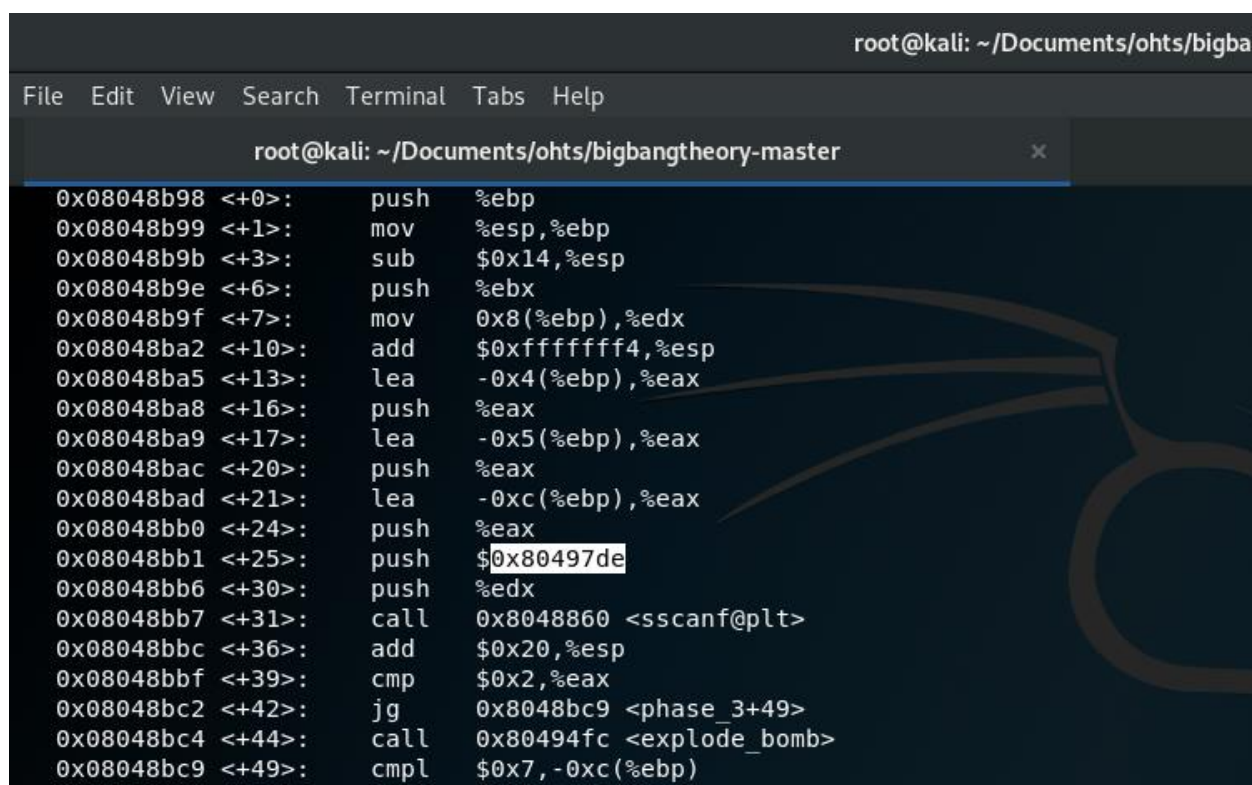


Sheldon Phase 3- Writeup

Objective of this report is to introduce the used approach to find the passphrase for the phase_3 of the Sheldon_1 binary file.

Since we already know how the program works, let's start with disassembling the phase_3 to analyze its functionality. Also, notice that the 'b phase_3' command was used to set a breakpoint at phase_3 from the beginning.

