CSAPP-bomblab-writeup

原创

CodeStarr 于 2022-03-26 12:40:39 发布 1178 收藏

分类专栏： bin # re 文章标签： 安全

本文链接：https://blog.csdn.net/Ga4ra/article/details/123753886

文章目录

文章已首发于看雪论坛

Bomb这个程序是《深入理解计算机系统》的配套实验的一部分，一共6关。

原地址：

- http://csapp.cs.cmu.edu/3e/labs.html
- http://csapp.cs.cmu.edu/3e/bomb.tar

环境：

- Ubuntu 18.04
- IDA 远程调试

phase 1

第一关热身题，输入一段明文保存的字符串：

```
.text:0000000000400EE4 mov     esi, offset aBorderRelation     ; "Border relations with Canada have never"...
.text:0000000000400EE9 call    strings_not_equal
.text:0000000000400EEE test    eax, eax
.text:0000000000400EF0 jz      short loc_400EF7
.text:0000000000400EF2 call    explode_bomb
```

key: Border relations with Canada have never been better.

phase 2

要求输入6个数：

```
.text:000000000040145C public read_six_numbers
.text:000000000040145C read_six_numbers proc near
...
.text:0000000000401480 mov     esi, offset aDDDDDD             ; "%d %d %d %d %d %d"
.text:0000000000401485 mov     eax, 0
.text:000000000040148A call    ___isoc99_sscanf
```

输入的6个数字会存在数组里，临时取名input。接下来rbx执行`input[1]`，rbp指向`input[6]`（用来结束循环）：

```
.text:0000000000400F30 loc_400F30:
.text:0000000000400F30                                         ; phase_2+19↑j
.text:0000000000400F30 lea     rbx, [rsp+38h+var_34]           ; rbx = pNum = &input[1]
.text:0000000000400F35 lea     rbp, [rsp+38h+var_20]           ; rbp = &input[6] = pEnd
.text:0000000000400F3A jmp     short loopCompare
```

`input[i]`与`input[i-1] * 2`比较，相等即可，所以输入一个等比数列即可。

```
.text:0000000000400F17 loopCompare:
.text:0000000000400F17                                         ; phase_2+3E↓j
.text:0000000000400F17 mov     eax, [rbx-4]                    ; eax = *(--pNum)
.text:0000000000400F1A add     eax, eax                        ; eax *= 2
.text:0000000000400F1C cmp     [rbx], eax
.text:0000000000400F1E jz      short loc_400F25                ; if *pNum == 2*eax   continue
.text:0000000000400F20 call    explode_bomb
.text:0000000000400F25 ; ---------------------------------------------------------------------------
.text:0000000000400F25
.text:0000000000400F25 loc_400F25:
.text:0000000000400F25 add     rbx, 4
.text:0000000000400F29 cmp     rbx, rbp                        ; if i == len(input)  break
.text:0000000000400F2C jnz     short loopCompare               ; eax = *(--pNum)
.text:0000000000400F2E jmp     short loc_400F3C
```

key: 1 2 4 8 16 32

phase 3

输入两个数 num1, num2

```
.text:0000000000400F51 mov     esi, (offset aDDDDDD+0Ch)       ; "%d %d"
.text:0000000000400F56 mov     eax, 0
.text:0000000000400F5B call    ___isoc99_sscanf
```

有一个长度为8的跳转表，所以会检查`num1<=7`，之后跳转到`jpt_400F75[num1]`，每个case都有一个数，与num2相等则通过

```
.text:0000000000400F6A loc_400F6A:
.text:0000000000400F6A cmp     [rsp+18h+var_10], 7             ; switch 8 cases
.text:0000000000400F6F ja      short def_400F75                ; jumptable 0000000000400F75 default case
.text:0000000000400F71 mov     eax, [rsp+18h+var_10]
.text:0000000000400F75 jmp     ds:jpt_400F75[rax*8]            ; switch jump
```

