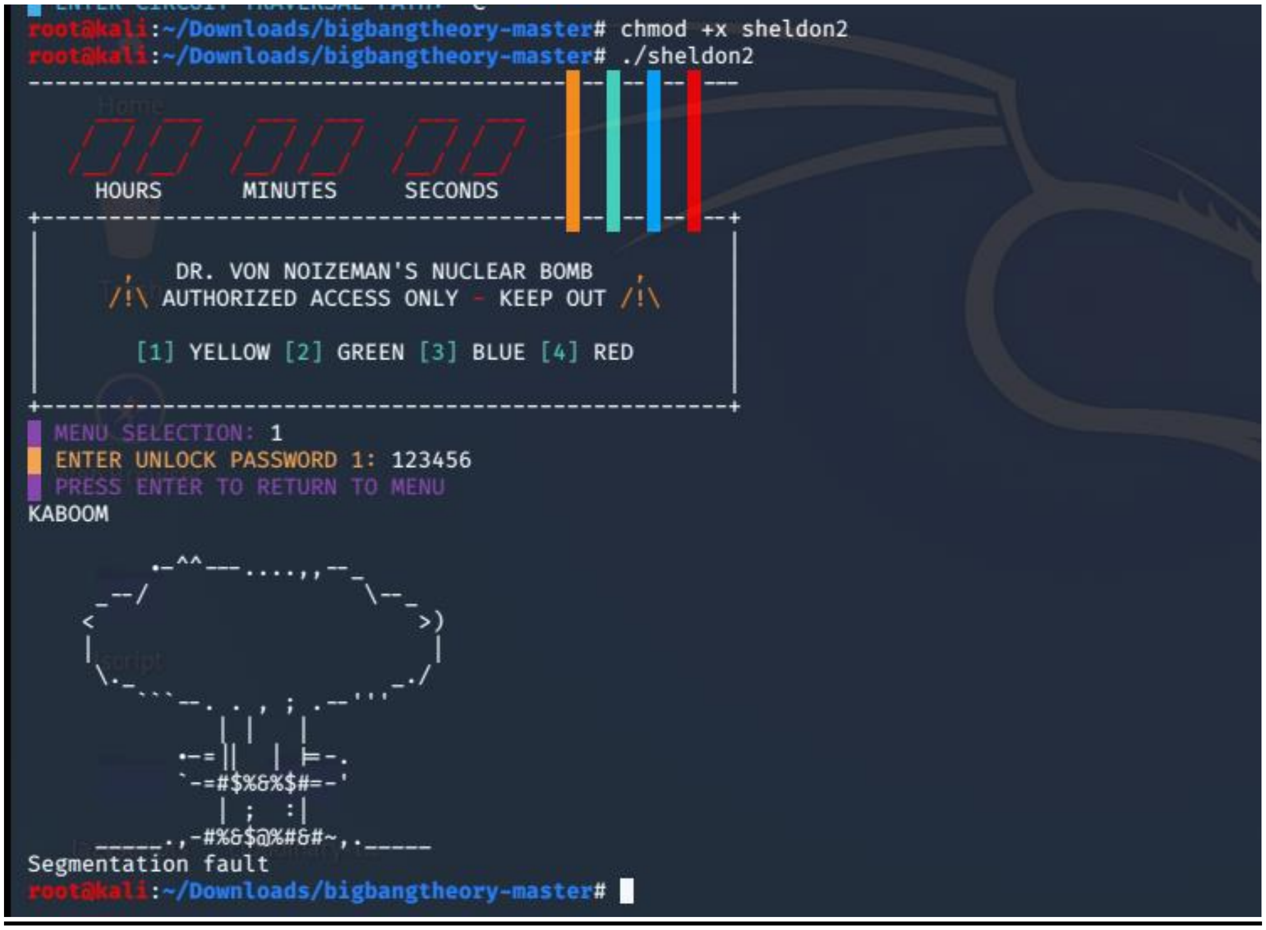
<u>**Sheldon2**</u>

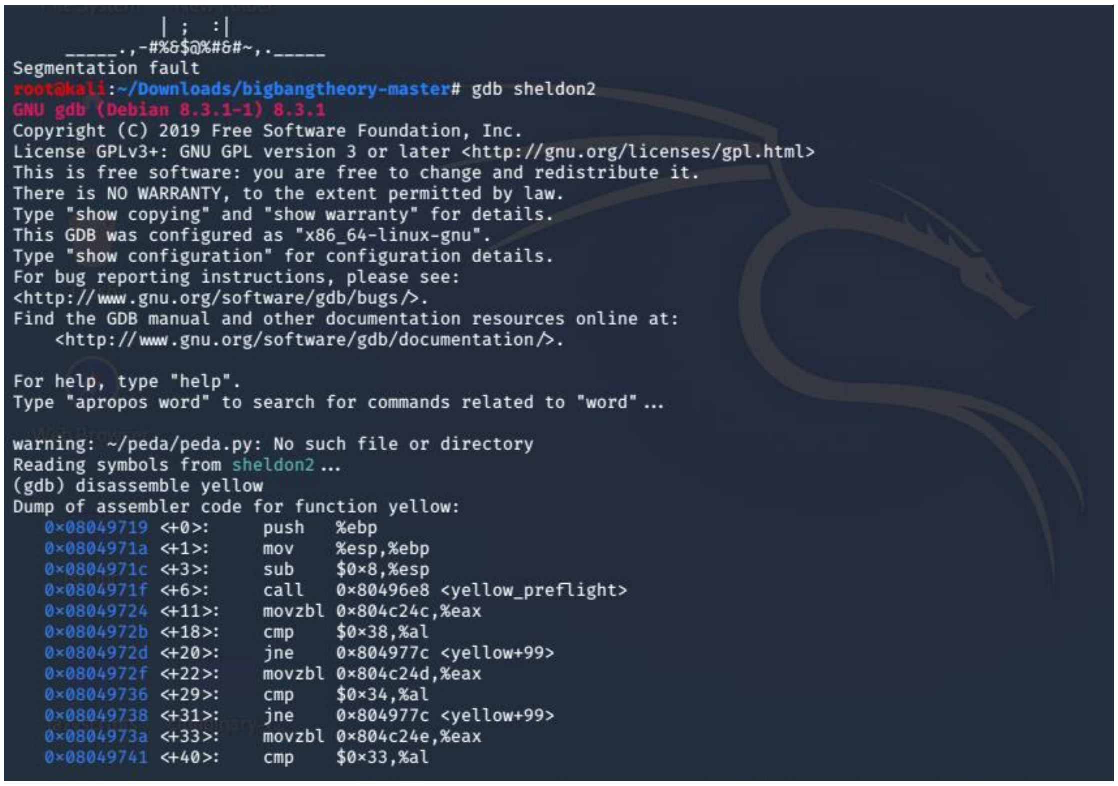
Run the sheldon2 file using “chcmod +x sheldon2” and “./sheldon2”.



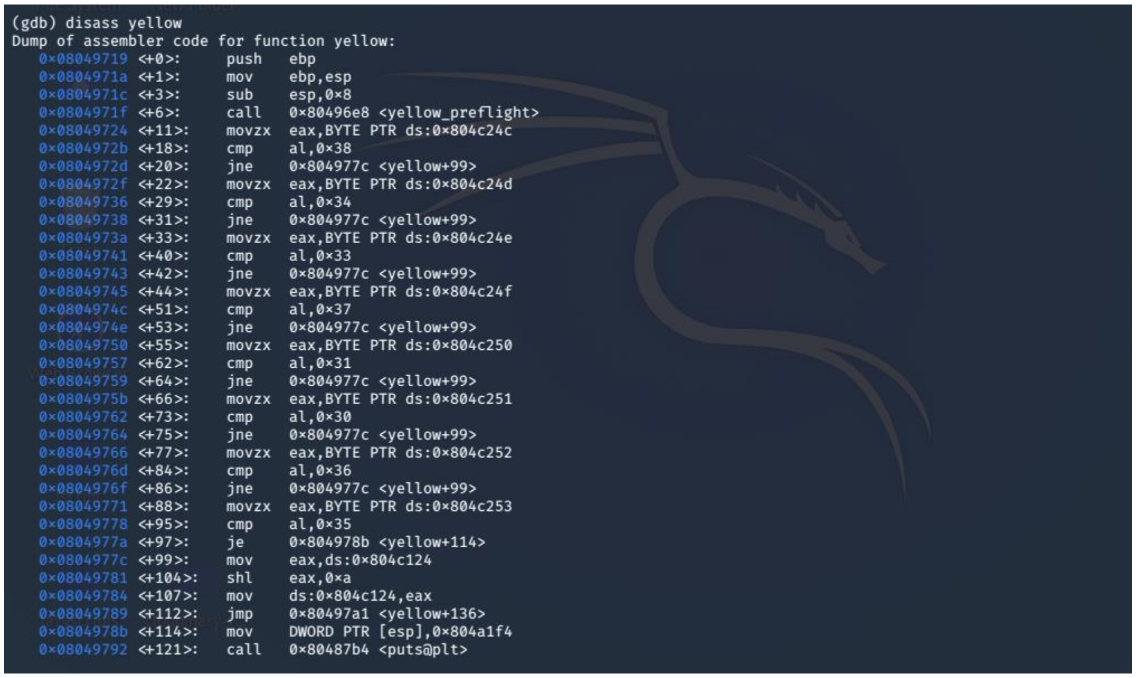
```
root@kali:~/Downloads/bigbangtheory-master# chmod +x sheldon2
root@kali:~/Downloads/bigbangtheory-master# ./sheldon2

   HOURS      MINUTES      SECONDS

        DR. VON NOIZEMAN'S NUCLEAR BOMB
   /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

     [1] YELLOW [2] GREEN [3] BLUE [4] RED

MENU SELECTION: 1
ENTER UNLOCK PASSWORD 1: 123456
PRESS ENTER TO RETURN TO MENU
KABOOM
Segmentation fault
root@kali:~/Downloads/bigbangtheory-master#
```

