств. |
| `-G, --grow` | Режим расширения или уменьшения размера массива. |
| `-N, --name` | Устанавливает имя массива. |
| `-n, --raid-devices` | Указывает количество активных устройств в массиве. |
| `-x, --space-device` | Указывает количество запасных устройств в массиве. |
| `-z, --size` | Указывает объем пространства, используемого для каждого диска. |
| `-l, --level` | Устанавливает уровень массива. |

| Опция | Значение |
|---|---|
| `-c, --config` | Указывает файл конфигурации. По умолчанию `/etc/mdadm.conf`. |
| `-f, --fail` | Помечает перечисленные устройства как неисправные. |
| `-S, --stop` | Деактивирует массив и освобождает все ресурсы. |
| `-V --version` | Выводит информацию о версии утилиты. |
| `-h, --help` | Выводит справку об утилите. |

# 12. ФИЛЬТРАЦИЯ СЕТЕВОГО ПОТОКА

## 12.1. Введение

Средства фильтрации сетевого потока предоставляют возможности фильтрации входящих и исходящих сетевых потоков на основе установленного набора правил с учетом атрибутов безопасности и используемых сетевых протоколов, а также управления правилами фильтрации сетевых потоков; регистрации и учета выполнения проверок при фильтрации сетевых потоков.

## 12.2. Настройка файрвола (брандмауэра)

Утилита `firewall-cmd` позволяет настраивать работу файрвола (брандмауэра), осуществляющего фильтрацию сетевых потоков при помощи определения так называемых зон, иными словами, наборов правил, которые управляют трафиком на основе уровня доверия к той или иной сети.

Существуют следующие зоны:

- `drop` — самый низкий уровень доверия к сети. Весь входящий трафик сбрасывается без ответа. Поддерживаются только исходящие соединения;
- `block` — эта зона похожа на предыдущую, но при этом входящие запросы сбрасываются с сообщением `icmp-host-prohibited` или `icmp6-adm-prohibited`;
- `public` — эта зона представляет публичную сеть, которой нельзя доверять, однако поддерживает входящие соединения в индивидуальном порядке;
- `external` — зона внешних сетей. Поддерживает маскировку NAT, благодаря чему внутренняя сеть остается закрытой, но с возможностью получения доступа;
- `internal` — обратная сторона зоны `external`. Компьютерам в этой зоне можно доверять.

Доступны дополнительные сервисы:

- `dmz` — используется для компьютеров, расположенных в DMZ (зонах изолированных компьютеров, которые не будут иметь доступа к остальной части сети). Поддерживает только некоторые входящие соединения;
- `work` — зона рабочей сети. Большинству машин в сети можно доверять, доступны дополнительные сервисы;

- `home` — зона домашней сети. Окружению можно доверять, но поддерживаются только определённые пользователем входящие соединения;
- `trusted` — всем машинам в сети можно доверять.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 37 - Опции утилиты `firewall-cmd` и их значения

| Опция | Значение |
|---|---|
| `--state` | Вывести состояние файрвола. |
| `--reload` | Перезагрузить правила из постоянной конфигурации. |
| `--complete-reload` | Жёсткая перезагрузка правил с разрывом всех соединений. |
| `--runtime-to-permanent` | Перенести настройки `runtime` в постоянную конфигурацию. |
| `--permanent` | Использовать постоянную конфигурацию. |
| `--get-default-zone` | Отобразить зону, используемую по умолчанию. |
| `--set-default-zone` | Установить зону по умолчанию. |
| `--get-active-zones` | Отобразить активные зоны. |
| `--get-zones` | Отобразить все доступные зоны. |
| `--get-services` | Вывести предопределенные сервисы. |
| `--list-all-zones` | Вывести конфигурацию всех зон. |
| `--new-zone` | Создать новую зону. |
| `--delete-zone` | Удалить зону. |
| `--list-all` | Вывести всё, что добавлено, из выбранной зоны. |
| `--list-services` | Вывести все сервисы, добавленные к зоне. |
| `--add-service` | Добавить сервис к зоне. |
| `--remove-service` | Удалить сервис из зон. |
| `--list-ports` | Отобразить порты, добавленные к зоне. |
| `--add-port` | Добавить порт к зоне. |
| `--remove-port` | Удалить порт из зоны. |
| `--query-port` | Показать, добавлен ли порт к зоне. |

| Опция | Значение |
|---|---|
| `--list-protocols` | Вывести протоколы, добавленные к зоне. |
| `--add-protocol` | Добавить протокол к зоне. |
| `--remove-protocol` | Удалить протокол из зоны. |
| `--list-source-ports` | Вывести порты источника, добавленные к зоне. |
| `--add-source-port` | Добавить порт-источник к зоне. |
| `--remove-source-port` | Удалить порт-источник из зоны. |
| `--list-icmp-blocks` | Вывести список блокировок icmp. |
| `--add-icmp-block` | Добавить блокировку icmp. |
| `--remove-icmp-block` | Удалить блокировку icmp. |
| `--add-forward-port` | Добавить порт для перенаправления в NAT. |
| `--remove-forward-port` | Удалить порт для перенаправления в NAT. |
| `--add-masquerade` | Включить NAT. |
| `--remove-masquerade` | Удалить NAT. |

**Пример**: настройка правила блокировки адреса получателя может выглядеть следующим образом:

```
$ sudo firewall-cmd --permanent --direct --add-rule ipv4 filter
 ↪OUTPUT 0 -d 192.168.10.20 -j DROP
success
```

**Пример**: настройка правила отбрасывания всех входящих соединений по протоколу IPv4 может выглядеть следующим образом:

```
$ sudo firewall-cmd --permanent --direct --add-rule ipv4 filter
 ↪INPUT 0 -j DROP
success
```

**Пример**: настройка правила отбрасывания всех исходящих пакетов UDP может выглядеть следующим образом:

```
$ sudo firewall-cmd --permanent --direct --add-rule ipv4 filter
 ↪OUTPUT 0 -p upd -j DROP
success
```

### 12.3. Конфигурационный файл /etc/firewalld/firewalld.conf

Конфигурационный файл `/etc/firewalld/firewalld.conf` содержит основные параметры конфигурации для файрвола `firewalld`.

- `DefaultZone` — устанавливает зону по умолчанию для соединений или интерфейсов;
- `MinimalMark` — с этой опцией блок меток может быть зарезервирован для частного использования. Используются только отметки над этим значением. Значение по умолчанию равно `100`;
- `CleanupOnExit` — если `firewalld` останавливается, он очищает все правила. Если для этого параметра установлено значение `no` или `false`, текущие правила останутся нетронутыми. Значением по умолчанию является `yes` или `true`;
- `Lockdown` — если эта опция включена, изменения `firewalld` с интерфейсом D-Bus будут ограничены приложениями, которые перечислены в белом списке блокировки. Значением по умолчанию является `no` или `false`;
- `IPv6_rpfilter` — если эта опция включена, выполняется проверка фильтра обратного пути для пакета для IPv6. Если ответ на пакет будет отправлен через тот же интерфейс, на который поступил пакет, пакет совпадет и будет принят. В противном случае он будет отброшен. Для IPv4 `rp_filter` управляется с помощью `sysctl`;
- `IndividualCalls` — если этот параметр отключен, используются комбинированные вызовы restore, а не отдельные вызовы, чтобы применить изменения к файрволу. Использование отдельных вызовов увеличивает время, необходимое для применения изменений;
- `LogDenied` — добавление правил ведения журнала непосредственно перед отклонением и удалением правил в цепочках `INPUT`, `FORWARD` и `OUTPUT` для правил по умолчанию, а также окончательных правил отклонения и отбрасывания в зонах для настроенного типа пакета канального уровня. По умолчанию установлено `off` — отключение ведения журнала.
- `AutomaticHelpers` — для безопасного использования протокола IPv4 `iptables` и помощников по отслеживанию соединений этот параметр рекомендуется отключить. Возможные значения: `yes`, `no`, `system`. По

умолчанию установлено `system`;

- `FirewallBackend` — выбирает реализацию брандмауэра. Возможные значения: `nftables` (по умолчанию) или `iptables`. Это относится ко всем примитивам `firewalld`. Единственным исключением являются прямые и сквозные правила, которые всегда используют традиционные `iptables`, `ip6tables` и `ebtables`.

# 13. МОНИТОРИНГ ФУНКЦИОНИРОВАНИЯ

## 13.1. Введение

Средства мониторинга функционирования предоставляют возможности слежения и сбора информации о выполнении пользовательских процессов и состоянии сетевого трафика.

## 13.2. Анализ системных журналов

Утилита `logwatch` позволяет проводить анализ системных журналов по различным критериям с возможностью составления отчётов.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 38 - Опции утилиты `logwatch` и их значения

| Опция | Значение |
|---|---|
| `--detail level` | Уровень детализации отчета. Может быть положительным целым числом или `high`, `med`, `low`, которые соответствуют целым числам 10, 5 и 0 соответственно. |
| `--debug level` | Уровень отладки. Может варьироваться от 0 до 100. |
| `--logfile log-file-group` | Обрабатывать только набор указанных файлов журналов. |
| `--service service-name` | Обрабатывать только указанную службу. |
| `--print` | Вывести результаты на экран. |
| `--mailto address` | Отправить результаты по указанному адресу электронной почты. |
| `--save file-name` | Сохранить вывод в указанный файл вместо отображения на экране или отправки по электронной почте. |
| `--range range` | Диапазон дат для обработки. |
| `--archives` | Искать в архивных журналах. |
| `--logdir directory` | Обрабатывать файлы журналов из указанного каталога, а не из каталога по умолчанию. |

| Опция | Значение |
|---|---|
| `--hostname hostname` | Обрабатывать файлы журналов только указанного хоста. |

## 13.3. Получение информации о выполняемых процессах

Утилита `top` предназначена для получения информации о выполняемых процессах.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 39 - Опции утилиты `top` и их значения

| Опция | Значение |
|---|---|
| `-u` | Отображать только процессы с заданным идентификатором или именем пользователя. |
| `-S` | Отображать системные процессы. |
| `-n` | Изменить число отображаемых процессов на заданное число. |
| `-i` | Работа в интерактивном режиме. Задается по умолчанию. |
| `-I` | Не отображать бездействующие процессы. По умолчанию отображаются как активные, так и бездействующие процессы. |
| `-c` | Переключение отображения командных строк на отображение имён программ и наоборот. |
| `-s` | Задает временной интервал задержки между обновлениями экрана. По умолчанию 5 секунд. |
| `-b` | Работа в пакетном режиме. Может использоваться для отправки результатов в другие программы или в файл. |
| `-o` | Задает имя поля, по которому будет осуществляться сортировка. Используется в основном для пакетного режимах. |

| Опция | Значение |
|---|---|
| -w | Задает форматирование вывода по ширине. Количество строк считается неограниченным. |
| -v | Показать версию утилиты и выйти. |
| -h | Показать справку и выйти. |

## 13.4. Получение информации о состоянии текущих процессов

Утилита ps используется для получения информации о состоянии текущих процессов.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 40 - Опции утилиты ps и их значения

| Опция | Значение |
|---|---|
| -u | Выводить информацию только о процессах с заданными списком эффективными идентификационными номерами или идентификаторами пользователей. |
| -Y | Выводить информацию только о процессах с заданными списком реальными идентификационными номерами или идентификаторами пользователей. |
| -g | Выводить информацию только о процессах с заданными списком идентификационными номерами групп. |
| -G | Выводить информацию только о процессах с заданными списком реальными идентификационными номерами групп. |
| -a | Выводить информацию о состоянии наиболее часто запрашиваемых процессов. |
| -e | Выводить информацию для всех процессов. |

| Опция | Значение |
|---|---|
| `-d` | Выводить информацию о всех процессах, кроме лидеров сеансов. |
| `-p` | Выводить информацию только для запущенных процессов. |
| `-G` | Выводить информацию о процессах, чьи реальные номера групп указаны в заданном списке. |
| `-o` | Выводить информацию в заданном формате. |

### 13.5. Мониторинг и анализ сетевого трафика

Утилита `tcpdump` предназначена для мониторинга и анализа сетевого трафика.

Состоит из двух частей: захват пакетов с копированием их в так называемый буфер и отображение захваченных пакетов из буфера.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 41 - Опции утилиты `tcpdump` и их значения

| Опция | Значение |
|---|---|
| `-i` | Задает интерфейс, с которого необходимо анализировать трафик. |
| `-y` | Устанавливает тип канала передачи данных для использования во время захвата пакетов. |
| `-e` | Включает вывод данных канального уровня. |
| `-v` | Вывод дополнительной информации. |
| `-w` | Задает имя файла, в котором будет сохраняться собранная информация. |
| `-p` | Захватывать только трафик, предназначенный данному узлу. |
| `-q` | Переводит работу в «бесшумный режим», в котором пакет анализируется на транспортном уровне, а не на сетевом. |

| Опция | Значение |
|---|---|
| `-t` | Отключает вывод меток времени. |
| `-A` | Вывод пакетов в формате ASCII без заголовков канального уровня. |
| `-B` | Установить размер буфера захвата. |
| `-D` | Вывести список доступных сетевых интерфейсов, на которых может осуществляться захват пакетов. |

## 13.6. Получение информации о сеансах пользователей

Утилита `ac` предназначена для получения информации о сеансах пользователей.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 42 - Опции утилиты `ac` и их значения

| Опция | Значение |
|---|---|
| `-p` | Выводить итоговое время сеансов каждого пользователя. |
| `-d` | Кроме общих итогов, выводить итоги за каждый день. |
| `-a` | При выводе ежедневных итогов не пропускать дни, когда входов в систему не было. |
| `-y` | Выводить год при отображении даты. |
| `-z` | Если итоговое значение равно нулю, то выводить его. По умолчанию не выводится. |
| `-v` | Вывести номер версии. |
| `-h` | Вывести краткую справку. |

## 13.7. Получение информации о последних выполненных командах

Утилита `lastcomm` позволяет получить информацию о последних выполненных командах.

Режимы работы утилиты и выполняемые функции задаются набором опций, перечисленных в таблице.

Таблица 43 - Опции утилиты `lastcomm` и их значения

| Опция | Значение |
|---|---|
| `-E` | Выводить время начала процесса выполнения команды. |
| `-S` | Выводить время завершения процесса выполнения команды. |
| `-c` | Выводить количество использованного процессорного времени. |
| `-e` | Выводить количество использованного прошедшего времени. |
| `-s` | Выводить количество использованного системного времени. |
| `-u` | Выводить количество использованного пользовательского времени. |
| `-f` | Использовать заданный файл в качестве источника учетных данных. Он может быть либо стандартным, либо расширенным файлом учёта процесса. |
| `-x` | Использовать текущий расширенный файл учёта процесса. |

# 14. СИСТЕМА ВИРТУАЛИЗАЦИИ

## 14.1. Введение

В данном разделе описывается процедура установки, настройки и использования платформы виртуализации в операционной системе МСВСфера 9.

Виртуализация — это технология, которая позволяет одновременно запускать на одном компьютере несколько виртуальных машин под управлением различных операционных систем. Каждая виртуальная машина при этом работает со своим набором виртуализированных вычислительных ресурсов, таких как центральный процессор, оперативная память, устройства хранения, сетевые интерфейсы и т.п., выделяемым из общего пула аппаратных ресурсов под управлением гипервизора, в нашем случае — операционной системы МСВСфера 9.

### 14.1.1. Основные компоненты платформы виртуализации

- **Гипервизор** — это программное обеспечение, которое обеспечивает возможность одновременного параллельного выполнения нескольких виртуальных машин на одном физическом компьютере. Также гипервизор предоставляет средства для управления виртуальными машинами.
- **KVM** (Kernel-based Virtual Machine) — это часть ядра Linux и ключевой компонент гипервизора, который реализует функцию аппаратной виртуализации, используя возможности современных центральных процессоров. По сути KVM предоставляет в пользовательское пространство необходимый набор ioctl вызовов для создания виртуальной машины, выделения для неё оперативной памяти и виртуального центрального процессора и управления этими ресурсами.
- **QEMU** — компонент гипервизора, работающий в пользовательском пространстве, основной задачей которого является управление ресурсами и эмуляция полноценной аппаратной платформы, на которой сможет работать гостевая операционная система.
- **libvirt** — набор программных компонентов, предоставляющий интерфейсы для управления виртуальными машинами и их оборудованием. По сути своей это высокоуровневая абстракция над QEMU/KVM.

Для удобства пользователя в МСВСфера доступны различные средства для

управления виртуальными машинами и их конфигурацией:

- **virsh, virt-install, virt-xml** — набор утилит командной строки.
- **Cockpit** — веб-интерфейс для управления сервером, который в том числе поддерживает и управление виртуальными машинами.
- **virt-manager** — графическое приложение рабочего стола.

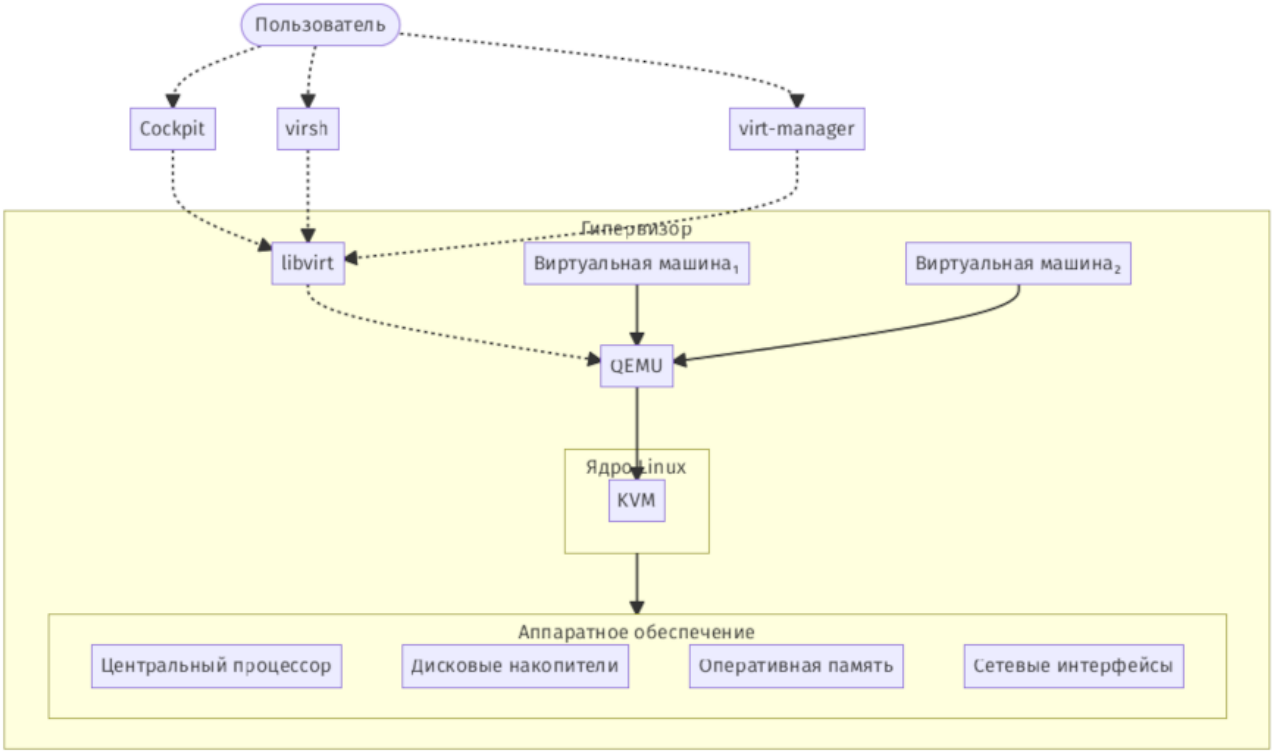



Рис. 24: Основные компоненты платформы виртуализации

## 14.2. Установка

### 14.2.1. Системные требования

Минимальные требования к аппаратному обеспечению:

- 6 ГБ дискового пространства для гипервизора и по 6 ГБ для каждой виртуальной машины;
- 2 ГБ оперативной памяти для гипервизора и по 2 ГБ для каждой виртуальной машины;
- центральный процессор с поддержкой аппаратной виртуализации.

Проверить поддержку центральным процессором виртуализации можно с помощью следующей команды:

```
# ожидаемый вывод для систем на базе процессоров AMD
$ lscpu | grep Virtualization
Virtualization:                      AMD-V

# ожидаемый вывод для систем на базе процессоров Intel
$ lscpu | grep Virtualization
Virtualization:                      VT-x
```

В случае отсутствия флагов AMD-V/VT-x проверьте настройки BIOS/UEFI и спецификацию своего оборудования — некоторые процессоры не поддерживают технологию виртуализации.

### 14.2.2. Установка гипервизора

Для установки гипервизора и базового набора утилит для управления выполните следующую команду:

```
$ sudo dnf install -y qemu-kvm libvirt virt-install
```

Если вы работаете с локальной системой в графическом режиме, то вам также потребуется пакет `virt-viewer`, который позволяет работать с графической консолью виртуальной машины:

```
$ sudo dnf install -y virt-viewer
```

Далее необходимо запустить сервисы виртуализации:

```
$ for service in qemu interface network nodedev nwfilter secret
 ↪storage; do
    sudo systemctl start virt${service}d{,-ro,-admin}.socket
  done
```

Начиная с МСВСфера ОС 9 `libvirt` имеет модульную структуру, соответственно, вышеприведённая команда запускает следующие сервисы:

- `virtqemud` — основной сервис для управления гипервизором;
- `virtinterfaced` — вспомогательный сервис для управления сетевыми интерфейсами хост-системы. Одним из примеров использования является объединение нескольких физических сетевых устройств в одно логическое (network bond).

- `virtnetworkd` — вспомогательный сервис для управления виртуальными сетями. Используется, например, для создания внутренней сети между виртуальными машинами.
- `virtnodedevd` — вспомогательный сервис для управления оборудованием, подключённым к хост-системе. Используется, например, для того чтобы отключить PCI-устройство от хост-системы и подключить его к виртуальной машине.
- `virtnwfilterd` — вспомогательный сервис для управления сетевым экраном (брандмауэром) на хост-системе. Одним из сценариев использования является настройка правил фильтрации пакетов, поступающих на виртуальные машины.
- `virtsecretd` — вспомогательный сервис для хранения различных секретов, например, ключей от зашифрованных разделов виртуальных машин.
- `virtstoraged` — вспомогательный сервис для управления пулами хранения и томами/разделами в этих пулах.

Для проверки возможностей системы виртуализации выполните команду:

```
$ sudo virt-host-validate
  QEMU: проверка для аппаратной виртуализации      : OK
  QEMU: проверка if device /dev/kvm exists
↪          : OK
  QEMU: проверка if device /dev/kvm is accessible
↪          : OK
  QEMU: проверка if device /dev/vhost-net exists
↪          : OK
  QEMU: проверка if device /dev/net/tun exists
↪          : OK
  QEMU: проверка for cgroup 'cpu' controller support
↪          : OK
  QEMU: проверка for cgroup 'cpuacct' controller support
↪          : OK
  QEMU: проверка for cgroup 'cpuset' controller support
↪          : OK
  QEMU: проверка for cgroup 'memory' controller support
↪          : OK
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 QEMU: проверка for cgroup 'devices' controller support
↪         : OK
 QEMU: проверка for cgroup 'blkio' controller support
↪         : OK
 QEMU: проверка для поддержки сопоставления устройств IOMMU: OK
 QEMU: проверка если блок IOMMU включён в ядре          :
↪ПРЕДУПРЕЖДЕНИЕ (IOMMU appears to be disabled in kernel. Add
↪intel_iommu=on to kernel cmdline arguments)
 QEMU: проверка для поддержки безопасных гостевых систем:
↪ПРЕДУПРЕЖДЕНИЕ (Unknown if this platform has Secure Guest
↪support)
```

Если команда `virt-host-validate` возвращает OK (PASS в случае использования английской локали) для всех проверок, то ваша система корректно настроена и готова к работе с виртуальными машинами.

Если какая-то из проверок возвращает ОШИБКУ (FAIL в случае использования английской локали), то необходимо следовать инструкциям на экране для её устранения, чтобы обеспечить корректную работу гипервизора.

Если какая-то из проверок возвращает ПРЕДУПРЕЖДЕНИЕ (WARN в случае использования английской локали), то рекомендуется следовать подсказкам для устранения причины предупреждения.

Устранение неполадок и предупреждений:

- `QEMU: проверка если блок IOMMU включён в ядре` — IOMMU — это технология виртуализации вывода, также известная как Intel VT-d или AMD-Vi. В первую очередь поддержка IOMMU необходима для проброса реальных (не виртуализированных) устройств из хост системы в виртуальную машину, например, видеокарт. Для включения IOMMU на системах на базе процессоров Intel выполните команду:

  ```
  $ sudo grubby --update-kernel=ALL --args="intel_iommu=on"
  ```

  Команда для систем на базе процессоров AMD:

  ```
  $ sudo grubby --update-kernel=ALL --args="amd_iommu=on"
  ```

  В обоих случаях после выполнения команды потребуется перезагрузка компьютера для применения изменений. В некоторых случаях для

включения технологии Intel VT-d/AMD-Vi может также потребоваться изменение соответствующей настройки в BIOS/UEFI компьютера. Следует обратить внимание, что не все центральные процессоры и материнские платы поддерживают данную технологию. Отсутствие поддержки IOMMU не является критичным для работы гипервизора в МСВСфера ОС, однако проброс физического оборудования будет невозможен без этой функции.

- `QEMU: проверка для поддержки безопасных гостевых систем` — это проверка на наличие функции Secure Encrypted Virtualization (SEV), доступной только на современных центральных процессорах от AMD. Для систем на базе процессоров Intel или ARM64 данное предупреждение можно игнорировать.

## 14.3. Режимы работы гипервизора

### 14.3.1. Введение

Система виртуализации МСВСфера ОС поддерживает работу в двух режимах: **пользовательском** и **системном**. Оба режима будут подробно рассмотрены далее в этой главе, однако, для начала достаточно будет упомянуть, что в пользовательском режиме гипервизор запускается от имени обычного (а не привилегированного) пользователя и, соответственно, имеет ограниченный функционал. В системном режиме гипервизор запускается в привилегированном режиме без каких-либо ограничений по поддерживаемым возможностям.

По умолчанию `virsh`, `virt-install` и другие утилиты для работы с гипервизором выбирают с каким режимом гипервизора взаимодействовать на основе имени пользователя: команды от пользователя `root` выполняются в системном режиме, а команды от остальных пользователей, соответственно, в пользовательском. Это поведение можно изменить — процедура настройки описана в разделе «*14.3.4. Выбор режима работы гипервизора*» данной главы.

### 14.3.2. Пользовательский режим

В пользовательском режиме процессы гипервизора и виртуальные машины запускаются с правами текущего непривилегированного пользователя. Соответственно, каждый пользователь может иметь собственный набор виртуальных машин и выделенное хранилище для них.

Это более простой и, отчасти, более безопасный за счёт функциональных ограничений способ работы с гипервизором. Среди ограничений стоит отметить:

- ограничены возможности по конфигурации сетевых устройств;
- не поддерживается автоматический запуск виртуальных машин;
- не поддерживается проброс PCI-устройств в виртуальную машину;
- не выполняется регистрация событий безопасности;
- не применяются общесистемные политики Polkit для контроля и разграничения доступа.

Таким образом, использование гипервизора в пользовательском режиме следует скорее рассматривать как простое решение для работы с виртуальными машинами на рабочей станции, чем комплексное решение для применения в инфраструктуре предприятия.

В совокупности с графической утилитой управления `virt-manager` или веб-панелью `Cockpit` использование данного режима обеспечивает функциональность и пользовательский опыт близкий к таким решениям как Oracle VirtualBox или VMware Player/Workstation.

### 14.3.3. Системный режим

В отличие от пользовательского, в системном режиме процессы гипервизора запускаются с правами привилегированного пользователя и получают полный доступ к оборудованию хост-системы, что позволяет использовать его без каких-либо ограничений. При этом виртуальные машины запускаются от непривилегированного пользователя `qemu`, обеспечивая таким образом необходимый уровень безопасности.

Относительно пользовательского режима становятся доступны следующие функции:

- Автоматический запуск виртуальных машин.
- Проброс PCI-устройств в виртуальную машину.
- Работа с сетевыми устройствами без ограничений.

- *14.14. Регистрация событий безопасности*.
- Использование общесистемных политик Polkit для контроля доступа и реализации ролевой модели.
- Миграция виртуальных машин между гипервизорами.
- Мониторинг состояния виртуальных машин и гипервизора.
- Централизованное управление хранилищами, томами, виртуальными машинами и т.п.

