

A dynamic comic book illustration of an explosion. The word "BOOM" is rendered in large, bold, orange 3D block letters with black outlines and internal shading, positioned in the lower-left foreground. Above it, the text "Phase 1" is written in a black, italicized serif font. The explosion itself is depicted with a large, billowing white cloud that has grey shading to indicate depth and movement. Behind the cloud, a bright orange and yellow flame-like shape represents the core of the blast. A thick, multi-colored rainbow streaks diagonally across the upper right portion of the image. The background is black with numerous thin, grey lines radiating outwards from the center of the explosion, creating a sense of intense energy and impact. Several five-pointed stars in red and yellow are scattered throughout the scene, further emphasizing the explosive nature of the event.

*Phase 1*

**BOOM**



# Set up breakpoints

(gdb) break [location]

```
philip_nelson@virtualbox:~/Documents/Architecture/bomb3$ gdb bomb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 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 bomb...done.
(gdb) break *phase_1
Breakpoint 1 at 0x400edd
(gdb) break *explode_bomb
Breakpoint 2 at 0x401606
```

Now lets run

(gdb) run

```
(gdb) r
Starting program: /home/philip_nelson/Documents/Architecture/bomb3/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
```

Lets give the bomb a “test string”

```
(gdb) r
Starting program: /home/philip_nelson/Documents/Architecture/bomb3/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
test string

Breakpoint 1, 0x000000000000400edd in phase_1 ()
(gdb)
```

And we can see GDB broke on our first breakpoint: phase\_1



```
B+> 0x400edd <phase_1> sub $0x8,%rsp
0x400ee1 <phase_1+4> mov $0x4025b0,%esi
0x400ee6 <phase_1+9> callq 0x40137b <strings_not_equal>
0x400eeb <phase_1+14> test %eax,%eax
0x400eed <phase_1+16> je 0x400ef4 <phase_1+23>
0x400eef <phase_1+18> callq 0x401606 <explode_bomb>
0x400ef4 <phase_1+23> add $0x8,%rsp
0x400ef8 <phase_1+27> retq
0x400ef9 <phase_2> push %rbp
0x400efa <phase_2+1> push %rbx
0x400efb <phase_2+2> sub $0x28,%rsp
0x400eff <phase_2+6> mov %fs:0x28,%rax
0x400f08 <phase_2+15> mov %rax,0x18(%rsp)
0x400f0d <phase_2+20> xor %eax,%eax
0x400f0f <phase_2+22> mov %rsp,%rsi
0x400f12 <phase_2+25> callq 0x40163c <read_six_numbers>
0x400f17 <phase_2+30> cmpl $0x0,(%rsp)
0x400f1b <phase_2+34> jne 0x400f24 <phase_2+43>
0x400f1d <phase_2+36> cmpl $0x1,0x4(%rsp)
0x400f22 <phase_2+41> je 0x400f29 <phase_2+48>
0x400f24 <phase_2+43> callq 0x401606 <explode_bomb>
0x400f29 <phase_2+48> mov %rsp,%rbx
0x400f2c <phase_2+51> lea 0x10(%rsp),%rbp
0x400f31 <phase_2+56> mov 0x4(%rbx),%eax
0x400f34 <phase_2+59> add (%rbx),%eax
0x400f36 <phase_2+61> cmp %eax,0x8(%rbx)
0x400f39 <phase_2+64> je 0x400f40 <phase_2+71>
0x400f3b <phase_2+66> callq 0x401606 <explode_bomb>
0x400f40 <phase_2+71> add $0x4,%rbx
0x400f44 <phase_2+75> cmp %rbp,%rbx
0x400f47 <phase_2+78> jne 0x400f31 <phase_2+56>
0x400f49 <phase_2+80> mov 0x18(%rsp),%rax
0x400f4e <phase_2+85> xor %fs:0x28,%rax
0x400f57 <phase_2+94> je 0x400f5e <phase_2+101>
0x400f59 <phase_2+96> callq 0x400b40 <__stack_chk_fail@plt>
0x400f5e <phase_2+101> add $0x28,%rsp
0x400f62 <phase_2+105> pop %rbx

native process 3917 In: phase 1
(gdb) L?? PC: 0x400edd
```