Phase Yellow

Assembly code for the function yellow:



```
               | ;  :|
     ______.,-#%&$@%#&#~,.______
Segmentation fault
root@kali:~/Downloads/bigbangtheory-master# gdb sheldon2
GNU gdb (Debian 8.3.1-1) 8.3.1
Copyright (C) 2019 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"...

warning: ~/peda/peda.py: No such file or directory
Reading symbols from sheldon2...
(gdb) disassemble yellow
Dump of assembler code for function yellow:
   0x08049719 <+0>:     push   %ebp
   0x0804971a <+1>:     mov    %esp,%ebp
   0x0804971c <+3>:     sub    $0x8,%esp
   0x0804971f <+6>:     call   0x80496e8 <yellow_preflight>
   0x08049724 <+11>:    movzbl 0x804c24c,%eax
   0x0804972b <+18>:    cmp    $0x38,%al
   0x0804972d <+20>:    jne    0x804977c <yellow+99>
   0x0804972f <+22>:    movzbl 0x804c24d,%eax
   0x08049736 <+29>:    cmp    $0x34,%al
   0x08049738 <+31>:    jne    0x804977c <yellow+99>
   0x0804973a <+33>:    movzbl 0x804c24e,%eax
   0x08049741 <+40>:    cmp    $0x33,%al
```



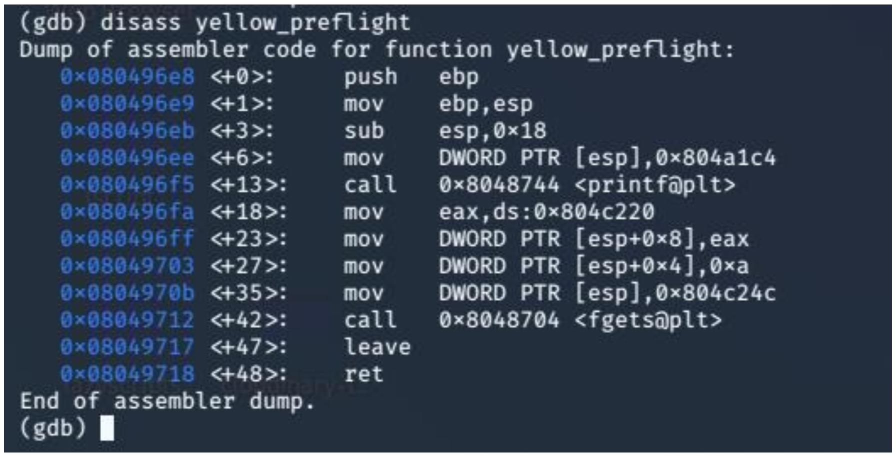
```
(gdb) disass yellow
Dump of assembler code for function yellow:
   0x08049719 <+0>:     push   ebp
   0x0804971a <+1>:     mov    ebp,esp
   0x0804971c <+3>:     sub    esp,0x8
   0x0804971f <+6>:     call   0x80496e8 <yellow_preflight>
   0x08049724 <+11>:    movzx  eax,BYTE PTR ds:0x804c24c
   0x0804972b <+18>:    cmp    al,0x38
   0x0804972d <+20>:    jne    0x804977c <yellow+99>
   0x0804972f <+22>:    movzx  eax,BYTE PTR ds:0x804c24d
   0x08049736 <+29>:    cmp    al,0x34
   0x08049738 <+31>:    jne    0x804977c <yellow+99>
   0x0804973a <+33>:    movzx  eax,BYTE PTR ds:0x804c24e
   0x08049741 <+40>:    cmp    al,0x33
   0x08049743 <+42>:    jne    0x804977c <yellow+99>
   0x08049745 <+44>:    movzx  eax,BYTE PTR ds:0x804c24f
   0x0804974c <+51>:    cmp    al,0x37
   0x0804974e <+53>:    jne    0x804977c <yellow+99>
   0x08049750 <+55>:    movzx  eax,BYTE PTR ds:0x804c250
   0x08049757 <+62>:    cmp    al,0x31
   0x08049759 <+64>:    jne    0x804977c <yellow+99>
   0x0804975b <+66>:    movzx  eax,BYTE PTR ds:0x804c251
   0x08049762 <+73>:    cmp    al,0x30
   0x08049764 <+75>:    jne    0x804977c <yellow+99>
   0x08049766 <+77>:    movzx  eax,BYTE PTR ds:0x804c252
   0x0804976d <+84>:    cmp    al,0x36
   0x0804976f <+86>:    jne    0x804977c <yellow+99>
   0x08049771 <+88>:    movzx  eax,BYTE PTR ds:0x804c253
   0x08049778 <+95>:    cmp    al,0x35
   0x0804977a <+97>:    je     0x804978b <yellow+114>
   0x0804977c <+99>:    mov    eax,ds:0x804c124
   0x08049781 <+104>:   shl    eax,0xa
   0x08049784 <+107>:   mov    ds:0x804c124,eax
   0x08049789 <+112>:   jmp    0x80497a1 <yellow+136>
   0x0804978b <+114>:   mov    DWORD PTR [esp],0x804a1f4
   0x08049792 <+121>:   call   0x80487b4 <puts@plt>
```

