Денис Колисниченко

На страже системы

В прошлом выпуске журнала рассказывалось о системах управления доступом в общем, а кроме того, была подробно рассмотрена одна из самых популярных систем — Grsecurity. В этот раз мы поговорим о не менее интересном программном продукте, который называется LIDS (Linux Intrusion Detection System).

Изначально LIDS задумывалась как простая система обнаружения вторжения, но с годами благодаря усилиям разработчиков ей удалось вырасти в комплексный механизм обеспечения безопасности Linux-машины. Сейчас можно с уверенностью заявить, что LIDS уже достигла уровня Grsecurity, а в кое в чем даже превзошла ее. Преимуществом второй версии LIDS является поддержка LSM (загружаемых модулей безопасности), которой не было в предыдущих версиях, и это обстоятельство делало установку системы и работу с ней весьма затруднительным мероприятием.

Установка

LIDS реализована как патч ядра и набор пользовательских утилит, загрузить которые можно с сайта, расположенного по адресу www.lids.org/download.html. В отличие от других проектов, патчи которых подходят только к какой-либо конкретной версии ядра, патчи LIDS подходят для всех версий 2.4.x и 2.6.x, поэтому данная система способна работать с любым ядром.

Конфигурирование ядра

Распакуйте архив с LIDS, перейдите в каталог ядра и примените следующий патч:

```
# cd /usr/src
# wget http://www.lids.org/download/v2.6/2.6.7/lids-2.2.0rc3-2.6.7.tar.gz
# tar -zxvf lids-2.2.0rc3-2.6.7.tar.gz
# cd linux-2.6.7
# patch -p1 < /usr/src/lids-2.2.0rc3-2.6.7/lids-2.2.0rc3-2.6.7.patch
```

После этого можно приступать к конфигурации ядра. LIDS требует включить поддержку алгоритма SHA256. Соответствующую опцию можно найти в меню «Cryptography Options/ Cryptography API». Опции конфигурации LIDS находятся в разделе «Security Options». Желательно отключить SELinux и Capabilities.

Рассмотрим опции LIDS:

Attempt Not to Flood Logs (CONFIG_LIDS_NO_LOOD_LOG). Данная опция ограничивает частоту протоколирования идентичных сообщений.

Allow Switching the LFS and States (CONFIG_LIDS_ALLOW_SWITCH). LFS (LIDS-free session, о ней мы поговорим чуть позже) позволяет администратору выполнять команды без каких-либо ограничений со стороны LIDS. Это довольно полезно, но может стать источником для атаки вашей системы. Рекомендуем активировать эту опцию на первые несколько дней — пока вы экспериментируете с LIDS, а потом, когда все будет настроено, можно ее спокойно отключить.

Allow Switch Off the Linux Free Session (CONFIG_LIDS_ALLOW_LFS). Позволяет отключить LIDS во время выполнения системы. Так безопаснее.

Restrict Mode Switching to Special Terminals (CONFIG_LIDS_RESTRICT_MODE_SWITCH). Позволяет задавать терминалы, с которых разрешается LFS. Они делятся на три класса: консоль (console), последовательная консоль (serial console) и PTY. Третий — наиболее опасный, поскольку дает возможность злоумышленнику удаленно запустить LFS. Выберите только первый класс, позволяющий запускать LFS лишь пользователям, физически работающим с машиной.

lidstools

После компиляции ядра можно установить пакет lidstools. Сценарию ./configure необходимо явно передать каталог ядра:

```
$ tar -zvxf lidstools-2.2.5
$ cd lidstools-2.2.5rc1.tar.gz
$ ./configure KERNEL_DIR=/usr/src/linux-2.6.7
$ make
$ su
# make install
```

При установке программы (make install) вас попросят ввести пароль для администрирования LIDS — он не должен совпадать с паролем root!

Чтобы использовать новое ядро, нужно перезагрузить систему. Если необходимо отключить LIDS, перед загрузкой системы передайте ядру параметры security=0.

Администрирование LIDS

Поскольку конфигурирование ядра мы уже рассмотрели, займемся конфигурированием пользовательского уровня. Как и Grsecurity, LIDS позволяет определить, каким образом файлы и процессы будут взаимодействовать между собой в системе. Кроме этого LIDS имеет две очень полезные функции — LFS и «опечатывание» ядра (sealing the kernel).

Загружаемые модули очень полезны, так как позволяют добавлять код во время выполнения ядра без его перекомпиляции. С другой стороны, злоумышленник может добавить в ядро свои собственные модули, а для нас это очень нежелательно. Конечно, самое лучшее — включить весь необходимый код в состав ядра и вообще отключить поддержку загружаемых модулей, но вряд ли это решение окажется для вас удобным.

