ОПЕРАЦИОННАЯ СИСТЕМА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ «ASTRA LINUX SPECIAL EDITION» РУСБ.10152-02

Руководство администратора. Часть 1

Оперативное обновление 4.7.3

Бюллетень № 2022-1121SE47

Листов 22

2022

АННОТАЦИЯ

В настоящем руководстве приводятся кумулятивные изменения в документ РУСБ.10152-02 95 01-1 «Операционная система специального назначения «Astra Linux Special Edition». Руководство администратора. Часть 1» из комплектности изделия РУСБ.10152-02 «Операционная система специального назначения «Astra Linux Special Edition» (далее по тексту — ОС), которые необходимо учитывать при настройке и эксплуатации ОС с установленным оперативным обновлением согласно бюллетеню № 2022-1121SE47.

Руководство предназначено для администраторов ОС и сети.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

В настоящем руководстве приведены кумулятивные изменения в документ РУСБ.10152-02 95 01-1: измененные разделы, подразделы и пункты документа, а также добавленные разделы, подразделы и пункты.

При администрировании ОС с установленным оперативным обновлением согласно бюллетеню № 2022-1121SE47 рекомендуется руководствоваться документом РУСБ.10152-02 95 01-1 совместно с настоящим руководством.

2. ПЕРЕЧЕНЬ ИЗМЕНЕНИЙ

2.1. Пункт «2.3.1. Уровень защищенности «Базовый»»

Заголовок пункта 2.3.1 изложить в редакции:

2.3.1. Базовый уровень защищенности («Орел»)

2.2. Пункт «2.3.2. Уровень защищенности «Усиленный»»

Заголовок пункта 2.3.2 изложить в редакции:

2.3.2. Усиленный уровень защищенности («Воронеж»)

2.3. Пункт «2.3.3. Уровень защищенности «Максимальный»»

Заголовок пункта 2.3.3 изложить в редакции:

2.3.3. Максимальный уровень защищенности («Смоленск»)

2.4. Подраздел «2.4. Создание LiveCD»

Подраздел 2.4 изложить в редакции:

2.4. Создание LiveCD

LiveCD — это ОС, загружаемая с носителя (CD/DVD-диска, USB-носителя) без установки на жесткий диск, при этом пользователю доступен весь функционал ОС.

В состав ОС входит инструмент командной строки `live-build-astra` для сборки образа LiveCD в формате `iso` (Live-образа). Параметры инструмента приведены в таблице 2.

Таблица 2

Параметр	Описание
`-h`	Вывести справку
`-d <distribution>`	Указать название дистрибутива, для которого создается LiveCD
`-i "<filename>"`	Указать абсолютный путь к ISO-файлу источника пакетов (репозитория) ОС, для которой создается LiveCD. Несколько ISO-файлов могут быть указаны через точку с запятой
`-r "<URL>"`	Указать сетевой адрес репозитория ОС, для которой создается LiveCD. Несколько репозиториев могут быть указаны через точку с запятой

Собрать Live-образ ОС можно с использованием ISO-образа диска с дистрибутивом ОС и репозитория со средствами разработки ОС (базовый репозиторий).

Примечание. Количество используемых ISO-файлов или сетевых репозиториев не должно превышать четырех. Первым должен быть указан тот источник, на основе которого будет собран `debootstrap` для Live-образа.

При запуске команды `live-build-astra` без параметров Live-образ будет собран на основе текущей ОС с использованием в качестве репозиториев первые два источника

из файла /etc/apt/sources.list. При этом в файле /etc/apt/sources.list первым должен быть указан источник, на основе которого будет собран debootstrap.

При сборке Live-образа в него не добавляются некоторые стандартные пакеты ОС. Список исключаемых пакетов приведен в файле /usr/share/live-build-astra/customyze/shrink.roster. Не рекомендуется расширять список исключаемых пакетов, так как пакеты будут исключены со всеми зависимостями, что может привести к неработоспособности собранного Live-образа.

При необходимости добавить в сборку Live-образа дополнительные пакеты из репозиториев требуется указать имена пакетов (без указания версии) в файле /usr/share/live-build-astra/customyze/astra_extend.list. Каждый пакет должен быть указан на отдельной строке. При указании пакета в файле /usr/share/live-build-astra/customyze/astra_extend.list необходимо проверить отсутствие данного пакета в файле /usr/share/live-build-astra/customyze/shrink.roster и при наличии — исключить его.

