Защита информации и безопасность в компьютерных сетях

УДК 681.322:621.382.26:004.056.52

Адаптация системы работы с файлами устройств для SELinux

Д. С. Кулябов, А. В. Королькова

Кафедра систем телекоммуникаций
Российский университет дружбы народов
Россия, 117198, Москва, ул. Миклухо-Маклая, 6

В данной статье рассматриваются подходы к работе с файлами устройств в ОС Linux, описывается структура системы udev, приводится подробное описание разработанных политик `udev` для SELinux.

Ключевые слова: SELinux, udev, политика, контекст безопасности, домен, тип, роль.

Введение

Ядро 2.6 ОС Linux требует для функционирования устройств установки системы udev [1, 2]. В стандартной реализации SELinux имеется только базовая политика для системы `udev`, которая на практике оказывается неработоспособной. Поэтому были разработаны контекст безопасности и политика `udev` для работы данной системы в дистрибутиве Gentoo с поддержкой SELinux. Разработка политики для udev была осложнена тем, что udev начинает функционировать до запуска основной системы, вследствие чего было довольно сложно отследить его работу в реальном масштабе времени.

Статья имеет следующую структуру.

В разделе 1 даны основные понятия и определения, имеющие отношение к SELinux [3, 4].

В разделе 2 рассмотрены подходы к работе с файлами устройств в операционной системе Linux [5, 6], описана структура системы udev [1, 2].

В разделе 3 приведена структура конфигурационных файлов политик SELinux, приведено описание некоторых типов, используемых в разработке политик для системы udev [7–10].

В разделе 4 приведено подробное описание разработанных файла контекста безопасности `udev.fc` и файла политики `udev.te`, необходимых при адаптации udev для SELinux [11, 12].

1. Терминология

Определение 1. *Контекстом безопасности* называется набор всех атрибутов, связанных с объектами (файлами, каталогами, процессами, TCP-сокетами и т.п.). Контекст безопасности состоит из сущности, роли и домена (или типа).

Определение 2. *Сущность пользователя* определяет, какие роли и домены могут быть использованы[1] .

[1]Сущность в SELinux не является т.н. традиционным идентификатором пользователя (Unix uid, user id). Они могут сосуществовать в одной системе, но их смысл совершенно разный. Сущность в SELinux

Определение 3. *Политикой* называется набор правил, определяющих список ролей, к которым имеет доступ пользователь, списки доменов, к которым имеют доступ роли, и списки типов, к которым имеют доступ домены.

2. Подходы к работе с файлами устройств в ОС Linux

Системная файловая система `sysfs` [5] является визуальным представлением дерева устройств (т.е. дерево устройств представлено так, как видит его ядро) и монтируется в каталог `/sys`. Файловая система `sysfs` дополняет файловую систему `proc` [6], предназначенную для процессов, и файловую систему `devfs`, предназначенную для устройств.

Ядро получает информацию о дереве устройств непосредственно от подсистемы объектов ядра: когда создается объект ядра, то также создается файл или каталог в `/sys` (но возможен и явный отказ объекта ядра от создания записи в `sysfs`). После того как каждое устройство в системе получает уникальную структуру каталогов, происходит экспорт известных атрибутов устройства (имя устройства, режим питания, прерывания и т.п.) в дерево таким образом, чтобы эти атрибуты были понятны администратору. Это позволяет перенести аппаратно-зависимые случаи использования `/proc/sys` в `/sys`. Так, например, демон `udev` [1, 2, 11, 12] считывает информацию из каталога `/sys` и, если в нем появляется новое устройство, то `udev` создает такое же устройство в `/dev`, что позволяет динамически работать с периодически отключаемыми устройствами.