LIDS предлагает концепцию «опечатывания» ядра. Если ядро «опечатано», никто не может загрузить или выгрузить модуль. Сделать это можно с помощью команды lidsadm -I. Ее необходимо поместить в сценарии загрузки системы, только при этом убедитесь, чтобы еще до начала выполнения этой команды все необходимые модули были уже загружены. «Опечатывание» ядра также предусматривает некоторые ограничения набора возможностей, но об этом мы поговорим чуть позже.

LFS (LIDS-free Sessions) — сессии без LIDS

LFS — это всего лишь оболочка, на выполнение команд которой не накладываются ограничения LIDS, что позволяет администратору работать в системе как обычно, без выключения основной системы безопасности, ведь когда LIDS включена, ограничения накладываются даже на пользователя root. Однако LFS потенциально опасна: если злоумышленнику удастся завладеть доступом к ней, то он получит полный контроль над системой — вплоть до отключения LIDS.

Доступ к LFS регулируется установленным ранее паролем. Дополнительно при конфигурации ядра можно указать терминалы, с которых доступ к LFS будет открыт (очень хорошее решение).

Главное назначение LFS — разрешить администратору редактировать файлы в каталоге /etc/lids, который недоступен во время работы LIDS даже пользователю root. В этом каталоге находятся следующие файлы:

- lids.cap — ограниченный набор возможностей;
- lids.conf — ACL (будет рассмотрен позже);
- lids.pw — пароль администратора LIDS;
- lids.ini — начальные конфигурационные значения.

lidsadm

Администрирование LIDS выполняется программой lidsadm. Рассмотрим подробнее ее опции:

- -P — зашифровать пароль LIDS, например lidsadm -P mypassword;
- -S — изменить аспект защиты LIDS;
- -I — «опечатать» ядро; для этой опции не нужен пароль;
- -V — просмотр состояния системы;
- -h — вывести краткую справку;
- -v — вывести версию lidsadm.

Опция -S используется вместе с одним из следующих флагов, предваряемых либо знаком «+» — включить, либо «-» — выключить:

- LIDS_GLOBAL — включить/выключить LIDS глобально;
- RELOAD_CONF — перезагрузить файл lids.conf и обновить список защищенных инодов (об этом мы поговорим чуть позже);
- LIDS — включить/выключить LIDS локально, то есть создать LFS, которая будет применена только к текущей оболочке;
- ACL_DISCOVERY — используется для отладки; когда включена, нарушения правил не запрещаются;
- SHUTDOWN — переключается в состояние shutwodn.

Например, чтобы войти в LFS, выполните нижеприведенную команду:

```
# lidsadm -S — -LIDS
```

Чтобы выключить защиту LIDS, проделайте следующее:

```
# lidsadm -S — -LIDS_GLOBAL
```

ACL файлов и возможностей

ACL используется для управления доступом к различным объектам, например файлам или ресурсам (но в отличие от Grsecurity в LIDS нет ролей). В системе LIDS есть ACL двух типов: ACL файлов (контролирует доступ к файлам и каталогам) и ACL возможностей (регулирует возможности исполнимых файлов).

ACL файлов

LIDS определяет четыре режима для объектов:

- **DENY**. Доступ к файлу запрещен. При обращении к нему

приложения (например, ls или cat) будет получено сообщение об ошибке «No such file or directory (ENOET)» — «Нет такого файла или каталога». Внешне это будет выглядеть так, как будто файла вообще не существует.

- **READ**. Объект может быть открыт в режиме «только для чтения», запись запрещена.
- **APPEND**. Объект может быть открыт для чтения или добавления информации. Данный режим удобно использовать для файлов журналов.
- **WRITE**. Чтение и запись не ограничены. LIDS не защищает этот файл.

Данные режимы могут быть заданы как для файла, так и для каталога. Если режим используется для каталога, то он будет примерен ко всем файлам, входящим в этот каталог.

В LIDS ACL описывается следующим образом:

<Тип ACL> <субъект> <объект> <доступ> <наследование>

Тип ACL определяет, на какой стадии работы будет контролироваться доступ к системе. Доступно четыре типа:

- BOOT. Доступ будет контролироваться на стадии загрузки.
- POSTBOOT. После загрузки.
- SHUTDOWN. При разгрузке системы.
- null. Контроль доступа вне зависимости от стадии работы.

