```
Profile:   CIS Docker Benchmark Profile (cis-docker-benchmark)
Version:   2.1.4
Target:    local://
Target ID: 1897a396-b53f-525a-97ad-e0073be7931a

  ×  docker-4.1: Create a user for the container (1 failed)
       ✔  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["Config", "User"] is expected not to eq nil
       ×  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["Config", "User"] is expected to eq "ubuntu"

     expected: "ubuntu"
          got: ""

     (compared using ==)
```

docker-4.1: Убедитесь, что пользователь для контейнера создан (Вручную)

Обоснование:
Хорошей практикой является запуск контейнера от имени пользователя, не являющегося пользователем root, если это возможно. Это может быть либо через директиву USER в Dockerfile, либо через gosu или аналогичные программы, если они используется как часть директив CMD или ENTRYPOINT.

Последствия:
Работа от имени пользователя, не являющегося пользователем root, может вызвать трудности, когда вы хотите связать монтирование
тома с базового хоста. В этом случае необходимо убедиться, что пользователь, запускающий содержащийся процесс, может читать и записывать в связанный каталог в соответствии с
в соответствии с их требованиями.

Устранение:
Вы должны убедиться, что Dockerfile для каждого образа контейнера содержит следующую информацию:

USER <имя пользователя или ID>.

В данном случае имя пользователя или ID относится к пользователю, который был найден в базовом образе контейнера. Если в базовом образе контейнера не создан определенный пользователь, воспользуйтесь командой useradd, чтобы добавить определенного пользователя перед инструкцией USER в Dockerfile.
Например, добавьте следующие строки в Dockerfile для создания пользователя в контейнере:

```
RUN useradd -d /home/username -m -s /bin/bash username
USER username
```

Примечание: Если в образе есть пользователи, которые не нужны, следует подумать об их удалении. После удаления этих пользователей зафиксируйте образ, а затем создайте новые экземпляры контейнеров.
В качестве альтернативы, если невозможно установить директиву USER в Dockerfile, можно использовать сценарий выполняемый в разделах CMD или ENTRYPOINT Dockerfile, должен использоваться для того, чтобы обеспечить переключение процесса контейнера на пользователя, не являющегося пользователем root.

```
  ×  docker-4.2: Use trusted base images for containers
     ×  Environment variable DOCKER_CONTENT_TRUST content is expected to eq "1"

     expected: "1"
          got: nil

     (compared using ==)
```

docker-4.2: Убедитесь, что контейнеры используют только доверенные базовые образы (Вручную)

Обоснование:
Официальные репозитории содержат образы Docker, курируемые и оптимизированные сообществом Docker сообществом или их поставщиком. Нет никакой гарантии, что эти образы безопасны и не содержат не содержат уязвимостей безопасности или вредоносного кода. Поэтому следует проявлять осторожность проявлять осторожность при получении образов контейнеров от Docker и третьих сторон, а запуск этих образов следует рассматривать в соответствии с политикой безопасности организации.

Устранение:
Следующие процедуры полезны для установления доверия для конкретного образа.
- Настройте и используйте доверие к содержимому Docker.
- Просмотрите историю каждого образа Docker, чтобы оценить его риск, зависящий от чувствительности приложения, которое вы хотите развернуть с его помощью.
- Регулярно проверяйте образы Docker на наличие уязвимостей.

```
↺  docker-4.3: Do not install unnecessary packages in the container
↺  Do not install unnecessary packages in the container
```

docker-4.3: Убедитесь, что ненужные пакеты не установлены в контейнер (Вручную)

Обоснование:
Ненужное программное обеспечение не должно устанавливаться в контейнеры, поскольку это увеличивает их поверхность атаки. Следует устанавливать только пакеты, строго необходимые для правильной работы развертываемого приложения.

Устранение:
Вы не должны устанавливать в контейнер ничего, что не требуется. Вам следует рассмотреть возможность использования минимального базового образа, а не стандартных образов Centos, Debian или Red Hat, если это возможно. Некоторые из доступных вариантов включают BusyBox и Alpine.
Это не только значительно сократит размер образа, но и уменьшит количество программ, которые могут содержать векторы. частей программного обеспечения, которые могут содержать векторы для атак.

↺ docker-4.4: Rebuild the images to include security patches
↺ Rebuild the images to include security patches

docker-4.4: Обеспечить сканирование и восстановление образов для включения исправлений безопасности исправлений (вручную)

Обоснование:
Уязвимости - это лазейки или ошибки, которые могут быть использованы хакерами или злоумышленниками, а исправления безопасности - это обновления, устраняющие эти уязвимости. Инструменты сканирования уязвимостей изображений можно использовать для поиска уязвимостей в изображениях и последующей проверки доступные исправления для их устранения. Патчи обновляют систему до более новой кодовой базы, которая не содержит этих проблем, а наличие поддерживаемой версии кодовой базы очень важно, так как производители, как правило, не поставляют исправления для старых версий. версии, которая перестала поддерживаться. Патчи безопасности должны быть оценены перед и исправления должны быть внедрены в соответствии с политикой ИТ-безопасности организации. политики. Следует с осторожностью относиться к результатам, выдаваемым инструментами оценки уязвимостей, поскольку некоторые из них просто выдают результаты, основанные на баннерах программного обеспечения, и они могут быть не совсем точными. не совсем точными

Устранение:
Образы следует пересобрать, убедившись, что используется последняя версия базовых образов, чтобы поддерживать уровень исправлений операционной системы на должном уровне. После того, как образы контейнеры должны быть перезапущены с использованием обновленных образов.

× docker-4.5: Enable Content trust for Docker
     × Environment variable DOCKER_CONTENT_TRUST content is expected to eq "1"

     expected: "1"
          got: nil

     (compared using ==)

docker-4.5: Убедитесь, что доверие к содержимому для Docker включено (вручную)

Обоснование:
Доверие к содержимому обеспечивает возможность использования цифровых подписей для данных, отправляемых и получаемых из удаленных реестров Docker. Эти подписи позволяют на стороне клиента проверить личности и издателя определенных тегов образов и гарантируют происхождение
образов контейнеров.

Устранение:
Чтобы включить доверие к содержимому в оболочке bash, необходимо ввести следующую команду:

```
export DOCKER_CONTENT_TRUST=1
```

Также вы можете установить эту переменную окружения в файле профиля, чтобы доверие к содержимому доверие к содержимому включается при каждом входе в систему.

× docker-4.6: Add HEALTHCHECK instruction to the container image
     × #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста)..RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["Config", "Healthcheck"] is expected not to eq nil

     expected: value != nil
          got: nil

     (compared using ==)

docker-4.6: Убедитесь, что инструкции HEALTHCHECK были добавлены к образы контейнеров (Вручную)

Обоснование:
Важным элементом контроля безопасности является контроль доступности. Добавление инструкции HEALTHCHECK в в ваш образ контейнера гарантирует, что движок Docker будет периодически проверять запущенные экземпляры контейнеров по этой инструкции, чтобы убедиться, что контейнеры все еще работоспособны.
На основании результатов проверки работоспособности движок Docker может завершить работу контейнеров. На основании результатов проверки работоспособности движок Docker может завершить работу контейнеров, которые не отвечают должным образом, и создать новые.

Устранение:
Вы должны следовать документации Docker и перестроить образы контейнеров, чтобы включить инструкцию HEALTHCHECK.

```
  ×  docker-4.7: Do not use update instructions alone in the Dockerfile (6
failed)
     ×  Command: `docker history --no-trunc
sha256:384de598d957b2d3fc3c0cf4aa1f7660d63ffe07fd771523dc6599e4f7d02daf| grep -e
'update'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 weeks ago         RUN /b...
48.4MB    \n"

     (compared using ==)

     Diff:
     @@ -1,4 +1,8 @@
     +<missing>
3 weeks ago         RUN /bin/sh ...
     +<missing>
3 days ago          /bin/sh -c set ...
     +<missing>
3 days ago          /bin/sh -c apt-get ...
     +<missing>
3 days ago          /bin/sh -c set -eux;  ...
48.4MB

     ×  Command: `docker history --no-trunc
sha256:3a01fb98dddd0dc2955a19bed37a379349e7cf4cea51dd0f7bc8ef94611f5be1| grep -e
'update'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 weeks ago         RUN /b...
48.4MB    \n"

     (compared using ==)

     Diff:
     @@ -1,4 +1,8 @@
     +<missing>
3 weeks ago         RUN /bin/sh -c set ...
     +<missing>
3 days ago          /bin/sh -c set -ex;  ...
     +<missing>
3 days ago          /bin/sh -c apt-get ...
     +<missing>
3 days ago          /bin/sh -c set -eux; ...
48.4MB
```

```
 ×  Command: `docker history --no-trunc
sha256:a95645c4a87caf92f3b816aac47679d9fe137f4f6f1fe4a1164ec124178b6e1f| grep -e
'update'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 weeks ago         RUN /b...
48.4MB    \n"

     (compared using ==)

     Diff:
     @@ -1,4 +1,8 @@
     +<missing>
3 weeks ago         RUN /bin/sh -c set ...
     +<missing>
3 days ago          /bin/sh -c set -ex; ...
     +<missing>
3 days ago          /bin/sh -c apt-get ...
     +<missing>
3 days ago          /bin/sh -c set -eux; ...
48.4MB

     ×  Command: `docker history --no-trunc
sha256:1ce25a82f60076186033f4f7e85c396ddeadf439128ca91f52c83694bc5283cc| grep -e
'update'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 weeks ago         RUN /b...
48.4MB    \n"

     (compared using ==)

     Diff:
     @@ -1,4 +1,8 @@
     +<missing>
3 weeks ago         RUN /bin/sh -c set -eux;  ...
     +<missing>
3 days ago          /bin/sh -c set -ex;  apt-get ...
     +<missing>
3 days ago          /bin/sh -c apt-get update ...
     +<missing>
3 days ago          /bin/sh -c set -eux;  apt-get update;  ...

     ×  Command: `docker history --no-trunc
sha256:e480a148f48408d740a01c2c59585428d294e1f1c0ddabc89cd2335ce86a9bea| grep -e
'update'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 weeks ago         RUN /b...
48.4MB    \n"

     (compared using ==)

     Diff:
     @@ -1,4 +1,8 @@
     +<missing>
3 weeks ago         RUN /bin/sh -c set -eux;  ...
     +<missing>
3 days ago          /bin/sh -c set -ex;  apt-get ...
     +<missing>
3 days ago          /bin/sh -c apt-get update ...
```

```
     +<missing>
3 days ago          /bin/sh -c set -eux;  ...

     ×  Command: `docker history --no-trunc
sha256:2e0fee02aedc429050e0d898f0ffdb68ea0ed9dc3cd9b1623e293375a19e517b| grep -e
'update'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 weeks ago   RUN /bin/sh ...
48.4MB    \n"

     (compared using ==)

     Diff:
     @@ -1,4 +1,8 @@
     +<missing>
3 weeks ago   RUN /bin/sh -c set -eux ...
     +<missing>
3 days ago    /bin/sh -c set -ex;  apt-get ...
     +<missing>
3 days ago    /bin/sh -c apt-get update ...
     +<missing> 3 days ago    /bin/sh -c set -eux;  …
```

docker-4.7: Убедитесь, что инструкции по обновлению не используются в файлах Dockerfiles в одиночку (Руководство)

Обоснование:
Добавление инструкций обновления в одну строку в Dockerfile приведет к тому, что слой обновления будет кэшироваться. При последующей сборке любого образа с использованием той же инструкции это приведет к тому. будет использоваться ранее кэшированный слой обновлений, что потенциально предотвратит применение свежих обновлений к более поздним сборкам.
обновления не будут применены к последующим сборкам.