При необходимости добавить в сборку Live-образа дополнительные пакеты, отсутствующие в репозиториях, следует deb-файлы данных пакетов скопировать в каталог /usr/share/live-build-astra/customyze/extra_pkgs.

Также при сборке Live-образа возможно использовать дополнительные репозитории, в которых будет выполняться поиск пакетов. Для добавления дополнительных репозиториев необходимо указать их описание в файле /usr/share/live-build-astra/customyze/external_repo.list. При этом пакеты, которые требуется добавить из дополнительных репозиториев, необходимо указать в файле /usr/share/live-build-astra/customyze/astra_extend.list.

Примечание. Использовать дополнительные репозитории следует с осторожностью, так как полученный в результате сборки набор пакетов может оказаться неработоспособным.

Live-образ, собранный в результате выполнения команды live-build-astra, размещается в каталоге /opt/live_CD.

В случае неудачной сборки информацию об ошибках можно посмотреть в журнале сборки /opt/live_report/build.log.

Созданный Live-образ можно использовать для загрузки ОС как с CD/DVD-диска, так и с USB-носителя.

Для подготовки USB-носителя необходимо ISO-файл с Live-образом побайтово записать на USB-накопитель с использованием команды dd либо с помощью утилиты fly-admin-iso (описание утилиты приведено в электронной справке).

Пример

Запись ISO-образа `livecd.iso` на подключенный USB-носитель, представленный в системе файлом устройства `/dev/sdb`

```
dd if=livecd.iso of=/dev/sdb bs=1M
```

2.5. Подраздел «6.8. Программный коммутатор Open vSwitch»

Подраздел 6.8 изложить в редакции:

6.8. Программный коммутатор Open vSwitch

Open vSwitch (OVS) — программный многоуровневый коммутатор для виртуальных сетей, обеспечивающий агрегацию портов, обнаружение петель, зеркалирование портов, сбор статистики о трафике на NetFlow-коллектор, изоляцию сети с помощью VLAN путем тегирования портов, а также фильтрацию сетевого трафика и централизованное управления программными коммутаторами с помощью протокола OpenFlow.

Архитектура OVS состоит из трех основных компонентов: базы данных, программного коммутатора и управляющего контроллера. На каждом из физических узлов вместе с гипервизором располагаются собственные БД и коммутатор. БД обеспечивает хранение всей конфигурации своего узла: настройки интерфейсов, портов, правила и др. Коммутатор передает пакеты. Распределенность OVS достигается с помощью контроллера.

Коммутация пакетов происходит на уровне ядра, также поддерживается коммутация в пользовательском пространстве.

ВНИМАНИЕ! Программный коммутатор Open vSwitch не поддерживает классификационные метки и не может использоваться при включенном в ОС режиме мандатного управления доступом (МРД).

6.8.1. Установка

На каждом узле, предназначенном для включения в сеть, должен быть установлен пакет программного коммутатора OVS. Для установки выполнить команду:

```
sudo apt install openvswitch-switch
```

6.8.2. Конфигурирование коммутатора

Конфигурация всех коммутаторов OVS и портов, а также настройки поддерживаемых протоколов хранятся в собственной базе данных OVS (OVSDB). В стандартной конфигурации в OVSDB существуют следующие таблицы:

- Open_vSwitch;
- Bridge;
- Port;
- Interface;
- Flow_Table;

- QoS;
- Mirror;
- Controller;
- Manager;
- NetFlow;
- SSL;
- sFlow;
- IPFIX;
- Flow_Sample_Collector_Set.

После установки OVS почти все таблицы пусты, так как конфигурация отсутствует. Инструмент командной сроки `ovs-vsctl` используется для конфигурирования OVS и для внесения изменений в OVSDB. Синтаксис команды:

```
ovs-vsctl -- [<параметр>] <команда> [<параметры_команды>] [-- [<параметр>] <команда> [параметры_команды]]...
```

Для каждой виртуальной сети необходимо создать OVS с одним и тем же именем на всех узлах.

Создание программного коммутатора осуществляется командой:

```
ovs-vsctl add-br <имя_коммутатора>
```

Добавить в коммутатор новый порт возможно командой:

```
ovs-vsctl add-port <имя_коммутатора> <имя_порта>
```

Пример

Для подключения программного коммутатора `ovs-sw0` к внешней сети необходимо добавить ему в качестве порта физический интерфейс `eth0`:

```
ovs-vsctl add-port ovs-sw0 eth0
```

Для просмотра информации об установленных коммутаторах используется команда:

```
ovs-vsctl show
```

Для просмотра информации о портах используется команда:

```
ovs-vsctl list port
```

Для вывода списка портов, подключенных к конкретной VLAN, выполнить команду:

```
ovs-vsctl find port tag=10
```

Полное описание использования инструмента `ovs-vsctl` приведено в `man ovs-vsctl`.

