2. Pluggable Authentication Module

- Что такое Pluggable Authentication Module
- Формат конфигурационных файлов PAM
- Примеры конфигурации

Pluggable Authentication Module

- Подключаемые модули аутентификации (*Pluggable Authentication Module*, *PAM*) — это модульный подход к системе аутентификации, позволяющий сторонним службам использовать специальные модули для взаимодействия с различными провайдерами аутентифицирующей информации
- Применение PAM позволяет разнести точку аутентификации (приложение, службу, сервис) и собственно службу аутентификации
- Модульный подход позволяет подключать различные провайдеры без необходимости модификации кода приложений (перекомпиляция, соответственно, также не требуется)
- PAM реализуется в виде библиотеки libpam.so и системы модулей (динамически подгружаемых библиотек `.so`), располагающихся в одной из директорий (в зависимости от дистрибутива):
  - `/lib/security`
  - `/lib64/security`
  - `/usr/lib/x86_64-linux-gnu/security` (на Ubuntu Linux)
- Обнаружить расположение модулей можно, проверив, откуда загружаются `.so` при загрузке т.н. PAM-aware утилиты (например, `login(1)`):

```
$ ldd /bin/login
    linux-vdso.so.1 (0x00007ffeca7c0000)
    libpam.so.0 => /lib/x86_64-linux-gnu/libpam.so.0 (0x00007d7056ad9000)
    libpam_misc.so.0 => /lib/x86_64-linux-gnu/libpam_misc.so.0 (0x00007d7056ad2000)
    libaudit.so.1 => /lib/x86_64-linux-gnu/libaudit.so.1 (0x00007d7056aa4000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007d7056800000)
    libcap-ng.so.0 => /lib/x86_64-linux-gnu/libcap-ng.so.0 (0x00007d7056a9c000)
    /lib64/ld-linux-x86-64.so.2 (0x00007d7056b08000)
$
```

Формат конфигурационного файла PAM

- Изначально конфигурация выполнялась путём задания правил в файле `/etc/pam.conf`
- В современных системах аутентификация и авторизация настраивается для каждого сервиса (приложения) независимо – путём создания одноимённого файла в директории `/etc/pam.d/`; в случае наличия директории `pam.d/` файл `pam.conf` игнорируется
- Third-party правила (в частности, конфигурационные файлы устанавливаемых приложений) могут располагаться в `/usr/lib/pam.d/`, но в случае коллизии имён будет использован конфиг из `/etc/pam.d/`
- Выполняемые модулями задачи подразделяются на четыре группы управления (четыре типа):
  - **account** - задачи по верификации аккаунта (не истёк ли срок жизни пароля, имеет ли пользователь право доступа к ресурсу)
  - **authentication** - аутентификация по паре username/pass (по схеме вызов-ответ, как при парольной аутентификации, но не только)
  - **password** - задачи по обновлению механизма аутентификации (смена пароля)
  - **session** - настройка окружения (установка env. vars, монтирование директорий, логирование попытки входа и проч.)
- Строки в `/etc/pam.conf` имеют вид:

```
service type control module-path module-arguments
```

- Формат файлов в `/etc/pam.d/` по сути тот же:

```
type control module-path module-arguments
```

- **type** – определяет группу управления правила, решаемую с помощью правила задачу
- Группы управления обозначаются следующими ключевыми словами: ***account***, ***auth***, ***password***, ***session***