To step through the assembly, let's enter assembly mode

(gdb) layout asm

```

B+ 0x400edd <phase_1>      sub    $0x8,%rsp
    0x400ee1 <phase_1+4>    mov     $0x4025b0,%esi
> 0x400ee6 <phase_1+9>    callq  0x40137b <strings_not_equal>
    0x400eeb <phase_1+14>  test   %eax,%eax
    0x400eed <phase_1+16>  je      0x400ef4 <phase_1+23>
    0x400eef <phase_1+18>  callq  0x401606 <explode_bomb>
    0x400ef4 <phase_1+23>  add     $0x8,%rsp
    0x400ef8 <phase_1+27>  retq
    0x400ef9 <phase_2>     push    %rbp
    0x400efa <phase_2+1>   push    %rbx
    0x400efb <phase_2+2>   sub     $0x28,%rsp
    0x400eff <phase_2+6>   mov     %fs:0x28,%rax
    0x400f08 <phase_2+15>  mov     %rax,0x18(%rsp)
    0x400f0d <phase_2+20>  xor     %eax,%eax
    0x400f0f <phase_2+22>  mov     %rsp,%rsi
    0x400f12 <phase_2+25>  callq  0x40163c <read_six_numbers>
    0x400f17 <phase_2+30>  cmpl    $0x0,(%rsp)
    0x400f1b <phase_2+34>  jne     0x400f24 <phase_2+43>
    0x400f1d <phase_2+36>  cmpl    $0x1,0x4(%rsp)
    0x400f22 <phase_2+41>  je      0x400f29 <phase_2+48>
    0x400f24 <phase_2+43>  callq  0x401606 <explode_bomb>
    0x400f29 <phase_2+48>  mov     %rsp,%rbx
    0x400f2c <phase_2+51>  lea     0x10(%rsp),%rbp
    0x400f31 <phase_2+56>  mov     0x4(%rbx),%eax
    0x400f34 <phase_2+59>  add     (%rbx),%eax
    0x400f36 <phase_2+61>  cmp     %eax,0x8(%rbx)
    0x400f39 <phase_2+64>  je      0x400f40 <phase_2+71>
    0x400f3b <phase_2+66>  callq  0x401606 <explode_bomb>
    0x400f40 <phase_2+71>  add     $0x4,%rbx
    0x400f44 <phase_2+75>  cmp     %rbp,%rbx
    0x400f47 <phase_2+78>  jne     0x400f31 <phase_2+56>
    0x400f49 <phase_2+80>  mov     0x18(%rsp),%rax
    0x400f4e <phase_2+85>  xor     %fs:0x28,%rax
    0x400f57 <phase_2+94>  je      0x400f5e <phase_2+101>
    0x400f59 <phase_2+96>  callq  0x400b40 <__stack_chk_fail@plt>
    0x400f5e <phase_2+101> add     $0x28,%rsp
    0x400f62 <phase_2+105> pop     %rbx

```

```

native process 30474 In: phase_1
(gdb) si
0x0000000000400ee1 in phase_1 ()
0x0000000000400ee6 in phase_1 ()
(gdb)

