

## Сервис динамической аутентификации *PAM*

Сервис *PAM* (Pluggable Authentication Modules) — представляет собой систему динамической аутентификации пользователя в приложениях (или службах) в операционной системе *Linux* по средствам высокоуровневых *API* набора базовых библиотек.

Таким образом динамическая аутентификация с едиными *API* позволяет всем приложениям иметь политику аутентификации, настроенную системным администратором. При этом задача построения логики аутентификации снимается с разработчика приложений. Документация по *PAM* может быть найдена на сайте разработчиков, <http://www.kernel.org/pub/linux/libs/pam/>, а также например на ресурсе разработчиков *centOS* [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/html/system-level\\_authentication\\_guide/pam\\_configuration\\_files](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/pam_configuration_files).

Проверить наличие и версию *PAM* можно при помощи запроса

```
sudo rpm -qa | grep pam
```

Не все приложения используют *PAM*, использование данного сервиса не является обязательным требованием. Чтобы узнать, является ли программа «осведомленной о *PAM*» или нет, проверьте, скомпилирована ли она с библиотекой *PAM* с помощью команды *ldd*.

Синтаксис команды *ldd* предполагает полный путь к файлу. Если такой путь неизвестен, то можно воспользоваться утилитой *whereis*, например

```
whereis login
```

затем можно узнав полный путь можно проверить на совместимость с *PAM*

```
$ ldd /bin/login | grep libpam.so
```

```
libpam.so.0 => /lib64/libpam.so.0 (0xb7ef3000)
```

Конфигурационные файлы динамической настройки аутентификации находятся в */etc/pam.d/\** и ( в ряде систем также есть файл */etc/pam.conf*, но его конфигурирование это устаревший метод).

Следует отметить, что разработчики *centOS* рекомендуют использовать утилиту *authconfig* для внесения изменений в конфигурационные файлы *PAM*. При этом, хотя в большинстве случаев, неверные строки игнорируются, в некоторых модулях это может привести к ошибкам. Такие ошибки журналируются при помощи *journald* сервиса.

Проверки *PAM* включают четыре группы служб, организованных в виде очереди. Задействование тех или иных групп определяется запросами пользователя.

<i>auth</i>	Аутентификация пользователей, к примеру, когда нужно ввести пароль.
<i>account</i>	Управление учетными записями, например: проверка устаревания пароля и проверки по времени доступа. Когда пользователь идентифицирован с помощью модулей <i>auth</i> , модули <i>account</i> определяют, можно ли пользователю давать доступ.
<i>password</i>	Содержит функции, связанные с паролями, например обновление пароля.
<i>session</i>	Управление соединениями, например, журналирование входов в систему, журналирование выполненных действий или выполнение очистки по завершению сессии.

Каждое приложение *PAM* имеет свой конфигурационный, например,

**Задание 1:** посмотрите файл *setup*,

структура файла состоит из строк, описывающих последовательно (шаг за шагом) директивы (политики) аутентификации, вида:

<i>module_interface</i>	<i>control_flag</i>	<i>module_name</i>	<i>module_arguments</i>
название службы	флаг	имя модуля	доп. настройки

например:

```
[root@MyServer ~]# cat /etc/pam.d/setup
auth    sufficient    pam_rootok.so
auth    include       system-auth
account required     pam_permit.so
session required     pam_permit.so
```

В данном примере указывается на то, что, например служба *account* (Управление учетными записями) требует успешного завершения (флаг *required*) работы модуля *pam\_permit.so*. При этом модуль *pam\_permit.so* включает набор библиотек находящихся по стандартным путем, указанным в *libpam*.

Вызов каждого модуля заканчивается результатом *success* или *failure*. Контрольные флаги определяют, что нужно делать в этих случаях. Это, например, важно если модули последовательно связаны. Конфигурационные файлы *PAM* предполагают следующие виды флагов.

<i>required</i>	Этот модуль должен завершиться успешно для продолжения. При этом пользователю не сообщать о результате пока остальные модули не будут проверены. Если все модули помечены как <i>required</i> , тогда непрохождение любой из проверок будет означать отказ в доступе, хотя все другие модули в группе также будут исполнены.
<i>requisite</i>	Этот модуль должен завершиться успешно. Если модуль провален, то пользователю необходимо сообщить сразу.
<i>sufficient</i>	Модуль игнорируется, если провален. Если модуль завершился успешно и до него не было <i>required</i> модулей, которые провалены, то дальнейшей проверки не проводится.
<i>optional</i>	Результат работы игнорируется если он не единственный. Если в конфигурации нет модулей типа <i>required</i> или <i>sufficient</i> , то для разрешения доступа хотя бы один из модулей типа <i>optional</i> должен завершиться успешно.
<i>Include</i> <i>u</i> <i>substack</i>	Флаг подключения другого модуля.