Пакет `udev` имеет следующую структуру:

— `udev`
  создает ноды устройств в каталоге `/dev` или переименовывает сетевые интерфейсы, отвечая на запросы `hotplug`;
— `udevd`
  представляет собой демон, упорядывающий запросы `hotplug` перед передачей их `udev`;
— `udevsend`
  доставляет запросы `hotplug` к `udevd`;
— `udevstart`
  создает в директории `/dev` ноды устройств, отвечающие драйверам, вкомпилированным в ядро ОС (т.е. происходит симуляция событий `hotplug` при неподмонтированной корневой файловой системе);
— `udevinfo`
  позволяет пользователям запрашивать базу данных `udev` для получения информации о каждом устройстве системы и предоставляет путь для запроса любого устройства в дереве `sysfs`, что необходимо для создания правил `udev`;
— `udevtest`
  симулирует запуск `udev` для выбранного устройства и выдает имя реальной ноды, созданной `udev` или имя переименованного сетевого интерфейса;
— `/etc/udev`
  содержит конфигурационные файлы `udev`, права доступа к устройствам и правила для именования устройств.

3. Структура конфигурационных файлов SELinux

3.1. Файлы политики SELinux

Политика SELinux [4, 7–9] расположена в директории `/etc/security/selinux/src/policy` и состоит из нескольких файлов и директорий для определения политики. Для облегчения создания политики используется макропроцессор `m4`. Файлы политики обрабатываются `m4`, затем при помощи компилятора `checkpolicy` политика проверяется на отсутствие синтаксических

формирует часть контекста безопасности, задающего домены, в которые можно войти, и может иметь одинаковое с именем пользователя символьное представление.

ошибок и создается бинарный файл политики, который затем загружается в ядро SELinux.

`Makefile` предназначен для компиляции и установки политики [10]. Описание команд:

— `make policy`
  компилирует текущую политику в бинарный файл;
— `make install`
  компилирует и устанавливает текущую политику в директорию `/etc/security/selinux/src`;
— `make load`
  компилирует, инсталирует и запускает текущую политику в работающее в данное время ядро;
— `make initrd`
  компилирует и инсталирует политику и создает `ramdisk` для первоначальной загрузки политики;
— `make relabel`
  перемаркировывает файловую систему, используя контекстный файл политики.

3.2. Описание других файлов и директорий

Здесь приводится описание некоторых файлов и директорий, расположенных в `/etc/security/selinux/src/policy`.

— `assert.te`
  Правила, содержащиеся в этом файле, проверяются на финальной фазе компиляции. Если права доступа в соответствующей политике содержат одно из утверждений данного файла, то такая скомпилированная политика признается непригодной к использованию.
— `attrib.te`
  В данном файле явным образом задаются атрибуты, которые могут быть ассоциированы с конкретным типом при его определении. Имена атрибутов могут быть затем использованы во всей конфигурации для быстрой установки типов и ассоциированных с ними атрибутов. Атрибут может быть использован для идентификации набора типов со своими свойствами. Каждый тип может иметь любое количество атрибутов и каждый атрибут может быть ассоциирован с любым количеством типов.
— `constraints`
  Данный файл определяет дополнительные условия на форму выражений, которые должны удовлетворять предоставляемым правам доступа. Эти ограничения используются для уточнения таблиц прав доступа и ролей. Обычно эти условия используются для усиления идентификатора пользователя или роли конкретного домена.
— `domains/`
  Данная директория содержит типы принудительных правил. В ней находятся файлы `admin.te`, `staff.te` и `user.te`, которые содержат векторы доступа к ролям `sysadm_r`, `staff_r` и `user_r`. Директория `program/` состоит из всех активных политик для различных программ и демонов. Директория `misc/` содержит все остальные файлы политики.
— `file_contexts/`
  Данная директория содержит те файлы, которые описывают контекст безопасности для всех файлов ОС. Она содержит файл `types.fc`, который содержит все контексты безопасности для ОС вцелом. Файл `users.fc` содержит контекст безопасности для домашних директорий пользователей, которые имеют идентификатор SELinux (selinux identities). Директория `program/` содержит контексты безопасности для демонов, находящихся в директории `domains/program/`. Это взаимнооднозначное отображение. Например, файлу `domains/program/syslogd.te` соответствует файл контекста безопасности `file_contexts/program/syslog.fc`.

