汇编实验

实验内容

- BombLab
- BufLab

BombLab背景

- 任务就是要破解一批二进制炸弹。
- 每个人的炸弹都**不一样**。

BombLab代码（部分公布）

```
/* Hmm...  Six phases must be more secure than one phase! */
input = read_line();              /* Get input                    */
phase_1(input);                   /* Run the phase                */
phase_defused();                  /* Drat!  They figured it out!
                    * Let me know how they did it. */
printf("Phase 1 defused. How about the next one?\n");

/* The second phase is harder.  No one will ever figure out
 * how to defuse this... */
input = read_line();
phase_2(input);
phase_defused();
printf("That's number 2.  Keep going!\n");

/* I guess this is too easy so far.  Some more complex code will
 * confuse people. */
input = read_line();
phase_3(input);
phase_defused();
printf("Halfway there!\n");
```

BombLab实验

- Phase1：stringcomparison
- Phase2：loops
- Phase3：conditionals/switches
- Phase4：recursive calls and the stack discipline
- Phase5：pointers
- Phase6：linked lists/pointers/structs

BombLab背景

- 破解方法，即向stdin输入正确的字符串序列。

Terminal

File Edit View Search Terminal Help

```
lovfu@linuxmint ~/PPT $ ./bomb-quiet
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
hello

BOOM!!!
The bomb has blown up.
lovfu@linuxmint ~/PPT $ ./bomb-quiet < solution.txt
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Phase 1 defused. How about the next one?
That's number 2.  Keep going!
Halfway there!
So you got that one.  Try this one.
Good work!  On to the next...
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Wow! You've defused the secret stage!
Congratulations! You've defused the bomb!
lovfu@linuxmint ~/PPT $
```

BufLab背景

- 本次试验，将利用**缓冲区溢出漏洞**攻击一个名为BufBomb的程序。
- 每个人所破解的BufBomb是不同的。
- 根本原因：栈帧结构的特性。
- 直接原因：字符串越界。

缓冲区溢出漏洞原理

```
ESP==>|        :        |
      |        .        |
      +-------------------------+
      | 被调用者保存的寄存器现场  |
      | EBX，ESI和EDI（根据需要）|
      +-------------------------+
      | 临时空间            |
      +-------------------------+
      | 局部变量#2
      +-------------------------+
      | char    #1          | [EBP - n]
      +-------------------------+
EBP==>| 调用者的EBP       |     4
      +-------------------------+
test() ==> | 返回地址          |     4
      +-------------------------+
      | 实际参数#1          | [EBP + 8]
      +-------------------------+
      | 实际参数#2          | [EBP + 12]
      +-------------------------+
      | 实际参数#3          | [EBP + 16]
      +-------------------------+
      | 调用者保存的寄存器现场   |
      | EAX，ECX和EDX（根据需要）|
      +-------------------------+
```

```
                | 局部变量#2          |
                +-------------------------+
                | char      #1        | [EBP - n]
                +-------------------------+
EBP==>| 调用者的EBP        |     4
                +-------------------------+
smoke()=>| 返回地址          |     4
                +-------------------------+
```

输入=>n+8位

最后4位==>smoke()地址

BufLab代码（不公布）

```c
void test()
{
    int val;
    /* Put canary on stack to detect possible corruption */
    volatile int local = uniqueval();

    val = getbuf();

    /* Check for corrupted stack */
    if (local != uniqueval()) {
        printf("Sabotaged!: the stack has been corrupted\n");
    }
    else if (val == cookie) {
        printf("Boom!: getbuf returned 0x%x\n", val);
        validate(3);
    } else {
        printf("Dud: getbuf returned 0x%x\n", val);
    }
}
```

```c
int getbuf()
{
    char buf[NORMAL_BUFFER_SIZE];
    Gets(buf);
    return 1;
}
```

BufLab实验

- Level 0: Candle (10 pts)
- Level 1: Sparkler (10 pts)
- Level 2: Firecracker (15 pts)
- Level 3: Dynamite (20 pts)
- Level 4: Nitroglycerin (10 pts)

