Лабораторна робота

SELinux

Система SELinux (Security-Enhanced Linux - Linux з поліпшеною безпекою) була розроблена міністерством оборони США і всього за кілька років стала стандартом в області систем контролю прав доступу. Вона була включена в ядро Linux версії 2.6.0 і вперше з'явилася як робоче рішення «з коробки» в дистрибутиві Red Hat Enterprise Linux 4. Згодом підтримка SELinux була інтегрована в такі дистрибутиви, як Debian, OpenSUSE і Ubuntu, а додаткові пакети, що активують систему, були додані в багато інших більш-менш популярних дистрибутивів. Саме завдяки SELinux дистрибутиву RHEL5, що працює на серверах IBM, вдалося отримати сертифікат безпеки EAL4 Augmented with ALC_FLR.3, який в той час мала лише одна операційна система - Trusted Solaris. SELinux істотно розширює ядро Linux, роблячи операційні системи, побудовані на його основі, придатними до використання в сферах, де доступ до інформації повинен бути строго розмежовано, а застосунки, що запускаються користувачами, повинні мати певний набір прав, який не виходить за рамки мінімально необхідного. Це робить систему привабливою для держустанов і військових, проте здається не зовсім зрозумілим, навіщо вона потрібна адміністраторам рядових серверів, а вже тим більше - звичайним користувачам (в Fedora SELinux за замовчуванням включений).

Творці UNIX наділили свою ОС простим, але вельми гнучким і красивим механізмом безпеки. В його основі лежать всім нам відомі права доступу на файли, які визначають коло осіб (користувачів, процесів], здатних виконувати будь-які дії над файлами і каталогами. Наприклад, ми можемо створити виконуваний файл і виставити на нього права доступу «*-rwxr- xr-x*». що буде означати, що ми [тобто власник файлу] можемо робити з ним все що захочемо, користувачі, що входять в нашу групу, можуть його читати і виконувати, а всі інші - тільки виконувати. з огляду на той факт, що всі пристрої в UNIX представлені файлами, такий механізм дозволяє не тільки чітко розмежовувати доступ користувачів [їх застосунків] до інформації, але і до пристроїв і навіть до деяких функцій операційної системи [procfs і sysfs]. Однак, у механізму прав доступу є два серйозні недоліки. По-перше, їх реалізація дуже незграбна. Вона добре підходить для розмежування процесів та конфіденційної інформації користувачів, але абсолютно не підходить для гнучкого управління їх можливостями. Наприклад, щоб дати будь-якій дисковій утиліти [cfdisk, наприклад] можливість модифікації інформації, що зберігається на диску, ми можемо або дозволити повний доступ до диска групі, до якої належить її власник, або запустити її з правами користувача root. Як в першому, так і в другому випадку ми створимо потенційну загрозу безпеці: всі інші користувачі/застосунки групи отримають права доступу до диска, або сама утиліта отримає найвищі і безмежні права в системі. А що якщо потрібно дати доступ не до модифікації диска, а тільки до виклику

певних його функцій [ioctl]. Тут права доступу взагалі не допоможуть. По-друге, файлами в Linux представлені далеко не всі ресурси операційної системи. Величезну кількість функцій ядра приховано в більш ніж трьох сотнях системних викликів, велика частина яких доступна для використання абсолютно всім процесам, а якщо процес має права root, то його можливості взагалі ніяк не обмежуються. Якщо, наприклад, ми захочемо запустити FTP-сервер на стандартному для нього 21 порту, який доступний тільки процесам з правами root, нам доведеться повністю довіритися його розробнику, оскільки працюючи від root, FTP-сервер буде здатний робити абсолютно все і помилка, знайдена в його коді, дасть зломщикові повний контроль над системою. І це через просту потребу слухати привілейований порт! SELinux вирішує ці проблеми, дозволяючи надзвичайно гнучко контролювати відносини процесу і операційної системи. З його допомогою можна обмежити процес в можливості звернення до тих чи інших системних викликів або файлів, контролювати те, як відбувається системний виклик, які при цьому використовуються аргументи, забороняти або дозволяти прив'язку процесу до певних портів, коротше кажучи, керувати можливостями процесів на найнижчому рівні.

* Підсистема SELinux працює опісля класичного механізму управління доступом UNIX, тому з її допомогою не можна дозволити те, що вже заборонено за допомогою традиційних прав доступу.