Устранение:
Вы должны использовать инструкции по обновлению вместе с инструкциями по установке и привязке версий для пакетов при их установке. Это предотвратит кэширование и заставит извлекать необходимые версии.
Также вы можете использовать флаг --no-cache в процессе сборки docker, чтобы избежать использования кэшированных слоев.

```
  ↺  docker-4.8: Remove setuid and setgid permissions in the images
     ↺  Use DevSec Linux Baseline in Container
```

4.8 Убедитесь, что разрешения setuid и setgid удалены (Вручную)

Обоснование:
разрешения setuid и setgid могут быть использованы для повышения привилегий. Несмотря на то, что эти разрешения могут быть иногда оправданно необходимы, вам следует рассмотреть возможность их удаления их из пакетов, которые в них не нуждаются. Это должно быть рассмотрено для каждого образа.

Устранение:
Вы должны разрешить разрешения setuid и setgid только для исполняемых файлов, которые требуют их наличия их. Вы можете удалить эти разрешения во время сборки, добавив следующую команду в свой Dockerfile, предпочтительно в конец Dockerfile:

```
RUN find / -perm /6000 -type f -exec chmod a-s {} \; || true
```

```
×  docker-4.9: Use COPY instead of ADD in Dockerfile (6 failed)
     ×  Command: `docker history --no-trunc
sha256:384de598d957b2d3fc3c0cf4aa1f7660d63ffe07fd771523dc6599e4f7d02daf| grep
'ADD'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 days ago          /bin/s...
116MB     \n"

     (compared using ==)

     Diff:
     @@ -1 +1,2 @@
     +<missing>
3 days ago          /bin/sh -c ...

     ×  Command: `docker history --no-trunc
sha256:3a01fb98dddd0dc2955a19bed37a379349e7cf4cea51dd0f7bc8ef94611f5be1| grep
'ADD'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 days ago          /bin/s...
116MB     \n"

     (compared using ==)

     Diff:
     @@ -1 +1,2 @@
     +<missing>
3 days ago          /bin/sh -...

     ×  Command: `docker history --no-trunc
sha256:a95645c4a87caf92f3b816aac47679d9fe137f4f6f1fe4a1164ec124178b6e1f| grep
'ADD'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 days ago          /bin/s...
116MB     \n"

     (compared using ==)

     Diff:
     @@ -1 +1,2 @@
     +<missing>
3 days ago          /bin/sh -c ...

     ×  Command: `docker history --no-trunc
sha256:1ce25a82f60076186033f4f7e85c396ddeadf439128ca91f52c83694bc5283cc| grep
'ADD'` stdout is expected to eq ""

     expected: ""
          got: "<missing>
3 days ago          /bin/s...
116MB     \n"

     (compared using ==)

     Diff:
     @@ -1 +1,2 @@
     +<missing>
3 days ago          /bin/sh -c ...
```

```
    ×  Command: `docker history --no-trunc
sha256:e480a148f48408d740a01c2c59585428d294e1f1c0ddabc89cd2335ce86a9bea| grep
'ADD'` stdout is expected to eq ""

    expected: ""
         got: "<missing>
3 days ago          /bin/s...

    (compared using ==)

    Diff:
    @@ -1 +1,2 @@
    +<missing>
3 days ago          /bin/sh -c ...

    ×  Command: `docker history --no-trunc
sha256:2e0fee02aedc429050e0d898f0ffdb68ea0ed9dc3cd9b1623e293375a19e517b| grep
'ADD'` stdout is expected to eq ""

    expected: ""
         got: "<missing>
3 days ago    /bin/sh -c #...

    (compared using ==)

    Diff:
    @@ -1 +1,2 @@
    +<missing> 3 days ago    /bin/sh -c …
```

docker-4.9: Убедитесь, что COPY используется вместо ADD в файлах Dockerfiles (Вручную)

Обоснование: Инструкция COPY просто копирует файлы с локальной хост-машины в файловую систему контейнера систему контейнера. Инструкция ADD потенциально может получать файлы с удаленных URL и выполнять такие операции, как их распаковка. Поэтому инструкция ADD создает риски для безопасности. Например, вредоносные файлы могут быть получены напрямую с URL-адресов без сканирования, или могут существовать уязвимости, связанные с их распаковкой.

Устранение: В файлах Dockerfiles следует использовать инструкции COPY, а не ADD.

```
  ↺  docker-4.10: Do not store secrets in Dockerfiles
     ↺  Manually verify that you have not used secrets in images
```

docker-4.10: Убедитесь, что секреты не хранятся в файлах Dockerfiles (вручную)

Обоснование:
Образы Docker не являются непрозрачными и содержат информацию о командах, использованных для их создания. Поэтому секреты не следует включать в Docker-файлы, используемые для сборки образов. поскольку они будут видны любому пользователю образа.

Устранение:
Не храните секреты в файлах Dockerfiles. Если секреты требуются в процессе в процессе сборки, используйте инструмент для управления секретами, например, buildkit builder входящий в состав Docker.

↺ docker-4.11: Install verified packages only
↺ Manually verify that you installed verified packages

docker-4.11: Убедитесь, что установлены только проверенные пакеты (Вручную)

Обоснование:
Проверка подлинности пакетов программного обеспечения необходима для создания безопасного образа контейнера изображение. Пакеты, происхождение которых неизвестно, потенциально могут быть вредоносными или иметь уязвимости, которые могут быть использованы.

Устранение:
Вы должны использовать выбранный вами механизм безопасного распространения пакетов для обеспечения подлинности пакетов программного обеспечения.

✔ docker-5.1: Verify AppArmor Profile, if applicable
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["AppArmorProfile"] is expected to include "docker-default"
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["AppArmorProfile"] is expected not to eq nil

docker-5.1: Убедитесь, что, если применимо, профиль AppArmor включен (вручную).

Обоснование:
AppArmor защищает ОС Linux и приложения от различных угроз путем применения политики безопасности, которая также известна как профиль AppArmor. Вы можете создать свой собственный профиль AppArmor для контейнеров или использовать профиль по умолчанию Docker. Включение этой функции обеспечивает применение политик безопасности для контейнеров, определенных в профиле

Устранение:
Если AppArmor применим к вашей ОС Linux, вам следует включить его.
1. Убедитесь, что AppArmor установлен.
2. Создайте или импортируйте профиль AppArmor для контейнеров Docker.
3. Включите применение политики.
4. Запустите контейнер Docker, используя настроенный профиль AppArmor. Для например:

```
docker run --interactive --tty --security-opt="apparmor:PROFILENAME" ubuntu /bin/bash
```

В качестве альтернативы можно использовать политику AppArmor по умолчанию в Docker.

↺ docker-5.2: Verify SELinux security options, if applicable
↺ Skipped control due to only_if condition.

docker-5.2: Убедитесь, что, если применимо, параметры безопасности SELinux установлены (Вручную)

Обоснование:
SELinux предоставляет систему обязательного контроля доступа (MAC), которая значительно расширяет стандартную модель дискреционного контроля доступа (DAC). Поэтому вы можете добавить дополнительный уровень безопасности для ваших контейнеров, включив SELinux на вашем Linux-хосте.

Устранение:
Если SELinux применим для вашей ОС Linux, вы должны использовать его.
1. Установите состояние SELinux.
2. Установите политику SELinux.
3. Создайте или импортируйте шаблон политики SELinux для контейнеров Docker.
4. Запустите Docker в режиме демона с включенным SELinux. Например:

```
docker daemon —selinux-enabled
```

или добавив следующее в конфигурационный файл daemon.json:

```
{
 "selinux-enabled": true
}
```

5. Запустите свой контейнер Docker, используя параметры безопасности. Например:

```
docker run --interactive --tty --security-opt label=level:TopSecret centos
/bin/bash
```

```
  ×  docker-5.3: Restrict Linux Kernel Capabilities within containers (3 failed)
     ×  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "CapDrop"] is expected not to eq nil

     expected: value != nil
          got: nil

     (compared using ==)

     ×  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "CapAdd"] is expected to eq "NET_ADMIN,SYS_ADMIN"

     expected: "NET_ADMIN,SYS_ADMIN"
          got: nil

     (compared using ==)
```

docker-5.4: Убедитесь, что возможности ядра Linux ограничены внутри контейнеров (Вручную)

Обоснование:
Docker поддерживает добавление и удаление возможностей. Вы должны удалить все возможности, не требующиеся для правильного функционирования контейнера.
В частности, из набора возможностей по умолчанию, предоставляемых Docker, следует удалить возможность NET_RAW, если она явно не требуется, поскольку она может дать злоумышленнику, имеющему доступ к контейнеру, возможность создавать поддельный сетевой трафик.

Устранение:
Вам следует выполнить приведенную ниже команду, чтобы добавить необходимые возможности:

```
docker run --cap-add={"Capability 1","Capability 2"} <Run arguments>
<Continer image Name or ID> <Command>
```

Для удаления ненужных возможностей следует выполнить приведенную ниже команду:

```
docker run --cap-drop={"Capability 1","Capability 2"} <Run arguments>
<Continer image Name or ID> <Command>
```

В качестве альтернативы, вы можете удалить все настроенные в данный момент возможности, а затем восстановить только те, которые вы конкретно используете:

```
docker run --cap-drop=all --cap-add={"Capability 1","Capability 2"} <Run
arguments>  <Continer image Name or ID> <Command>
```

Обратите внимание, что некоторые параметры также могут быть настроены с помощью опции --sysctl, что еще больше снижает потребность в возможностях контейнера. К ним относятся непривилегированные сокеты ICMP echo без NET_RAW и разрешение открытия любого порта менее 1024 без NET_BIND_SERVICE.
Добавление и удаление возможностей также возможно при использовании команды docker service:

```
docker service create --cap-drop=all --cap-add={"Capability 1","Capability 2"}
<Run arguments>  <Continer image Name or ID> <Command>
```

✔ docker-5.4: Do not use privileged containers
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста)... RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["HostConfig", "Privileged"] is expected not to eq true

docker-5.4: Убедитесь, что привилегированные контейнеры не используются

Обоснование:
Флаг --privileged предоставляет все возможности контейнеру, к которому он применяется, а также снимает все ограничения, наложенные контроллером cgroup устройства. Как следствие, контейнер имеет большинство прав базового хоста. Этот флаг существует только для определенных случаев использования (например, запуск Docker внутри Docker) и обычно не должен использоваться.

Устранение:
Не следует запускать контейнеры с флагом --privileged.

✔ docker-5.6: Do not run ssh within containers
✔ Command: `docker exec df79ee59ba498b020c207775f900b76ac0531a84dc164948c7dd3a8af6bcb387 ps -e` stdout is expected not to match /ssh/

docker-5.6: Убедитесь, что sshd не запускается внутри контейнеров (вручную)

Обоснование:
Запуск SSH в контейнере повышает сложность управления безопасностью, делая это:
- Сложность управления политиками доступа и соответствия требованиям безопасности для сервера SSH
- Сложность управления ключами и паролями в различных контейнерах
- Сложность управления обновлениями безопасности для SSH-сервера.
Можно иметь доступ к контейнеру без использования SSH, однако следует избегать излишнего усложнения управления безопасностью.