Dump of assembler code for function yellow:

```
   0x08049719 <+0>:     push   ebp
   0x0804971a <+1>:     mov    ebp,esp
   0x0804971c <+3>:     sub    esp,0x8
   0x0804971f <+6>:     call   0x80496e8 <yellow_preflight>
   0x08049724 <+11>:    movzx  eax,BYTE PTR ds:0x804c24c
   0x0804972b <+18>:    cmp    al,0x38
   0x0804972d <+20>:    jne    0x804977c <yellow+99>
   0x0804972f <+22>:    movzx  eax,BYTE PTR ds:0x804c24d
   0x08049736 <+29>:    cmp    al,0x34
   0x08049738 <+31>:    jne    0x804977c <yellow+99>
   0x0804973a <+33>:    movzx  eax,BYTE PTR ds:0x804c24e
   0x08049741 <+40>:    cmp    al,0x33
   0x08049743 <+42>:    jne    0x804977c <yellow+99>
   0x08049745 <+44>:    movzx  eax,BYTE PTR ds:0x804c24f
   0x0804974c <+51>:    cmp    al,0x37
   0x0804974e <+53>:    jne    0x804977c <yellow+99>
   0x08049750 <+55>:    movzx  eax,BYTE PTR ds:0x804c250
   0x08049757 <+62>:    cmp    al,0x31
   0x08049759 <+64>:    jne    0x804977c <yellow+99>
   0x0804975b <+66>:    movzx  eax,BYTE PTR ds:0x804c251
   0x08049762 <+73>:    cmp    al,0x30
   0x08049764 <+75>:    jne    0x804977c <yellow+99>
   0x08049766 <+77>:    movzx  eax,BYTE PTR ds:0x804c252
   0x0804976d <+84>:    cmp    al,0x36
   0x0804976f <+86>:    jne    0x804977c <yellow+99>
   0x08049771 <+88>:    movzx  eax,BYTE PTR ds:0x804c253
   0x08049778 <+95>:    cmp    al,0x35
   0x0804977a <+97>:    je     0x804978b <yellow+114>
   0x0804977c <+99>:    mov    eax,ds:0x804c124
   0x08049781 <+104>:   shl    eax,0xa
   0x08049784 <+107>:   mov    ds:0x804c124,eax
   0x08049789 <+112>:   jmp    0x80497a1 <yellow+136>
   0x0804978b <+114>:   mov    DWORD PTR [esp],0x804a1f4
   0x08049792 <+121>:   call   0x80487b4 <puts@plt>
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049797 <+126>:   mov    DWORD PTR ds:0x804c124,0x0
   0x080497a1 <+136>:   leave
   0x080497a2 <+137>:   ret
End of assembler dump.
```

There's a function named yellow_preflight;

Assembly code for the function yellow_preflight;



