Network Security

Project 3

ARP Spoofing via Software Exploit

Instructor: Shiuhpyng Shieh

TA： E-Lin Ho, Jui-Chien Jao

1. Description

In this project, you need to do ARP spoofing and exploit the Format-String(FS) and Buffer-Overflow(BOF) vulnerabilities in the target machine.

Recall that in project 2.1 an assumption that the network traffic between Alice and Bob has been hijacked via ARP spoofing. However, in real cases, an attacker may need to firstly compromise a device/server to fulfill the network hijacking. Thereby, in this project, you are required to compromise the target machine before the network can be ARP spoofed.

The purpose of this project is to let you learn the causes of common software vulnerabilities, FS and BOF, and how to do ARP spoofing.

The scenario is depicted as Fig. 1.

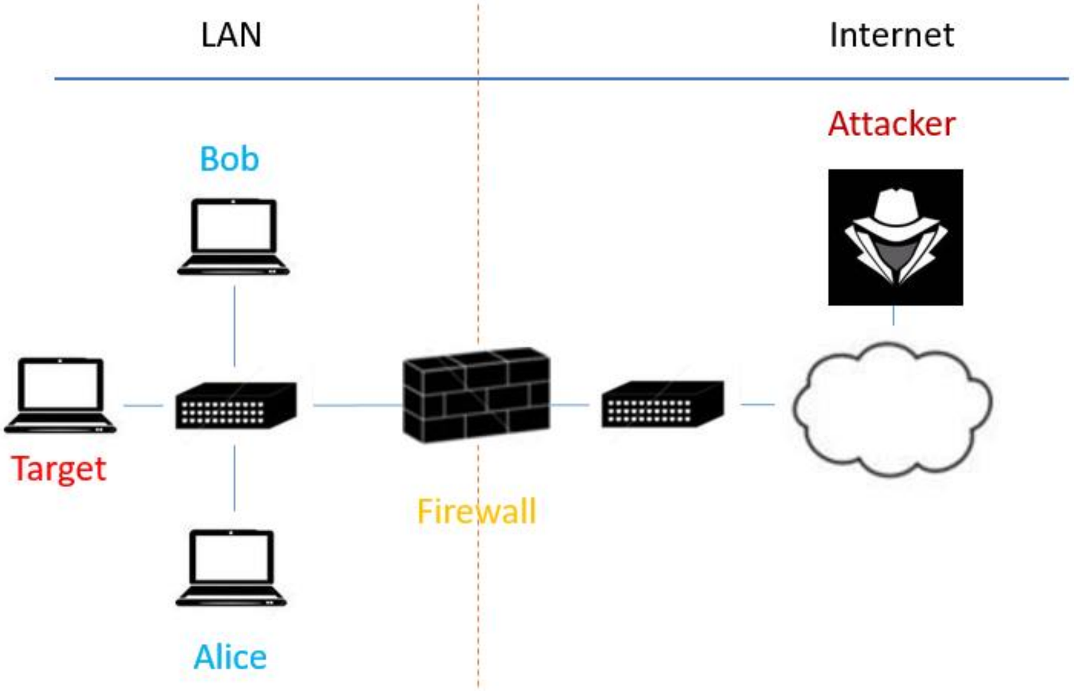


Fig. 1 The scenario of project 2.2

Part 1 -- Software Exploit

In this part, you need to exploit the program executing on the target machine, and get the control of the machine.

There are software vulnerabilities including FS and BOF in the program. You need to know how they make programs vulnerable and exploit them in order to get the shell of the machine. After that, you will see six files in the current directory, including the binary of the program exploited by you.