Использование гипервизора в системном режиме является рекомендованным решением для развёртывания технологии виртуализации в промышленной (production) среде.

### 14.3.4. Выбор режима работы гипервизора

Режим работы гипервизора, с которым будет взаимодействовать пользователь, определяется строкой подключения к гипервизору, которую также называют URI (Uniform Resource Identifier — унифицированный идентификатор ресурса).

Для подключения к локальному гипервизору используется одна из следующих форм:

- `qemu:///system` — подключение к системной сессии;
- `qemu:///session` — подключение к пользовательской сессии.

По умолчанию команды, запущенные от имени пользователя `root` подключаются к системной сессии гипервизора, а команды, запущенные от остальных пользователей, — к пользовательской.

Для подключения к удалённому гипервизору необходимо в строке подключения указать его доменное имя или IP-адрес, также рекомендуется использовать защищённый протокол SSH:

- `qemu+ssh://root@example.com/system` — подключение к системной сессии гипервизора на узле `example.com`;
- `qemu+ssh://user@example.com/session` — подключение к сессии пользователя `user` на узле `example.com`.

Задать строку подключения к гипервизору можно одним из следующих способов, указанных в порядке приоритета от большего к меньшему:

1. С помощью аргумента командной строки `-c, --connect <URI>`, который принимает большинство утилит для работы с гипервизором (`virsh`,

`virt-install` и т.д.). Например:

```
$ virsh --connect qemu:///system list
 ID   Имя               Состояние
-----------------------------------
 1    msvsphere-9-arm   работает
```

2. С помощью переменной окружения `LIBVIRT_DEFAULT_URI`:

```
$ LIBVIRT_DEFAULT_URI='qemu:///system' virsh list
 ID   Имя               Состояние
-----------------------------------
 1    msvsphere-9-arm   работает
```

Для отдельного пользователя эту переменную можно определять автоматически, добавив соответствующее объявление в файл `~/.bashrc`:

```
$ cat << EOF >> ~/.bashrc
export LIBVIRT_DEFAULT_URI='qemu:///system'
EOF
```

Установить значение переменной глобально для всех пользователей в системе можно, добавив соответствующее определение в файл `/etc/environment`:

```
$ cat << EOF | sudo tee -a /etc/environment
LIBVIRT_DEFAULT_URI='qemu:///system'
EOF
```

3. С помощью опции `uri_default` в конфигурационном файле libvirt. Для пользователя `root` используется системный конфигурационный файл `/etc/libvirt/libvirt.conf`, а для всех остальных пользователей — файл `~/.config/libvirt/libvirt.conf`.
   Пример настройки автоматического подключения к системному гипервизору для обычного пользователя:

```
$ mkdir -p ~/.config/libvirt
$ cat << EOF >> ~/.config/libvirt/libvirt.conf
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
uri_default = "qemu:///system"
EOF
```

## 14.4. Создание виртуальной машины

### 14.4.1. Создание виртуальной машины с помощью утилиты virt-install

Одним из самых простых способов создания виртуальных машин с помощью командной строки является утилита `virt-install`.

Для вызова команды `virt-install` используется стандартный синтаксис:

```
$ virt-install [аргументы]
```

`virt-install` поддерживает как графический режим установки операционной системы с использованием протоколов VNC или SPICE, так и установку в текстовом режиме с помощью последовательной консоли. Во время создания виртуальной машины она может быть настроена на использование одного или нескольких дисков, сетевых интерфейсов, аудио устройств, аппаратных USB или PCI устройств и т.д.

В качестве установочного носителя может использоваться ISO-образ или виртуальный CD-ROM накопитель, установочное дерево дистрибутива, доступное по протоколам HTTP, HTTPS, FTP либо размещённое локально. Также поддерживается сетевая загрузка с использованием протокола PXE, импорт готовых образов дисков, полностью автоматическая установка операционной системы с помощью kickstart-файлов или опции `--unattended`.

#### 14.4.1.1. Аргументы командной строки virt-install

У многих аргументов команды `virt-install` есть дополнительные параметры, которые указываются следующим образом: `--аргумент опция1=значение опция2=значение`. Используйте синтаксис `--аргумент=?` чтобы увидеть полный список таких параметров, например:

```
$ virt-install --disk=?
```

Большинство аргументов `virt-install` являются опциональными. В

случае задания значения опции `--os-variant` либо успешного автоматического определения типа гостевой системы, для таких аргументов будут использованы соответствующие значения по умолчанию, определённые профилем устанавливаемой операционной системы. Профили предоставляются пакетом `osinfo-db`. В случае отсутствия профиля для устанавливаемой ОС потребуется определить как минимум следующие опции: `--memory`, настройки хранилища (`--disk` или `--filesystem`) и метод установки (`--cdrom`, `--location`).

Аргументы, передаваемые утилите `virt-install`, можно условно сгруппировать по их назначению:

- Параметры подключения — определяют тип используемого гипервизора и путь (ссылку) для подключения к нему.
- Общие параметры — общие параметры, применимые ко всем типам гостевых систем.
- Параметры установки — определяют каким образом будет выполняться установка гостевой операционной системы.
- Параметры гостевой системы — задают тип устанавливаемой операционной системы либо управляют настройками автоматического определения типа.
- Параметры хранилища — опции, связанные с настройкой хранилища виртуальной машины.
- Параметры сети — опции, связанные с настройкой сети виртуальной машины;
- Параметры графики — опции, связанные с настройкой графической подсистемы виртуальной машины.
- Параметры виртуализации — опции для переопределения используемого механизма виртуализации.
- Параметры устройств — опции для подключения физических и виртуальных устройств к виртуальной машине.
- Другие опции — опции, не вошедшие ни в одну из предыдущих групп.

#### 14.4.1.2. Примеры использования virt-install

Следующая команда создаст в пользовательской сессии QEMU (см. «*14.3.2. Пользовательский режим*») виртуальную машину `msvsphere-9-server` с двумя гигабайтами оперативной памяти, двумя виртуальными процессорами и виртуальным qcow2-диском объёмом двадцать гигабайт. Виртуальная машина будет запущена в режиме BIOS, в качестве установочного носителя будет использован ISO-образ `/srv/iso/MSVSphere-9.5-x86_64-server.iso`:

```
$ virt-install --name msvsphere-9-server \
      --cdrom /srv/iso/MSVSphere-9.5-x86_64-server.iso \
      --memory 2048 --vcpus 2 --disk size=20 --os-variant
 ↪msvsphere9
```

Установка операционной системы МСВСфера 9 в режиме UEFI с отключённой поддержкой Secure Boot, в качестве источника установки используется установочное дерево дистрибутива, размещённое на официальном зеркале:

```
$ virt-install --name msvsphere-9-server \
      --memory 2048 --vcpus 2 --disk size=20 --os-variant
 ↪msvsphere9 \
      --location https://repo1.msvsphere-os.ru/msvsphere/9/BaseOS/
 ↪x86_64/os/ \
      --boot uefi,loader=/usr/share/edk2/ovmf/OVMF_CODE.fd,loader_
 ↪ro=yes,loader_type=pflash,nvram_template=/usr/share/edk2/ovmf/
 ↪OVMF_VARS.fd,loader_secure=no
```

Следующая команда создаст виртуальную машину в системной сессии QEMU (см. «*14.3.3. Системный режим*») и выполнит автоматическую установку операционной системы МСВСфера 9 в режиме UEFI с включённой поддержкой Secure Boot, сценарий установки определён в kickstart-файле `msvsphere-9.ks`:

```
$ virt-install --name msvsphere-9-server --connect qemu:///system \
      --memory 2048 --vcpus 2 --disk size=20 --os-variant
 ↪msvsphere9 \
      --location https://repo1.msvsphere-os.ru/msvsphere/9/BaseOS/
 ↪x86_64/os/ \
      --boot uefi,loader=/usr/share/edk2/ovmf/OVMF_CODE.secboot.fd,
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪loader_ro=yes,loader_type=pflash,nvram_template=/usr/share/edk2/
 ↪ovmf/OVMF_VARS.secboot.fd,loader_secure=yes \
      --initrd-inject msvsphere-9.ks --extra-args "inst.ks=file:/
 ↪msvsphere-9.ks"
```

Пример kickstart-файла (`msvsphere-9.ks` в примере выше) для автоматической установки системы в минимальной конфигурации без графического интерфейса:

```
# путь к установочному дереву дистрибутива
url --url https://repo1.msvsphere-os.ru/msvsphere/9/BaseOS/x86_64/
 ↪kickstart/

# список репозиториев, которые необходимо подключить во время
 ↪установки
repo --name=BaseOS --baseurl=https://repo1.msvsphere-os.ru/
 ↪msvsphere/9/BaseOS/x86_64/os/
repo --name=AppStream --baseurl=https://repo1.msvsphere-os.ru/
 ↪msvsphere/9/AppStream/x86_64/os/

# выполнять установку в текстовом режиме
text
# не выполнять настройку графического сервера Xorg/Wayland
skipx
# автоматически принимать условия лицензии
eula --agreed
# не запускать ассистента по настройке во время первого запуска
firstboot --disabled

# использовать английский язык как во время установки, так и на
 ↪установленной
# системе. Дополнительно включить поддержку русского языка
lang en_US --addsupport=ru_RU
# настраивает раскладку клавиатуры, в данном случае будет
 ↪использоваться только
# английская в американском варианте
keyboard us
# установить часовой пояс в московское время (GMT+3), флаг --utc
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪указывает на
# то, что аппаратные часы хранят время в часовом поясе UTC
timezone Europe/Moscow --utc

# автоматически настроить сеть используя протокол DHCP
network --bootproto=dhcp
# включить брандмауэр и отрыть доступ по протоколу SSH
firewall --enabled --service=ssh
# отключить службу kdump и включить службы chronyd, rsyslog и sshd
services --disabled="kdump" --enabled="chronyd,rsyslog,sshd"
# включить SELinux
selinux --enforcing

# настроить вывод на последовательный порт чтобы можно было
 ↪подключаться к
# консоли виртуальной машины без графической сессии
bootloader --timeout=1 --append="console=tty0 console=ttyS0,
 ↪115200n8 no_timer_check crashkernel=auto net.ifnames=0"
# использовать только диск /dev/vda во время установки
ignoredisk --only-use=vda
# создать новую таблицу разделов на диске /dev/vda
clearpart --initlabel --drives=vda
# использовать автоматическую разбивку диска без отдельного раздела
 ↪/home
autopart --nohome

# заблокировать вход пользователем root
rootpw --lock
# создать пользователя msvsphere с паролем msvsphere, сделать его
# администратором путём добавления в группу wheel
user --groups="wheel" --name msvsphere --password="msvsphere"

# автоматически перезагрузить систему после завершения установки и
 ↪извлечь
# установочный носитель
reboot --eject

# блок %packages определяет какие пакеты, группы и модули
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪необходимо установить.
# опция --inst-langs определяет список языков, для которых
 ↪необходимо добавлять
# поддержку
%packages --inst-langs=en,ru
# установить пакеты из группы core
@core
# установить пакеты из группы guest-agents
@guest-agents
# установить пакеты самоидентификации серверного варианта МСВСфера
 ↪ОС
sphere-release-identity-server
sphere-release-server
sphere-release
%end

# отключить расширение kdump
%addon com_redhat_kdump --disable
%end
```

На системах с графическим интерфейсом после запуска команды `virt-install` автоматически запустится программа `virt-viewer`, с помощью которой вы сможете взаимодействовать с виртуальной машиной и выполнить установку операционной системы (см. рис. 25: Запуск virtviewer).

На системах без графического интерфейса вы можете использовать последовательную консоль, если устанавливаемая операционная система поддерживает такой режим. Для большинства ОС на базе GNU/Linux будет достаточно передать следующие аргументы команде `virt-install`:

```
$ virt-install ... -console pty,target_type=virtio --graphics none
 ↪\
      --serial pty --extra-args 'console=ttyS0,115200n8'
```

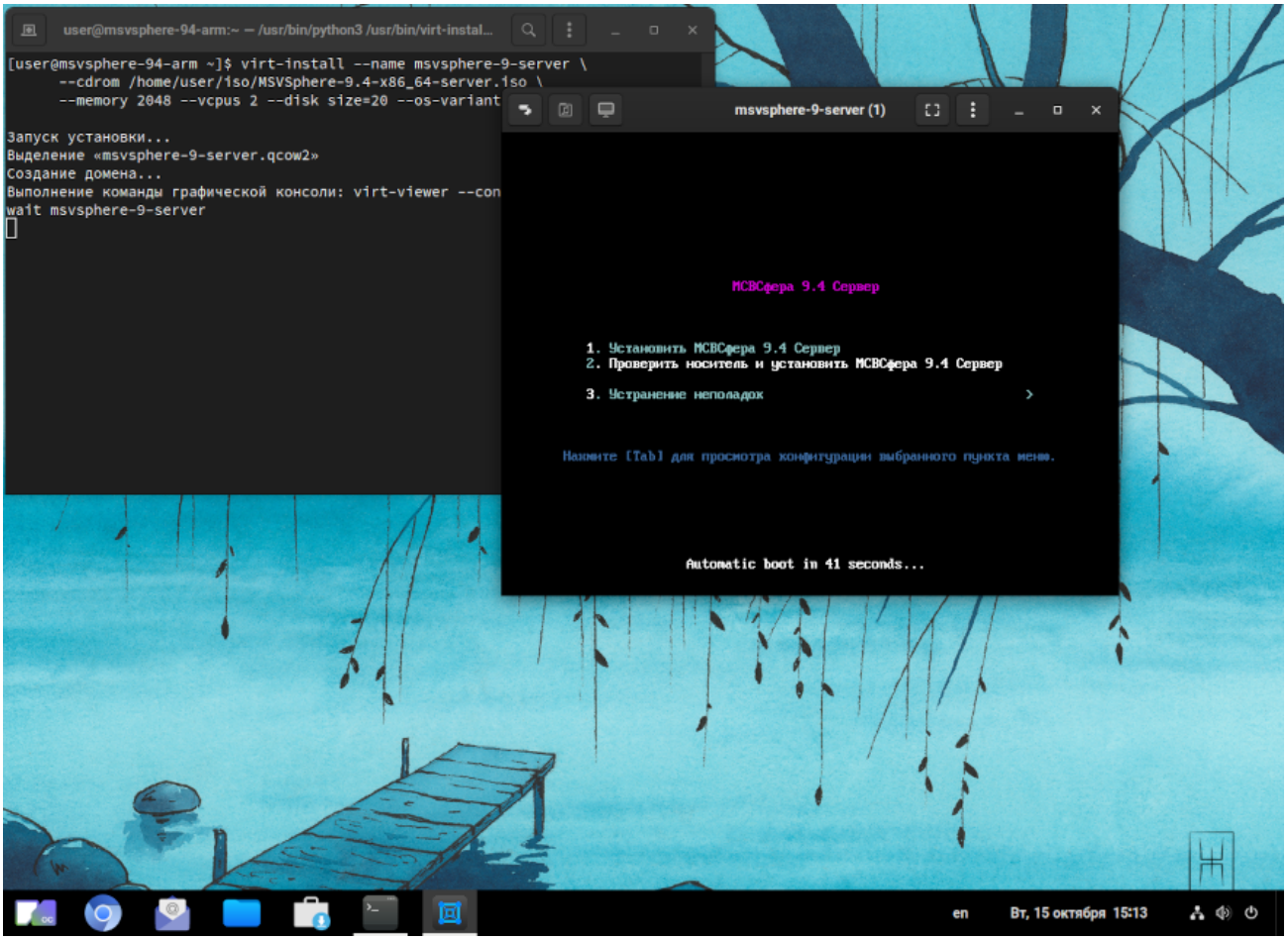



Рис. 25: Запуск virt-viewer

## 14.5. Запуск виртуальной машины

### 14.5.1. Запуск виртуальной машины с помощью командной строки

Для запуска ранее созданных (см. «*14.4. Создание виртуальной машины*») виртуальных машин из командной строки применяется команда `virsh start`. Ниже приведены типовые сценарии её использования:

- Запуск локальной виртуальной машины `msvsphere-9-arm`:

```
$ virsh start msvsphere-9-arm
Domain 'msvsphere-9-arm' started
```

- Запуск виртуальной машины `msvsphere-9-arm` на удалённом сервере `192.168.1.15` от имени пользователя `user`:

```
$ virsh -c qemu+ssh://user@192.168.1.15/session start
 ↪msvsphere-9-arm
Domain 'msvsphere-9-arm' started
```

### 14.6. Подключение к виртуальной машине

Системой виртуализации МСВСфера ОС поддерживается несколько способов подключения к виртуальной машине:

- подключение к графической консоли с помощью утилиты `virt-viewer`;

#### 14.6.1. Подключение к виртуальной машине с помощью virt-viewer

Одним из способов подключения к графической консоли виртуальной машины является использование утилиты `virt-viewer`. Ниже описаны типовые сценарии использования.

- Для подключения к локальной виртуальной машине с именем `msvsphere-9-arm` выполните команду:

```
$ virt-viewer msvsphere-9-arm
```

- Для подключения к удалённой виртуальной машине с именем `msvsphere-9-arm`, запущенной на сервере `192.168.1.15` от имени пользователя `user`, выполните команду:

```
$ virt-viewer --connect qemu+ssh://user@192.168.1.15/session
 ↪msvsphere-9-arm
```

### 14.7. Выключение виртуальной машины

#### 14.7.1. Выключение виртуальной машины средствами гостевой ОС

Если у вас есть доступ к виртуальной машине через графическую или текстовую консоль или по SSH, то вы можете воспользоваться штатными средствами операционной системы для её выключения.

В случае операционных систем на базе GNU/Linux это может быть:

- команда `shutdown -h now` или `poweroff`;

- функция выключения компьютера, встроенная в графическую оболочку.

Для операционной системы Windows:

- команда `shutdown /s`;
- соответствующий пункт в меню «Пуск».

### 14.7.2. Выключение виртуальной машины с помощью командной строки

Для штатного выключения работающей виртуальной машины используется команда `virsh shutdown`. Ниже приведены типовые сценарии её использования.

- Выключение локальной виртуальной машины `msvsphere-9-arm`:

```
$ virsh shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown
```

- Выключение виртуальной машины `msvsphere-9-arm` на удалённом сервере `192.168.1.15` от имени пользователя `user`:

```
$ virsh -c qemu+ssh://user@192.168.1.15/session shutdown
 ↪msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown
```

Для принудительного выключения виртуальной машины используется команда `virsh destroy`. Такое выключение можно сравнить с отключением компьютера от электрической цепи — при этом корректное выключение операционной системы не выполняется. Соответственно, использовать `virsh destroy` рекомендуется только в случаях когда виртуальная машина «зависла» и не отвечает на штатную процедуру выключения. Ниже приведено несколько примеров использования этой команды:

- Принудительное выключение локальной виртуальной машины `msvsphere-9-arm`:

```
$ virsh destroy msvsphere-9-arm
Domain 'msvsphere-9-arm' destroyed
```

- Принудительное выключение виртуальной машины `msvsphere-9-arm` на удалённом сервере `192.168.1.15` от имени пользователя `user`:

```
$ virsh -c qemu+ssh://user@192.168.1.15/session destroy
 ↪msvsphere-9-arm
Domain 'msvsphere-9-arm' destroyed
```

## 14.8. Управление конфигурацией виртуальной машины

### 14.8.1. Изменение объёма оперативной памяти

За объём оперативной памяти, доступной виртуальной машине, отвечают два основных параметра в её конфигурационном файле:

- `maxMemory` — задаёт максимальный объём оперативной памяти, доступный к использованию виртуальной машиной. Изменить это значение можно только для выключенной виртуальной машины.
- `memory` — задаёт фактический объём оперативной памяти, выделяемый виртуальной машине. Это значение может быть меньше значения `maxMemory`, но не должно превышать его. Допускается изменение «на лету» для работающей виртуальной машины.

Соответственно, для изменения объёма выделяемой виртуальной машине оперативной памяти необходимо модифицировать эти значения. Далее рассмотрим несколько способов решения этой задачи.

#### 14.8.1.1. Изменение объёма оперативной памяти с помощью командной строки

Посмотреть текущую конфигурацию виртуальной машины можно с помощью команды `virsh dominfo`. Ниже приведён пример подключения к системной сессии (см. «*14.3.3. Системный режим*») гипервизора и получение информации о виртуальной машине с именем `msvsphere-9-arm`:

```
$ virsh --connect qemu:///system dominfo msvsphere-9-arm
Id:             8
Name:           msvsphere-9-arm
UUID:           edb3edb0-a096-4f06-b153-8f77c6bdd38f
OS Type:        hvm
State:          running
CPU(s):         2
CPU time:       149.1s
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Max memory:     2097152 KiB
Used memory:    2097152 KiB
Persistent:     yes
Autostart:      enable
Managed save:   no
Security model: selinux
Security DOI:   0
Security label: system_u:system_r:svirt_t:s0:c36,c784 (enforcing)
```

В данном случае нас интересуют только значения `Max memory` и `Used memory`. В текущей конфигурации фактический объём выделенной памяти уже равен максимальному объёму (2 гибибайта или $1024^3$ байт), соответственно, увеличение «на лету» уже невозможно.

Выключим виртуальную машину:

```
$ virsh --connect qemu:///system shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown
```

Для увеличения максимального объёма оперативной памяти служит команда `virsh setmaxmem` — увеличим его до 8 гибибайт:

```
$ virsh --connect qemu:///system setmaxmem msvsphere-9-arm 8G --
 ↪config
```

Аргумент `--config` указывает на то, что необходимо внести изменения в конфигурационный файл, они будут применены во время следующего запуска виртуальной машины.

Удостоверимся что конфигурация была успешно изменена:

```
$ virsh --connect qemu:///system dominfo msvsphere-9-arm | grep
 ↪memory
Max memory:     8388608 KiB
Used memory:    2097152 KiB
```

Теперь мы можем увеличить объём фактически выделяемой оперативной памяти как «на лету» для запущенной виртуальной машины, так и для выключенной. За изменение этого параметра отвечает команда `virsh setmem`, увеличим его до 3 гибибайт и запустим виртуальную машину:

```
$ virsh --connect qemu:///system setmem msvsphere-9-arm 3G --config

$ virsh --connect qemu:///system dominfo msvsphere-9-arm | grep
 ↪memory
Max memory:     8388608 KiB
Used memory:    3145728 KiB

$ virsh --connect qemu:///system start msvsphere-9-arm
Domain 'msvsphere-9-arm' started
```

Как и ранее для команды `virsh setmaxmem`, опция `--config` указывает на то, что необходимо внести изменения в конфигурацию виртуальной машины.

Для изменения фактического объёма оперативной памяти на лету нужно использовать команду `virsh setmem` с аргументом `--live`:

```
$ virsh --connect qemu:///system setmem msvsphere-9-arm 4G --live

$ virsh --connect qemu:///system dominfo msvsphere-9-arm | grep
 ↪memory
Max memory:     8388608 KiB
Used memory:    4194304 KiB
```

Однако такое изменение будет действовать только до выключения виртуальной машины:

```
$ virsh --connect qemu:///system shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown

$ virsh --connect qemu:///system dominfo msvsphere-9-arm | grep
 ↪memory
Max memory:     8388608 KiB
Used memory:    3145728 KiB
```

Чтобы внести изменения и в конфигурационный файл, и в параметры работающей виртуальной машины используйте комбинацию аргументов `--live --config`:

```
$ virsh --connect qemu:///system setmem msvsphere-9-arm 4G --live -
 ↪-config
```

Сделанное таким образом изменение конфигурации сохранится и после выключения виртуальной машины.

#### 14.8.1.2. Изменение количества виртуальных процессоров

Количество виртуальных процессоров, выделенных для виртуальной машины, можно посмотреть с помощью команды `virsh vcpucount`:

```
$ virsh --connect qemu:///system vcpucount msvsphere-9-arm
maximum      config         2
maximum      live           2
current      config         2
current      live           2
```

Как и в случае с оперативной памятью, используются два параметра — максимальное количество процессоров и фактическое, при этом фактическое не может превышать максимальное.

Для увеличения максимального количества процессоров используется следующая команда:

```
$ virsh --connect qemu:///system setvcpus msvsphere-9-arm 4 --
 ↪maximum --config
```

Где аргумент `--maximum` указывает на то, что необходимо изменить именно максимальное количество процессоров, а `--config` — на то, что необходимо внести изменения в конфигурацию виртуальной машины. Изменение максимального количества процессоров «на лету» без остановки не поддерживается.

После выполнения предыдущей команды конфигурация виртуальной машины изменится следующим образом:

```
$ virsh --connect qemu:///system vcpucount msvsphere-9-arm
maximum      config         4
maximum      live           2
current      config         2
current      live           2
```

Остановим и снова запустим виртуальную машину:

```
$ virsh --connect qemu:///system shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown

$ virsh --connect qemu:///system start msvsphere-9-arm
Domain 'msvsphere-9-arm' started
```

И посмотрим как изменился вывод `virsh vcpucount`:

```
$ virsh --connect qemu:///system vcpucount msvsphere-9-arm
maximum      config         4
maximum      live           4
current      config         2
current      live           2
```

Значения опций `maximum config` и `maximum live` совпадают — теперь можно увеличивать фактическое количество выделяемых виртуальных процессоров.

Для изменения конфигурации «на лету» до следующего перезапуска виртуальной машины используется команда `virsh setvcpus` с аргументом `--live`:

```
$ virsh --connect qemu:///system setvcpus msvsphere-9-arm 3 --live

$ virsh --connect qemu:///system vcpucount msvsphere-9-arm
maximum      config         4
maximum      live           4
current      config         2
current      live           3
```

Для изменения постоянной конфигурации виртуальной машины выполните команду `virsh setvcpus` с аргументом `--config`:

```
$ virsh --connect qemu:///system setvcpus msvsphere-9-arm 3 --
 ↪config

$ virsh --connect qemu:///system vcpucount msvsphere-9-arm
maximum      config         4
maximum      live           4
current      config         3
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
current      live           3
```

Допускается одновременное использование аргументов `--live` и `--config`, чтобы изменить настройки уже запущенной виртуальной машины и её постоянную конфигурацию:

```
$ virsh --connect qemu:///system setvcpus msvsphere-9-arm 2 --
 ↪config --live

$ virsh --connect qemu:///system vcpucount msvsphere-9-arm
maximum      config         4
maximum      live           4
current      config         2
current      live           2
```

## 14.9. Резервное копирование

### 14.9.1. Резервное копирование настроек гипервизора

Системные настройки гипервизора `libvirt` хранятся в виде файлов в каталоге `/etc/libvirt`. Соответственно, для создания их резервной копии может быть применён любой инструмент резервного копирования, умеющий работать с файлами и каталогами. В состав операционной системы включены утилиты `tar`, `rsync` и система резервного копирования `bacula`.

Пример создания локальной резервной копии с помощью команды `tar`:

```
$ tar -cjpf "etc-libvirt.$(date --iso-8601).tar.bz2" /etc/libvirt
```

В результате выполнения в текущем каталоге будет создан файл `etc-libvirt-YYYY-MM-DD.tar.bz2` где `YYYY` — год, `MM` — месяц и `DD` — сегодняшнее число.

Автоматизировать создание резервных копий можно с помощью службы периодического выполнения заданий `cron` или таймеров `systemd`.

Например, для создания ежедневных архивов вы можете создать файл (в данном примере `libvirt-backup.sh`) в каталоге `/etc/cron.daily` следующего содержания:

```
#!/bin/bash

# прекратить выполнение если любая из команд вернёт ненулевой код
 ↪возврата
set -e

# сменить текущий каталог на /srv/backup
cd /srv/backup
# создать архив с конфигурационными файлами гипервизора libvirt
tar -cjpf "etc-libvirt.$(date --iso-8601).tar.bz2" /etc/libvirt
```

И сделать его исполняемым:

```
$ sudo chmod +x /etc/cron.daily/libvirt-backup.sh
```

После этого служба `cron` будет автоматически выполнять сценарий каждый день.

Для восстановления можно использовать следующую команду (замените `etc-libvirt.YYYY-MM-DD.tar.bz2` на реальное имя файла):

```
$ tar -C / -xpvf etc-libvirt.YYYY-MM-DD.tar.bz2
```

#### 14.9.1.1. Резервное копирование виртуальных машин

В данном разделе описаны различные подходы к резервному копированию виртуальных машин.

Для начала определимся с данными, которые подлежат резервному копированию:

- XML конфигурация виртуальной машины — содержит все метаданные, необходимые для запуска виртуальной машины: название и идентификатор, описание выделяемых ресурсов, виртуальных устройств и т.д.;
- диск (или несколько дисков, в зависимости от конфигурации) виртуальной машины — на нём хранятся разделы и файловые системы, с которыми работает виртуальная машина.