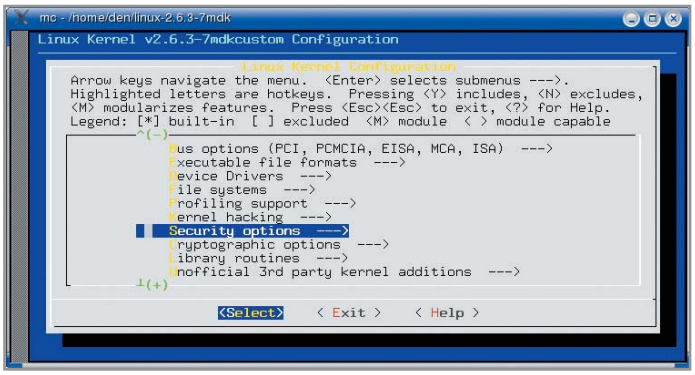
Обычно тип ACL просто не указывается (null) — это делается для постоянного контроля, а первые три типа используются для ослабления определенных ограничений.

Субъект — это приложение, которому предоставляется доступ (режимы доступа описаны выше) к объекту — файлу или каталогу. Последнее поле, <наследование>, определяет, будет ли ACL наследоваться дочерними процессами или нет, и может принимать значения 0, 1 и -1.

Объект и субъект тесно связаны друг с другом. Например, субъекту /usr/bin/sshd (демону SSH) требуется доступ к объекту /var/run/sshd.pid. Вы определяете доступ, например WRITE. Но вы должны понимать, что данное правило определяет доступ только данного субъекта к конкретному объекту, оно никак не относится к другим субъектам. Если же поле <субъект> пусто (то есть субъект не указан), то данный режим доступа применяется к объекту для любого субъекта (то есть любого процесса).

Многие приложения (особенно shell-сценарии) во время выполнения вызывают другие программы, и это обязательно нужно учитывать при разработке ACL. По умолчанию наследование выключено, и дочерние процессы не наследуют режимы доступа, определенные для родительского процесса. Такой режим работы может стать настоящей головной болью для администратора — ему придется устанавливать режим доступа для каждого процесса, вызываемого сценарием. Наследование можно включить, установив значение 1 в поле <наследование>: это так называемый нерекурсивный режим наследования, то есть режим доступа будет унаследован только дочерним процессом. Если же вы хотите, чтобы режим доступа был унаследован не только дочерним процессом, но и всеми его потомками, установите рекурсивный режим — значение -1 для поля <наследование>.

ACL хранятся в файле /etc/lids/lisd.conf. Открыв его, вы увидите, что правила записаны в непонятном для человека виде. Но вам не нужно редактировать этот файл вручную — для



этого существует утилита lidsconf. Но прежде чем мы перейдем к ее рассмотрению, все-таки необходимо понять, что означают все эти числа:

```
0:0::1:0:1114128:834:/sbin:0-0
0:0::1:0:1933326:834:/bin:0-0
...
```

ACL системы LIDS содержит номера инодов вместо имен файлов — например, если мы установим режим APPEND для /var/log/messages, LIDS сохранит в ACL номер инода этого файла. Это позволит более жестко контролировать доступ к файлу, ведь имя ничего не означает, главное — инод: можно удалить файл с именем /var/log/messages и заново создать файл с тем же именем. При этом имя не изменится, а инод — да.

lidsconf

Начиная со второй версии для администрирования и конфигурирования LIDS используются две разные утилиты. Утилиту администрирования — lidsadm — мы уже рассмотрели. Осталось исследовать утилиту конфигурирования — lidsconf. Ниже приведены ее наиболее важные опции:

- **-A, --add**. Добавить запись.
- **-C,--check**. Проверить существующие записи.
- **-D, --delete**. Удалить запись.
- **-Z, --zero**. Удалить все записи.
- **-U, --update**. Обновить /dev и номера инодов.
- **-L, --list**. Вывести все записи.

Следующий пример показывает ACL по умолчанию, который поставляется вместе lidstools-2.2.5rc1 (для удобства мы добавили в листинг номера строк):

```
# lidsconf -L
      Subject    ACCESS      inherit      Object
---------------
1)    Any file   READONLY:    0            /sbin
2)    Any file   READONLY:    0            /bin
3)    Any file   READONLY:    0            /boot
4)    Any file   READONLY:    0            /lib
5)    Any file   READONLY:    0            /usr
6)    Any file   READONLY:    0            /etc
7)    Any file   DENY:        0            /etc/lids
8)    Any file   DENY:        0            /etc/shadow
9)    Any file   APPEND:      0            /var/log
10)   Any file   WRITE:       0            /var/log/wtmp
11)   /bin/login READONLY:    0            /etc/shadow
12)   /bin/login GRANT:       0            CAP_SETUID
```

```
13      )/usr/sbin/sshd GRANT:  0       CAP_NET_ADMIN
14)     /bin/login GRANT:       0       CAP_GETID
```

Если несколько правил применимы к одному объекту, будет использовано самое последнее из них. Лучше всего это можно продемонстрировать на примере каталога /etc, который указан в правилах 6, 7, 8 и 11. Сначала /etc объявляется как READONLY (только чтение), затем объекты /etc/lids и /etc/shadow делаются невидимыми (DENY). Поскольку субъект не указан (Any file — «любой файл»), то ограничение применяется к любому процессу. А в строке 11 READONLY доступ к файлу /etc/shadow предоставляется только субъекту /bin/login.