比如下面这个`num1==0`的case:

```
.text:0000000000400F7C loc_400F7C:
.text:0000000000400F7C                                         ; DATA XREF: .rodata:jpt_400F75↓o
.text:0000000000400F7C mov     eax, 0CFh                       ; jumptable 0000000000400F75 case 0
.text:0000000000400F81 jmp     short loc_400FBE
```

所以输入`0 207`（0xcf==207） 即可，其它7个case也可以，所以应该是有8对数作为flag。

phase 4

还是输入两个数num1 num2, 要求`num1 <= 0xe`

```
.text:000000000040101A mov     esi, (offset aDDDDDD+0Ch)       ; "%d %d"
.text:000000000040101F mov     eax, 0
.text:0000000000401024 call    ___isoc99_sscanf
.text:0000000000401029 cmp     eax, 2
.text:000000000040102C jnz     short loc_401035
.text:000000000040102E cmp     [rsp+18h+var_10], 0Eh           ; num 1 <= 0xe
.text:0000000000401033 jbe     short loc_40103A
.text:0000000000401035
.text:0000000000401035 call    explode_bomb
```

然后调用一个func4(0xe, 0, num1), 要求返回0，并且num2也需要为0：

```
.text:000000000040103A mov     edx, 0Eh
.text:000000000040103F mov     esi, 0
.text:0000000000401044 mov     edi, [rsp+18h+var_10]           ; edi = num1
.text:0000000000401048 call    func4
.text:000000000040104D test    eax, eax
.text:000000000040104F jnz     short loc_401058                ; if func4() != 0    bomb
.text:0000000000401051 cmp     [rsp+18h+var_C], 0              ; if num2!=0  bomb
.text:0000000000401056 jz      short loc_40105D
```

分析一下func4函数，这是一个递归函数：

```
.text:0000000000400FCE public func4
.text:0000000000400FCE                                          ; func4+30↓p ...
.text:0000000000400FCE ; __unwind {
.text:0000000000400FCE sub     rsp, 8
.text:0000000000400FD2 mov     eax, edx                         ; eax = edx(init 0xe)
.text:0000000000400FD4 sub     eax, esi                         ; eax =  edx(init 0xe) - esi(init 0)
.text:0000000000400FD6 mov     ecx, eax
.text:0000000000400FD8 shr     ecx, 1Fh                         ; ecx = eax >> 1F
.text:0000000000400FD8                                          ; highest bit --> 0
.text:0000000000400FDB add     eax, ecx                         ; eax += ecx
.text:0000000000400FDD sar     eax, 1                           ; eax = eax >> 1
.text:0000000000400FDD                                          ; highest bit --> 0

.text:0000000000400FDF lea     ecx, [rax+rsi]                   ; ecx = rax+rsi
.text:0000000000400FE2 cmp     ecx, edi                         ; if ecx > num1
.text:0000000000400FE4 jle     short loc_400FF2
.text:0000000000400FE6 lea     edx, [rcx-1]                     ;   func(rcx-1, esi)
.text:0000000000400FE9 call    func4
.text:0000000000400FEE add     eax, eax
.text:0000000000400FF0 jmp     short ret_func4
.text:0000000000400FF2 ; ---------------------------------------------------------------------------
.text:0000000000400FF2 loc_400FF2:
.text:0000000000400FF2 mov     eax, 0                           ; else {
.text:0000000000400FF7 cmp     ecx, edi                         ;   if ecx < num1 {
.text:0000000000400FF9 jge     short ret_func4                  ;       ret = func4(edx, rcx+1)
.text:0000000000400FF9                                          ;       return 2*ret + 1
.text:0000000000400FFB lea     esi, [rcx+1]                     ;    }
.text:0000000000400FFE call    func4
.text:0000000000401003 lea     eax, [rax+rax+1]                 ;   else {  return eax}
.text:0000000000401003                                          ; }
.text:0000000000401007
.text:0000000000401007 ret_func4:                               ; CODE XREF: func4+22↑j
.text:0000000000401007                                          ; func4+2B↑j
.text:0000000000401007 add     rsp, 8
.text:000000000040100B retn
.text:000000000040100B ; } // starts at 400FCE
.text:000000000040100B func4 endp
```