BufLab背景

- 攻击方法：即向stdin输入一定的字符串序列。

Terminal

File Edit View Search Terminal Help

```
lovfu@linuxmint ~/PPT $ ./bufbomb -u 2012210853
Userid: 2012210853
Cookie: 0x2899448c
Type string:hello
Dud: getbuf returned 0x1
Better luck next time
lovfu@linuxmint ~/PPT $ ./bufbomb -u 2012210853
Userid: 2012210853
Cookie: 0x2899448c
Type string:hello hello hello hello hello hello hello hello hello hello hello he
llo hello hello hello hello hello hello hello hello hello hello hello hello hell
o hello hello hello hello hello hello hello hello hello hello hello hello hello
hello hello hello hello hello hello hello hello hello hello hello hello hello he
llo
Ouch!: You caused a segmentation fault!
Better luck next time
lovfu@linuxmint ~/PPT $ ./bufbomb -u 2012210853 < smoke-2012210853-raw.txt
Userid: 2012210853
Cookie: 0x2899448c
Type string:Smoke!: You called smoke()
VALID
NICE JOB!
lovfu@linuxmint ~/PPT $
```

实验环境

- HTTP服务器：61.4.83.220
- BombLab端口：15213
- BufLab端口：18213
- BombLab
  - 获取程序：61.4.83.220:15213
  - 实时成绩：61.4.83.220:15213/scoreboard（不使用）
- BufLab
  - 获取程序：61.4.83.220:18213
  - 实时成绩：61.4.83.220:18213/scoreboard（不使用）

实验环境

CS:APP Binary Bomb Request

Fill in the form and then click the Submit button.

Hit the Reset button to get a clean form.

Legal characters are spaces, letters, numbers, underscores ('_'), hyphens ('-'), at signs ('@'), and dots ('.').

User name
Enter your Unix login ID
2012210853

Email address
88wanghongwei@gmail.com

Submit Reset

Buffer Lab Scoreboard

Here is the latest information that we have received from your buffer bomb. If your submission is marked as **invalid**, then the testing code thinks your solution is invalid. Some possible reasons are:

- Your solution is not sufficiently robust, a good possibility at the nitroglycerin level. Try to design your exploit to be more tolerant of fluctuations in the stack position.
- You somehow bypassed the protocol used by our autograding service.

Last updated: Tue Aug 20 02:36:43 2013 (updated every 30 secs)

#	Cookie	Score (65)	Candle (0: 10pt)	Sparkler (1: 10pt)	Firecracker (2: 15pt)	Dynamite (3: 20pt)	Nitro (4: 10pt)
1	2899448c	10	10	-	-	-	-

1 students.
Valid submissions: Level 0: 1, Level 1: 0, Level 2: 0, Level 3: 0, Level 4: 0

BombLab实验方式

- ./bomb < solution.txt
- ./bomb，结果上传服务器
- 通过网络学堂提交**BombID**、学号、solution.txt、bomb、Readme文件

BufLab 实验方式

- ./hex2raw<smoke-2012210853.txt>smoke-2012210853-raw.txt
- ./bufbomb -u StudentID < smoke-2012210853-raw.txt
- ./makecookie 2012210853
- ./bufbomb -u StudentID，本地实验
- ./bufbomb -u StudentID -s，结果上传服务器
- 通过网络学堂提交**Cookie**、学号、5个答案.txt、
- bufbomb、 Readme

gdb 基础

- break(b) : 设置断点
  - break 57: 设置57行为断点(本实验用不着)
  - break *0x08048dfa: 设置0x08048dfa为断点

gdb 基础

- run (r): 运行
- continue(c): 从当前位置继续执行
- next(n): 不进入调用函数单步执行
- step(s): 进入调用函数单步执行
- list(l):查看代码
- info:查看相应信息
- quit(q):退出

gdb 基础

- delete(d) break(br): 删除所有断点
- set args [args]: 设置args
- disassemble(disas) [func_name]: 反汇编函数
- print 和display 打印变量的值