— `flask`
  Данная директория содержит конфигурационные файлы, не зависящие от политики. Эти файлы содержат определения, соответствующие определениям заголовков ядра. Эти файлы не должны подвергаться изменениям.
— `fs_use`
  Данный файл описывает используемые файловые системы. Данный файл не должен быть изменен.
— `genfs_contexts`
  Данный файл содержит контексты безопасности для файлов и файловых систем, которые не поддерживают постоянную файловую маркировку, такие как `/proc`.
— `initial_sid_contexts`
  Данный файл содержит первоначальный контекст для идентификатора безопасности. Данный файл не должен быть изменен.
— `macros/`
  Данная директория содержит m4-макросы, используемые для быстрого создания и поддержки политики. Она содержит файл макроса `admin_macros.te` для описания административных доменов, таких как `sysadm_t`. Файл `user_macros.te` содержит макрос, используемый в пользовательских доменах, таких как `user_t` и `staff_t`. Файл `global_macros.te` содержит макрос, используемый для всей политики в целом.
— `net_contexts`
  Данный файл определяет контексты безопасности сетевых объектов, таких как порты, интерфейсы и узлы. Если демоны используют для работы нестандартные порты, то потребуется изменить данный файл.
— `rbac`
  Данный файл содержит контроль доступа к файлам, основанный на понятии роли. Он описывает основные разрешения переключения ролей. Дополнительные переключения ролей должны быть разрешены в основных TE файлах (`user.te`, `staff.te` и т.п.).
— `tmp/`
  Эта директория используется для хранения промежуточных файлов при компиляции политики. При необходимости данная директория может быть безопасно удалена.
— `types/`
  Эта директория содержит файлы для декларации основных типов системы, которые не удовлетворяют индивидуальным политикам программ.

3.3. Типы файлов

Файл `file.te` содержит определения типов файлов. Приведем описание некоторых типов.

— `unlabeled_t`
  Используется при контроле доступа к файлам на файловой системе, не поддерживающей метки. Для этого типа доменов не выделено специальных прав доступа.
— `fs_t`
  Определяет контроль доступа к используемой файловой системе. Этот тип задается по умолчанию только для файловой системы `ext2`, которая не использует метки во время первоначального монтирования. Все типы файлов могут быть созданы в этой файловой системе. Всем доменам разрешено поддерживать атрибуты этого типа файловой системы. `kernel_t`, `initrsc_t`, и администратор доменов предоставляют полномочия для монтирования и отмонтирования этого типа.
— `file_t`
  Определяет контроль доступа к файлам. Этот тип автоматически применяется к файлам файловой системы `ext2`, которая не использует метки во время первоначального монтирования. Все типы директорий с правами `root` могут быть подмонтированы к директории этого типа. Административный домен и

`initrc_t` предоставляют права использования директорий с этим типом в качестве точек монтирования.

— `tmp_t`
Определяет контроль доступа к временным директориям. Все домены предоставляют право на создание и разделение файлов в этих директориях. Обеспечивается разделение временных файлов по подтипам в соответствии с доменами, к которым относятся эти файлы.

— `etc_t`
Определяет контроль доступа к информации о конфигурации системы. Этот тип предоставляет право на чтение всем доменам, но изменен может быть только административным доменом и доменом с типом `passwd_t`. Тип `etc_runtime_t` присваивается файлам, создаваемым во время загрузки.

— `lib_t`
Используется для контроля доступа к системным библиотекам. Все домены имеют право на чтение файлов и директорий этого типа. Только административный домен может предоставить права на изменение этого типа.