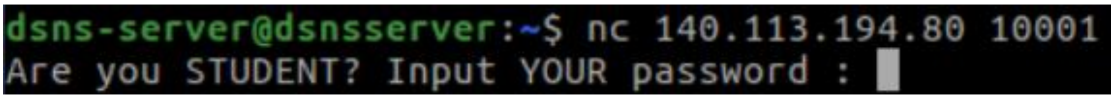
※ More information in [Reference]

※ Server running on **Ubuntu 16.04 32bits**

※ Find your target from here, don’t attack others’ machine

- https://docs.google.com/spreadsheets/d/1BKeSjwcLVmMNhZhJpUZVkETlwWY899kdYgF6j9sotjM/edit?usp=sharing

STEP 1 – format string vulnerability



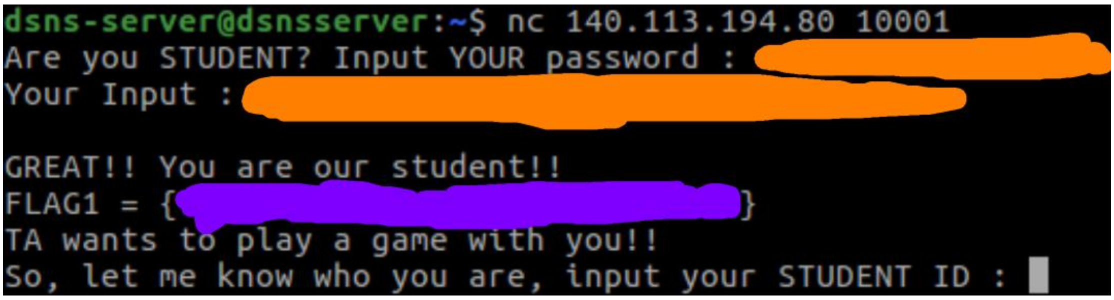
```
dsns-server@dsnsserver:~$ nc 140.113.194.80 10001
Are you STUDENT? Input YOUR password : 
```

When you access in server, you will see this line and need using password to login, but we won’t give you password, you need to exploit it to get one. The password doesn’t be stored in this file, but you can leak memory information on stack to get the password.

Exploit format string vulnerability can arbitrary read memory on stack or arbitrary write to stack. In this project, you can focus on arbitrary read format string vulnerability.

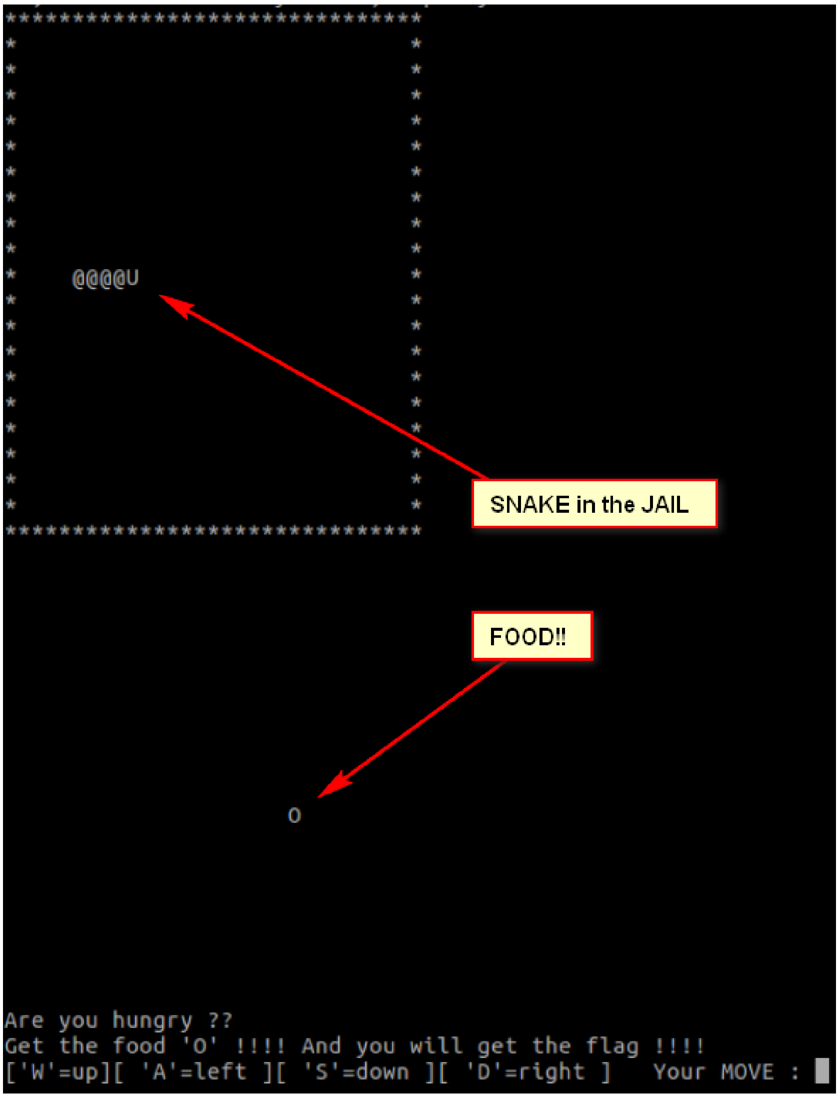
Using `%x`、`%N$x`、`%s`、`%N$s` to finish it.

