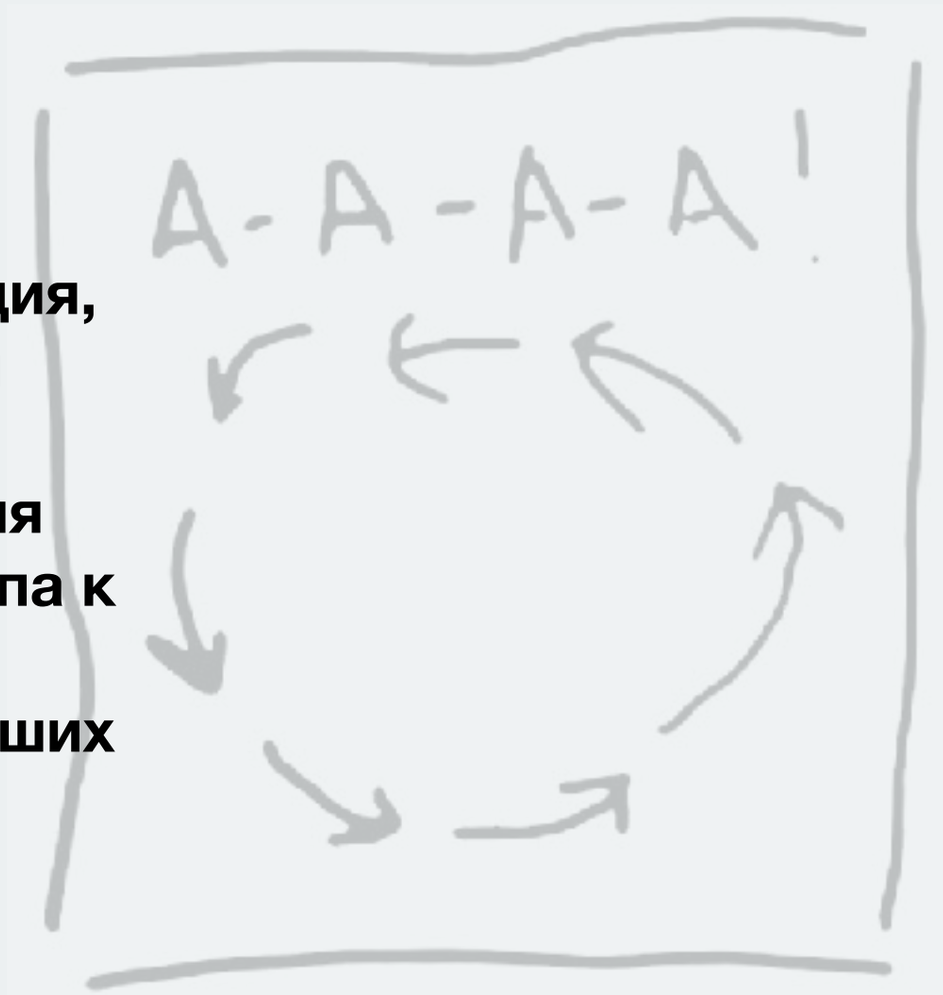
ОНЛАЙН-ОБРАЗОВАНИЕ

AAA

Основы безопасности в Linux

Александр Румянцев

AAA



- **Authentication - Аутентификация, идентификация, процесс подтверждения пользователем своей “подлинности”. Ввод логина и пароля.**
- **Authorization - Авторизация, процесс наделения пользователя правами (предоставления доступа к каким-либо объектам)**
- **Accounting - Запись информации о произошедших событиях.**

Классическая аутентификация

/etc/passwd - раньше хранил все реквизиты, включая пароль. Теперь пароль вынесен в отдельный файл /etc/shadow, который недоступен для чтения непривилегированным пользователям. Кроме того, shadow содержит:

- **имя пользователя**
- **шифрованный пароль**
- **дата последней смены пароля**
- **дней должно пройти между сменами пароля**
- **дней перед необходимостью смены пароля**
- **дней вывода предупреждения об устаревании**
- **дней перед тем как учётная запись заблокируется**
- **дата устаревания учётной записи**

Классическая аутентификация

Классические условия:

- **шелл должен существовать и быть перечисленным в /etc/shells**
- **root может залогиниться только с терминала, перечисленного в /etc/securetty**

Классическая аутентификация

Блокировка пользователя делается несколькими методами:

- **Установка shell в /bin/nologin**
- **Установка expiry date в прошлое**
- **Блокировка пароля (делаем его непроверяемым, добавляя "!")**

UID. Системные и пользовательские

Организационное разделение, при котором существуют две группы пользователей: одна группа - для демонов, другая - для живых пользователей.
Определено в /etc/login.defs, используется в системных утилитах работы с пользователями и группами

Классические аттрибуты файлов

UID, GID
owner (rwx), group (rwx), other (rwx)
directory bit, suid bit, sgid bit.