Для добавления новых записей используется опция -A программы lidsconf:

```
lidsconf -A [тип ACL] [-s субъект] -o объект [-t с-по]
[-i уровень] -j действие
```

Как видите, обязательными в этом случае являются только опции A, o и j:

```
lidsconf -A -o /etc/hosts.conf -j READ
```

Все опции нам понятны, кроме разве что [-t с-по]. Она позволяет установить время действия правила, которое обычно указывается в формате ЧЧММ-ЧЧММ. Например, чтобы правило действовало с 8:00 по 19:35, используйте следующую опцию:

```
-t 0800-1935.
```

Для удаления записи используется синтаксис:

```
lidsconf -D [тип ACL] [-s субъект] [-o объект]
```

Мы можем указать субъект и/или объект — будут удалены все совпадающие правила. Можно также обозначить и тип ACL, например POSTBOOT.

Возможности

В предыдущем ACL, наверное, вы заметили действие GRANT, предоставляющее возможность CAP_SETUID, а именно их мы до этого и не рассматривали, поэтому самое время приступить.

LIDS предоставляет расширенное использование возможностей (модуль возможностей capability security module должен быть отключен в ядре). Кроме стандартных возможностей Linux система LIDS предлагает две собственные:

- **CAP_HIDDEN**. Процессы с данной установленной возможностью не будут отображаться в /proc, что позволит скрыть процесс от программ ps, lsof и top;
- **CAP_INIT_KILL**. Если эта возможность выключена для демона, то он не будет получать KILL-сигналы.

CAP_HIDDEN не гарантирует, что процесс будет полностью невидимым: например, сетевой демон можно обнаружить с помощью netstat или сканера портов, а также по наличию файла /var/run/название.pid.

А теперь перейдем к CAP_INIT_KILL. Рассмотрим дерево процессов с помощью pstree:

```
$ pstree -a
init)
  |-atd)
  |-(bdflush)
  |-crond)
  |-httpd)
  |  |-httpd)
  |  |-httpd)
  |  |-httpd)
  |-(keventd)
  |-(khubd)
  |-(kjournald)
  |-klogd) -x
```

В случае с CAP_INIT_KILL демон определяется как процесс, исходящий непосредственно от Init (PID Init всегда равен 1). К сожалению, это недостаток. Поскольку демон не может получать сигналы, администратор не сумеет ни остановить (SIGKILL), ни перезагрузить процесс (SIGHUP). Ко всему прочему, некоторые процессы, например Apache, которые «общаются» со своими «родственниками» с помощью сигналов, не смогут нормально работать. Если же вы хотите использовать CAP_INIT_KILL, первая проблема может быть решена с помощью LFS — отсюда разрешено отправлять процессам сигналы. А вот для решения второй проблемы вы должны включить CAP_INIT_KILL для определенного процесса, например Apache. По-другому тут никак нельзя.

LIDS немного модифицирует возможность CAP_BIND_NET_SERVICE. Обычно она включается для процесса, которому нужно «привязаться» к привилегированному порту (с номером 0-1024). Но LIDS расширяет ее синтаксис, позволяя указывать порт или диапазон портов, к которым разрешена привязка процесса. Например, для Apache раньше разрешалась привязка к любому привилегированному порту, а теперь мы можем четко указать, к каким именно портам разрешается привязываться этому сервису — 80 и 443.

Ограниченный набор возможностей

Это список возможностей, доступных (но необязательно установленных) процессу в системе. Если какая-то из них не значится в данном списке, ее нельзя назначать процессу. Соответственно, для каждого процесса можно определить свои списки возможностей.

Конфигурация LIDS по умолчанию (/etc/lids.cap) разрешает все возможности, за исключением следующих:

- **CAP_SETPCAP**. Возможность устанавливать возможности другого процесса.
- **CAP_SYS_MODULE**. Возможность загружать и выгружать модули ядра.
- **CAP_SYS_RAWIO**. Возможность прямого ввода/вывода, то есть доступа к файлам /dev/port, /dev/mem, /dev/kmem, а также прямого доступа к дискам (например, /de/hda).
- **CAP_KILL_PROTECTED**. Возможность «убивать» защищенные процессы.