STEP 2 – buffer overflow



```
dsns-server@dsnsserver:~$ nc 140.113.194.80 10001
Are you STUDENT? Input YOUR password : [redacted]
Your Input : [redacted]

GREAT!! You are our student!!
FLAG1 = {[redacted]}
TA wants to play a game with you!!
So, let me know who you are, input your STUDENT ID : 
```

After Step 1, you will get the first flag.

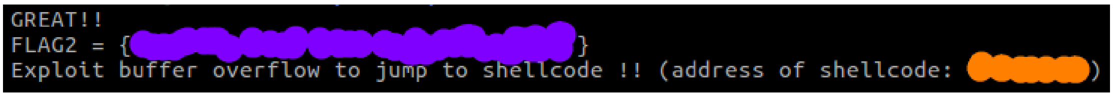


```
*******************************
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*    @@@@U                    *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*                             *
*******************************










                      O









Are you hungry ??
Get the food 'O' !!!! And you will get the flag !!!!
['W'=up][ 'A'=left ][ 'S'=down ][ 'D'=right ]   Your MOVE :
```

Step 2 you need to play a game “SNAKE”, the goal is the food out of the jail. Move to eat that, and you will get the second flag.

Can you eat that?

STEP 3 – buffer overflow and using shell code



```
GREAT!!
FLAG2 = {[redacted]}
Exploit buffer overflow to jump to shellcode !! (address of shellcode: [redacted])
```

There is a global string stored shell code.

In order to reduce the working what you need to do, TA provides the global string’s address. So you can jump to the address and the program will execute the code (/bin/sh)

After you get the shell, we strongly suggest you to add `2>&1` to the back of all commands, or you would lose important messages from the system.

Part 2 -- ARP Spoofing

In this part, you need to do ARP spoofing so as to hijack messages in the LAN. After you dominate the machine, you can use `*nmap*` to find that there are two other users in the LAN, Alice and Bob. Alice maintains a service on a specific port and she will send the magic number in flag to Bob after getting a student ID no matter who sends. You may need to use the command, `nc`, to connect to Alice, but before ARP spoofing, you will get WRONG flag!!

STEP 1 – nmap

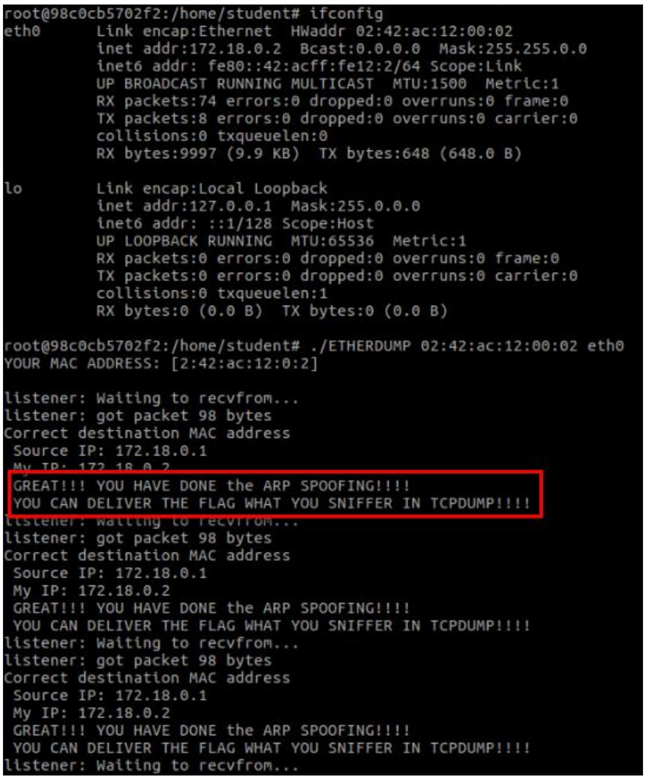
Usually, the first step to attack on internet is finding what service running on which port on which computer.