Устранение:
Удалите демон SSH из контейнера и используйте docker exec для входа в контейнер на удаленном хосте.

```
docker exec --interactive --tty <CONTAINER ID> sh
```

OR

```
docker attach <CONTAINER ID>
```

✔ docker-5.7: Do not map privileged ports within containers
✔ false is expected to eq false
✔ false is expected to eq false

docker-5.7: Убедитесь, что привилегированные порты не сопоставлены внутри контейнеров

Обоснование:
По умолчанию, если пользователь специально не объявляет сопоставление порта контейнера с портом хоста, Docker автоматически и правильно сопоставляет порт контейнера с одним из доступных в диапазоне 49153-65535 на хосте. Однако Docker позволяет сопоставить порт контейнера с привилегированным портом на хосте, если пользователь явно объявит об этом. Это происходит потому, что контейнеры выполняются с возможностью ядра Linux NET_BIND_SERVICE, которая не ограничивает сопоставление привилегированных портов. Привилегированные порты принимают и передают различные части данных, которые чувствительны к безопасности, и разрешение контейнерам использовать их не соответствует хорошей практике безопасности.

Устранение:
При запуске контейнера не следует сопоставлять порты контейнера с привилегированными портами хоста. Также следует убедиться, что в Dockerfile нет объявлений о сопоставлении привилегированных портов контейнера и хоста.

✔ docker-5.9: Do not share the host's network namespace
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["HostConfig", "Memory"] is expected not to eq 0

```
expected: value != 0
     got: 0

(compared using ==)
```

docker-5.9: Убедитесь, что сетевое пространство имен хоста не является общим

Обоснование:
Выбор этой опции потенциально опасен. Он позволяет процессу контейнера открывать зарезервированные порты с низким номером так, как это может делать любой другой корневой процесс. Он также позволяет контейнеру получать доступ к сетевым службам, таким как D-bus на хосте Docker. Потенциально процесс контейнера может выполнять нежелательные действия, например, выключать хост Docker. Эту опцию не следует использовать, если нет очень конкретной причины для ее включения.

Устранение:
Вы не должны передавать параметр --net=host при запуске любого контейнера.

```
×  docker-5.11: Set container CPU priority appropriately (1 failed)
     ×  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "CpuShares"] is expected not to eq 0

     expected: value != 0
          got: 0

     (compared using ==)

     ✔  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "ReadonlyRootfs"] is expected to eq true

     expected: true
          got: false

     (compared using ==)

     Diff:
     @@ -1 +1 @@
     -true
     +false
```

docker-5.11:Убедитесь, что приоритет ЦП установлен надлежащим образом для контейнеров

Обоснование:
По умолчанию процессорное время делится между контейнерами поровну. Если вы хотите контролировать доступные ресурсы ЦП между экземплярами контейнеров, вы можете использовать функцию разделения ЦП. Совместное использование ЦП позволяет установить приоритет одного контейнера над другими и не дает контейнерам с более низким приоритетом
контейнерам поглощать ресурсы ЦП, которые могут потребоваться другим процессам. Это гарантирует, что контейнеры с высоким приоритетом смогут претендовать на необходимое им время работы ЦП.

Устранение:
Вы должны управлять временем работы процессора между контейнерами в зависимости от их приоритета в вашей организации. Для этого запустите контейнер, используя аргумент --cpu-shares.
Например, вы можете запустить контейнер следующим образом:

```
docker run --interactive --tty --cpu-shares 512 centos /bin/bash
```

В приведенном выше примере контейнер запускается с долей ЦП, составляющей 50% от того, что используют другие контейнеры. Таким образом, если другой контейнер имеет долю ЦП 80%, этот контейнер будет иметь долю ЦП 40%.
Каждый новый контейнер по умолчанию будет иметь 1024 доли ЦП.

✔ docker-5.13: Bind incoming container traffic to a specific host interface
  ✔ false is expected not to eq "0.0.0.0"
  ✔ false is expected not to eq "0.0.0.0"

docker-5.13: Убедитесь, что входящий контейнерный трафик привязан к определенному интерфейсу хоста

Обоснование:
Если на хост-машине имеется несколько сетевых интерфейсов, контейнер может принимать подключения к открытым портам на любом сетевом интерфейсе. Это может быть нежелательно и небезопасно. Во многих случаях определенный, нужный интерфейс выставляется наружу, и такие службы, как обнаружение вторжений, предотвращение вторжений, брандмауэр, балансировка нагрузки и т.д., намеренно запускаются там для проверки входящего публичного трафика. Поэтому вы не должны принимать входящие соединения на любом случайном интерфейсе, а только на том, который предназначен для данного типа трафика.

Устранение:
Вы должны привязать порт контейнера к определенному интерфейсу хоста на нужном порту хоста.
Например:

```
docker run --detach --publish 10.2.3.4:49153:80 nginx
```

В приведенном примере порт контейнера 80 привязан к порту хоста на 49153 и будет принимать входящее соединение только с внешнего интерфейса 10.2.3.4.

✔ docker-5.14: Set the 'on-failure' container restart policy to 5
  ✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["HostConfig", "RestartPolicy", "Name"] is expected to eq "no"

docker-5.14: Убедитесь, что политика перезапуска контейнеров 'при сбое' установлена на '5'

Обоснование:
Если бесконечно продолжать попытки запуска контейнера, это может привести к отказу в обслуживании на хосте. Это может стать простым способом распределенной атаки типа "отказ в обслуживании", особенно если у вас много контейнеров на одном хосте. Кроме того, игнорирование статуса выхода контейнера и постоянная попытка перезапустить контейнер приводит к тому, что не исследуется основная причина завершения работы контейнеров. Если контейнер завершается, необходимо выяснить причину этого, а не просто бесконечно пытаться его перезапустить. Вы должны использовать политику перезапуска при неудаче, чтобы ограничить количество перезапусков контейнера максимум 5 попытками.

Устранение:
Если вы хотите, чтобы контейнер автоматически перезапускался, пример команды приведен ниже:

```
docker run --detach --restart=on-failure:5 nginx
```

✔ docker-5.15: Do not share the host's process namespace
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["HostConfig", "IpcMode"] is expected not to eq "host"

docker-5.15:Убедитесь, что пространство имен процессов хоста не является общим

Обоснование:
Пространство имен PID обеспечивает разделение между процессами. Оно предотвращает видимость системных процессов и позволяет повторно использовать идентификаторы процессов, включая PID 1. Если пространство имен PID хоста будет совместно использоваться с контейнерами, это позволит им видеть все процессы на хост-системе. Это уменьшает преимущество изоляции на уровне процессов между хостом и контейнерами. В таких условиях злоумышленник, имеющий доступ к контейнеру, может получить доступ к процессам на самом хосте, манипулировать ими и даже убить их. Это может привести к отключению самого хоста, что может быть очень серьезным, особенно в многопользовательской среде. Не следует разделять пространство имен процессов хоста с запущенными на нем контейнерами.

Устранение:
Не следует запускать контейнер с аргументом --pid=host.
Например, не запускайте контейнер с помощью приведенной ниже команды:

```
docker run --interactive --tty --pid=host centos /bin/bash
```

✔ docker-5.17: Do not directly expose host devices to containers
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["HostConfig", "Devices"] is expected to be empty

docker-5.17: Убедитесь, что хост-устройства не подвергаются прямому воздействию контейнеров

Обоснование:
Опция --device раскрывает устройства хоста для контейнеров, в результате чего контейнеры могут напрямую обращаться к этим устройствам. Контейнеру не нужно будет работать в привилегированном режиме для доступа и манипулирования ими, поскольку по умолчанию контейнеру предоставляется такой доступ. Кроме того, контейнеры могут удалять блочные устройства с хоста. Поэтому не следует напрямую предоставлять контейнерам доступ к устройствам хоста. Если по какой-то причине вы хотите открыть хост-устройство для контейнера, вам следует рассмотреть, какие разрешения общего доступа вы хотите использовать в каждом конкретном случае в соответствии с требованиями вашей организации:
r - только чтение
w - возможность записи
m - разрешен mknod

Устранение:
Не следует напрямую подключать хост-устройства к контейнерам. Если вам необходимо открыть хост-устройства для контейнеров, вы должны использовать гранулированные разрешения, соответствующие вашей организации:
Например, не запускайте контейнер с помощью приведенной ниже команды:

```
docker run --interactive --tty --device=/dev/tty0:/dev/tty0:rwm --device=/dev/temp_sda:/dev/temp_sda:rwm centos bash
```

Вы должны предоставлять общий доступ к главному устройству только с использованием соответствующих разрешений:

```
docker run --interactive --tty --device=/dev/tty0:/dev/tty0:rw --device=/dev/temp_sda:/dev/temp_sda:r centos bash
```

✔ docker-5.18: Override default ulimit at runtime only if needed
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> ["HostConfig", "Ulimits"] is expected to eq nil

docker-5.18: Убедитесь, что значение ulimit по умолчанию перезаписывается во время выполнения, если это необходимо

Обоснование:
ulimit обеспечивает контроль над ресурсами, доступными оболочке и запущенным ею процессам. Разумная установка ограничений на системные ресурсы защищает от условий отказа в обслуживании. В некоторых случаях легитимные пользователи и процессы могут случайно перерасходовать системные ресурсы, что приведет к ухудшению работы системы или даже к отсутствию реакции.
Следует соблюдать стандартные ulimit, установленные на уровне демона Docker. Если настройки ulimit по умолчанию не подходят для конкретного экземпляра контейнера, вы можете отменить их в виде исключения, но это не следует делать регулярно. Если многие из ваших экземпляров контейнеров превышают настройки ulimit, вам следует изменить настройки по умолчанию на более подходящие для ваших нужд.

Устранение:
Вы должны отменять настройки ulimit по умолчанию только в том случае, если это необходимо в конкретном случае.
Например, чтобы отменить стандартные настройки ulimit, запустите контейнер следующим образом:

```
docker run --ulimit nofile=1024:1024 --interactive --tty centos /bin/bash
```

✔ docker-5.19: Do not set mount propagation mode to shared
✔ is expected not to eq "shared"

docker-5.19: Убедитесь, что режим распространения монтирования не установлен на общий

Обоснование:
Общее монтирование реплицируется на все монтирования, и изменения, сделанные в любой точке монтирования, распространяются на все остальные точки монтирования. Монтирование тома в режиме общего доступа не ограничивает любой другой контейнер от монтирования и внесения изменений в этот том.
Поскольку с точки зрения безопасности это, скорее всего, нежелательный вариант, эту функцию не следует использовать, если она явно не требуется.

Устранение:
Не монтируйте тома в режиме распространения с общим доступом.
Например, не запускайте контейнер, как показано ниже:

```
docker run <Run arguments> --volume=/hostPath:/containerPath:shared <Container Image Name or ID> <Command>
```

✔ docker-5.20: Do not share the host's UTS namespace
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "UTSMode"] is expected not to eq "host"

!!! Необычно !!!

docker-5.20: Убедитесь, что пространство имен UTS хоста не является общим

Обоснование:
Совместное использование пространства имен UTS с хостом обеспечивает полное разрешение для каждого контейнера на изменять имя узла. Это не соответствует хорошей практике безопасности и не должно не должно быть разрешено.

Устранение:
Не следует запускать контейнер с аргументом --uts=host.
Например, не запускайте контейнер с помощью приведенной ниже команды:

```
docker run --rm --interactive --tty --uts=host rhel7.2
```

× docker-5.21: Do not disable default seccomp profile (2 failed)
× #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "SecurityOpt"] is expected to include /seccomp/
expected nil to include /seccomp/, but it does not respond to `include?`
× #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "SecurityOpt"] is expected not to include
/seccomp[=|:]unconfined/
expected nil not to include /seccomp[=|:]unconfined/, but it does not
respond to `include?`

docker-5.21: Убедитесь, что профиль seccomp по умолчанию не отключен.