Формат конфигурационного файла PAM

- **control** определяет обработку результата работы модуля, определены два синтаксиса: простой (краткий) и подробный
- Простая форма:
  - **required** - неудача (ошибка модуля) учитывается, но выполнение сценария всегда продолжается
  - **requisite** - неудача учитывается, в случае неудачи сценарий сразу прекращается
  - **sufficient** - в случае удачи сценарий прекращается, неудача игнорируется (сценарий продолжает выполнение)
  - **optional** - результат учитывается только если это правило единственное в сценарии
- Подробная форма:
  - правила имеют вид `[value1=action1 value2=action2 ...]`
  - **value** принимает одно из значений: `success`, `new_authtok_reqd`, `ignore`, `default` и др.
  - **action** может быть:
    - ignore - результат работы модуля не учитывается
    - bad - в случае ошибки ret модуля используется в качестве результата всего сценария, выполнение сценария продолжается (значение по умолчанию)
    - die - то же, что и bad, но сценарий сразу останавливается
    - ok - в случае успеха переписывает результат сценария, в случае неудачи не используется
    - done - то же, что и ok, но выполнение сценария сразу останавливается
    - N - переход через N правил в сценарии
    - reset - сброс и продолжение исполнения стека модулей
- Сокращённая и подробная форма соотносятся следующим образом:
  - required - `[success=ok new_authtok_reqd=ok ignore=ignore default=bad]`
  - requisite - `[success=ok new_authtok_reqd=ok ignore=ignore default=die]`
  - sufficient - `[success=done new_authtok_reqd=done default=ignore]`
  - optional - `[success=ok new_authtok_reqd=ok default=ignore]`

Формат конфигурационного файла PAM

- **module-path** задаёт относительный или абсолютный путь к модулю (`.so` файлу)
- **module-arguments** задаёт аргументы модуля
- Отдельные модули могут обладать собственными конфиг. файлами, располагающимися в директории `/etc/security`
- Ошибки логируются в `syslog`, местоположение `/var/log/syslog` (можно найти в конфиге `/etc/syslog.conf`)

```
$ less /var/log/syslog
…
$ tail -f /var/log/syslog
```

Пример 1: Запрет логина всем пользователям, кроме **root**

- Изучим содержимое конфиг. файлов `/etc/pam.d/login` и `/etc/pam.d/sshd`, отвечающих за конфигурацию логина через терминал/GUI и `ssh`, соответственно:

```
$ cat /etc/pam.d/login
...
auth    requisite    pam_nologin.so
...
```

```
$ cat /etc/pam.d/sshd
...
auth    required    pam_nologin.so
...
```

- Обе конфигурации используют модуль `pam_nologin.so`, поэтому для запрета логина всем, кроме **root**, достаточно создать файл `/etc/nologin`, содержащий произвольное сообщение пользователю:

```
$ echo "You shall not pass!" | sudo tee /etc/nologin
```

- После создания файла `/etc/nologin` попытка выполнить логин завершится неудачно для обычного пользователя **x** и успешно для **root**:

```
$ ssh x@127.42.42.42
x@127.42.42.42's password:
You shall not pass!
```

```
$ ssh root@127.42.42.42
root@127.42.42.42's password:
You shall not pass!

You shall not pass!
root@vbox:~#
```

- Вместо стандартного файла `/etc/nologin` можно использовать любой другой:

```
$ echo "You shall not pass!" | sudo tee /home/x/nologin
$ sudo nano /etc/pam.d/sshd
...
auth    requisite    pam_nologin.so file=/home/x/nologin
```

- Файлы, содержащие информацию о попытках логина:
  - `sudo last -f /var/log/btmp` - провальные попытки логина
  - `sudo last -f /var/run/utmp` - текущие пользователи в системе
  - `sudo last -f /var/log/wtmp` - история логинов

Пример 2: Запрет логина пользователю **x**

pam_access.so

- Реализация различных сценариев логина пользователя (например, локально или по сети) может быть осуществлена средствами pam_access.so:

```
$ sudo nano /etc/pam.d/common-auth
...
account required pam_access.so
$ sudo nano /etc/security/access.conf
...
-:x:ALL
```

- В результате попытка выполнить логин из-под пользователя **x** завершится неудачно:

```
$ ssh x@127.42.42.42
x@127.42.42.42's password:
Connection closed by 127.42.42.42 port 22
```

pam_listfile.so

- Другой способ заблокировать логин конкретного пользователя – использовать модуль pam_listfile.so
- Модуль pam_listfile.so позволяет настроить правила доступа (разрешение, запрет) для некоторого объекта (пользователь, группа, терминал, командная оболочка и др.):

```
$ echo "x" | sudo tee /etc/loginusers
$ sudo nano /etc/pam.d/common-auth
...
auth    requisite    pam_listfile.so onerr=fail item=user sense=deny file=/etc/loginusers
```