This phase can take multiple combination of values as inputs. Disassembled code implied that this might include a case-based scenario to check the input values.



```
root@kali: ~/Documents/ohts/bigbangtheory-master
File Edit View Search Terminal Tabs Help
root@kali: ~/Documents/ohts/bigbangtheory-master
0x08048b98 <+0>: push %ebp
0x08048b99 <+1>: mov %esp,%ebp
0x08048b9b <+3>: sub $0x14,%esp
0x08048b9e <+6>: push %ebx
0x08048b9f <+7>: mov 0x8(%ebp),%edx
0x08048ba2 <+10>: add $0xffffffff4,%esp
0x08048ba5 <+13>: lea -0x4(%ebp),%eax
0x08048ba8 <+16>: push %eax
0x08048ba9 <+17>: lea -0x5(%ebp),%eax
0x08048bac <+20>: push %eax
0x08048bad <+21>: lea -0xc(%ebp),%eax
0x08048bb0 <+24>: push %eax
0x08048bb1 <+25>: push $0x80497de
0x08048bb6 <+30>: push %edx
0x08048bb7 <+31>: call 0x8048860 <sscanf@plt>
0x08048bbc <+36>: add $0x20,%esp
0x08048bbf <+39>: cmp $0x2,%eax
0x08048bc2 <+42>: jg 0x8048bc9 <phase_3+49>
0x08048bc4 <+44>: call 0x80494fc <explode_bomb>
0x08048bc9 <+49>: cmpl $0x7,-0xc(%ebp)
```

Figure 1: Disassembled phase_3 function

Prior to the call of scanf function, there's a value getting pushed to the stack at the line 25. That looks like an address we must check before executing anything [Figure 1].

```
root@kali: ~/Documents/ohts/bigbangtheory-master
File Edit View Search Terminal Tabs Help

root@kali: ~/Documents/ohts/bigbangtheory-master x
0x08048c3a <+162>: lea    0x0(%esi),%esi
0x08048c40 <+168>: mov    $0x6f,%bl
0x08048c42 <+170>: cmpl   $0xa0,-0x4(%ebp)
0x08048c49 <+177>: je     0x8048c8f <phase_3+247>
0x08048c4b <+179>: call   0x80494fc <explode_bomb>
0x08048c50 <+184>: jmp    0x8048c8f <phase_3+247>
0x08048c52 <+186>: mov    $0x74,%bl
0x08048c54 <+188>: cmpl   $0x1ca,-0x4(%ebp)
0x08048c5b <+195>: je     0x8048c8f <phase_3+247>
0x08048c5d <+197>: call   0x80494fc <explode_bomb>
0x08048c62 <+202>: jmp    0x8048c8f <phase_3+247>
0x08048c64 <+204>: mov    $0x76,%bl
0x08048c66 <+206>: cmpl   $0x30c,-0x4(%ebp)
0x08048c6d <+213>: je     0x8048c8f <phase_3+247>
0x08048c6f <+215>: call   0x80494fc <explode_bomb>
0x08048c74 <+220>: jmp    0x8048c8f <phase_3+247>
0x08048c76 <+222>: mov    $0x62,%bl
0x08048c78 <+224>: cmpl   $0x20c,-0x4(%ebp)
0x08048c7f <+231>: je     0x8048c8f <phase_3+247>
0x08048c81 <+233>: call   0x80494fc <explode_bomb>
0x08048c86 <+238>: jmp    0x8048c8f <phase_3+247>
0x08048c88 <+240>: mov    $0x78,%bl
--Type <RET> for more, q to quit, c to continue without paging--
0x08048c8a <+242>: call   0x80494fc <explode_bomb>
0x08048c8f <+247>: cmp    -0x5(%ebp),%bl
0x08048c92 <+250>: je     0x8048c99 <phase_3+257>
0x08048c94 <+252>: call   0x80494fc <explode_bomb>
0x08048c99 <+257>: mov    -0x18(%ebp),%ebx
0x08048c9c <+260>: mov    %ebp,%esp
0x08048c9e <+262>: pop    %ebp
0x08048c9f <+263>: ret
End of assembler dump.
(gdb) x/s 0x80497de
0x80497de:      "%d %c %d"
(gdb) □
```