用c还原：

```
int func4(int nEdx, int nEsi, int nEdi)
{
 int nRet = 0;
 int nEcx = 0;
 nRet = nEdx - nEsi;
 nRet += nRet >> 0x1F;
 nRet = nRet >> 1;

 nEcx = nRet + nEsi;
    // 调试到这里, nEcx==7
 if (nEcx > nEdi)
 {
  nEdx = nEcx - 1;
  nRet = func4(nEdx, nEsi, nEdi);
  nRet += nRet;
 }
 else if (nEcx < nEdi)
 {
  nRet = 2 * func4(nEdx, nEsi, nEdi) + 1;
 }

 // 终止条件是 nEcx == nEdi

 return nRet;
}

int main()
{
 int nNum1 = 0;
 int ret = func4(0xe, 0, nNum1);
 return 0;
}
```

调试一下，执行到if时nEcx是7，所以key为 `7 0`

phase 5

输入一段字符串，要求长度为6

```
.text:000000000040107A call    string_length
.text:000000000040107F cmp     eax, 6
.text:0000000000401082 jz      short loc_4010D2
.text:0000000000401084 call    explode_bomb
...
.text:00000000004010D2 mov     eax, 0
.text:00000000004010D7 jmp     short loc_40108B
```

然后根据输入的字符串，映射为另一个字符串。

```
.text:000000000040108B loc_40108B:                             ; CODE XREF: phase_5+4A↓j
.text:000000000040108B                                         ; phase_5+75↓j
.text:000000000040108B movzx   ecx, byte ptr [rbx+rax]         ; rbx=input
.text:000000000040108B                                         ; rax-->i
.text:000000000040108F mov     byte ptr [rsp+28h+var_28], cl
.text:0000000000401092 mov     rdx, [rsp+28h+var_28]
.text:0000000000401096 and     edx, 0Fh                        ; edx = 0x0F & input[i]
.text:0000000000401099 movzx   edx, byte ptr [rdx+4024B0h]    ; string table
.text:00000000004010A0 mov     [rsp+rax+28h+var_18], dl
.text:00000000004010A4 add     rax, 1
.text:00000000004010A8 cmp     rax, 6
.text:00000000004010AC jnz     short loc_40108B                ; rbx=input
.text:00000000004010AC                                         ; rax-->i
.text:00000000004010AE mov     [rsp+28h+var_12], 0
.text:00000000004010B3 mov     esi, offset aFlyers             ; "flyers"
.text:00000000004010B8 lea     rdi, [rsp+28h+var_18]
.text:00000000004010BD call    strings_not_equal
.text:00000000004010C2 test    eax, eax
.text:00000000004010C4 jz      short loc_4010D9
.text:00000000004010C6 call    explode_bomb
```

映射的逻辑是，每个输入的字符的低4位，作为0x4024B0处字符串表（如下）的偏移。比如输入小写a（0x61），偏移就是`0x61 & 0x0f == 0x01`，映射为4024B1处的0x61。

```
00000000004024B0  6D 61 64 75 69 65 72 73  6E 66 6F 74 76 62 79 6C  maduiersnfotvbyl
00000000004024C0  53 6F 20 79 6F 75 20 74  68 69 6E 6B 20 79 6F 75  So you think you
00000000004024D0  20 63 61 6E 20 73 74 6F  70 20 74 68 65 20 62 6F   can stop the bo
00000000004024E0  6D 62 20 77 69 74 68 20  63 74 72 6C 2D 63 2C 20  mb with ctrl-c,
00000000004024F0  64 6F 20 79 6F 75 3F 00  43 75 72 73 65 73 2C 20  do you?.Curses,
0000000000402500  79 6F 75 27 76 65 20 66  6F 75 6E 64 20 74 68 65  you've found the
0000000000402510  20 73 65 63 72 65 74 20  70 68 61 73 65 21 00 00   secret phase!..
0000000000402520  42 75 74 20 66 69 6E 64  69 6E 67 20 69 74 20 61  But finding it a
0000000000402530  6E 64 20 73 6F 6C 76 69  6E 67 20 69 74 20 61 72  nd solving it ar

maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?.Curses, you've found the secret phase!..
But finding it and solving it ar
```