Конфигурация OVS, заданная командами `ovs-vsctl`, применяется автоматически. Управление интерфейсом OVS осуществляется путем редактирования файла `/etc/network/interfaces`.

2.6. Пункт «7.1.2. Пример настройки кластера»

Пункт 7.1.2 изложить в редакции:

7.1.2. Пример настройки кластера

Настройка Pacemaker и Corosync на примере двух серверов с ОС: `server-1` и `server-2`. Оба сервера должны видеть друг друга по имени, для этого должен быть настроен DNS или в файле `/etc/hosts` содержаться соответствующие записи. Для настройки необходимо выполнить следующие действия:

1) на каждом сервере настроить синхронизацию времени по сети (служба `ntp`);

2) на каждом сервере удалить возможно сохранившуюся предыдущую конфигурацию кластера:

```
sudo pcs cluster destroy
```

3) на каждом сервере установить одинаковый пароль (например, 12345678) пользователю `hacluster`:

```
sudo passwd hacluster
```

4) на первом (главном) сервере настроить авторизацию, выполнив команду:

```
sudo pcs host auth server-1 server-2 -u hacluster -p 12345678
```

Результат выполнения команды:

```
server-2: Authorized
server-1: Authorized
```

5) на первом сервере создать и запустить кластер, выполнив команду:

```
sudo pcs cluster setup --force --start mycluster server-1 server-2
```

где `mycluster` — имя создаваемого кластера.

Результат выполнения команды:

```
No addresses specified for host 'server-1', using 'server-1'
No addresses specified for host 'server-2', using 'server-2'
Destroying cluster on hosts: 'server-1', 'server-2'...
server-1: Successfully destroyed cluster
server-2: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'server-1', 'server-2'
server-1: successful removal of the file 'pcsd settings'
server-2: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'server-1', 'server-2'
server-1: successful distribution of the file 'corosync authkey'
server-1: successful distribution of the file 'pacemaker authkey'
server-2: successful distribution of the file 'corosync authkey'
server-2: successful distribution of the file 'pacemaker authkey'
Synchronizing pcsd SSL certificates on nodes 'server-1', 'server-2'...
```

```
server-1: Success
server-2: Success
Sending 'corosync.conf' to 'server-1', 'server-2'
server-1: successful distribution of the file 'corosync.conf'
server-2: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
Starting cluster on hosts: 'server-1', 'server-2'...
```

6) на обоих серверах перезапустить службу pcsd:

```
sudo systemctl restart pcsd
```

7) на первом сервере включить автозапуск кластера:

```
sudo pcs cluster enable --all
```

Результат выполнения команды:

```
server-1: Cluster Enabled
server-2: Cluster Enabled
```

8) для текущего кластера, состоящего из двух узлов, задать базовые настройки, выполнив команды:

```
sudo pcs property set stonith-enabled=false
sudo pcs property set symmetric-cluster=false
sudo pcs property set no-quorum-policy=ignore
```

Для управления кластером Pacemaker используются инструменты командной строки pcs и crm_mon.

Для проверки статуса кластера выполнить команду:

```
sudo pcs status
```

Результат выполнения команды:

```
Cluster name: mycluster
Stack: corosync
Current DC: server-1 (version 2.0.1-9e909a5bdd) - partition with quorum
Last updated: Wed Jul 27 16:08:22 2022
Last change: Wed Jul 27 16:07:41 2022 by root via cibadmin on server-1

2 nodes configured
0 resources configured

Online: [ server-1 server-2 ]

No resources

Daemon Status:
```

```
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled
```

Управление кластером может также осуществляться через веб-интерфейс:

```
https://server-1:2224/
```

2.7. Подпункт «8.2.6.2. Использование RPC интерфейса»

Пятый абзац подпункта 8.2.6.2 изложить в редакции:

Существует ряд специальных RPC команд, применяемых к любому компьютеру домена ALD (в качестве аргумента команды может указываться имя компьютера):

- `rpc-status` — получение информации о роли компьютера в домене;
- `rpc-statistics` — получение статистической информации о RPC сервере `aldd` указанного компьютера;
- `rpc-execute` — выполнение указанной команды на удаленном компьютере (выполнение команды возможно только от имени учетной записи `admin/admin`).

