Lab2.二进制炸弹

2020/10/19

实验内容

- 邪恶的Dr. Cang在机房中埋放了“二进制炸弹” :-(
- 二进制炸弹bomb是一个可执行文件，包含若干个关卡
- 每个关卡都需要输入密码
- 如果密码正确，可以进入下一关 :-)
- 如果密码错误，炸弹就会爆炸 :-(
- 目标是通过所有关卡，解除炸弹

实验内容

```
theo@theo:~/Desktop$ ./bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
???

BOOM!!!
The bomb has blown up.
```

```
theo@theo:~/Desktop$ ./bomb
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!

Halfwave 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!
```

实验步骤

- 于elearning获取实验所需可执行文件bomb
- 反汇编：得到bomb的汇编代码，通过阅读代码，推测密码
  - Objdump
- 调试：设置断点、单步运行、查看寄存器和内存状态，确定密码
  - gdb
- 解开密码
- 书写实验报告

实验内容

- Phase 1: string comparison
- Phase 2: loops
- Phase 3: conditionals/switches
- Phase 4: recursive calls and the stack discipline
- Phase 5: pointers
- Phase 6: linked lists/pointers/structs
- Secret phase:

objdump

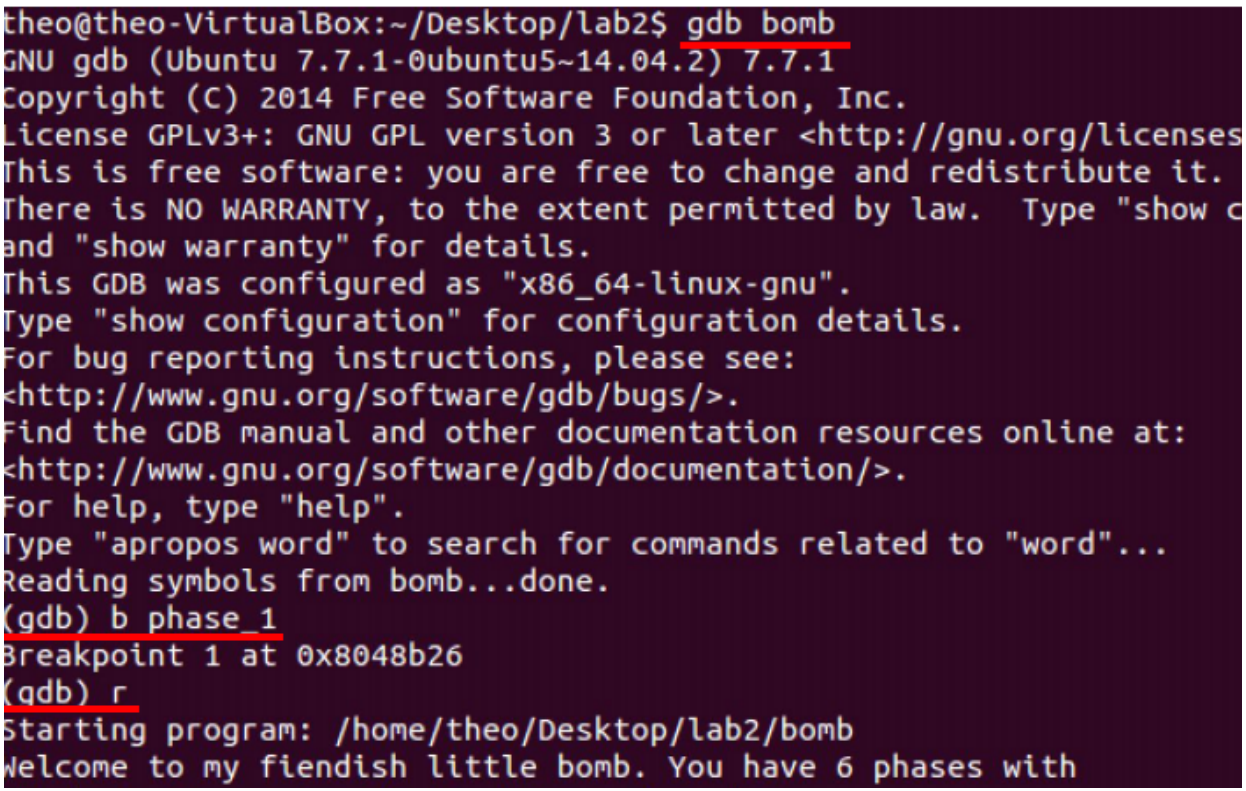
- 打印bomb的汇编代码：objdump –d bomb
- 建议objdump –d bomb > filename 输出到文件之后查看
- 更建议打印成纸质文档查看（？）

```
380 08048b20 <phase_1>:
381  8048b20:       55                      push   %ebp
382  8048b21:       89 e5                   mov    %esp,%ebp
383  8048b23:       83 ec 08                sub    $0x8,%esp
384  8048b26:       8b 45 08                mov    0x8(%ebp),%eax
385  8048b29:       83 c4 f8                add    $0xfffffff8,%esp
386  8048b2c:       68 c0 97 04 08          push   $0x80497c0
387  8048b31:       50                      push   %eax
388  8048b32:       e8 f9 04 00 00          call   8049030 <strings_not_equal>
389  8048b37:       83 c4 10                add    $0x10,%esp
390  8048b3a:       85 c0                   test   %eax,%eax
391  8048b3c:       74 05                   je     8048b43 <phase_1+0x23>
392  8048b3e:       e8 b9 09 00 00          call   80494fc <explode_bomb>
393  8048b43:       89 ec                   mov    %ebp,%esp
394  8048b45:       5d                      pop    %ebp
395  8048b46:       c3                      ret
396  8048b47:       90                      nop
397
```