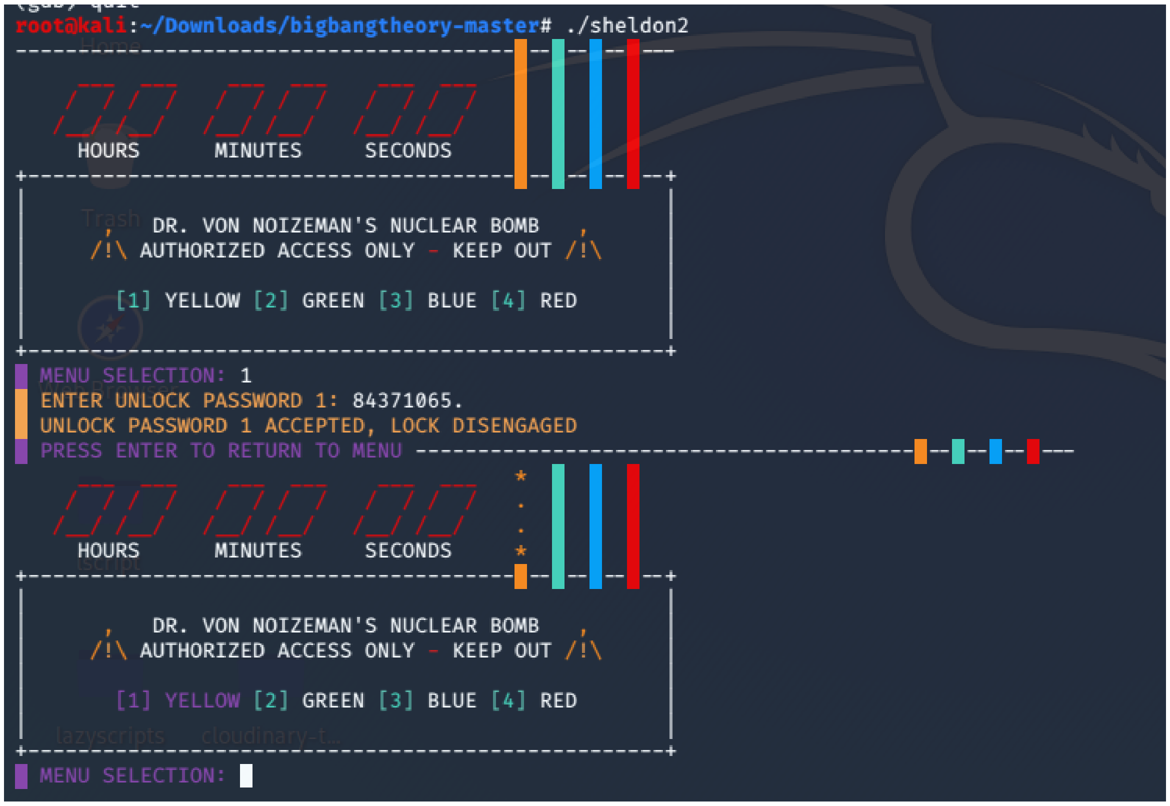
```
(gdb) disass yellow_preflight
Dump of assembler code for function yellow_preflight:
   0x080496e8 <+0>:     push   ebp
   0x080496e9 <+1>:     mov    ebp,esp
   0x080496eb <+3>:     sub    esp,0x18
   0x080496ee <+6>:     mov    DWORD PTR [esp],0x804a1c4
   0x080496f5 <+13>:    call   0x8048744 <printf@plt>
   0x080496fa <+18>:    mov    eax,ds:0x804c220
   0x080496ff <+23>:    mov    DWORD PTR [esp+0x8],eax
   0x08049703 <+27>:    mov    DWORD PTR [esp+0x4],0xa
   0x0804970b <+35>:    mov    DWORD PTR [esp],0x804c24c
   0x08049712 <+42>:    call   0x8048704 <fgets@plt>
   0x08049717 <+47>:    leave
   0x08049718 <+48>:    ret
End of assembler dump.
(gdb) █
```

```
Dump of assembler code for function yellow_preflight:
   0x080496e8 <+0>:     push   ebp
   0x080496e9 <+1>:     mov    ebp,esp
   0x080496eb <+3>:     sub    esp,0x18
   0x080496ee <+6>:     mov    DWORD PTR [esp],0x804a1c4
   0x080496f5 <+13>:    call   0x8048744 <printf@plt>
   0x080496fa <+18>:    mov    eax,ds:0x804c220
   0x080496ff <+23>:    mov    DWORD PTR [esp+0x8],eax
   0x08049703 <+27>:    mov    DWORD PTR [esp+0x4],0xa
   0x0804970b <+35>:    mov    DWORD PTR [esp],0x804c24c
   0x08049712 <+42>:    call   0x8048704 <fgets@plt>
   0x08049717 <+47>:    leave
   0x08049718 <+48>:    ret
End of assembler dump.
```

In function yellow;

Every char of your password is compared with a fixed value.

