Доклад на тему SELinux.

Университет: Российский Университет Дружбы Народов

Факультет: ФИЗМАТ

Группа: НКНбд-01-18

Предмет: Информационная безопасность

Студент: Подмогильный Иван Александрович

Преподаватель: Кулябов Дмитрий Сергеевич

## 

## 

## Содержание:

1. Введение
2. Преимущества SELinux
3. Основные понятия
4. Основы взаимодействия

Введение:

SELinux (англ. Security-Enhanced Linux — Linux с улучшенной безопасностью) — реализация системы принудительного контроля доступа (MAC - Mandatory Access Control), которая может работать параллельно с классической избирательной системой контроля доступа (DAC - Discretionary Access Control). Изначально была разработана Агентством национальной безопасности США (NSA).

В то время как в обычном Linux права на файл определяются через группы - владелец, участник группы, и все остальные пользователи (rwx), а также имеется возможность перехода в режим суперпользователя, в котором пользователь может определять любые права на файлы и процессы.

В SELinux права доступа определяются самой системой при помощи специально определенных политик. Политики работают на уровне системных вызовов и применяются самим ядром (но можно реализовать и на уровне приложения). SELinux действует после классической модели безопасности Linux. Иными словами, через SELinux нельзя разрешить то, что запрещено через права доступа пользователей или групп. Политики описываются при помощи специального гибкого языка описания правил доступа. В большинстве случаев правила SELinux «прозрачны» для приложений, и не требуется никакой их модификации. В состав некоторых дистрибутивов входят готовые политики, в которых права могут определяться на основе совпадения типов процесса (субъекта) и файла (объекта) — это основной механизм SELinux. Две других формы контроля доступа — доступ на основе ролей и на основе многоуровневой системы безопасности.

Преимущества SELinux

SE Linux обеспечивает большую безопасность вашей системы. Пользователям могут быть назначены предопределенные роли таким образом, что они не смогут получить доступ к файлам и процессам, которыми они не владеют. При этом не существует эквивалента операции "chmod 777". Это отличается от обычной системы Unix-привилегий в том, что определённые пользователем роли, или контексты безопасности в которых они находятся, имеют ограниченный, но более управляемый доступ к файлам и другим ресурсам. Рассмотрим пользовательский файл .rhosts на обычной Unix системе. Если всем дать доступ на запись в этот файл, тогда кто угодно сможет зайти в систему. Используя SE Linux, можно контролировать возможность пользователя изменять права доступа в своему файлу .rhosts, а кроме того запретить другим людям писать в этот файл даже после того, как владелец это разрешил.

Допустим, есть ошибка в программе /usr/bin/passwd, которая позволяет выполнить команду chmod 666 /etc/shadow. В этом случае вступают в действия права SE Linux, которые предотвратят неавторизованный доступ к файлу.

Основные понятия

Сущность в SE Linux это не то же, что и традиционный идентификатор пользователя (Unix uid, user id). Они могут сосуществовать в одной системе, но их смысл совершенно разный. Сущность в SE Linux формирует часть контекста безопасности, который задает домены, в которые можно войти. То есть что собственно можно сделать. Сущность SELinux может иметь одинаковое с именем пользователя символьное представление (чаще всего так и есть), но важно понимать, что это две разные вещи. Выполнение команды su не меняет сущности пользователя в SELinux.

### Пример:

Непривилегированный пользователь faye выполняет команду id (в SELinux) и видит свой контекст безопасности: context=faye:user\_r:user\_t Часть контекста "faye" представляет сущность. Теперь, пользователь faye выполняет su, чтобы получить привилегии пользователя root, и вызывает id, и видит, что контекст всё ещё:

context=faye:user\_r:user\_t то есть, контекст остался прежним и не изменился на контекст пользователя root. Однако, если сущность faye разрешает доступ к роли sysadm\_r, и пользователь это сделает (введя команду newrole -r), и снова выполнит id, то увидит уже:

context=faye:sysadm\_r:sysadm\_t

Итак, сущность осталась той же, но роль и домен (второе и третье поле соответственно) изменились. Такой стиль работы с сущностью обеспечивает возможность идентификации пользователя. Ключевым моментом в безопасности системы является то, что сущность пользователя определяет какие роли и домены могут быть использованы.

## Домен (domain)

Каждый процесс выполняется в домене. Домен однозначно определяет

привилегии процесса. По существу домен это список того, что может

делать процесс, или какие действия процесс может выполнять над

различными типами. Домен можно понимать, как стандартный Unix uid.

Пусть у пользователя root есть какая-то программа, для которой он

выполнил команду chmod 4777 (установил атрибут setuid). Кто угодно в

системе, даже пользователь nobody, может выполнить эту программу с

полномочиями пользователя root, тем самым, нарушая безопасность

системы. При использовании SELinux, процесс, инициирующий переход в

привилегированный домен, должен иметь роль, которой разрешено

использовать этот домен, иначе процесс работать не сможет.

В качестве примеров доменов можно привести sysadm\_t, домен системной

администрации, и user\_t, который является общим доменом для

непривилегированных пользователей. Процесс init выполняется в домене

init\_t, а named -- в домене named\_t.