gdb

- 运行gdb gdb bomb
- 断点　break/b functionName
- 运行　run/r
- 单步　stepi/nexti (stepNumber)
- 查询　print/x format register/address

gdb


theo@theo-VirtualBox:~/Desktop/lab2$ gdb bomb
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show c
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from bomb...done.
(gdb) b phase_1
Breakpoint 1 at 0x8048b26
(gdb) r
Starting program: /home/theo/Desktop/lab2/bomb
Welcome to my fiendish little bomb. You have 6 phases with

gdb

```
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Please do not blow me up.

Breakpoint 1, 0x08048b26 in phase_1 ()
(gdb) nexti
0x08048b29 in phase_1 ()
(gdb) nexti
0x08048b2c in phase_1 ()
(gdb) nexti
0x08048b31 in phase_1 ()
(gdb) nexti 3
0x08048b3a in phase_1 ()
(gdb)
```

gdb

```
(gdb) print $eax
$1 = 1
(gdb) print (char *) $edx
$2 = 0x80497dd ""
(gdb) print 0xdeadbeef
$3 = 3735928559
(gdb) print *0xdeadbeef
Cannot access memory at address 0xdeadbeef
(gdb) p $ebx
$4 = -1073745724
(gdb) p/a $ebx
$5 = 0xbffff0c4
(gdb) p/t 0xdeadbeef
$6 = 11011110101011011011111011101111
```

gdb

- 更多东西清参看elearning上的gdb手册

实验报告

1. 学号.pdf
2. 每个关卡的密码
3. 每个密码的推演过程（重要，如函数说明，实现方式等）:-)
4. 实际操作的心得感想（可选）
5. 表意清晰
6. 精简（更重要）:-) //但不简陋

实验提交

- 上传实验报告至elearning的对应作业
- Deadline：2020/10/29

实验评分

- 解开每个关卡（6 * 10% = 60%）
- 解开隐藏关卡（20%）
- 实验报告（20%）:-)
- 抄袭按0分处理 :-(//甚至更坏

- //有一说一，占比不重要，重要的将会是相对分差