Figure 2: Identification of the input pattern

By checking the address (0x80497de), we can see the input pattern of the phase_3. Input is limited to two integers and for a one character [Figure 2]. Also, it is possible to see the values getting pushed to their respective locations based on their data type sizes at the lines 13,17 and 21 [Figure 3].

```
root@kali: ~/Documents/ohts/bigbangtheory-master
File Edit View Search Terminal Tabs Help
root@kali: ~/Documents/ohts/bigbangtheory-master
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from sheldon1...done.
(gdb) b phase_3
Breakpoint 1 at 0x08048b9f
(gdb) disass phase_3
Dump of assembler code for function phase_3:
   0x08048b98 <+0>:    push    %ebp
   0x08048b99 <+1>:    mov     %esp,%ebp
   0x08048b9b <+3>:    sub     $0x14,%esp
   0x08048b9e <+6>:    push    %ebx
   0x08048b9f <+7>:    mov     0x8(%ebp),%edx
   0x08048ba2 <+10>:   add     $0xffffffff4,%esp
   0x08048ba5 <+13>:   lea     -0x4(%ebp),%eax
   0x08048ba8 <+16>:   push    %eax
   0x08048ba9 <+17>:   lea     -0x5(%ebp),%eax
   0x08048bac <+20>:   push    %eax
   0x08048bad <+21>:   lea     -0xc(%ebp),%eax
   0x08048bb0 <+24>:   push    %eax
   0x08048bb1 <+25>:   push    $0x80497de
   0x08048bb6 <+30>:   push    %edx
   0x08048bb7 <+31>:   call    0x8048860 <sscanf@plt>
   0x08048bbc <+36>:   add     $0x20,%esp
   0x08048bbf <+39>:   cmp     $0x2,%eax
   0x08048bc2 <+42>:   jg      0x8048bc9 <phase_3+49>
```

Figure 3 : Values and their positions

At the line 39 we can see a condition related to eax register value and for the value 2 [Figure 4].


```
root@kali: ~/Documents/ohts/bigbangtheory-master
File Edit View Search Terminal Tabs Help
root@kali: ~/Documents/ohts/bigbangtheory-master
(gdb) b phase_3
Breakpoint 1 at 0x8048b9f
(gdb) disass phase_3
Dump of assembler code for function phase_3:
0x08048b98 <+0>:      push    %ebp
0x08048b99 <+1>:      mov     %esp,%ebp
0x08048b9b <+3>:      sub     $0x14,%esp
0x08048b9e <+6>:      push    %ebx
0x08048b9f <+7>:      mov     0x8(%ebp),%edx
0x08048ba2 <+10>:     add     $0xffffffff4,%esp
0x08048ba5 <+13>:     lea     -0x4(%ebp),%eax
0x08048ba8 <+16>:     push    %eax
0x08048ba9 <+17>:     lea     -0x5(%ebp),%eax
0x08048bac <+20>:     push    %eax
0x08048bad <+21>:     lea     -0xc(%ebp),%eax
0x08048bb0 <+24>:     push    %eax
0x08048bb1 <+25>:     push    $0x80497de
0x08048bb6 <+30>:     push    %edx
0x08048bb7 <+31>:     call    0x8048860 <sscanf@plt>
0x08048bbc <+36>:     add     $0x20,%esp
0x08048bbf <+39>:     cmp     $0x2,%eax
0x08048bc2 <+42>:     jg      0x8048bc9 <phase_3+49>
0x08048bc4 <+44>:     call    0x80494fc <explode_bomb>
0x08048bc9 <+49>:     cmpl    $0x7,-0xc(%ebp)
0x08048bcd <+53>:     ja      0x8048c88 <phase_3+240>
```

