# SELinux

Фёдор Сахаров
sakharov@lvk.cs.msu.su

Лаборатория вычислительных комплексов ВМК
МГУ имени М.В.Ломоносова

Москва, 2011

# Краткое содержание лекции

## Сегодня в планах:

- Стандартная модель безопасности в UNIX-подобных системах
- Расширения стандартной модели безопасности.
- SELinux: motivation
- SELinux: theory
- SELinux: practice
- SELinux: hit the fan

# Стандартная модель безопасности в UNIX

Основная идея — система привилегий.

В системе существует набор пользователей.

```
# cat /etc/passwd | sed 's/:.*//g'
root
bin
daemon
adm
...
```

Множество пользователей разбивается на подмножества, называемые группами.

# Стандартная модель безопасности в UNIX

В системе существует множество файлов. У каждого файла есть владелец(один из пользователей) и группа владельцев.

С каждым из файлов ассоциирован набор прав доступа.

## Набор прав включает в себя:

- Права владельца
- Права группы владельцев
- Права остальных пользователей

```
-rw-r--r-- 1 root root /etc/passwd
```

# Стандартная модель безопасности в UNIX

**Любой процесс, запущенный пользователем, обладает всеми правами этого пользователя.**

# Стандартная модель безопасности в UNIX

**Любой процесс, запущенный пользователем, обладает всеми правами этого пользователя.**

Это значит, что:

- **Права процессов ограничены правами пользователя и это хорошо. Позволяет во многих случаях сохранить конфиденциальность и целостность информации.**
- **Права процессов ограничены правами пользователя и это плохо. Набор привилегий пользователя, как правило, является слишком широким для любого из процессов, запущенных данным пользователем.**

# Стандартная модель безопасности в UNIX

Набор привилегий пользователя, как правило, является слишком широким для любого из процессов, запущенных данным пользователем.

Любое приложение имеет доступ к любым ресурсам, к которым имеет доступ пользователь.

Например, skype имеет право на доступ к файлам в /.mozilla/firefox.

# Стандартная модель безопасности в UNIX

Набор привилегий пользователя, как правило, является слишком широким для любого из процессов, запущенных данным пользователем.

В результате, компрометация любого из пользовательских приложений позволяет злоумышленнику получить все привилегии пользователя.

```
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "Htf4CRBDpVhcwwC9\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (10.0.0.100:4444 -> 10.0.0.3:38149) at Mon May

whoami
distccd
```

# Стандартная модель безопасности в UNIX

С точки зрения «Общих Критериев» (Common Criteria) в стандартной модели безопасности UNIX существует большой потенциал для повсеместных улучшений.

В частности, могут быть реализованы расширения:

- Role Based Access Control
- Mandatory Access Control
- etc.

# Расширения стандартной модели безопасности

В результате попыток расширить стандартную систему безопасности UNIX появилось множество проектов.

## В разное время разными людьми были сделаны:

- SELinux (selinuxproject.org)
- AppArmor (apparmor.net)
- Tomoyo Linux (tomoyo.sourceforge.jp)
- GRSecurity (grsecurity.net)
- TrustedBSD, SEBSD, SEDarwin, Darwin Seatbelt

# SELinux

Далее речь пойдет о системе SELinux (Security Enhanced Linux) — проект реализованный в NSA и вошедший в ядро Linux.

## SELinux реализует следующие модели безопасности:

- Mandatory Access Control, представленный двумя типами:
  - Type Enforcement (TE)
  - Multi-Level Security (MLS)
- Role Based Acces Control

# SELinux Type Enforcement

Каждой сущности в системе SELinux ставит в соответствие некоторый тип.

Типы могут относиться к:

- Субъектам (процессы в системе)
- Объектам (файлы, устройства, IPC, etc.)

Для каждого типа субъектов в явном виде описываются разрешенные операции над какими-либо типами объектов.

# SELinux Type Enforcement

Конфигурация SELinux производится с помощью описания набора типов, ролей и правил в политике SELinux.

Политика SELinux описывается на специальном языке и компилируется в бинарное представление.
Скомпилированное представление загружается в ядро.

# SELinux Type Enforcement

Существует готовая политика SELinux, которая содержит описание типов и профилей для большого числа приложений.

Исходные коды политики можно найти в $SELINUX_POLICY_SRC/policy/modules/

## Дерево исходных кодов политики содержит:

- admin – Профили административных утилит (su, netutils, rpm, apt, etc.)
- apps – Профили приложений (mozilla, wireshark, java, etc.)
- kernel – Профили стафа, относящегося к ядру
- roles – Описания ролей для ролевого контроля доступа
- services – Различные сервисы (ssh, snmp, jabber, etc.)
- system – Системные приложения (init, logging, udev, xen, etc.)

# SELinux Type Enforcement

Основная идея принудительного присвоения типов (TE) – принудительное присвоение типов.

Если говорить точнее, каждому объекту или субъекту в системе принудительно ставится в соответствие тройка, содержащая в терминах SELinux идентификатор владельца данной сущности, роль и тип:

```
system_u:object_r:etc_t
```

Подобная тройка называется **контекстом безопасности.**

# SELinux Type Enforcement

Каждому файлу в системе присваивается определенный контекст безопасности.