Установленный suid/sgid биты для исполняемых файлов означает, что UID процесса будет установлен по UID файла, а не отнаследован от родительского процесса.

su и sudo - одни из немногих утилит, использующих suid бит

Для директорий смысл немного другой: создаваемые файлы внутри такой директории будут иметь UID и/или GUD родительской директории. Для создаваемых директорий биты будут наследоваться. Простой классический способ создать "общую папку" для разных пользователей

root caps

Когда-то пользователь с UID 0 был "богом в системе", но теперь это не так. "Божественные" функции побиты на 30+ привилегий и могут выставляться в атрибутах файла вместо suid/sgid битов, тем самым "божественными" функциями можно наделить любых пользователей, аналогично использованию suid бита.

Аналог "рута" в терминах привилегий - CAP_SYS_ADMIN
Аттрибуты хранятся в extended attributes в ключе security.capability

```
sudo setcap cap_net_raw+p /bin/ping
```

man 7 capabilities
man 8 setcap
man 1 capsh
https://www.opennet.ru/man.shtml?topic=capabilities

Наследование прав

Все процессы порождаются с помощью системного вызова fork(), который копирует пространство процесса, включающие все открытые сокеты и разнообразные аттрибуты процесса, в т.ч. и uid, gid, capabilities.

Процесс с привелегией CAP_SETUID (например, agetty, висящий на vty), вызывает fork(), потом setuid(), после чего запускает (execve) уже bash. Именно поэтому при изменении, например, членства в группе надо "перелогиниться" или перезапустить сервис

PAM

Библиотека, реализующая возможность AAA с помощью стандартизированных динамически подгружаемых модулей.

Конфигурируется в /etc/pam.d

Имя файла - это имя, которое задается при вызове функции аутентификации, как правило совпадает с именем демона

PAM

Аутентификация:

- **pam_unix - аутентификация в /etc/passwd**
- **pam_userdb - аутентификация против (against) BDB-файла**
- **pam_rootok - разрешение root’у всего.**

Авторизация:

- **pam_wheel - авторизация для сервиса с использованием группы wheel**

Модули не только позволяют реализовать процесс аутентификации, авторизации, но и обрабатывать некоторые дополнительные вещи.

- **pam_mkhomedir - создание домашнего каталога пользователя**

PAM

Конфигурация для каждого сервиса определяется в формате:

`type control module-path module-arguments`

В конфигурации, внутри каждого стека (*`type`*) есть один или несколько модулей, которые просматриваются по порядку (сверху-вниз) и, в зависимости от control-a (`sufficient, required`) проверка либо продолжается, либо прекращается.

PAM type

- `auth[entication]` - **Секция для аутентификации**
- `account` - **Секция посвященная авторизации и настроек аккаунта (заблокирован ли?)**
- `password` - **Секция описывающая модули используемые для модификации пароля. Разные бэкенды требуют разных действий для смены пароля.**
- `session` - **Accounting секция, также тут можно выполнять различные pre/post задачи для установки сессии.**

PAM control

- `required` - **Для успешного завершения проверки этот модуль должен сработать, проверка продолжается вне зависимости от результата.**
- `requisite` - **как required, только при ошибке проверка прекращается. Возвращается ошибка первого required/requisite модуля вернувшего ошибку.**
- `sufficient` - **При успехе возвращается OK и проверка завершается**
- `optional` - **опциональный модуль. Успех или Ошибка важны только в случае если это единственный модуль в стеке.**
- `include` - **включить в текущий стек модулей стек модулей из файла**
- `substack` - **включить в текущий стек результат модулей из файла**

PAM

С помощью пакета pam_script можно быстро и гибко расширять AAA-процесс и использовать возможности PAM. Например можно настроить авторизацию пользователя в зависимости от погоды в Гонолулу.

man 7 pam-script

Пользователям можно назначить привелегии через pam_cap.so

http://www.tin.org/bin/man.cgi?section=8&topic=pam_cap
http://www.tin.org/bin/man.cgi?section=5&topic=capability.conf

Спасибо за внимание!

Вопросы?