```

Now let's step through some instructions

(gdb) stepi

look, a function called strings\_not\_equal



```
0x40137b <strings_not_equal>      push    %r12
0x40137d <strings_not_equal+2>    push    %rbp
0x40137e <strings_not_equal+3>    push    %rbx
0x40137f <strings_not_equal+4>    mov     %rdi,%rbx
0x401382 <strings_not_equal+7>    mov     %rsi,%rbp
> 0x401385 <strings_not_equal+10>  callq   0x40135d <string_length>
0x40138a <strings_not_equal+15>    mov     %eax,%r12d
0x40138d <strings_not_equal+18>    mov     %rbp,%rdi
0x401390 <strings_not_equal+21>    callq   0x40135d <string_length>
0x401395 <strings_not_equal+26>    mov     $0x1,%edx
0x40139a <strings_not_equal+31>    cmp     %eax,%r12d
0x40139d <strings_not_equal+34>    jne     0x4013db <strings_not_equal+96>
0x40139f <strings_not_equal+36>    movzbl  (%rbx),%eax
0x4013a2 <strings_not_equal+39>    test    %al,%al
0x4013a4 <strings_not_equal+41>    je      0x4013c8 <strings_not_equal+77>
0x4013a6 <strings_not_equal+43>    cmp     0x0(%rbp),%al
0x4013a9 <strings_not_equal+46>    je      0x4013b2 <strings_not_equal+55>
0x4013ab <strings_not_equal+48>    jmp     0x4013cf <strings_not_equal+84>
0x4013ad <strings_not_equal+50>    cmp     0x0(%rbp),%al
0x4013b0 <strings_not_equal+53>    jne     0x4013d6 <strings_not_equal+91>
0x4013b2 <strings_not_equal+55>    add     $0x1,%rbx
0x4013b6 <strings_not_equal+59>    add     $0x1,%rbp
0x4013ba <strings_not_equal+63>    movzbl  (%rbx),%eax
0x4013bd <strings_not_equal+66>    test    %al,%al
0x4013bf <strings_not_equal+68>    jne     0x4013ad <strings_not_equal+50>
0x4013c1 <strings_not_equal+70>    mov     $0x0,%edx
0x4013c6 <strings_not_equal+75>    jmp     0x4013db <strings_not_equal+96>
0x4013c8 <strings_not_equal+77>    mov     $0x0,%edx
0x4013cd <strings_not_equal+82>    jmp     0x4013db <strings_not_equal+96>
0x4013cf <strings_not_equal+84>    mov     $0x1,%edx
0x4013d4 <strings_not_equal+89>    jmp     0x4013db <strings_not_equal+96>
0x4013d6 <strings_not_equal+91>    mov     $0x1,%edx
0x4013db <strings_not_equal+96>    mov     %edx,%eax
0x4013dd <strings_not_equal+98>    pop     %rbx
0x4013de <strings_not_equal+99>    pop     %rbp
0x4013df <strings_not_equal+100>   pop     %r12
0x4013e1 <strings_not_equal+102>   retq
```

```
native process 16937 In: strings not equal
(gdb) si
0x0000000000400ee1 in phase_1 ()
0x0000000000400ee6 in phase_1 ()
0x000000000040137b in strings_not_equal ()
0x000000000040137d in strings_not_equal ()
0x000000000040137e in strings_not_equal ()
0x000000000040137f in strings_not_equal ()
0x0000000000401382 in strings_not_equal ()
0x0000000000401385 in strings_not_equal ()
(gdb) █
```

Some more steps later, look, a function called string\_length!

It might compare the length of our “test string” to the answer

Let’s look at what is in those registers that were set right before the function call

The first is \$rbx

(gdb) p/x \$rbx

This gives us the memory address stored there

```
(gdb) p/x $rbx
```

```
$2 = 0x604bc0
```

```
(gdb) x /25c 0x604bc0
```

```
0x604bc0 <input_strings>: 116 't' 101 'e' 115 's' 116 't' 32 ' ' 115 's' 116 't' 114 'r'
0x604bc8 <input_strings+8>: 105 'i' 110 'n' 103 'g' 0 '\000' 0 '\000' 0 '\000'
0x604bd0 <input_strings+16>: 0 '\000' 0 '\000' 0 '\000' 0 '\000'
0x604bd8 <input_strings+24>: 0 '\000'
```

```
(gdb) █
```

```
(gdb) x /25c 0x604bc0
```

This prints the first 25 bytes starting at the specified address as chars

't''e''s''t'' ''s''t''r''i''n''g'



The second is \$rbp

(gdb) p/x \$rbp

This gives us the memory address stored there

```
(gdb) p/x $rbp
$3 = 0x4025b0
(gdb) x /25c 0x4025b0
0x4025b0:      89 'Y'   111 'o'   117 'u'   32 ' '   99 'c'   97 'a'   110 'n'   32 ' '
0x4025b8:      82 'R'   117 'u'   115 's'   115 's'   105 'i'   97 'a'   32 ' '   102 'f'
0x4025c0:     114 'r'   111 'o'   109 'm'   32 ' '   108 'l'   97 'a'   110 'n'   100 'd'
0x4025c8:      32 ' '
(gdb) █
```

(gdb) x /25c 4025b0

This prints the first 25 bytes starting at the specified address as chars

'Y' 'o' 'u' " 'c' 'a' 'n' " 'R' 'u' 's' 's' 'i' 'a' " 'f' 'r' 'o' 'm' " 'l' 'a' 'n' 'd'

