OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

PAM

Необходимо решить задачу по ограничению доступа пользователей в систему по ssh. Это будут пользователи: “**day**“**,** “**night**”**,** “**friday**”. Введем для них соответственно ограничения:

- ”**day**” - имеет удаленный доступ каждый день с 8 до 20;
- “**night**” - с 20 до 8;
- “**friday**” - в любое время, если сегодня пятница.

На стендовой виртуальной машине создадим 3х пользователей:

```
[vagrant@host ~]$ sudo useradd day && \
sudo useradd night && \
sudo useradd friday
```

Назначим им пароли:

```
[vagrant@host ~]$ echo "Otus2019"|sudo passwd --stdin day &&\
echo "Otus2019" | sudo passwd --stdin night  &&\
echo "Otus2019" | sudo passwd --stdin friday
```

Чтобы быть уверенными, что на стенде разрешен вход через ssh по паролю выполним:

```
[vagrant@host ~]$ sudo bash -c "sed -i
's/^PasswordAuthentication.*$/PasswordAuthentication yes/'
/etc/ssh/sshd_config && systemctl restart sshd.service"
```

Теперь стенд готов к работе

PAM

PAM (Pluggable Authentication Modules - подключаемые модули аутентификации) - это набор библиотек, которые позволяют интегрировать различные методы аутентификации в виде единого API, что позволяет предоставить единые механизмы для управления, встраивания прикладных программ в процесс аутентификации. PAM решает следующие задачи:

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.

Таким образом для решения задачи необходимо на первом или втором этапе применить необходимые нам проверки. Их можно реализвать несколькими способами. Рассмотрим их.

PAM. Модуль pam_time

Модуль ***pam_time*** позволяет достаточно гибко настроить доступ пользователя с учетом времени. Настройки данного модуля хранятся в файле ***/etc/security/time.conf***. Данный файл содержит в себе пояснения и примеры использования. Добавим в конец файла строки:

```
*;*;day;Al0800-2000
*;*;night;!Al0800-2000
*;*;friday;Fr
```

Разные параметры отделяются символом ";". Разберем первую строку:

- “*” сервис, к которому применяется правило
- "*" имя терминала, к которому применяется правило
- имя пользователя, для которого данное правило будет действовать
- время, когда правило носит разрешающий характер

PAM. Модуль pam_time

Теперь настроим PAM, так как по-умолчанию данный модуль не подключен.
Для этого приведем файл **/etc/pam.d/sshd** к виду:

```
...
account    required     pam_nologin.so
account    required     pam_time.so
...
```

После чего в отдельном терминале можно проверить доступ к серверу по ssh для созданных пользователей.

PAM. Модуль pam_exec

Еще один способ реализовать задачу это выполнить при подключении пользователя скрипт, в котором мы сами обработаем необходимую информацию.

Удалим из **/etc/pam.d/sshd** изменения из предыдущего этапа и приведем его к следующему виду:

```
...
account  required   pam_nologin.so
account  required   pam_exec.so  /usr/local/bin/test_login.sh
...
```

Мы добавили модуль **pam_exec** и, в качестве параметра, указали скрипт, который осуществит необходимые проверки. Создадим сам скрипт. Пример на следующем слайде, а готовый можно взять по ссылке: *gist test_login.sh*

PAM. Модуль pam_exec

```
[day@host ~]$  cat <<'EOF' >> /usr/local/bin/test_login.sh
#!/bin/bash

if [ $PAM_USER = "friday" ]; then
  if [ $(date +%a) = "Fri" ]; then
      exit 0
    else
      exit 1
  fi
fi

hour=$(date +%H)

is_day_hours=$(($(test $hour -ge 8; echo $?)+$(test $hour -lt 20; echo $?)))

if [ $PAM_USER = "day" ]; then
  if [ $is_day_hours -eq 0 ]; then
      exit 0
    else
      exit 1
  fi
fi

if [ $PAM_USER = "night" ]; then
  if [ $is_day_hours -eq 1 ]; then
      exit 0
    else
      exit 1
  fi
fi
EOF
```

PAM. Модуль pam_exec

При запуске данного скрипта PAM-модулем будет передана переменная окружения **PAM_USER**, содержащая имя пользователя. Скрипт содержит простую логику. Если имя пользователя **friday**, то проверям день недели, если пятница, то возвращаем 0, если нет, то 1 и завершаем скрипт.

Если же указан другой пользователь, то в строке

```
is_day_hours=$(($(test $hour -ge 8; echo $?)+$(test $hour -lt 20; echo $?)))
```

происходит проверка принадлжит ли текущее значение времени (переменная **hour**) диапазону от 8 до 20 часов. Если да, то **is_day_hours** примет значение 0, если нет 1. Дальше проверяем имя пользователя и соотвествие ему. Если пользователь **day** и часы "дневные", то возвращаем 0, если пользователь **night** и часы НЕ дневные, то так же возвращаем ноль. В противном случае скрипт вернет 1. Если в **PAM_USER** указано какое-то другое имя пользователя, то скрипт вернет 0.