Имея резервную копию вышеперечисленных данных, виртуальную машину можно будет восстановить как на исходном узле гипервизора, так и на любом другом узле, предоставляющем совместимую конфигурацию.

#### 14.9.1.2. Ручное создание полной копии файлов виртуальной машины

Самым простым, но при этом самым неэффективным способом резервного копирования виртуальной машины является создание полной копии её дисков вручную:

**-достоинства:**

- простая процедура создания резервной копии;
- простая процедура восстановления;

**-недостатки:**

- требуется остановка виртуальной машины на время создания копии;
- каждая последующая резервная копия будет занимать такой же объём дискового пространства, как и исходные диск(и) виртуальной машины.

**-требования:**

- диски виртуальной машины хранятся в виде файлов образов в хранилище типа «каталог» (`dir`).

##### Создание резервной копии

Рассмотрим процедуру создания ежедневной резервной копии на примере виртуальной машины `msvsphere-9-arm`.

1. Создайте каталог для хранения резервной копии:

```
$ mkdir -p -v "/srv/backup/msvsphere-9-arm/$(date --iso-
↪8601)"
mkdir: created directory '/srv/backup/msvsphere-9-arm/2024-
↪12-16'
```

   Название каталога может быть любым, в этом примере используется вывод команды `date --iso-8601`, возвращающий текущую дату в формате `YYYY-MM-DD`.

2. Остановите виртуальную машину, в противном случае вы рискуете получить неконсистентные данные в резервной копии:

```
$ virsh --connect qemu:///system shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown
```

3. Получите список дисков виртуальной машины с помощью команды `virsh domblklist`:

```
$ virsh --connect qemu:///system domblklist msvsphere-9-arm
Target   Source
------------------------------------------------------------
vda      /var/lib/libvirt/images/msvsphere-9-arm.qcow2
sda      -
```

4. Сделайте копию каждого из дисков виртуальной машины в ранее созданный каталог:

```
$ cp -a /var/lib/libvirt/images/msvsphere-9-arm.qcow2 \
     /srv/backup/msvsphere-9-arm/2024-12-16/
```

5. Сделайте копию XML-конфигурации виртуальной машины:

```
$ virsh --connect qemu:///system dumpxml msvsphere-9-arm >
 ↪\
     /srv/backup/msvsphere-9-arm/2024-12-16/msvsphere-9-
 ↪arm.xml
```

6. Резервное копирование виртуальной машины завершено и теперь её можно включить:

```
$ virsh --connect qemu:///system start msvsphere-9-arm
Domain 'msvsphere-9-arm' started
```

Как и в случае с резервным копированием конфигурации гипервизора, для автоматизации можно применять shell-сценарий и `cron` или таймеры `systemd`. Ниже представлен пример такого сценария.

```
#!/bin/bash

set -eo pipefail

# список имён виртуальных машин, для которых необходимо выполнять
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪резервное
# копирование
VM_NAMES=('msvsphere-9-arm' 'msvsphere-9-server')
# каталог для хранения резервных копий
BACKUP_DIR='/srv/backup'
TODAY="$(date --iso-8601)"

# функция возвращает текущее состояние виртуальной машины
get_vm_state() {
    local -r vm_name="${1}"
    virsh --connect qemu:///system domstate "${vm_name}"
}

# функция выполняет команду выключения виртуальной машины и ожидает
# соответствующего изменения её статуса
shutdown_vm() {
    local -r vm_name="${1}"
    local -r max_try=10
    local vm_state
    local cur_try=0
    local output
    if ! output="$(virsh --connect qemu:///system shutdown ${vm_
 ↪name} 2>&1)"; then
        echo "VM ${vm_name} shutdown request failed: ${output}" >&2
        return 1
    fi

    while [ ${cur_try} -lt ${max_try} ]; do
        vm_state="$(get_vm_state ${vm_name})"
        if [ "${vm_state}" == 'shut off' ]; then
            return 0
        else
            cur_try=$((cur_try + 1))
            sleep 5
        fi
    done
    echo "VM ${vm_name} shutdown timeout, state: ${vm_state}" >&2
    return 1
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
}

# функция выполняет резервное копирование всех дисков и
 ↪конфигурационного файла
# виртуальной машины
backup_vm() {
    local -r vm_name="${1}"
    local -r backup_dir="${BACKUP_DIR}/${vm_name}/${TODAY}"
    local disk

    mkdir -p "${backup_dir}"

    for disk in $(virsh --connect qemu:///system domblklist "${vm_
 ↪name}" --details | awk '/disk/{print $4}'); do
        echo "Backing up VM ${vm_name} disk ${disk}" >&2
        ln -s "${disk}" "${backup_dir}/"
        echo "Backing up VM ${vm_name} configuration" >&2
        virsh --connect qemu:///system dumpxml "${vm_name}" > "$
 ↪{backup_dir}/${vm_name}.xml"
    done
}

# обработка заданных виртуальных машин в цикле
for vm_name in "${VM_NAMES[@]}"; do
    echo "Processing VM ${vm_name}" >&2
    vm_state="$(get_vm_state ${vm_name})"
    if [ "${vm_state}" == 'running' ]; then
        # выключение виртуальной машины если она запущена
        echo "VM ${vm_name} is running, shutting it off" >&2
        shutdown_vm "${vm_name}"
        echo "VM ${vm_name} has been shutted off"
    elif [ "${vm_state}" != 'shut off' ]; then
        # поддерживаются только два статуса виртуальной машины:
 ↪running и
        # shut off
        echo "VM ${vm_name} state is not supported: ${vm_state}" >&
 ↪2
        exit 1
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    fi
    # вызов функции резервного копирования
    backup_vm "${vm_name}"
    # включение виртуальной машины, которая была запущена перед
 ↪началом
    # процедуры резервного копирования
    if [ "${vm_state}" == 'running' ]; then
        echo "Starting VM ${vm_name} back" >&2
        virsh --connect qemu:///system start "${vm_name}"
    fi
done
```

### Восстановление из резервной копии

Для восстановления виртуальной машины из полной копии необходимо выполнить следующие операции.

1. Если виртуальная машина в данный момент запущена, то необходимо её остановить:

```
$ virsh --connect qemu:///system shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown
```

2. Восстановите конфигурацию виртуальной машины из сохранённого XML-файла:

```
$ virsh --connect qemu:///system define /srv/backup/
 ↪msvsphere-9-arm/2024-12-16/msvsphere-9-arm.xml
Domain 'msvsphere-9-arm' defined from /srv/backup/
 ↪msvsphere-9-arm/2024-12-16/msvsphere-9-arm.xml
```

3. Восстановите образ диска виртуальной машины из резервной копии:

```
$ sudo /usr/bin/cp /srv/backup/msvsphere-9-arm/2024-12-16/
 ↪msvsphere-9-arm.qcow2 \
      /var/lib/libvirt/images/msvsphere-9-arm.qcow2
```

Если вы забыли в каком каталоге должен находиться файл или как он должен называться, то эта информация доступна в блоке `disk` XML-файла с конфигурацией виртуальной машины:

```
<domain type='kvm'>
  ...
  <devices>
    ...
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2' discard='unmap'/>
      <source file='/var/lib/libvirt/images/msvsphere-9-
 ↪arm.qcow2'/>
     <target dev='vda' bus='virtio'/>
     <address type='pci' domain='0x0000' bus='0x04' slot=
 ↪'0x00' function='0x0'/>
   </disk>
   ...
  </devices>
  ...
</domain>
```

4. Восстановите владельца и права доступа к файлу образа диска:

```
$ sudo chown qemu:qemu /var/lib/libvirt/images/msvsphere-9-
 ↪arm.qcow2
$ sudo chmod 600 /var/lib/libvirt/images/msvsphere-9-arm.
 ↪qcow2
```

5. После этого восстановленную виртуальную машину можно запускать и продолжать работу:

```
$ virsh --connect qemu:///system start msvsphere-9-arm
Domain 'msvsphere-9-arm' started
```

В случае необходимости, вы можете внести необходимые правки в XML-конфигурацию перед её загрузкой в гипервизор. Например, вам может потребоваться изменить имя (тег `name`) виртуальной машины если вы восстанавливаете её на другом узле, где уже существует виртуальная машина с таким именем.

## 14.10. Удаление виртуальной машины

### 14.10.1. Удаление виртуальной машины с помощью командной строки

Для удаления виртуальных машин из командной строки служит команда `virsh undefine`, которая принимает следующие аргументы:

```
$ virsh undefine <domain> [--managed-save] [--storage <string>] \
    [--remove-all-storage] [--delete-storage-volume-snapshots] \
    [--wipe-storage] [--snapshots-metadata] [--checkpoints-
↪metadata] \
    [--nvram] [--keep-nvram] [--tpm] [--keep-tpm]
```

Таблица 44 - Аргументы команды virsh undefine

| Аргумент | Описание |
|---|---|
| `<domain>` | Название виртуальной машины или её уникальный идентификатор ID/UUID. |
| `--managed-save` | Удалить файл сохранённого состояния виртуальной машины (см. описание команды `virsh managedsave`). |
| `--storage <string>` | При использовании этой опции через запятую перечисляются имена томов хранилища для удаления. |
| `--remove-all-storage` | Помимо удаления виртуальной машины также удалить все связанные с ней тома хранилища. Используйте этот аргумент только если другие виртуальные машины не используют тома, связанные с удаляемой машиной. |
| `--delete-storage-volume-snapshots` | Кроме удаления связанных с виртуальной машиной томов хранилища также удалить все снимки дисков этих томов. Используется только вместе с `--remove-all-storage`. На текущий момент только драйвер хранилища `rbd` (Ceph RBD) поддерживает эту функциональность. |

| Аргумент | Описание |
|---|---|
| `--wipe-storage` | Перед удалением томов хранилища перезаписать на них данные случайной битовой последовательностью. Перезапись данных осуществляется за один проход, что затруднит восстановление удалённых данных, но не сделает эту процедуру невозможной. Libvirt поддерживает более надёжные и, соответственно, более медленные методы затирания данных в томах через команду `virsh vol-wipe`. |
| `--snapshots-metadata` | Удалить все метаданные снимков виртуальной машины. |
| `--checkpoints-metadata` | Удалить все метаданные точек восстановления виртуальной машины. |
| `--nvram` | Удалить файл NVRAM. |
| `--keep-nvram` | Не удалять файл NVRAM. |
| `--tpm` | Удалить состояние TPM модуля. |
| `--keep-tpm` | Не удалять состояние TPM модуля. |

Примеры использования команды `virsh undefine`:

- Удаление локальной виртуальной машины `msvsphere-9-arm`, всех ассоциированных с ней томов хранилищ и файл NVRAM:

```
$ virsh undefine msvsphere-9-arm --remove-all-storage --
 ↪nvram
Domain 'msvsphere-9-arm' has been undefined
Volume 'vda'(/home/user/.local/share/libvirt/images/
 ↪msvsphere-9-arm.qcow2) removed.
```

- Удаление виртуальной машины `msvsphere-9-arm` на удалённом сервере `192.168.1.15` от имени пользователя `user`:

```
$ virsh -c qemu+ssh://user@192.168.1.15/session undefine
 ↪msvsphere-9-arm \
    --remove-all-storage --nvram
Domain 'msvsphere-9-arm' has been undefined
Volume 'vda'(/home/user/.local/share/libvirt/images/
 ↪msvsphere-9-arm.qcow2) removed.
```

### 14.11. Миграция виртуальной машины

Миграция — это процесс перемещения виртуальной машины с одного узла гипервизора на другой. Необходимость в таком перемещении возникает в следующих случаях:

- Когда узлы гипервизора загружены неравномерно и нужно перераспределить нагрузку.
- Когда один из узлов гипервизора необходимо остановить для проведения обслуживания.

#### 14.11.1. Типы миграции

Гипервизором в операционной системе МСВСфера поддерживаются следующие типы миграции:

1. Миграция без остановки виртуальной машины («живая» миграция) — используется для миграции виртуальных машин, для которых установлено минимальное время простоя. В процессе такой миграции виртуальная машина продолжает работать на исходном узле гипервизора, в то время как KVM передает страницы памяти исходной машины на целевой хост. Когда миграция почти завершена, KVM приостанавливает исходную виртуальную машину и возобновляет её запуск на целевом узле гипервизора. Виртуальные машины, которые изменяют страницы памяти быстрее, чем KVM может их передать, например, такие, как виртуальные машины с большой нагрузкой ввода-вывода, не могут быть перенесены в этом режиме. Для таких виртуальных машин необходимо использовать миграцию с остановкой виртуальной машины.
2. Миграция с остановкой виртуальной машины. В процессе миграции виртуальная машина останавливается, на целевой узел гипервизора копируется её конфигурация и оперативная память, после чего работа

виртуальной машины возобновляется. Рекомендуется для виртуальных машин, которые активно используют оперативную память. Данный тип миграции более надёжен, чем «живая» миграция, но при этом возникает время простоя.

3. Автономная миграция — применяется когда виртуальная машина остановлена. При этом на целевой гипервизор копируется конфигурация виртуальной машины и, при необходимости, образ виртуального диска.

Для первых двух типов миграции необходимо, чтобы образ диска виртуальной машины находился на общем сетевом хранилище (например, сервере NFS). Процедура настройки NFS-сервера описана в конце этого раздела. Для третьего типа миграции необходимости в общем хранилище нет, файлы конфигурации виртуальной машины и образа диска можно перенести любыми доступными средствами, включая утилиты `scp`, `rsync` и др..

### 14.11.2. Предварительные условия

- Виртуальная машина не должна использовать переадресацию (проброс) физических USB, PCI, vGPU и других устройств, иначе миграция будет невозможна.
- Чтобы виртуальная машина не утратила доступ к сети, оба узла гипервизора должны иметь одинаковое имя сетевого моста (network bridge), который обеспечивает подключение виртуальных машин к сети.
- Для «живой» миграции оба узла гипервизора должны иметь центральные процессоры одинаковой архитектуры и производителя: либо только Intel, либо только AMD. Хотя процессоры от Intel и AMD поддерживают одинаковый набор команд, «живая» миграция между ними невозможна.

### 14.11.3. Развёртывание хранилища NFS

Самым простым способом реализовать общее сетевое хранилище между несколькими узлами гипервизора является развёртывание сервера NFS на отдельном физическом или виртуальном сервере.

#### 14.11.3.1. Настройка сервера NFS

Установите пакет `nfs-utils`:

```
$ sudo dnf install nfs-utils
```

Откройте необходимые порты в брандмауэре:

```
$ sudo firewall-cmd --permanent --add-service=nfs
$ sudo firewall-cmd --permanent --add-service=mountd
$ sudo firewall-cmd --permanent --add-service=rpc-bind
$ sudo firewall-cmd --permanent --add-port=2049/tcp
$ sudo firewall-cmd --permanent --add-port=2049/udp
$ sudo firewall-cmd --reload
```

Запустите службу NFS:

```
$ sudo systemctl enable --now nfs-server
```

Для настройки на сервере NFS общего ресурса, в котором будут храниться образы дисков виртуальных машин, необходимо создать каталог:

```
$ sudo mkdir –p /var/lib/libvirt/images
```

После чего отредактировать файл `/etc/exports`, создав в нём следующую строку:

```
/var/lib/libvirt/shared hv1.domain.local(rw,no_root_squash) hv2.
 ↪domain.local(rw,no_root_squash)
```

Где `hv1.domain.local` и `hv2.domain.local` — доменные адреса узлов гипервизора. Вместо доменных адресов можно указывать IP-адреса. После этого нужно перезапустить службу NFS командой:

```
$ sudo systemctl reload nfs-server
```

Для проверки правильности настройки сервера нужно выполнить команду и проверить её вывод:

```
$ sudo exportfs
/var/lib/libvirt/shared
    192.168.10.251
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
/var/lib/libvirt/shared
    192.168.10.252
```

Где `192.168.10.251` и `192.168.10.252` — это IP-адреса узлов `hv1.domain.local` и `hv2.domain.local`.

#### 14.11.3.2. Настройка клиентов NFS

Для подключения общего ресурса к узлам гипервизора на каждом из них необходимо выполнить команду:

```
$ sudo mount nfs.domain.local:/var/lib/libvirt/shared /var/lib/
 ↪libvirt/images
```

Замените `nfs.domain.local` на реальное доменное имя или IP-адрес сервера NFS. Подразумевается, что изначально никаких образов в каталоге `/var/lib/libvirt/images` нет.

Чтобы подключение общего ресурса автоматически восстанавливалось после перезагрузки компьютера, необходимо добавить соответствующую строку в файл `/etc/fstab`:

```
nfs.domain.local:/var/lib/libvirt/shared /var/lib/libvirt/images
 ↪nfs defaults 0 0
```

Также на узлах гипервизора необходимо снять ограничение SELinux на использование подсистемой виртуализации libvirt хранилища NFS. Для этого необходимо установить значение переменной SELinux `virt_use_nfs` равным `1`, выполнив команду:

```
$ sudo setsebool virt_use_nfs 1
```

#### Настройка узлов гипервизора

В брандмауэре узлов гипервизора необходимо разрешить диапазон портов, которые используются службой libvirtd для передачи оперативной памяти виртуальных машин:

```
$ sudo firewall-cmd --add-port=49152-49215/tcp --permanent
$ sudo firewall-cmd --reload
```

### Миграция виртуальной машины с помощью командной строки

Для миграции виртуальных машин из командной строки служит команда `virsh migrate`. В данном разделе представлено несколько примеров её использования.

#### 14.11.3.3. Миграция запущенной виртуальной машины

Пример «живой» миграции запущенной виртуальной машины:

```
$ virsh migrate --persistent --live msvsphere9 qemu+ssh://hv2.
↪domain.local/system
```

где `msvsphere9` — название виртуальной машины, а `qemu+ssh://hv2.domain.local/system` — URL узла гипервизора, на который необходимо осуществить миграцию.

Если для утилиты virsh не указана опция `--verbose`, на консоль не выводится ничего кроме ошибок. Если процесс миграции уже запущен, для отображения статистики можно использовать команду `virsh domjobinfo`. Чтобы убедиться, что виртуальная машина была успешно перенесена, нужно снова выполнить команду `virsh list` на обоих узлах гипервизора. В случае успешной миграции она покажет, что виртуальная машина выполняется на узле `hv2.domain.local`.

#### 14.11.3.4. Миграция приостановленной виртуальной машины

Миграция приостановленной виртуальной машины `msvsphere9` с узла `hv1.domain.local` на узел `hv2.domain.local` с переносом её конфигурации:

```
$ virsh migrate msvsphere9 qemu+ssh://hv1.domain.local/system \
      qemu+ssh://hv2.domain.local/system
```

#### 14.11.3.5. Миграция остановленной виртуальной машины

Миграция остановленной виртуальной машины `msvsphere9` с узла `hv1.domain.local` на узел `hv2.domain.local`:

```
$ virsh migrate --offline --persistent --undefinesource msvsphere9
 ↪\
      qemu+ssh://hv1.domain.local/system qemu+ssh://hv2.domain.
 ↪local/system
```

### 14.12. Система управления доступом

#### 14.12.1. Введение

Система виртуализации МСВСфера ОС предоставляет гибкий механизм управления доступом основанный на политиках `polkit` для локальных и удалённых подключений к гипервизору, а также на основе политик `D-Bus` для управления правами доступа через веб-интерфейс системы управления `Cockpit`.

**Внимание:** функция управления доступом на основе политик поддерживается только для гипервизора, работающего в режиме «*14.3.3. Системный режим*».

#### 14.12.2. Управление политиками доступа polkit

##### 14.12.2.1. Включение драйвера контроля доступа polkit

За функцию контроля доступа к тем или иным API вызовам гипервизора `libvirt` отвечают подключаемые модули — драйверы.

В конфигурации по умолчанию используется драйвер-заглушка, который фактически не выполняет каких-либо проверок контроля доступа. Соответственно, любой пользователь, успешно прошедший аутентификацию, получает полный доступ ко всем функциям гипервизора без каких-либо ограничений.

При таких настройках использование гипервизора в системном режиме доступно следующим пользователям:

- `root` — глобальный администратор системы, ввод пароля не требуется.
- любой пользователь, добавленный в системную группу `libvirt`, ввод пароля не требуется.

- любой пользователь с правами администратора (группа `wheel` в конфигурации системы по умолчанию), при этом требуется ввод пароля.

В состав системы виртуализации входит драйвер контроля доступа на основе политик `polkit`, который позволяет администратору выполнить детализированную настройку правил доступа на основе встроенных в `libvirt` объектов и разрешений (см. «*14.12.3. Объекты и разрешения libvirt*»).

Для включения драйвера контроля доступа `polkit` раскомментируйте строку `#access_drivers = [ "polkit" ]` в следующих конфигурационных файлах:

- `/etc/libvirt/virtqemud.conf`
- `/etc/libvirt/virtinterfaced.conf`
- `/etc/libvirt/virtnetworkd.conf`
- `/etc/libvirt/virtnodedevd.conf`
- `/etc/libvirt/virtnwfilterd.conf`
- `/etc/libvirt/virtsecretd.conf`
- `/etc/libvirt/virtstoraged.conf`
- `/etc/libvirt/libvirtd.conf`
- `/etc/libvirt/virtproxyd.conf`

Изменения можно выполнить как вручную, так и с помощью следующей команды:

```
$ sudo find /etc/libvirt -name '*virt*d.conf' \
    -exec sed -i 's/#access_drivers = \[ "polkit" \]/access_drivers
↪= \[ "polkit" \]/g' {} +
```

Для активации драйвера контроля доступа `polkit` перезапустите соответствующие сервисы:

```
$ sudo systemctl restart virtqemud
$ sudo systemctl restart virtinterfaced
$ sudo systemctl restart virtnetworkd
$ sudo systemctl restart virtnodedevd
$ sudo systemctl restart virtnwfilterd
$ sudo systemctl restart virtsecretd
$ sudo systemctl restart virtstoraged
```

Если вы по каким-то причинам используете `virtproxyd`, то перезапустите

так же этот сервис:

```
$ sudo systemctl restart virtproxyd
```

#### 14.12.2.2. Разрешения для объектов libvirt

Объекты и разрешения libvirt (см. «*14.12.3. Объекты и разрешения libvirt*») преобразуются в названия действий (actions) polkit по следующей схеме:

```
org.libvirt.api.$object.$permission
```

где:

- `$object` — название объекта в API libvirt;
- `$permission` — название разрешения к API вызовам заданного объекта.

#### 14.12.2.3. Дополнительные атрибуты объектов libvirt

Чтобы сделать возможным описание правил контроля доступа для отдельных экземпляров объектов `libvirt` предоставляет для объектов следующие дополнительные атрибуты:

**`-connect` (`virConnectPtr`):**

- `connect_driver` — название драйвера подключения `libvirt`.

**`-domain` (`virDomainPtr`):**

- `connect_driver` — название драйвера подключения `libvirt`.
- `domain_name` — название виртуальной машины, уникальное в пределах данного узла гипервизора.
- `domain_uuid` — уникальный идентификатор (UUID) виртуальной машины, уникальный для всего кластера `libvirt`.

**`-interface` (`virInterfacePtr`):**

- `connect_driver` — название драйвера подключения `libvirt`.
- `interface_name` — название сетевого интерфейса, уникальное в пределах данного узла гипервизора.
- `interface_macaddr` - MAC адрес сетевого интерфейса.

**-network (virNetworkPtr):**

- connect_driver — название драйвера подключения libvirt.
- network_name — название сети, уникальное в пределах данного узла гипервизора.
- network_uuid — уникальный идентификатор (UUID) сетевого интерфейса, уникальный для всего кластера libvirt.

**-node-device (virNodeDevicePtr):**

- connect_driver — название драйвера подключения libvirt.
- node_device_name — название аппаратного устройства, уникальное в пределах данного узла гипервизора.

**-nwfilter (virNWFilterPtr):**

- connect_driver — название драйвера подключения libvirt.
- nwfilter_name — название сетевого фильтра, уникальной в пределах данного узла гипервизора.
- nwfilter_uuid — уникальный идентификатор (UUID) сетевого фильтра, уникальный для всего кластера libvirt.

**-secret (virSecretPtr):**

- connect_driver — название драйвера подключения libvirt.
- secret_uuid — уникальный идентификатор приватного ключа, уникальный для всего кластера libvirt.
- secret_usage_volume — название тома, к которому привязан данный приватный ключ, если такая привязка существует.
- secret_usage_ceph — название Ceph сервера, к которому привязан данный приватный ключ, если такая привязка существует.
- secret_usage_target — название iSCSI-таргета, к которому привязан данный приватный ключ, если такая привязка существует.

- `secret_usage_name` — название TLS-ключа, к которому привязан данный приватный ключ, если такая привязка существует.

**`-storage-pool` (`virStoragePoolPtr`):**

- `connect_driver` — название драйвера подключения `libvirt`.
- `pool_name` — название пула хранения, уникальное в пределах данного узла гипервизора.
- `pool_uuid` — уникальный идентификатор (UUID) пула хранения, уникальный для всего кластера `libvirt`.

**`-storage-vol` (`virStorageVolPtr`):**

- `connect_driver` — название драйвера подключения `libvirt`.
- `pool_name` — название пула хранения, в котором расположен данный том, уникальное в пределах данного узла гипервизора.
- `pool_uuid` — уникальный идентификатор (UUID) пула хранения, в котором расположен данный том, уникальный для всего кластера `libvirt`.
- `vol_name` — название тома хранения, уникальное в пределах пула хранения.
- `vol_key` — ключ тома, уникальный для всего кластера `libvirt`.

#### 14.12.2.4. Создание политик контроля доступа polkit

По умолчанию libvirt не предоставляет каких-либо правил для `polkit` кроме следующих:

- разрешения, доступные только для чтения (см. третью колонку в таблице разрешений в разделе «*14.12.3. Объекты и разрешения libvirt*» для каждого типа объектов), предоставляются всем пользователям;
- все остальные операции запрещены.

Политики `polkit` представляют собой код на языке программирования JavaScript, размещённый в виде файлов с расширением `.rules` в каталоге `/etc/polkit-1/rules.d`. Соответственно, для определения собственных правил

системному администратору необходимо добавить в этот каталог новый файл, допустим, `99-libvirt-fstec.rules`.

Подробное описание процесса создания собственных политик `polkit` доступно в соответствующей документации (`man 8 polkit`), также доступна веб-версия на официальном сайте проекта: polkit.8.html.

В данном разделе мы разберём лишь один из примеров практического применения политик `polkit` для `libvirt`.

### 14.12.3. Объекты и разрешения libvirt

#### 14.12.3.1. Описание объектной модели API libvirt

Система виртуализации `libvirt` использует объектно-ориентированную модель для предоставления доступа к ресурсам гипервизора через API. Соответственно, каждый тип ресурса, будь то виртуальная машина, сетевой интерфейс, пул хранения и т.д., представлен в виде объекта и набора API-вызовов для работы с этим объектом/ресурсом. Даже само подключение к гипервизору является таким объектом.

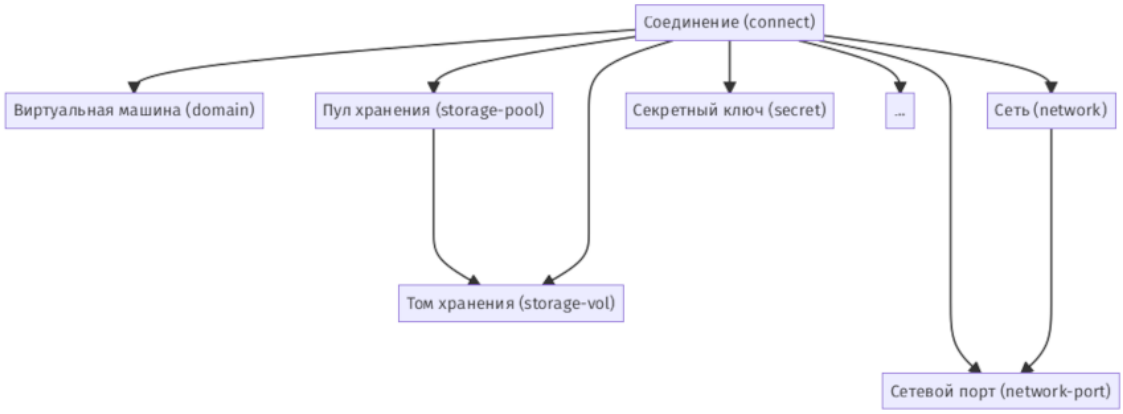



Рис. 26: Описание объектной модели API libvirt

В свою очередь, для каждого API-вызова объекта определяется свой набор разрешений (прав), которые должны быть присвоены пользователю системным администратором чтобы он мог выполнить данный API вызов.