Обоснование:
Каждому пользовательскому процессу доступно большое количество системных вызовов, многие из которых остаются неиспользованными в течение всего времени жизни процесса. Большинство приложений не нуждаются во всех этих системных вызовах и поэтому выиграют от уменьшения набора доступных системных вызовов. Сокращение набора системных вызовов уменьшает общую площадь ядра, открытую для приложения, и таким образом повышает безопасность приложения.

Устранение:
По умолчанию профили seccomp включены. Вам не нужно ничего делать, если вы не хотите изменять и использовать модифицированный профиль seccomp.

✔ docker-5.22: Do not docker exec commands with privileged option
✔ is expected to be empty

docker-5.22: Убедитесь, что команды docker exec не используются с привилегированной опцией

Обоснование:
Использование опции --privileged в командах docker exec дает команде расширенные возможности Linux. Это потенциально может быть небезопасной практикой, особенно когда вы запускаете контейнеры с ограниченными возможностями или с расширенными ограничениями.

Устранение:
Вы не должны использовать опцию --privileged в командах docker exec.

✔ docker-5.23: Do not docker exec commands with user option
✔ is expected to be empty

docker-5.23: Убедитесь, что команды docker exec не используются с параметром user=root

Обоснование:
Использование опции --user=root в команде docker exec выполняет ее внутри контейнера от имени пользователя root. Это может быть потенциально небезопасно, особенно когда вы запускаете контейнеры с ограниченными возможностями или расширенными ограничениями.
Например, если ваш контейнер работает от имени пользователя tomcat (или любого другого пользователя, не являющегося root), можно выполнить команду через docker exec от имени root с опцией --user=root. Это может быть потенциально опасно.

Устранение:
Вы не должны использовать опцию --user=root в командах docker exec.

✔ docker-5.24: Confirm cgroup usage
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "CgroupParent"] is expected to be empty

docker-5.24: Убедитесь, что использование cgroup подтверждено

Обоснование:
Системные администраторы обычно определяют cgroups, в которых должны запускаться контейнеры. Если системный администратор явно не определил cgroups, контейнеры по умолчанию запускаются в docker cgroup.
Во время выполнения можно прикрепить контейнер к другой cgroup, отличной от первоначально определенной. Это использование должно контролироваться и подтверждаться, так как при присоединении к другой cgroup контейнеру могут быть предоставлены избыточные разрешения и ресурсы, что может представлять угрозу безопасности.

Устранение:
Вы не должны использовать опцию --cgroup-parent в команде docker run без крайней необходимости.

× docker-5.25: Restrict container from acquiring additional privileges
× #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
["HostConfig", "SecurityOpt"] is expected to include /no-new-privileges/
expected nil to include /no-new-privileges/, but it does not respond to
`include?`

docker-5.25: Убедитесь, что контейнер ограничен в получении дополнительных привилегий

Обоснование:
Процесс может установить в ядре бит no_new_priv, который сохраняется при форках, клонировании и execve. Бит no_new_priv гарантирует, что процесс и его дочерние процессы не получат никаких дополнительных привилегий через биты suid или sgid. Это уменьшает опасность, связанную со многими операциями, поскольку снижается вероятность подрыва привилегированных двоичных файлов.

Устранение:
Вы должны запустить свой контейнер с помощью приведенного ниже варианта:

```
docker run --rm -it --security-opt=no-new-privileges ubuntu bash
```

```
  ×  docker-5.26: Check container health at runtime
     ×  #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array
["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0
State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-
01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false
Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">>
State.Health.Status
     undefined method `Status' for nil:NilClass
```

docker-5.26: Обеспечить проверку работоспособности контейнера во время исполнения

Обоснование:
Если используемый вами образ контейнера не имеет предопределенной инструкции HEALTHCHECK, используйте параметр --health-cmd для проверки здоровья контейнера во время выполнения.
На основе полученного состояния здоровья при необходимости можно предпринять корректирующие действия.

Устранение:
Вам следует запустить контейнер, используя параметр --health-cmd.
Например:

```
docker run -d --health-cmd='stat /etc/passwd || exit 1' nginx
```

```
  ↺  docker-5.27: Ensure docker commands always get the latest version of the
image
     ↺  Ensure docker commands always get the latest version of the image
```

docker-5.27: Убедитесь, что команды Docker всегда используют последнюю версию своего образа

Обоснование:
Известно, что многие команды Docker, такие как docker pull, docker run и т.д., имеют проблему, когда по умолчанию они извлекают локальную копию образа, если таковая имеется, даже если в репозитории upstream есть обновленная версия образа с тем же тегом. Это может привести к использованию старых образов, содержащих известные уязвимости.

Устранение:
Вы должны использовать надлежащие механизмы привязки версий (тег "latest", который назначается по умолчанию, все еще уязвим для атак кэширования), чтобы избежать извлечения кэшированных старых версий.
Механизмы привязки версий должны использоваться для базовых образов, пакетов и целых образов.
Вы можете настроить правила привязки версий в соответствии с вашими требованиями

✔ docker-5.28: Use PIDs cgroup limit
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> HostConfig.PidsLimit is expected not to cmp == 0
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> HostConfig.PidsLimit is expected not to cmp == -1

!!! необычно !!!

docker-5.28: Убедитесь, что используется лимит группы PIDs cgroup

Обоснование:
Злоумышленники могут запустить fork бомбу с помощью одной команды внутри контейнера. Эта fork бомба может привести к краху всей системы и потребует перезагрузки узла, чтобы система снова стала работоспособной. Использование параметра PIDs cgroup --pids-limit предотвратит такую атаку, ограничив количество форков, которые могут произойти внутри контейнера за определенный промежуток времени.

Устранение:
Используйте флаг --pids-limit с соответствующим значением при запуске контейнера.
Например:

```
docker run -it --pids-limit 100 <Image_ID>
```

↺ docker-5.29: Do not use Docker's default bridge docker0
↺ Not implemented yet

docker-5.29: Убедитесь, что мост Docker по умолчанию "docker0" не используется

Обоснование:
Docker подключает виртуальные интерфейсы, созданные в режиме моста, к общему мосту под названием docker0. Эта сетевая модель по умолчанию уязвима для атак ARP spoofing и MAC flooding, поскольку к ней не применяется фильтрация.

Устранение:
Необходимо следовать документации Docker и создать пользовательскую сеть. Все контейнеры должны быть запущены в этой сети.

✔ docker-5.30: Do not share the host's user namespaces
✔ #<Hashie::Mash AppArmorProfile="docker-default" Args=#<Hashie::Array ["run", "--host=0.0.0.0"]> ..(много текста).. RestartCount=0 State=#<Hashie::Mash Dead=false Error="" ExitCode=0 FinishedAt="0001-01-01T00:00:00Z" OOMKilled=false Paused=false Pid=10345 Restarting=false Running=true StartedAt="2023-07-07T17:48:57.19486946Z" Status="running">> HostConfig.UsernsMode is expected to eq ""

docker-5.30: Убедитесь, что пользовательские пространства имен хоста не являются общими

Обоснование:
Пространства имен пользователей гарантируют, что корневой процесс внутри контейнера будет сопоставлен с некорневым процессом вне контейнера. Поэтому совместное использование пространств имен пользователей хоста и контейнера не изолирует пользователей на хосте от пользователей в контейнерах.

Устранение:
Не следует делить пространства имен пользователей между хостом и контейнерами. Например, не следует выполнять приведенную ниже команду:

```
docker run --rm -it --userns=host ubuntu bash
```

↺ docker-2.1: Restrict network traffic between containers
↺ Can't find file: /etc/docker/daemon.json

docker-2.1: Обеспечьте ограничение сетевого трафика между контейнерами на мосту по умолчанию

Обоснование:
По умолчанию неограниченный сетевой трафик включен между всеми контейнерами на одном хосте на сетевом мосту по умолчанию. Таким образом, каждый контейнер имеет возможность читать все пакеты через контейнерную сеть на одном хосте. Это может привести к непреднамеренному и нежелательному раскрытию информации другим контейнерам. Следовательно, ограничьте межконтейнерное взаимодействие на сетевом мосту по умолчанию.

Устранение:
Отредактируйте конфигурационный файл демона Docker, чтобы убедиться, что icc отключен. Он должен включать следующий параметр

```
"icc": false
```

В качестве альтернативы запустите демон docker напрямую и передайте в качестве аргумента --icc=false.
Например:

```
dockerd --icc=false
```

В качестве альтернативы можно следовать документации Docker и создать пользовательскую сеть и присоединять только те контейнеры, которые должны взаимодействовать с этой пользовательской сетью. Параметр --icc применяется только к мосту docker по умолчанию, если используются пользовательские сети, то вместо него следует использовать подход сегментирования сетей.

↺ docker-2.2: Set the logging level
↺ Can't find file: /etc/docker/daemon.json

docker-2.2: Убедитесь, что уровень ведения журнала установлен на 'info'

Обоснование:
Установка соответствующего уровня журнала настраивает демон Docker на регистрацию событий, которые вы захотите просмотреть позже. Базовый уровень журнала info и выше будет фиксировать все журналы, кроме отладочных. Пока нет необходимости, не следует запускать демон Docker на уровне журнала отладки.

Устранение:
Убедитесь, что в конфигурационный файл демона Docker включена следующая конфигурация

```
"log-level": "info"
```

В качестве альтернативы запустите демон Docker, как показано ниже:

```
dockerd --log-level="info"
```

↺ docker-2.3: Allow Docker to make changes to iptables
↺ Can't find file: /etc/docker/daemon.json

docker-2.3: Убедитесь, что Docker разрешено вносить изменения в iptables

Обоснование:
Docker никогда не будет вносить изменения в правила iptables вашей системы, если вы не разрешите ему это сделать.
Если вы разрешите это, сервер Docker автоматически внесет все необходимые изменения. Мы рекомендуем разрешить Docker вносить изменения в iptables автоматически, чтобы избежать неправильной конфигурации сети, которая может повлиять на связь между контейнерами и с внешним миром. Кроме того, это снижает административные затраты на обновление iptables при каждом добавлении контейнеров или изменении сетевых параметров.

Устранение:
Не запускайте демон Docker с параметром --iptables=false. Например, не запускайте демон Docker, как показано ниже:

```
dockerd --iptables=false
```

↺ docker-2.4: Do not use insecure registries
↺ Can't find file: /etc/docker/daemon.json

docker-2.4: Убедитесь, что не используются небезопасные реестры

Обоснование:
Безопасный реестр использует TLS. Копия сертификата ЦС реестра размещается на хосте Docker в каталоге /etc/docker/certs.d/<имя реестра>/. Небезопасный реестр - это тот, у которого нет действительного сертификата реестра, или тот, который не использует TLS. Небезопасные реестры не следует использовать, поскольку они представляют риск перехвата и модификации трафика.
Кроме того, если реестр помечен как небезопасный, такие команды, как docker pull, docker push и docker search, не будут выдавать сообщения об ошибке, и пользователи могут неопределенное время работать с таким типом небезопасного реестра, не получая уведомления о риске потенциальной компрометации.

Устранение:
Вы должны убедиться, что не используются небезопасные реестры.