- В результате попытка выполнить логин из-под пользователя **x** завершится неудачно:

```
$ ssh x@127.42.42.42
x@127.42.42.42's password:
Permission denied, please try again.
x@127.42.42.42's password:
Permission denied, please try again.
x@127.42.42.42's password:
Permission denied, please try again.
x@127.42.42.42: Permission denied (publickey,password).
```

Пример 3: Запрет логина пользователю в определённое время

- Ограничение логина в определённый промежуток времени может быть реализовано посредством модуля `pam_time.so`:

```
$ sudo nano /etc/security/time.conf
...
sshd;*;x;!Al0000-2400
$ sudo nano /etc/pam.d/common-auth
...
account    requisite    pam_time.so
```

- В результате попытка выполнить логин из-под пользователя **x** завершится неудачно:

```
$ ssh x@127.42.42.42
x@127.42.42.42's password:
Connection closed by 127.42.42.42 port 22
```

- Ограничения могут распространятся также и на суперпользователя **root**:

```
$ sudo nano /etc/security/time.conf
...
sshd;*;x;!Al0000-2400
*;*;root;Wd1100-2300
```

- В результате попытка выполнить логин из-под пользователя **root** в терминале tty* окажется неудачной в выходные дни и рабочие дни вне промежутка с 11:00 по 23:00

Пример 4: Ограничение доступных пользователю ресурсов

- Изучим содержимое конфиг. файла `/etc/pam.d/sshd`, отвечающего за конфигурацию логина по `ssh`:

```
$ cat /etc/pam.d/sshd
...
# Set up user limits from /etc/security/limits.conf.
session    required    pam_limits.so
...
```

- Как видно, по умолчанию процедура настройки окружения (тип `session`) предполагает установление ограничений, заданных в файле `/etc/security/limits.conf`
- Установим ограничение на максимальный размер файла, создаваемого пользователем **x**:

```
$ sudo nano /etc/security/limits.conf
...
x    hard    fsize    1024
```

- В результате любой процесс, запущенный от пользователя **x**, при попытке создать файл, размером превышающий 1024 байта, будет аварийно завершен:

```
$ ssh x@127.42.42.42
x@127.42.42.42's password:
$ cd /tmp
$ dd if=/dev/urandom bs=1024 count=32 of=/tmp/random_file
$ echo $?
0
$ python3 -c "import os; open('/tmp/random_file2', 'wb+').write(os.urandom(2**10 * 2**11))"
$ Traceback (most recent call last):
  File "<string>", line 1, in <module>
OSError: [Errno 27] File too large echo $?
```

- В результате файл `random_file` размером 32Кб будет создан, а файл `random_file2` размером 2 Мб – нет

Пример 5: Конфигурация парольной политики

- Главный модуль в конфигурации парольной политики – `pam_pwquality.so`
- Правила проверки нового пароля задаются либо в списке аргументов модуля – в конфигурационном файле PAM конкретной службы/сервиса, либо в файле параметров `/etc/security/pwquality.conf`
- Правила, заданные в файле параметров модуля, применяются по умолчанию – всякий раз, как используется модуль; при этом явно заданные аргументы модуля имею более высокий приоритет
- В качестве примера, настроим следующую парольную политику:
  - минимум две цифры в пароле (`dcredit=-2`)
  - минимум два символа верхнего регистра (`ucredit=-2`)
  - минимум два символа нижнего регистра (`lcredit=-2`)
  - минимум два прочих (не буквы/цифры) символа (`ocredit=-2`)
  - минимальная длина пароля – 10 символов (`minlen=10`)
  - максимальное количество попыток задания пароля – 7 (`retry=7`)
- Зададим указанные правила с помощью аргументов модуля в основном конфиг. файле `common-password`:

```
$ sudo nano /etc/pam.d/common-password
...
password    requisite    pam_pwquality.so retry=7 dcredit=-2 ucredit=-2 ocredit=-2 lcredit=-2 dictcheck=0 minlen=10
```

- В результате попытка установить новый пароль, не удовлетворяющей введённым правилам, окажется неудачной:

```
$ passwd
New password: 1234567890
BAD PASSWORD: The password contains less than 2 uppercase letters
New password: AB34567890
BAD PASSWORD: The password contains less than 2 lowercase letters
New password: ABcd567890
BAD PASSWORD: The password contains less than 2 non-alphanumeric characters
New password: ABcd%^yuio
BAD PASSWORD: The password contains less than 2 digits
New password: AAbb##44
BAD PASSWORD: The password is shorter than 10 characters
New password: ABcd%^7890
passwd: password updated successfully
```