#### 14.12.3.2. Типы объектов

#### connect (virConnectPtr)

Объект `connect` используется API для подключения к гипервизору.

Таблица 45 - Список разрешений для объекта `connect`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `detect-storage-pools` | Обнаружение доступных хранилищ. | Нет |
| `getattr` | Подключение к гипервизору и получение информации о системе. | |
| `interface-transaction` | Управление транзакциями при изменении сетевых интерфейсов. | Нет |
| `pm-control` | Управление питанием узла (сервера) гипервизора и гостевых машин. | Нет |
| `read` | Получение информации об узле (сервере) гипервизора. | Да |
| `search-domains` | Получение списка виртуальных машин. | Да |
| `search-interfaces` | Получение списка сетевых интерфейсов. | Да |
| `search-networks` | Получение списка сетей. | Да |
| `search-node-devices` | Получение списка оборудования. | Да |
| `search-nwfilter-bindings` | Получение списка привязок сетевых фильтров к сетевым интерфейсам. | Да |
| `search-nwfilters` | Получение списка сетевых фильтров. | Да |
| `search-secrets` | Получение списка секретов. | Да |
| `search-storage-pools` | Получение списка пулов хранилищ. | Да |
| `write` | Управление параметрами узла (сервера) гипервизора. | Нет |

### domain (virDomainPtr)

Объект `domain` используется API для управления виртуальными машинами.

Таблица 46 - Список разрешений для объекта `domain`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `block-read` | Чтение содержимого диска виртуальной машины. | Нет |
| `block-write` | Изменение диска виртуальной машины. | Нет |
| `checkpoint` | Управление точками восстановления виртуальной машины. | Нет |
| `core-dump` | Управление дампами памяти виртуальной машины. | Нет |
| `delete` | Удаление виртуальной машины. | Нет |
| `fs-freeze` | Управление «заморозкой» файловых систем виртуальной машины. | Нет |
| `fs-trim` | Использование команды TRIM для дисков виртуальной машины. | Нет |
| `getattr` | Получение списка виртуальных машин. | Да |
| `read` | Получение информации о виртуальных машинах. | Да |
| `hibernate` | Управление спящим режимом виртуальной машины. | Нет |
| `init-control` | Выключение и перезагрузка виртуальной машины. | Нет |
| `inject-nmi` | Разрешает отправку NMI прерывания виртуальной машине. | Нет |
| `mem-read` | Разрешает чтение оперативной памяти виртуальной машины. | Нет |

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `migrate` | Управление миграцией виртуальной машины на другой узел гипервизора. | Нет |
| `open-device` | Разрешает работу с каналами, последовательными терминалами, последовательными и параллельными портами, подключенными к виртуальной машине. | Нет |
| `open-graphics` | Разрешает подключение к графическому дисплею виртуальной машины. | Нет |
| `open-namespace` | Разрешает работу с пространствами имён виртуальной машины (LXC-контейнера). | Нет |
| `pm-control` | Управление питанием виртуальной машины. | Нет |
| `read` | Чтение настроек виртуальной машины, не связанных с безопасностью. | Да |
| `read-secure` | Чтение настроек виртуальной машины, связанных с безопасностью. | Нет |
| `reset` | Принудительная перезагрузка (аналог аппаратной кнопки reset) виртуальной машины. | Нет |
| `save` | Изменение конфигурационного файла виртуальной машины. | Нет |
| `screenshot` | Создание снимка экрана виртуальной машины. | Нет |

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `send-input` | Клавиатурный ввод через виртуализированную клавиатуру виртуальной машины. | Нет |
| `send-signal` | Отправка сигналов процессам внутри виртуальной машины. | Нет |
| `set-password` | Установка пароля пользователю внутри виртуальной машины. | Нет |
| `set-time` | Установка времени на виртуальной машине. | Нет |
| `snapshot` | Создание снимков (snapshots) виртуальной машины. | Нет |
| `start` | Запуск виртуальной машины. | Нет |
| `stop` | Остановка виртуальной машины. | Нет |
| `suspend` | Приостановка виртуальной машины. | Нет |
| `write` | Изменение виртуальной машины. | Нет |

### interface (virInterfacePtr)

Объект `interface` используется API для управления сетевыми интерфейсами.

Таблица 47 - Список разрешений для объекта `interface`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `delete` | Удаление сетевого интерфейса. | Нет |
| `getattr` | Получение списка сетевых интерфейсов. | Да |
| `read` | Получение информации о сетевом интерфейсе. | Да |

| Разрешение | Описание | Только для чтения |
|---|---|---|
| save | Добавление сетевого интерфейса или изменение его конфигурации. | Нет |
| start | Активация сетевого интерфейса. | Нет |
| stop | Деактивация (остановка) сетевого интерфейса. | Нет |
| write | Изменение конфигурации сетевого интерфейса. | Нет |

### network (virNetworkPtr)

Объект network используется API для управления сетями.

Таблица 48 - Список разрешений для объекта network

| Разрешение | Описание | Только для чтения |
|---|---|---|
| delete | Удаление сети. | Нет |
| getattr | Получение списка сетей. | Да |
| read | Чтение настроек сети. | Да |
| save | Добавление сети или изменение её конфигурации. | Нет |
| search-ports | Получение списка сетевых портов. | Нет |
| start | Активация сети. | Нет |
| stop | Деактивация сети. | Нет |
| write | Изменение конфигурации сети. | Нет |

### network-port (virNetworkPortPtr)

Объект network-port используется API для управления сетевыми портами.

Таблица 49 - Список разрешений для объекта network-port

| Разрешение | Описание | Только для чтения |
|---|---|---|
| create | Создание нового сетевого порта. | Нет |

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `delete` | Удаление сетевого порта. | Нет |
| `getattr` | Получение списка сетевых портов. | Да |
| `read` | Чтение настроек сетевого порта. | Да |
| `write` | Изменение конфигурации сетевого порта. | Нет |

### node-device (virNodeDevicePtr)

Объект `node-device` используется API для управления аппаратными устройствами, подключенными к узлу гипервизора.

Таблица 50 - Список разрешений для объекта `node-device`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `delete` | Удаление конфигурации устройства на хост системе. | Нет |
| `detach` | Отсоединение устройства от хост системы. | Нет |
| `getattr` | Получение списка устройств. | Да |
| `read` | Чтение информации об устройстве. | Да |
| `start` | Запуск устройства. | Нет |
| `stop` | Остановка устройства. | Нет |
| `write` | Изменение конфигурации устройства. | Нет |

### nwfilter (virNWFilterPtr)

Объект `nwfilter` используется API для управления сетевыми фильтрами (брандмауэром).

Таблица 51 - Список разрешений для объекта `nwfilter`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `delete` | Удаление сетевого фильтра. | Нет |
| `getattr` | Получение списка сетевых фильтров. | Да |
| `read` | Чтение настроек сетевого фильтра. | Да |
| `save` | Добавление нового сетевого фильтра или редактирование его конфигурации. | Нет |
| `write` | Изменение конфигурации сетевого фильтра. | Нет |

### nwfilter-binding (virNWFilterBindingPtr)

Объект `nwfilter-binding` используется API для привязки сетевых фильтров к сетевым интерфейсам.

Таблица 52 - Список разрешений для объекта `nwfilter-binding`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `create` | Привязывание сетевого фильтра к порту виртуальной машины. | |
| `delete` | Удаляет привязку сетевого фильтра к порту. | |
| `getattr` | Получение списка привязок сетевых фильтров к портам. | Да |
| `read` | Получение информации о сетевом фильтре. | Да |

### secret (virSecretPtr)

Объект `secret` используется API для работы с хранилищем секретных ключей `libvirt`. Это хранилище может использоваться для хранения ключей от LUKS или iSCSI разделов, пароля от виртуализированного TPM устройства, ключа от Ceph RBD и т.д..

Таблица 53 - Список разрешений для объекта `secret`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `delete` | Удаление секретного ключа. | |
| `getattr` | Получение списка секретных ключей. | |
| `read` | Получение информации о секретном ключе. | Да |
| `read-secure` | Получение (значения) секретного ключа. | |
| `save` | Создание нового секретного ключа или изменение значения существующего. | |
| `write` | Изменение значения секретного ключа. | |

### storage-pool (virStoragePoolPtr)

Объект `storage-pool` используется API для управления пулами хранения на гипервизоре.

Таблица 54 - Список разрешений для объекта `storage-pool`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `delete` | Удаление пула хранения. | Нет |
| `format` | Форматирование пула хранения или его удаление. | Нет |
| `getattr` | Получение списка пулов хранения. | Да |

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `read` | Получение информации о пуле хранения и его конфигурации. | Да |
| `refresh` | Обновление списка томов пула хранения. | Нет |
| `save` | Создание нового пула хранение или редактирование конфигурации существующего. | Нет |
| `search-storage-vols` | Получение списка томов, находящихся в пуле хранения. | Нет |
| `start` | Запуск пула хранения. | Нет |
| `stop` | Остановка пула хранения. | Нет |
| `write` | Изменение конфигурации пула хранения. | Нет |

### storage-vol (virStorageVolPtr)

Объект `storage-vol` используется API для управления томами в пулах хранения.

Таблица 55 - Список разрешений для объекта `storage-vol`

| Разрешение | Описание | Только для чтения |
|---|---|---|
| `create` | Создание тома. | Нет |
| `data-read` | Чтение данных из тома через поток. | Нет |
| `data-write` | Запись данных на том через поток. | Нет |
| `delete` | Удаление тома из пула хранения. | Нет |
| `format` | Форматирование тома. | Нет |
| `getattr` | Получение списка томов в пуле хранения. | Да |
| `read` | Получение информации о томе. | Да |
| `resize` | Изменение размера тома. | Нет |

## 14.13. Реализация ролевой модели управления доступом

### 14.13.1. Введение

В этом разделе описывается способ реализации ролевой модели управления доступом к функциям гипервизора.

Для средства виртуализации ОС должны быть реализованы четыре роли:

- разработчик виртуальной машины;
- администратор безопасности средства виртуализации;
- администратор средства виртуализации;
- администратор виртуальной машины.

В данном примере для реализации ролевой модели будут использоваться стандартные группы пользователей GNU/Linux. Соответственно, для назначения пользователю той или иной роли, системный администратор должен будет добавить пользователя в соответствующую роли группу или несколько групп, если пользователь выполняет несколько ролей.

### 14.13.2. Роль администратора средства виртуализации

Роль администратора средства виртуализации должна позволять:

- создавать учётные записи пользователей средства виртуализации;
- управлять учётными записями пользователей средства виртуализации;
- назначать права доступа пользователям средства виртуализации к виртуальным машинам;
- создавать и удалять виртуальное оборудование средства виртуализации;
- изменять конфигурации виртуального оборудования средства виртуализации;
- управлять доступом виртуальных машин к физическому и виртуальному оборудованию;
- управлять квотами доступа виртуальных машин к физическому и виртуальному оборудованию;
- управлять перемещением виртуальных машин;
- удалять виртуальные машины;
- запускать и останавливать виртуальные машины;
- создавать снимки состояния виртуальных машин, включающих файл конфигурации виртуальной машины, образа виртуальной машины и

образа памяти виртуальной машины.

Назовём соответствующую группу пользователей `libvirt-admin` и создадим её:

```
$ sudo groupadd --system libvirt-admin
```

Для работы с гипервизором `libvirt` используются системные учётные записи, соответственно, для реализации первых двух функций потребуется предоставить группе право на выполнение следующих команд:

- `/usr/sbin/useradd` — добавляет новую учётную запись пользователя, позволяет назначить пользователю группы и установить пароль;
- `/usr/sbin/usermod` — позволяет модифицировать учётную запись пользователя: заблокировать её, изменить домашний каталог, командную оболочку, изменить набор пользовательских групп и т.д.;
- `/usr/sbin/userdel` — удаляет учётную запись пользователя;
- `/usr/bin/passwd` — устанавливает пароль для учётной записи пользователя;
- `/usr/bin/gpasswd` — позволяет управлять группами пользователей: добавлять или удалять участников, ограничить доступ к группе и т.д.;
- `/usr/bin/chage` — устанавливает срок действия учётной записи и пароля пользователя.

Для предоставления группе `libvirt-admin` прав на запуск данных команд с правами администратора будет использоваться утилита `sudo`. Создайте файл `/etc/sudoers.d/libvirt-fstec` следующего содержания:

```
# запретить участникам группы libvirt-admin вызов любых команд
%libvirt-admin      ALL=(ALL)       !ALL

# разрешить участникам группы libvirt-admin вызов перечисленных
 ↪команд. При
# этом команда sudo попросит ввести пароль текущего пользователя
 ↪для
# авторизации.
%libvirt-admin      ALL=/usr/sbin/useradd, /usr/sbin/usermod, /usr/
 ↪sbin/userdel, /usr/bin/passwd, /usr/bin/gpasswd, /usr/bin/chage
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# если требуется использовать перечисленные команды без пароля, то
 ↪вместо
# предыдущей строки используйте следующую:
# %libvirt-admin      ALL=NOPASSWD: /usr/sbin/useradd, /usr/sbin/
 ↪usermod, /usr/sbin/userdel, /usr/bin/passwd, /usr/bin/gpasswd, /
 ↪usr/bin/chage
```

Установите для созданного файла соответствующие права доступа и владельца:

```
$ sudo chown root:root /etc/sudoers.d/libvirt-fstec
$ sudo chmod 440 /etc/sudoers.d/libvirt-fstec
```

Теперь перейдём к остальным функциям, перечисленным в техническом задании для роли администратора средства виртуализации. Для их реализации потребуется предоставить этой роли полный доступ ко всем API-вызовам гипервизора, для этого потребуется реализовать соответствующую `polkit` политику:

```
// объявляем переменную, значением которой будет название группы
 ↪пользователей,
// для которой реализуется роль администратора средства
 ↪виртуализации
libvirtAdminGroup = 'libvirt-admin';


// метод addRule объекта polkit добавляет новое правило в политику
 ↪управления
// доступом.

// Правила реализуются в виде функций, принимающих два аргумента:
//   action — действие, для которого выполняется проверка;
//   subject — информация о процессе, который запрашивает
 ↪разрешение на
//       выполнение действия. Среди прочей информации объект
 ↪subject содержит
//       идентификатор процесса (PID), имя пользователя, от
 ↪которого был запущен
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
//       процесс и список групп, в которые входит этот
 ↪пользователь.

// По результатам проверки функция должна вернуть одно из следующих
 ↪значений:
//   polkit.Result.YES — разрешить доступ;
//   polkit.Result.NO — запретить доступ;
//   polkit.Result.AUTH_SELF — запросить авторизацию от имени
 ↪пользователя,
//       запустившего сессию, и предоставить доступ в случае
 ↪успеха;
//   polkit.Result.AUTH_SELF_KEEP — то же самое, что и AUTH_SELF,
 ↪но авторизация
//       сохраняется (кешируется) на короткий промежуток времени;
//   polkit.Result.AUTH_ADMIN — запросить авторизацию от имени
 ↪пользователя с
//       правами администратора и предоставить доступ в случае
 ↪успеха;
//   polkit.Result.AUTH_ADMIN_KEEP — аналогично AUTH_ADMIN, но
 ↪авторизация
//       сохраняется (кешируется) на короткий промежуток времени;
//   polkit.Result.NOT_HANDLED — указывает на то, что данная
 ↪функция не
//       осуществляет запрошенную проверку. В таком случае polkit
 ↪запустит
//       проверку следующего правила. Это также является поведением
 ↪по
//       умолчанию, если функция не вернула ничего.
polkit.addRule(function(action, subject) {
    // действие org.libvirt.unix.manage запрашивается при
 ↪подключении к
    // локальному Unix сокету гипервизора libvirt
    if (action.id == 'org.libvirt.unix.manage') {
        // разрешить доступ без пароля для всех пользователей,
 ↪включённых в
        // группу администраторов средства виртуализации
        if (subject.isInGroup(libvirtAdminGroup)) {
            return polkit.Result.YES;
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
        }
        // запретить доступ для всех остальных пользователей
        else {
            return polkit.Result.NO;
        }
    }
    // данное правило обрабатывает только события, связанные с API-
↪вызовами
    // гипервизора libvirt
    else if (action.id.indexOf('org.libvirt.api.') != 0) {
        return polkit.Result.NOT_HANDLED;
    }

     // разрешить выполнение любого действия org.libvirt.api.*
↪пользователям в
    // группе администраторов средства виртуализации
    if (subject.isInGroup(libvirtAdminGroup)) {
        return polkit.Result.YES;
    }

    // запретить выполнение действий всем остальным пользователям
    return polkit.Result.NO;
});
```

Сохраните код политики в файл `/etc/polkit-1/rules.d/99-libvirt-fstec.rules` и установите для него корректные права:

```
$ sudo chown root:root /etc/polkit-1/rules.d/99-libvirt-fstec.rules
$ sudo chmod 644 /etc/polkit-1/rules.d/99-libvirt-fstec.rules
```

Служба `polkit` автоматически загрузит и применит обновлённые правила.

В совокупности с правилами для `sudo` данная политика позволяет роли администратора средства виртуализации выполнять все требуемые операции за исключением предоставления прав доступа пользователям средства виртуализации к виртуальным машинам — подход к реализации этого требования будет описан в конце этой главы.

Для назначения роли администратора средства виртуализации необходимо добавить пользователя в ранее созданную группу `libvirt-admin`:

```
# замените user на реальное имя пользователя
$ sudo gpasswd -a user libvirt-admin
```

Если пользователь уже вошёл в систему, то необходимо либо выйти и зайти заново, либо выполнить от его имени команду:

```
$ newgrp libvirt-admin
```

После этого пользователь получит соответствующие роли полномочия. В следующих разделах политика доступа к API гипервизора будет доработана.

### 14.13.3. Роль администратора виртуальной машины

Роль администратора виртуальной машины должна позволять осуществлять доступ пользователя средства виртуализации к виртуальной машине посредством интерфейса средства виртуализации.

По сути это означает возможность подключения к графической или терминальной сессии виртуальной машины используя утилиты `virt-manager`, `virsh console` или `virt-viewer`.

Назовём соответствующую группу пользователей `libvirt-user` и создадим её:

```
$ sudo groupadd --system libvirt-user
```

Для подключения к сессии виртуальной машины команды `virt-manager` и `virsh console` используют API гипервизора `libvirt`, ниже представлен минимально необходимый набор разрешений:

- `connect.getattr` — подключение к API гипервизора и получение информации о системе;
- `connect.read` — получение информации об узле гипервизора;
- `connect.search-domains` — получение списка доступных виртуальных машин (используется утилитой `virt-manager` и командой `virsh list`);
- `domain.getattr` — получение списка виртуальных машин;
- `domain.read` — получение информации о виртуальной машине;
- `domain.read-secure` — чтение настроек виртуальной машины, связанных с безопасностью (используется утилитой *virt-manager*);
- `domain.open-device` — подключение к каналам, последовательным

и параллельным портам виртуальной машины. Данное разрешение требуется для подключения к последовательному терминалу (serial console).

- `domain.open-graphics` — подключение к «аппаратному» графическому терминалу виртуальной машины (`virt-manager --show-domain-console`);

для предоставления этих разрешений группе `libvirt-user` потребуется внести соответствующие изменения в политику `polkit`:

```
libvirtAdminGroup = 'libvirt-admin';

// объявляем переменную, значением которой будет название группы
 ↪пользователей,
// для которой реализуется роль администратора (пользователя)
 ↪виртуальных машин
libvirtUserGroup = 'libvirt-user';

// список разрешений libvirt API для роли пользователя виртуальных
 ↪машин
libvirtUserActions = [
    'connect.getattr',
    'connect.read',
    'connect.search-domains',
    'domain.getattr',
    'domain.read',
    'domain.read-secure',
    'domain.open-device',
    'domain.open-graphics'
];

polkit.addRule(function(action, subject) {
    if (action.id == 'org.libvirt.unix.manage') {
        // разрешить подключение к API гипервизора через Unix сокет
 ↪группам,
        // указанным в переменных libvirtAdminGroup и
 ↪libvirtUserGroup
        if (subject.isInGroup(libvirtAdminGroup)
            || subject.isInGroup(libvirtUserGroup)) {
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
            return polkit.Result.YES;
        } else {
            return polkit.Result.NO;
        }
    } else if (action.id.indexOf('org.libvirt.api.') != 0) {
        return polkit.Result.NOT_HANDLED;
    }

    // заменить префикс libvirt API org.libvirt.api на пустую
↪строку чтобы не
    // приходилось его дублировать в списке разрешённых действий.
↪Таким образом
    // действие org.libvirt.api.connect.read преобразуется в
↪connect.read.
    var api = action.id.replace('org.libvirt.api.', '');

    if (subject.isInGroup(libvirtAdminGroup)) {
        return polkit.Result.YES;
    }
    // разрешить действие пользователю из группы libvirtUserGroup,
↪если это
    // действие перечислено в списке libvirtUserActions
    else if (subject.isInGroup(libvirtUserGroup)
             && libvirtUserActions.includes(api)) {
        return polkit.Result.YES;
    }

    return polkit.Result.NO;
});
```

Кроме команд `virt-manager` и `virsh console` для подключения к виртуальным машинам также может использоваться команда `virt-viewer`, которая, в отличии от двух других, подключается к виртуальной машине по протоколам VNC/SPICE без использования API `libvirt`. Соответственно, для ограничения возможности такого подключения администратор средства виртуализации должен либо отключить поддержку протоколов VNC/SPICE, либо установить пароль для подключения и предоставить его только тем пользователям, которые должны иметь возможность такого подключения.

### 14.13.4. Роль разработчика виртуальной машины

Роль разработчика виртуальной машины должна позволять:

- создавать виртуальные машины;
- изменять конфигурации виртуальных машин.

Назовём соответствующую данной роли группу пользователей `libvirt-vm-dev` и создадим её:

```
$ sudo groupadd --system libvirt-vm-dev
```

Как уже было рассмотрено ранее, основным способом создания виртуальных машин в среде виртуализации МСВСфера ОС является использование команды `virt-install`.

Полный список доступных для использования в политиках `polkit` разрешений `libvirt` доступен в разделе «*14.12.3. Объекты и разрешения libvirt*», а ниже представлен минимальный набор разрешений, необходимый для корректной работы команды `virt-install`:

- `connect.getattr` — подключение к API гипервизора и получение информации о системе;
- `connect.read` — получение информации об узле гипервизора;
- `connect.search-storage-pools` — получение списка доступных пулов хранения;
- `domain.getattr` — получение списка виртуальных машин;
- `domain.read` — получение информации о виртуальной машине;
- `domain.save` — изменение конфигурационного файла виртуальной машины;
- `domain.start` — запуск виртуальной машины;
- `domain.write` — изменение виртуальной машины;
- `network.getattr` — получение списка сетей;
- `network.read` — чтение настроек сети;
- `network-port.create` — создание сетевого порта;
- `network-port.delete` — удаление сетевого порта;
- `network-port.read` — получение информации о сетевом порте;
- `storage-pool.getattr` — получение списка пулов хранения;
- `storage-pool.read` — получение информации о пуле хранения;
- `storage-pool.refresh` — обновление списка томов хранения в пуле;

- `storage-pool.search-storage-vols` — получение списка томов хранения в пуле;
- `storage-vol.create` — создание тома хранения;
- `storage-vol.getattr` — получение списка томов в пуле хранения;
- `storage-vol.read` — получение информации о пуле хранения.

Однако, с вышеперечисленным набором функциональность будет крайне ограничена:

- будет невозможно использовать графическую утилиту для создания и настройки виртуальных машин `virt-manager`;
- функции выключения и перезагрузки виртуальной машины будут недоступны;
- в случае ошибки при создании виртуальной машины утилита `virt-install` не сможет автоматически удалить созданный в процессе том хранения.

Для решения вышеперечисленных проблем рекомендуется добавить следующие разрешения роли разработчика виртуальной машины:

- `domain.init-control` — перезагрузка или выключение виртуальной машины;
- `domain.read-secure` — чтение настроек виртуальной машины, связанных с безопасностью (используется утилитой `virt-manager`);
- `domain.open-device` — подключение к каналам, последовательным и параллельным портам виртуальной машины. Данное разрешение требуется для подключения к последовательному терминалу (serial console).
- `domain.open-graphics` — подключение к «аппаратному» графическому терминалу виртуальной машины (`virt-manager --show-domain-console`);
- `connect.search-domains` — получение списка доступных виртуальных машин (используется утилитой `virt-manager` и командой `virsh list`);
- `connect.search-networks` — получение списка доступных сетей (функция доступных сетей в `virt-manager` и работа команды `virsh net-list`);
- `storage-vol.delete` — удаление тома хранения. Является необходимым для автоматической очистки, если операция по созданию виртуальной

машины завершилась с ошибкой.

Создав группу пользователей и определившись с необходимыми разрешениями для доступа к API `libvirt`, доработаем политику `polkit` чтобы предоставить необходимые полномочия группе:

```
libvirtAdminGroup = 'libvirt-admin';

// объявляем переменную, значением которой будет название группы
 ↪пользователей,
// для которой реализуется роль разработчика виртуальных машин
libvirtVMDevGroup = 'libvirt-vm-dev';

libvirtUserGroup = 'libvirt-user';

libvirtUserActions = [
    'connect.getattr',
    'connect.read',
    'connect.search-domains',
    'domain.getattr',
    'domain.read',
    'domain.read-secure',
    'domain.open-device',
    'domain.open-graphics'
];

// список разрешений libvirt API для роли разработчика виртуальных
 ↪машин. За
// основу берётся список разрешений для пользователя чтобы избежать
// дублирования кода
libvirtVMDevActions = libvirtUserActions.concat([
    'connect.search-storage-pools',
    'connect.search-networks',
    'domain.init-control',
    'domain.save',
    'domain.start',
    'domain.write',
    'network.getattr',
    'network.read',
    'network-port.create',
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    'network-port.delete',
    'network-port.read',
    'storage-pool.getattr',
    'storage-pool.read',
    'storage-pool.refresh',
    'storage-pool.search-storage-vols',
    'storage-vol.create',
    'storage-vol.delete',
    'storage-vol.getattr',
    'storage-vol.read'
]);

polkit.addRule(function(action, subject) {
    if (action.id == 'org.libvirt.unix.manage') {
        // разрешить подключение к API гипервизора через Unix сокет
 ↪группам,
        // указанным в переменных libvirtAdminGroup,
 ↪libvirtVMDevGroup и
        // libvirtUserGroup
        if (subject.isInGroup(libvirtAdminGroup)
            || subject.isInGroup(libvirtVMDevGroup)
            || subject.isInGroup(libvirtUserGroup)) {
            return polkit.Result.YES;
        } else {
            return polkit.Result.NO;
        }
    } else if (action.id.indexOf('org.libvirt.api.') != 0) {
        return polkit.Result.NOT_HANDLED;
    }

    var api = action.id.replace('org.libvirt.api.', '');

    if (subject.isInGroup(libvirtAdminGroup)) {
        return polkit.Result.YES;
    }
    // разрешить действие пользователю из группы libvirtVMDevGroup
 ↪если это
    // действие перечислено в списке libvirtVMDevActions
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    else if (subject.isInGroup(libvirtVMDevGroup)
             && libvirtVMDevActions.includes(api)) {
        return polkit.Result.YES;
    }
    else if (subject.isInGroup(libvirtUserGroup)
             && libvirtUserActions.includes(api)) {
        return polkit.Result.YES;
    }

    return polkit.Result.NO;
});
```

На этом реализацию роли разработчика виртуальных машин можно считать завершённой.

### 14.13.5. Роль администратора безопасности средства виртуализации

Роль администратора безопасности средства виртуализации должна позволять:

- иметь доступ на чтение к журналу событий безопасности средства виртуализации;
- формировать отчёты с учетом заданных критериев отбора, выгрузку (экспорт) данных из журнала событий безопасности средства виртуализации.

Для работы с журналом событий безопасности пользователю достаточно иметь права на запуск следующих утилит от имени суперпользователя:

- `/usr/sbin/ausearch` — утилита для поиска событий в журналах безопасности службы `auditd`;
- `/usr/sbin/aureport` — утилита для построения отчётов о событиях безопасности на основе журналов безопасности службы `auditd`.

Документация по использованию данных утилит доступна в руководстве администратора МСВСфера ОС в главе «*6. РЕГИСТРАЦИЯ СОБЫТИЙ БЕЗОПАСНОСТИ*».

Как и в случае с ролью администратора средства виртуализации, для предоставления прав на запуск этих команд с полномочиями администратора

будет реализован через утилиту sudo.

