«СА» – 20 лет: наши темы, наши авторы

ISSN 1813-5579
22011
9 771813 557005

Системный администратор

ежемесячный журнал www.samag.ru

№11(240)
2022

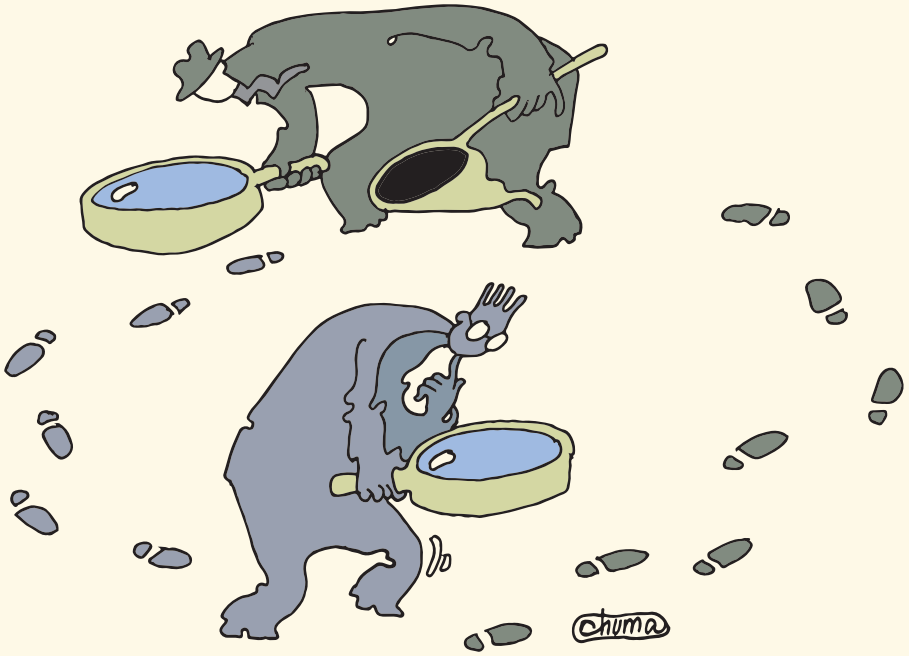
Дискуссия: «Почему не ЭДО?»

Автоматическое лицензирование Р7 - ОФИС

App Armor – как система управления мандатным доступом в Linux и системах контейнеризации

«СА» рекомендует PowerShell для сисадминов

Робоистория Роботы заселяют Землю. И ее окрестности



Вакансия: Специалист по информационной безопасности

Наука и технологии

Наука и технологии

16+

Обзор современного состояния кортикальных алгоритмов и их применение для анализа сигналов в реальном времени

Методика идентификации психофизиологического состояния человека на основе анализа электрокардиограммы в режиме реального времени

Визитка

СЕРГЕЙ ГОЛОВАШОВ,
руководитель центра компетенций DevOPS/ DevSecOPS компании Bell Integrator

Визитка

ИВАН АГАТИЙ,
к.п.н., магистр юриспруденции, главный специалист-эксперт ФСТЭК России

App Armor

как система управления мандатным доступом в Linux и системах контейнеризации

В статье рассказывается про продолжение ограничения приложений и мандатном доступе в ОС linux, а также о решениях по защите внутри контейнеров.

Security-Enhanced Linux (SELinux) – это новый метод контроля доступа в Linux на основе модуля ядра Linux Security (LSM). SELinux включен по умолчанию во многих дистрибутивах на основе Red Hat, использующих пакетную базу rpm, например, Fedora, CentOS и т.д.

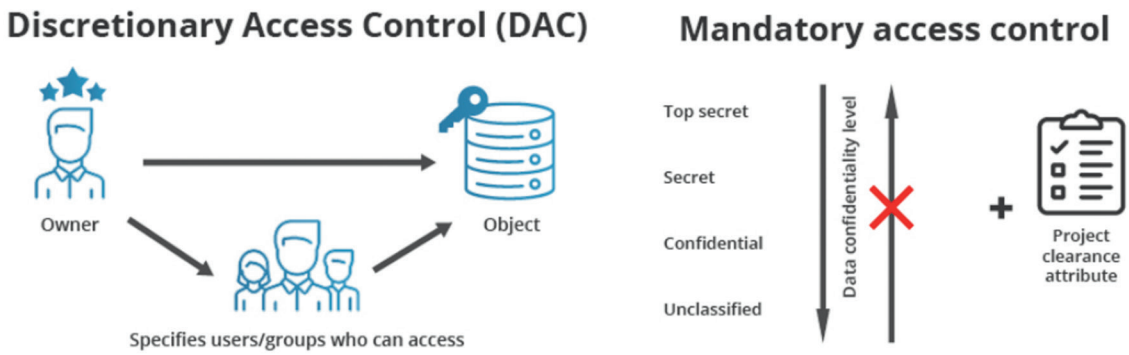
Но не только Security-Enhanced Linux (SELinux) способен обеспечивать защиту приложений на уровне ядра операционной системы. Про один из альтернативных вариантов мы сейчас и поговорим.