2.8. Пункт «8.3.12. Сквозная аутентификация в СУБД»

Пункт 8.3.12 изложить в редакции:

8.3.12. Сквозная аутентификация в СУБД

Для работы СУБД PostgreSQL с FreeIPA необходимо выполнение следующих условий:

1) наличие в системах, на которых функционируют сервер и клиенты СУБД PostgreSQL, установленного пакета клиентской части FreeIPA `freeipa-client`;

2) разрешение имен должно быть настроено таким образом, чтобы имя системы разрешалось, в первую очередь, как полное имя (например, `postgres.example.ru`);

3) клиентская часть FreeIPA должна быть настроена на используемый FreeIPA домен (8.3.6).

Подробное описание работы с защищенной СУБД PostgreSQL приведено в документе РУСБ.10152-02 95 01-2.

Для обеспечения совместной работы сервера СУБД PostgreSQL с FreeIPA необходимо, чтобы сервер СУБД PostgreSQL функционировал как служба Kerberos. Выполнение данного условия требует наличия в БД Kerberos принципала для сервера СУБД PostgreSQL, имя которого задается в формате:

```
postgres/hostname@realm
```

где `hostname` — полное доменное имя системы, на которой функционирует сервер СУБД PostgreSQL;

`realm` — имя домена FreeIPA.

Для обеспечения совместной работы сервера СУБД PostgreSQL с FreeIPA выполнить следующие действия:

1) создать в БД FreeIPA с помощью утилиты администрирования FreeIPA принципала, соответствующего устанавливаемому серверу PostgreSQL. Принципал создается с автоматически сгенерированным случайным ключом;

Пример

```
ipa service-add postgres/postgres.example.ru
```

2) создать файл ключа Kerberos для сервера СУБД PostgreSQL с помощью утилиты администрирования FreeIPA `ipa service-add`.

Пример

Создание файла ключа Kerberos на контроллере домена

```
ipa-getkeytab -s domain.example.ru -k /etc/apache2/keytab
     -p HTTP/apache2.example.ru
```

Полученный файл должен быть доступен серверу СУБД PostgreSQL по пути, указанному в конфигурационном параметре `krb_server_keyfile` (для приведенного примера путь `/etc/apache2/keytab`). Пользователю, от имени которого работает сервер СУБД PostgreSQL (по умолчанию `postgres`), должны быть предоставлены права на чтение данного файла;

3) назначить владельцем файла `krb5.keytab` пользователя `postgres`, выполнив команду:

```
chown postgres /etc/postgresql/x.x/main/krb5.keytab
```

4) задать в конфигурационном файле сервера СУБД PostgreSQL `/etc/postgreslq/x.x/main/postgresql.conf` значение для параметра `krb_server_keyfile`:

```
krb_server_keyfile = '/etc/postgresql/x.x/main/krb5.keytab'
```

5) указать для внешних соединений в конфигурационном файле сервера СУБД PostgreSQL `/etc/postgreslq/x.x/main/pg_hba.conf` метод аутентификации `gss`.

Пример

```
host all all 192.168.32.0/24 gss
```

2.9. Пункт «9.2.1. Установка Docker»

Пункт 9.2.1 изложить в редакции:

9.2.1. Установка Docker

Установка Docker возможна либо через графический менеджер пакетов Synaptic, либо через терминал с помощью команды:

```
sudo apt install docker.io
```

После установки возможно добавить пользователя в группу `docker`, что позволит работать с Docker без использования `sudo`.

Для включения пользователя в группу `docker` выполнить команду:

```
sudo usermod -aG docker <имя_пользователя>
```

Текущего пользователя можно включить в группу командой:

```
sudo usermod -aG docker $USER
```

Для применения действия необходимо выйти из текущей сессии пользователя и зайти повторно.

2.10. Пункт «9.2.2. Работа с Docker»

Пункт 9.2.2 изложить в редакции:

9.2.2. Работа с Docker

Полный список команд для работы с Docker доступен на странице помощи:

```
docker help
```

Информацию о параметрах конкретной команды можно получить на странице помощи или в справочной странице man.

Пример

```
docker attach --help
man docker-attach
```

ВНИМАНИЕ! Описание работы с образами и контейнерами Docker приведено для привилегированного режима. Данный режим не рекомендуется к применению в связи с потенциальной небезопасностью использования контейнеров в привилегированном режиме. Рекомендуется работать с Docker в непривилегированном (rootless) режиме в соответствии с описанием 9.2.3.