Якби ти читав товсту книжку без картинок, то тут тебе мали чекати міркування про такі штуки, як дискреційний і мандатний контроль доступу, RBAC, MCS і інших навколонаукових речах. Зрозуміти принцип роботи SELinux і закладені в нього ідеї найпростіше розібравшись з тим, що відбувається під час його активації. Що робить SELinux для того, щоб контролювати звернення процесів до ресурсів ОС? Концептуально можна виділити чотири етапи:
1. Всі суб'єкти [процеси] і об'єкти [файли, системні виклики і т.д.] взаємодії позначаються за допомогою спеціальних міток, які називаються контекстом безпеки [процеси під час запуску, файли - під час створення або установки ОС, їх мітки зберігаються в розширених атрибутах ФС, системні виклики - при компіляції модуля SELinux].
2. Коли суб'єкт намагається виконати будь-яку дію щодо об'єкта, інформація про цю дію надходить до виконуючої підсистеми SELinux,
3. Виконуюча підсистема враховує контексти безпеки суб'єкта і об'єкта і, звіряючись з написаними раніше правилами (так звана політика), приймає рішення про дозволеність цієї дії.
4. Якщо дія виявляється правомочною (політика її дозволяє), об'єкт [програма] продовжує працювати в звичайному режимі, в іншому випадку - вона або примусово завершується, або отримує ввічливу відмову. В реальній ситуації все це виглядає приблизно так: один з скриптів ініціалізації дистрибутива, що має мітку initrc_t [насправді цю мітку мають породжувані їм процеси, але зараз це не важливо], запускає веб-сервер Apache за допомогою файлу /usr/sbin/httpd, який має позначку httpd_exec_t. Як і всі інші дії, запит на цю операцію надходить в SELinux, в правилах [політиці] якого зазначено, що initrc_t не тільки

має праао запускати файл з міткою httpd_exec_t, а й те, що при запуску цього файлу процес і його нащадки повинні отримати мітку httpd_t. Тепер в системі з'являється один або кілька процесів Apache, позначених як httpd_t. Кожен з них буде підкорятися правилам SELinux, що відповідають цій мітці, тому, якщо в політиці зазначено, що httpd_t може отримувати доступ тільки до файлів, помічених як httpd_sys_content_t і 80 порту, Apache не зможе порушити ці правила [переключивши його на інший порт ми просто отримаємо помилку запуску]. Резонне питання: звідки беруться правила? Їх пишуть люди, а точніше - мантейнери дистрибутивів, що не заважає системному адміністратору виправити їх відповідно до своїх потреб. Зазвичай правила завантажуються на першому етапі ініціалізації ОС, але за допомогою особливих прийомів їх можна підсунути і в уже працюючу систему (Не перезавантажувати ж сервер з аптаймом 1,5 року тільки для того, щоб дозволити FTP-сервера ходити по симлінках). Типовий обсяг файлу політік SELinux для основних додатків і сервісів дистрибутива може містити до декількох сотень тисяч рядків, тому, щоб не обтяжувати адмінів рутинною роботою, були створені інструменти для їх автоматичної генерації за допомогою навчання (наприклад, утиліта audit2allow дозволяє складати правила SELinux на основі лог-файлів підсистеми audit). Далі докладніше зупинимося на цьому питанні, однак для початку слід розібратися з «політикою управління доступом на основі ролей».

* Обмежені користувачі
Крім SELinux-користувача unconfined_u, який по дефолту присвоюється всім користувачам системи, в цільовій політиці також описано кілька непривілейованих користувачів, яких можна використовувати для створення гостьових облікових записів, процеси яких будуть дуже обмежені в правах [можливості і відмінності цих користувачів дивись в таблиці «дефолтовая користувачі SELinux »]. Щоб створити такого користувача, достатньо виконати наступну команду:

```
#useradd -Z xguest_u имя_юзера
```

Крім того можна зробити його дефолтовим, так що обмеженими будуть все новостворені Linux-користувачі:

```
#semanage login -m -S targeted -s "xguest_u" -r s0 _default_
```

Подивитися список поточних SELinux-користувачів можна так:

```
#/usr/sbin/semanage login -l
```

Керування доступом на основі ролей (RBAC)

Само собою зрозуміло, що SELinux не зміг би так радувати військових, якби в його основі лежали тільки мітки, що визначають контекст безпеки, і механізм розмежування доступу на їх основі [до речі, це

називається «мандатним керуванням доступом», і багато хто помилково приписує його SELinux]. Через свою низькорівневість така система безпеки була б занадто складною в управлінні і негнучкою, тому творці SELinux забезпечили її ще одним рівнем доступу, що має назву «ролями».

