Axiom Linux
Рекомендации по обеспечению безопасности контейнеров

Axiom Linux | Ноябрь 2024

AXIOM JDK

Программное обеспечение АКСИОМ содержит программное обеспечение с открытым исходным кодом. Дополнительная информация о коде сторонних разработчиков доступна на сайте https://axiomjdk.ru/third_party_licenses. Для дополнительной информации о том, как получить копию исходного кода, можно обратиться по адресу info@axiomjdk.ru.

ДАННАЯ ИНФОРМАЦИЯ МОЖЕТ ИЗМЕНЯТЬСЯ БЕЗ ПРЕДВАРИТЕЛЬНОГО УВЕДОМЛЕНИЯ. АКСИОМ ПРЕДОСТАВЛЯЕТ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ "КАК ЕСТЬ" БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, АКСИОМ ПРЯМО ОТКАЗЫВАЕТСЯ ОТ ВСЕХ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ, ВКЛЮЧАЯ, НО НЕ ОГРАНИЧИВАЯСЬ ПОДРАЗУМЕВАЕМЫМИ ГАРАНТИЯМИ ТОВАРНОЙ ПРИГОДНОСТИ И ПРИГОДНОСТИ ДЛЯ ОПРЕДЕЛЕННОЙ ЦЕЛИ.

АКСИОМ НИ ПРИ КАКИХ ОБСТОЯТЕЛЬСТВАХ НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ЛЮБЫЕ КОСВЕННЫЕ, СЛУЧАЙНЫЕ, СПЕЦИАЛЬНЫЕ, ШТРАФНЫЕ ИЛИ КОСВЕННЫЕ УБЫТКИ, ИЛИ УБЫТКИ ОТ ПОТЕРИ ПРИБЫЛИ, ДОХОДА, ДАННЫХ ИЛИ ИСПОЛЬЗОВАНИЯ ДАННЫХ, ПОНЕСЕННЫЕ ВАМИ ИЛИ ЛЮБОЙ ТРЕТЬЕЙ СТОРОНОЙ, БУДЬ ТО В РЕЗУЛЬТАТЕ ДЕЙСТВИЯ ДОГОВОРА ИЛИ ДЕЛИКТА, ДАЖЕ ЕСЛИ АКСИОМ БЫЛО ПРЕДУПРЕЖДЕНО О ВОЗМОЖНОСТИ ТАКИХ УБЫТКОВ.

Использование любого программного продукта АКСИОМ регулируется соответствующим лицензионным соглашением, которое никоим образом не изменяется условиями данного уведомления. Программные продукты и фирменные наименования: Axiom JDK, Axiom JDK Pro, Axiom Runtime Container Pro, Axiom Linux, Libercat, Libercat Certified и АКСИОМ принадлежат АКСИОМ и их использование допускается только с разрешения правообладателя.

Товарный знак Linux® используется в соответствии с сублицензией от Linux Foundation, эксклюзивного лицензиата Линуса Торвальдса, владельца знака на всемирной основе. Java и OpenJDK являются товарными знаками или зарегистрированными товарными знаками компании Oracle и/или ее аффилированных лиц. Другие торговые марки являются собственностью их соответствующих владельцев и используются только в целях идентификации.

Содержание

1. Введение

В этом документе представлены несколько рекомендаций и сведений по безопасности, которые помогут вам защитить контейнеры и среду исполнения.

> **Примечание:**
>
> Примеры в этом документе в основном основаны на Podman, но вы можете применить все рекомендации как к настройке Docker, так и к Podman

2. Настройка хоста

Следующие рекомендации помогут вам безопасно настроить и обслуживать ваш хост-сервер.

- Регулярно обновляйте хост-систему контейнера, такую как ядро и другие программные компоненты. Убедитесь, что функции ядра, связанные с пространствами имен ядра, частными сетями и группами управления, обновлены до последней версии.
- Сконфигурируйте хост-систему контейнера так, чтобы она использовала минимальные настройки операционной системы и применяла все рекомендации по обеспечению безопасности. В идеале такие системы должны быть настроены только для размещения контейнеров и не использоваться ни для чего другого.

  Как правило, мы рекомендуем сократить количество служб, запущенных в одной системе, до необходимого минимума. Если некоторые службы необходимы для рабочего процесса, рассмотрите возможность переноса всех остальных служб для запуска в контейнерах, контролируемых Podman, или переноса их в другие хост-системы.