2.11. Подпункт «9.2.2.1. Работа с Docker»

Подпункт 9.2.2.1 изложить в редакции:

9.2.2.1. Создание образа Docker

Образ — это шаблон контейнера, включающий в себя:

1) базовую файловую систему;

2) слои — изменения в файловой системе, расположенные друг над другом в том порядке, в котором эти изменения были произведены;

3) параметры выполнения, используемые при запуске контейнера из данного образа.

Примечание. Из одного образа возможно запускать несколько контейнеров.

Каждый слой образа представляет собой инструкцию, выполняемую в базовой файловой системе при создании образа. В процессе работы контейнера изменения файловой системы образуют новый слой контейнера, а слои образа остаются неизменными.

Слои могут быть последовательно записаны в текстовом документе, который называется докерфайлом (Dockerfile).

Образ возможно создать тремя способами:

- из chroot-окружения;
- с помощью докерфайла.
- на основе контейнера.

Создание образа из chroot-окружения

Для создания собственных образов Docker из chroot-окружения необходимо установить пакет debootstrap. Это можно сделать либо с помощью графического менеджера пакетов Synaptic, либо из терминала, выполнив команду:

```
sudo apt -y install debootstrap
```

Для создания образа Docker необходимо:

1) собрать chroot-окружение;
2) настроить chroot-окружение;
3) конвертировать chroot-окружение в образ Docker.

Сборка chroot-окружения выполняется инструментом командной строки debootstrap от имени администратора.

Загрузка пакетов для сборки chroot-окружения может быть выполнена из репозитория, доступного по сети.

Пример

```
sudo debootstrap --verbose
--components=main,contrib,non-free 4.7_arm /var/docker-chroot
http://dl.astralinux.ru/astra/stable/4.7_arm/repository-main
```

где 4.7_arm — код дистрибутива;

/var/docker-chroot — каталог сборки окружения;

http://dl.astralinux.ru/astra/stable/4.7_arm/repository-main — расположение репозитория в сети.

Загрузка пакетов для сборки chroot-окружения также может быть выполнена из репозитория в локальной ФС.

При сборке chroot-окружения для удобства дальнейшей работы можно сразу установить пакеты ncurses-term, mc, locales, nano, gawk, lsb-release, acl, perl-modules.

Пример

```
sudo debootstrap --verbose --include ncurses-term,mc,locales,nano,gawk,
lsb-release,acl,perl-modules-5.28
4.7_arm /var/docker-chroot file:///srv/repo
```

где `4.7_arm` — код дистрибутива;

`/var/docker-chroot` — каталог сборки окружения;

`file:///srv/repo` — каталог локального репозитория.

Примечание. При включенном МРД и/или МКЦ рекомендуется размещать каталог сборки `chroot`-окружения в `/var`. Данный каталог имеет метку безопасности `3:63:-1:ccnr`, что позволяет создавать в нем файловые объекты с любыми метками безопасности. Для работы пользователей в непривилегированном режиме администратором системы должен быть создан доступный пользователю каталог с необходимой меткой безопасности.

Настройка `chroot`-окружения выполняется от имени администратора в следующей последовательности:

1) при необходимости настроить для `chroot`-окружения разрешение имен в файле `/etc/resolv.conf` и список репозиториев в `/etc/apt/sources.list` (например, скопировать одноименные файлы из корневой ФС в каталог для `chroot`-окружения);

2) перейти в `chroot`-окружение командой `sudo chroot` и обновить пакеты окружения:

```
sudo chroot /var/docker-chroot
apt update
apt dist-upgrade
exit
```

Для создания образа Docker следует добавить настроенное `chroot`-окружение в архив с инструмента утилиты `tar`, запущенной от имени администратора, и конвертировать полученный архив в образ командой `docker import`.

Пример

Создать образ `wiki/astralinux:se` из `chroot`-окружения:

```
sudo tar -C /var/docker-chroot -cpf - . | sudo docker import -
wiki/astralinux:se --change "ENV PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"  --change
'CMD ["/bin/bash"]'
```

где `-C /var/docker-chroot` — задать каталог в качестве рабочего каталога для архивирования;

`-cpf - .` — создать новый архив из рабочего каталога с сохранением разрешений, установленных на входящие в него каталоги и файлы, и передать архив в `stdin`;

`docker import -` — импортировать данные для создания образа из `stdout`;

`-change "ENV PATH ..."` — задать переменную окружения `PATH`;

`-change 'CMD ["/bin/bash"]'` — задать команду, которая будет автоматически выполнена в контейнере при запуске контейнера из данного образа.