Коли говорили про мітки SELinux, навмисно опущено одну важливу деталь. Насправді контекст безпеки [мітка] складається не з одного, а з трьох компонентів (є ще і четвертий компонент, але про це поки не варто замислюватися): імені користувача, ролі і типу суб'єкта [той самий компонент, який закінчується на «_t »]. Кожен користувач SELinux [який може бути пов'язаний з обліковим записом користувача Linux, але зазвичай все Linux-користувачі відображаються в SELinux-користувача unconfined_u] може виконувати кілька ролей, в рамках кожної з яких йому доступні кілька типів суб'єктів, а кожен суб'єкт, в свою чергу, може мати доступ до деякій кількості об'єктів.

Пользователь	Домен	X Window System	su и sudo	Выполнение в домашнем каталоге и /tmp/	Сетевые функции
guest_u	guest_t	нет	нет	опционально	нет
xguest_u	xguest_t	да	нет	опционально	только **Firefox**
user_u	user_t	да	нет	опционально	да
staff_u	staff_t	да	только **sudo**	опционально	да

Дефолтові користувачі SELinux

Як і групи користувачів в класичній моделі керування доступом UNIX, ролі використовуються для наділення процесів користувача різними видами повноважень. Однак, фактично вони потрібні тільки для того, щоб тонко регулювати доступ користувачів до даних, тобто при супроводі комп'ютерної інфраструктури великих підприємств, співробітники яких можуть мати різні рівні доступу до таємної інформації (тобто, ті самі військові і держустанови). При використанні SELinux на звичайних серверах ролі не грають великої ролі [парадокс, та й годі]. Будь-який дистрибутив Linux, з коробки оснащений SELinux, за замовчуванням використовує так звану «цільову політику», яка передбачає наявність всього декількох загальних SELinux-користувачів і ролей. Наприклад, цільова політика Fedora і RHEL визначає тільки двох дефолтових користувачів (насправді, є ще кілька спеціальних користувачів, але зазвичай вони не використовуються] і дві ролі: користувач system_u, роль system_r і користувач unconfined_u, роль unconfined_r. Перші використовуються для запуску системних процесів під час ініціалізації системи і в політиці чітко вказано, що додатки, які запускаються користувачем system_u [який має роль system_r] повинні отримувати конкретний тип суб'єкта (наприклад, httpd_t), який визначається типом об'єкта (файлу), з якого вони за пускаються [наприклад, httpd_exec_t]. Щодо

них діють суворі правила обмеження, про які ми говорили в попередньому розділі. Користувач unconfined_u і роль unconfined_r призначені для звичайних Linux-користувачів. На останньому етапі ініціалізації системи запускається менеджер входу в систему (він працює в домені system_u: system_r: login_t), який приймає ім'я користувача і його пароль і запускає шелл/графічну оболонку. Однак замість того, щоб зберегти поточний контекст безпеки, або змінити його на system_u: system_r: shell_t відповідно до правил політики SELinux [якби такі були], він звертається до PAM-модулю pam_selinux, який повідомляє SELinux, що запущений менеджером входу процес [шелл або оболонка] повинен отримати контекст unconfined_u: unconfined_r: unconfined_t. Сенс цього переходу в тому, що цільова політика SELinux має правила, які дозволяють застосункам, що працюють в цьому контексті, робити що завгодно, в тому числі - запускати інші програми. Однак, якщо користувач запустить застосунок, в контексті безпеки якого вказан SELinux-користувач system_u і тип, для якого в політиці є правила, процеси цього застосунку будуть обмежені в правах точно так же, як якщо б вони були запущені під час ініціалізації.

Якщо говорити про контекст безпеки файлів, то тут ролі не використовуються в принципі, і друге поле завжди дорівнює object_r, а перше буде або syslem_u, або unconfined_u, якщо це файл, створений користувачем.

Таким чином, можна сказати, що при використанні цільової політики значущим полем контексту безпеки залишається тільки тип суб'єкта / об'єкта, тоді як поля користувач і роль просто визначають ставлення SELinux до процесу [або він знаходиться під контролем правил, або ні].