Настройте аудит для отслеживания действий в системе

Одной из наиболее важных мер безопасности на хостингах Linux является проведение надлежащего аудита конфиденциальных действий в системе. Инструментом, который может помочь в отслеживании таких действий, является поддержка аудита в ядре Linux и соответствующих инструментах/пакетах пользовательского пространства. Аудит позволяет системным администраторам отслеживать события, относящиеся к безопасности, и сообщать о них в журнале, таком как `audit.log`. Этот файл журнала может быть создан в локальной или удаленной системе для лучшей защиты.

Благодаря реализации Podman, позволяющей использовать "fork/exec" для запуска контейнеров, функция аудита работает более корректно и сохраняет необходимые данные в файлах журнала. В Docker `auditd` `auid` не задан в журнале, что означает, что системный администратор может видеть, что процесс, связанный с контейнером, получил доступ к конфиденциальной информации, но идентификатор не записывается в журнал. В Podman `auid` отображается в файле журнала аудита.

Например, исполняемый файл `cat` из `/usr/bin/coreutils` использовался для доступа к конфиденциальной информации, и Podman предоставляет всю необходимую информацию об идентификаторе, в то время как в Docker `auid` находится в неустановленном состоянии.

```
type=SYSCALL msg=audit(02/04/23 20:31:56.759:5) : arch=x86_64 syscall=open
success=yes exit=3 a0=0x7ffe5a873ecb a1=O_RDONLY a2=0x0 a3=0x0 items=1
```

```
ppid=2531 pid=2685 auid=test uid=root gid=root euid=root suid=root fsuid=root
egid=root sgid=root fsgid=root tty=pts0 ses=unset comm=cat
exe=/usr/bin/coreutils key=(null)
```

Рекомендуется использовать только Podman для настройки контейнерных хостов и использовать audit для отслеживания доступа к конфиденциальной информации.

Для настройки аудита в Axiom Linux требуется установить пакет аудита и запустить службу аудита следующим образом.

```
sudo rc-service auditd start
```

Используйте cgroups

Чтобы правильно настроить управление ресурсами, система должна быть настроена на использование Control Group v2 (**cgroup v2**). Выполните следующие действия для использования cgroup v2.

Настройте UIDs

Настройте `/etc/subuid` and `/etc/subgid`.

Podman запускает контейнер внутри пространства имен пользователя, которое сопоставляется с диапазоном UID, определенным для пользователя в `/etc/subuid` и `/etc/subgid`. Обновите эти файлы для каждого пользователя, которому разрешено создавать контейнеры. Если вы обновите либо "/etc/subuid", либо "/etc/subgid", все запущенные контейнеры, принадлежащие соответсвующим пользователям, должны быть остановлены. Это можно сделать автоматически с помощью следующей команды, которая останавливает все контейнеры для пользователя и завершает процесс приостановки:

```
podman system migrate
```

Уберите предупреждения об отсутствии systemd

Axiom Linux не использует `systemd`, и вы можете наблюдать следующее сообщение, если запустите Podman в Axiom Linux:

```
WARN[0000] "/" is not a shared mount, this could cause issues or missing mounts
with rootless containers.
```

Выполните следующую команду, чтобы исправить настройку.

```
sudo mount –make-rshared /
```

Установите пакет fuse-overlayfs

Обратите внимание, что пакет `fuse-overlayfs` по умолчанию не устанавливается вместе с пакетом Podman. Установите пакет с помощью команды `apk add`.

```
sudo apk add fuse-overlayfs
```

Включите cgroups

Выполните следующие команды.

```
sudo rc-update add cgroups
sudo rc-service cgroups start
```

Мы настоятельно рекомендуем использовать cgroups v2, которые можно настроить, отредактировав `/etc/rc.conf`. Назначьте опцию `unified` для `rc_cgroup_mode` и включите контроллеры. Возможно, вам придется запустить следующую команду, чтобы изменения вступили в силу.

```
sudo rc-service cgroups restart
```

Используйте среду выполнения crun

Чтобы ограничить ресурсы для cgroups v2, используйте в настройке соответствующую среду выполнения OCI. Мы рекомендуем настроить среду выполнения Podman `crun` в качестве среды выполнения по умолчанию. Для этого отредактируйте файл `/etc/containers/containers.conf` и измените значение `Default OCI runtime`, чтобы оно всегда было `crun`.