На основании кода завершения скрипта модуль **pam_exec** принимает решение. Если вернулся 0, то все в порядке и пользователь будет авторизован, в обратном случае нет.

PAM. Модуль pam_script

Данный модуль не входит, как предыдущие в базовую систему и должен быть установлен из отдельного репозитория **Extra Packages for Enterprise Linux (EPEL)**. Подключим репозиторий и установим **pam_script**:

```
[vagrant@host ~]$for pkg in epel-release pam_script; do yum install -y $pkg; done
```

Так же как и **pam_exec** модуль предназначен для выполнения произвольного скрипта в процессе авторизации, аутентификации или аккаунтинга пользователя.
По сравнению с предыдущим примером в файле **/etc/pam.d/sshd** нужно просто переименовать **pam_exec** в **pam_script**:

```
...
account  required  pam_nologin.so
Account  required  pam_script.so  /usr/local/bin/test_login.sh
...
```

Скрипт, который будет использоваться можно взять из предыдущего примера.

PAM. Модуль pam_cap

Для демонстрации работы модуля установим дополнительный пакет **nmap-ncat(CentOS)**. Пользователь **day** остался из предыдущего примера. Если стенд пересоздавался, то пользователя так же необходимо создать.

Войдем на стендовую машину под пользователем **day** и попробуем выполнить команду **nc** и получим сообщение об ошибке. Пример ниже.

```
[day@host ~]$ ncat -l -p 80
Ncat: bind to :::80: Permission denied. QUITTING.
```

Это связано с тем, что непривелигерованный пользователь **day**, от имени которого выполняется команда, не может открыть для прослушивания 80й порт.

PAM. Модуль pam_cap

Эту задачу можно решить несколькими способами:

- установить suid-бит. Установка данного бита позволит выполнить **ncat** так, будто он запущен от root. Способ имеет низкую гибкость, так как установка бита позволит **любому** пользователю выполнить команду;
- предоставить пользователю права (возможности), чтобы он смог открыть порт. Способ более гибкий, потому что можно указать что именно, кому и при помощи какой программы мы разрешаем;

Решим задачу вторым способом. Для этого воспользумся pam-модулем **pam_cap**. Поскольку это демо стенд, то **SELinux** можно просто выключить выполнив

```
[vagrant@host ~]$sudo setenforce 0
```

Отключать SELinux в продакшене крайне не рекомендуется.

PAM. Модуль pam_cap

Приведем файл **/etc/pam.d/sshd** к виду:

```
...
auth       include      postlogin
auth       required     pam_cap.so
...
```

Таким образом мы включили обработку **capabilities** при подключении по **ssh**. Пропишем необходимые права пользователю **day**. Для этого создадим файл **/etc/security/capability.conf** содержащий одну строку:

```
cap_net_bind_service     day
```

Теперь необходимо программе (**/usr/bin/ncat**), при помощи которой будет открываться порт, так же выдать разрешение на данное действие:

```
[vagrant@host ~]$ sudo setcap cap_net_bind_service=ei
/usr/bin/ncat
```

Мы сопоставили права, выданные пользователю с правами выданными на программу. Снова зайдем на стенд под пользователем **day** и проверим, что мы получили необходимые права:

```
[day@host ~]$ capsh --print
Current: = cap_net_bind_service+i
```

Теперь попробуем выполнить команду:

```
[day@host ~]$ ncat -l -p 80
```

Теперь ошибки не возникло. Теперь можно открыть еще одну консоль и там выполнить:

```
[day@host ~]$ echo "Make Linux great again!" > \
/dev/tcp/127.0.0.7/80
```

и увидеть сообщение в перовой консоли.

Права администратора

Помимо внесения ограничений на вход пользователя в систему, мы так же можем предоставить выбранному пользователю разные права. Для примера рассмотрим предоставление прав root'а определеному пользователю в системе. Обычно для этого используются следующие варианты:

- пользователь заносится в группу **wheel**;
- для него создается отдельный файл в **/etc/sudoers.d/**;
- отдельная строка в **/etc/sudoers**.

Первый способ реализуется очень просто. Зайдя в систему под root'ом нужно выполнить:

```
usermod -G wheel day
```

Теперь зайдя в систему под пользователем **day** можно выполнить команду **sudo -i** и получить консоль пользователя **root**. При этом будет запрошен пароль того пользователя, под которым осуществлен вход в систему (в данном случае **day**).

Права администратора

Последующие два способа сводятся к добавлению строки:

```
day   ALL=(ALL)       ALL
```

или

```
day         ALL=(ALL)       NOPASSWD: ALL
```

Первый вариант по-прежнему будет спрашивать пароль текущего пользователя, а второй нет.
Одну из приведенных выше строк необходимо добавить к файлу **/etc/sudoers** и тогда пользователь **day** получит возможность выполнить “**sudo**”.

Вариант с созданием файла **/etc/sudoers.d/day** и добавлением строки в него является более гибким и удобным.