Если все операции выполнены успешно, то созданный образ будет отображаться в списке образов, доступном по команде:

```
sudo docker images
```

Для запуска контейнера из созданного образа используется команда:

```
sudo docker run -it --rm <имя_образа>
```

где `-i` — запустить контейнер в интерактивном режиме;

`-t` — выделить терминал для контейнера;

`--rm` — удалить контейнер после выхода из него.

Создание образа с использованием докерфайла

Докерфайл представляет собой инструкции для создания образа Docker. Используя инструкции из докерфайла и контекст (совокупность каталогов и файлов в указанном месте) возможно создавать новые образы на основе существующих с помощью команды:

```
docker build
```

При этом полное содержимое контекста рекурсивно пересылается службе `dockerd` и командами, указанными в докерфайле, может быть скопировано в создаваемый образ. Поэтому не рекомендуется использовать в качестве контекста корневой каталог файловой системы ОС

Местоположение контекста может быть задано как путь к каталогу в файловой системе либо как ссылка на репозиторий в сети.

По умолчанию используется докерфайл с именем `Dockerfile`, расположенный в каталоге контекста сборки. Произвольное расположение докерфайла задается параметром `-f`:

```
docker build -f <путь_к_докерфайлу> .
```

Перед выполнением инструкций в докерфайле проводится их проверка на корректность. Если в инструкциях содержится ошибка (например, неправильный синтаксис), при попытке собрать образ будет выведено сообщение об ошибке.

Пример

Создать образ с использованием докерфайла, содержащего несуществующую инструкцию RUNCMD:

```
docker build -t test/myimg .
```

В терминале будет выведено сообщение об ошибке:

```
Sending build context to Docker daemon 2.048 kB
Error response from daemon: Unknown instruction: RUNCMD
```

При создании нового образа инструкции выполняются последовательно и результат выполнения каждой инструкции записывается в отдельный слой образа.

Пример

Для сборки нового образ на основе существующего образа wiki/astralinux:se следует:

Далее по тексту...

2.12. Подпункт «9.2.2.6. Работа с Docker в непривилегированном режиме»

Изменить нумерацию подпункта 9.2.2.6 на 9.2.3 и изложить в редакции:

9.2.3. Работа с Docker в непривилегированном режиме

Работа с образами и контейнерами Docker в непривилегированном (rootless) режиме подразумевает работу от имени пользователя без использования механизма sudo. В непривилегированном режиме служба контейнеризации и контейнеры не получают прав суперпользователя в хостовой ОС, при этом для приложения в контейнере служба контейнеризации работает как суперпользователь. Режим не поддерживается в ядре hardened.

Для работы с Docker в непривилегированном режиме используется инструмент командной строки rootlessenv, который настраивает окружение для работы в данном режиме. При работе в непривилегированном режиме инструмент rootlessenv должен использоваться вместо механизма sudo в командах при настройке и работе с образами и контейнерами Docker (см. 9.2.2).

Для настройки работы в режиме rootless необходимо выполнить следующие шаги:

1) установить Docker в соответствии с 9.2.1;

2) установить пакет rootless-helper-astra для использования непривилегированного режима :

```
sudo apt install rootless-helper-astra
```

3) запустить службы rootless Docker для пользователя, который будет использовать образы и контейнеры Docker в непривилегированном режиме:

```
sudo systemctl start rootless-docker@<имя_пользователя>
```

4) при необходимости настроить автозапуск служб rootless Docker выполнить:

```
sudo systemctl enable rootless-docker@<имя_пользователя>
```

Запуск и настройка автозапуска служб могут быть выполнены для нескольких пользователей, для этого необходимо выполнить соответствующие команды отдельно для каждого пользователя.

Чтобы запустить контейнер от имени текущего пользователя с использованием `rootlessenv`, следует выполнить:

```
rootlessenv docker run --rm -ti <имя_образа>
```

Для запуска контейнера от имени произвольного пользователя с `rootlessenv` выполнить:

```
sudo -u <имя_пользователя> rootlessenv docker run --rm -ti <имя_образа>
```

Работа с образами и контейнерами, созданными в режиме rootless, возможна только в режиме rootless.