— `shlib_t`
Используется для контроля доступа к разделяемым системным библиотекам. Тип `ld_so_t` используется для контроля доступа к динамическим разделяемым библиотекам. Все домены имеют право на чтение файлов и директорий этих типов, выполнять программы, имеющие тип `ld_so_t`, и исполнять код, имеющий тип `shlib_t`. Только административный домен может предоставить права на изменение этих типов.

— `bin_t`
Используется для контроля доступа к системным бинарным файлам. Все домены имеют право на чтение файлов и директорий этого типа, и некоторые домены имеют право на их исполнение. Только административный домен может предоставить права на изменение этого типа. `sbin_t` используется для контроля доступа к бинарным файлам суперпользователя (`superuser`) и аналогичен `bin_t`.

— `var_t`
Определяет контроль доступа к директории `var`. Этот тип частично эквивалентен типу директории `/root`. Для различных поддиректорий определены различные подтипы: `var_run_t` — для директории `/var/run`, `var_log_t` — для директории `/var/log` и т.д. Все домены имеют право на чтение файлов и директорий этих типов. Административный домен может предоставить права на изменение этих типов. Кроме того, `var_run_t` может быть изменен демонами и доменом, имеющим тип `initrc_t`; `var_log_t` может быть изменен доменами, имеющими типы `initrc_t`, `syslogd_t`, `crond_t`, `logrotate_t`, и доменом, который отвечает за процедуру регистрации (login).

4. Адаптация системы udev для SELinux

Для использования системы `udev` для SELinux необходимо создать файл контекста безопасности `udev.fc` и файл политики `udev.te`, отредактировать файл `tunable.te`.

4.1. Файл tunable.te

В файле `tunable.te` необходимо добавить следующую запись:

```
define('dev_on_tmpfs')
define('distro_gentoo')
```

где `define('distro_gentoo')` определяет дистрибутив Gentoo, а `define('dev_on_tmpfs')` вводится для монтирования каталога `/dev` на файловую систему `tmpfs`.

4.2. Файл контекста безопасности udev.fc

Файл контекста безопасности `udev.fc` имеет следующий вид:

```
## udev.fc
/sbin/udevsend                -- system_u:object_r:udev_exec_t
/sbin/udev                    -- system_u:object_r:udev_exec_t
/sbin/udevd                   -- system_u:object_r:udev_exec_t
/sbin/start_udev              -- system_u:object_r:udev_exec_t
/sbin/udevstart               -- system_u:object_r:udev_exec_t
/usr/bin/udevinfo             -- system_u:object_r:udev_exec_t
/etc/dev\.d/.+                -- system_u:object_r:udev_helper_exec_t
/etc/udev/scripts/.+          -- system_u:object_r:udev_helper_exec_t
/etc/udev/devices/.*             system_u:object_r:device_t
/etc/hotplug\.d/default/udev.*-- system_u:object_r:udev_helper_exec_t

/dev/udev\.tbl                -- system_u:object_r:udev_tbl_t
/dev/\.udev\.tdb(/.*)?        -- system_u:object_r:udev_tdb_t
/dev/\.udevdb                 -d system_u:object_r:udev_tdb_t
/dev/\.udevdb/.*              -- system_u:object_r:udev_tdb_t
/sbin/wait_for_sysfs          -- system_u:object_r:udev_exec_t
```

Первая колонка файла `udev.fc` является регулярным выражением для имен файлов, к которым применяется политика.

Вторая колонка представляет собой контекст, к которому применяется политика. При этом в именах файлов «+» обозначает хотя бы один символ, «?» — только один символ, «*» — сколько угодно символов, «\.» — символ точки в именах файлов.

Контекст безопасности может иметь следующие опции: отсутствие опций обозначает все файлы данной директории, «-c» — символьные файлы, «-d» — директории, «-b» — бинарные файлы, «--» — все директории с подкаталогами и файлами.