AppArmor – программный инструмент упреждающей защиты, основанный на политиках безопасности (известных также как профили), которые определяют, к каким системным ресурсам и с какими привилегиями может получить доступ то или иное приложение. В AppArmor включён набор стандартных профилей, а также инструменты статического анализа и инструменты, основанные на обучении, позволяющие ускорить и упростить построение новых профилей.

Изначально программа была разработана компанией Immunix. После её приобретения компанией Novell инструмент был открыт под лицензией GNU GPL и включён в openSUSE. Позже адаптирован для Ubuntu. В конце лета 2008 года Рассел Кокер, один из авторов SELinux, высказал мнение, что AppArmor бесперспективен, объяснив это тем, что даже в openSUSE появляется поддержка аналогичного и более популярного решения – SELinux. Однако вскоре разработку AppArmor продолжил сотрудник Canonical, а в июле 2010 года было объявлено о том, что AppArmor войдет в состав Linux-ядра версии 2.6.36. В мае 2013 года поддержка инструмента была внедрена в Debian 7 Wheezy.

Также, как и SELinux, AppArmor является реализацией системы Mandatory Access Control (MAC), основанной на архитектуре Linux Security Modules (LSM). Модель безопасности Apparmor заключается в привязке атрибутов контроля доступа не к пользователям, а к программам. AppArmor обеспечивает изоляцию с помощью профилей, загружаемых в ядро, как правило, при загрузке.

AppArmor отличается от остальных реализаций MAC в Linux принципом действия на основе путей, еще он позволяет смешивать профили принудительного исполнения и режима предупреждений. Кроме того, AppArmor использует вложенные файлы для облегчения разработки и имеет гораздо более пологий барьер для входа, чем тот же SELinux.



AppArmor — программный инструмент упреждающей защиты, основанный на политиках безопасности (известных также как профили), которые определяют, к каким системным ресурсам и с какими привилегиями может получить доступ то или иное приложение

DAC и MAC

Архитектура Discretionary Access Control (DAC) ограничивает доступ к критически важным ресурсам в зависимости от атрибутов субъектов или группы, к которой они принадлежат. Эти атрибуты определяют права доступа к ресурсам файловой системы. Каждому админу хорошо известно значение привилегий чтение (Read), запись (Write), и исполнение (eXecute).

Эти атрибуты распространяются на три категории пользователей: пользователь (owner), группа (group), остальные (other). Категория «пользователь» относится к одному единственному пользователю ОС, в то время как группа может содержать множество пользователей ОС. В категорию «остальные» входят те пользователи, которые не принадлежат к первым двум.

DAC модель дает владельцу ресурса право определять тип доступа для указанных категорий пользователей. Такое разграничение доступов подходит для защиты от непреднамеренных действий пользователей и позволяет ответить на следующие вопросы:

- Какие ресурсы ФС доступны данному пользователю ОС для чтения, записи и исполнения?
- Какие ресурсы ФС доступны данной группе для чтения, записи и исполнения?
- Какие ресурсы ФС доступны остальным пользователям для чтения, записи и исполнения?
- Кто из пользователей обладает достаточными правами для запуска данного процесса?

Система безопасности Mandatory Access Control (MAC) предполагает централизованный контроль над правилами политики доступа, при котором рядовые пользователи не имеют возможность вносить в них какие-либо изменения. Разработчик политики определяет, какие программы или процессы могут выполнять определенные действия с системными ресурсами. MAC фокусируется в большей степени на программах, нежели на пользователях, и решает задачу разграничения доступа процессов к ресурсам ОС.

В сущности, дизайн MAC старается копировать разграничение привилегий доступа к документации в физическом мире. Если некий сотрудник имеет права читать документы с грифом «совершенно секретно», то к стандартным конфиденциальным и внутренним документам он тоже имеет доступ.

Обратное, однако, не верно. То же самое имеет место в контексте привилегий доступа процессов ОС в архитектуре MAC. Так, если программа может читать файл /etc/sudoers, то доступ к /etc/hosts у нее тоже имеется, но обратное также неверно.