Группа пользователей, выполняющих роль администратора безопасности средства виртуализации будет называться libvirt-sec-admin, создадим её:

```
$ sudo groupadd --system libvirt-sec-admin
```

Далее, добавьте в файл /etc/sudoers.d/libvirt-fstec следующие настройки (вероятно, вам потребуется использовать либо редактор visudo, либо временно изменить права доступа к файлу, поскольку ранее мы установили их в режим только для чтения владельцем и группой — 440):

```
# запретить участникам группы libvirt-sec-admin вызов любых команд
%libvirt-sec-admin      ALL=(ALL)       !ALL

# разрешить участникам группы libvirt-sec-admin вызов команд /usr/
 ↪sbin/ausearch
# и /usr/sbin/aureport. При этом команда sudo попросит ввести
 ↪пароль текущего
# пользователя для авторизации.
%libvirt-sec-admin      ALL=/usr/sbin/ausearch, /usr/sbin/aureport

# если требуется использовать команды /usr/sbin/ausearch и /usr/
 ↪sbin/aureport
# без пароля, то вместо предыдущей строки используйте следующую:
# %libvirt-sec-admin      ALL=NOPASSWD: /usr/sbin/ausearch, /usr/
 ↪sbin/aureport
```

Если для редактирования вы изменяли права доступа к файлу, то восстановите исходные значения:

```
$ sudo chown root:root /etc/sudoers.d/libvirt-fstec
$ sudo chmod 440 /etc/sudoers.d/libvirt-fstec
```

Далее, добавьте пользователя, который будет выполнять роль администратора безопасности средства виртуализации в группу libvirt-sec-admin:

```
# замените "user" на реальное имя пользователя
$ sudo gpasswd -a user libvirt-sec-admin
```

Если пользователь уже вошёл в систему, то необходимо либо выйти и зайти заново, либо выполнить от его имени команду:

```
$ newgrp libvirt-sec-admin
```

Теперь пользователь сможет работать с журналом безопасности используя команды `sudo ausearch` и `sudo aureport`.

Если вы хотите предоставить группе `libvirt-sec-admin` права на чтение файлов журналов службы аудита напрямую, то в конфигурационном файле `/etc/audit/auditd.conf` необходимо установить значение опции `log_group` равным имени группы:

```
log_group = libvirt-sec-admin
```

И подать службе `auditd` сигнал перечитать конфигурационный файл:

```
$ sudo auditctl --signal reload
```

После этого права на файлы журналов в каталоге `/var/log/audit` поменяются с `-rw-------. 1 root root` на `-rw-r-----. 1 root libvirt-sec-admin`:

```
$ sudo ls -la /var/log/audit/
drwxr-x---.  2 root libvirt-sec-admin    4096 окт 15 03:30 .
drwxr-xr-x. 16 root root                 4096 ноя 20 11:16 ..
-rw-r-----.  1 root libvirt-sec-admin 5994208 ноя 20 18:59 audit.
 ↪log
```

Таким образом пользователи, входящие в группу `libvirt-sec-admin`, получат возможность выполнять чтение файлов журналов аудита.

Если вам требуется предоставить пользователю доступ к журналам `systemd`, допустим, для просмотра логов Polkit или гипервизора Libvirt, то вы можете добавить пользователя в системную группу `systemd-journal`, которая предоставляет такую возможность:

```
# замените "user" на реальное имя пользователя
$ sudo gpasswd -a user systemd-journal
```

После этого пользователь получит возможность просматривать системные журналы с помощью команды `journalctl`.

### 14.13.6. Регистрация событий запуска и остановки виртуальных машин

Одной из важных особенностей гипервизора `libvirt`, работающего в системном режиме (см. «*14.3.3. Системный режим*»), является то, что в системном журнале событий безопасности службы `auditd` инициатором всех событий, связанных с запуском или остановом виртуальных машин, является пользователь `root`:

```
type=VIRT_CONTROL msg=audit(1733316312.117:1249): pid=10098 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm op=stop reason=shutdown vm="msvsphere-
 ↪9-server" uuid=9faaa743-e098-4925-8c4e-8854b88e7a25 vm-pid=0 exe=
 ↪"/usr/sbin/virtqemud" hostname=? addr=? terminal=? res=success'
```

Соответственно, из системного журнала невозможно узнать какой конкретно пользователь изменил состояние виртуальной машины.

Однако, для решения этой задачи можно применить политику `polkit`. Реализуем следующую функцию, которая будет выводить название и идентификатор виртуальной машины, а также имя пользователя, который вызвал изменение состояния виртуальной машины через API:

```
function logVMActions(action, subject, api) {
    var vmName = action.lookup('domain_name');
    var vmUUID = action.lookup('domain_uuid');

    var actionText;

    switch(api) {
    case 'domain.delete':
        actionText = 'deletion';
        break;
    case 'domain.init-control':
        actionText = 'reboot or shutdown';
        break;
    case 'domain.start':
    case 'domain.stop':
        actionText = api.split('.')[1];
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
        break;
    case 'domain.save':
    case 'domain.write':
        actionText = 'modification';
        break;
    }

    if (actionText) {
        polkit.log(`virtual machine "${vmName}" (UUID="${vmUUID}")
 ↪${actionText} initiated by "${subject.user}" user`);
    }
}
```

И добавим вызов этой функции перед возвратом из функции, осуществляющей проверку прав доступа:

```
polkit.addRule(function(action, subject) {
    if (action.id == 'org.libvirt.unix.manage') {
        if (subject.isInGroup(libvirtAdminGroup)
            || subject.isInGroup(libvirtVMDevGroup)
            || subject.isInGroup(libvirtUserGroup)) {
            return polkit.Result.YES;
        } else {
            return polkit.Result.NO;
        }
    } else if (action.id.indexOf('org.libvirt.api.') != 0) {
        return polkit.Result.NOT_HANDLED;
    }

    var api = action.id.replace('org.libvirt.api.', '');
    var result = polkit.Result.NO;

    if (subject.isInGroup(libvirtAdminGroup)) {
        result = polkit.Result.YES;
    } else if (subject.isInGroup(libvirtVMDevGroup)
               && libvirtVMDevActions.includes(api)) {
        result = polkit.Result.YES;
    } else if (subject.isInGroup(libvirtUserGroup)
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
                && libvirtUserActions.includes(api)) {
            result = polkit.Result.YES;
        }

        // вызвать функцию логирования изменения состояния виртуальной
 ↪машины в
        // случае предоставления доступа к API
        if (result == polkit.Result.YES) {
            logVMActions(action, subject, api);
        }

        return result;
});
```

Теперь при каждом изменении состояния или конфигурации виртуальной машины в журнал службы `polkit` (`journalctl -u polkit`) будет попадать соответствующее сообщение:

```
дек 04 15:47:40 msvsphere-94-arm.msvsphere.test polkitd[961]: <no
 ↪filename>:189: virtual machine "msvsphere-9-server" (UUID=
 ↪"9faaa743-e098-4925-8c4e-8854b88e7a25") start initiated by
 ↪"virtadmin" user
```

Теперь, получив идентификатор виртуальной машины из журнала `auditd` можно будет узнать по нему какие пользователи выполняли те или иные действия с этой виртуальной машиной.

### 14.13.7. Ограничение прав доступа к виртуальным машинам

Гипервизор `libvirt`, используемый в МСВСфера ОС, не предоставляет готового решения для проблемы ограничения прав доступа к виртуальным машинам, поскольку каждая организация использует свои правила и политики доступа.

Однако, поскольку для реализаций политик `polkit` используется язык программирования JavaScript, не составляет труда реализовать практически любую логику внутри политики.

Рассмотрим практическую реализацию на простом примере: предположим, что в организации существуют два отдела, которые используют виртуальные

машины — разработчики приложений и тестировщики. Сотрудникам из отдела разработки нужно предоставить доступ ко всем виртуальным машинам, имя которых начинается с префикса `dev-`, а тестировщикам — ко всем виртуальным машинам, имя которых начинается с `qa-`.

Для каждого отдела создадим соответствующую группу и добавим туда всех сотрудников:

```
# создаём группу для разработчиков devs и добавляем туда
 ↪пользователей devuser1
# и devuserN
$ sudo groupadd -U devuser1,devuserN devs

# создаём группу для тестировщиков qa и добавляем туда
 ↪пользователей qauser1 и
# qauserN
$ sudo groupadd -U qauser1,qauserN qa
```

Не забудьте добавить пользователей в группу `libvirt-user` чтобы предоставить им права на подключения к виртуальным машинам.

Теперь реализуем функцию для проверки доступа к виртуальным машинам:

```
function checkUserVMAccess(action, subject) {
    // сохраняем название виртуальной машины в переменную
    var vmName = action.lookup('domain_name');
    // если имя не определено, то разрешаем доступ — будут
 ↪применяться
    // ограничения роли
    if (!vmName) {
        return true;
    }

    // если название виртуальной машины начинается с "dev-" и
 ↪пользователь
    // входит в группу devs, то разрешаем доступ
    if (vmName.startsWith('dev-') && subject.isInGroup('devs')) {
        return true;
    }
    // если название виртуальной машины начинается с "qa-" и
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
↪пользователь
    // входит в группу qa, то разрешаем доступ
    else if (vmName.startsWith('qa-') && subject.isInGroup('qa')) {
        return true;
    }
    // запрещаем доступ в остальных случаях
    return false;
}
```

И включим эту функцию в общую логику проверки для роли пользователя виртуальных машин:

```
polkit.addRule(function(action, subject) {
    if (action.id == 'org.libvirt.unix.manage') {
        if (subject.isInGroup(libvirtAdminGroup)
            || subject.isInGroup(libvirtVMDevGroup)
            || subject.isInGroup(libvirtUserGroup)) {
            return polkit.Result.YES;
        } else {
            return polkit.Result.NO;
        }
    } else if (action.id.indexOf('org.libvirt.api.') != 0) {
        return polkit.Result.NOT_HANDLED;
    }

    var api = action.id.replace('org.libvirt.api.', '');
    var result = polkit.Result.NO;

    if (subject.isInGroup(libvirtAdminGroup)) {
        result = polkit.Result.YES;
    } else if (subject.isInGroup(libvirtVMDevGroup)
               && libvirtVMDevActions.includes(api)) {
        result = polkit.Result.YES;
    }
    // разрешить пользователю доступ к виртуальной машине если
↪проверка ролевых
    // полномочий завершилась успешно и функция checkUserVMAccess
↪вернула true
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
    else if (subject.isInGroup(libvirtUserGroup)
                && libvirtUserActions.includes(api)
                && checkUserVMAccess(action, subject)) {
        result = polkit.Result.YES;
    }

    if (result == polkit.Result.YES) {
        logVMActions(action, subject, api);
    }

    return result;
});
```

В результате применения новой политики обычные пользователи средства виртуализации получат доступ к виртуальной машине только в следующих случаях:

- пользователь входит в группы `libvirt-user` и `devs`, название виртуальной машины начинается с префикса `dev-`;
- пользователь входит в группы `libvirt-user` и `qa`, название виртуальной машины начинается с префикса `qa-`.

Для пользователей, входящих в группы `libvirt-admin` (администратор средства виртуализации) и `libvirt-vm-dev` (разработчик виртуальных машин), дополнительные ограничения применяться не будут.

### 14.13.8. Итоговая реализация в МСВСфера ОС

Описанная выше реализация ролевой модели на базе политик `polkit` поставляется в RPM-пакете `libvirt-fstec`. Перед его установкой вам необходимо включить драйвер контроля доступа `polkit` в настройках гипервизора — данная процедура описана в разделе «*14.12.2.1. Включение драйвера контроля доступа polkit*».

После включения поддержки `polkit` установите пакет `libvirt-fstec` с помощью следующей команды:

```
$ sudo dnf install -y libvirt-fstec
```

Пакет создаст в вашей системе группы `libvirt-vm-dev`, `libvirt-admin`,

`libvirt-user`, `libvirt-sec-admin` и файл `/etc/sudoers.d/libvirt-fstec`, содержащий описанные в этой главе правила `sudo` для реализации ролевой модели.

Так же в каталоге `/etc/polkit-1/rules.d` будут созданы следующие файлы:

- `97-libvirt-fstec-vars.rules` — в этом файле объявляются названия групп, соответствущие ролям, а также список действий, разрешённых для ролей разработчика и пользователя виртуальной машины.
- `99-libvirt-fstec.rules` — реализация правил для контроля доступа к API `libvirt`.

Вам, как системному администратору, потребуется самостоятельно создать файл `/etc/polkit-1/rules.d/98-libvirt-user-rules` и определить в нём собственную реализацию описанных ранее в этой главе функций `logVMActions` и `checkUserVMAccess`, соответствующую политике безопасности вашего предприятия и решаемой задаче. Реализация ролевой модели управления доступом сознательно разделена на несколько файлов, чтобы вы могли вносить изменения, не нарушая при этом целостность файлов RPM-пакета `libvirt-fstec`.

В качестве стартовой точки вы можете создать файл `/etc/polkit-1/rules.d/98-libvirt-user-rules` следующего содержания:

```
function logVMActions(action, subject, api) {
    var vmName = action.lookup('domain_name');
    var vmUUID = action.lookup('domain_uuid');

    var actionText;

    switch(api) {
    case 'domain.delete':
        actionText = 'deletion';
        break;
    case 'domain.init-control':
        actionText = 'reboot or shutdown';
        break;
    case 'domain.start':
    case 'domain.stop':
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
        actionText = api.split('.')[1];
        break;
    case 'domain.save':
    case 'domain.write':
        actionText = 'modification';
        break;
    }

    if (actionText) {
        polkit.log(`virtual machine "${vmName}" (UUID="${vmUUID}")
 ↪${actionText} initiated by "${subject.user}" user`);
    }
}

function checkUserVMAccess(action, subject) {
    // в этой функции вам необходимо реализовать собственные
 ↪правила проверки
    // доступа к виртуальной машине для пользователей средства
 ↪виртуализации
    return true;
}
```

И доработать функцию `checkUserVMAccess` по своему усмотрению.

В случае необходимости, используя файл `98-libvirt-user-rules` вы также можете предоставить пользователям дополнительные права доступа к API-вызовам гипервизора, например:

```
libvirtUserActions.push(...[
    'domain.start',
    'network.getattr',
    'network.read',
    'network-port.create',
    'network-port.delete',
    'network-port.read',
]);
```

## 14.14. Регистрация событий безопасности

### 14.14.1. Введение

Система виртуализации МСВСфера ОС поддерживает регистрацию событий безопасности в системном журнале аудита через встроенный механизм `auditd`. Таким образом системные администраторы и администраторы безопасности получают возможность отслеживать историю изменения состояния виртуальных машин и их конфигурации, а также осуществлять мониторинг заданных событий в реальном времени.

Так же с помощью `auditd` осуществляется регистрация изменений в конфигурационных файлах гипервизора и политиках безопасности.

По умолчанию системный журнал событий безопасности записывается в файл `/var/log/audit/audit.log`, процедура работы с журналом описана в соответствующем разделе Руководства администратора МСВСфера ОС (см. «*6. РЕГИСТРАЦИЯ СОБЫТИЙ БЕЗОПАСНОСТИ*»).

**Внимание:** функция регистрации событий безопасности поддерживается только для гипервизора, работающего в системном режиме (см. «*14.3.3. Системный режим*»).

### 14.14.2. Настройка гипервизора

По умолчанию функция регистрации событий безопасности включена и какая-либо дополнительная конфигурация на стороне гипервизора не требуется. Нижеприведённая информация о настройке гипервизора имеет справочный характер.

Для включения или отключения функции регистраций событий безопасности необходимо внести соответствующие изменения в следующие конфигурационные файлы:

- `/etc/libvirt/virtqemud.conf` — основной сервис для управления гипервизором.
- `/etc/libvirt/virtinterfaced.conf` — сервис для управления сетевыми интерфейсами хост-системы.
- `/etc/libvirt/virtnetworkd.conf` — сервис для управления виртуальными сетями.
- `/etc/libvirt/virtnodedevd.conf` — сервис для управления

оборудованием, подключённым к хост-системе.

- `/etc/libvirt/virtnwfilterd.conf` — сервис для управления сетевым экраном (брандмауэром) на хост-системе.
- `/etc/libvirt/virtsecretd.conf` — сервис хранения секретов.
- `/etc/libvirt/virtstoraged.conf` — сервис для управления пулами хранения и томами/разделами в этих пулах.
- `/etc/libvirt/virtproxyd.conf` — сервис, который обеспечивает обратную совместимость для клиентов, которые были ранее настроены на работу с UNIX-сокетом монолитного сервиса `libvirtd`, также опционально позволяет принимать и обрабатывать RPC команды по сети. В общем случае у вас не возникнет необходимость включения данного сервиса, однако в целях унификации конфигурации рекомендуется внести соответствующие правки в конфигурационный файл.
- `/etc/libvirt/libvirtd.conf` — конфигурационный файл от монолитного варианта сервиса *libvirtd*, поставляется для обеспечения совместимости.

За управление функцией регистрации событий безопасности отвечает опция `audit_level`, которая может принимать следующие значения:

- `audit_level=0` — регистрация событий безопасности отключена.
- `audit_level=1` — регистрация событий безопасности включена, если на сервере запущена подсистема аудита (сервис `auditd`), в противном случае функция отключается. Также это является поведением по умолчанию если опция не определена в конфигурационном файле.
- `audit_level=2` — регистрация событий безопасности включена в обязательном порядке, если подсистема аудита не запущена, то сервис `libvirtd` выдаст ошибку и не запустится.

Таким образом, для отключения функции регистрации событий безопасности необходимо в каждом из вышеперечисленных файлов установить значение опции `audit_level` равным `0`, а для включения — либо `1`, либо `2`.

После внесения правок в конфигурационные файлы необходимо перезапустить сервисы `libvirt` для принятия соответствующих изменений:

```
$ sudo systemctl restart virtqemud
$ sudo systemctl restart virtinterfaced
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
$ sudo systemctl restart virtnetworkd
$ sudo systemctl restart virtnodedevd
$ sudo systemctl restart virtnwfilterd
$ sudo systemctl restart virtsecretd
$ sudo systemctl restart virtstoraged
```

Если вы по каким-то причинам используете `virtproxyd`, то перезапустите также этот сервис:

```
$ sudo systemctl restart virtproxyd
```

В дополнение к записи событий безопасности в системный журнал аудита `libvirt` поддерживает дублирование этих событий в свой собственный журнал. За это отвечает опция `audit_logging` в конфигурационных файлах, которая может принимать следующие значения:

- `audit_logging=0` — не дублировать события безопасности в собственный журнал `libvirt` (поведение по умолчанию).
- `audit_logging=1` — дублировать события безопасности в собственный журнал `libvirt`.

### 14.14.3. Типы сообщений о событиях безопасности гипервизора

Поставляемый в МСВСфера ОС гипервизор `libvirt` использует три типа сообщений о событиях безопасности:

- `VIRT_CONTROL` — сообщение об изменении состояния виртуальной машины;
- `VIRT_MACHINE_ID` — сообщение о назначении контекста безопасности SELinux (маркировке) виртуальной машине;
- `VIRT_RESOURCE` — сообщения об использовании виртуальной машиной ресурсов хост-системы. Во время первого запуска виртуальной машины сообщения будут отправлены обо всех подключённых устройствах, в дальнейшем сообщения будут направляться о внесении изменений в конфигурацию оборудования виртуальной машины (изменение объёма выделенных ресурсов, подключение новых устройств и т.п.).

Примеры сообщений от гипервизора, полученные с помощью утилиты

ausearch:

```
$ sudo ausearch -m VIRT_RESOURCE,VIRT_CONTROL,VIRT_MACHINE_ID

time->Mon Oct 21 23:12:15 2024
type=VIRT_CONTROL msg=audit(1729541535.433:276): pid=3373 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm op=start reason=booted vm="msvsphere-9-
 ↪arm" uuid=03347cd5-8fbd-42ba-88d0-8c2c5968e3f1 vm-pid=0 exe="/
 ↪usr/sbin/virtqemud" hostname=? addr=? terminal=? res=failed'
----
time->Mon Oct 21 23:15:18 2024
type=VIRT_MACHINE_ID msg=audit(1729541718.054:322): pid=3373 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm vm="msvsphere-9-arm" uuid=d5aad9c6-
 ↪7e09-4b07-a89a-3c039d0ad8b3 vm-ctx=system_u:system_r:svirt_
 ↪t:s0:c675,c813 img-ctx=system_u:object_r:svirt_image_t:s0:c675,
 ↪c813 model=selinux exe="/usr/sbin/virtqemud" hostname=? addr=?
 ↪terminal=? res=success'
----
time->Mon Oct 21 23:15:18 2024
type=VIRT_RESOURCE msg=audit(1729541718.681:378): pid=3373 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm resrc=disk reason=start vm="msvsphere-
 ↪9-arm" uuid=d5aad9c6-7e09-4b07-a89a-3c039d0ad8b3 old-disk="?"
 ↪new-disk="/var/lib/libvirt/images/msvsphere-9-arm.qcow2" exe="/
 ↪usr/sbin/virtqemud" hostname=? addr=? terminal=? res=success'
```

Любое из трёх приведённых типов сообщений о событиях безопасности является в равной степени важным с точки зрения обеспечения информационной безопасности средств виртуализации. Администратор системы имеет возможность отфильтровать сообщения гипервизора по определённым типам с использованием утилиты ausearch из состава ОС МСВСфера. Инструкции по использованию ausearch доступны в соответствующем разделе Руководства администратора МСВСфера ОС (см. «*6. РЕГИСТРАЦИЯ СОБЫТИЙ БЕЗОПАСНОСТИ*»), а также на странице документации (man ausearch).

Для всех видов сообщений libvirt использует следующий базовый набор полей:

- pid — идентификатор процесса гипервизора libvirt, отправившего сообщение в журнал событий безопасности;
- uid — идентификатор пользователя, от имени которого был запущен процесс-отправитель сообщения;
- subj — контекст безопасности SELinux процесса-отправителя сообщения;
- msg — строка, содержащая разделённый пробелом список пар ключ=значение, специфичных для данного вида сообщения.

Ниже приведены общие ключи для поля msg у разных типов сообщений:

- virt — тип используемого драйвера виртуализации: qemu или lxc;
- vm — имя виртуальной машины;
- uuid — уникальный идентификатор виртуальной машины;
- exe — путь к исполняемому файлу процесса libvirt, выполнившему операцию;
- hostname — в настоящее время не используется;
- addr — в настоящее время не используется;
- terminal — в настоящее время не используется;
- res — статус выполнения операции: success в случае успешного выполнения, failed — в случае неудачи;

#### 14.14.3.1. VIRT_CONTROL

Сообщение типа VIRT_CONTROL уведомляет об изменении состояния виртуальной машины. У такого сообщения поле msg будет содержать следующие данные:

- op — тип выполненной операции: start, stop или init;
- reason — причина, по которой была выполнена операция. Несколько примеров:
    - op=stop reason=shutdown — виртуальная машина остановлена по команде безопасной остановки (virsh shutdown);
    - op=stop reason=destroy — виртуальная машины была остановлена по команде немедленной остановки (virsh destroy).
- vm-pid — идентификатор основного процесса виртуальной машины;
- init-pid — идентификатор init-процесса внутри контейнера.

Используется только если `op=init` и `virt=lxc`;

- `pid-ns` — идентификатор пространства имён (namespace) `init`-процесса внутри контейнера. Используется только если `op=init` и `virt=lxc`;

Ниже представлены несколько примеров сообщений типа `VIRT_CONTROL` в файле журнала `audit.log`:

- удачный запуск виртуальной машины:

```
type=VIRT_CONTROL msg=audit(1729541718.681:386): pid=3373
 ↪uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_
 ↪r:virtd_t:s0-s0:c0.c1023 msg='virt=kvm op=start
 ↪reason=booted vm="msvsphere-9-arm" uuid=d5aad9c6-7e09-
 ↪4b07-a89a-3c039d0ad8b3 vm-pid=4107 exe="/usr/sbin/
 ↪virtqemud" hostname=? addr=? terminal=? res=success'
```

- неудачный запуск виртуальной машины:

```
type=VIRT_CONTROL msg=audit(1729541535.433:276): pid=3373
 ↪uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_
 ↪r:virtd_t:s0-s0:c0.c1023 msg='virt=kvm op=start
 ↪reason=booted vm="msvsphere-9-arm" uuid=03347cd5-8fbd-
 ↪42ba-88d0-8c2c5968e3f1 vm-pid=0 exe="/usr/sbin/virtqemud"
 ↪hostname=? addr=? terminal=? res=failed'
```

- остановка виртуальной машины по команде `virsh shutdown`:

```
type=VIRT_CONTROL msg=audit(1729597082.830:613): pid=3373
 ↪uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_
 ↪r:virtd_t:s0-s0:c0.c1023 msg='virt=kvm op=stop
 ↪reason=shutdown vm="msvsphere-9-arm" uuid=d5aad9c6-7e09-
 ↪4b07-a89a-3c039d0ad8b3 vm-pid=0 exe="/usr/sbin/virtqemud"
 ↪hostname=? addr=? terminal=? res=success'
```

#### 14.14.3.2. VIRT_MACHINE_ID

Сообщение типа `VIRT_MACHINE_ID` уведомляет о назначении контекста безопасности SELinux (маркировке) виртуальной машине. У такого сообщения поле `msg` будет содержать следующие данные:

- `model` — тип драйвера безопасности: `selinux` или `apparmor`. МСВСфера ОС поддерживает только SELinux;
- `vm-ctx` — контекст безопасности для процесса, в котором запущена виртуальная машина;
- `img-ctx` — контекст безопасности для образа диска виртуальной машины и других её ресурсов.

Пример сообщения в файле журнала `audit.log`:

```
type=VIRT_MACHINE_ID msg=audit(1729541534.997:199): pid=3373 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm vm="msvsphere-9-arm" uuid=03347cd5-
 ↪8fbd-42ba-88d0-8c2c5968e3f1 vm-ctx=system_u:system_r:svirt_
 ↪t:s0:c29,c465 img-ctx=system_u:object_r:svirt_image_t:s0:c29,c465
 ↪model=selinux exe="/usr/sbin/virtqemud" hostname=? addr=?
 ↪terminal=? res=success'
```

#### 14.14.3.3. VIRT_RESOURCE

Сообщение типа `VIRT_RESOURCE` уведомляет об использовании ресурсов хост-системы виртуальной машиной. Во время первого запуска виртуальной машины сообщения будут отправлены обо всех подключённых устройствах, в дальнейшем сообщения будут направляться о внесении изменений в конфигурацию оборудования виртуальной машины (изменение объёма выделенных ресурсов, подключение новых устройств и т.п.).

Набор данных в поле `msg` отличается для разных типов ресурсов, однако эти два значения используются для всех ресурсов:

- `reason` — причина, по которой произошло выделение или изменение ресурса;
- `resrc` — тип выделяемого/изменяемого ресурса.