you can achieve this aim by using command `nmap` to scan network , including

1) Scan computers in specified subnet
2) Scan opened ports on specified computer

STEP 1 – arping

You may need to use the command, `*arping*`, to finish ARP spoofing in order to falsify Alice's ARP table. Finally, if you do it correctly, Alice will send the secret message to you instead of Bob after you send your student ID to her.



```
root@98c0cb5702f2:/home/student# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:12:00:02
          inet addr:172.18.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
          inet6 addr: fe80::42:acff:fe12:2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:74 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9997 (9.9 KB)  TX bytes:648 (648.0 B)

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:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@98c0cb5702f2:/home/student# ./ETHERDUMP 02:42:ac:12:00:02 eth0
YOUR MAC ADDRESS: [2:42:ac:12:0:2]

listener: Waiting to recvfrom...
listener: got packet 98 bytes
Correct destination MAC address
 Source IP: 172.18.0.1
 My IP: 172.18.0.2
 GREAT!!! YOU HAVE DONE the ARP SPOOFING!!!!
 YOU CAN DELIVER THE FLAG WHAT YOU SNIFFER IN TCPDUMP!!!!
listener: Waiting to recvfrom...
listener: got packet 98 bytes
Correct destination MAC address
 Source IP: 172.18.0.1
 My IP: 172.18.0.2
 GREAT!!! YOU HAVE DONE the ARP SPOOFING!!!!
 YOU CAN DELIVER THE FLAG WHAT YOU SNIFFER IN TCPDUMP!!!!
listener: Waiting to recvfrom...
listener: got packet 98 bytes
Correct destination MAC address
 Source IP: 172.18.0.1
 My IP: 172.18.0.2
 GREAT!!! YOU HAVE DONE the ARP SPOOFING!!!!
 YOU CAN DELIVER THE FLAG WHAT YOU SNIFFER IN TCPDUMP!!!!
listener: Waiting to recvfrom...
```

You can use `tcpdump` to get (1) IP Address (2) MAC Address (3) Package data.

Every time you give Alice student ID, Alice will send message to Bob's IP, but only finished ARP spoofing, you will get the correct one. Make sure three things about Alice's packages : (1) destination IP address is Bob's IP (2) destination MAC address is server's MAC address (3) in TCP data, there will be a message "FLAG:{ xxxxxxxxxx }"

Another way to check whether you done the ARP spoofing, you can execute the file 'ETHERDUMP' which is under the same directory. After ARP spoofing is done, when you give Alice your student, and run the Ethernet package dump, you can see the message "GREAT!!!".

The message in ETHERDUMP is not flag, the flag is still in data frame of `tcpdump`, and you need to verify the destination IP address is Bob's IP.

Notice

1. Using `arping` to do ARP spoofing, you need to modify a file under /proc/ , but in TA's system, you have no privilege to overwrite it. There is a directory: /writable-proc/ , and you can overwrite files under the directory.
2. If you make the machine crash, please wait 4 hours or mail to TA.
3. Do Not scan other computers in Campus with TA's server.

Flags

There will be four flags in this project. You can get three of them if you do FS and BOF exploits correctly, and the last one is in the magic file Alice sends to you.