В контексте безопасности сначала указывается пользователь (в данном случае `system_u`), затем роль (в данном случае `oblect_r`) и тип (например, `udev_exec_t`, обозначающий тип исполняемых файлов `udev`).

4.3. Файл политики udev.te

При разработке файла политики `udev.te` использовались сообщения, отображаемые кэшом вектора доступа (Access Vector Cache, AVC) при использовании команды:

```
dmesg | audit2allow -d -v > pol.6v
```

где `pol.6v` — имя файла, содержащего правила, которые необходимо разрешить (прописать) в файле `udev.te`, ключ `-d` означает, что входные данные имеют формат программы `dmesg`, ключ `-v` позволяет сделать вывод информации более подробным.

Например, в данном файле может содержаться запись

```
allow udev_t policy_config_t:file { getattr read };
#EXE=/sbin/udev  NAME=file_contexts : read
#EXE=/sbin/udev  PATH=/etc/security/selinux/file_contexts : getattr
```

говорящая о том, что необходимо прописать в файле `udev.te` правило «`allow udev_t policy_config_t:file { getattr read };`», которое разрешит домену `udev_t` читать и получать атрибуты файлов, имеющих тип `policy_config_t`.

Опишем содержимое разработанного нами файла `udev.te`.

```
# udev.te
################################################################
# Правила для домена udev_t.
################################################################
## Определяем макрос daemon_domain, который задает стандартные права
```

```
## доступа для работы udev.
daemon_domain(udev, ', nscd_client_domain, privmodule, privmem,
              fs_domain, privfd, privowner')

## Определяем основной домен доступа.
general_domain_access(udev_t)

## Определяем домен для конфигурационных файлов.
etc_domain(udev)

## Определяем типы udev_helper_exec_t (вспомогательные скрипты для
## работы udev) и udev_tdb_t, при этом разрешаем ассоциировать
## данный тип с типом обычных файлов, типом файлов системного
## администратора, типом исполняемых файлов и т.п.
type udev_helper_exec_t, file_type, sysadmfile, exec_type;
type udev_tdb_t, file_type, sysadmfile, dev_fs;

## Типу udev_tdb_t присваиваем псевдоним udev_tbl_t (для совместимости).
typealias udev_tdb_t alias udev_tbl_t;

########################################################################
# Макросы, необходимые для работы udev
########################################################################
## Макрос file_type_auto_trans необходим для взаимодействия типов и в
## качестве своих параметром использует создаваемый домен, тип
## родительской директории, тип нового файла, и множество классов файлов.
file_type_auto_trans(udev_t, device_t, udev_tdb_t, file)

## Макрос domain_auto_trans необходим для взаимодействия доменов и в
## качестве своих параметром использует текущий домен, тип
## программы и тип нового домена.
domain_auto_trans(kernel_t, udev_exec_t, udev_t)
domain_auto_trans(udev_t, restorecon_exec_t, restorecon_t)
domain_auto_trans(udev_t, ifconfig_exec_t, ifconfig_t)

## Макрос can_exec авторизует домен udev_t на исполнение типов,
## указанных в качестве параметра, без изменения домена.
can_exec(udev_t, { shell_exec_t bin_t sbin_t etc_t udev_exec_t} )

## Макрос can_exec_any авторизует домен udev_t на выполнение типов для
## набора системных файлов.
can_exec_any(udev_t)

## Макрос can_getsecurity разрешает домену udev_t получать политику.
can_getsecurity(udev_t)

## Макрос can_setfscreate разрешает домену udev_t устанавливать
## контекст на файловую систему.
can_setfscreate(udev_t)

## Макрос сообщает об изменениях в устройствах прикладных программ
## (взаимодействие с демоном dbus, сообщающего графическим программам
## об изменениях в системе).
dbusd_client(system, udev)

########################################################################
# Правила, необходимые для работы udev
########################################################################