↺ docker-2.5: Do not use the aufs storage driver
↺ Can't find file: /etc/docker/daemon.json

!!! необычно !!!

docker-2.5: Убедитесь, что драйвер хранения aufs не используется

Обоснование:
Драйвер хранения данных aufs - это самый старый драйвер хранения данных, используемый в системах Linux. Он основан на наборе исправлений ядра Linux, который вряд ли в будущем будет объединен с основным ядром ОС. Драйвер aufs также известен тем, что вызывает некоторые серьезные сбои ядра. aufs имеет унаследованную поддержку только в системах, использующих Docker. Самое главное, что aufs не поддерживается во многих дистрибутивах Linux, использующих последние версии ядра Linux.

Устранение:
Не используйте явным образом aufs в качестве драйвера хранилища.
Например, не запускайте демон Docker, как показано ниже:

```
dockerd --storage-driver aufs
```

↺ docker-2.6: Configure TLS authentication for Docker daemon
↺ Can't find file: /etc/docker/daemon.json

docker-2.6: Убедитесь, что аутентификация TLS для демона Docker настроена

Обоснование:
По умолчанию демон Docker привязывается к несетевому сокету Unix и запускается с привилегиями root. Если вы измените стандартную привязку демона Docker к порту TCP или любому другому сокету Unix, любой, кто имеет доступ к этому порту или сокету, может получить полный доступ к демону Docker и, следовательно, к хост-системе. По этой причине не следует привязывать демон Docker к другому IP/порту или сокету Unix.
Если вам необходимо открыть демон Docker через сетевой сокет, вам следует настроить TLS-аутентификацию для демона и для всех API Docker Swarm (если они используются). Этот тип конфигурации ограничивает соединения с демоном Docker по сети ограниченным числом клиентов, имеющих доступ к учетным данным клиента TLS.

Устранение:
Выполните действия, указанные в документации Docker или других справочных материалах.

↺ docker-2.7: Set default ulimit as appropriate
↺ Can't find file: /etc/docker/daemon.json

docker-2.7: Убедитесь, что значение ulimit по умолчанию настроено должным образом

Обоснование:
ulimit обеспечивает контроль над ресурсами, доступными оболочке и процессам, которые она запускает. Разумная установка ограничений на системные ресурсы может спасти вас от таких катастроф, как бомба с вилкой. Иногда даже доброжелательные пользователи и легитимные процессы могут чрезмерно использовать системные ресурсы и сделать систему непригодной для использования.
Установка по умолчанию ulimit для демона Docker влечет за собой применение ulimit для всех экземпляров контейнеров. В этом случае вам не нужно будет устанавливать ulimit для каждого экземпляра контейнера.
Однако при необходимости ulimit по умолчанию может быть отменен во время выполнения контейнера.
Поэтому, чтобы иметь надлежащий контроль над системными ресурсами, определите ulimit по умолчанию, как это необходимо в вашей среде.

Устранение:
Запустите Docker в режиме демона и передайте параметр --default-ulimit в качестве аргумента с соответствующими ulimits, которые подходят для вашей среды и соответствуют вашей политике безопасности.
Для примера

```
dockerd --default-ulimit nproc=1024:2048 --default-ulimit nofile=100:200
```

↺ docker-2.8: Enable user namespace support (1 skipped)
↺ Can't find file: /etc/docker/daemon.json
✔ File /etc/subuid is expected to exist
✔ File /etc/subuid is expected to be file
✔ File /etc/subgid is expected to exist
✔ File /etc/subgid is expected to be file

docker-2.8: Включите поддержку пространства имен пользователей

Обоснование:
Поддержка "пространства имен пользователей" ядра Linux в демоне Docker обеспечивает дополнительную безопасность для хост-системы Docker. Она позволяет контейнеру иметь уникальный диапазон идентификаторов пользователей и групп, которые находятся за пределами традиционного диапазона пользователей и групп, используемых хост-системой.
Например, пользователь root может иметь ожидаемые административные привилегии внутри контейнера, но в хост-системе он может быть сопоставлен с непривилегированным UID.

Устранение:
Пожалуйста, обратитесь к документации Docker для различных способов настройки в зависимости от ваших требований. Ваши действия также могут отличаться в зависимости от платформы - например, на Red Hat создание связок sub-UIDs и sub-GIDs не работает автоматически.
Возможно, вам придется создавать собственные отображения.

Ниже приведены шаги высокого уровня:
Шаг 1: Убедитесь, что файлы /etc/subuid и /etc/subgid существуют:

```
touch /etc/subuid /etc/subgid
```

Шаг 2: Запустите демон docker с флагом --userns-remap

```
dockerd --userns-remap=default
```

↺ docker-2.9: Confirm default cgroup usage
↺ Can't find file: /etc/docker/daemon.json

docker-2.9: Убедитесь, что использование cgroup по умолчанию подтверждено

Обоснование:
Системные администраторы обычно определяют cgroups, под которыми должны запускаться контейнеры. Даже если cgroups не определены системными администраторами явно, контейнеры по умолчанию запускаются под docker cgroup. Можно прикрепить контейнеры к другой cgroup, отличной от той, которая используется по умолчанию, однако такое использование должно контролироваться и подтверждаться, поскольку прикрепление к другой cgroup, отличной от той, которая используется по умолчанию, может привести к неравномерному распределению ресурсов, что вызовет проблемы с использованием ресурсов на хосте.

Устранение:
Настройка по умолчанию соответствует хорошей практике безопасности и может быть оставлена на месте. Если вы хотите специально установить cgroup не по умолчанию, передайте параметр --cgroup-parent демону Docker при его запуске.
Например:

```
dockerd --cgroup-parent=/foobar
```

↺ docker-2.10: Do not change base device size until needed
↺ Can't find file: /etc/docker/daemon.json

docker-2.10: Убедитесь, что размер базового устройства не изменяется до тех пор, пока это не потребуется

Обоснование:
Размер базового устройства может быть увеличен при перезапуске демона. Увеличение размера базового устройства позволяет всем будущим образам и контейнерам иметь новый размер базового устройства. Пользователь может использовать эту опцию для увеличения размера базового устройства, однако уменьшение не допускается. Это значение влияет на общесистемную "базовую" пустую файловую систему, которая может быть уже инициализирована и, следовательно, унаследована извлекаемыми образами.
Хотя файловая система не выделяет увеличенный размер, пока она пуста, для дополнительных образов будет выделено больше места. Это может вызвать отказ в обслуживании, если выделенный раздел станет переполненным.

Устранение:
Не устанавливайте параметр --storage-opt dm.basesize до тех пор, пока он не понадобится.

↺ docker-2.11: Use authorization plugin
↺ Can't find file: /etc/docker/daemon.json

docker-2.11: Убедитесь, что авторизация для команд клиента Docker включена

Обоснование:
В настоящее время модель авторизации Docker "из коробки" работает по принципу "все или ничего". Это означает, что любой пользователь, имеющий разрешение на доступ к демону Docker, может выполнить любую команду клиента Docker. То же самое справедливо для удаленных пользователей, обращающихся к API Docker для связи с демоном.
Если вам требуется более строгий контроль доступа, вы можете создать плагины авторизации и добавить их в конфигурацию демона Docker. Используя плагин авторизации, администратор Docker может настроить гранулированные политики доступа для управления доступом к демону Docker.
Сторонние интеграции Docker могут реализовать свои собственные модели авторизации, требующие авторизации с демоном Docker за пределами встроенного плагина авторизации docker (например, Kubernetes, Cloud Foundry, Openshift).

Устранение:
Шаг 1: Установите/создайте плагин авторизации.
Шаг 2: Настройте политику авторизации по своему усмотрению.
Шаг 3: Запустите демон docker, как показано ниже:

```
dockerd --authorization-plugin=<PLUGIN_ID>
```

↺ docker-2.12: Configure centralized and remote logging
↺ Can't find file: /etc/docker/daemon.json

docker-2.12: Обеспечьте настройку централизованного и удаленного протоколирования

Обоснование:
Централизованное и удаленное протоколирование гарантирует сохранность всех важных записей журнала даже в случае серьезной проблемы с доступностью данных. Docker поддерживает различные методы протоколирования, и вы должны использовать тот, который лучше всего соответствует вашей политике ИТ-безопасности.

Устранение:
Шаг 1: Настройте нужный драйвер журнала, следуя документации к нему.
Шаг 2: Запустите демон docker, используя этот драйвер протоколирования.
Например:

```
dockerd --log-driver=syslog --log-opt syslog-address=tcp://192.xxx.xxx.xxx
```

↺ docker-2.13: Disable operations on legacy registry (v1)
↺ Can't find file: /etc/docker/daemon.json

docker-2.13: Отключить операции над унаследованным реестром

Это предупреждение исправляется строкой "disable-legacy-registry": true в конфигурационном файле демона. Это отключает небезопасный протокол регистрации устаревших образов. Поскольку поддержка этого протокола уже удалена из демона Docker, этот флаг находится в процессе устаревания.

↺ docker-2.14: Enable live restore
↺ Can't find file: /etc/docker/daemon.json

docker-2.14: Убедитесь, что функция восстановления в реальном времени включена

Обоснование:
Одной из важных триад безопасности является доступность. Установка флага --live-restore в демоне Docker гарантирует, что выполнение контейнера не будет прервано, когда он недоступен. Это также облегчает обновление и исправление демона Docker без простоя приложения.

Устранение:
Запустите Docker в режиме демона и передайте ему в качестве аргумента --live-restore.
Например,

```
dockerd --live-restore
```

✔ docker-2.15: Do not enable swarm mode, if not needed
✔ #<Hashie::Mash Architecture="x86_64" BridgeNfIp6tables=true BridgeNfIptables=true CPUSet=true CPUShares=true CgroupDriver="systemd" CgroupVersion="2" ..(много текста).. "name=cgroupns"]> ServerVersion="20.10.21" SwapLimit=true Swarm=#<Hashie::Mash ControlAvailable=false Error="" LocalNodeState="inactive" NodeAddr="" NodeID="" RemoteManagers=nil> SystemTime="2023-07-07T19:09:53.748521358Z" Warnings=nil> Swarm.LocalNodeState is expected to eq "inactive"

docker-2.15: Убедитесь, что режим swarm не включен, если в нем нет необходимости

Обоснование:
По умолчанию экземпляр движка Docker не прослушивает никаких сетевых портов, а все соединения с клиентом происходят через сокет Unix. Когда на экземпляре движка Docker включен режим роя, в системе открываются несколько сетевых портов, которые становятся доступны другим системам в сети для управления кластером и связи между узлами.
Открытие сетевых портов на системе увеличивает ее поверхность атаки, поэтому этого следует избегать, если нет необходимости.
Следует отметить, что режим роя необходим для работы компонентов Docker Enterprise

Устранение:
Если режим роя был включен на системе по ошибке, необходимо выполнить приведенную ниже команду:

```
docker swarm leave
```

```
  ↺  docker-2.16: Control the number of manager nodes in a swarm
     ↺  Skipped control due to only_if condition.
```

Docker-2.16: Убедитесь, что в swarm создано минимальное количество управляющих узлов

Обоснование:
Управляющие узлы в рое имеют контроль над роем и могут изменять его конфигурацию, включая изменение параметров безопасности. Наличие чрезмерного количества управляющих узлов может сделать рой более восприимчивым к компрометации.
Если отказоустойчивость управляющих узлов не требуется, то в качестве управляющего должен быть избран один узел. Если отказоустойчивость необходима, то следует настроить наименьшее нечетное число для достижения соответствующего уровня отказоустойчивости. Это всегда должно быть нечетное число, чтобы обеспечить кворум.