Lets see if we can find this in the strings

```
philip_nelson@virtualbox:~/Documents/Architecture/bomb3$ strings bomb | less
```

```
/lib64/ld-linux-x86-64.so.2
y0CFFmt
libc.so.6
socket
fflush
strcpy
__printf_chk
exit
fopen
__isoc99_sscanf
connect
signal
puts
__stack_chk_fail
stdin
strtol
fgets
__errno_location
read
__fprintf_chk
stdout
__memmove_chk
__ctype_b_loc
getenv
stderr
alarm
gethostbyname
gethostname
close
sleep
__sprintf_chk
__libc_start_main
write
__gmon_start__
GLIBC_2.3
GLIBC_2.7
GLIBC_2.3.4
GLIBC_2.4
GLIBC_2.2.5
5"5
%$5
%"5
%Z4
%r4
%j4
%b4
%Z4
%R4
%R3
=A>
AUATUSH
D$X1
Tt H
\ $ H
D$ H
t$HH
D$XdH3
h[]A\A]
/You can Russia from land
```

With ‘/’ we can  
search for our  
partial string

/You can Russia from land



```
You can Russia from land here in Alaska.
Wow! You've defused the secret stage!
So you think you can stop the bomb with ctrl-c, do you?
Initialization error: Running on an illegal host [1]
ERROR: Input string is too large.
Your instructor has been notified.
Curses, you've found the secret phase!
But finding it and solving it are quite different...
Congratulations! You've defused the bomb!
Your instructor has been notified and will verify your solution.
Well...
OK. :-)
Invalid phase%s
Initialization error:
defused
exploded
%d:%s:%d:%s
BOOM!!!
The bomb has blown up.
%d %d %d %d %d %d
Error: Premature EOF on stdin
GRADE_BOMB
Error: Input line too long
%d %d %s
DrEvil
greatwhite.ics.cs.cmu.edu
angelshark.ics.cs.cmu.edu
makoshark.ics.cs.cmu.edu
Program timed out after %d seconds
Error: HTTP request failed with error %d: %s
GET /%s/submitr.pl/?userid=%s&lab=%s&result=%s&submit=submit HTTP/1.0
Error: Unable to connect to server %s
%%02X
%s %d %[a-zA-Z ]
pine.cs.usu.edu
AUTORESULT_STRING=%s
USUSpr17
;*3$"
philip_nelson
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
<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!
Halfway there!
%So you got that one. Try this one.
Good work! On to the next...
/usr/include/x86_64-linux-gnu/bits
/usr/lib/gcc/x86_64-linux-gnu/5/include
/usr/include
bomb.c
stdio2.h
stddef.h
types.h
stdio.h
libio.h
stdlib.h
support.h
:
```

Check that out, we found it!

The whole string is “You can Russia from land here in Alaska.”

# Back to gdb

## Let's source our breakpoints

```
philip_nelson@virtualbox:~/Documents/Architecture/bomb3$ gdb bomb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 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 bomb...done.
(gdb) source source.txt
Breakpoint 1 at 0x401606
Breakpoint 2 at 0x400edd
(gdb) █
```

```
philip_nelson@virtualbox:~/Documents/Architecture/bomb3$ gdb bomb
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 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 bomb...done.
(gdb) source source.txt
Breakpoint 1 at 0x401606
Breakpoint 2 at 0x400edd
(gdb) r
Starting program: /home/philip_nelson/Documents/Architecture/bomb3/bomb
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
You can Russia from land in Alaska
```

And check out that  
string



```
philip_nelson@virtualbox:~/Documents/Architecture/bomb3$ gdl
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/
This is free software: you are free to change and redistrib
There is NO WARRANTY, to the extent permitted by law. Type
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
<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) source source.txt
Breakpoint 1 at 0x401606
Breakpoint 2 at 0x400edd
(gdb) r
Starting program: /home/philip_nelson/Documents/Architecture
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
You can Russia from land here in Alaska

Breakpoint 2, 0x0000000000400edd in phase_1 ()
(gdb) c
Continuing.

Breakpoint 1, 0x0000000000401606 in explode_bomb ()
(gdb) █
```

WOAH! Looks like  
we typed in the  
string incorrectly.

Good thing we have  
a breakpoint set  
before the bomb  
explodes



```
philip_nelson@virtualbox:~/Documents/Architecture/bomb3$ gd
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.04) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/
This is free software: you are free to change and redistrib
There is NO WARRANTY, to the extent permitted by law. Type
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 onlin
<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) source source.txt
Breakpoint 1 at 0x401606
Breakpoint 2 at 0x400edd
(gdb) r
Starting program: /home/philip_nelson/Documents/Architectur
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
You can Russia from land here in Alaska.

Breakpoint 2, 0x0000000000400edd in phase_1 ()
(gdb) c
Continuing.
Phase 1 defused. How about the next one?
```

With the string  
typed in correctly  
this time we have  
the first phase  
defused