Описание полей, специфичных для отдельных типов ресурсов:

- центральный процессор (`resrc=vcpu`):
    - `old-vcpu` — изначальное количество виртуальных процессоров

или `0`

- `new-vcpu` — обновлённое количество виртуальных процессоров

- оперативная память (`resrc=mem`):
  - `old-mem` — изначальный размер оперативной памяти в байтах
  - `new-mem` — обновлённый размер оперативной памяти в байтах
- диск (`resrc=disk`):
  - `old-disk` — путь к изначальному файлу диска или дисковому устройству на хост-системе
  - `new-disk` — обновлённый путь к файлу диска или дисковому устройству на хост-системе
- сетевой интерфейс (`resrc=net`):
  - **для виртуальных сетевых устройств:**
    - `old-net` — изначальный MAC-адрес сетевого устройства
    - `new-net` — обновлённый MAC-адрес сетевого устройства
  - **для физических устройств хост-системы, назначенных виртуальной машине:**
    - `net` — MAC-адрес устройства на хост-системе
    - `rdev` — название сетевого интерфейса на хост-системе
- файловая система (`resrc=fs`):
  - `old-fs` — изначальный каталог, файл или путь к устройству, на котором находится файловая система, предоставляемая виртуальной машине или контейнеру
  - `new-fs` — обновлённый каталог, файл или путь к устройству, на котором находится предоставляемая файловая система
- физическое устройство (`resrc=hostdev` (блочные или символьные устройства) или `resrc=dev` (USB, PCI или SCSI-устройства)):
  - `dev` — уникальный идентификатор USB, PCI или SCSI-устройства (`resrc=dev`)
  - `disk` — путь к блочному устройству, выделенному для виртуальной машины (`resrc=hostdev`)
  - `chardev` — путь к символьному устройству, выделенному для

виртуальной машины (`resrc=hostdev`)

- TPM (Trusted Platform Module) модуль (`resrc=tpm` или `resrc=tpm-emulator`):
    - `device` — путь к TPM-устройству, выделенному для виртуальной машины
- генератор случайных чисел (`resrc=rng`):
    - `old-rng` — изначальный путь к источнику энтропии на хост-системе
    - `new-rng` — обновлённый путь к источнику энтропии на хост-системе
- последовательный порт, параллельный порт, терминал (`resrc=chardev`):
    - `old-chardev` — изначальный путь к символьному устройству, используемому для эмуляции устройства
    - `new-chardev` — обновлённый путь к символьному устройству, используемому для эмуляции устройства
- смарт-карта (`resrc=smartcard`):
    - `old-smartcard` — изначальный путь к устройству смарт-карты для проброса в виртуальную машину
    - `new-smartcard` — обновлённый путь к устройству смарт-карты для проброса в виртуальную машину
- перенаправленное USB-устройство (`resrc=redir`):
    - `bus` — тип шины, на текущий момент поддерживается только `usb`
    - `device` — тип устройства, на текущий момент поддерживается только `USB redir`
- контрольная группа cgroup (`resrc=cgroup`):
    - `cgroup` — название контроллера группы cgroup
- разделяемая память (`resrc=shmem`):
    - `size` — размер выделяемой области памяти
    - `shmem` — название выделяемой области памяти
    - `source` — путь к символьному устройству, используемому для эмуляции устройства

Пример сообщений типа `VIRT_RESOURCE` в файле журнала `audit.log`:

```
type=VIRT_RESOURCE msg=audit(1729619180.453:1009): pid=12366 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm resrc=disk reason=start vm="msvsphere-
 ↪9-arm" uuid=87b934dd-ba95-4930-84fe-f5caf0996964 old-disk="?"
 ↪new-disk="/var/lib/libvirt/images/msvsphere-9-arm.qcow2" exe="/
 ↪usr/sbin/virtqemud" hostname=? addr=? terminal=? res=success'

type=VIRT_RESOURCE msg=audit(1729619180.453:1010): pid=12366 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm resrc=disk reason=start vm="msvsphere-
 ↪9-arm" uuid=87b934dd-ba95-4930-84fe-f5caf0996964 old-disk="?"
 ↪new-disk="/srv/iso/MSVSphere-9.4-x86_64-arm.iso" exe="/usr/sbin/
 ↪virtqemud" hostname=? addr=? terminal=? res=success'

type=VIRT_RESOURCE msg=audit(1729619180.453:1011): pid=12366 uid=0
 ↪auid=4294967295 ses=4294967295 subj=system_u:system_r:virtd_t:s0-
 ↪s0:c0.c1023 msg='virt=kvm resrc=net reason=start vm="msvsphere-9-
 ↪arm" uuid=87b934dd-ba95-4930-84fe-f5caf0996964 old-net="?" new-
 ↪net="52:54:00:5c:ae:7c" exe="/usr/sbin/virtqemud" hostname=?
 ↪addr=? terminal=? res=success'
```

### 14.14.4. Отслеживание изменений в конфигурационных файлах

С точки зрения информационной безопасности рекомендуется настроить операционную систему на отслеживание изменений в конфигурационных файлах и политиках безопасности гипервизора.

Одним из способов решения этой задачи является добавление соответствующих правил для `auditd` в вашу систему. Готовый набор правил поставляется в составе пакета `sphere-libvirt-integrity`, однако, следует иметь ввиду, что установка этого пакета также автоматически включает функцию контроля целостности конфигурации виртуальных машин (см. «*14.15.2. Контроль целостности конфигурации виртуальной машины*»). Если такая конфигурация является для вас нежелательной, вы сможете самостоятельно настроить правила для подсистемы аудита следуя инструкциям в конце этого раздела.

Для установки пакета `sphere-libvirt-integrity` выполните следующую команду:

```
$ sudo dnf install sphere-libvirt-integrity
```

После установки пакета служба аудита начнёт автоматически отслеживать и регистрировать изменения конфигурационных файлов гипервизора в системном журнале событий безопасности. Далее эти события могут быть обработаны стандартными средствами аудита, включёнными в состав операционной системы МСВСфера. Дополнительная информация по этим инструментам доступна в разделе «*6. РЕГИСТРАЦИЯ СОБЫТИЙ БЕЗОПАСНОСТИ*» руководства администратора.

Примеры событий безопасности, регистрирующих изменения конфигурационных файлов гипервизора:

```
$ sudo ausearch -k libvirt-config-changes

time->Wed Oct 23 18:46:45 2024
type=PROCTITLE msg=audit(1729698405.750:1233):
 ↪proctitle=2F7573722F62696E2F6D \
63002D50002F7661722F746D702F6D632D726F6F742F6D632E7077642E3134373238
type=PATH msg=audit(1729698405.750:1233): item=0 name="/etc/
 ↪libvirt/libvirt-admin.conf" inode=264181 dev=fd:00 mode=0100644
 ↪ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:virt_etc_t:s0
 ↪nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_
 ↪frootid=0
type=CWD msg=audit(1729698405.750:1233): cwd="/etc/libvirt"
type=SYSCALL msg=audit(1729698405.750:1233): arch=c000003e
 ↪syscall=92 success=yes exit=0 a0=55b5c7d34e90 a1=0 a2=0 a3=0
 ↪items=1 ppid=14728 pid=14768 auid=1666 uid=0 gid=0 euid=0 suid=0
 ↪fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=2 comm="mc" exe="/usr/
 ↪bin/mc" subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.
 ↪c1023 key=2F6574632F6C6962766972742F6C6962766972742D61 \
646D696E2E636F6E66016C6962766972742D636F6E6669672D6368616E676573
----
time->Wed Oct 23 18:46:45 2024
type=PROCTITLE msg=audit(1729698405.750:1234):
 ↪proctitle=2F7573722F62696E2F6D63 \
002D50002F7661722F746D702F6D632D726F6F742F6D632E7077642E3134373238
type=PATH msg=audit(1729698405.750:1234): item=0 name="/etc/
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 ↪libvirt/libvirt-admin.conf" inode=264181 dev=fd:00 mode=0100644
 ↪ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:virt_etc_t:s0
 ↪nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_
 ↪frootid=0
type=CWD msg=audit(1729698405.750:1234): cwd="/etc/libvirt"
type=SYSCALL msg=audit(1729698405.750:1234): arch=c000003e
 ↪syscall=90 success=yes exit=0 a0=55b5c7d34e90 a1=81a4
 ↪a2=55b5c7d33020 a3=0 items=1 ppid=14728 pid=14768 auid=1666 uid=0
 ↪gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=2
 ↪comm="mc" exe="/usr/bin/mc" subj=unconfined_u:unconfined_
 ↪r:unconfined_t:s0-s0:c0.c1023
 ↪key=2F6574632F6C6962766972742F6C6962766972742 \
D61646D696E2E636F6E66016C6962766972742D636F6E6669672D6368616E676573
----
time->Wed Oct 23 18:46:45 2024
type=PROCTITLE msg=audit(1729698405.750:1235):
 ↪proctitle=2F7573722F62696E2F6D630 \
02D50002F7661722F746D702F6D632D726F6F742F6D632E7077642E3134373238
type=PATH msg=audit(1729698405.750:1235): item=1 name="/etc/
 ↪libvirt/libvirt-admin.conf" inode=264181 dev=fd:00 mode=0100644
 ↪ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:virt_etc_t:s0
 ↪nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_
 ↪frootid=0
type=PATH msg=audit(1729698405.750:1235): item=0 name="/etc/
 ↪libvirt/" inode=263878 dev=fd:00 mode=040700 ouid=0 ogid=0
 ↪rdev=00:00 obj=system_u:object_r:virt_etc_t:s0 nametype=PARENT
 ↪cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=0
type=CWD msg=audit(1729698405.750:1235): cwd="/etc/libvirt"
type=SYSCALL msg=audit(1729698405.750:1235): arch=c000003e
 ↪syscall=257 success=yes exit=13 a0=ffffff9c a1=55b5c7d34e90
 ↪a2=241 a3=81a4 items=2 ppid=14728 pid=14768 auid=1666 uid=0 gid=0
 ↪euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts2 ses=2 comm=
 ↪"mc" exe="/usr/bin/mc" subj=unconfined_u:unconfined_r:unconfined_
 ↪t:s0-s0:c0.c1023 key=2F6574632F6C6962766972742F6C6962766972742 \
D61646D696E2E636F6E66016C6962766972742D636F6E6669672D6368616E676573
```

Для ручной настройки правил аудита создайте файл */etc/audit/rules.d/msvsphere-libvirt.rules* следующего содержания:

```
#
# Watch for Libvirt and QEMU configuration files changes
#
-w /etc/libvirt/libvirt-admin.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/libvirt.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/libvirtd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/qemu.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/qemu-lockd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtinterfaced.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtlockd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtlogd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtnetworkd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtnodedevd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtnwfilterd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtproxyd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtqemud.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtsecretd.conf -p wa -k libvirt-config-changes
-w /etc/libvirt/virtstoraged.conf -p wa -k libvirt-config-changes

#
# Watch for Libvirt Polkit actions changes
#
-w /usr/share/polkit-1/actions/org.libvirt.api.policy -p wa -k
 ↪libvirt-polkit-changes
-w /usr/share/polkit-1/actions/org.libvirt.unix.policy -p wa -k
 ↪libvirt-polkit-changes
-w /usr/share/polkit-1/rules.d/50-libvirt.rules -p wa -k libvirt-
 ↪polkit-changes
```

Установите на него корректные права:

```
$ sudo chown root:root /etc/audit/rules.d/msvsphere-libvirt.rules
$ sudo chmod 600 /etc/audit/rules.d/msvsphere-libvirt.rules
```

И загрузите новые правила:

```
$ sudo augenrules --load
```

На этом процедуру настройки можно считать завершённой — служба аудита будет автоматически применять эти правила во время загрузки системы.

## 14.15. Контроль целостности

### 14.15.1. Контроль целостности исполняемых файлов и конфигурации гипервизора

Контроль целостности конфигурационных и исполняемых файлов, а также разделяемых библиотек гипервизора и загрузчиков (прошивок) BIOS и UEFI осуществляется с помощью утилиты AIDE (Advanced Intrusion Detection Environment), которая входит в комплект поставки операционной системы МСВСфера.

Контролю целостности подлежат следующие объекты:

- `/etc/libvirt/*.conf` — конфигурационные файлы компонентов среды виртуализации;
- `/usr/share/edk2/ovmf/*.fd` — файлы UEFI-прошивки;
- `/usr/share/seabios/bios-256k.bin` — файл BIOS-прошивки;
- исполняемые файлы и разделяемые библиотеки компонентов гипервизора.

Процедура установки, настройки и работы с утилитой `aide` описана в главе «*9.2. Программа для контроля целостности AIDE*» руководства администратора.

В конфигурации по умолчанию `aide` автоматически отслеживает изменения файлов прошивок, исполняемых файлов и разделяемых библиотек гипервизора.

Для отслеживания изменений конфигурационных файлов перед инициализацией базы данных `aide` добавьте следующий блок в конфигурационный файл `/etc/aide.conf` перед строкой `/etc PERMS`:

```
# libvirt components configuration
/etc/libvirt/.*.conf$ CONTENT_EX
```

Если база данных `aide` уже была инициализирована, обновите её согласно разделу «*9.2.3. Обновление базы данных aide*» после изменения конфигурации.

После этого вы можете использовать команду `aide --check` для проверки целостности в том числе и компонентов гипервизора, в случае обнаружения нарушений вы получите соответствующее предупреждение в отчёте:

```
$ sudo  aide --check
Start timestamp: 2025-01-22 13:58:18 +0300 (AIDE 0.16)
AIDE found differences between database and filesystem!!
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
Summary:
  Total number of entries:      191099
  Added entries:                0
  Removed entries:              0
  Changed entries:              3

---------------------------------------------------
Changed entries:
---------------------------------------------------
f   ...    .C... : /etc/libvirt/virtqemud.conf
f   ..g    ..... : /usr/bin/virt-qemu-run
f   ..g    ..... : /usr/share/seabios/bios-256k.bin

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

File: /etc/libvirt/virtqemud.conf
  SHA512   : rIklyUlys8wRXd81qxtyLDD4xQyRJRFo |
 ↪yG0f0GD9rvQGa1CRkxuYSTTmNgBBjTO6
             lMx9va0S8rFIWej/xY32WbWGTyU6bRIF |
 ↪DhgUMGd7EH03616hyxOOkyDGIM96Is9x
             NkQaE1F5J3Bvjoo0t/CVKw==         | epJF/
 ↪77v7xXI1m31YgzX0A==

File: /usr/bin/virt-qemu-run
  Gid      : 0                               | 1000

File: /usr/share/seabios/bios-256k.bin
  Gid      : 0                               | 1000
```

Также вы можете настроить периодический запуск `aide` через службу `cron` чтобы получать отчёты о целостности системы по электронной почте.

### 14.15.2. Контроль целостности конфигурации виртуальной машины

Гипервизор libvirt, поставляемый в составе сертифицированной версии операционной системы МСВСфера поддерживает автоматический контроль целостности конфигурации виртуальной машины, а также её первичного загрузчика (BIOS или UEFI-прошивки). В случае обнаружения факта нарушения целостности запуск виртуальной машины будет заблокирован системой.

Для активации функции контроля целостности вам необходимо установить пакет `sphere-libvirt-integrity`:

```
$ sudo dnf install sphere-libvirt-integrity
```

Какая-либо дополнительная конфигурация не требуется — проверка будет автоматически выполняться для всех виртуальных машин.

Эталонные контрольные суммы конфигурационных файлов и первичного загрузчика виртуальной машины вычисляются в момент легитимного изменения конфигурации виртуальной машины. Таковым считается изменение, выполненное пользователем с соответствующими правами доступа в рамках установленной ролевой модели с помощью штатных инструментов гипервизора: команд `virsh edit` или `virsh define`, утилиты `virt-manager` или API libvirt. Изменение, выполненное любым другим способом, будет считаться нарушением целостности.

Если на момент установки пакета `sphere-libvirt-integrity` в вашей системе уже были созданы виртуальные машины, то вам необходимо выполнить любое изменение их конфигурации одним из вышеперечисленных способов — это станет триггером для вычисления системой эталонных контрольных сумм. Например:

```
# экспортировать конфигурацию виртуальной машины msvsphere-9-arm в
 ↪файл
# msvsphere-9-arm.xml
$ virsh dumpxml --inactive --security-info msvsphere-9-arm >
 ↪msvsphere-9-arm.xml

# обновить конфигурацию виртуальной машины msvsphere-9-arm из файла
# msvsphere-9-arm.xml
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
$ virsh define msvsphere-9-arm.xml
Domain 'msvsphere-9-arm' defined from msvsphere-9-arm.xml
```

На этапе сохранения конфигурации будут вычислены контрольные суммы, необходимые для работы системы.

В случае выявления нарушения целостности объектов контроля запуск такой виртуальной машины будет заблокирован:

```
$ virsh start msvsphere-9-arm
ошибка: Failed to start domain 'msvsphere-9-arm'
ошибка: Ошибка выполнения сценария обработчика: внутренняя ошибка:
 ↪Дочерний процесс (LC_ALL=C PATH=/usr/local/sbin:/usr/local/bin:/
 ↪usr/sbin:/usr/bin /etc/libvirt/hooks/qemu msvsphere-9-arm prepare
 ↪begin -), не ожидалось состояние выхода 1
```

А в системный журнал событий безопасности службы `auditd` будет добавлено соответствующее событие с типом `VIRT_INTEGRITY_CHECK`:

```
$ sudo ausearch -m VIRT_INTEGRITY_CHECK -ts recent
----
time->Mon Feb 10 20:54:40 2025
type=VIRT_INTEGRITY_CHECK msg=audit(1739210080.578:262): pid=2131
 ↪uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_
 ↪r:virtd_t:s0-s0:c0.c1023 msg='virt=kvm vm="msvsphere-9-arm"
 ↪uuid=459f0e74-2c86-42a8-8f24-5a45cc437a9b op=vm-config-check
 ↪reason="Config-hash-mismatch-aborting-start" reason-code=20 exe=
 ↪"/usr/bin/python3.9" hostname=? addr=? terminal=? res=failed'
```

Формат данных, используемый в записях журнала аудита, подробно описан в разделе «*6.4.1. Формат файла журнала событий безопасности*» руководства администратора, так что здесь рассмотрим только поля, специфичные для данного типа события:

- `vm="msvsphere-9-arm"` — название виртуальной машины, для которой выполнялась проверка целостности;
- `uuid=459f0e74-2c86-42a8-8f24-5a45cc437a9b` — уникальный идентификатор (UUID) виртуальной машины;
- `op=vm-config-check` — название проверки, во время которой было обнаружено нарушение целостности.

Допустимые значения:

- `vm-config-check` — проверка конфигурационного файла и загрузчика виртуальной машины;
- `vm-files-check` — проверка целостности файлов внутри виртуальной машины. Эта функция будет рассмотрена в следующем разделе.

- `reason="Config-hash-mismatch-aborting-start"` — причина, по которой проверка завершилась неудачей. В данном случае — несоответствие контрольной суммы конфигурации виртуальной машины.
- `reason-code=20` — код ошибки.

  Допустимые значения:

  - `20` — нарушение целостности конфигурационного файла виртуальной машины;
  - `21` — нарушение целостности первичного загрузчика (прошивка BIOS/UEFI) виртуальной машины;
  - `22` — ошибка при чтении файла с эталонными контрольными суммами.

### 14.15.3. Контроль целостности файлов виртуальной машины

Гипервизор libvirt из состава сертифицированной версии МСВСфера ОС также поддерживает функцию контроля целостности файлов, которые находятся внутри виртуальной машины.

В текущей реализации этой функции есть следующие ограничения:

- поддерживаются только гостевые операционные системы на базе GNU/Linux;
- поддерживаемые файловые системы: Ext 2/3/4 и XFS;
- не поддерживается работа с зашифрованными дисками и сетевыми хранилищами;
- не поддерживается автозапуск виртуальных машин с включённой функцией контроля целостности файлов.

Для активации функции контроля целостности файлов вам необходимо установить пакет `sphere-libvirt-integrity` если вы не сделали этого ранее:

```
$ sudo dnf install sphere-libvirt-integrity
```

Далее, вам необходимо выключить виртуальную машину и задать список файлов для отслеживания с помощью команды `virsh file-integrity`. Например, следующие команды включат проверку целостности для файлов `/etc/passwd` и `/etc/shadow` виртуальной машины `msvsphere-9-arm`:

```
$ virsh shutdown msvsphere-9-arm
Domain 'msvsphere-9-arm' is being shutdown

$ virsh file-integrity msvsphere-9-arm --path-add /etc/passwd
$ virsh file-integrity msvsphere-9-arm --path-add /etc/shadow
```

Список отслеживаемых файлов можно посмотреть с помощью следующей команды:

```
$ virsh file-integrity msvsphere-9-arm --list-files
 FilePath      Hash
----------------------
 /etc/passwd
 /etc/shadow
```

Обратите внимание, в столбце Hash отсутствует информация по причине того, что эталонные контрольные суммы файлов ещё не были вычислены. Для их вычисления выполните команду:

```
$ sudo virsh file-integrity msvsphere-9-arm --update-hash
 FilePath      Hash
------------------------------------------------------------------------
 ↪--------------
 /etc/passwd
 ↪870994e271e33bd7c83ef53c1955a1a1afd91f11af4c035f09c6ab1e5224b6af
 /etc/shadow
 ↪475fc32b7d05c65d9334d5c0030f2371bdb8112233c19d755d20fec4d61ba8e9
```

На этом процедура настройки завершена — виртуальную машину можно включать.

В случае обнаружения нарушения целостности отслеживаемых файлов во время запуска виртуальной машины запуск будет заблокирован:

```
$ virsh start msvsphere-9-arm
ошибка: Failed to start domain 'msvsphere-9-arm'
ошибка: Ошибка выполнения сценария обработчика: внутренняя ошибка:
 ↪Дочерний процесс (LC_ALL=C PATH=/usr/local/sbin:/usr/local/bin:/
 ↪usr/sbin:/usr/bin /etc/libvirt/hooks/qemu msvsphere-9-arm prepare
 ↪begin -), не ожидалось состояние выхода 1
```

А в системный журнал событий безопасности будет добавлена соответствующая запись, имеющая тип `VIRT_INTEGRITY_CHECK`:

```
$ sudo ausearch -m VIRT_INTEGRITY_CHECK -ts recent
----
time->Tue Feb 11 19:41:23 2025
type=VIRT_INTEGRITY_CHECK msg=audit(1739292083.307:724): pid=5959
 ↪uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_
 ↪r:virtd_t:s0-s0:c0.c1023 msg='virt=kvm vm="msvsphere-9-arm"
 ↪uuid=459f0e74-2c86-42a8-8f24-5a45cc437a9b op=vm-files-check
 ↪reason="Hash-of-file-mismatch" reason-code=4 filepath="/etc/
 ↪passwd" filehash=
 ↪"e0937f15d8fd333aff2acc9c79c00dc2d10287502093305289e8d5cd3b70bcfa
 ↪" expectedhash=
 ↪"870994e271e33bd7c83ef53c1955a1a1afd91f11af4c035f09c6ab1e5224b6af
 ↪" exe="/usr/bin/python3.9" hostname=? addr=? terminal=?
 ↪res=failed'
```

Формат данных, используемый в записях журнала аудита, подробно описан в разделе «*6.4.1. Формат файла журнала событий безопасности*» руководства администратора, так что здесь рассмотрим только поля, специфичные для данного типа события:

- `vm="msvsphere-9-arm"` — название виртуальной машины, для которой выполнялась проверка целостности;
- `uuid=459f0e74-2c86-42a8-8f24-5a45cc437a9b` — уникальный идентификатор (UUID) виртуальной машины;
- `op=vm-files-check` — название проверки, во время которой было обнаружено нарушение целостности. Допустимые значения:
    - `vm-files-check` — проверка целостности файлов внутри виртуальной машины. Эта функция будет рассмотрена в следующем разделе.

    - `vm-config-check` — проверка конфигурационного файла и загрузчика виртуальной машины;
- `reason="Hash-of-file-mismatch"` — причина, по которой проверка завершилась неудачей. В данном случае — несоответствие контрольной суммы файла.
- `reason-code=4` — код ошибки. Допустимые значения:
    - `4` — реальная контрольная сумма файла отличается от эталонной;
    - `3` — ошибка вычисления контрольной суммы файла;
    - `2` — отслеживаемый файл не найден в виртуальной машине;
    - `1` — возникла ошибка при монтировании диска виртуальной машины;
    - `0` — диск не найден в конфигурации виртуальной машины.
- `filepath="/etc/passwd"` — путь к файлу;
- `filehash=""` — реальная контрольная сумма файла;
- `expectedhash=""` — ожидаемая контрольная сумма файла.

В случае если обнаруженные изменения являются корректными, вам необходимо обновить эталонные контрольные суммы файлов перед повторным запуском виртуальной машины:

```
$ sudo virsh file-integrity msvsphere-9-arm --update-hash
 FilePath      Hash
------------------------------------------------------------------------
 ↪--------------
 /etc/passwd
 ↪e0937f15d8fd333aff2acc9c79c00dc2d10287502093305289e8d5cd3b70bcfa
 /etc/shadow
 ↪a1218b7c44b77929d5068689a31ef66fc2144f98a4817e46dd241e53b2496a8e
```

При необходимости, отключить контроль целостности файла (в данном примере `/etc/passwd`) можно с помощью следующей команды:

```
$ virsh file-integrity msvsphere-9-arm --path-delete /etc/passwd

$ virsh file-integrity msvsphere-9-arm --list-files
 FilePath      Hash
------------------------------------------------------------------------
 ↪--------------
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 /etc/shadow
↪a1218b7c44b77929d5068689a31ef66fc2144f98a4817e46dd241e53b2496a8e
```

Для отключения функции контроля целостности файлов остановите виртуальную машину, запустите редактор XML конфигурации виртуальной машины:

```
$ virsh shutdown msvsphere-9-arm

$ virsh edit msvsphere-9-arm
```

И полностью удалите секцию `<fi:fileintegrity ...>...</fi:fileintegrity>` не изменяя при этом остальные данные:

```
<domain type='kvm'>
  <name>msvsphere-9-arm</name>
  <uuid>459f0e74-2c86-42a8-8f24-5a45cc437a9b</uuid>
  <metadata>
    <libosinfo:libosinfo xmlns:libosinfo="http://libosinfo.org/
↪xmlns/libvirt/domain/1.0">
      <libosinfo:os id="http://msvsphere-os.ru/msvsphere/9"/>
    </libosinfo:libosinfo>
    <fi:fileintegrity xmlns:fi="http://libvirt.org/schemas/domain/
↪metadata/fileintegrity/1.0" check="1">
      <fi:item path="/etc/shadow" hash=
↪"a1218b7c44b77929d5068689a31ef66fc2144f98a4817e46dd241e53b2496a8e
↪"/>
    </fi:fileintegrity>
  </metadata>
  ...
</domain>
```

После сохранения изменений функция контроля целостности для виртуальной машины будет отключена.

## 14.16. Оптимизация

Данный раздел описывает методы оптимизации работы гипервизора и виртуальных машин.

### 14.16.1. Kernel Same-Page Merging

#### 14.16.1.1. Описание технологии KSM

KSM (Kernel Same-Page Merging) — это технология ядра Linux, позволяющая объединять одинаковые страницы памяти между различными пользовательскими процессами и виртуальными машинами в одну для совместного использования, реализуя таким образом дедупликацию данных для оперативной памяти.

Соответственно, использование KSM позволяет более эффективно расходовать оперативную память повышая тем самым плотность виртуальных машин на одном физическом сервере.

Однако, дедупликация данных — это ресурсоёмкий процесс, который увеличивает нагрузку на центральный процессор, что в свою очередь может привести к ухудшению производительности для некоторых классов задач.

Также в прошлом известны случаи успешных атак из одной виртуальной машины на другую в пределах одного физического сервера. Один из таких случаев описан в статье Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth and Berk Sunar - Wait a minute! A fast, Cross-VM attack on AES. В настоящий момент информации об открытых уязвимостях, связанных с KSM, нет. Однако, использование данной технологии не рекомендуется в сертифицированных системах, системах с повышенными требованиями к защищённости и/или работающих с конфиденциальными данными.

По умолчанию технология KSM отключена в МСВСфера ОС, проверить её статус можно с помощью следующей команды:

```
$ cat /sys/kernel/mm/ksm/run
0
```

возможные значения:

- `0` — поддержка KSM выключена, все ранее объединённые страницы остаются объединёнными;

- `1` — поддержка KSM включена;
- `2` — поддержка KSM выключена, все ранее объединённые страницы разъединяются.

#### 14.16.1.2. Включение поддержки KSM

Для включения поддержки KSM необходимо установить пакет `ksmtuned`:

```
$ sudo dnf install -y ksmtuned
```

И активировать сервисы `ksm` и `ksmtuned`:

```
$ sudo systemctl enable --now ksm
$ sudo systemctl enable --now ksmtuned
```

После этого проверьте включена ли технология KSM в ядре:

```
$ cat /sys/kernel/mm/ksm/run
1
```

Следить за количеством объединённых страниц памяти (одна страница — 4096 байт) можно с помощью следующей команды:

```
$ watch cat /sys/kernel/mm/ksm/pages_sharing
```

После включения KSM вам необходимо некоторое время следить за производительностью и потреблением ресурсов на сервере виртуализации, особенно за использованием центрального процессора. Если желаемый эффект не достигнут, значит KSM не подходит под ваш сценарий использования и эту технологию необходимо выключить.

#### 14.16.1.3. Отключение поддержки KSM

Для отключения KSM необходимо остановить соответствующие сервисы:

```
$ sudo systemctl disable --now ksmtuned
$ sudo systemctl disable --now ksm
```

Опционально, можно также отменить уже выполненное объединение страниц памяти:

```
$ echo 2 | sudo tee /sys/kernel/mm/ksm/run
```

При необходимости, для удаления пакета `ksmtuned` выполните следующую команду:

```
$ sudo dnf erase -y ksmtuned
```

# 15. СРЕДСТВА КОНТЕЙНЕРИЗАЦИИ

## 15.1. Введение

В данной главе описана процедура установки, настройки и использования средств контейнеризации в операционной системе МСВСфера 9.

Контейнеризация — это технология, которая позволяет упаковывать приложения вместе со всеми их зависимостями в изолированные среды, называемые контейнерами. Применение этого механизма позволяет запускать приложение и необходимый набор системных библиотек в полностью стандартизированном контейнере, который взаимодействует через определённые интерфейсы. Контейнеры используют ядро операционной системы хостовой машины и, в отличие от полной виртуализации, не требуют эмуляции аппаратного обеспечения. Приложения, работающие в разных контейнерах, изолированы друг от друга и не могут взаимодействовать или влиять на работу соседних контейнеров. В ОС МСВСфера 9 в качестве средства контейнеризации используется программное обеспечение Docker. Docker является одним из самых популярных инструментов для контейнеризации. В этой главе рассмотрим основные шаги и команды для работы с Docker.