Устранение:
Если настроено чрезмерное количество менеджеров, лишние узлы могут быть понижены до рабочих с помощью следующей команды:

```
docker node demote <ID>
```

Где - значение ID узла менеджера, которого нужно понизить в должности.

```
  ↺  docker-2.17: Bind swarm services to a specific host interface
     ↺  Skipped control due to only_if condition.
```

docker-2.17: Убедитесь, что службы роя привязаны к определенному интерфейсу хоста

Обоснование:
При инициализации swarm значение по умолчанию для флага --listen-addr равно 0.0.0.0:2377, что означает, что службы роя будут прослушивать все интерфейсы хоста. Если узел имеет несколько сетевых интерфейсов, это может быть нежелательно, так как может привести к тому, что службы роя будут прослушивать сети, не участвующие в работе роя.
Передавая определенный IP-адрес в параметре --listen-addr, можно указать конкретный сетевой интерфейс, ограничив эту возможность.

Устранение:
Решение этой проблемы требует повторной инициализации swarm с указанием конкретного интерфейса для параметра --listen-addr.

```
  ↺  docker-2.18: Disable Userland Proxy (1 failed) (1 skipped)
     ↺  Can't find file: /etc/docker/daemon.json
     ×  ["/usr/bin/dockerd -H fd:// 
--containerd=/run/containerd/containerd.sock"] is expected to include "userland-
proxy=false"
     expected ["/usr/bin/dockerd -H fd:// 
--containerd=/run/containerd/containerd.sock"] to include "userland-proxy=false"
```

docker-2.18: Убедитесь, что все оверлейные сети swarm Docker зашифрованы

Обоснование:
По умолчанию данные, которыми обмениваются контейнеры на узлах оверлейной сети, не шифруются. Это может потенциально раскрыть трафик между контейнерами.

Устранение:
Вы должны создавать оверлейные сети с флагом --opt encrypted.

↺ docker-2.19: Encrypt data exchanged between containers on different nodes on the overlay network
↺ Skipped control due to only_if condition.

↺ docker-2.20: Apply a daemon-wide custom seccomp profile, if needed
↺ Can't find file: /etc/docker/daemon.json

docker-2.20: Убедитесь, что пользовательский профиль seccomp для всего демона применяется при необходимости

Обоснование:
Каждому пользовательскому процессу доступно большое количество системных вызовов, причем многие из них не используются в течение всего времени существования процесса. Многие приложения не нуждаются во всех этих системных вызовах и поэтому выигрывают, если каждый используемый системный вызов будет рассмотрен в соответствии с политикой безопасности организации. Сокращение набора системных вызовов уменьшает общую поверхность ядра, открытую для приложения, и, следовательно, повышает безопасность приложения.
Пользовательский профиль seccomp может быть применен вместо профиля seccomp по умолчанию в Docker.
Кроме того, если профиль по умолчанию Docker подходит для вашей среды, вы можете проигнорировать эту рекомендацию.

Устранение:
По умолчанию применяется стандартный профиль seccomp Docker. Если он подходит для вашей среды, никаких действий предпринимать не нужно. Если же вы хотите применить собственный профиль seccomp, используйте флаг --seccomp-profile при запуске демона или поместите его в файл параметров времени выполнения демона

```
dockerd --seccomp-profile </path/to/seccomp/profile>
```

✔ docker-2.21: Avoid experimental features in production
✔ false is expected to eq "false"

docker-2.21: Убедитесь, что экспериментальные функции не внедрены в производство

Обоснование:
"Experimental" в настоящее время является флагом Docker daemon во время выполнения, а не функцией отдельной сборки. Передача --experimental в качестве флага времени выполнения демону докера активирует экспериментальные функции. Несмотря на то, что "Experimental" считается стабильным релизом, он имеет ряд функций, которые, возможно, не были полностью протестированы и не гарантируют стабильность API.

Устранение:
Вы не должны передавать параметр --experimental в качестве параметра времени выполнения демону Docker на производственных системах.

↺ docker-2.22: Use Docker's secret management commands for managing secrets in a Swarm cluster
↺ Skipped control due to only_if condition.

docker-2.22: Убедитесь, что команды управления секретами Docker используются для управления секретами в swarm кластере

Обоснование:
Docker имеет различные команды для управления секретами в swarm кластере

Устранение:
Вы должны следовать документации по docker secret и использовать ее для эффективного управления секретами.

↺ docker-2.23: Run swarm manager in auto-lock mode
↺ Skipped control due to only_if condition.

Docker-2.23: Убедитесь, что менеджер swarm работает в режиме автоблокировки

Обоснование:
При перезапуске Docker ключ TLS, используемый для шифрования связи между узлами роя, и ключ, используемый для шифрования и расшифровки журналов Raft на диске, загружаются в память каждого узла менеджера. Вы можете защитить взаимный ключ шифрования TLS и ключ, используемый для шифрования и расшифровки журналов Raft в состоянии покоя. Эта защита может быть включена при инициализации swarm с флагом --autolock.
При использовании опции --autolockenabled при перезапуске Docker необходимо сначала разблокировать swarm, используя ключ шифрования, сгенерированный Docker при инициализации swarm.
Это имеет свои преимущества в среде с высоким уровнем безопасности, однако они должны быть сбалансированы с проблемами поддержки, вызванными тем, что swarm не запускается автоматически, если, например, на хосте произошел сбой.

Устранение:
Если вы инициализируете swarm, используйте приведенную ниже команду.

```
docker swarm init --autolock
```

Если вы хотите установить --autolock на существующем узле менеджера swarm, используйте следующую команду

```
docker swarm update --autolock
```

✔ docker-3.1: Verify that docker.service file ownership is set to root:root
✔ File /lib/systemd/system/docker.service is expected to exist
✔ File /lib/systemd/system/docker.service is expected to be file
✔ File /lib/systemd/system/docker.service is expected to be owned by "root"
✔ File /lib/systemd/system/docker.service is expected to be grouped into "root"

docker-3.1: Убедитесь, что право собственности на файл docker.service установлено на root:root

Обоснование:
Файл docker.service содержит важные параметры, которые могут изменить поведение демона Docker. Поэтому он должен находиться под индивидуальным и групповым владением пользователя root, чтобы гарантировать, что он не будет изменен или поврежден менее привилегированным пользователем.

Устранение:
Шаг 1: Выясните местоположение файла:

```
systemctl show -p FragmentPath docker.service
```

Шаг 2: Если файл не существует, данная рекомендация неприменима. Если файл существует, необходимо выполнить приведенную ниже команду, указав правильный путь к файлу, чтобы установить права собственности и групповые права для файла на root.
Например,

```
chown root:root /usr/lib/systemd/system/docker.service
```

✔ docker-3.2: Verify that docker.service file permissions are set to 644 or more restrictive
✔ File /lib/systemd/system/docker.service is expected to exist
✔ File /lib/systemd/system/docker.service is expected to be file
✔ File /lib/systemd/system/docker.service is expected to be readable by owner
✔ File /lib/systemd/system/docker.service is expected to be writable by owner
✔ File /lib/systemd/system/docker.service is expected to be readable by group
✔ File /lib/systemd/system/docker.service is expected not to be writable by group
✔ File /lib/systemd/system/docker.service is expected to be readable by other
✔ File /lib/systemd/system/docker.service is expected not to be writable by other
✔ File /lib/systemd/system/docker.service is expected not to be executable

docker-3.2: Убедитесь, что разрешения файла docker.service установлены должным образом

Обоснование:
Файл docker.service содержит важные параметры, которые могут изменить поведение демона Docker. Поэтому он не должен быть доступен для записи другим пользователям, кроме root, чтобы гарантировать, что он не может быть изменен менее привилегированными пользователями.

Устранение:
Шаг 1: Выясните местоположение файла:

```
systemctl show -p FragmentPath docker.service
```

Шаг 2: Если файл не существует, данная рекомендация неприменима. Если файл существует, выполните приведенную ниже команду, указав правильный путь к файлу, чтобы установить разрешения файла на 644.
Например,

```
chmod 644 /usr/lib/systemd/system/docker.service
```

✔ docker-3.3: Verify that docker.socket file ownership is set to root:root
✔ File /lib/systemd/system/docker.socket is expected to exist
✔ File /lib/systemd/system/docker.socket is expected to be file
✔ File /lib/systemd/system/docker.socket is expected to be owned by "root"
✔ File /lib/systemd/system/docker.socket is expected to be grouped into "root"

docker-3.3: Убедитесь, что право собственности на файл docker.service установлено на root:root

Обоснование:
Файл docker.service содержит важные параметры, которые могут изменить поведение демона Docker. Поэтому он должен находиться под индивидуальным и групповым владением пользователя root, чтобы гарантировать, что он не будет изменен или поврежден менее привилегированным пользователем.

Устранение:
Шаг 1: Выясните местоположение файла:

```
systemctl show -p FragmentPath docker.service
```

Шаг 2: Если файл не существует, данная рекомендация неприменима. Если файл существует, необходимо выполнить приведенную ниже команду, указав правильный путь к файлу, чтобы установить права собственности и групповые права для файла на root.
Например,

```
chown root:root /usr/lib/systemd/system/docker.service
```

```
  ✔  docker-3.4: Verify that docker.socket file permissions are set to 644 or
more restrictive
     ✔  File /lib/systemd/system/docker.socket is expected to exist
     ✔  File /lib/systemd/system/docker.socket is expected to be file
     ✔  File /lib/systemd/system/docker.socket is expected to be readable by
owner
     ✔  File /lib/systemd/system/docker.socket is expected to be writable by
owner
     ✔  File /lib/systemd/system/docker.socket is expected to be readable by
group
     ✔  File /lib/systemd/system/docker.socket is expected not to be writable
by group
     ✔  File /lib/systemd/system/docker.socket is expected to be readable by
other
     ✔  File /lib/systemd/system/docker.socket is expected not to be writable
by other
     ✔  File /lib/systemd/system/docker.socket is expected not to be executable
  ✔  docker-3.5: Verify that /etc/docker directory ownership is set to
root:root
     ✔  File /etc/docker is expected to exist
     ✔  File /etc/docker is expected to be directory
     ✔  File /etc/docker is expected to be owned by "root"
     ✔  File /etc/docker is expected to be grouped into "root"
```

docker-3.4: Убедитесь, что разрешения файла docker.service установлены должным образом

Обоснование:
Файл docker.service содержит важные параметры, которые могут изменить поведение демона Docker. Поэтому он не должен быть доступен для записи другим пользователям, кроме root, чтобы гарантировать, что он не может быть изменен менее привилегированными пользователями.

Устранение:
Шаг 1: Выясните местоположение файла:

```
systemctl show -p FragmentPath docker.service
```

Шаг 2: Если файл не существует, данная рекомендация неприменима. Если файл существует, выполните приведенную ниже команду, указав правильный путь к файлу, чтобы установить разрешения файла на 644.
Например,

```
chmod 644 /usr/lib/systemd/system/docker.service
```

✔️ docker-3.6: Verify that /etc/docker directory permissions are set to 755 or more restrictive
    ✔️ File /etc/docker is expected to exist
    ✔️ File /etc/docker is expected to be directory
    ✔️ File /etc/docker is expected to be readable by owner
    ✔️ File /etc/docker is expected to be writable by owner
    ✔️ File /etc/docker is expected to be executable by owner
    ✔️ File /etc/docker is expected to be readable by group
    ✔️ File /etc/docker is expected not to be writable by group
    ✔️ File /etc/docker is expected to be executable by group
    ✔️ File /etc/docker is expected to be readable by other
    ✔️ File /etc/docker is expected not to be writable by other
    ✔️ File /etc/docker is expected to be executable by other

docker-3.6: Убедитесь, что разрешения каталога /etc/docker установлены на 755 или более строгое значение

Обоснование:
Каталог /etc/docker содержит сертификаты и ключи в дополнение к различным конфиденциальным файлам. Поэтому он должен быть доступен для записи только root, чтобы исключить возможность его изменения менее привилегированным пользователем.