В соответствии с выше сказанным рассмотренный выше файл *setup* подразумевает следующую логику аутентификации соответствующей утилиты (*setup*). Если аутентификация *pam\_rootok.so* закончилась успешно, команда выполняется, если нет, то проверка продолжается. При этом подключается библиотека */etc/pam.d/system-auth*; для учетных записей выполняется *pam\_permit.so* (разрешение на перезагрузку), затем производится попытка журналирования данного действия.

Следует отметить также, что помимо вышеописанных конфигурационные файлы PAM могут содержать дополнительную информацию, например, запись

*password requisite pam\_pwquality.so enforce\_for\_root retry=3*

обязывает проводить проверку пароля не более трех раз.

Вот также некоторые из распространенных дополнений.

<i>debug</i>	модуль передаёт дополнительную информацию в систему syslog для отладки.
<i>audit</i>	модуль передаёт расширенную информацию в систему syslog
<i>no_warn</i>	Запрещает передавать приложению предупреждающие сообщения.
<i>use_first_pass</i>	использовать пароль, полученный от предыдущего модуля. Если аутентификация прошла неудачно, попытки получить другой пароль не предпринимаются. Аргумент только для <i>auth</i> и <i>password</i>
<i>try_first_pass</i>	Аналогично предыдущему, но в случае провала будет запрошен другой пароль.
<i>likeauth</i>	Результат работы модуля не зависит от того проверяется пароль или задается новый.

Большинство модулей для конфигурирования *PAM* хранятся в каталогах */lib/security* и */lib64/security* (для 64-битных систем), а некоторые модули могут быть помещены в каталоге */usr/lib/security*. Также имеется возможность написания своих модулей.

Информацию по большинству модулей можно получить при помощи команды *man*. Например

```
man pam_pwcheck
```

Вот примеры распространённых модулей:

- *pam\_access*: разрешает или запрещает доступ, в зависимости от *IP*-адреса, имени пользователя, имени хоста или доменного имени и т.п. По умолчанию, правила доступа определены в файле */etc/security/access.conf*.
- *pam\_exec*: вызывает внешнюю программу.
- *pam\_limits*: устанавливает ограничения на системные ресурсы, используемые пользователем. Ограничения по умолчанию берутся из файла *etc/security/limits.conf*.
- *pam\_rootok*: разрешает доступ для пользователя *root* без дополнительных проверок.
- *pam\_unix* или *pam\_unix2*: классическая аутентификация в *UNIX*-стиле, основана на файлах */etc/passwd* и */etc/shadow*.
- *pam\_userdb*: аутентифицирует пользователя с помощью базы данных. См. также модуль *pam\_unix*.
- *pam\_wheel*: позволяет *root*-доступ лишь для членов группы *wheel*.

## Задание 2:

Для понимания что такое динамическое конфигурирование аутентификации рассмотрим модуль *su* (режим суперпользователя).

1. Убедитесь, что модуль *su* поддерживает динамическую аутентификацию.
2. Перейдите в режим *su* при помощи запроса

```
$ su —
```

Запрос должен потребовать у вас введения пароля.

3. Выйдите из данного режима при помощи запроса:

```
$ su — имя_пользователя.
```

4. Найдите информацию по модулю *pam\_permit*
5. Включите в начало файла */etc/pam.d/su* (в стеке *auth*) строку:

```
auth sufficient pam_permit.so
```

Для этого можно использовать редактор *vi*, режим редактирования включится клавишей *i*, выход *Esc*, режим команд: команда сохранения *w*, команда выхода *q*.

6. Зайдите в систему как обычный пользователь и проверьте, что можете переключиться в режим *root* с помощью *su* без ввода пароля.
7. Верните систему в исходное состояние.

**Задание 3:** запретите всем пользователям команду *su*:

1. Закомментируйте все строки в файле */etc/pam.d/su*, внесите строку:  
*auth requisite pam\_deny.so*
2. Зайдите в систему как обычный пользователь и убедитесь, что переключения в режим *root* не происходит.
3. Верните систему в исходное состояние.

**Задание 4:** настройка проверки качества пароля при помощи модуля *pam\_pwquality* :

1. Добавьте в файл */etc/security/pwquality.conf* проверку пароля на наличие не менее 2 цифр и запомним 3 старых пароля в файле */etc/security/opasswd* и введем проверку на различия как минимум 3 символов в новом и старом паролях:  
*dcredit = 2 remember = 3 difok = 3*
2. Добавьте в файл */etc/pam.d/system-auth* следующую строку:  
*password required pam\_pwquality.so retry=3*
3. Попробуйте изменить свой пароль, какова его минимальная длина. Сколько попыток даётся пользователю?
4. Попробуйте настроить пароль так, чтобы там была не менее 2 цифр и 2 больших букв и 2 символов, при этом общая длина была не менее 8 символов.
5. Верните систему в исходное состояние.

**Задание 5:**

1. Добавьте проверку пароля на соответствие списку *pam\_cracklib* – что при этом изменилось?