## 15.2. Установка

Для установки Docker и необходимых зависимостей выполните следующую команду:

```
$ sudo dnf install -y docker-ce docker-ce-cli containerd
```

Запустите и включите сервис Docker:

```
$ sudo systemctl start docker

$ sudo systemctl enable docker
```

Убедитесь, что Docker установлен и работает:

```
$ sudo systemctl status docker
```

## 15.3. Управление контейнерами

### 15.3.1. Команда docker container

Команда `docker container` используется для управления контейнерами. Синтаксис команды:

```
docker container <команда>
```

Используемые команды управления контейнерами перечислены в таблице.

Таблица 56 - Команды управления контейнерами

| Команда | Описание |
|---|---|
| `docker container attach` | Прикрепить локальные стандартные потоки ввода, вывода и вывода ошибок к запущенному контейнеру. |
| `docker container commit` | Создание нового образа из изменений контейнера. |
| `docker container cp` | Копирование файлов/папок между контейнером и хостовой файловой системой. |
| `docker container create` | Создание нового контейнера. |
| `docker container diff` | Проверка изменений файлов или каталогов в файловой системе контейнера. |
| `docker container exec` | Выполнение команды в запущенном контейнере. |
| `docker container export` | Экспорт файловой системы контейнера в виде tar-архива. |
| `docker container inspect` | Вывод подробной информации об одном или нескольких контейнерах. |
| `docker container kill` | Принудительное завершение одного или нескольких запущенных контейнеров. |
| `docker container logs` | Просмотр журналов регистрации контейнера. |
| `docker container ls` | Вывод списка контейнеров. |
| `docker container pause` | Приостановка всех процессов в одном или нескольких контейнерах. |
| `docker container port` | Список портов контейнера. |

| Команда | Описание |
|---|---|
| `docker container prune` | Удаление всех остановленных контейнеров. |
| `docker container rename` | Изменение имени контейнера. |
| `docker container restart` | Перезапуск одного или нескольких контейнеров. |
| `docker container rm` | Удаление одного или нескольких контейнеров. |
| `docker container run` | Создание и запуск нового контейнера из образа. |
| `docker container start` | Запуск одного или нескольких остановленных контейнеров. |
| `docker container stat` | Отображение статистики использования ресурсов контейнера(ов) в реальном времени. |
| `docker container stop` | Остановка одного или нескольких запущенных контейнеров. |
| `docker container top` | Отображение запущенных процессов контейнера. |
| `docker container unpause` | Отключение всех процессов в одном или нескольких контейнерах. |
| `docker container update` | Обновление конфигурации одного или нескольких контейнеров. |
| `docker container wait` | Блокировка остановки одного или нескольких контейнеров, вывод их кодов выхода. |

При использовании команд допускается сокращение записи до вида:

```
docker <команда>
```

Далее будут рассмотрены основные команды управления контейнерами.

С подробной информацией обо всех командах и их работе можно ознакомиться, выполнив:

```
$ docker container <команда> --help
```

### 15.3.2. Команда docker build

Команда `docker build` используется для создания образа на основе файла Dockerfile и контекста.

Контекст — это набор файлов, находящихся по пути, определённому с помощью переменной `PATH` или `URL`.

- `PATH` — это директория в локальной системе.
- `URL` — это удалённый репозиторий.

Контекст сборки обрабатывается рекурсивно, поэтому `PATH` включает как директорию, так и все её поддиректории, а `URL` — как репозиторий, так и все его подмодули.

Синтаксис команды:

```
docker build [<опции>] PATH | URL | -
```

Опции команды перечислены в таблице.

Таблица 57 - Опции команды `docker build`

| Команда | Описание |
|---|---|
| `-f, --file <string>` | Имя Dockerfile. |
| `--isolation <string>` | Технология изоляции контейнеров, по умолчанию `default`. |
| `--label <list>` | Установка метаданных для образа. |
| `-m, --memory <bytes>` | Ограничение памяти. |
| `--network <string>` | Установить режим работы с сетью для инструкций `RUN` во время сборки (по умолчанию `default`). |
| `--rm` | Удалять промежуточные контейнеры после успешной сборки (по умолчанию `true`). |

### 15.3.3. Команда docker image

Команда `docker image` предназначена для управления образами.

Синтаксис команды:

```
docker image <команда>
```

Доступные команды перечислены в таблице.

Таблица 58 - Команды `docker image`

| Команда | Описание |
|---|---|
| `docker image build` | Создание образа из Docker-файла. |
| `docker image history` | Вывод истории образа. |
| `docker image import` | Импорт содержимого из tarball для создания образа файловой системы. |
| `docker image inspect` | Вывод подробной информации об одном или нескольких образах. |
| `docker image load` | Загрузка образа из tar-архива или STDIN. |
| `docker image ls` | Вывод списка образов. |
| `docker image prune` | Удаление неиспользуемых образов. |
| `docker image pull` | Загрузка образа из registry. |
| `docker image push` | Загрузка образа в registry. |
| `docker image rm` | Удалить один или несколько образов. |
| `docker image save` | Сохранить один или несколько образов в tar-архив (по умолчанию передается в STDOUT). |
| `docker image tag` | Создать тег `TARGET_IMAGE`, который ссылается на `SOURCE_IMAGE`. |

### 15.3.4. Команда docker images

Команда `docker images` используется для вывода списка доступных образов.

Синтаксис команды:

```
docker images [<опции>] [<репозиторий>[:<тег>]]
```

Опции команды перечислены в таблице.

Таблица 59 - Опции команды `docker images`

| Команда | Описание |
|---|---|
| `--all, -a` | Показать все образы (по умолчанию промежуточные образы скрыты). |
| `--digests` | Показать дайджест. |

| Команда | Описание |
|---|---|
| `--filter, -f` | Фильтрация выходных данных на основе предоставленных условий. |
| `--format` | Форматирование вывода с использованием пользовательского шаблона `table`. Вывод в формате таблицы с заголовками столбцов (по умолчанию) `table TEMPLATE`. |
| `--no-trunc` | Не обрезать вывод. |
| `--quiet, -q` | Показывать только идентификаторы образов. |

### 15.3.5. Команда docker create

Команда `docker container create` (или сокращённо: `docker create`) создает новый контейнер из указанного образа, не запуская его. При создании контейнера демон docker создает записываемый слой контейнера поверх указанного образа и готовит его к выполнению указанной команды. Идентификатор контейнера затем выводится в STDOUT. Действия похожи на выполнение команды `docker run -d`, за исключением того, что контейнер никогда не запускается. Затем можно использовать команду `docker container start` (или сокращённо: `docker start`) для запуска контейнера в любой момент.

Синтаксис команды:

```
docker create [<опции>] <образ> [<команда>] [<аргументы>...]
```

Опции команды перечислены в таблице.

Таблица 60 - Опции команды `docker create`

| Команда | Описание |
|---|---|
| `--attach ( -a)` | Позволяет манипулировать вводом и выводом по мере необходимости, для вывода информации используется стандартный ввод/вывод контейнера STDIN, STDOUT и STDERR. |
| `--cpus` | Определяет количество процессоров. |
| `--device` | Добавить хост-устройство в контейнер. |

| Команда | Описание |
|---|---|
| `--hostname, -h` | Имя хоста контейнера. |
| `--interactive, -i` | Оставить STDIN открытым без присоединения к терминалу. |
| `--memory, -m` | Лимит памяти. Данный параметр устанавливает верхний предел памяти, доступной для контейнера. Предел задаётся в контрольной группе, и приложения в контейнере могут запрашивать его по адресу `/sys/fs/cgroup/memory/memory.limit_in_bytes`. |
| `--rm` | Автоматическое удаление контейнер после выхода. |
| `--tty, -t` | Выделить псевдотерминал. |
| `--user, -u` | Имя пользователя или UID (формат: `<name\|uid>[:<group\|gid>]`). |
| `--volume, -v` | Примонтировать том. |
| `--volumes-from` | Смонтировать тома из указанного(ых) контейнера(ов). |
| `--workdir, -w` | Задать рабочий каталог внутри контейнера. |

### 15.3.6. Команда docker run

Команда `docker run` используется для создания и запуска контейнера из образа. Может принимать несколько аргументов, таких как имя образа, параметры запуска контейнера и команду, которую необходимо выполнить внутри контейнера.

Синтаксис команды:

```
docker run [<опции>] <образ> [<команды>] [<аргументы>…]
```

Основные опции команды перечислены в таблице.

Таблица 61 - Опции команды `docker run`

| Команда | Описание |
|---|---|
| `--attach ( -a)` | Позволяет манипулировать вводом и выводом по мере необходимости, для вывода информации используется стандартный ввод/вывод контейнера STDIN, STDOUT и STDERR. |
| `--cpus` | Определяет количество процессоров. |
| `--detach, -d` | Запустить контейнер в фоновом режиме и напечатать ID контейнера. |
| `--device` | Добавить хост-устройство в контейнер. |
| `--hostname, -h` | Имя хоста контейнера. |
| `--interactive, -i` | Оставить STDIN открытым без присоединения к терминалу. |
| `--memory, -m` | Лимит памяти. Данный параметр устанавливает верхний предел памяти, доступной для контейнера. Предел задаётся в контрольной группе, и приложения в контейнере могут запрашивать его по адресу `/sys/fs/cgroup/memory/memory.limit_in_bytes`. |
| `--privileged` | Предоставление расширенных прав контейнеру |
| `--rm` | Автоматическое удаление контейнер после выхода. |
| `--tty, -t` | Выделить псевдотерминал. |
| `--user, -u` | Имя пользователя или UID (формат: `<name\|uid>[:<group\|gid>]`). |
| `--volume, -v` | Примонтировать том. |
| `--workdir, -w` | Задать рабочий каталог внутри контейнера. |

### 15.3.7. Команда docker start

Команда `docker start` предназначена для запуска контейнера.

Синтаксис команды:

```
docker start [<опции>] <контейнер> [<контейнер_2>...]
```

Используемые опции команды перечислены в таблице.

Таблица 62 - Опции команды `docker start`

| Команда | Описание |
|---|---|
| `--attach, -a` | Прикрепить STDOUT/STDERR и переадресовать сигналы. |
| `--detach-keys` | Переопределение последовательности клавиш для отсоединения контейнера. |
| `--interactive, -i` | Прикрепить STDIN контейнера. |

### 15.3.8. Команда docker stop

Команда `docker stop` предназначена для остановки работы контейнера.

Основной процесс внутри контейнера получит `SIGTERM`, а по истечении времени ожидания — `SIGKILL`.

Синтаксис команды:

```
docker stop [<опции>] <контейнер> [<контейнер_2>...]
```

Используемые опции команды перечислены в таблице.

Таблица 63 - Опции команды `docker stop`

| Команда | Описание |
|---|---|
| `--time , -t` | Время ожидания (в секундах) перед принудительной остановкой контейнера. |

### 15.3.9. Команда docker pause

Команда `docker pause` приостанавливает все процессы в указанных контейнерах.

Синтаксис команды:

```
docker pause <контейнер> [<контейнер_2>...]
```

### 15.3.10. Команда docker login

Для доступа в некоторые централизованные хранилища образов необходимо иметь учётную запись. После регистрации войти в систему можно, используя команду `docker login`.

Синтаксис команды:

```
docker login [<опции>] [<сервер>]
```

Для входа в систему необходимо указать имя пользователя и пароль своей учётной записи. Для доступа к собственным хранилищам образов необходимо указать имя сервера, если сервер не указан, он определяется демоном docker по умолчанию.

Основные параметры команды перечислены в таблице.

Таблица 64 - Опции команды `docker login`

| Команда | Описание |
|---|---|
| `--password, -p <пароль>` | Пароль. |
| `--password-stdin` | Получить пароль со стандартного ввода. |
| `--username, -u <имя>` | Имя пользователя. |

### 15.3.11. Команда docker attach

Команда `docker attach` позволяет прикрепить стандартный ввод (STDIN), вывод (STDOUT) и вывод ошибок (STDERR) локального терминала к контейнеру, используя идентификатор или имя контейнера. Это позволит просматривать его текущий вывод или управлять им в интерактивном режиме, как если бы команды выполнялись непосредственно в локальном терминале.

Для остановки контейнера используется сочетание клавиш «CTRL+C», которое передаст контейнеру сигнал SIGKILL. Для остановки контейнера может быть использована команда `exit`, которая корректно завершает его работу — это наиболее предпочтительный вариант остановки контейнера. Если контейнер был запущен с параметрами `-i` и `-t`, можно отсоединиться от контейнера и оставить его запущенным, используя последовательность клавиш «CTRL+P» и «CTRL+Q».

Синтаксис команды:

```
docker attach [<опции>] <контейнер>
```

Основные параметры команды перечислены в таблице.

Таблица 65 - Опции команды `docker attach`

| Команда | Описание |
|---|---|
| `--detach-keys` | Переопределить последовательность клавиш для отсоединения контейнера. |
| `--no-stdin` | Не присоединять STDIN. |
| `--sig-proxy` | Передать все полученные сигналы процессу. Значение по умолчанию — `true`. |

### 15.3.12. Команда docker exec

Команда `docker exec` запускает новую команду в работающем контейнере. Команда, запущенная с помощью `docker exec`, выполняется только во время работы основного процесса контейнера (PID 1), и она не перезапускается, если перезапускается контейнер. Команда запускается в директории контейнера по умолчанию. Если базовый образ имеет пользовательский каталог, указанный директивой `WORKDIR` в Dockerfile, тогда используется данный каталог. Команда должна быть исполняемым файлом. Команда, заключённая в кавычки, не сработает. Например, `docker exec -it my_container sh -c "echo a && echo b"` будет выполнена, а `docker exec -it my_container "echo a && echo b"` — нет.

Синтаксис команды:

```
docker exec [<опции>] <контейнер> <команда> [<аргументы>...]
```

Опции команды перечислены в таблице.

Таблица 66 - Опции команды `docker exec`

| Команда | Описание |
|---|---|
| `-d, --detach` | Режим отсоединения: выполнение команды в фоновом режиме. |
| `--detach-keys <string>` | Переопределение последовательности клавиш для отсоединения контейнера. |

| Команда | Описание |
|---|---|
| `-e, --env <list>` | Установка переменных окружения. |
| `--env-file <list>` | Чтение файла с переменными окружения. |
| `-i, --interactive` | Оставить STDIN открытым, даже если он не подключён. |
| `--privileged` | Предоставить расширенные привилегии команде. |
| `-t, --tty` | Выделить псевдотерминал. |
| `-u, --user <string>` | Имя пользователя или UID (формат: `<name\|uid>[:<group\|gid>]`). |
| `-w, --workdir <string>` | Рабочая директория внутри контейнера. |

### 15.3.13. Команда docker logs

Команда `docker logs` предназначена для вывода журналов контейнера. Синтаксис команды:

```
docker logs [<опции>] <контейнер>
```

Используемые параметры команды перечислены в таблице.

Таблица 67 - Опции команды `docker logs`

| Команда | Описание |
|---|---|
| `--details` | Показать дополнительные детали, предоставляемые в журнале. |
| `-f, --follow` | Следить за выводом журналов. |
| `--since <string>` | Показывать записи с указанной временной метки (например, `2022-03-01T13:23:37Z`) или с определённого промежутка времени (например, `42m` — отобразит записи журнала за последние 42 минуты). |
| `-n, --tail <string>` | Количество строк для отображения записей с конца журнала (по умолчанию `все`). |
| `-t, --timestamps` | Показать временные метки. |

| Команда | Описание |
|---|---|
| `--until <string>` | Показать записи до указанной временной метки (например, `2022-03-01T13:23:37Z`) или до определённого промежутка времени (например, `42m` — отобразит записи, сделанные 42 минуты назад и ранее). |

### 15.3.14. Команда docker stats

Команда `docker stats` предназначена для отображения статистики использования ресурсов контейнеров в реальном времени. Для того чтобы ограничить данные одним или несколькими конкретными контейнерами, необходимо указать список имён или идентификаторов контейнеров, разделённых пробелом. Также можно указать остановленный контейнер, но остановленные контейнеры не возвращают никаких данных.

Синтаксис команды:

```
docker stats [<опции>] [<контейнер>...]
```

Используемые опции команды перечислены в таблице.

Таблица 68 - Опции команды `docker stats`

| Команда | Описание |
|---|---|
| `--all, -a` | Показать все контейнеры (по умолчанию отображаются только запущенные). |
| `--format` | Форматирование вывода с использованием пользовательского шаблона `table` — вывод в формате таблицы с заголовками столбцов (по умолчанию) `table TEMPLATE`. |
| `--no-stream` | Отключить потоковую статистику и получать только первый результат. |
| `--no-trunc` | Не обрезать вывод. |

### 15.3.15. Команда docker tag

Команда `docker tag` предназначена для присвоения определённого имени и, по желанию, тега образу в локальном хранилище.

Синтаксис команды:

```
docker tag SOURCE_IMAGE[:<тег>] TARGET_IMAGE[:<тег>]
```

- `SOURCE_IMAGE` — это имя образа или его ID, которому требуется назначить новое имя и/или тег.
- `TARGET_IMAGE` — новое имя образа, которое необходимо ему присвоить.

Дополнительный параметр `<тег>` может использоваться для указания конкретной версии образа.

Если задано имя собственного хранилища образов, за ним может дополнительно следовать номер порта в формате `:8080`. Имя хранилища должно соответствовать стандартным правилам DNS, но не может содержать символы подчёркивания. Компоненты имени могут содержать строчные буквы, цифры и разделители. Разделитель определяется как точка, один или два знака подчёркивания или один или несколько дефисов. Компонент имени не может начинаться или заканчиваться разделителем. Имя тега может содержать строчные и прописные буквы, цифры, символы подчёркивания, точки и дефисы. Имя тега не может начинаться с точки или дефиса и может содержать не более 128 символов.

## 15.4. Безопасность средства контейнеризации

Настройка функций безопасности средства контейнеризации поставляется в пакете `docker_security`.

Для установки выполните команду:

```
$ sudo dnf install docker_security
```

В процессе установки пакета осуществляется остановка и запрет общесистемного демона Docker, работающего с правами `root`, удаляется его сокет.

Работа общесистемного демона Docker противоречит требованиям безопасности функционирования Docker в непривилегированном (rootless) режиме, поэтому администратору запрещается выполнять следующие команды:

```
$ sudo systemctl enable docker.service docker.socket

$ sudo systemctl start docker.service
```

### 15.4.1. Настройка программного комплекса

Для настройки уведомлений посредством корпоративной электронной почты требуется заполнить файл настроек параметров: `/etc/docker_security/mail/send_mail_settings.json`. С этой целью необходимо открыть в редакторе файл `send_mail_settings.json` и заполнить данные сервера электронной почты отправителя, почтовый адрес отправителя, почтовый адрес получателя (подразумевается администратор системы), почтовый адрес получателя копии письма (подразумевается пользователь системы) и т.д.

Пример содержимого файла `send_mail_settings.json` сразу после установки программного комплекса:

```
"smtp_server_type": "",
"smtp_server_type_comment": "Сетевой протокол сервера: smtps или
 ↪smtp",
"smtp_server": "",
"smtp_server_comment": "Имя сервера отправки почты или IP-адрес.
 ↪Пример: smtp.mail.ru",
"smtp_port": 465,
"smtp_port_comment": "Порт сервера отправки почты. Пример: 465",
"smtp_user": "",
"smtp_user_comment": "Имя пользователя на сервере отправки почты.
 ↪Пример: name_sender@mail.ru",
"smtp_pass": "",
"smtp_pass_comment": "Пароль пользователя на сервере отправки
 ↪почты. Пример: xxx123",
"sender": "",
"sender_comment": "Почтовый адрес отправителя письма. Пример: name_
 ↪sender@mail.ru",
"recipient": "",
"recipient_comment": "Почтовый адрес получателя письма. Пример:
 ↪name_recipient@mail.ru",
"recipient_copy": "",
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
"recipient_copy_comment": "Почтовый адрес получателя письма (копия
↪письма). Пример: name_recipient_copy@mail.ru",
"authentification": "1",
"authentification_comment": "нужна ли аутентификация для отправки:
↪1 – нужна, 0 – не нужна"
```

**Примечание**

Некоторые SMTP-серверы ожидают совпадающих параметров `smtp_user` и `sender`.

Далее следует выполнить команду запуска скрипта подсистемы инициализации программного комплекса `/etc/sbin/docker_s`:

```
$ sudo docker_s
```

В процессе выполнения скрипта, администратору системы выводится на экран содержимое файла `/etc/docker_security/mail/send_mail_settings.json` и запрашивается подтверждение правильности заполнения.

- Нажмите `y` и клавишу «Ввод» для подтверждения правильности заполнения файла `send_mail_settings.json` и продолжения выполнения скрипта `docker_s`.
- Нажмите `n` и клавишу «Ввод» для прерывания выполнения скрипта `docker_s`. После чего заново заполните файл `send_mail_settings.json`.

После подтверждения правильности заполнения файла настроек `send_mail_settings.json` скрипт `docker_s` производит следующие действия:

- копирует файл `daemon.json` в каталоги всех пользователей системы. Если файл уже существует, то он будет перезаписан;
- выполняет настройку дополнительных правил аудита (формирует файл `/etc/audit/rules.d/docker_users.rules` и перезапускает сервис `auditd`);
- запускает необходимые сервисы.

**Примечание**

Скрипт `docker_s` должен завершить свою работу в штатном режиме, без

ошибок.

После успешного завершения процесса настройки, администратор должен выполнить перезагрузку компьютера. После перезагрузки, пользователи системы могут переходить к этапу персональной настройки демона Docker, описанной в руководстве пользователя системы.

**Примечание**

Настройку пакета необходимо производить до того, как в операционной системе будут созданы пользователи.

### 15.4.2. Остановка программного комплекса

Для полной остановки утилиты выполнить команду:

```
$ sudo docker_s –u
```

**Примечание**

Скрипт осуществит запрет и остановку только сервисов программного комплекса. Все настройки программного комплекса, системные и пользовательские настройки, пользовательские образы, контейнеры и сервисы Docker удалены и изменены не будут.

Для выборочной остановки подсистем утилиты выполните команды:

```
$ sudo systemctl disable <Имя сервиса программного комплекса>

$ sudo systemctl stop <Имя сервиса программного комплекса>
```

### 15.4.3. Запуск программного комплекса

**Примечание**

Запуск программного комплекса происходит в автоматическом режиме при загрузке компьютера. Процесс запуска описан для случая, когда программный комплекс был остановлен администратором системы в ручном режиме полностью или частично.

Для полного запуска программного комплекса выполните команду:

```
$ sudo docker_s -s
```

**Примечание**

Скрипт осуществит запуск всех сервисов программного комплекса. Все пользовательские файлы настроек, образы и контейнеры Docker изменениям не подвергнутся.

Для выборочного запуска подсистем программного комплекса выполните команды:

```
$ sudo systemctl enable <Имя сервиса программного комплекса>

$ sudo systemctl start <Имя сервиса программного комплекса>
```

### 15.4.4. Удаление программного комплекса

Для удаления программного комплекса выполните команду:

```
$ sudo dnf remove docker_security-VERSION.noarch.rpm
```

Будут удалены все файлы, которые были установлены в процессе установки программного комплекса.

**Примечание**

Все пользовательские настройки, образы и контейнеры Docker удалены

не будут. Работоспособность пользовательских образов и контейнеров Docker сохранится. В случае запуска общесистемного демона Docker, работающего с правами суперпользователя `root`, вероятно нарушение безопасности при работе с Docker.

## 15.5. Функции безопасности

### 15.5.1. Изоляция контейнеров

Использование непривилегированного (rootless) режима работы Docker позволяет обеспечить механизмы изоляции штатными средствами операционной системы. Каждый экземпляр Docker запускается в контексте конкретного пользователя операционной системы. Остальные пользователи не имеют доступа к данному контексту по определению. Таким образом, использование штатных средств операционной системы и работа Docker в непривилегированном (rootless) режиме в полной и достаточной степени реализуют требуемый уровень механизмов изоляции.

Возможность получить доступ к чужому контексту пользователя есть только у администратора, но нет у пользователей. Это также реализуется штатными средствами операционной системы МСВСфера. Возможность получить доступ к чужому контексту пользователя есть только у суперпользователя, но нет у простых пользователей. Это также реализуется штатными средствами ОС МСВСфера.

В качестве дополнительного средства обеспечения изоляции пространства имён процессов хост-системы и пространства имён контейнеров для межпроцессорного взаимодействия контейнеров также используется настройка `cgroup-parent` со значением, установленным по умолчанию, которая прописывается в файле настроек `daemon.json` демона Docker для каждого пользователя. Файл настроек `daemon.json` защищён от изменения и удаления рядовым пользователем штатными средствами ОС МСВСфера. Настройка защиты осуществляется программным комплексом.

В качестве дополнительного средства обеспечения механизмов изоляции, программный комплекс в оперативном режиме отслеживает момент запуска контейнера и контролирует параметры запуска контейнера. При обнаружении того, что контейнер был запущен с использованием недопустимых параметров или того, что контейнер был запущен с отсутствием обязательных параметров,

исполнение запуска контейнера прерывается с выводом пользователю соответствующего информационного сообщения и отправкой по электронной почте сообщений администратору и пользователю.

Параметры запуска контейнера, недопустимые к использованию, перечислены в таблице.

Таблица 69 - Параметры запуска контейнера, недопустимые к использованию

| Параметр | Причина |
|---|---|
| `--net=host` | Из-за угрозы нарушения изоляции сетевых пространств контейнеров. |
| `--pid=host` | Из-за угрозы нарушения изоляции пространств идентификаторов процессов контейнеров между процессами хост-системы и процессами контейнера. |
| `--ipc=host` | Из-за угрозы нарушения изоляции пространств имён процессов хост-системы и пространства имён контейнеров для межпроцессорного взаимодействия контейнеров. |
| `--device` | Из-за угрозы нарушения ограничения прав прикладного программного обеспечения, выполняемого внутри контейнера, на использование периферийных устройств, устройств хранения данных и съёмных машинных носителей информации (блочных устройств), входящих в состав информационной системы, а также изоляции пространств имён файловой системы хоста и пространств имён файловых систем контейнеров. |
| `--userns=host` | Из-за угрозы нарушения изоляции пространств имён пользователей и групп пользователей хост-системы и пространств имён пользователей и групп контейнеров. |

Параметры запуска контейнеров, обязательные к использованию, перечислены в таблице.

Таблица 70 - Параметры запуска контейнера, обязательные к использованию

| Параметр | Причина |
|---|---|
| `--memory` | Для обеспечения ограничения прав прикладного программного обеспечения, выполняемого внутри контейнера, на использование оперативной памяти хостовой операционной системы. |
| `--read-only` | Для обеспечения ограничения монтирования корневой файловой системы хостовой операционной системы в режиме «только для чтения». |

При запуске контейнера с недопустимым параметром пользователю в командной строке будет выведено предупреждение. По электронной почте администратору и пользователю будут также отправлены сообщения.

### 15.5.2. Выявление уязвимостей в образах контейнеров

Перечень утилит для сканирования:

- `trivy`
- `dockle`
- `docker-bench-security`

#### 15.5.2.1. Настройка периодичности сканирования системы для выявления уязвимостей средств контейнеризации

По умолчанию, настроено ежедневное сканирование системы в 14:00. Администратор системы может изменить периодичность сканирования.

Для это выполните следующие действия.

1. Настроите файл таймера `/usr/lib/systemd/system/docker_scan_time.timer`.
2. Для перезапуска подсистемы сканирования и формирования отчетов выполните команду:

```
$ sudo systemctl daemon-reload
```

Сервисами программного комплекса в адрес администратора системы и пользователя системы, который в данный момент времени осуществляет работу в

системе высылаются отчёты по результатам сканирования, контроля целостности и контроля параметров запуска контейнеров.

Администратор системы после анализа полученных данных принимает решения по вопросам безопасности.

В случае, если образ Docker содержит уязвимости того уровня опасности, с которыми образ не допущен к использованию, администратор обязан проконтролировать удаление данного образа и созданных на его основе контейнеров пользователем системы и поставить в известность разработчика образа о выявленных уязвимостях.

В случае, если отчёты о сканировании содержат информацию о том, что какой-либо образ не мог быть отсканирован, администратор обязан проанализировать данный вопрос и оказать помощь пользователю системы по переустановке данного образа. Если проблема сканирования не будет решена, администратор системы должен проконсультироваться у разработчика образа и принять решение о возможности дальнейшего использования данного образа.