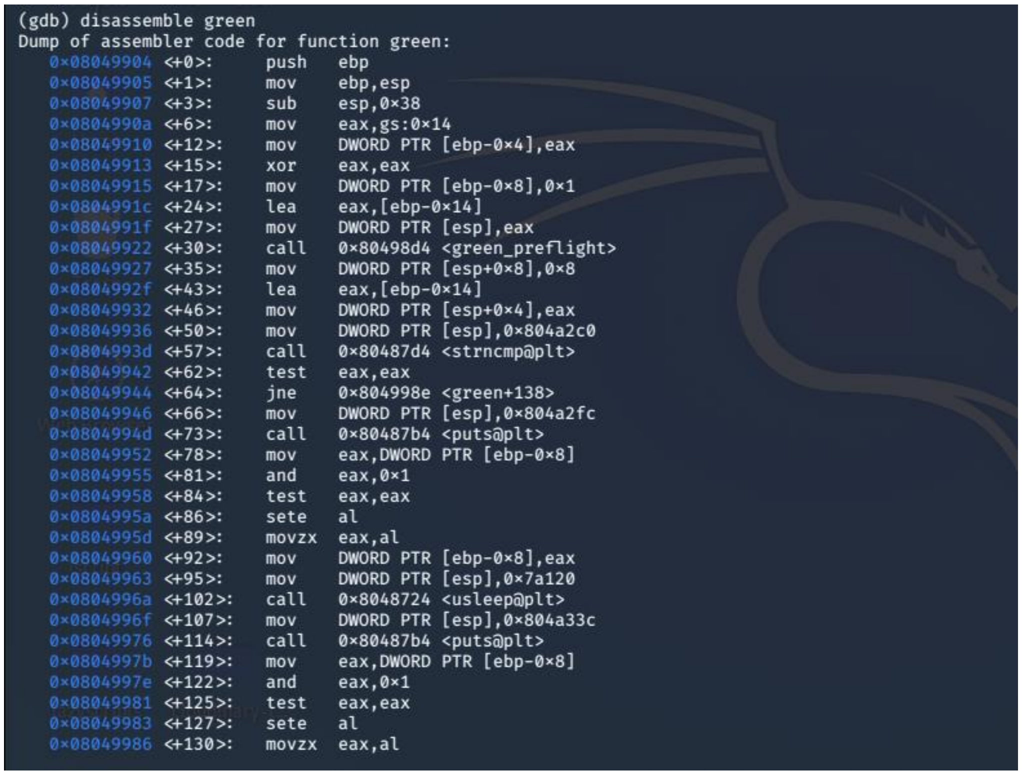
If you group all the values being compared, you'll have the final password: 84371065.



```
(gdb) quit
root@kali:~/Downloads/bigbangtheory-master# ./sheldon2
----------------------------------------------------
      HOURS        MINUTES        SECONDS
+---------------------------------------------------+
|                                                   |
|        DR. VON NOIZEMAN'S NUCLEAR BOMB  ,         |
|     /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\     |
|                                                   |
|         [1] YELLOW [2] GREEN [3] BLUE [4] RED     |
|                                                   |
+---------------------------------------------------+
MENU SELECTION: 1
ENTER UNLOCK PASSWORD 1: 84371065.
UNLOCK PASSWORD 1 ACCEPTED, LOCK DISENGAGED
PRESS ENTER TO RETURN TO MENU ----------------------------------------
      HOURS        MINUTES        SECONDS
+---------------------------------------------------+
|                                                   |
|        DR. VON NOIZEMAN'S NUCLEAR BOMB  ,         |
|     /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\     |
|                                                   |
|         [1] YELLOW [2] GREEN [3] BLUE [4] RED     |
|                                                   |
+---------------------------------------------------+
MENU SELECTION:
```