Загрузите дополнительные модули ядра

Загрузите следующие модули, поскольку они не загружаются по умолчанию.

```
sudo modprobe tun
sudo modprobe fuse
```

AppArmor

Мы рекомендуем использовать AppArmor при настройке хоста контейнера на Axiom Linux. Правильная конфигурация AppArmor помогает предотвратить вредоносные утечки и защищает конфиденциальные ресурсы хоста.

3. Запускайте контейнеры без root-прав

Разумная практика обеспечения безопасности заключается в том, чтобы никогда не запускать контейнеры с привилегиями root, поскольку привилегии root применяются для доступа к конфиденциальным ресурсам в самой хост-системе. Поскольку контейнеры с приложениями могут быть загружены из Интернета или иметь некоторые недостатки в системе безопасности, опасно предоставлять им root-привилегии.

В большинстве случаев нет необходимости запускать контейнеры с правами root. В зависимости от приложений существуют конкретные случаи, когда запуск контейнера от имени root имеет смысл. Одной из причин может быть то, что контейнерам требуется доступ к определенным подключениям, устройствам на хосте или необходимо прослушивать порты с адресами менее 1024 в сети хоста.

Таким образом, наиболее типичным режимом выполнения был бы без root-прав, и хотя Podman поддерживает его из коробки, он должен стать основным выбором для настройки контейнерных сред.

По замыслу Podman без root-прав запускается внутри контейнера с правами root, если его не перенастраивать. Эта политика означает, что все процессы в контейнере имеют список возможностей пространства имен по умолчанию и позволяют процессам действовать как root внутри пользовательского пространства имен, включая изменение их UID и изменение права собственности пользователя и группы на файлы, которые сопоставлены с пользовательским пространством имен.

4. Настройте пространство имен

Podman использует преимущества пользовательских пространств имен, так что root в контейнере сопоставляется с некорневым UID на хосте. Эта настройка позволяет Podman безопасно устанавливать пакеты и запускать службы из контейнера, не влияя на хост.

Администраторы могут использовать пространство имен пользователя для установки сопоставления идентификатора пользователя (UID) и идентификатора группы (GID) для запуска контейнера. Это означает, что процесс может выполняться как UID 0 внутри контейнера и как UID 123456 вне контейнера. Другими словами, если контейнерный процесс выходит за пределы контейнера, ядро Linux будет обрабатывать этот процесс как UID 123456.

Следующий пример настройки `--uidmap` предписывает Podman сопоставить диапазон из пяти тысяч (5000) UID внутри контейнера, начиная с UID 100000 вне контейнера (таким образом, диапазон равен 100000-104999) с диапазоном, начинающимся с UID 0 внутри контейнера (таким образом, диапазон равен 0-4999). Если процесс запущен с UID 1 внутри контейнера, то на хосте он равен 100001.

```
sudo podman run -d bellsoft/alpaquita-linux-base sleep 1000
sudo podman top --latest user huser
USER        HUSER
root        root

sudo podman run --uidmap 0:100000:5000 -d bellsoft/alpaquita-linux-base sleep
1000
sudo podman top --latest user huser
USER        HUSER
root        100000
```

Вы можете использовать следующие типы пространств имен:

- Mount (mnt): изолирует точки монтирования;
- Process ID (pid): изолирует идентификаторы процессов;
- Network (net): изолирует сетевой стек;
- Interprocess Communication (ipc): изолирует ресурсы межпроцессного взаимодействия;
- UTS: изолирует имена хостов и доменов;
- User ID (user): изолирует идентификаторы пользователя и группы;
- Control groups (cgroups): изолирует cgroups;

- Time: изолирует время.

5. Образы контейнеров и удаленные источники

Важно убедиться, что образы Podman получены и развернуты без изменений из исходного каталога с надежной репутацией и проверенной аутентификацией. При извлечении образов из удаленных источников убедитесь, что соединение защищено и что для запроса на получение используется протокол HTTPS. Небезопасно использовать каталоги образов не защищенные протоколом TLS. Обратите внимание на следующие рекомендации.