Установка App Armor

Мы возьмем ОС Linux Ubuntu, но процесс на других ОС этого семейства схожий.

Для установки выполним команду:

```
sudo apt-get install apparmor-profiles
```

Профили AppArmor имеют два режима выполнения:

- **Фиксации/Обучения:** нарушения профиля разрешаются и сохраняются в журнале. Полезно для тестирования и разработки новых профилей.
- **Предписаний/Ограничений:** принуждает следовать политике профиля, при этом также записывает нарушения в журнал.

Использование AppArmor

Пакет apparmor-utils содержит утилиты командной строки, которые можно использовать для изменения режима выполнения AppArmor, поиска статуса профиля, создания новых профилей и т.п.

1. **apparmor_status** используется для просмотра текущего статуса профиля AppArmor.

```
sudo apparmor_status
```

2. **aa-complain** переводит профиль в режим обучения (complain).

```
sudo aa-complain /path/to/bin
```

3. **aa-enforce** переводит профиль в режим ограничений (enforce).

```
sudo aa-enforce /path/to/bin
```

4. Профили AppArmor расположены в каталоге /etc/apparmor.d. Его можно использовать для управления режимом всех профилей. Введите следующую команду для перевода всех профилей в режим обучения:

```
sudo aa-complain /etc/apparmor.d/*
```

Перевод всех профилей в режим ограничений:

```
sudo aa-enforce /etc/apparmor.d/*
```

5. Команда **apparmor_parser** используется для загрузки профиля в ядро. Она также может использоваться для повторной загрузки загруженного профиля при использовании опции '-r'. Для загрузки введите:

```
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -a
```

Для перезагрузки:

```
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -r
```

6. /etc/init.d/apparmor служит для перезагрузки всех профилей:

```
sudo /etc/init.d/apparmor reload
```

7. Директория /etc/apparmor.d/disable может использоваться совместно с опцией **apparmor_parser -R** для отключения профиля.

```
sudo ln -s /etc/apparmor.d/profile.name /etc/apparmor.d/disable/
sudo apparmor_parser -R /etc/apparmor.d/profile.name
```

Для активации отключенного профиля удалите символическую ссылку на профиль в /etc/apparmor.d/disable/. Затем загрузите профиль, используя опцию '-a'.

```
sudo rm /etc/apparmor.d/disable/profile.name
cat /etc/apparmor.d/profile.name | sudo apparmor_parser -a
```

8. AppArmor можно отключить, а модуль ядра выгрузить следующей командой:

```
sudo /etc/init.d/apparmor stop
sudo update-rc.d -f apparmor remove
```

9. Для повторной активации AppArmor введите:

```
sudo /etc/init.d/apparmor start
sudo update-rc.d apparmor defaults
```

Замените **profile.name** на имя вашего профиля, которым вы хотите управлять. Также необходимо заменить /path/to/bin/ на реальный путь выполняемого файла. Например, для команды ping используйте /bin/ping

Профили

Не забывайте, что содержимое пакета apparmor-profiles находится в папке /usr/share/apparmor/extra-profiles/, готовых профилей там больше ста! И не всегда нужно писать что-то новое. Перед тем как профиль станет активным, необходимо перенести его из папки /usr/share/apparmor/extra-profiles/ в /etc/apparmor.d/.

Замените profile.name на имя вашего профиля, которым вы хотите управлять. Также необходимо заменить /path/to/bin/ на реальный путь выполняемого файла. Например, для команды ping используйте /bin/ping

Теперь его можно изучить и при желании изменить. Файлы профиля называются соответственно полному пути до исполняемого файла, которым они управляют, с заменой символа «/» на «.». Например, /etc/apparmor.d/bin.ping – это профиль AppArmor для команды /bin/ping.

Существует два основных типа правил, используемых в профиле:

1. **Записи путей** (Path entries): которые описывают, к каким файлам приложение имеет доступ в файловой системе.
2. **Записи разрешений** (Capability entries): определяют, какие права ограничиваемый процесс имеет право использовать.

В качестве примера посмотрим /etc/apparmor.d/bin.ping:

```
#include <tunables/global>
/bin/ping flags=(complain) {
   #include <abstractions/base>
   #include <abstractions/consoles>
   #include <abstractions/nameservice>

   capability net_raw,
   capability setuid,
   network inet raw,

   /bin/ping mixr,
   /etc/modules.conf r,
}
```

- **#include <tunables/global>:** включает операторы из других файлов. Это позволяет операторам, относящимся к нескольким приложениям, находиться в одном общем файле.
- **/bin/ping flags=(complain):** путь к программе, управляемой профилем, также устанавливающий режим обучения.
- **capability net_raw,:** разрешает приложению доступ к возможностям CAP_NET_RAW Posix.1e.
- **/bin/ping mixr,:** разрешает приложению доступ на чтение и выполнение файла.