Пример 5: Конфигурация парольной политики

- Те же правила могут быть заданы в файле `/etc/security/pwquality.conf`
- Ещё один важный модуль `pam_pwhistory.so` применяется для предотвращения возможности использования повторяющихся паролей
- Зададим проверку повторяющихся паролей – новый пароль сравнивается с тремя последними использованными данным пользователем **x** и отвергается в случае совпадения:

```
$ sudo nano /etc/pam.d/common-password
...
password    requisite pam_pwquality.so retry=7 dcredit=-2 ucredit=-2 ocredit=-2 lcredit=-2 dictcheck=0 minlen=10
password    requisite                   pam_pwhistory.so remember=3
password    [success=2 default=ignore]  pam_unix.so obscure use_authtok try_first_pass yescrypt
...
```

- В результате попытка установить новый пароль, совпадающий с одним из трёх предыдущих, завершается ошибкой:

```
$ passwd
Current password: ABcd%^7890
New password: ABcd%^7891
passwd: password updated successfully
$ passwd
Current password: ABcd%^7891
New password: ABcd%^7890
Password has been already used. Choose another.
$
```

- По умолчанию использованные ранее пароли (хэши от паролей) хранятся в файле `/etc/security/opasswd`

Пример 6: Использование PAM в собственных приложениях

- Помимо конфигурации стандартных утилит и сторонних приложений интерфейс PAM может применятся для реализации настраиваемого функционала аутентификации и авторизации в собственных приложениях
- Пример подобного приложения – утилита `pam_authenticated_printf.c`, запрашивающая пароль и выводящая строку на печать
- Для сборки приложения с PAM необходимо установить пакет `libpam-dev` и скомпоновать с `libpam.so` (опционально, `libpam_misc.so`):

```
$ sudo apt install libpam-dev
$ gcc pam_authenticated_printf.c -o ./pam_authenticated_printf -lpam -lpam_misc
$ ./pam_authenticated_printf
Password:
Authenticated printf :-)
```

- Без создания конфигурационного файла PAM для нашей утилиты `pam_authenticated_printf` применяется сценарий `/etc/pam.d/other`
- Однако использование PAM позволяет конфигурировать правила для приложения аналогично стандартным утилитам:

```
$ sudo mkdir -p /var/opt/pam_authenticated_printf
$ echo "x" | sudo tee /var/opt/pam_authenticated_printf/loginusers
$ sudo nano /usr/lib/pam.d/pam_authenticated_printf
auth    required    pam_listfile.so onerr=fail item=user sense=allow file=/var/opt/pam_authenticated_printf/loginusers
@include common-auth
$ ./pam_authenticated_printf
Password:
Authenticated printf :-)
$ sudo –u user ./pam_authenticated_printf
Password:
pam_authenticate failed: Authentication failure
```

Источники

1. The Linux-PAM System Administrators' Guide
2. FreeBSD жив. Подключаемые Модули Аутентификации (PAM)
3. Документация RedOS. Подключаемые модули аутентификации (PAM). Общие сведения
4. RHEL docs. Sample PAM Configuration Files
5. https://man7.org/linux/man-pages/man8/pam.8.html
6. https://man7.org/linux/man-pages/man5/pam.d.5.html
7. https://man7.org/linux/man-pages/man8/pam_unix.8.html
8. https://linux.die.net/man/8/pam_pwquality
9. https://man7.org/linux/man-pages/man8/pam_time.8.html
10. https://man7.org/linux/man-pages/man5/time.conf.5.html
11. https://www.man7.org/linux/man-pages/man8/pam_limits.8.html
12. https://www.man7.org/linux/man-pages/man5/limits.conf.5.html
13. https://man7.org/linux/man-pages/man3/pam.3.html
14. Habr. Аудит входа в Linux через Slack. Разбираемся с PAM
15. Habr. Разработка и применение модуля PAM для аутентификации в Astra Linux с использованием Рутокен ЭЦП и Рутокен S
16. libpam. Examples