Помните, что X Window требует возможности CAP_SYS_RAWIO. Если вам нужна эта система, установите данную возможность для исполнимого файла X.

Установка и модификация возможностей

Установка возможностей производится с помощью все той же утилиты lidsconf. Синтаксис добавления возможности — тот

же, что и в случае добавления правила для файла, только для разрешения этой возможности используется действие GRANT. Добавим возможность запуска X Window:

lidsconf -A -s /usr/X11/bin/X -o CAP_SYS_RAWIO -j GRANT

В этом случае нужно конкретно указывать субъект правила — процесс, которому разрешается та или иная возможность. В вышеприведенном примере мы предоставляем возможность прямого ввода/вывода CAP_SYS_RAWIO процессу /usr/X11/bin/X.

А теперь разрешим веб-серверу Apache привязываться к непривилегированному порту:

lidsconf -A -s /usr/sbin/httpd -o CAP_BIND_NET_SERVICE -j GRANT

Более безопасным будет вариант разрешения привязки Apache к портам 80 и 443. Синтаксис возможности не предусматривает задание единичных портов, а только диапазонов, поэтому мы будем использовать нулевые диапазоны 80-80 и 443-443:

lidsconf -A -s /usr/sbin/httpd -o CAP_BIND_NET_SERVICE 80-80,443-443 -j GRANT

Также рекомендуем установить для Apache возможность CAP_INIT_KILL, чтобы он мог «общаться» со своими потомками (но Apache — это только пример, не забывайте и о других сервисах, которые используют подобную форму IPC!):

lidsconf -A -s /usr/sbin/apache -o CAP_INIT_KILL -j GRANT

Реализация LIDS

Теперь, когда мы знаем, как работает LIDS, и знакомы с ее основными опциями, можно приступать к ее практической реализации в нашей системе.

Помните, что разработка ACL для всей системы — очень трудная и объемная задача, поэтому мы рекомендуем вам создать shell-сценарий, содержащий вызовы lidsconf. Первой командой будет являться lidsconf -Z — эта директива удаляет все ранее существующие ACL. Данный сценарий нужно поместить в /etc/lids – это сделает его не видимым за пределами LFS-сессии.

Защита системных программ

Какие же системные файлы и каталоги требуют защиты LIDS? Защищать нужно содержимое /bin, /sbin, /lib, /usr/bin, /usr/lib и /usr/sbin. В эти каталоги производится установка программного обеспечения, и, если мы сделаем их доступными только для чтения (READONLY), злоумышленник не сможет записывать троянские версии системных программ. Также не забудьте защитить каталоги /usr/local/bin и /usr/local/lib, если придерживаетесь стратегии установки программ в /usr/local. Конечно, при установке новой программы защита этих каталогов может создать некоторые неудобства для администратора, но поверьте, это стоит того.

/etc и /etc/shadow

Конфигурационные файлы — это другая область защиты. Подавляющее большинство файлов в этом каталоге требует доступ «только для чтения», поэтому можно сначала установить для всего каталога режим READONLY, а затем разрешить доступ в режиме WRITE к каким-то отдельным файлам — глобально или для определенных субъектов.

Наиболее важными файлами в каталоге /etc являются passwd/passwd- и shadow/shadow-: вы должны запретить всем субъектам доступ к shadow/shadow- (DENY), за исключением субъектов /bin/login, su и /usr/sbin/sshd — им полагается доступ READONLY.

Но мы еще не учли утилиту /usr/bin/passwd. Ей нужен WRITE-доступ к файлу /etc/shadow, чтобы пользователь имел возможность изменить свой пароль. Когда он сделает это, файл /etc/shadow будет создан заново, а не просто модифицируется. В результате изменится инод файла, а поскольку LIDS привязывается к инодам, то после его изменения она уже не будет защищать файл — ведь нового инода в «базе данных» LIDS не существует. Кроме того, нужно предоставить программе passwd WRITE-доступ ко всему каталогу /etc, поскольку запись файла — это изменение каталога. Если на месте passwd будет эксплоит злоумышленника, он сможет получить доступ ко всем файлам в каталоге /etc.