Контексты безопасности хранятся в расширенных атрибутах файловой системы. (supported by ext, xfs, jfs and some others)

Определение контекстов безопасности файлов производится в файлах политики с расширением **fc**

# SELinux Type Enforcement

## Пример определения файловых контекстов безопасности:

Выдержка из файла ./policy/modules/services/ssh.fc

```
/etc/ssh/primes           -- gen_context(system_u:object_r:sshd_key_t,s0)
/etc/ssh/ssh_host_key     -- gen_context(system_u:object_r:sshd_key_t,s0)
/etc/ssh/ssh_host_dsa_key -- gen_context(system_u:object_r:sshd_key_t,s0)
/etc/ssh/ssh_host_rsa_key -- gen_context(system_u:object_r:sshd_key_t,s0)
```

Здесь определяются файловые контексты для ключей ssh. Тип данных файлов – ssh_key_t.

# SELinux Type Enforcement

Все объекты в системе делятся на классы.

Классы могут относиться к:

- файловым объектам: filesystem, file, dir
- сетевым объектам: tcp_socket, netif

С каждым из классов ассоциируется набор операций, которые субъект может попытаться произвести с объектом, чей тип принадлежит данному классу.
Такой набор операций называется вектором доступа (Access Vector)

# SELinux Type Enforcement: classes

Основные классы определены в файле ./policy/flask/

## Пример класса файла:

```
common file
{
    ioctl
    read
    write
    ...
}
```

Класс может быть унаследован от другого класса, в таком случае его вектор доступа будет объединен с вектором доступа родительского класса:

```
class dir
inherits file
{
    add_name
    remove_name
    ...
}
```

# SELinux Type Enforcement: attributes

Атрибуты являются способом группировки схожих типов. У типа может быть любое число атрибутов, атрибут может быть приписан любому числу типов.

```
# The domain attribute identifies every type that can be
# assigned to a process.  This attribute is used in TE rules
# that should be applied to all domains, e.g. permitting
# init to kill all processes.
attribute domain;

# Identifies all default types assigned to packets received
# on network interfaces.
attribute netmsg_type;
```

# SELinux Type Enforcement: types

Для того, чтобы тип можно было использовать, он должен быть определен. Определение типов даётся в файлах с расширением *.te.

Синтаксис определения типов:

```
type <typename> [aliases] [attributes];
```

Примеры определения типов:

```
type httpd_config_t, file_type, sysadmfile;
```

Тип httpd_config_t является файлом системной конфигурации.

# SELinux Type Enforcement: type transitions

Правила переходов типов описывают изменения типов процессов при вызовах exec, типы объектов, создаваемых субъектами.

Синтаксис описания перехода типов:

```
type_transition <source_type(s)> <target_type(s)> : \
                      <class(es)> <new_type>
```

Пример описания перехода типов:

```
type_transition unconfined_t test_exec_t:process test_t;
```

Когда запускается файл с типом test_exec_t новый процесс должен иметь тип test_t.

# SELinux Type Enforcement: access vectors

Привилегии субъектов определяются следующим образом.

```
<av_kind> <source_type(s)> <target_type(s)>:<class(es)> \
        <permission(s)>
```

## av_kind может быть трех типов:

- allow
- auditallow
- dontaudit
- neverallow

# SELinux Type Enforcement: access vectors

## Примеры

```
allow sshd_t sshd_exec_t:file {  getattr open \
                            read execute ioctl };
```

Разрешить процессам с типом sshd_t доступ к файлам с типом sshd_exec_t с правами на чтение, исполнение, вызов ioctl.

```
allow sysadm_ssh_agent_t etc_t:dir { getattr \
                                     search open };
```

Разрешить процессам с типом sysadm_ssh_agent_t доступ к файлам с типом etc_t с правами getattr, search, open.

# SELinux Type Enforcement: access vectors

Когда SELinux запрещает процессу доступ к какому-либо объекту, информация об этом заносится в логи.

## Пример:

```
avc:  denied  { rename } for  pid=4220 comm="semodule" name="active"
dev=hda ino=26785 scontext=unconfined_u:unconfined_r:semanage_t:s0-s0:c0.c1023
tcontext=unconfined_u:object_r:selinux_config_t:s0 tclass=dir
```

Процессу semodule с контекстом unconfined_u:unconfined_r:semanage_t: было запрещено переименовать файл active с контекстом unconfined_u:object_r:selinux_config_t.

# SELinux Type Enforcement: users and roles

Роли определяют, к каким доменам может иметь доступ пользователь.
Определение роли:

```
role <rolename> types <domain(s)>;

role sysadm_r types ldconfig_t;
/* administrators are allowed access
   to ldconfig_t domain */
```

Возможность юзеру в роли user_r изменять роль на sysadm_r

```
allow user_r sysadm_r;
```

# SELinux Type Enforcement: users and roles

```
role_transition sysadm_r $1_exec_t system_r;
```

Когда администратор запускает файл с типом $_exec_t, процесс изменяет свою роль с sysadm_r на system_r.

```
role_transition unconfined_r direct_init_entry system_r;
```

Когда пользователь в неопределенной роли запускает файл, у которого есть атрибут direct_init_entry, изменить роль процесса на system_r.

# SELinux Type Enforcement: users and roles

Пользователи в терминах SELinux отличаются от пользователей в UNIX.

Идентификатор пользователя в терминах SELinux является частью контекста безопасности.

Пользователи определяются в файле ./policy/users

# SELinux Type Enforcement: not covered in this talk

За рамками рассказа остались следующие особенности SELinux:

- Constraints
- Special interfaces and filesystems
- libselinux API
- Core SELinux policy workflow

# SELinux Type Enforcement: hit the fan

SELinux является весьма мощным расширением стандартной модели безопасности UNIX.
Тем не менее, у него есть масса недостатков:

- Сложность описания политик и администрирования
- Необходимость давать приложениям весь набор прав, необходимый им для нормального исполнения.

# Спасибо за внимание.

Вопросы?