Для просмотра списка контейнеров, созданных в режиме rootless, выполнить команду:

```
rootlessenv docker container list
```

Работа с `rootless-helper-astra` и `rootlessenv` более подробно описана в `man rootless-helper-astra` и `man rootlessenv`, соответственно.

Описание работы с образами и контейнерами Docker в непривилегированном режиме с ненулевыми метками безопасности приведено в документе РУСБ.10152-02 97 01-1.

2.13. Раздел «11. Защищенная графическая подсистема»

Ввести новый подраздел после 11.6, а также изменить нумерацию подраздела «Мандатное управление доступом» с 11.7 на 11.8 в связи с добавлением нового подраздела:

11.7. Блокировка экрана при бездействии

Блокировка экрана при неактивности задается в конфигурационных файлах типов сессий `*themerc*`, расположенных в каталоге пользователя `/home/<имя_пользователя>/.fly/theme/`, следующими параметрами:

```
ScreenSaverDelay=0/<время_неактивности_в_секундах>
LockerOnSleep=true/false
LockerOnDPMS=true/false
LockerOnLid=true/false
LockerOnSwitch=true/false
```

При этом имена актуальных для сессии пользователя конфигурационных файлов начинаются с `current`, а файлы, имена которых начинаются с `default`, используются для создания и восстановления файлов `current`.

При создании учетной записи пользователя и его первом входе конфигурационные файлы `default.themerc*` копируются из каталога `/usr/share/fly-wm/theme/` в каталог пользователя `/home/<имя_пользователя>/.fly/theme/`.

Пользователю доступно управление блокировкой экрана своей сессии при неактивности из графической утилиты `fly-admnin-theme` (см. электронную справку).

Администратору для управления блокировкой экрана пользователей, в т.ч. централизованного, доступен конфигурационный файл `/usr/share/fly-wm/theme.master/themerc`. В файле указываются строки:

```
[Variables]
ScreenSaverDelay=0/<время_неактивности_в_секундах>
LockerOnSleep=true/false
LockerOnDPMS=true/false
LockerOnLid=true/false
LockerOnSwitch=true/false
```

При входе пользователя в сессию после считывания параметров из конфигурационных файлов пользователя проверяется наличие файла `/usr/share/fly-wm/theme.master/themerc` с секцией `[Variables]`. При наличии файла из него считываются параметры, и считанные параметры переопределяют аналогичные параметры, считанные ранее из конфигурационных файлов пользователя.

В ОС выполняется мониторинг каталога `/usr/share/fly-wm/theme.master/` и файла `/usr/share/fly-wm/theme.master/themerc`. При создании/изменении файла `/usr/share/fly-wm/theme.master/themerc` срабатывает механизм мониторинга и параметры из файла считываются и применяются к текущим сессиям всех пользователей.

Каталог `/usr/share/fly-wm/theme.master/` может являться разделяемым ресурсом.

Пользователю не доступна возможность переопределить параметры, заданные в `/usr/share/fly-wm/theme.master/themerc`.

11.8. Мандатное управление доступом

2.14. Подраздел «15.1. Аудит»

Подраздел 15.1 изложить в новой редакции и ввести новый подраздел после 15.1 с соответствующим изменением нумерации подраздела «Средства централизованного протоколирования» с 15.2 на 15.3:

15.1. Аудит

В ОС отправка и регистрация информации о событиях в системе осуществляется в соответствии со стандартом Syslog. Стандарт определяет формат сообщений о событиях и правила их передачи и регистрации в журналах. Основное расположение файлов журналов — системный каталог `/var/log`.

Аудит основных системных событий с момента запуска системы ведется в системном журнале `/var/log/syslog`.

Аудит событий постановки/снятия с контроля целостности исполняемых модулей и файлов данных, а также событий неудачного запуска неподписанных файлов осуществляется в журнале `/var/log/kern.log`.

Аудит событий создания/удаления/изменения настроек учетных записей пользователей и начала/окончания сеансов работы учетных записей пользователей осуществляется в журнале `/var/log/auth.log`.

Аудит событий изменения для учетных записей полномочий по доступу к информации осуществляется в журнале `/var/log/auth.log`.

Аудит событий смены аутентифицирующей информации учетных записей осуществляется в журнале `/var/log/auth.log`.

Аудит событий вывода текстовых (графических) документов на бумажный носитель осуществляется в журнале `/var/log/cups/page_log`.

Для аудита ОС также могут использоваться журналы различных служб и программ.

Для регистрации событий безопасности в ОС используется служба аудита `auditd`, описание которой приведено в РУСБ.10152-02 97 01-1.