После редактирования файла профиля, он должен быть перезагружен.

Создание собственного профиля

Итак, приступим к созданию собственного профиля. Делать это будем в несколько шагов:

1. **Разработка плана тестирования:** Попробуйте подумать о том, как приложение будет выполняться. План тестирования стоит разделить на маленькие тестовые блоки. Каждый тестовый блок должен иметь краткое описание и перечень шагов выполнения. Некоторые стандартные тестовые блоки:
   - Запуск программы.
   - Остановка программы.
   - Перезагрузка программы.
   - Тестирование всех команд, поддерживаемых сценарием инициализации.
2. **Создание нового профиля:** Используйте **aa-genprof** для создания нового профиля. Команда в терминале:

```
sudo aa-genprof executable
```

Например:

```
sudo aa-genprof slapd
```

3. Чтобы получить ваш новый профиль в составе пакета **apparmor-profiles**, зарегистрируйте проблему в Launchpad для пакета AppArmor:
   - Включите ваш план тестирования и тестовые блоки.
   - Присоедините ваш новый профиль к зарегистрированной проблеме.

Обновление профилей

Когда программа ведет себя неправильно, проанализируйте сообщения, отправленные в файлы журналов. Программа aa-logprof может быть использована для сканирования файлов журнала AppArmor для проверки сообщений, их рассмотрения (анализа) и обновления профилей. Команда в терминале:

```
sudo aa-logprof
```

Профили безопасности AppArmor для Docker

Docker автоматически создает и загружает профиль по умолчанию для контейнеров с именем docker-default. Двоичный файл Docker создает этот профиль в tmpfs, а затем загружает его в ядро.

Внимание! Мы рекомендуем отказаться от использования Docker и перейти на решения Podman, что позволит не выдавать избыточные права пользователям группы Docker вашего сервера виртуализации!

Примечание. Этот профиль используется в контейнерах, а не в Docker Daemon.

Профиль для демона Docker Engine существует, но в настоящее время он не установлен с пакетами deb. Если вас интересует источник для профиля демона, он находится в contrib / apparmor в исходном репозитории Docker Engine.

Применение политики для конейнера

docker-default профиль по умолчанию для запуска контейнеров.

Это умеренно защитный профиль, который обеспечивает широкую совместимость приложений. Когда вы запускаете контейнер, то он будет использовать политику docker-default, только если вы не переопределите его с параметром security-opt. Например, следующее явно указывает политику по умолчанию:

```
$ docker run --rm -it --security-opt apparmor=docker-default
 hello-world
```

Профили загрузки и выгрузки

Для загрузки нового профиля в AppArmor для использования с контейнерами:

```
$ apparmor_parser -r -W /path/to/your_profile
```

Затем запустите пользовательский профиль с параметром --security-opt следующим образом:

```
$ docker run --rm -it --security-opt apparmor = ваш_профайл
привет-мир
```

Чтобы выгрузить профиль из AppArmor:

```
# выгружаем профиль $ apparmor_parser -R /path/to/profile
```

Nginx пример профиля

В данном примере вы создаете пользовательский профиль AppArmor для Nginx. Ниже приведен пользовательский профиль.

```
#include <tunables/global>
profile docker-nginx flags=(attach_disconnected,mediate_
deleted) {
   #include <abstractions/base>
   network inet tcp,
   network inet udp,
   network inet icmp,
   deny network raw,
   deny network packet,
   file,
   umount,
   deny /bin/** wl,
   deny /boot/** wl,
   deny /dev/** wl,
   deny /etc/** wl,
   deny /home/** wl,
   deny /lib/** wl,
   deny /lib64/** wl,
   deny /media/** wl,
   deny /mnt/** wl,
   deny /opt/** wl,
   deny /proc/** wl,
   deny /root/** wl,
   deny /sbin/** wl,
   deny /srv/** wl,
   deny /tmp/** wl,
   deny /sys/** wl,
   deny /usr/** wl,
   audit /** w,
   /var/run/nginx.pid w,
   /usr/sbin/nginx ix,
   deny /bin/dash mrwklx,
   deny /bin/sh mrwklx,
```

```
   deny /usr/bin/top mrwklx,
   capability chown,
   capability dac_override,
   capability setuid,
   capability setgid,
   capability net_bind_service,
   deny @{PROC}/* w,# запретить запись для всех файлов
непосредственно в / proc (не в подкаталог)
   # запретить запись в файлы не в / proc / <number> / **
или / proc / sys / **
   deny @{PROC}/{[^1-9],[^1-9][^0-9],[^1-9s][^0-9y][^0-9s],
[^1-9][^0-9][^0-9][^0-9]*}/** w,
   deny @{PROC}/sys/[^k]** w,   # запретить / proc / sys,
кроме / proc / sys / k * (фактически / proc / sys / kernel)
   deny @{PROC}/sys/kernel/{?,??,[^s][^h][^m]**} w,
# запретить все, кроме shm * в / proc / sys / kernel /
   deny @{PROC}/sysrq-trigger rwklx,
   deny @{PROC}/mem rwklx,
   deny @{PROC}/kmem rwklx,
   deny @{PROC}/kcore rwklx,
   deny mount,
   deny /sys/[^f]*/** wklx,
   deny /sys/f[^s]*/** wklx,
   deny /sys/fs/[^c]*/** wklx,
   deny /sys/fs/c[^g]*/** wklx,
   deny /sys/fs/cg[^r]*/** wklx,
   deny /sys/firmware/** rwklx,
   deny /sys/kernel/security/** rwklx,
}
```