Робота з системою

Головна перевага SELinux в тому, що він абсолютно непомітний для багатьох адміністраторів і звичайних користувачів. Застосунки, для яких є правила в політиці, будуть автоматично обмежені в правах, інші програми зможуть функціонувати як ні в чому не бувало. Однак, час від часу сисадміни натикаються на деякі проблеми в роботі системи, які змушують їх відключити SELinux. У цьому розділі ми розберемося як вирішувати ці проблеми в рамках можливостей SELinux, але спочатку кілька важливих порад.

1. SELinux потрібно включити. Нерозумно відключати SELinux тільки тому, що так рекомендують робити багато сисадмінів та користувачів. Роботу штатних сервісів він порушити не може, а якщо тобі потрібно розширити можливості будь-якої програми, це завжди можна зробити за допомогою зміни мета-налаштувань або відключення перевірок для певного типу суб'єкта [пізніше буде сказано, як це зробити].

2. «-Z> Твій друг. Так, вся ця метушня з контекстами безпеки може вивести з себе. Але є багато інструментів, які дозволяють з'ясувати

поточний контекст безпеки процесів і файлів:
```
# id -Z
# ps auxZ
# ls -Z
```

Знайти файли з потрібним контекстом:
```
# find /etc -context '* net_conf_t'
```

Відновити правильний контекст файлу:
```
# restorecon -v /usr/sbin/httpd
```

І навіть дізнатися, яким повинен бути контекст файлу і порівняти його з поточним:
```
# matchpathcon -V /var/www/html/ *
```

3. Скажи mv - ні! Під час встановлення дистрибутива всі файли отримують певний контекст безпеки, а всі файли, створені в процесі роботи, - контекст безпеки, який визначається правилами на основі батьківського каталогу [наприклад, якщо створити файл в каталозі /etc, його тип автоматично стане etc_t, а для файлів каталогу /var/www/html - httpd_sys_content_t]. Однак це працює тільки в відношенні новостворених файлів. Під час переміщення файлу за допомогою mv його контекст зберігається, що може привести до відмови в доступі [наприклад, Apache не зможе отримати доступ до файлу, якщо його тип не httpd_sys_content_t].

4. Мани - наше все. У дистрибутивах Fedora і RHEL є велика кількість man-сторінок, які роз'яснюють всі обмеження SELinux щодо сервісів і демонів. Наприклад, на сторінці httpd_selinux описано, в якому контексті безпеки працює Apache, які у нього є повноваження, які контексти повинні мати доступні йому файли. Тепер розберемося з тим, що потрібно робити, коли SELinux починає заважати. Зазвичай це відбувається внаслідок того, що адмін намагається включити певну функцію сервісу або якось переналаштувати його, а SELinux починає сипати на екран попереджувальні повідомлення. Перше, що потрібно зробити в цій ситуації, це перевірити лог-файли. Головний журнальний файл SELinux носить ім'я /var/log/audit/audit.log. Туди надходять «необроблені» повідомлення, які можуть бути корисні іншим застосункам, але важкі для читання людиною. Тому існує друге місце, куди надходять ті ж повідомлення, але в набагато більш зрозумілому форматі. Це файл /var/log/messages, в ньому повідомлення можуть виглядати так:

```
# grep "SELinux is preventing" /var/log/messages
May 7 18:55:56 localhost setroubleshoot; SELinux is preventing
httpd (httpd_t) "getattr" to /var/www/html/index.html (home_dir_t).
For complete SELinux messages, run sealert -1 de7e3ed6-5488-466d-
a606-92c9f40d316d
```

Тут все повинно бути зрозуміло: SELinux заборонив суб'єкту httpd_t

[веб-сервер Apache] доступ до файлу /var/www/html/index.html на тій підставі, що останній має неправильний тип об'єкта [home_dir_t присвоюється файлам, створеним в домашньому каталозі користувача] . Для отримання більш детальної інформації SELinux рекомендує виконати команду sealert -l <бла-бла-бла>. Ця команда виведе дуже інформативне повідомлення, де будуть описані всі обставини події, причини, за якими вони могли з'явитися, а також шляхи вирішення проблеми. У нашому випадку причина проста: адмін перемістив файл index.html з свого домашнього каталогу за допомогою mv, виставив на нього потрібного власника і права, але забув змінити контекст. Виходи із ситуації два. Або надати файлу правильний контекст за допомогою команди chcon:

```
# chcon -t httpd_sys_content_t /var/www/html/index.html
```

Або змусити систему «скинути» контексти всіх файлів каталогу:

```
# restorecon -v /var/www/html
```