- Ищите образы по их полным именам вместо сокращенных, чтобы избежать возможности копирования из другого каталога.
- Вы можете настроить Podman для работы только с надежными образами из удаленного источника, так, чтобы образы были подписаны и подписи могли быть проверены с помощью локального открытого ключа.
- Образы можно подписывать аналогично пакетам, а хост Podman можно настроить так, чтобы изображения из удаленного реестра были подписаны и проверены перед локальным использованием.
- Создавайте надежно воспроизводимые образы из файлов-контейнеров и необходимых пакетов. Убедитесь, что в новых образах используются базовые образы, а имеющееся у вас программное обеспечение тщательно проверено на наличие уязвимостей безопасности. Для этого:
    - Определите фиксированную версию базового образа в файле-контейнере изображения;
    - Определите фиксированные версии пакета на этапах сборки файла-контейнера образа;
    - Убедитесь, что пакеты на этапах сборки используют надежные и проверенные источники и репозитории.
- Сократите до минимума количество пакетов, установленных на образах.

Мы не рекомендуем устанавливать ненужные пакеты в новые сборки образа. Правильный просмотр файлов-контейнеров помогает удалить ненужные этапы установки, а образы используются по своему основному назначению.

6. Другие настройки и ресурсы

Ограничьте использование процессора и памяти

Используйте параметр `-m` для ограничения используемой памяти и параметр `-c` для ограничения использования ЦП.

Ограничьте доступ к файлам контейнера

При создании и запуске контейнеров ограничьте доступ к файлам с помощью опции `-v <каталог хоста>:<каталог контейнера>:ro` или флага `--readonly`. Безопаснее явно создавать тома для приложений, работающих внутри контейнера. Мониторинг изменений файлов в этих томах может помочь предотвратить нарушения безопасности. Тома, предназначенные для записи в контейнере, необходимо регулярно очищать. Вот пример того, как использовать опцию `:ro` и смонтировать каталог хоста таким образом, чтобы каталог/файл хоста был доступен только для чтения в контейнере:

```
podman run -v /host_directory:/container_directory:ro axiom/axiom-linux-base
```

`host_directory` - это каталог/файлы хоста, которые будут смонтированы как том и станут доступны в "container_directory" в контейнере. Мы настоятельно не рекомендуем вам монтировать следующие конфиденциальные каталоги хост-системы во время работы контейнера. `/`, `/boot`, `/dev`, `/etc`, `/lib`, `/usr`, `/sys`, `/proc` и так далее.

Ограничьте перезапуск контейнера

Злонамеренный или случайный отказ в обслуживании (denial-of-service) может произойти с контейнером, который выдает много ошибок, поэтому ограничение перезапусков контейнера является хорошей практикой и может быть выполнено с помощью параметра `--restart=on-failure:N` при создании или запуске контейнера.

Ограничьте доступ к сети из контейнеров

Очень хорошей практикой является ограничение доступа к сети из контейнера, за исключением случаев, когда это необходимо для запуска определенных приложений. При публикации портов на хосте укажите IP-адрес интерфейса, к которому будет привязан порт, чтобы поверхность атаки была ограничена сетевым интерфейсом прослушивающим контейнер. Podman публикует на всех интерфейсах `0.0.0.0` по умолчанию, если IP-адрес не указан при использовании опции `--publish`.

Ознакомьтесь со следующими рекомендациями:

- Не запускайте SSH внутри контейнеров;
- Не перенаправляйте привилегированные порты (< 1024) внутри контейнеров;
- Не используйте параметр режима `--net=<имя хоста>` при запуске контейнера. Этот параметр предоставляет контейнеру полный доступ к локальным системным службам и небезопасен.

Проверьте разрешенные операции ядра в контейнерах

Следующие разрешенные операции ядра обычно предоставляются контейнеру по умолчанию. Просмотрите их и отключите ненужные.

```
– CHOWN
– DAC_OVERRIDE
– FSETID
– FOWNER
– NET_RAW
– SETGID
– SETUID
– SETFCAP
– SETPCAP
– NET_BIND_SERVICE
– SYS_CHROOT
– KILL
```

Параметр "--privileged" отключает меры безопасности, используемые для изоляции контейнера от хоста, поэтому его не следует использовать без крайней необходимости. Этот параметр устраняет барьеры, связанные с изолированными возможностями, устройствами с ограниченным доступом, точками монтирования и томами только для чтения и так далее.