Figure 4 : Clue to start with

Also, we can see there's another condition with the first position of the input and if the input value is greater than 7 the bomb will explode (insert the first integer value as 8 and use `ni`, `i r` and `disas phase_3` commands to follow the bomb execution behavior). These conditions guaranteed that the first integer value should be a value under number 7 [Figure 5].

```
root@kali: ~/Documents/oht
File Edit View Search Terminal Tabs Help
root@kali: ~/Documents/ohts/bigbangtheory-master
(gdb) b phase_3
Breakpoint 1 at 0x8048b9f
(gdb) disass phase_3
Dump of assembler code for function phase_3:
0x08048b98 <+0>:    push    %ebp
0x08048b99 <+1>:    mov     %esp,%ebp
0x08048b9b <+3>:    sub     $0x14,%esp
0x08048b9e <+6>:    push    %ebx
0x08048b9f <+7>:    mov     0x8(%ebp),%edx
0x08048ba2 <+10>:   add     $0xffffffff4,%esp
0x08048ba5 <+13>:   lea     -0x4(%ebp),%eax
0x08048ba8 <+16>:   push    %eax
0x08048ba9 <+17>:   lea     -0x5(%ebp),%eax
0x08048bac <+20>:   push    %eax
0x08048bad <+21>:   lea     -0xc(%ebp),%eax
0x08048bb0 <+24>:   push    %eax
0x08048bb1 <+25>:   push    $0x80497de
0x08048bb6 <+30>:   push    %edx
0x08048bb7 <+31>:   call    0x8048860 <sscanf@plt>
0x08048bbc <+36>:   add     $0x20,%esp
0x08048bbf <+39>:   cmp     $0x2,%eax
0x08048bc2 <+42>:   jg      0x8048bc9 <phase_3+49>
0x08048bc4 <+44>:   call    0x80494fc <explode_bomb>
0x08048bc9 <+49>:   cmpl    $0x7,-0xc(%ebp)
0x08048bcd <+53>:   ja      0x8048c88 <phase_3+240>
0x08048bd3 <+59>:   mov     -0xc(%ebp),%eax
0x08048bd6 <+62>:   jmp     *0x80497e8(,%eax,4)
0x08048bdd <+69>:   lea     0x0(%esi),%esi
```

Figure 5 : Identification of the highest number for the first integer position

After inserting an input which starts with the integer value 2, we can follow through the execution by using the command next instruction(ni). First integer (value 2) will eventually direct the execution to line number 126 [Figure 6].

```
root@kali: ~/Documents/ohts/big
File Edit View Search Terminal Tabs Help
root@kali: ~/Documents/ohts/bigbangtheory-master
0x08048c02 <+100>:  cmpl    $0x00, -0x4(%ebp)
0x08048c09 <+113>:   je      0x8048c8f <phase_3+247>
--Type <RET> for more, q to quit, c to continue without paging--
0x08048c0f <+119>:   call   0x80494fc <explode_bomb>
0x08048c14 <+124>:   jmp     0x8048c8f <phase_3+247>
=> 0x08048c16 <+126>:  mov     $0x62,%bl
0x08048c18 <+128>:  cmpl    $0x2f3, -0x4(%ebp)
0x08048c1f <+135>:   je      0x8048c8f <phase_3+247>
0x08048c21 <+137>:   call   0x80494fc <explode_bomb>
0x08048c26 <+142>:   jmp     0x8048c8f <phase_3+247>
0x08048c28 <+144>:   mov     $0x6b,%bl
0x08048c2a <+146>:  cmpl    $0xfb, -0x4(%ebp)
0x08048c31 <+153>:   je      0x8048c8f <phase_3+247>
0x08048c33 <+155>:   call   0x80494fc <explode_bomb>
0x08048c38 <+160>:   jmp     0x8048c8f <phase_3+247>
0x08048c3a <+162>:   lea     0x0(%esi),%esi
0x08048c40 <+168>:   mov     $0x6f,%bl
0x08048c42 <+170>:  cmpl    $0xa0, -0x4(%ebp)
0x08048c49 <+177>:   je      0x8048c8f <phase_3+247>
0x08048c4b <+179>:   call   0x80494fc <explode_bomb>
0x08048c50 <+184>:   jmp     0x8048c8f <phase_3+247>
```