15.2. Подсистема регистрации событий

В ОС реализована подсистема регистрации событий, которая собирает информацию о событиях из различных источников и предоставляет инструменты для просмотра собранных данных и реагирования на события.

Подсистема регистрации событий включает следующие инструменты:

1) менеджер и маршрутизатор событий `syslog-ng` — основная служба подсистемы регистрации событий, которая обеспечивает регистрацию событий в соответствии со стандартом Syslog. Служба `syslog-ng` принимает информацию о событиях из различных источников (события от `auditd`, собственные подключаемые модули, файлы, прикладное ПО и др.), выполняет фильтрацию и обработку полученных данных и, в зависимости от конфигурации, сохраняет в файл, отправляет по сети и т. д.;

2) модуль `syslog-ng-mod-astra` — модуль для `syslog-ng`, выполняющий дополнительную обработку и фильтрацию событий;

3) `astra-event-watcher` — демон уведомления пользователя о событиях, обработанных менеджером `syslog-ng`;

4) `ksystemlog` («Системный журнал») — графическая утилита просмотра журналов событий;

5) `fly-admin-events` («Настройка регистрации системных событий») — графическая утилита для настройки регистрируемых событий (описание утилиты приведено в электронной справке);

6) astra-admin-events — инструмент командной строки для импорта и экспорта конфигурационных настроек регистрации событий. Порядок использования инструмента приведен на странице помощи astra-admin-events --help;

7) fly-event-viewer («Журнал системных событий») — графическая утилита для просмотра зарегистрированных событий (описание утилиты приведено в электронной справке).

Для установки подсистемы регистрации событий выполнить команду:

```
sudo apt install syslog-ng syslog-ng-mod-python syslog-ng-mod-astra
astra-event-watcher fly-admin-events
```

Работа модуля syslog-ng-mod-astra настраивается в следующих конфигурационных файлах:

1) /etc/astra-syslog.conf — список регистрируемых событий;

2) /var/cache/astra-syslog/ — каталог с файлами настроек по умолчанию для каждого события.

Модуль syslog-ng-mod-astra информацию о событиях регистрирует в следующих файлах:

1) /parsec/log/astra/events — журнал событий в формате json (попытки запуска неподписанных файлов, успешная и неуспешная авторизация, данные о пользовательских сессиях и др.). Доступен для чтения и записи в конец файла только администратору;

2) /var/log/astra/prevlogin<имя_пользователя> — журнал в формате json сводной статистики предыдущих входов в систему пользователя <имя_пользователя>. Включает данные о последней завершенной сессии данного пользователя, а также количество успешных и неуспешных входов данного пользователя со времени начала ведения статистики. Доступен для чтения только пользователю <имя_пользователя>.

Действия с журналом событий (удаление, переименование, перемещение, ротация файла журнала событий) регистрируются подсистемой регистрации событий и указываются первой записью в журнале событий, а также регистрируются службой auditd и указываются в журнале аудита.

Действия с журналом аудита службы auditd (удаление, переименование, перемещение файла журнала аудита) регистрируются подсистемой регистрации событий и указываются в журнале событий.

Настройка отображения уведомлений демона astra-event-watcher выполняется в файле /usr/share/knotifications5/astra-event-watcher.notifyrc.

15.3. Средства централизованного протоколирования

2.15. Раздел «17. Средства разграничения доступа к подключаемым устройствам»

Текст между разделом 17 и подразделом 17.1 дополнить примечанием:

Примечание. Присвоение устройствам мандатных атрибутов доступа, а также мандатное управление доступом к устройствам реализуется только на уровне защищенности «Смоленск» при включенном мандатном управлении доступом. Правила разграничения доступа к устройствам, применяемые на уровне защищенности, отличном от «Смоленск», не должны содержать мандатные атрибуты доступа.

2.16. Подраздел «17.3. Разграничение доступа к устройствам на основе генерации правил udev»

Первый абзац подраздела 17.3 изложить в редакции:

17.3. Разграничение доступа к устройствам на основе генерации правил udev

Разграничение доступа к устройству осуществляется на основе генерации правил для менеджера устройств `udev`, которые хранятся в соответствующих файлах в каталогах `/etc/udev/rules.d` и `/run/udev/rules.d`. Генерация правил осуществляется автоматически для символьных и блочных устройств с использованием базы учета устройств, ведущейся в локальной системе (файл `/etc/parsec/PDAC/devices.cfg`) или в ALD/FreeIPA (см. раздел 8).