Устранение:
Вам следует выполнить следующую команду:

```
chmod 755 /etc/docker
```

```
  ×  docker-3.7: Verify that registry certificate file ownership is set to
root:root (12 failed)
     ×  File /etc/docker/certs.d is expected to exist
     expected File /etc/docker/certs.d to exist
     ×  File /etc/docker/certs.d is expected to be directory
     expected `File /etc/docker/certs.d.directory?` to be truthy, got false
     ×  File /etc/docker/certs.d is expected to be owned by "root"
     expected `File /etc/docker/certs.d.owned_by?("root")` to be truthy, got
false
     ×  File /etc/docker/certs.d is expected to be grouped into "root"
     expected `File /etc/docker/certs.d.grouped_into?("root")` to be truthy, got
false
     ×  File /etc/docker/certs.d/registry_hostname:port is expected to exist
     expected File /etc/docker/certs.d/registry_hostname:port to exist
     ×  File /etc/docker/certs.d/registry_hostname:port is expected to be
directory
     expected `File /etc/docker/certs.d/registry_hostname:port.directory?` to be
truthy, got false
     ×  File /etc/docker/certs.d/registry_hostname:port is expected to be owned
by "root"
     expected `File /etc/docker/certs.d/registry_hostname:port.owned_by?
("root")` to be truthy, got false
     ×  File /etc/docker/certs.d/registry_hostname:port is expected to be
grouped into "root"
     expected `File /etc/docker/certs.d/registry_hostname:port.grouped_into?
("root")` to be truthy, got false
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to
exist
     expected File /etc/docker/certs.d/registry_hostname:port/ca.crt to exist
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to be
file
     expected `File /etc/docker/certs.d/registry_hostname:port/ca.crt.file?` to
be truthy, got false
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to be
owned by "root"
     expected `File /etc/docker/certs.d/registry_hostname:port/ca.crt.owned_by?
("root")` to be truthy, got false
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to be
grouped into "root"
     expected `File
/etc/docker/certs.d/registry_hostname:port/ca.crt.grouped_into?("root")` to be
truthy, got false
```

docker-3.7: Убедитесь, что право собственности на файл сертификата реестра установлено на root:root

Обоснование:
Каталог /etc/docker/certs.d/<имя реестра> содержит сертификаты реестра Docker. Эти файлы сертификатов должны иметь индивидуальное и групповое владение root, чтобы менее привилегированные пользователи не могли изменять содержимое каталога.

Устранение:
Может быть выполнена следующая команда:

```
chown root:root /etc/docker/certs.d/<имя реестра>/*
```

```
  ×  docker-3.8: Verify that registry certificate file permissions are set to
444 or more restrictive (3 failed)
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to
exist
     expected File /etc/docker/certs.d/registry_hostname:port/ca.crt to exist
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to be
file
     expected `File /etc/docker/certs.d/registry_hostname:port/ca.crt.file?` to
be truthy, got false
     ×  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected to be
readable
     expected File /etc/docker/certs.d/registry_hostname:port/ca.crt to be
readable
     ✔  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected not
to be executable
     ✔  File /etc/docker/certs.d/registry_hostname:port/ca.crt is expected not
to be writable
```

docker-3.8: Убедитесь, что разрешения файла сертификата реестра установлены на 444 или более ограничительно

Обоснование:
Каталог /etc/docker/certs.d/<имя реестра> содержит сертификаты реестра Docker. Эти файлы сертификатов должны иметь разрешения 444 или более строгие разрешения для того, чтобы непривилегированные пользователи не имели к ним полного доступа.

Устранение:
Выполните следующую команду:

```
chmod 444 /etc/docker/certs.d/<имя реестра>/*
```

```
  ×  docker-3.9: Verify that TLS CA certificate file ownership is set to
root:root (4 failed)
     ×  File  is expected to exist
     expected File  to exist
     ×  File  is expected to be file
     expected `File .file?` to be truthy, got false
     ×  File  is expected to be owned by "root"
     expected `File .owned_by?("root")` to be truthy, got false
     ×  File  is expected to be grouped into "root"
     expected `File .grouped_into?("root")` to be truthy, got false
```

docker-3.9: Убедитесь, что владельцем файла сертификата TLS CA установлен root:root

Обоснование:
Файл сертификата TLS CA должен быть защищен от любого вмешательства. Он используется для аутентификации сервера Docker на основе данного сертификата CA. Поэтому он должен быть индивидуальным и групповым, принадлежащим root, чтобы гарантировать, что он не может быть изменен менее привилегированными пользователями.

Устранение:
Выполните следующую команду:

```
chown root:root <path to TLS certificate file>
```

Это устанавливает индивидуальное владение и групповое владение для файла сертификата TLS CA на root.

```
  ×  docker-3.10: Verify that TLS CA certificate file permissions are set to 444
or more restrictive (3 failed)
     ×  File  is expected to exist
     expected File  to exist
     ×  File  is expected to be file
     expected `File .file?` to be truthy, got false
     ×  File  is expected to be readable
     expected File  to be readable
     ✔  File  is expected not to be executable
     ✔  File  is expected not to be writable
```

docker-3.10: Убедитесь, что разрешения на файл сертификата ЦС TLS установлены на 444 или более ограничительно

Обоснование:
Файл сертификата TLS CA должен быть защищен от любого вмешательства. Он используется для аутентификации сервера Docker на основе данного сертификата CA. Поэтому он должен иметь разрешения 444 или более строгие разрешения, чтобы гарантировать, что файл не может быть изменен менее привилегированным пользователем.

Устранение:
Выполните следующую команду:

```
chmod 444 <path to TLS certificate file>
```

Это устанавливает права доступа к файлу TLS CA на 444.

```
  ×  docker-3.11: Verify that Docker server certificate file ownership is set to
root:root (4 failed)
     ×  File  is expected to exist
     expected File  to exist
     ×  File  is expected to be file
     expected `File .file?` to be truthy, got false
     ×  File  is expected to be owned by "root"
     expected `File .owned_by?("root")` to be truthy, got false
     ×  File  is expected to be grouped into "root"
     expected `File .grouped_into?("root")` to be truthy, got false
```

docker-3.11: Убедитесь, что владельцем файла сертификата сервера Docker установлен root:root

Обоснование:
Файл сертификата сервера Docker должен быть защищен от любого вмешательства. Он используется для аутентификации сервера Docker на основе заданного сертификата сервера. Поэтому он должен находиться в индивидуальном владении и групповом владении root для предотвращения модификации менее привилегированными пользователями.

Устранение:
Вы должны выполнить следующую команду

```
chown root:root <path to Docker server certificate file>
```

Это устанавливает индивидуальное право собственности и групповое право собственности для файла сертификата сервера Docker на root.

```
  ×  docker-3.12: Verify that Docker server certificate file permissions are set
to 444 or more restrictive (3 failed)
     ×  File  is expected to exist
     expected File  to exist
     ×  File  is expected to be file
     expected `File .file?` to be truthy, got false
     ×  File  is expected to be readable
     expected File  to be readable
     ✔  File  is expected not to be executable
     ✔  File  is expected not to be writable
```

docker-3.12: Убедитесь, что разрешения на файл сертификата сервера Docker установлены на 444 или более ограничительно

Обоснование:
Файл сертификата сервера Docker должен быть защищен от любого вмешательства. Он используется для аутентификации сервера Docker на основе данного сертификата сервера. Поэтому он должен иметь права доступа 444 для предотвращения его модификации.

Устранение:
Выполните следующую команду:

```
chmod 444 <path to Docker server certificate file>
```

Это устанавливает права доступа к файлу сертификата сервера Docker на 444.

```
  ×  docker-3.13: Verify that Docker server certificate key file ownership is
set to root:root (4 failed)
     ×  File  is expected to exist
     expected File  to exist
     ×  File  is expected to be file
     expected `File .file?` to be truthy, got false
     ×  File  is expected to be owned by "root"
     expected `File .owned_by?("root")` to be truthy, got false
     ×  File  is expected to be grouped into "root"
     expected `File .grouped_into?("root")` to be truthy, got false
```

docker-3.13: Убедитесь, что для файла ключа сертификата сервера Docker установлено значение root:root

Обоснование:
Файл ключа сертификата сервера Docker должен быть защищен от любого вмешательства или ненужного чтения/записи. Поскольку в нем хранится закрытый ключ сертификата сервера Docker, он должен находиться в индивидуальном владении и групповом владении root, чтобы обеспечить невозможность доступа к нему менее привилегированных пользователей.

Устранение:
Выполните следующую команду:

```
chown root:root <path to Docker server certificate key file>
```

Это устанавливает индивидуальное владение и групповое владение для файла ключа сертификата сервера Docker на root.

```
  ×  docker-3.14: Verify that Docker server certificate key file permissions are
set to 444 or more restrictive (3 failed)
     ×  File  is expected to exist
     expected File  to exist
     ×  File  is expected to be file
     expected `File .file?` to be truthy, got false
     ×  File  is expected to be readable
     expected File  to be readable
     ✔  File  is expected not to be executable
     ✔  File  is expected not to be writable
```

docker-3.14: Убедитесь, что разрешения файла ключа сертификата сервера Docker установлены на 444 или более ограничительно

Обоснование:
Ключевой файл сертификата сервера Docker должен быть защищен от любого вмешательства или ненужного чтения. В нем хранится закрытый ключ сертификата сервера Docker. Поэтому он должен иметь права доступа 444, чтобы гарантировать, что файл ключа сертификата не будет изменен.

Устранение:
Необходимо выполнить следующую команду:

```
chmod 444 <path to Docker server certificate key file>
```

Это установит права доступа к файлу ключа сертификата сервера Docker на 444.

```
  ✔  docker-3.15: Verify that Docker socket file ownership is set to
root:docker
     ✔  File /var/run/docker.sock is expected to exist
     ✔  File /var/run/docker.sock is expected to be socket
     ✔  File /var/run/docker.sock is expected to be owned by "root"
     ✔  File /var/run/docker.sock is expected to be grouped into "docker"
```

docker-3.15: Убедитесь, что право собственности на файл сокета Docker установлено на root:docker

Обоснование:
Демон Docker запускается от имени root. Поэтому сокет Unix по умолчанию должен принадлежать
root. Если любой другой пользователь или процесс владеет этим сокетом, то для этого непривилегированного пользователя или процесса может оказаться возможным взаимодействие с демоном Docker. Кроме того, в этом случае непривилегированный пользователь или процесс может иметь возможность взаимодействовать с контейнерами, что не является ни безопасным, ни желательным поведением.
Кроме того, программа установки Docker создает группу Unix под названием docker. Вы можете добавить пользователей в эту группу, и в этом случае эти пользователи смогут читать и писать на стандартный Unix-сокет Docker. Членство в группе docker жестко контролируется системным администратором. Однако если сокет принадлежит какой-либо другой группе, то члены этой группы могут взаимодействовать с демоном Docker. Такая группа может контролироваться не так жестко, как группа docker. Опять же, это не соответствует хорошей практике безопасности.
По этой причине файл сокета Docker Unix по умолчанию должен принадлежать root, а группа должна принадлежать docker, чтобы сохранить целостность файла сокета.