### 15.5.3. Проверка корректности конфигурации контейнеров

Пользователь системы не является администратором системы и не имеет прав суперпользователя. Штатные настройки операционной системы не дают рядовому пользователю прав на операции монтирования. Системный администратор должен явно предоставить дополнительные права конкретному пользователю на право монтировать определённую файловую систему. Только суперпользователь сможет выполнять данные действия.

Использование непривилегированного (rootless) режима работы Docker не позволит внутри контейнера получить права на использование периферийных устройств, устройств хранения данных и съёмных машинных носителей информации (блочных устройств), входящих в состав информационной (автоматизированной) системы.

В непривилегированном (rootless) режиме, по умолчанию, только memory и pids контроллеры делегируются пользователям, не являющимся суперпользователем. Чтобы разрешить делегирование полномочий другим контроллерам, необходимо изменить конфигурацию `systemd`, для чего необходимо явным образом выполнить соответствующие настройки `systemd` и обладать правами суперпользователя.

Ограничение прав прикладного программного обеспечения, выполняемого внутри контейнера, на использование вычислительных ресурсов (оперативной памяти) обеспечивается контролем программного комплекса наличия в параметрах запуска обязательного к использованию параметра `--memory`.

Монтирование корневой файловой системы хостовой операционной системы в режиме «только для чтения» обеспечивается контролем программного комплекса наличия в параметрах запуска контейнера обязательного к использованию параметра `--read-only`.

При обнаружении того, что контейнер был запущен с использованием недопустимых параметров или отсутствующими обязательными параметрами, исполнение запуска контейнера прерывается с выводом пользователю соответствующего информационного сообщения и отправкой по электронной почте сообщений администратору и пользователю.

### 15.5.4. Контроль целостности контейнеров и их образов

Для реализации контроля целостности образов используется инструмент «Docker Content Trust» (DCT). Он позволяет разработчикам подписывать свои образы электронной цифровой подписью (ЭЦП), а пользователям проверять эту ЭЦП. Прохождение проверки гарантирует целостность и аутентичность образа. Docker проверяет подписи при каждой операции `docker pull` и `docker run`, что минимизирует риск использования изменённых или поддельных образов.

При скачивании пользователем образа по имени и тэгу Docker сначала проверяет ЭЦП манифеста.

В случае успешной проверки Docker скачивает необходимые слои, считает от них хэш и сверяет с хэшами из манифеста, то есть проверяет целостность образа.

Инструмент «Docker Content Trust» включается путем добавления в переменные системного окружения значения `DOCKER_CONTENT_TRUST=1`. Это позволит Docker проверять подписи при каждой операции `docker pull` и `docker push`, что минимизирует риск использования изменённых или поддельных образов. Внесение любых изменений в файлы образа, вызовет изменение значений рассчитываемых Docker хэшей и, соответственно, запрет Docker на исполнение последующих действий. В консоль пользователя будет выведено стандартное сообщение Docker об ошибке.

Пользователи, при создании собственных образов, обязаны их подписывать с помощью ЭЦП. Описание технологии создания самоподписанной ЭЦП или создания ЭЦП в сертифицированном центре, и интегрирование ЭЦП в ОС для работы со средствами Docker, не рассматривается. Методика работы с ЭЦП описана в официальной документации Docker.

После настройки пакета, файл эталонной базы `/etc/docker_security/integrity/integrity.md5`, который должен содержать контрольные суммы контролируемых файлов, отсутствует. Сервис подсистемы контроля целостности `docker_integrity.service` при своём первом запуске на основании предопределённого перечня файлов контроля из файла `/etc/docker_security/integrity/integrity.list` формирует файл эталонной базы.

Администратор обязан убедиться, что по электронной почте пришло уведомление с темой письма «Файл контрольных сумм не существует! Инициализация системы контроля целостности…»

Администратор системы может выполнить дополнительную настройку перечня контролируемых файлов `integrity.list` и пересоздать эталонную базу `integrity.md5`. Для этого необходимо выполнить следующие действия.

- Внести изменения в файл `/etc/docker_security/integrity/integrity.list`.
- Удалить файл эталонной базы `integrity.md5` с помощью следующей команды:

```
$ sudo rm /etc/docker_security/integrity/integrity.md5
```

- Перезапустить сервис подсистемы контроля целостности `docker_integrity.service` с помощью следующей команды:

```
$ sudo systemctl restart docker_integrity.service
```

Администратор системы должен пересоздать эталонную базу `integrity.md5` при добавлении нового пользователя в систему. Для этого необходимо выполнить следующие действия.

- Удалить файл эталонной базы `integrity.md5` с помощью следующей команды:

```
$ sudo rm /etc/docker_security/integrity/integrity.md5
```

- Перезапустить сервис подсистемы контроля целостности `docker_integrity.service` с помощью следующей команды:

```
$ sudo systemctl restart docker_integrity.service
```

Пользователю при работе с контейнерами при нарушении контроля целостности образов в консоль пользователя будет выведено стандартное сообщение Docker об ошибке. Пользователь системы обо всех проблемах при работе со средствами контейнеризации обязан информировать администратора системы. Обязанности пользователя по информированию администратора системы определены в руководстве пользователя системы (**todo дать ссылку**).

Сервис программного комплекса `docker_integrity` обеспечивает автоматизированный контроль целостности файлов дистрибутива docker и файлов конфигурации. При выявлении нарушения сервис выполняет запись о данном событии в системный журнал и отправляет уведомление администратору в виде почтового сообщения. Для контроля целостности используется утилита `md5sum` из состава ОС МСВСфера.

Целостность сведений о событиях безопасности, зафиксированных в системных журналах ОС, обеспечиваются штатными средствами ОС МСВСфера.

## 15.6. Проверка уязвимостей в контейнерах

### 15.6.1. Сканер уязвимостей OpenSCAP

В состав операционной системы МСВСфера входит инструмент для автоматизированного управления уязвимостями OpenSCAP, который в том числе может использоваться и для выявления уязвимостей в образах Docker-контейнеров.

Предварительные требования:

- Необходимо установить пакет `openscap-utils`, в состав которого входит утилита `oscap-docker`, используемая для проверки образов контейнеров:

```
$ sudo dnf install openscap-utils
```

- Пользователю, который будет выполнять проверку, необходимо предоставить права на запуск команды `oscap-docker` от имени

привилегированного пользователя поскольку для проверки утилита выполняет операцию `chroot` в файловой системе образа контейнера.
Для этого создайте файл `/etc/sudoers.d/oscap-docker` (замените `testuser` на имя реального пользователя перед запуском команды):

```
$ cat << EOF | sudo tee -a /etc/sudoers.d/oscap-docker
testuser   ALL=/bin/oscap-docker
EOF

$ sudo chmod 640 /etc/sudoers.d/oscap-docker
```

- Вам потребуется OVAL-файл с описанием уязвимостей для операционной системы, которую вы используете внутри контейнера. Для некоторых отечественных дистрибутивов OVAL-файлы предоставляет ФСТЭК на основе своей базы данных уязвимостей (БДУ), их вы можете найти на странице загрузки программы ScanOVAL для Linux в виде Deb или RPM-пакетов.
  Для извлечения файлов из RPM-пакета вы можете использовать следующую команду:

```
$ rpm2cpio PATH_TO_RPM_FILE | cpio -idmv
```

Для извлечения файлов из Deb-пакета используйте следующий набор команд:

```
# получение списка файлов в Deb-пакете. Интересующий вас
 ↪файл будет иметь имя
# data.tar.xz или data.tar.gz
$ ar -t PATH_TO_DEB_FILE
debian-binary
control.tar.gz
data.tar.xz

# извлечение файла data.tar.xz из Deb-пакета
$ ar -x PATH_TO_DEB_FILE data.tar.xz

# извлечение всех файлов из архива data.tar.xz. Если в
 ↪скачанном вами Deb-
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
# пакете находится файл data.tar.gz, то используйте аргумент
 ↪"z" вместо "J":
# $ tar -xzvf data.tar.gz
$ tar -xJvf data.tar.xz
```

Также многие производители операционных систем самостоятельно предоставляют OVAL-файлы для своих продуктов. Например, OVAL-файлы для МСВСфера ОС 9 доступны по адресу repo1.msvsphere-os.ru/msvsphere/9/security/oval/. Обратитесь к документации или технической поддержке производителя ОС, используемой вами в контейнере, для получения необходимой информации.

После завершения предварительной настройки вы можете запустить проверку образа следующим образом:

```
$ sudo oscap-docker image b30a739d04aa oval eval --results results.
 ↪xml \
      --report report.html msvsphere-9.en.oval.xml
Creating a temporary container for the image...
Docker container
 ↪0365b199c28d289de69156fa83d7e8009d5cef236e325957125421e2458f907e
 ↪ready to be scanned.
Definition oval:ru.msvsphere-os.infsa:def:20250377: false
Definition oval:ru.msvsphere-os.infsa:def:20250334: false
...
Definition oval:ru.msvsphere-os.infba:def:20245691: true
Definition oval:ru.msvsphere-os.infba:def:20243840: false
Evaluation done.

Temporary container
 ↪0365b199c28d289de69156fa83d7e8009d5cef236e325957125421e2458f907e
 ↪cleaned
Cleaning temporary extracted container...
```

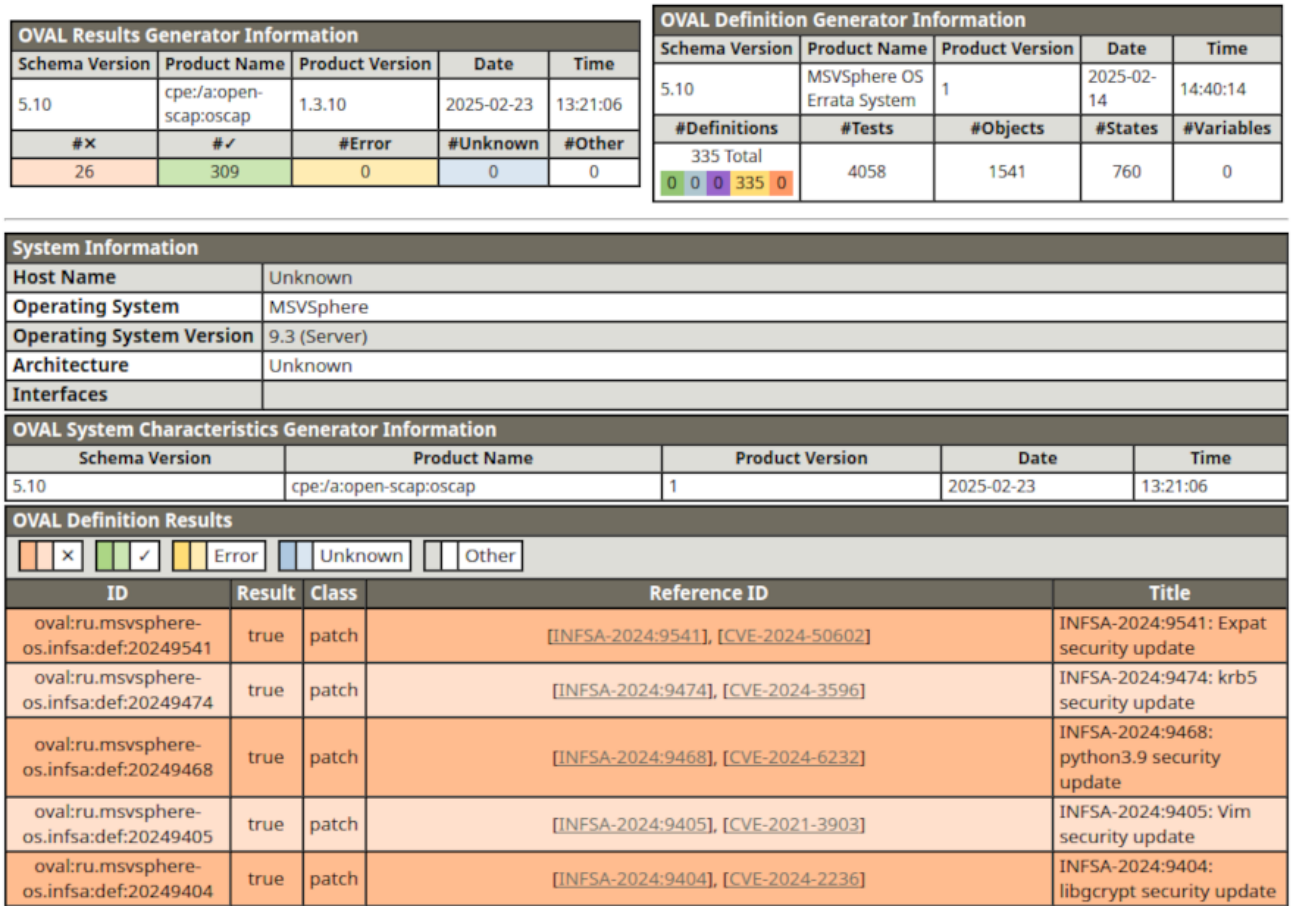
Где:

- `b30a739d04aa` — идентификатор образа контейнера для проверки, вы можете получить список образов в вашей системе с помощью команды `docker image ls`:

```
$ docker image ls
REPOSITORY          TAG       IMAGE ID       CREATED
 ↪SIZE
inferit/msvsphere   9.5       92b2dfaaa72e   12 days ago
 ↪212MB
inferit/msvsphere   9.3       b30a739d04aa   15 months ago
 ↪152MB
```

- `--results results.xml` — путь к файлу, в который необходимо сохранить отчёт о сканировании в XML формате;
- `--report report.html` — путь к файлу, в который необходимо сохранить отчёт о сканировании в HTML формате;
- `msvsphere-9.en.oval.xml` — путь к OVAL-файлу для тестируемой операционной системы.

Откройте сгенерированный файл `report.html` с помощью браузера.



**OVAL Results Generator Information**

| Schema Version | Product Name | Product Version | Date | Time |
|---|---|---|---|---|
| 5.10 | cpe:/a:open-scap:oscap | 1.3.10 | 2025-02-23 | 13:21:06 |

| #× | #✓ | #Error | #Unknown | #Other |
|---|---|---|---|---|
| 26 | 309 | 0 | 0 | 0 |

**OVAL Definition Generator Information**

| Schema Version | Product Name | Product Version | Date | Time |
|---|---|---|---|---|
| 5.10 | MSVSphere OS Errata System | 1 | 2025-02-14 | 14:40:14 |

| #Definitions | #Tests | #Objects | #States | #Variables |
|---|---|---|---|---|
| 335 Total: 0 0 0 335 0 | 4058 | 1541 | 760 | 0 |

**System Information**

| | |
|---|---|
| Host Name | Unknown |
| Operating System | MSVSphere |
| Operating System Version | 9.3 (Server) |
| Architecture | Unknown |
| Interfaces | |

**OVAL System Characteristics Generator Information**

| Schema Version | Product Name | Product Version | Date | Time |
|---|---|---|---|---|
| 5.10 | cpe:/a:open-scap:oscap | 1 | 2025-02-23 | 13:21:06 |

**OVAL Definition Results**

× ✓ Error Unknown Other

| ID | Result | Class | Reference ID | Title |
|---|---|---|---|---|
| oval:ru.msvsphere-os.infsa:def:20249541 | true | patch | [INFSA-2024:9541], [CVE-2024-50602] | INFSA-2024:9541: Expat security update |
| oval:ru.msvsphere-os.infsa:def:20249474 | true | patch | [INFSA-2024:9474], [CVE-2024-3596] | INFSA-2024:9474: krb5 security update |
| oval:ru.msvsphere-os.infsa:def:20249468 | true | patch | [INFSA-2024:9468], [CVE-2024-6232] | INFSA-2024:9468: python3.9 security update |
| oval:ru.msvsphere-os.infsa:def:20249405 | true | patch | [INFSA-2024:9405], [CVE-2021-3903] | INFSA-2024:9405: Vim security update |
| oval:ru.msvsphere-os.infsa:def:20249404 | true | patch | [INFSA-2024:9404], [CVE-2024-2236] | INFSA-2024:9404: libgcrypt security update |

Рис. 27: Отчёт OpenScap о проверке образа контейнера

Исходя из приведённого в качестве примера отчёта следует что:

- было обработано 335 бюллетеней безопасности из OVAL-файла (ячейка «#Definitions»);
- было обнаружено 26 срабатываний (ячейка «#x»);
- для 309 бюллетеней срабатываний не обнаружено (ячейка «#✓»);
- тестировалась операционная система МСВСфера 9.3 Сервер (ячейки «Operating system» и «Operating System Version»);
- таблица с результатами проверки по каждому из бюллетеней находится в конц отчёта, в начале располагаются бюллетени, для которых были обнаружены срабатывания;
- для каждого бюллетеня в таблице:
    - в колонке «Title» указано название бюллетеня;
    - в колонке «Reference ID» указаны ссылки на страницу бюллетеня в базе данных уязвимостей, а та же ссылки на информацию о конкретных уязвимостях (CVE), которые были исправлены в этом обновлении.

### 15.6.2. Сканер уязвимостей Trivy

Trivy — ещё один сканер уязвимостей с открытым исходным кодом, включённый в состав дистрибутива МСВСфера ОС. Как и OpenSCAP, он может выполнять функцию поиска уязвимостей в образах контейнеров, но, в отличии от OpenScap, использует свою собственную базу данных об уязвимостях для работы.

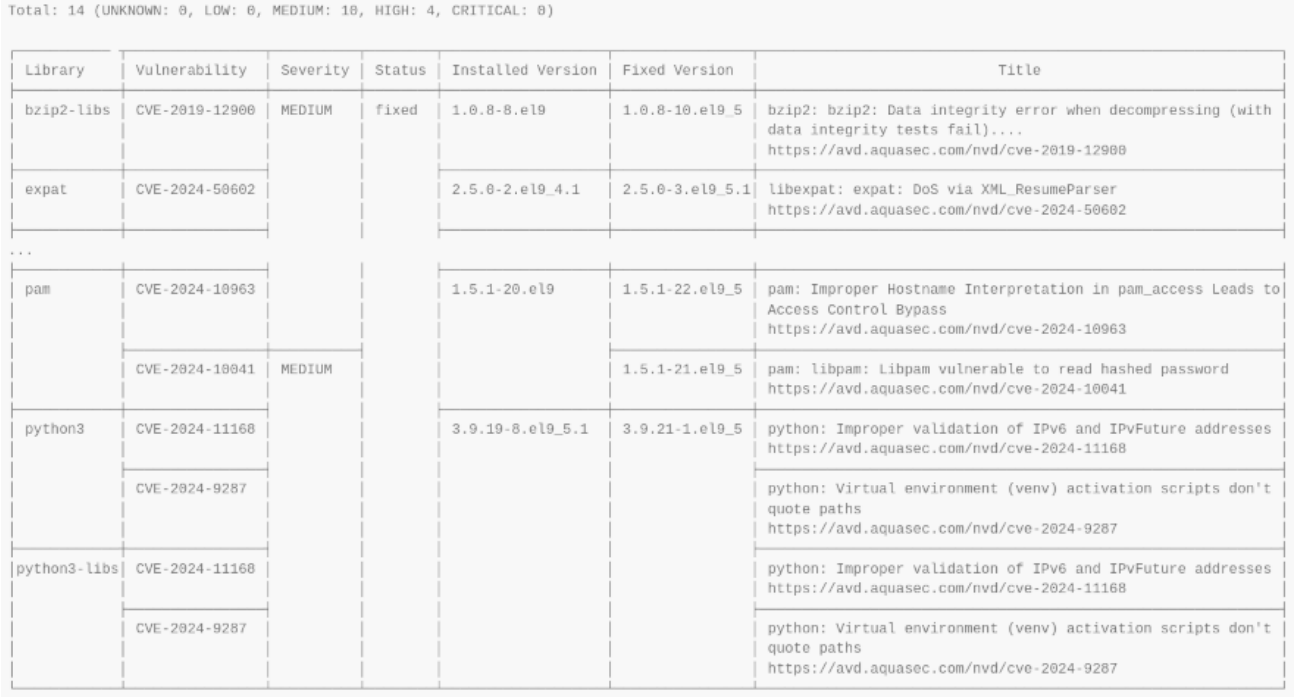
Для установки Trivy выполните следующую команду:

```
$ sudo dnf install trivy
```

Для сканирования образа контейнера выполните команду `trivy image`, передав ей в качестве аргумента идентификатор образа или путь к нему в формате `репозиторий:тег`:

```
$ trivy image 80c04158e975
```

Результат работы команды:



Total: 14 (UNKNOWN: 0, LOW: 0, MEDIUM: 10, HIGH: 4, CRITICAL: 0)

| Library | Vulnerability | Severity | Status | Installed Version | Fixed Version | Title |
|---|---|---|---|---|---|---|
| bzip2-libs | CVE-2019-12900 | MEDIUM | fixed | 1.0.8-8.el9 | 1.0.8-10.el9_5 | bzip2: bzip2: Data integrity error when decompressing (with data integrity tests fail).... https://avd.aquasec.com/nvd/cve-2019-12900 |
| expat | CVE-2024-50602 | | | 2.5.0-2.el9_4.1 | 2.5.0-3.el9_5.1 | libexpat: expat: DoS via XML_ResumeParser https://avd.aquasec.com/nvd/cve-2024-50602 |
| ... | | | | | | |
| pam | CVE-2024-10963 | | | 1.5.1-20.el9 | 1.5.1-22.el9_5 | pam: Improper Hostname Interpretation in pam_access Leads to Access Control Bypass https://avd.aquasec.com/nvd/cve-2024-10963 |
| | CVE-2024-10041 | MEDIUM | | | 1.5.1-21.el9_5 | pam: libpam: Libpam vulnerable to read hashed password https://avd.aquasec.com/nvd/cve-2024-10041 |
| python3 | CVE-2024-11168 | | | 3.9.19-8.el9_5.1 | 3.9.21-1.el9_5 | python: Improper validation of IPv6 and IPvFuture addresses https://avd.aquasec.com/nvd/cve-2024-11168 |
| | CVE-2024-9287 | | | | | python: Virtual environment (venv) activation scripts don't quote paths https://avd.aquasec.com/nvd/cve-2024-9287 |
| python3-libs | CVE-2024-11168 | | | | | python: Improper validation of IPv6 and IPvFuture addresses https://avd.aquasec.com/nvd/cve-2024-11168 |
| | CVE-2024-9287 | | | | | python: Virtual environment (venv) activation scripts don't quote paths https://avd.aquasec.com/nvd/cve-2024-9287 |

Рис. 28: Результат работы команды `trivy`

В результате выполнения Trivy выдаст таблицу со списком всех обнаруженных уязвимостей. Формат вывода результатов можно настроить с помощью аргумента `--format`, для просмотра списка поддерживаемых форматов и других опций выполните команду `trivy image --help`.

По умолчанию Trivy самостоятельно скачивает актуальную базу данных уязвимостей с внешнего ресурса, соответственно, для работы сканера в таком режиме требуется доступ к сети Интернет.

В случае необходимости использования в изолированной среде вы можете скачать базу данных вручную на компьютере, подключённом к сети:

```
$ trivy --cache-dir DB_DIR_PATH image --download-db-only
```

Замените `DB_DIR_PATH` на путь к каталогу, в котором необходимо сохранить базу данных.

После этого скопируйте каталог с базой данных на целевой компьютер и запускайте Trivy, используя следующий набор аргументов:

```
$ trivy image --cache-dir DB_DIR_PATH --skip-db-update
```

### 15.6.3. Автоматическая проверка образов контейнеров на наличие уязвимостей

В сертифицированной версии операционной системы МСВСфера реализована функция автоматической проверки образов контейнеров на уязвимости во время следующих операций:

- скачивание образа контейнера из реестра (`docker pull`);
- запуск контейнера из образа (`docker run`);
- сборка образа контейнера (`docker build`);
- импорт образа контейнера (`docker import`).

Для активации этой функции после установки и настройки службы `docker` из пакета `docker-ce` установите сканер `docker-revisor`:

```
$ sudo dnf install docker-revisor
```

Какая-либо дополнительная настройка со стороны `docker` не требуется — сервис модифицирован таким образом, что автоматически вызывает функцию проверки образа в случае обнаружения в системе исполняемого файла `/usr/bin/docker-revisor`.

Сканер `docker-revisor` представляет собой обёртку над описанными ранее в этой главе сканерами OpenSCAP и Trivy. Настройка сканера осуществляется путём редактирования конфигурационного файла `/etc/docker-revisor.yaml`. Ниже приведена конфигурация по умолчанию с описаним доступных опций:

```
---
# блок "scanners" позволяет включить (yes) или отключить (no)
 ↪использование
# внешних сканеров программой
scanners:
  # включить/выключить проверку с помощью Trivy
  trivy: yes
  # включить/выключить проверку с помощью OpenSCAP
  openscap: yes

# trivy:
#   # по умолчанию Trivy скачивает/обновляет свою базу данных
 ↪уязвимостей
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
#   # автоматически для каждого пользователя. Опция "offline_db_
 ↪path" позволяет
#   # задать путь к заранее скачанной базе данных, а также
 ↪отключает
#   # автоматическое обновление базы данных перед сканированием
 ↪образа.
#   # Таким образом становится возможным использование сканера
 ↪Trivy в
#   # защищённых окружениях без доступа к сети Интернет.
#   # Для скачивания/обновления базы данных выполните следующую
 ↪команду на
#   # компьютере, подключённом к сети:
#   #     $ docker-revisor trivy-sync-db
#   # Программа docker-revisor сохранит актуальную базу данных в
 ↪каталоге,
#   # указанном в "offline_db_path". Далее, вам необходимо будет
 ↪скопировать
#   # базу данных на компьютер, который будет осуществлять проверку
 ↪образов
#   # контейнеров, с помощью внешнего накопителя или внутренней
 ↪локальной сети.
#   offline_db_path: /var/lib/docker-revisor/trivy/

# Список путей к OVAL-файлам для сканируемых операционных систем.
 ↪Системный
# администратор должен самостоятельно скачать их, прежде чем
 ↪программа сможет
# использовать эти данные для сканирования.
oval: []

# Пример конфигурации OVAL для ОС МСВСфера 9
# oval:
#     # название операционной системы, как правило это значение
 ↪поля ID
#     # из файла /etc/os-release
#   - os_name: msvsphere
#     # версия операционной системы, как правило это значение поля
 ↪VERSION_ID
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
#     # из файла /etc/os-release. Проверка выполняется по наличию
 ↪заданной
#     # подстроки в начале значения, соответственно, "9" будет
 ↪подходить как
#     # для версий "9.x", так и для версии "9".
#     os_version: '9'
#     # редакция операционной системы. Как правило вам не
 ↪потребуется указывать
#     # это поле, однако, у операционной системы МСВСфера есть
 ↪несколько
#     # редакций с разными значениями в поле CPE_NAME.
#     edition: baseos
#     # путь к OVAL-файлу для заданной операционной системы
#     path: /var/lib/docker-revisor/oval/msvsphere-9.en.oval.xml
```

Процедура получения OVAL-файлов для целевой операционной системы описана ранее в этой главе в разделе «*15.6.1. Сканер уязвимостей OpenSCAP*». Вам необходимо самостоятельно скачать их, поместить в доступный для чтения всеми пользователями Docker каталог, допустим, `/var/lib/docker-revisor/oval` и внести соответствующие изменения в конфигурационный файл.

На этом процедуру настройки можно считать завершённой, проверка образа на наличие уязвимостей будет выполняться автоматически. В случае обнаружения уязвимостей программа `docker` выведет пользователю соответствующее сообщение и заблокирует скачивание или запуск такого образа или контейнера:

```
$ docker run -it --rm inferit/msvsphere:9.3
docker: Error response from daemon:
 ↪sha256:b30a739d04aaffcb5ba2770e17bc2c90220544d289ab0604721 \
3128ac5b5b078 image vulnerability scan failed:
Trivy scan skipped: OS is not supported
OpenScap scan completed:
  oval:ru.msvsphere-os.infsa:def:20249541:
    title: INFSA-2024:9541: Expat security update
    CVEs:
      CVE-2024-50602  [moderate]   https://nvd.nist.gov/vuln/
 ↪detail/CVE-2024-50602
...
```

(продолжение на следующей странице)

(продолжение с предыдущей страницы)

```
 oval:ru.msvsphere-os.infba:def:20245691:
   title: INFBA-2024:5691: ca-certificates security update
   CVEs:
     CVE-2023-37920  [low]        https://nvd.nist.gov/vuln/
↪detail/CVE-2023-37920
 Total CVEs found: 46.
```

## 16. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

| Изм. | Номера страниц | | | Номера извещений | Подпись | Дата внесения изм. | Дата введения изм. |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |