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.

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.

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

Каждое приложение *PAM* имеет свой конфигурационный, например,

Задание: посмотрите файл *setup,*

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

module_interface control_flag module_name module_arguments

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.

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

Следует отметить также, что помимо вышеописанных конфигурационные файлоы *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.

debug	модуль передаёт дополнительную информацию в систему syslog для отладки.
audit	модуль передаёт расширенную информацию в систему syslog
no_warn	Запрещает передавать приложению предупреждающие сообщения.
use_first_pass	использовать пароль, полученный от предыдущего модуля. Если аутентификация прошла неудачно, попытки получить другой пароль не предпринимаются. Аргумент только для auth и password
try_first_pass	Аналогично предыдущему, но в случае провала будет запрошен другой пароль.
likeauth	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 1: 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 2: запретите всем пользователям команду su:

1. Закомментируйте все строки в файле */etc/pam.d/su,* внестие 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 3: 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. Попытайтесь изменить свой пароль, какова его минимальная длина. Сколько попыток даётся пользователю?
4. Попытайтесь 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 4: Добавьте проверку пароля на соответствие списку *pam_cracklib* – что при этом изменилось?

Смотрите больше интересных примеров конфигурирования файлов тут

https://github.com/efanov/mephi/wiki/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0-%22%D0%98%D0%B7%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-PAM%22