## Системе udev разрешается выполнять операции в домене udev_t.
allow udev_t self:capability
  { chown dac_override dac_read_search fowner fsetid sys_admin
    sys_nice mknod net_raw net_admin };

## Данные правила разрешают домену udev_t поиск в директориях, имеющих
## тип policy_config_t, var_log_t, var_lock_t, sysctl_dev_t
## или mnt_t, что необходимо для работы с политиками, log-файлами
```

```
## или файлами устройств.
allow udev_t {policy_config_t var_log_t var_lock_t }:dir search;
allow udev_t {sysctl_dev_t  mnt_t}:dir search;

## Данные правила разрешают домену udev_t поиск в директориях
## и получение атрибутов каталогов типа devpts_t, что необходимо для
## работы с псевдотерминалами.
allow udev_t devpts_t:dir { getattr search };

## Данное правило разрешает домену udev_t читать объекты класса dir
## и получать атрибуты директорий типа file_t.
allow udev_t file_t:dir { getattr read };

## Данное правило разрешает домену udev_t читать объекты класса file
## и получать атрибуты файлов типа udev_t, policy_config_t и т.п.
allow udev_t { self policy_config_t etc_runtime_t sysctl_dev_t
               sysctl_modprobe_t sysctl_kernel_t
               sysctl_hotplug_t }:file { getattr read };

## Данные правила разрешают домену udev_t получение атрибутов
## файлов типа proc_kcore_t и var_lock_t.
allow udev_t { proc_kcore_t var_lock_t }:file getattr;

## Данное правило разрешает домену udev_t читать объекты класса file,
## получать атрибуты, управлять вводом выводом  файлов типа
## etc_t и proc_t.
allow udev_t { etc_t proc_t }:file { getattr read ioctl };

## Данные правила необходимы для работы с файлами терминалов.
allow udev_t sysadm_tty_device_t:chr_file { read write };
allow udev_t file_t:chr_file { getattr ioctl read write };

## Данные правила разрешают домену udev_t совершать над сокетами
## типа udev_t  действия, указанные в качестве параметров.
allow udev_t self:unix_dgram_socket create_socket_perms;
allow udev_t self:unix_stream_socket
  {connectto create_stream_socket_perms};

## Данное правило дает доступ домену udev_t на чтение и запись
## именованных каналов типа udev_t.
allow udev_t self:fifo_file rw_file_perms;

## Данное правило разрешает домену udev_t читать и записывать файлы
## типа device_t.
allow udev_t device_t:file rw_file_perms;

## Данные правила дают доступ домену udev_t на чтение
## файлов типа initrc_var_run_t и modules_dep_t.
allow udev_t { initrc_var_run_t modules_dep_t }:file r_file_perms;

## Данное правило разрешает домену udev_t создавать сокет-файлы
## типа device_t.
allow udev_t device_t:sock_file create_file_perms;

## Данное правило разрешает домену udev_t создавать rawip-сокеты
## типа udev_t.
allow udev_t self:rawip_socket create_socket_perms;

## Данное правило разрешает домену udev_t создавать ссылки на файлы
## типа device_t.
allow udev_t device_t:lnk_file create_lnk_perms;

## Данные правила разрешают домену udev_t читать директории и ссылки
## на файлы типа bin_t и sbin_t.
allow udev_t { bin_t sbin_t }:dir r_dir_perms;
allow udev_t { bin_t sbin_t }:lnk_file read;
```

```
## Данное правило разрешает домену udev_t читать директории
## типа udev_helper_exec_t.
allow udev_t udev_helper_exec_t:dir r_dir_perms;

## Данное правило разрешает домену udev_t создавать и
## перемаркировывать директории типа device_t и device_type.
allow udev_t device_t:dir { relabelfrom relabelto create_dir_perms };

## Данное правило разрешает домену udev_t создавать и
## перемаркировывать файлы типа device_t и device_type.
allow udev_t { device_t device_type }:{ chr_file blk_file }
             { relabelfrom relabelto create_file_perms };

## Отключение аудита
dontaudit udev_t domain:dir r_dir_perms;
dontaudit udev_t ttyfile:chr_file unlink;
dontaudit udev_t initrc_var_run_t:file write;
dontaudit udev_t file_t:dir search;

## Домену udev_t разрешается читать директории и файлы
## типа sysfs_t и modules_object_t (необходимо для работы sysfs и
## взаимодействия с модулями ядра ОС).
r_dir_file(udev_t, sysfs_t)