## Тип (type)

Тип задаётся для объекта и определяет доступ к этому объекту. То есть. это

приблизительно то же самое, что и домен, но домен относится к

процессам, а тип к таким объектам, как каталоги, файлы, сокеты и так далее.

## Роль (role)

Роль определяет, какие домены могут быть использованы. Домены, к

которым имеет доступ пользовательская роль, предопределяются в

конфигурационных файлах политики. Если роль не имеет доступа к

заданному домену (в базе данных политики), то при попытке выполнить

это действие доступ будет запрещён.

### Пример:

Для того, чтобы разрешить пользователю из домена user\_t (домен

непривилегированных пользователей) выполнять команду passwd, в

соответствующем файле конфигурации указано:

role user\_r types user\_passwd\_t

Она означает, что пользователь с ролью user\_r может входить в домен

user\_passwd\_t, то есть может выполнять команду passwd.

## Контекст безопасности (security context)

Контекст безопасности это набор всех атрибутов, связанных с объектами типа файлов, каталогов, процессов, TCP сокетов и т.п. Контекст безопасности состоит из сущности, роли и домена или типа. Важно понимать различие между доменом и типом.

Рассмотрим файловую систему /proc. У каждого процесса есть домен, а в файловой системе /proc для каждого процесса есть каталог. Каждый процесс имеет метку, или точнее, контекст безопасности, в применении к файлу. Но в файловой системе /proc, метка содержит тип, а не домен. Не смотря на то, что /proc представляет собой выполняющиеся процессы, содержимое /proc является файлами, а потому имеет тип, а не домен.

Вызов команды ls --context /proc выдаёт следующую информацию для процесса init (процесс с идентификатором 1):

dr-xr-xr-x root root system\_u:system\_r:init\_t 1

Метка, или контекст безопасности, говорит нам о том, что этот файл имеет тип init\_t. Но, кроме того, это значит, что процесс init выполняется в домене init\_t. Каждый файл и каталог файловой системы /proc, соответствующий некоторому процессу, также следует этому соглашению, т.е. тип, указанный в выводе команды ls --context соответствует, так же, домену в котором выполняется процесс.

Ещё одним важным моментом является то, что команды chsid (изменить идентификатор безопасности) и chcon (изменить контекст) не работают на файловой системе /proc, т.к. она не поддерживает изменение меток. Контекст безопасности файла, например, может варьироваться в зависимости от домена, который создал файл. По умолчанию, новый файл или каталог наследует тип от родительского каталога, однако вы можете задать иную политику.

### Пример:

Пользователь faye создаёт файл test в своем домашнем каталоге. После чего выполняет команду ls --context test и видит:

-rw-r--r-- faye faye faye:object\_r:user\_home\_t test

Теперь faye создаёт файл в каталоге /tmp с именем tmptest и выполняет команду ls --context /tmp/tmptest. Теперь результат такой :

-rw-r--r-- faye faye faye:object\_r:user\_tmp\_t /tmp/tmptest

В первом примере контекст безопасности включал тип "user\_home\_t", который является типом по умолчанию для домашнего каталога непривилегированного пользователя с ролью user\_r. После выполнения второй команды ls --context, видим, что тип теперь user\_tmp\_t. Это тип по умолчанию для файлов, созданных процессами домена user\_t в каталоге типа tmp\_t.

## Переход (transition)

Решение о переходе, определяет контекст безопасности, который будет назначен запрошенной операции. Есть два основных типа переходов. Первый тип, это переход домена процесса. Он используется при выполнении процесса определённого типа. Второй, это переход типа файла, который применяется при создании файла в определенных каталогах.

### Пример:

При создании пользователем файла в каталоге /tmp происходит переход в тип user\_tmp\_t и новый файл помечается соответствующим образом.

Рассмотрим теперь пример перехода домена процесса. Если запустить ssh из-под обычного пользователя, или, точнее, из домена user\_t (для проверки текущего контекста безопасности можно использовать команду id). Теперь при выполнении ps ax --context мы найдём строку с данными о команде ssh. Полагая, что это сделал пользователь faye, она, в частности, увидит:

faye:user\_r:user\_ssh\_t

Процесс ssh выполняется в домене user\_ssh\_t потому что программа имеет тип ssh\_exec\_t, а роль user\_r имеет право доступа в домен user\_ssh\_t.

## Политика (policy)

Политики -- это наборы правил, контролирующие такие вещи как список ролей, к которым имеет доступ пользователь, какие роли имеют доступ к каким доменам и какие домены имеют доступ к каким типам. Можно редактировать файлы политик в соответствии с тем, как пользователь хочет настроить свою систему.

Основы взаимодействия

## Передача пользовательского контекста при регистрации в системе.

При установке пакета selinux-policy-default (или policy-sources для Fedora), политика разрешает вход в систему для роли пользователя по умолчанию.