我们的目标就是找到flyers这几个字母在以下字符串中的偏移，可以用python来解：

```
strtab = "maduiersnfotvbylSo you think you can stop the bomb with ctrl-c, do you?.Curses, you've found the secre
t phase!..But finding it and solving it are quite different"
str2search = "flyers"

for c in str2search:
    nIndex = strtab.find(c)
    print(chr(nIndex + 0x60), end="") # key: ionefg
```

phase 6

第6关，还是输入6个数字，紧接着是一个双重循环，要求每一个数字不能大于5，并且互不重复：

```
.text:0000000000401106 call    read_six_numbers
.text:000000000040110B mov     r14, rsp                        ; r14 = &(input_nums[6])
.text:000000000040110E mov     r12d, 0                         ; r12d --> i
.text:0000000000401114
.text:0000000000401114 big_loop_loc_401114:                    ; CODE XREF: phase_6+5D↓j
.text:0000000000401114 mov     rbp, r13                        ; out loop ==========================================
========================
.text:0000000000401114                                         ; r13 = &(input_nums[i])  too
.text:0000000000401117 mov     eax, [r13+0]
.text:000000000040111B sub     eax, 1
.text:000000000040111E cmp     eax, 5
.text:0000000000401121 jbe     short loc_401128                ; if inputnum[i] > 5    bomb  每个数不能大于5
.text:0000000000401123 call    explode_bomb
.text:0000000000401128 ; ---------------------------------------------------------------------------
.text:0000000000401128
.text:0000000000401128 loc_401128:                             ; CODE XREF: phase_6+2D↑j
.text:0000000000401128 add     r12d, 1
.text:000000000040112C cmp     r12d, 6
.text:0000000000401130 jz      short loc_401153                ; if r12d == 6   break big loop
.text:0000000000401132 mov     ebx, r12d                       ; ebx --> j
.text:0000000000401135
.text:0000000000401135 small_loop_loc_401135:                  ; CODE XREF: phase_6+57↓j
.text:0000000000401135 movsxd  rax, ebx                        ; loop ebx = 1-->5  ----------------------------
------
.text:0000000000401138 mov     eax, [rsp+rax*4+78h+arrInputNum]
.text:000000000040113B cmp     [rbp+0], eax                    ; if input_num[i] == inputnum[j]  bomb  // 互不重
复
.text:000000000040113E jnz     short loc_401145                ; if (++ebx > 5) break;
.text:0000000000401140 call    explode_bomb
.text:0000000000401145 ; ---------------------------------------------------------------------------
.text:0000000000401145
.text:0000000000401145 loc_401145:                             ; CODE XREF: phase_6+4A↑j
.text:0000000000401145 add     ebx, 1                          ; if (++ebx > 5) break;
.text:0000000000401148 cmp     ebx, 5
.text:000000000040114B jle     short small_loop_loc_401135     ; loop ebx = 1-->5  ----------------------------
------
.text:000000000040114D add     r13, 4                          ; ebx == 6
.text:000000000040114D                                         ; r13 = &inputnum[1]
.text:0000000000401151 jmp     short big_loop_loc_401114       ; out loop
```

