实 验 报 告

课程名称 《网络攻击与防御技术》

学　　院 计算机科学技术学院

专　　业 信息安全

姓　　名 黄　　力

学　　号 **15307130275**

开 课 时 间 **2018** 至 **2019** 学年第 **1** 学期

实验项目名　　称	Linux 栈溢出远程利用	成绩	

一、实验目的

（1）了解 C 语言程序中函数调用时程序栈的变化情况和栈溢出的基本原理

（2）通过实验掌握如何使用远程栈溢出漏洞程序获取远程服务器上带有 root 权限的 shell

（3）熟悉一些基本的 linux 命令，了解 linux 下的编程和调试基本工具。

二、实验内容

（1)在非 root 用户下编写 exploit 程序连接运行在目标机器 5050 端口的 echo_server 程序，该 echo_server 程序由 echo_server.c 编译得到，有栈溢出漏洞并具备 suid 标志位属性。通过 socket 向该程序发送带有 shellcode 的 buffer 进行溢出攻击，获取目标机上具有 root 权限的 shell

（2）分析实验成功或失败的原因

三、实验环境

（1）PC 机操作系统：macOS Mojave 10.14

（2）虚拟机操作系统（Parallels Desktop 13.1.1）：32 位 redhat3

四、实验原理

通过阅读 echo_server 程序的源代码 echo_server.c 可以发现该程序的主要功能是绑定本机的 5050 端口进行 tcp 通信，把从客户端发送过来的字符串再发送回客户端。其中将 recv_buf 拷贝到 send_buf 时用到了 c 语言中的 sprintf 函数，存在栈溢出漏洞。在本实验中，可以利用该栈溢出漏洞构造 shellcode 填充到程序栈中，并修改 sprintf 函数执行完后的返回地址至 shellcode 的起始地址使得程序从 sprintf 函数返回后继续执行构造的 shellcode 从而达到攻击目的。

五、实验步骤及结果

（1）安装虚拟机：

从云复旦 http://cloud.fudan.edu.cn/shareFolder/466220002/UHWpvrr 中下载 redhat3.rar，解压并利用其中的虚拟硬盘在 Parallels Desktop 安装 redhat 操作系统获得实验环境，使用 hacker 作为登入帐号（无密）登入，登入目录为/home/hacker，在此目录中已有编译好的具备 suid 标志位的 echo_server 程序。可使用 ls -l echo_server 命令查看，结果如下图：

（2）阅读源码，寻找溢出漏洞存在的地方：

阅读 echo_server.c 源码文件，main 函数中无溢出点。在其调用的函数中，echo_sever 函数中分别构造了大小为 200 的 send_buf 和大小为 2000 的 recv_buf。并且 echo_server 函数中有两处调用了 sprintf 函数，如下图所示：左图是第一次调用，右图是第二次调用。第一次调用拷贝的是定长字符串，不会产生溢出。第二次调用拷贝的 rcv_buf 大小（2000）远大于 send_buf（200），所以溢出漏洞在第二个 sprintf 函数上。

```
memset(recv_buf,0,sizeof(recv_buf));
printf("Now recv...\n");
sprintf(send_buf,"Input string, input 'end' to finish\n");
len = send(connfd,send_buf,strlen(send_buf),0);
```

```
printf("Now send...\n");
memset(send_buf,0,sizeof(send_buf));
sprintf(send_buf,"We send back the client input:  %s",recv_buf);
len = send(connfd,send_buf,strlen(send_buf),0);
```

（3）编写 exploit 程序：

使用 C 语言编写 exploit 程序 echo_server_exploit.c 对 echo_server 程序进行栈溢出攻击。

此程序的大致逻辑是构造需要发送给目标机的 buffer，然后创建套接字连接目标机并向 echo_server 运行的 5050 端口发送构造好的 buffer。下图是构造 buffer 部分的代码：

```
char buf[1600];
int s, i, size, ret;
struct sockaddr_in remote;
struct hostent *host;
sscanf(argv[3], "0x%x", &ret);
memset(buf, 0x90, 1600);
memcpy(buf + 1600 - sizeof(shellcode) - 2, shellcode, sizeof(shellcode));
for ( i = 0; i < 90 * 4; i += 4)
{
    *((int *)) &buf[i] = ret;
}
```

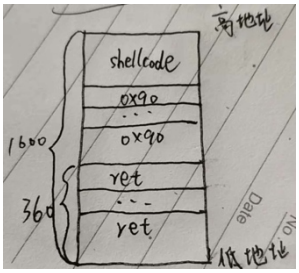
1、大小为 1600 的 buf 用于构造带有 shellcode 的发送内容，此处需要注意的是 buf 的大小必须是 4 的倍数（内存对齐原则，下同）。

2、调用 sscanf 函数从命令行参数中的第三个参数获取修改后的返回地址，此地址根据 echo_server 给的 esp 值计算。

3、调用 memset 函数将 buf 的值全部初始化为 0x90，即 nop 指令（程序执行此指令时什么也不做，并继续接下来的指令）。

4、调用 memcpy 函数将 shellcode 拷贝到 buf 末尾的 sizeof（shellcode）个字节中。

5、使用一个 for 循环将 buf 前 360 个字节赋值为返回地址（ret），此步需要注意的是覆盖的范围应该大于 200（echo_server 中 send_buf 的大小），还需要注意的就是这个值也需要是 4 的倍数。由此 buf 中的内容设置完毕，其布局如下图所示：

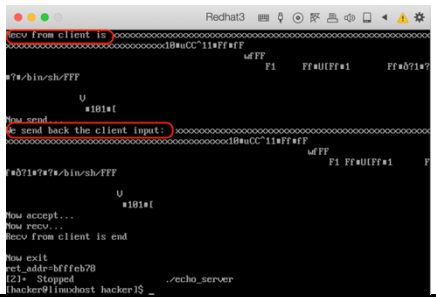


6、创建 tcp 套接字连接远程服务器并将 buf 内容发送给服务器的 5050 端口。

（4）编译运行 echo_server_exploit.c 获取 root 权限的 shell 并验证：

使用 gcc -o echo_server_exploit echo_server_exploit.c 命令编译得到攻击程序 echo_server_exploit，然后使用./echo server &命令后台运行 echo_server 程序。得到 esp 值为：0xbfffea78，如下图。

最后使用./ echo_server_exploit 127.0.0.1 5050 0xbfffeb78（0xbfffea78 + 0x100（4 的倍数））命令运行攻击程序，结果如下图：



从上图中可以看到数据的发送和接收是正常的：服务端收到客户端的数据并发送回去。exploit 程序没有成功拿到 shell，失败原因应该是新的返回地址没有计算正确。

六、实验总结

通过本次实验，我了解了远程栈溢出的基本原理，并尝试编写 exploit 程序利用漏洞获得了远程目标机上具有 root 权限的 shell。但是最后没有成功。本次实验的难点主要有两个：一是新返回地址的计算，即 exploit 程序中的第三个参数值。二是需要将 echo_server 程序后台运行，解决方法是在./echo_server 命令末尾加一个&。

七、参考资料

1、https://blog.csdn.net/tyskfs2/article/details/42318531