При регистрации как пользователь root, контекст безопасности будет root:user\_r:user\_t. При вводе команды id будет примерно следующее:

uid=0(root) gid=0(root) groups=0(root) context=root:user\_r:user\_t

итак, в этой строке контекст безопасности представлен выражением

root:user\_r:user\_t

Допустим, у пользователя заранее прописана возможность перехода в другой контекст. Есть два способа переключения между ролями. Во-первых, при регистрации. Допустим, пользователю faye разрешено входить в домен sysadm\_t. При регистрации на консоли она увидит вопрос "Your default context is faye:user\_r:user\_t. Do you want to choose a different one? [n]" (Ваш контекст по умолчанию faye:user\_r:user\_t. Хотите выбрать другой?). Если пользователь ответит y, система даст возможность выбора контекста:

[1] faye:user\_r:user\_t

[2] faye:sysadm\_r:sysadm\_t

Enter number of choice:

В этом примере мы видим, что сущности "faye" разрешён доступ к роли sysadm\_r и домену sysadm\_t. Выводятся лишь те варианты, к которым имеет доступ сущность. Если пользователь faye выберет второй вариант (чтобы получить роль sysadm\_r), после чего вызовет команду id, она увидит, что её контекст теперь:

context=faye:sysadm\_r:sysadm\_t

## Изменение контекста командой newrole -r

Второй способ переключения контекстов безопасности -- это команда newrole -r. Синтаксис этой команды:

newrole -r role

где role -- это желаемая роль. Например, для получения роли sysadm\_r

newrole -r sysadm\_r

Если у вас нет прав доступа к указанной роли, вы увидите следующее сообщение (положим, что сущность fred пытается получить доступ к роли sysadm\_r):

fred:sysadm\_r:sysadm\_t is not a valid context

Это сообщение означает, что fred не может войти в sysadm\_r:sysadm\_t роль/домен, так как у него нет на это прав.

## Разрешающий и принудительный режимы

Разрешающий режим -- это режим, в котором SE Linux лишь протоколирует сообщения, ничего более того (вы можете работать точно так же, как и на машине без SE Linux). Принудительный режим -- это режим, в котором применяются все настроенные политики. Разрешающий режим хорош для отладки работы системы. Сообщения можно увидеть с помощью команды dmesg.

Загружаться в принудительный режим стоит только когда в системе SELinux уже всё настроено. Для настройки рекомендуется использовать разрешающий режим. Если ядро будет скомпилировано без флага CONFIG\_SECURITY\_SELINUX\_DEVELOP, то пользователь не вообще не сможет зайти в разрешающий режим.

Для переключения в принудительный режим используется команда setenforce 0, аналогично для переключения в разрешающий режим используется команда setenforce 1.

## Сравнение результатов выполнения команд в разных ролях

При запуске из роли user\_r команды ps ax --context и просмотре результата, пользователь увидит не все процессы, а лишь список, который пользователю user\_r позволено видеть.

Если же выполнить команду ps ax --context в роли sysadm\_r, то будет выведен список всех процессов в системе. Это полезно, так как если злоумышленник в роли user\_r увидит все демоны, то увеличится вероятность найти уязвимый демон.

Другим примером уязвимости могут служить программы, которым пароль передается параметром командной строки. В обычной Linux-системе такой пароль будет виден всем пользователям, тогда как SE Linux этого не допустит (конечно, пароль в командной строке -- это все равно плохо).

Заключение.

Мы ознакомились с системой дополнительной безопасности SELinux, которая была изначально разработана NDA, исследовали её преимущества перед системой Linux. Далее мы рассмотрели основные понятия SELinux, что является базой для освоения всего остального материала, а также затронули некоторые команды взаимодействия с системой SELinux. Следует помнить, что SELinux не является отдельной ОС, как например, Ubuntu, также SELinux не является фаервлом. Хотя в каком-то смысле его можно понимать как фильтр предоставления данных.

Источники:

1. Фей Кокер, перевод Ивана Песина <https://www.opennet.ru/base/sec/intro_selinux.txt.html>
2. Официальный сайт: <https://selinuxproject.org/page/NewUsers>
3. FAQ официального сайта: <https://selinuxproject.org/page/FAQ>
4. Read Hat: <https://www.redhat.com/en/topics/linux/what-is-selinux>
5. Wikipedia: <https://en.wikipedia.org/wiki/Security-Enhanced_Linux>
6. Официальный репозиторий: <https://github.com/SELinuxProject/selinux>
7. Презентация NSA по SELinux: <https://www.nsa.gov/portals/75/documents/what-we-do/research/selinux/documentation/presentations/2003-ottawa-linux-symposium-bof-presentation.pdf>
8. Implementing SELinux as a Linux Security Module: <https://www.nsa.gov/portals/75/documents/resources/everyone/digital-media-center/publications/research-papers/implementing-selinux-as-linux-security-module-report.pdf>
9. Slides by Paul Wayper: <https://www.slideshare.net/PaulWay/selinux-for-everyday-users>
10. Paranoid Penguin - Introduction to SELinux: <https://www.linuxjournal.com/article/9500>
11. Paranoid Penguin - Introduction to SELinux, Part II: <https://www.linuxjournal.com/article/9542>