Устранение:
Выполните следующую команду:

```
chown root:docker /var/run/docker.sock
```

Это установит право собственности на root и право групповой собственности на docker для файла сокетов Docker по умолчанию.

```
  ✔  docker-3.16: Verify that Docker socket file permissions are set to 660 or
more restrictive
     ✔  File /var/run/docker.sock is expected to exist
     ✔  File /var/run/docker.sock is expected to be socket
     ✔  File /var/run/docker.sock is expected to be readable by owner
     ✔  File /var/run/docker.sock is expected to be writable by owner
     ✔  File /var/run/docker.sock is expected not to be executable by owner
     ✔  File /var/run/docker.sock is expected to be readable by group
     ✔  File /var/run/docker.sock is expected to be writable by group
     ✔  File /var/run/docker.sock is expected not to be executable by group
     ✔  File /var/run/docker.sock is expected not to be readable by other
     ✔  File /var/run/docker.sock is expected not to be writable by other
     ✔  File /var/run/docker.sock is expected not to be executable by other
```

docker-3.16: Убедитесь, что разрешения файла сокета Docker установлены на 660 или более ограничительно

Обоснование:
Только root и члены группы docker должны иметь право читать и писать в сокет Docker Unix по умолчанию. Поэтому файл сокета Docker должен иметь разрешения 660 или более строгие разрешения.

Устранение:
Выполните приведенную ниже команду.

```
chmod 660 /var/run/docker.sock
```

Это установит разрешения файла сокета Docker на 660.

```
  ×  docker-3.17: Verify that daemon.json file ownership is set to root:root (4
failed)
     ×  File /etc/docker/daemon.json is expected to exist
     expected File /etc/docker/daemon.json to exist
     ×  File /etc/docker/daemon.json is expected to be file
     expected `File /etc/docker/daemon.json.file?` to be truthy, got false
     ×  File /etc/docker/daemon.json is expected to be owned by "root"
     expected `File /etc/docker/daemon.json.owned_by?("root")` to be truthy, got
false
     ×  File /etc/docker/daemon.json is expected to be grouped into "root"
     expected `File /etc/docker/daemon.json.grouped_into?("root")` to be truthy,
got false
```

docker-3.17: Убедитесь, что право собственности на файл daemon.json установлено на root:root

Обоснование:
Файл daemon.json содержит важные параметры, которые могут изменить поведение демона docker. Поэтому он должен принадлежать root и группе, чтобы гарантировать, что он не может быть изменен менее привилегированными пользователями.

Устранение:
Выполните приведенную ниже команду:

```
chown root:root /etc/docker/daemon.json
```

Это установит права собственности и групповые права для файла на root.

```
  ×  docker-3.18: Verify that /etc/docker/daemon.json file permissions are set
to 644 or more restrictive (6 failed)
     ×  File /etc/docker/daemon.json is expected to exist
     expected File /etc/docker/daemon.json to exist
     ×  File /etc/docker/daemon.json is expected to be file
     expected `File /etc/docker/daemon.json.file?` to be truthy, got false
     ×  File /etc/docker/daemon.json is expected to be readable by owner
     expected File /etc/docker/daemon.json to be readable by owner
     ×  File /etc/docker/daemon.json is expected to be writable by owner
     expected File /etc/docker/daemon.json to be writable by owner
     ✔  File /etc/docker/daemon.json is expected not to be executable by owner
     ×  File /etc/docker/daemon.json is expected to be readable by group
     expected File /etc/docker/daemon.json to be readable by group
     ✔  File /etc/docker/daemon.json is expected not to be writable by group
     ✔  File /etc/docker/daemon.json is expected not to be executable by group
     ×  File /etc/docker/daemon.json is expected to be readable by other
     expected File /etc/docker/daemon.json to be readable by other
     ✔  File /etc/docker/daemon.json is expected not to be writable by other
     ✔  File /etc/docker/daemon.json is expected not to be executable by other
```

docker-3.18: Убедитесь, что разрешения файла daemon.json установлены на 644 или более строгое значение

Обоснование:
Файл daemon.json содержит важные параметры, которые могут изменить поведение демона docker. Поэтому он должен быть доступен для записи только root, чтобы исключить возможность его изменения менее привилегированными пользователями.

Устранение:
Вам следует выполнить команду, приведенную ниже

```
chmod 644 /etc/docker/daemon.json
```

Это установит разрешения для этого файла на 644.

```
  ×  docker-3.19: Verify that /etc/default/docker file ownership is set to
root:root (4 failed)
     ×  File /etc/default/docker is expected to exist
     expected File /etc/default/docker to exist
     ×  File /etc/default/docker is expected to be file
     expected `File /etc/default/docker.file?` to be truthy, got false
     ×  File /etc/default/docker is expected to be owned by "root"
     expected `File /etc/default/docker.owned_by?("root")` to be truthy, got
false
     ×  File /etc/default/docker is expected to be grouped into "root"
     expected `File /etc/default/docker.grouped_into?("root")` to be truthy, got
false
```

docker-3.19: Убедитесь, что право собственности файла /etc/default/docker установлено на root:root

Обоснование:
Файл /etc/default/docker содержит важные параметры, которые могут изменить поведение демона Docker. Поэтому он должен находиться в индивидуальном владении и групповом владении root, чтобы гарантировать невозможность его изменения менее привилегированными пользователями.

Устранение:
Вам необходимо выполнить следующую команду
chown root:root /etc/default/docker
Это установит право собственности и групповое право собственности на файл для root

```
  ×  docker-3.20: Verify that /etc/default/docker file permissions are set to
644 or more restrictive (6 failed)
     ×  File /etc/default/docker is expected to exist
     expected File /etc/default/docker to exist
     ×  File /etc/default/docker is expected to be file
     expected `File /etc/default/docker.file?` to be truthy, got false
     ×  File /etc/default/docker is expected to be readable by owner
     expected File /etc/default/docker to be readable by owner
     ×  File /etc/default/docker is expected to be writable by owner
     expected File /etc/default/docker to be writable by owner
     ✔  File /etc/default/docker is expected not to be executable by owner
     ×  File /etc/default/docker is expected to be readable by group
     expected File /etc/default/docker to be readable by group
     ✔  File /etc/default/docker is expected not to be writable by group
     ✔  File /etc/default/docker is expected not to be executable by group
     ×  File /etc/default/docker is expected to be readable by other
     expected File /etc/default/docker to be readable by other
     ✔  File /etc/default/docker is expected not to be writable by other
     ✔  File /etc/default/docker is expected not to be executable by other
```

docker-3.20: Убедитесь, что разрешения файла /etc/deafult/docker установлены на 644 или более строгое значение

Обоснование:
Файл /etc/default/docker содержит важные параметры, которые могут изменить поведение демона docker. Поэтому он должен быть доступен для записи только root, чтобы исключить возможность его изменения менее привилегированными пользователями.

Устранение:
Вам следует выполнить команду, приведенную ниже

```
chmod 644 /etc/default/docker
```

Это установит разрешения для этого файла на 644.

```
  ↺  docker-6.1: Perform regular security audits of your host system and
containers
     ↺  Perform regular security audits of your host system and containers
  ↺  docker-6.2: Monitor Docker containers usage, performance and metering
     ↺  Monitor Docker containers usage, performance and metering
  ↺  docker-6.3: Backup container data
     ↺  Backup container data
```

```
  ×  host-6.4: Avoid image sprawl
     ×
["sha256:2e0fee02aedc429050e0d898f0ffdb68ea0ed9dc3cd9b1623e293375a19e517b"] is
expected to be empty
     expected
`["sha256:2e0fee02aedc429050e0d898f0ffdb68ea0ed9dc3cd9b1623e293375a19e517b"].emp
ty?` to be truthy, got false

host-6.4: Избегайте разрастания изображения

  ✔  host-6.5: Avoid container sprawl
     ✔  -1 is expected to be <= 25
  ×  host-1.1: Create a separate partition for containers
     ×  Mount /var/lib/docker is expected to be mounted

     Mount /var/lib/docker is not mounted

host-6.5: Избегайте разрастания контейнеров

  ✔  host-1.2: Use the updated Linux Kernel
     ✔  true is expected to eq true

host-1.2: Используйте обновленное ядро Linux

  ↺  host-1.3: Harden the container host
     ↺  Harden the container host. Use the Dev-Sec Hardening Framework

host-1.3: Усилить контейнерный узел
Укрепление узла контейнера. Используйте Dev-Sec Hardening Framework

  ↺  host-1.4: Remove all non-essential services from the host
     ↺  Remove all non-essential services from the host. Use the Dev-Sec
Hardening Framework

host-1.4: Удалите все несущественные службы с хоста
Удалите все несущественные службы с хоста. Используйте Dev-Sec Hardening
Framework

  ✔  host-1.5: Keep Docker up to date
     ✔  Docker Host version.Client.Version is expected to cmp >= "17.06"
     ✔  Docker Host version.Server.Version is expected to cmp >= "17.06"

host-1.5: Поддерживайте Docker в актуальном состоянии

  ×  host-1.6: Only allow trusted users to control Docker daemon (1 failed)
     ✔  Group docker is expected to exist
     ×  #<Inspec::Resources::EtcGroupView:0x00007ff81232d388> users is expected
to include "vagrant"
     expected ["node"] to include "vagrant"

host-1.6: Разрешить только доверенным пользователям управлять демоном Docker

  ×  host-1.7: Audit docker daemon (4 failed)
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist
     ×  Service auditd is expected to be installed
     expected that `Service auditd` is installed
     ×  Service auditd is expected to be enabled
     expected that `Service auditd` is enabled
     ×  Service auditd is expected to be running
     expected that `Service auditd` is running

host-1.7: Аудит демона докера
```

```
  ×  host-1.8: Audit Docker files and directories - /var/lib/docker
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.8: Аудит файлов и каталогов Docker - /var/lib/docker

  ×  host-1.9: Audit Docker files and directories - /etc/docker
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.9: Аудит файлов и каталогов Docker - /etc/docker

  ×  host-1.10: Audit Docker files and directories - docker.service
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.10: Аудит файлов и каталогов Docker – docker.service

  ×  host-1.11: Audit Docker files and directories - docker.socket
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.11: Аудит файлов и каталогов Docker – docker.socket

  ×  host-1.12: Audit Docker files and directories - /etc/default/docker
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.12: Аудит файлов и каталогов Docker – /etc/default/docker

  ×  host-1.13: Audit Docker files and directories - /etc/docker/daemon.json
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.13: Аудит файлов и каталогов Docker – docker.service

  ×  host-1.14: Audit Docker files and directories - /usr/bin/docker-containerd
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.14: Аудит файлов и каталогов Docker – /usr/bin/docker-containerd

  ×  host-1.15: Audit Docker files and directories - /usr/bin/docker-runc
     ×  Auditd Rules
     Command `/sbin/auditctl` does not exist

host-1.15: Аудит файлов и каталогов Docker – /usr/bin/docker-runc


Profile Summary: 31 successful controls, 38 control failures, 33 controls
skipped
Test Summary: 110 successful, 98 failures, 34 skipped

Резюме профиля:
      31 успешный контроль,
      38 неудачных контролей,
      33 пропущенных контроля

Итог тестирования:
      110 успешных,
      98 неудачных,
      34 пропущенных
```