gdb 基础

- examine(x)/<n/f/u><addr>：显示内存地址的内容
- n:内存长度;f:格式;u:单位字节数
- 输出格式

  x：按十六进制格式显示变量。

  d ：按十进制格式显示变量。

  u：按十六进制格式显示无符号整型。

  o：按八进制格式显示变量。

  t：按二进制格式显示变量。

  c：按字符格式显示变量。

  f：按浮点数格式显示变量。

  s：按照字符串方式输出变量

gdb 基础

- 常用：

  x /4xh $ebp

  x /s $eax

Objdump 基础

- 常用：

  objdump -d bufbomb > bufbomb.s

```
08048dfa <main>:
 8048dfa:   55                      push   %ebp
 8048dfb:   89 e5                   mov    %esp,%ebp
 8048dfd:   83 e4 f0                and    $0xfffffff0,%esp
 8048e00:   57                      push   %edi
 8048e01:   56                      push   %esi
 8048e02:   53                      push   %ebx
 8048e03:   83 ec 24                sub    $0x24,%esp
 8048e06:   8b 75 08                mov    0x8(%ebp),%esi
 8048e09:   8b 5d 0c                mov    0xc(%ebp),%ebx
 8048e0c:   c7 44 24 04 ac 8a 04    movl   $0x8048aac,0x4(%esp)
 8048e13:   08
```

BombLab演示

Terminal

File Edit View Search Terminal Help

```
lovfu@linuxmint ~/PPT $ gdb bomb-quiet
GNU gdb (GDB) 7.5.91.20130417-cvs-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/lovfu/PPT/bomb-quiet...done.
(gdb) disas phase_1
Dump of assembler code for function phase_1:
   0x08048c23 <+0>:     push   %ebp
   0x08048c24 <+1>:     mov    %esp,%ebp
   0x08048c26 <+3>:     sub    $0x18,%esp
   0x08048c29 <+6>:     movl   $0x8049250,0x4(%esp)
   0x08048c31 <+14>:    mov    0x8(%ebp),%eax
   0x08048c34 <+17>:    mov    %eax,(%esp)
   0x08048c37 <+20>:    call   0x8048c6b <strings_not_equal>
   0x08048c3c <+25>:    test   %eax,%eax
   0x08048c3e <+27>:    je     0x8048c45 <phase_1+34>
   0x08048c40 <+29>:    call   0x8048d54 <explode_bomb>
   0x08048c45 <+34>:    leave
   0x08048c46 <+35>:    ret
End of assembler dump.
(gdb) b *0x08048c34
Breakpoint 1 at 0x8048c34
(gdb)
```

BombLab演示

Terminal

File Edit View Search Terminal Help

```
(gdb) disas phase_1
Dump of assembler code for function phase_1:
   0x08048c23 <+0>:     push   %ebp
   0x08048c24 <+1>:     mov    %esp,%ebp
   0x08048c26 <+3>:     sub    $0x18,%esp
   0x08048c29 <+6>:     movl   $0x8049250,0x4(%esp)
   0x08048c31 <+14>:    mov    0x8(%ebp),%eax
   0x08048c34 <+17>:    mov    %eax,(%esp)
   0x08048c37 <+20>:    call   0x8048c6b <strings_not_equal>
   0x08048c3c <+25>:    test   %eax,%eax
   0x08048c3e <+27>:    je     0x8048c45 <phase_1+34>
   0x08048c40 <+29>:    call   0x8048d54 <explode_bomb>
   0x08048c45 <+34>:    leave
   0x08048c46 <+35>:    ret
End of assembler dump.
(gdb) b *0x08048c34
Breakpoint 1 at 0x8048c34
(gdb) r
Starting program: /home/lovfu/PPT/bomb-quiet
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
hello

Breakpoint 1, 0x08048c34 in phase_1 ()
(gdb) x /s $eax
0x804a8c0 <input_strings>:      "hello"
(gdb) x 0x8049250
0x8049250:      "I am the mayor. I can do anything I want."
(gdb)
```

