实验 - Linux 实验环境初探

实验简介

FSB (Format String Bug) is just so powerful

基础部分

fsb复用地址修改全局数据为特定值 30分

```
// gcc -g -m32 -no-pie -fno-stack-protector hw1.c -o hw1
// ./hw1

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

char key_in_global[32] = "verysecure";

int main(int argc, char* argv[])
{
  char *keyptr = key_in_global;
  char buffer[512] = {0};

  printf("before fsb, key: %s\n", keyptr);

  scanf("%s", buffer);
  getchar();
  printf(buffer);

  printf("after fsb, key: %s\n", keyptr);

  return 0;
}
```

要求使用fsb将全局变量 `key_in_global` 修改为 `Verysecure`

要求报告中给出 exploit 代码并截图输出

fsb构造地址修改全局数据为特定值 30分

```
// gcc -g -m32 -no-pie -fno-stack-protector hw2.c -o hw2
// ./hw2

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

char key_in_global[32] = "verysecure";

int main(int argc, char* argv[])
{
  char buffer[512] = {0};

  printf("before fsb, key: %s\n", key_in_global);

  scanf("%s", buffer);
  getchar();
  printf(buffer);

  printf("after fsb, key: %s\n", key_in_global);

  return 0;
}
```

要求使用fsb，并自行布局地址，将全局变量 `key_in_global` 修改为 `verysecure?`

要求报告中给出 exploit 代码并截图输出

fsb构造地址劫持 exit GOT 表到后门 40分

```
// gcc -g -m32 -no-pie -z lazy -fno-stack-protector hw3.c -o hw3
// ./hw3

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

int main(int argc, char* argv[])
{
  char padding1[16] = {'a'};
  char buffer[512] = {0};
  char padding2[16] = {'b'};

  scanf("%s", buffer);
  getchar();
  printf(buffer);
```

```
  exit(0);
}

void backdoor(void)
{
  printf("Hi Backdoor!\n");
  system("/bin/sh");
}
```

要求使用fsb，并自行布局地址，将 exit GOT 表项覆盖为 backdoor 函数地址，成功运行后拿到shell，并任意执行一个命令（pwd, id）截图证明

挑战部分

- fsb 构造多次机会
  - zjusec chance: https://zjusec.com/challenges/35
- non-stack format string
  - fsb heap: https://zjusec.com/challenges/77

拓展问题和阅读

参考课堂上的推荐