r_dir_file(udev_t, modules_object_t)
r_dir_file(udev_t, domain)

## Домену udev_t разрешается читать директории и файлы
## типа selinux_config_t, file_context_t, default_context_t.
r_dir_file(udev_t,
          { selinux_config_t file_context_t default_context_t } )

## Данные правила разрешают домену udev_t совершать различные
## действия над объектами типа kernel_t (использование файловых
## дескрипторов, работа с datagram-сокетами, посылать
## сигналы процессам).
allow udev_t kernel_t:fd use;
allow udev_t kernel_t:unix_dgram_socket { sendto ioctl read write };
allow udev_t kernel_t:process signal;

## Отключение аудита при имеющейся известной ошибке в реализации,
## но при отсутствии угрозы безопасности.
ifdef(`hide_broken_symptoms', `
  dontaudit restorecon_t udev_t:unix_dgram_socket { read write };
  dontaudit ifconfig_t udev_t:unix_dgram_socket { read write };
')

## Данное выражение необходимо для работы xdm.
ifdef(`xdm.te', ` allow udev_t xdm_var_run_t:file { getattr read }; ')

## Данное выражение необходимо для интеграции hotplug и udev.
ifdef(`hotplug.te', `
  r_dir_file(udev_t, hotplug_etc_t)
  r_dir_file(udev_t, hotplug_var_run_t)
')

## Данные выражения необходимы для работы с консолью.
ifdef(`consoletype.te', `
  can_exec(udev_t, consoletype_exec_t)
')

## Данные выражения необходимы для взаимодействия с клиентом dhcp.
ifdef(`dhcpc.te', `
  domain_auto_trans(udev_t, dhcpc_exec_t, dhcpc_t)
')
```

```
ifdef(`dhcpd.te', `define(`use_dhcp')')
ifdef(`dhcpc.te', `define(`use_dhcp')')
ifdef(`use_dhcp', `
  allow udev_t dhcp_etc_t:file rw_file_perms;
  file_type_auto_trans(udev_t, etc_t, dhcp_etc_t, file)
')

## Правила для функционирования /var/lib/init.d в дистрибутиве Gentoo,
(init.d может монтироваться на tmpfs).
ifdef(`distro_gentoo', `
  allow udev_t tmpfs_t:file
    { append create getattr ioctl read rename setattr write };
  allow udev_t initrc_state_t:file
    { create getattr unlink append ioctl read rename setattr write };
  allow udev_t initrc_state_t:dir
    { add_name getattr read remove_name search write };
  allow udev_t initrc_exec_t:file
    { execute execute_no_trans ioctl read };
  allow udev_t var_lib_t:dir search;
')

## Правила для устройств, работающих на файловой системе tmpfs.
ifdef(`dev_on_tmpfs', `
  allow udev_tdb_t tmpfs_t:filesystem associate;
  allow device_t tmpfs_t:filesystem associate;
  allow tty_device_t tmpfs_t:filesystem associate;
  allow zero_device_t tmpfs_t:filesystem associate;
  allow urandom_device_t tmpfs_t:filesystem associate;
  allow random_device_t tmpfs_t:filesystem associate;
  allow ptmx_t tmpfs_t:filesystem associate;
  allow null_device_t tmpfs_t:filesystem associate;
  allow mouse_device_t tmpfs_t:filesystem associate;
  allow memory_device_t tmpfs_t:filesystem associate;
  allow devtty_t tmpfs_t:filesystem associate;
  allow console_device_t tmpfs_t:filesystem associate;
  allow clock_device_t tmpfs_t:filesystem associate;
  allow removable_device_t tmpfs_t:filesystem associate;
  allow fixed_disk_device_t tmpfs_t:filesystem associate;
  allow clock_device_t tmpfs_t:filesystem associate;
  allow udev_t tmpfs_t:file unlink;
  allow udev_t tmpfs_t:dir { rmdir create };
  allow udev_t tmpfs_t:dir rw_dir_perms;
  allow udev_t tmpfs_t:sock_file create_file_perms;
  allow udev_t tmpfs_t:lnk_file create_lnk_perms;
  allow udev_t tmpfs_t:{ chr_file blk_file }
    { relabelfrom relabelto create_file_perms };
  allow udev_t tmpfs_t:dir search;
')

## Данное правило разрешает системе udev выполнять программы, имеющие
## тип udev_helper_exec_t без перехода в другой домен.
allow udev_t udev_helper_exec_t:file execute_no_trans;

## Отключение защиты для домена udev_t
## (необходимо только при отладке политики).
ifdef(`unlimitedUtils', ` unconfined_domain(udev_t) ')
```