BombLab演示

Terminal

File Edit View Search Terminal Help

```
lovfu@linuxmint ~/PPT $ ./bomb-quiet
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
I am the mayor. I can do anything I want.
Phase 1 defused. How about the next one?
```

BufLab演示

```c
void test()
{
    int val;
    /* Put canary on stack to detect possible corruption */
    volatile int local = uniqueval();

    val = getbuf();

    /* Check for corrupted stack */
    if (local != uniqueval()) {
        printf("Sabotaged!: the stack has been corrupted\n");
    }
    else if (val == cookie) {
        printf("Boom!: getbuf returned 0x%x\n", val);
        validate(3);
    } else {
        printf("Dud: getbuf returned 0x%x\n", val);
    }
}
```

```
| 局部变量#2          |
+-----------------------+
| char     #1         | [EBP - n]
+-----------------------+
EBP==>| 调用者的EBP      | 4
+-----------------------+
smoke()=>| 返回地址        | 4
+-----------------------+
```

```c
int getbuf()
{
    char buf[NORMAL_BUFFER_SIZE];
    Gets(buf);
    return 1;
}
```

```c
* smoke - On return from getbuf(), the level 0 exploit executes
* the code for smoke() instead of returning to test().
*/
$begin smoke-c */
d smoke()

 printf("Smoke!: You called smoke()\n");
 validate(0);
 exit(0);
}
/* $end smoke-c */
```

BufLab演示

Terminal

File Edit View Search Terminal Help

```
lovfu@linuxmint ~/PPT $ gdb bufbomb
GNU gdb (GDB) 7.5.91.20130417-cvs-ubuntu
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/lovfu/PPT/bufbomb...(no debugging symbols found)...do
ne.
(gdb) disas getbuf
Dump of assembler code for function getbuf:
   0x08048bd4 <+0>:     push   %ebp
   0x08048bd5 <+1>:     mov    %esp,%ebp
   0x08048bd7 <+3>:     sub    $0x48,%esp
   0x08048bda <+6>:     lea    -0x34(%ebp),%eax
   0x08048bdd <+9>:     mov    %eax,(%esp)
   0x08048be0 <+12>:    call   0x8048b1a <Gets>
   0x08048be5 <+17>:    mov    $0x1,%eax
   0x08048bea <+22>:    leave
   0x08048beb <+23>:    ret
End of assembler dump.
(gdb)
```

- lea：将一个地址写入到寄存器
- 0x34=52
- 52+4=56

BufLab演示

Terminal

File Edit View Search Terminal Help

```
(gdb) disas getbuf
Dump of assembler code for function getbuf:
   0x08048bd4 <+0>:     push   %ebp
   0x08048bd5 <+1>:     mov    %esp,%ebp
   0x08048bd7 <+3>:     sub    $0x48,%esp
   0x08048bda <+6>:     lea    -0x34(%ebp),%eax
   0x08048bdd <+9>:     mov    %eax,(%esp)
   0x08048be0 <+12>:    call   0x8048b1a <Gets>
   0x08048be5 <+17>:    mov    $0x1,%eax
   0x08048bea <+22>:    leave
   0x08048beb <+23>:    ret
End of assembler dump.
(gdb) disas smoke
Dump of assembler code for function smoke:
   0x0804907a <+0>:     push   %ebp
   0x0804907b <+1>:     mov    %esp,%ebp
   0x0804907d <+3>:     sub    $0x18,%esp
   0x08049080 <+6>:     movl   $0x8049ee5,(%esp)
   0x08049087 <+13>:    call   0x804890c <puts@plt>
   0x0804908c <+18>:    movl   $0x0,(%esp)
   0x08049093 <+25>:    call   0x80490a4 <validate>
   0x08049098 <+30>:    movl   $0x0,(%esp)
   0x0804909f <+37>:    call   0x804895c <exit@plt>
End of assembler dump.
(gdb)
```

BufLab演示

- 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

  7a 90 04 08