Defusing Binary-Bomb

实验三　二进制炸弹拆除

一．实验要求

要求根据反汇编指令分析程序运行需要的参数，即需要正确的输入，以拆除炸弹。根据通过的关卡数目评判最终的实验得分。

二．实验目的

(1) 熟悉 MIPS 指令集；
(2) 根据反汇编程序可以分析程序的功能和执行流程；
(3) 熟悉 GDB 调试工具，帮助程序理解。

三．实验配置与步骤

1.准备工作

本实验设计为一个黑客拆解二进制炸弹的游戏。我们仅给黑客（同学）提供一个 MIPS 二进制可执行文件 bomb，不提供源代码（提供汇编代码帮助理解）。程序运行中有 6 个关卡（6 个 phase），每个关卡需要用户输入正确的字符串或数字才能通关，否则会引爆炸弹（打印出一条错误信息）！

要求同学运用 GDB 调试工具，通过分析汇编代码，找到在每个 phase 程序段中，引导程序跳转到“explode_bomb”程序段的地方，并分析其成功跳转的条件，以此为突破口寻找应该在命令行输入何种字符串来通关。

需要准备的软件：

- bomb 可执行文件（MIPS）

需要准备的硬件：

- 龙芯实验平台（MIPS）

2. **拆除 phase_1 演示**

2.1 参考命令

```
Dump of assembler code for function phase_1:
   0x00400d6c <+0>: addiu   sp,sp,-32
   0x00400d70 <+4>: sw  ra,28(sp)
   0x00400d74 <+8>: sw  s8,24(sp)
   0x00400d78 <+12>:    move    s8,sp
   0x00400d7c <+16>:    sw  a0,32(s8)
   0x00400d80 <+20>:    lw  a0,32(s8)
   0x00400d84 <+24>:    lui v0,0x40
   0x00400d88 <+28>:    addiu   a1,v0,10092
   0x00400d8c <+32>:    jal 0x401cf8 <strings_not_equal>
   0x00400d90 <+36>:    nop
   0x00400d94 <+40>:    beqz    v0,0x400da4 <phase_1+56>
   0x00400d98 <+44>:    nop
   0x00400d9c <+48>:    jal 0x4021f0 <explode_bomb>
   0x00400da0 <+52>:    nop
   0x00400da4 <+56>:    move    sp,s8
   0x00400da8 <+60>:    lw  ra,28(sp)
   0x00400dac <+64>:    lw  s8,24(sp)
   0x00400db0 <+68>:    addiu   sp,sp,32
   0x00400db4 <+72>:    jr  ra
   0x00400db8 <+76>:    nop
End of assembler dump.
```

图一　phase_1 的二进制代码

(1) **addiu　sp,sp,-32**　:调用子函数为当前函数开辟栈空间；

(2) **sw　　a0,32(s8)**　:$a0(参数寄存器)的内容被存入栈；
(思考：为什么需要对$a0 执行 sw 和 lw 操作？)

(3) **addiu a1,v0,10092**　：加载另外一个参数到$a1; (结合前一步的 lui 命令理解)

参数寄存器$a0 $a1 将参数传入 **strings_not_equal** 函数后进行比较，返回值放在$v0 内，看是否为 0（beqz），不为 0 则炸弹爆炸，因此可以推断出两个参数的值需要相同。接下来在实验平台中使用 GDB 调试一下，看看寄存器内是不是我们期望的内容。

2.2 调试步骤

```
[root@localhost ~]# gdb bomb
GNU gdb (GDB) Fedora (7.1-18.fc13)
Copyright (C) 2010 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 "mipsel-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /root/bomb...(no debugging symbols found)...done.
(gdb) disas phase_1
```

(1)打开终端，进入 gdb 调试窗口。

图二 使用 GDB 进行反汇编

(2)设置断点，然后用 r 命令运行程序。输入一串字符后会得到命中断点的提示。

更正：断点应该设置在 0x00400d90 ，即 b *0x00400d90，其他不变。

```
(gdb) b *0x00400d78
Breakpoint 1 at 0x400d78
(gdb) r
Starting program: /root/bomb
Please input your ID_number:
201440703044
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
I have no idea

Breakpoint 1, 0x00400d78 in phase_1 ()
```

图三 设置断点并运行

(3)使用 ni 命令进行单步调试，之后使用 **i r $a0** 和 **i r $a1** 可以查看当前 a0 和 a1 的寄存器状态（也可以直接使用 **i r** 命令查看所有寄存器的状态）。同时可以配合 x 命令查看内存中存储的数据，可以看到 a0 里面确实保存着我们输入的字串。

```
(gdb) i r $a0
a0: 0x413294
(gdb) x $a0
0x413294 <input_strings>:    0x61682049
(gdb) i r $a1
a1: 0x0
```

图四 查看寄存器内容

(4)继续使用 ni 命令，可以看到 a1 里面保存的字符串是“Let’s begin now!”。

```
(gdb) x /16c $a1
0x40276c:   76 'L'  101 'e' 116 't' 39 '\'' 115 's' 32 ' '  98 'b'  101 'e'
0x402774:   103 'g' 105 'i' 110 'n' 32 ' '  110 'n' 111 'o' 119 'w' 33 '!'
```

图五 参数寄存器内容

(5)退出当前调试，正确输入第一个炸弹内容并继续调试。

```
(gdb) r
Starting program: /root/bomb
Please input your ID_number:
201440703044
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
Let's begin now!
Phase 1 defused. How about the next one?
```

图六 拆除第一个炸弹

3.接下来的提示

上面给大家演示了拆除炸弹 1 的过程，这也是一个最简单的炸弹。接下来的 5 个炸弹难度逐级递增，大家不要拘泥于代码，死读代码；要结合 GDB 调试器，查看内存以及各个寄存器的值，这样对于绕过一些棘手的函数很有帮助，可以让大家切中炸弹爆炸条件的要害进行分析。GDB 的具体使用也不止上面所说的几个命令，其他的希望大家可以自行上网搜索，掌握 GDB 的其他命令。GDB 调试器可以在拆除炸弹的过程中给予你很大的帮助，但是仍希望你将它当作一个辅助的工具来使用，帮助你更好的了解与汇编代码有关的内容，对汇编代码的理解是该实验的重点。

祝大家拆弹顺利！

附：GDB 汇编调试指令合集（供参考）

https://blog.csdn.net/txx_683/article/details/53453548