Заключение

К сожалению, на данный момент не удалось разработать универсальную методологию написания политик безопасности SELinux для таких систем, как udev. Это связано с тем, что SELinux еще находится в стадии разработки и за небольшой

промежуток времени происходят достаточно существенные изменения в структуре системы, вследствие чего меняется синтаксис языка и семантика политики.

Литература

1. *Kroah-Hartman G.* udev and devfs - The final word. — 2003. — http://www.us.kernel.org/pub/linux/utils/kernel/hotplug/udev_vs_devfs.
2. Gentoo udev Guide. — 2003. — http://www.gentoo.org/doc/en/udev-guide.xml.
3. Security-Enhanced Linux. — http://www.nsa.gov/selinux/.
4. *Кокер Ф.* Введение в SELinux: новый SELinux. — 2003. — http://gazette.linux.ru.net/rus/articles/intro_selinux.html.
5. *Праневич Д.* Замечательный Мир Linux 2.6. — 2003. — http://palm.opennet.ru/base/sys/linux26_intro.txt.html.
6. *Salzman P. J., Burian M., Pomerantz O.* The Linux Kernel Module Programming Guide. Перевод: А. Киселёв. — 2004. — http://gazette.linux.ru.net/rus/articles/lkmpg.html.
7. *Loscocco P., Smalley S.* Integrating Flexible Support for Security Policies into the Linux Operating System. — 2001. — http://www.nsa.gov/selinux/slinux-abs.html.
8. *Loscocco P., Smalley S.* Meeting Critical Security Objectives with Security-Enhanced Linux. — 2001. — http://www.nsa.gov/selinux/papers/ottawa01-abs.cfm.
9. *Smalley S., Fraser T.* A Security Policy Configuration for the Security-Enhanced Linux: Technical report / NAI Labs. — 2001. — http://www.nsa.gov/selinux/papers/policy/policy.html.
10. *Morris J.* Filesystem Labeling in SELinux // Linux Journal. — 2004. — http://www.linuxjournal.com.
11. Frequently Asked Questions about udev. — 2003. — http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ.
12. *Drake D.* Writing udev rules. — 2004. — http://www.reactivated.net/udevrules.php.

UDC 681.322:621.382.26:004.056.52

Adaptation of the System udev for the SELinux

D. S. Kulyabov, A. V. Korolkova

Telecommunication Systems Department
Peoples' Friendship University of Russia
Miklukho-Macklaya str., 6, Moscow, 117198, Russia

In this article the authores examine approaches to the work with files of devices in OS Linux. Also the structure of udev system and the detailed description of developed policies are given here.