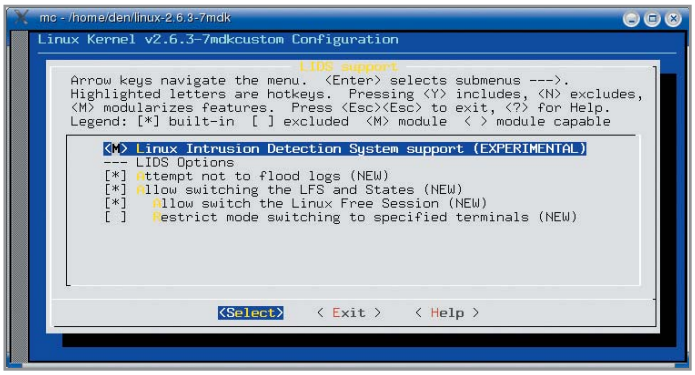
К сожалению, не существует простого способа решения этой проблемы: вы или должны использовать альтернативную схему аутентификации, например LDAP, чтобы запретить пользователям изменять свои пароли, или открыть WRITE-доступ к /etc. Существует, правда, еще один способ, самый безопасный, но очень неудобный для администратора. Вы запрещаете WRITE-доступ к /etc, а для того чтобы изменить свой пароль, пользователь должен будет обратиться непосредственно к вам. Вы сможете изменить его пароль в LFS-сессии, а заодно и проверите его на «стойкость». Такой вариант приемлем, если пользователей у вас немного — до 10 человек. Учитывая то, что пароли они меняют не слишком часто, раздражать это вас особо не будет. А вот если пользователей 200...

Если обеспечение полной защиты каталога /etc для вас слишком сложно, нужно обеспечить хотя бы защиту ключе-

Предыстория технологии

Модули защиты (LSM)

Все начиналось с разработки различных систем управления доступом, которые реализовывались в виде патчей ядра Linux. Но в результате из-за отсутствия централизованной координации получилось, что каждый проект защиты имел свой патч для ядра, часто несовместимый с другими патчами. В прошлом Линус Торвальдс категорически отвергал все эти патчи, но в 2001 году в ответ на технологию SELinux, представленную NSA (Агентство национальной безопасности) на Linux Kernel Summit, он заявил, что для включения в ядро будет рассматриваться более обширная система безопасности. Так появился проект LSM (Linux Security Modules). Его цель — предоставить разработчикам систем защиты общий (единый) интерфейс для реализации проектов, основанных на ядре. Это позволит уменьшить зависимость от ядра, кроме того, не нужно будет его перекомпилировать, как в случае с патчами. Все LSM-модули используют стандартный интерфейс для взаимодействия с ядром, который не будет изменяться с выходом следующего релиза (не версии) ядра.



вых файлов. Наиболее важными являются каталоги /etc/rc.d и /etc/rc.d/init.d.

Теперь перейдем к файлу /etc/lids/lids.cap. Как уже было отмечено, следующие возможности должны быть запрещены:

- **CAP_SYS_RAWIO**. Возможность прямого ввода/вывода, то есть доступа к файлам /dev/port, /dev/mem, /dev/kmem, а также прямого доступа к дискам (например, /de/hda).
- **CAP_SYS_PTRACE**. Возможность трассировки системных вызовов, производимых процессом.
- **CAP_SETPCAP**. Возможность устанавливать возможности другого процесса.
- **CAP_KILL_PROTECTED**. Возможность «убивать» защищенные процессы.
- **CAP_SYS_MODULE**. Возможность загружать и выгружать модули ядра.

Единственное приложение, которое требует первую возможность, — это X11. Для всех остальных приложений все эти возможности должны быть отключены.

Определение требуемого доступа

Как определить, к каким файлам и каталогам нужно обращаться тому или иному приложению? Отслеживать системные вызовы open(), chdir(), mkdir() и другие — дело неблагодарное, но вам все же нужно будет через это пройти. Немного облегчить задачу позволяет LIDS-FAQ, расположенный по адресу www.lids.org/fids-faaq/lids-faq.html. Там вы найдете ACL для таких приложений как login, su, MySQL, BIND, OpenSSH, Apache и других.

При самостоятельной разработке ACL для приложения, которое вы не встретите в LIDS-FAQ, последовательность действий приблизительно следующая:

- Проверьте, к каким файлам и каталогам необходим доступ для этого приложения. Это можно сделать с помощью ptrace. Какие файлы конфигурации использует программа? Будет ли она записывать что-то в /var/run?
- Нужно ли приложению привязываться к привилегированному порту? Если да, то для него нужно включить возможность CAP_BIND_NET_SERVICE, указав необходимые порты.

Проверить ACL просто: запустите приложение при включенной защите LIDS. Если что-то пойдет не так, в /var/log/messages появится соответствующее сообщение LIDS.

Пример ACL для DNS-сервера

Давайте рассмотрим пример ACL, позволяющий DNS-серверу работать на нашей машине. Этот ACL будет состоять из двух частей. Первая часть содержит общий набор правил, предоставляющих базовый доступ, — она подойдет не только для DNS-сервера, но и других приложений, запущенных в системе Linux. Во второй части будут описаны специфические для DNS-сервера правила, ограничивающие доступ к файлам, а также определяющие его возможности. Как было отмечено ранее, набор правил мы представим в виде shell-сценария, содержащего серию команд lidsconf.