Сохраните пользовательский профиль на диск в файле /etc/apparmor.d/containers/docker-nginx.

Путь к файлу в этом примере не является обязательным. В производстве можно использовать другой.

Загрузите профиль.

```
$ sudo apparmor_parser -r -W /etc/apparmor.d/containers/
docker-nginx
```

4. Запустите контейнер с профилем.

5. Для запуска nginx в автономном режиме:

```
$ docker run --security-opt "apparmor=docker-nginx" -p 80:80
-d --name apparmor-nginx nginx
```

6. Выполнить в бегущий контейнер.

```
$ docker container exec -it apparmor-nginx bash
```

7. Попробуйте несколько операций, чтобы проверить профиль.

На этот раз профиль является docker-default, который по умолчанию запускается в контейнерах, если только он не находится в privileged режиме. Эта строка показывает, что apparmor отклонил ptrace в контейнере. Это именно так, как и ожидалось

```
root@6da5a2a930b9:~# ping 8.8.8.8
ping: Lacking privilege for raw socket.

root@6da5a2a930b9:/# top
bash: /usr/bin/top: Permission denied

root@6da5a2a930b9:~# touch ~/thing
touch: cannot touch 'thing': Permission denied

root@6da5a2a930b9:/# sh
bash: /bin/sh: Permission denied

root@6da5a2a930b9:/# dash
bash: /bin/dash: Permission denied
```

Поздравляю! Вы только что установили контейнер, защищенный пользовательским профилем аппармера!

Отладка AppArmor

Вы можете использовать dmesg для задач отладки и aa-status проверки загруженных профилей.

Используйте dmesg

Ниже приведены некоторые полезные советы по отладке любых проблем, с которыми вы можете столкнуться при работе с AppArmor.

AppArmor отправляет довольно подробные сообщения в dmesg. Обычно строка AppArmor выглядит следующим образом:

```
[ 5442.864673] audit: type=1400 audit(1453830992.845:37):
apparmor="ALLOWED" operation="open" profile="/usr/bin/docker"
name="/home/jessie/docker/man/man1/docker-attach.1" pid=10923
comm="docker" requested_mask="r" denied_mask="r" fsuid=1000
ouid=0
```

В приведенном выше примере вы можете увидеть profile=/usr/bin/docker. Это означает, что у пользователя загружен профиль docker-engine (Docker Engine Daemon).

Посмотри на другую строку:

```
[ 3256.689120] type=1400 audit(1405454041.341:73):
apparmor="DENIED" operation="ptrace" profile="docker-default"
pid=17651 comm="docker" requested_mask="receive"
denied_mask="receive"
```

На этот раз профиль является docker-default, который по умолчанию запускается в контейнерах, если только он не находится в privileged режиме. Эта строка показывает, что apparmor отклонил ptrace в контейнере. Это именно так, как и ожидалось.

Вместо заключения

Ну вот мы и рассмотрели app armor как очень неплохую альтернативу Se Linux. Не забывайте смотреть обнаруженные уязвимости на сайте «Банк данных угроз безопасности информации», там уже освещены и проблемы данного решения: https://bdu.fstec.ru/vul/2020-04920?viewtype=tile

В целом не забывайте про безопасность ваших контейнерных решений и серверов! Ну а мы продолжим рассказывать вам о методах защиты серверов, практику использования защиты и ПО для этого. EOF

Ключевые слова: linux архитектура, app armor, режимы, политики, создание собственных политик.