之后，将每个数字与7做差，比如，输入1，2，3，4，5，6，现在变为6，5，4，3，2，1：

```
.text:0000000000401153 loc_401153:                             ; CODE XREF: phase_6+3C↑j
.text:0000000000401153 lea     rsi, [rsp+78h+var_60]           ; rsi = inputnum[] end
.text:0000000000401158 mov     rax, r14                        ; r14 = &inputnum[]
.text:000000000040115B mov     ecx, 7
.text:0000000000401160
.text:0000000000401160 reduced_by_7:                           ; CODE XREF: phase_6+79↓j
.text:0000000000401160 mov     edx, ecx                        ; for i 0-5
.text:0000000000401160                                         ;      inputnum_1[i] = inputnum[i] = 7 - inputnum
[i]
.text:0000000000401162 sub     edx, [rax]
.text:0000000000401164 mov     [rax], edx
.text:0000000000401166 add     rax, 4
.text:000000000040116A cmp     rax, rsi
.text:000000000040116D jnz     short reduced_by_7
.text:000000000040116F mov     esi, 0                          ; rsi --> i
.text:0000000000401174 jmp     short loc_401197
```

loc_401197部分开始稍微复杂：

```
.text:0000000000401197 loc_401197:                             ; CODE XREF: phase_6+80↑j
.text:0000000000401197 mov     ecx, [rsp+rsi+78h+arrInputNum]  ; ecx = inputnum_1[i]
.text:000000000040119A cmp     ecx, 1
.text:000000000040119D jle     short getTheFirstNode           ; if inputnum_1[0] <= 1
.text:000000000040119F mov     eax, 1                          ; eax = 1
.text:00000000004011A4 mov     edx, offset node1               ; edx = pHead
.text:00000000004011A9 jmp     short getLinklistNodeAtRsi      ; rdx = rdx->next
```

先注意到有个全局变量node1, 在IDA的符号窗口可以看到，一共有6个节点：

```
name window:
node1 00000000006032D0 P
node2 00000000006032E0 P
node3 00000000006032F0 P
node4 0000000000603300 P
node5 0000000000603310 P
node6 0000000000603320 P
```

数据窗口看一下，原来是一个链表（如果没有node符号，对这种指针数据应该敏感一点）：

```
hex view:
00000000006032D0  000000010000014C 00000000006032E0
00000000006032E0  00000002000000A8 00000000006032F0
00000000006032F0  000000030000039C 0000000000603300
0000000000603300  00000004000002B3 0000000000603310
0000000000603310  00000005000001DD 0000000000603320
0000000000603320  00000006000001BB 0000000000000000
```

结构还原：

```
struct S {
    int n1;
    int n2;
    S* next;
}
```

识别出数据结构就稍微容易点了。比如现在输入的数组已经变成6，5，4，3，2，1, 那么下面的逻辑会依次把第6，5，4，3，2，1个链表节点的地址存到局部数组里：

```
.text:0000000000401176 getLinklistNodeAtRsi:                   ; CODE XREF: phase_6+8B↓j
.text:0000000000401176                                         ; phase_6+B5↓j
.text:0000000000401176 mov     rdx, [rdx+8]                    ; rdx = rdx->next
.text:000000000040117A add     eax, 1                          ; if (++eax == ecx) break    ecx==7 0x00000000004
0115B
.text:000000000040117D cmp     eax, ecx
.text:000000000040117F jnz     short getLinklistNodeAtRsi      ; rdx = rdx->next
.text:0000000000401181 jmp     short loc_401188                ; word_array[ i * 2] = pEnd
.text:0000000000401183 ; ---------------------------------------------------------------------------
.text:0000000000401183
.text:0000000000401183 getTheFirstNode:                        ; CODE XREF: phase_6+A9↓j
.text:0000000000401183 mov     edx, offset node1               ; edx = pHead
.text:0000000000401188
.text:0000000000401188 loc_401188:                             ; CODE XREF: phase_6+8D↑j
.text:0000000000401188 mov     [rsp+rsi*2+78h+arrLinkListNodes], rdx     ; word_array[ i * 2] = pEnd
.text:000000000040118D add     rsi, 4
.text:0000000000401191 cmp     rsi, 18h
.text:0000000000401195 jz      short loc_4011AB                ; if (++i == 6) break;
```