Figure 6 : Identification of the second and third values

At line number 126 we can see that a value getting moved to the %bl [Figure 6]. A value is being compared with the second integer (third input value) at the line 128 [Figure 6]. Now, let's see what's inside those variables.


```

0x08048c31 <+153>: je 0x8048c8f <phase_3+247>
0x08048c33 <+155>: call 0x80494fc <explode_bomb>
0x08048c38 <+160>: jmp 0x8048c8f <phase_3+247>
0x08048c3a <+162>: lea 0x0(%esi),%esi
0x08048c40 <+168>: mov $0x6f,%bl
0x08048c42 <+170>: cmpl $0xa0,-0x4(%ebp)
0x08048c49 <+177>: je 0x8048c8f <phase_3+247>
0x08048c4b <+179>: call 0x80494fc <explode_bomb>
0x08048c50 <+184>: jmp 0x8048c8f <phase_3+247>
0x08048c52 <+186>: mov $0x74,%bl
0x08048c54 <+188>: cmpl $0x1ca,-0x4(%ebp)
0x08048c5b <+195>: je 0x8048c8f <phase_3+247>
0x08048c5d <+197>: call 0x80494fc <explode_bomb>
0x08048c62 <+202>: jmp 0x8048c8f <phase_3+247>
0x08048c64 <+204>: mov $0x76,%bl
0x08048c66 <+206>: cmpl $0x30c,-0x4(%ebp)
0x08048c6d <+213>: je 0x8048c8f <phase_3+247>
0x08048c6f <+215>: call 0x80494fc <explode_bomb>
0x08048c74 <+220>: jmp 0x8048c8f <phase_3+247>
0x08048c76 <+222>: mov $0x62,%bl
0x08048c78 <+224>: cmpl $0x20c,-0x4(%ebp)
0x08048c7f <+231>: je 0x8048c8f <phase_3+247>
0x08048c81 <+233>: call 0x80494fc <explode_bomb>
0x08048c86 <+238>: jmp 0x8048c8f <phase_3+247>
0x08048c88 <+240>: mov $0x78,%bl
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) p 0x62
$3 = 98
(gdb) p 0x2f3
$4 = 755
(gdb) 

```

Figure 7 : Printing the values in decimal

We found two values by following the code which was initiated with number two as the first integer. Now we know that the first integer value is 2 while the second integer value being 755 [Figure 7]. Though, we are sure about 98 belongs to the second input value, let's see how we can confirm it. By looking at the line number 247 [Figure 8], we can see the comparison happens with `ox5(position)` and `%bl` value. That confirms the second value is 98. But we found it was a character from the previous findings. Now we must find the ASCII character of the decimal value 98 by referring an ASCII table. The character turned out as 'b'.

```
root@kali: ~/Documents/ohts/
File Edit View Search Terminal Tabs Help

root@kali: ~/Documents/ohts/bigbangtheory-master x

0x08048c78 <+224>: cmpl $0x20c,-0x4(%ebp)
0x08048c7f <+231>: je 0x8048c8f <phase_3+247>
0x08048c81 <+233>: call 0x80494fc <explode_bomb>
0x08048c86 <+238>: jmp 0x8048c8f <phase_3+247>
=> 0x08048c88 <+240>: mov $0x78,%bl
--Type <RET> for more, q to quit, c to continue without paging--
0x08048c8a <+242>: call 0x80494fc <explode_bomb>
0x08048c8f <+247>: cmp -0x5(%ebp),%bl
0x08048c92 <+250>: je 0x8048c99 <phase_3+257>
0x08048c94 <+252>: call 0x80494fc <explode_bomb>
0x08048c99 <+257>: mov -0x18(%ebp),%ebx
0x08048c9c <+260>: mov %ebp,%esp
0x08048c9e <+262>: pop %ebp
0x08048c9f <+263>: ret
End of assembler dump.
(gdb) ni
0x08048c8a in phase_3 ()
(gdb)

BOOM!!!
The bomb has blown up.
[Inferior 1 (process 3420) exited with code 010]
(gdb) run pass
Starting program: /root/Documents/ohts/bigbangtheory-master/sheldon1 pass
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!
2 b 755

Breakpoint 1, 0x08048b9f in phase_3 ()
(gdb) continue
Continuing.
Halfway there!
█
```

Figure 8: Testing the functionality with the found input

After providing the '2 b 755' as the input the phase_3 was diffused [Figure 8].

At the beginning I've mentioned that there are several combinations which can be taken as inputs. Below are some other found valid inputs.

' 1 b 214 '

' 3 k 251 '