Після перезапуску Apache все повинно бути ок. Проте цей метод не спрацює, якщо ми захочемо перемістити кореневої каталог Apache в інше місце [наприклад, в /www]. Справа в тому, що chcon змінює контекст тільки до наступної переіндексації за допомогою restorecon, яка скине контекст файлів каталогу /www до default_t [за правилами політики всі каталоги, що не мають батьківського каталогу, і їх файли отримують такий тип]. Тому ми повинні змінити саму політику:

```
# semanage fcontext -a -t httpd_sys_content_t /www
# restorecon -v /www
```

Перша команда змінить політику так, щоб дефлтовим типом для каталогу /www і його вмісту був httpd_sys_content_t, а друга скине контекст його файлів, так що вони отримають той же тип [правила SELinux допускають, щоб дефолтовая мітки каталогів і їхнього вмісту відрізнялися, але для зручності semanage робить їх однаковими]. Також semanage може бути використана для перегляду списку мета-правил:

```
# semanage boolean -l
```

* За замовчуванням tar не зберігає контексти безпеки файлів, що може привести до проблем при подальшій розпакуванні. Параметр «--selinux» виправляє це.

За допомогою мета-правил можна контролювати такі аспекти роботи застосунків, як, наприклад, можливість веб-сервера підключатися до віддаленої базі даних [httpd_can_network_connect_db] або дозвіл ftp-сервер читати файли домашнього каталогу користувачів [ftp_home_dir | і т.д. Такі правила групують в собі велику кількість низькорівневих правил, що істотно спрощує життя сисадміна.
Щоб включити / відключити ту чи іншу правило, можна використовувати команду setsebool:

```
# setsebool httpd_can_network_connect_db on
# setsebool httpd_can_network_connect_db off
```

Вказавши прапор «-P». можна зробити ці правила постійними між перезавантаженнями. У самому крайньому випадку semanage можна використовувати для повного відключення перевірок SELinux щодо певного суб'єкта:

```
# semanage permissive -a httpd_t
```

Включення проводиться за допомогою схожої команди:

```
# semanage permissive -d httpd_t
```

Список портів і їх типів об'єктів:

```
# semanage port -l
```

SELinux далеко не такий страшний, як про нього говорять. Система хоч і складна в розумінні, але неймовірно логічна і зручна в супроводі, а наявні засоби керування дозволяють дуже точно діагностувати проблеми і легко їх усувати.

Хід роботи:

1. Активувати SELinux:

У файлі /etc/selinux/config встановити SELINUX = enforcing і (якщо раніше SELinux не запускався) перезапустити ОС. При цьому відбудеться початкове маркування контекстів.

Якщо SELinux був активований раніше, його можна включити командою:

```
#setenforce 1
```

Перевірити стан SELinux:

```
#sestatus
```

2. Під користувачем root перевірити поточні контексти користувача, процесів і файлів:

```
#id -Z
#ps auxZ
#ls -Z
```

3. Перевірка роботи процесів в обмеженому і необмеженому контексті:

Запустити web-сервер:

```
#service httpd start
```

Створити файл /var/www/html/testfile

Змінити контекст даного файлу:

```
#chcon -t samba_share_t /var/www/html/testfile
```

Перевірити, що процес httpd, який працює в контексті httpd_t, не зможе отримати доступ до цього файлу:

```
wget http://localhost/testfile
```

Змінити контекст процесу httpd на необмежений:
#chcon -t unconfined_exec_t /usr/sbin/httpd

Перезапустити httpd і переконатися, що тепер процес зможе прочитати testfile.

* Команда chcon змінює контекст тимчасово (до наступного перемаркування). Для постійної зміни контексту використовуйте команду:
#semanage fcontext -a -t samba_share_t /etc/file1

* Для відновлення контексту за замовчуванням:
#restorecon -v /usr/sbin/httpd

4. Створити двох обмежених користувачів:
#useradd -Z user_u user1
#useradd user2

Встановити зпівставлення між SELinux і Linux користувачем:
#semanage login -a -s guest_u user2

Перевірити, чи зможуть вони запускати програми в своєму домашньому каталозі (в тому числі з встановленими бітами SUID і SGID) і виконувати мережеві додатки (ping, traceroute, wget, nc).

* Щоб при створенні нових користувачів linux за замовчуванням надавався SELinux контекст user_u:
#semanage login -m -s user_u "__default__"

* Щоб дозволити процесу httpd використовувати нестандартний порт:
#semanage port -a -t httpd_port_t -p tcp 9876