执行完后的栈空间（arrLinkListNodes）：

```
00007FFC522B9E30  00603320  Node6
00007FFC522B9E34  00000000
00007FFC522B9E38  00603310  Node5
00007FFC522B9E3C  00000000
00007FFC522B9E40  00603300  Node4
00007FFC522B9E44  00000000
00007FFC522B9E48  006032F0  Node3
00007FFC522B9E4C  00000000
00007FFC522B9E50  006032E0  Node2
00007FFC522B9E54  00000000
00007FFC522B9E58  006032D0  Node1
```

继续跟到4011ab处的逻辑，这里会根据局部数组，改变节点的next，重新排列链表：

```
.text:00000000004011AB loc_4011AB:                             ;
.text:00000000004011AB mov     rbx, [rsp+78h+arrLinkListNodes] ; arrLinkListNodes[0]
.text:00000000004011B0 lea     rax, [rsp+78h+var_50]           ; pNode = &arrLinkListNodes[1]  rax=ppNode
.text:00000000004011B5 lea     rsi, [rsp+78h+var_28]           ; rsi = arrLinkListNodes[6] = NULL
.text:00000000004011BA mov     rcx, rbx                        ; rcx = arrLinkListNodes[0]
.text:00000000004011BD
.text:00000000004011BD relinkList_by_arrLinkListNodes:         ;
.text:00000000004011BD mov     rdx, [rax]                      ; edx = arrLinkListNodes[1] = *ppNode
.text:00000000004011C0 mov     [rcx+8], rdx                    ; arrLinkListNodes[0]->next = arrLinkListNodes[1]
.text:00000000004011C4 add     rax, 8
.text:00000000004011C8 cmp     rax, rsi                        ; if (++ppNode == NULL) break; 遍历完毕
.text:00000000004011CB jz      short loc_4011D2
.text:00000000004011CD mov     rcx, rdx                        ; rcx = pNode
.text:00000000004011D0 jmp     short relinkList_by_arrLinkListNodes ; edx = *pNode
.text:00000000004011D2 ; ---------------------------------------------------------------------------
.text:00000000004011D2
.text:00000000004011D2 loc_4011D2:
.text:00000000004011D2 mov     qword ptr [rdx+8], 0            ; pNode->next = NULL  最后一个节点next置为空
```

局部数组是6，5，4，3，2，1，所以链表就被逆过来了，并且node1.next被置为NULL。

最后看关键的判断炸弹是否爆炸的逻辑。4011DF这里会按照重新排列好的链表顺序，判断它们的n2成员是否从大到小排列，不是则爆炸。

```
.text:00000000004011DF loc_4011DF:                             ; CODE XREF: phase_6+101↓j
.text:00000000004011DF mov     rax, [rbx+8]                    ; rbx = arrLinkListNodes[0]
.text:00000000004011DF                                         ; rax = rbx->next
.text:00000000004011E3 mov     eax, [rax]
.text:00000000004011E5 cmp     [rbx], eax                      ; if (rbx->n2 < rbx->next->n2)  bomb
.text:00000000004011E7 jge     short loc_4011EE
.text:00000000004011E9 call    explode_bomb
```

让我们再回看一下数据窗口的链表：

```
hex view:
00000000006032D0  000000010000014C 00000000006032E0
00000000006032E0  00000002000000A8 00000000006032F0
00000000006032F0  000000030000039C 0000000000603300
0000000000603300  00000004000002B3 0000000000603310
0000000000603310  00000005000001DD 0000000000603320
0000000000603320  00000006000001BB 0000000000000000
```

它们的n2依次为14c, a8, 39c, 2b3, 1dd, 1bb, 写个脚本排列一下：

```python
dictNodes = {
    0x14C : 1,
    0xa8 : 2,
    0x39c : 3,
    0x2b3 : 4,
    0x1dd : 5,
    0x1bb : 6
}

def dict_val(item):
    return item[0]

dictNewNodes = {k : v for k, v in sorted(dictNodes.items(), key=dict_val, reverse=True)}
print("input key: 7-index:")
for k, v in dictNewNodes.items():
    # print("0x%04x-" % k, v, end="")
    print("%d "% (7-v), end="")  # 4 3 2 1 6 5
```

key : 4 3 2 1 6 5