The format of all flags ： FLAG = {xxxxxxxxxx}

Hints

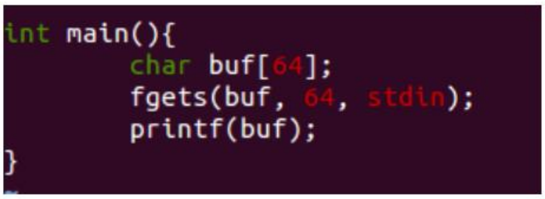
- After you reverse the binary of the program executing on the target machine, you can first read functions named by strings beginning with
  "PUZZLE_".
- You can ignore functions by strings beginning with
  "NEED_NOT_TO LOOK_".
- Please be care for the newline('\n') while you exploit the program.

2. Reminder

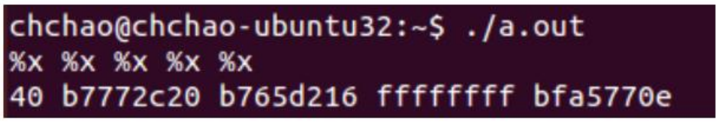
1) Binary server of the program executing on target machine is provided by TA's server. So you can't patch it or using `*gdb*` on server.
   But TA will provide the binary code, you can see the assembly code.
2) The server is running on Ubuntu 16.04 32bits. Maybe you need own your one.
3) In Network Security Project 3.zip, all of them need to put in same folder with the project3 binary.
   The 3 Flags and Password are not the same with what you need to deliver, that just a sample for you
4) Record the flags you get
   - Four flags
     i. Attack format string – 1flag
     ii. Attack buffer overflow to get shell – 2 flags
     iii. Do ARP spoofing and get from Alice – 1flag
5) You may need to implement some programs to exploit the binary.
6) You only need to use `*nc*` command to connect to Alice after you get shell.
7) Before ARP spoofing, remember to modify file in /writable-proc/ somewhere
8) Server will be restored every 4 hours (e.g. 12:00 PM, 16:00PM…)

3. Reference

1) Format string vulnerability



```
int main(){
        char buf[64];
        fgets(buf, 64, stdin);
        printf(buf);
}
```



```
chchao@chchao-ubuntu32:~$ ./a.out
%x %x %x %x %x
40 b7772c20 b765d216 ffffffff bfa5770e
```

- https://www.exploit-db.com/docs/28476.pdf
- https://hackmd.io/CwdgZsCGAMDGYFprAEwE4HDotAjArABwICmIsAjLLGmigCb41A==?view

2) Buffer overflow vulnerability

- https://en.wikipedia.org/wiki/Buffer_overflow
- https://hackmd.io/CwdgZsCGAMDGYFprAEwE4HDotAjArABwICmIsAjLLGmigCb41A==?view

3) *Important shell command*

   1) *Objdump [-d][-D][-S][-j]*
   2) *ifconfig*
   3) *nmap*
   4) *arping*
   5) *tcpdump*
   6) *nc*

using command `man` to get help information, or Google it !

4) *You can practice with*

Network Security Project Software Vulnerability_UPDATE_1.zip

which TA present last week.

4. Deliverables

Each student must work individually and submit **a compress file named by your student ID, e.g., “<YOUR_STUDENT_ID.zip>”** containing:

a) Source files named by your student ID of:

- ◆ The program that you exploit the machine
  - ● As far as possible writing comments in code
  - ● Comments source website of reference code

b) A report, text, Word or PDF, named by your student ID includes descriptions of:

- ◆ The 4 flags you get
- ◆ You need to record
  - ● Server IP address / MAC address
  - ● Alice IP address / MAC address
- ◆ The way that you finish this project
  - ● How you find the vulnerability and how to attack
- ◆ Brief introduction what you learned
- ◆ Suggest for TA

5. Warning - **STRICTLY PROHIBITED**

1) Every student has his/her own server, don’t access or attack others’ server
2) After you get shell, do not to access address 172.18.X.1, the address owned by firewall, don’t access it.
3) Be careful, don not use command `nmap` to scanning computer in campus, or NCTU computer center will lock your computer’s IP
4) Copy or piracy is strictly prohibited

If you have any question, please contact TA as soon as possible,
Or you can ask on discussion on E3.
Any anomaly connection such DDoS will be traced for penalty.
Deadline: 2016/12/29 (Thur.) 23:59:59