Сперва нужно сделать системные исполнимые файлы и библиотеки доступными только для чтения. Обойти это ограничение можно только в LFS-оболочке. Итак, нам нужно защитить каталоги /bin, /sbin, /usr и /opt:

```
/sbin/lidsconf -A -o /bin -j READONLY
/sbin/lidsconf -A -o /sbin -j READONLY
/sbin/lidsconf -A -o /usr -j READONLY
/sbin/lidsconf -A -o /opt -j READONLY
```

Мы не определили субъект, поэтому указанные объекты будут доступны только для чтения всем процессам в системе. Не забывайте, что ACL наследуется, то есть доступ READONLY получат также и все подкаталоги данных каталогов. Также следует помнить, что наследование не распространяется на подмонтированные файловые системы. Например, если к /usr/local подмонтирован другой раздел, то правила, примененные к /usr, не будут распространяться на файлы и каталоги, находящиеся на другом разделе.

Кроме этих каталогов нам нужно защитить также /etc и /boot. Однако, как было отмечено ранее, предоставление доступа READONLY к каталогу /etc довольно проблематично, поэтому мы сконцентрируемся на защите его ключевых файлов, ведь наибольший интерес для злоумышленника представляют именно они. Например, /etc/exports определяет экспортируемые файловые системы, а изменив файл /etc/resolv.conf, он может перенаправить запросы нашего резолвера на свой DNS-сервер, который будет выдавать неправильную информацию.

```
/sbin/lidsconf -A -o /boot                  -j READONLY
/sbin/lidsconf -A -o /etc/HOSTNAME          -j READONLY
/sbin/lidsconf -A -o /etc/apache            -j READONLY
/sbin/lidsconf -A -o /cron.daily            -j READONLY
/sbin/lidsconf -A -o /cron.hourly           -j READONLY
/sbin/lidsconf -A -o /cron.weekly           -j READONLY
/sbin/lidsconf -A -o /exports               -j READONLY
/sbin/lidsconf -A -o /hosts                 -j READONLY
/sbin/lidsconf -A -o /hosts.allow           -j READONLY
/sbin/lidsconf -A -o /hosts.deny            -j READONLY
/sbin/lidsconf -A -o /hosts.equiv           -j READONLY
/sbin/lidsconf -A -o /identd.conf           -j READONLY
/sbin/lidsconf -A -o /ld.so.conf            -j READONLY
/sbin/lidsconf -A -o /login.access          -j READONLY
/sbin/lidsconf -A -o /login.defs            -j READONLY
/sbin/lidsconf -A -o /logrotate.conf        -j READONLY
/sbin/lidsconf -A -o /mail                  -j READONLY
/sbin/lidsconf -A -o /modules.conf          -j READONLY
/sbin/lidsconf -A -o /named.conf            -j READONLY
/sbin/lidsconf -A -o /networks              -j READONLY
/sbin/lidsconf -A -o /ntp.conf              -j READONLY
```

```
/sbin/lidsconf -A -o /resolv.conf          -j READONLY
/sbin/lidsconf -A -o /rc.d                -j READONLY
/sbin/lidsconf -A -o /services            -j READONLY
/sbin/lidsconf -A -o /shells              -j READONLY
/sbin/lidsconf -A -o /ssh                 -j READONLY
/sbin/lidsconf -A -o /sudoers             -j READONLY
/sbin/lidsconf -A -o /sudoers.conf        -j READONLY
/sbin/lidsconf -A -o /etc/                -j READONLY
```

В зависимости от установленных в системе пакетов, возможно, придется добавить и другие файлы в этот список. Мы же описали наиболее критичные из них.

Также не нужно забывать про файлы журналов, которые находятся в каталоге /var/log. Для большинства из них можно установить режим APPEND, для некоторых — WRITE, но только для субъектов login, init и halt возможны лишь следующие режимы:

```
/sbin/lidsconf -A -o /var/log                -j APPEND
/sbin/lidsconf -A -s /bin/login -o /var/log/wtmp -j WRITE
/sbin/lidsconf -A -s /bin/login -o /var/log/lastlog -j WRITE
/sbin/lidsconf -A -s /sbin/init -o /var/log/wtmp -j WRITE
/sbin/lidsconf -A -s /sbin/init -o /var/log/lastlog -j WRITE
/sbin/lidsconf -A -s /sbin/halt -o /var/log/wtmp -j WRITE
/sbin/lidsconf -A -s /sbin/halt -o /var/log/lastlog -j WRITE
```