Отслеживайте любое вредоносное использование ресурсов в контейнере

Podman включает в себя необходимые функции для мониторинга использования ресурсов контейнера, таких как потребление памяти, процессорное время, ввод-вывод и использование сети. Мониторинг использования ресурсов контейнера на предмет производительности, обнаружения ошибок и аномального поведения, такого как подозрительный трафик или неожиданная активность пользователя, также помогает обнаружить недостатки безопасности.

Регулярно очищайте хост-систему контейнера

Ненужные образы и контейнеры следует удалять из хост-системы, чтобы избежать мусора от образов и контейнеров и защититься от случайного запуска старого, неиспользуемого образа или контейнера, которые могут иметь возможные проблемы безопасности. В Podman есть опция `auto-update`, позволяющая автоматизировать процесс обновления для новых версий образов и контейнеров в соответствии с их настройками автоматического обновления.

Ограничьте системные вызовы в контейнерах

По умолчанию контейнеры Podman ограничивают системные вызовы, доступные контейнерам, на основе вызовов, определенных в файле `/usr/share/containers/seccomp.json`. Этот список действителен для контейнеров общего назначения и совместим с большинством контейнеров, поэтому нет необходимости добавлять дополнительные системные вызовы. Однако вы можете сузить круг системных вызовов, необходимых для запуска конкретного контейнера. Для этого создайте профиль, сгенерированный с помощью seccomp, используя следующую команду.

```
sudo podman run --annotation io.containers.trace-syscall=”of:<file_path>”
<other_options> <container_name>
```

Вы можете повторно использовать его позже с помощью опции `--security-opt seccomp=<file_path>`.

Проверьте runlabel перед выполнением

В Podman есть полезная функция для создания ярлыков для `podman run` со всеми необходимыми

метаданными, поэтому фактический запуск контейнера может быть выполнен следующим образом:

```
podman container runlabel <my_label> <image_name>
```

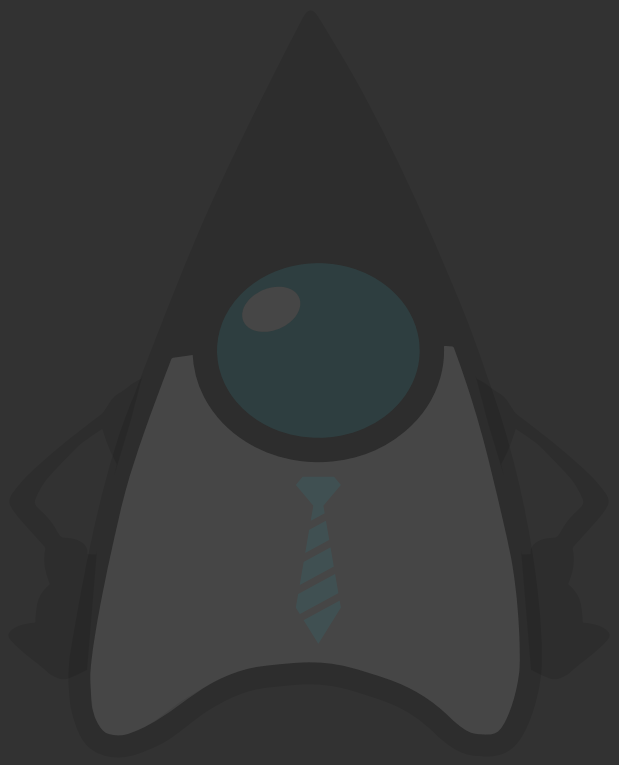
Перед фактическим выполнением мы рекомендуем проверить ярлык, указав опцию `runlabel --display`, чтобы избежать любой вредоносной команды `podman run`. Опция не выполняет команду label, а отображает только то, что будет выполнено.

Обезопасьте удаленное использование API

У Podman есть удаленная реализация API под названием `varlink`, которая работает через сокет. Конфигурация сокета аналогична сокету Unix, ограничивая его только локальным использованием. Запускайте этот сокет от имени пользователя, не являющегося root, особенно если контейнер не требует запуска от имени привилегированного пользователя. Сокет можно создать, выполнив следующую команду Podman.

```
podman varlink --timeout=0 unix:/run/user/$(id -u)/podman/io.podman
```

Создавая сокет с помощью команды Podman, вы гарантируете, что для сокета установлены надлежащие разрешения.



Axiom Linux
Рекомендации по обеспечению безопасности контейнеров

AXIOM JDK