Благодаря этим ограничениям злоумышленник, получивший root-доступ, не будет способен отредактировать эти файлы, чтобы скрыть свое присутствие. Недостаток этого метода заключается в том, что утилита logrotate не сможет больше функционировать, но всему есть своя цена. Теперь за «уборку» журналов отвечаете лично вы — администратор. Чтобы logrotate работала, ей нужно предоставить WRITE-доступ ко всему каталогу /var/log, но мы не рекомендуем этого делать, поскольку злоумышленник может начать запускать logrotate много раз подряд — до тех пор, пока из журнала не будут удалены следы его присутствия. Отключите logrotate и «почистите» журналы вручную. Во многих системах logrotate вызывается демоном cron — чтобы этого не происходило, нужно удалить файл /etc/cron.daily/logrotate или закомментировать его содержимое.

Теперь вернемся к правилам. Нам нужно определить, как named взаимодействует с системой; мы должны предугадать все файлы, доступ к которым понадобится приложению, а также определить возможности этого самого приложения. Сперва запретим доступ ко всей файловой системе:

```
/sbin/lidsconf -A -s /usr/sbin/named -o /          -j DENY
```

BIND должен получить доступ к файлу конфигурации (/etc/named.conf), а также файлам зоны (каталог /var/named):

```
/sbin/lidsconf -A -s /usr/sbin/named -o /etc/named.conf -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /var/named      -j READ
```

Как и любой другой демон, named записывает свой PID в файл, расположенный в каталоге /var/run. Обычно он называется /var/run/named.pid. Мы должны разрешить приложению создавать файл с таким именем:

```
/sbin/lidsconf -A -s /usr/sbin/named -o /var/run/named.pid -j WRITE
```

На данный момент мы позаботились обо всех файлах, которые нужны демону named. Теперь нужно определить, какие ему понадобятся библиотеки. Для этого мы будем использовать программу strace, которая отобразит все системные вызовы, а вместе с ними и внешние файлы. Опция -f не позволяет приложению перейти в фон:

```
# strace -f -o named_trace named
```

Вывод программы named будет записан в файл named_trace. Демон named должен поработать несколько часов, после этого завершите процесс (named, а не strace!) и проанализируйте файл named_trace:

```
# cat named_trace | grep open
```

Данная команда выведет все вызовы open() — вы увидите не только файлы, задействованные приложением, но и режимы, в которых они используются. На основании этого списка мы можем составить следующий список правил LIDS:

```
/sbin/lidsconf -A -s /usr/sbin/named -o /                  -j DENY
/sbin/lidsconf -A -s /usr/sbin/named -o /usr/lib           -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o / lib              -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /usr/share/locale -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /etc/ld.so.preload -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /etc/ld.so.cache -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /etc/localtime   -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /etc/rndc.key    -j READ
/sbin/lidsconf -A -s /usr/sbin/named -o /var/log       -j APPEND
/sbin/lidsconf -A -s /usr/sbin/named -o /dev/random      -j READ
```

Данный перечень несколько упрощен, поскольку named использует много библиотек в /usr/lib и /lib, но проще предоставить READ-доступ к этим каталогам, чем прописывать отдельно каждую библиотеку.

Теперь пришла очередь установки возможностей. Прежде всего, разрешим named привязываться к порту 53:

```
/sbin/lidsconf -s /usr/sbin/named -o CAP_NET_BIND_SER-
VICE 53-53       -j GRANT
```

При запуске named с опцией -u <имя пользователя> он снижает свои привилегии до уровня обычного пользователя, указанного с помощью опции u. Поэтому нужно разрешить ему производить вызовы SUID и SGID:

```
/sbin/lidsconf -s /usr/sbin/named -o CAP_SETUID 53-53
        -j GRANT
/sbin/lidsconf -s /usr/sbin/named -o CAP_SETGID 53-53
        -j GRANT
```

Если BIND запускается в chroot-окружении, нужно установить возможность CAP_SYS_CHROOT:

```
/sbin/lidsconf -s /usr/sbin/named -o CAP_SYS_CHROOT
        -j GRANT
```

Как и Apache, BIND может иметь несколько потомков, которым понадобится доступ ко всем файлам и возможностям, описанным ранее, поэтому для переноса возможностей другим процессам нужно разрешить CAP_SYETPCAP:

```
/sbin/lidsconf -s /usr/sbin/named -o CAP_SYETPCAP
        -j GRANT
```

Настало время протестировать созданный ACL. Запустите named и следите за системными журналами — в них вы найдете сообщения об ошибках, если что-то вдруг пойдет не так.

В завершение нужно сказать, что если у вас возникли проблемы с тем или иным сервисом, посетите сайт www.lids.org — там вы найдете множество готовых ACL для разных сервисов. |