Phase Green

Assembly code for function green:



```
(gdb) disassemble green
Dump of assembler code for function green:
   0x08049904 <+0>:     push   ebp
   0x08049905 <+1>:     mov    ebp,esp
   0x08049907 <+3>:     sub    esp,0x38
   0x0804990a <+6>:     mov    eax,gs:0x14
   0x08049910 <+12>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049913 <+15>:    xor    eax,eax
   0x08049915 <+17>:    mov    DWORD PTR [ebp-0x8],0x1
   0x0804991c <+24>:    lea    eax,[ebp-0x14]
   0x0804991f <+27>:    mov    DWORD PTR [esp],eax
   0x08049922 <+30>:    call   0x80498d4 <green_preflight>
   0x08049927 <+35>:    mov    DWORD PTR [esp+0x8],0x8
   0x0804992f <+43>:    lea    eax,[ebp-0x14]
   0x08049932 <+46>:    mov    DWORD PTR [esp+0x4],eax
   0x08049936 <+50>:    mov    DWORD PTR [esp],0x804a2c0
   0x0804993d <+57>:    call   0x80487d4 <strncmp@plt>
   0x08049942 <+62>:    test   eax,eax
   0x08049944 <+64>:    jne    0x804998e <green+138>
   0x08049946 <+66>:    mov    DWORD PTR [esp],0x804a2fc
   0x0804994d <+73>:    call   0x80487b4 <puts@plt>
   0x08049952 <+78>:    mov    eax,DWORD PTR [ebp-0x8]
   0x08049955 <+81>:    and    eax,0x1
   0x08049958 <+84>:    test   eax,eax
   0x0804995a <+86>:    sete   al
   0x0804995d <+89>:    movzx  eax,al
   0x08049960 <+92>:    mov    DWORD PTR [ebp-0x8],eax
   0x08049963 <+95>:    mov    DWORD PTR [esp],0x7a120
   0x0804996a <+102>:   call   0x8048724 <usleep@plt>
   0x0804996f <+107>:   mov    DWORD PTR [esp],0x804a33c
   0x08049976 <+114>:   call   0x80487b4 <puts@plt>
   0x0804997b <+119>:   mov    eax,DWORD PTR [ebp-0x8]
   0x0804997e <+122>:   and    eax,0x1
   0x08049981 <+125>:   test   eax,eax
   0x08049983 <+127>:   sete   al
   0x08049986 <+130>:   movzx  eax,al
```

Dump of assembler code for function green:

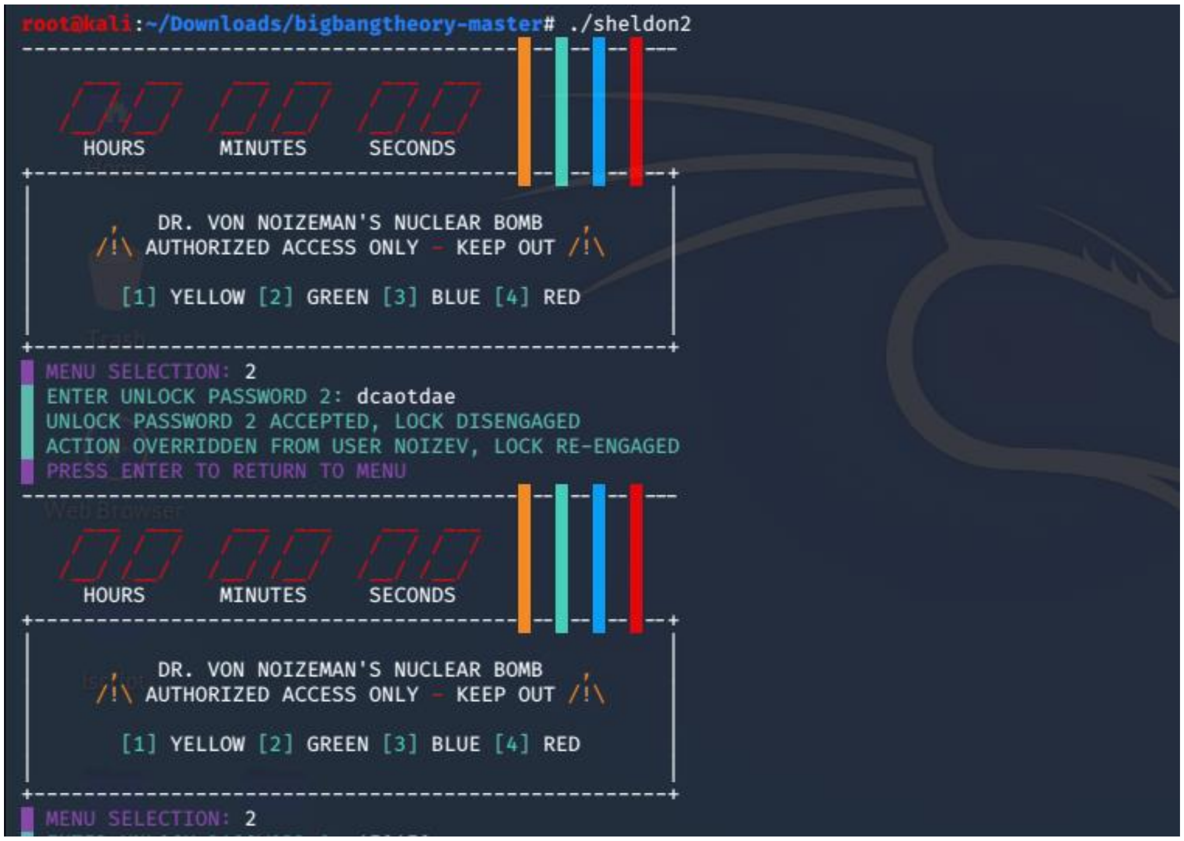
```
   0x08049904 <+0>:     push   ebp
   0x08049905 <+1>:     mov    ebp,esp
   0x08049907 <+3>:     sub    esp,0x38
   0x0804990a <+6>:     mov    eax,gs:0x14
   0x08049910 <+12>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049913 <+15>:    xor    eax,eax
   0x08049915 <+17>:    mov    DWORD PTR [ebp-0x8],0x1
   0x0804991c <+24>:    lea    eax,[ebp-0x14]
   0x0804991f <+27>:    mov    DWORD PTR [esp],eax
   0x08049922 <+30>:    call   0x80498d4 <green_preflight>
   0x08049927 <+35>:    mov    DWORD PTR [esp+0x8],0x8
   0x0804992f <+43>:    lea    eax,[ebp-0x14]
   0x08049932 <+46>:    mov    DWORD PTR [esp+0x4],eax
   0x08049936 <+50>:    mov    DWORD PTR [esp],0x804a2c0
   0x0804993d <+57>:    call   0x80487d4 <strncmp@plt>
   0x08049942 <+62>:    test   eax,eax
   0x08049944 <+64>:    jne    0x804998e <green+138>
   0x08049946 <+66>:    mov    DWORD PTR [esp],0x804a2fc
   0x0804994d <+73>:    call   0x80487b4 <puts@plt>
   0x08049952 <+78>:    mov    eax,DWORD PTR [ebp-0x8]
   0x08049955 <+81>:    and    eax,0x1
   0x08049958 <+84>:    test   eax,eax
   0x0804995a <+86>:    sete   al
   0x0804995d <+89>:    movzx  eax,al
   0x08049960 <+92>:    mov    DWORD PTR [ebp-0x8],eax
   0x08049963 <+95>:    mov    DWORD PTR [esp],0x7a120
   0x0804996a <+102>:   call   0x8048724 <usleep@plt>
   0x0804996f <+107>:   mov    DWORD PTR [esp],0x804a33c
   0x08049976 <+114>:   call   0x80487b4 <puts@plt>
   0x0804997b <+119>:   mov    eax,DWORD PTR [ebp-0x8]
   0x0804997e <+122>:   and    eax,0x1
   0x08049981 <+125>:   test   eax,eax
   0x08049983 <+127>:   sete   al
   0x08049986 <+130>:   movzx  eax,al
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049989 <+133>:   mov    DWORD PTR [ebp-0x8],eax
   0x0804998c <+136>:   jmp    0x804999a <green+150>
   0x0804998e <+138>:   mov    eax,ds:0x804c12c
   0x08049993 <+143>:   add    eax,eax
   0x08049995 <+145>:   mov    ds:0x804c12c,eax
   0x0804999a <+150>:   mov    eax,DWORD PTR [ebp-0x8]
   0x0804999d <+153>:   test   eax,eax
   0x0804999f <+155>:   jne    0x80499ad <green+169>
   0x080499a1 <+157>:   mov    eax,ds:0x804c12c
   0x080499a6 <+162>:   sar    eax,1
   0x080499a8 <+164>:   mov    ds:0x804c12c,eax
   0x080499ad <+169>:   mov    eax,DWORD PTR [ebp-0x4]
   0x080499b0 <+172>:   xor    eax,DWORD PTR gs:0x14
   0x080499b7 <+179>:   je     0x80499be <green+186>
   0x080499b9 <+181>:   call   0x8048784 <__stack_chk_fail@plt>
   0x080499be <+186>:   leave
   0x080499bf <+187>:   ret
End of assembler dump.
```

In line 57 there's a suspicious looking address loaded into ESP with a following strncmp.

We can looks whats inside that memory location.

```
gdb $ x/s 0x804a2c0
0x804a2c0 <password>:         "dcaotdae"
```

We can try this as our password.

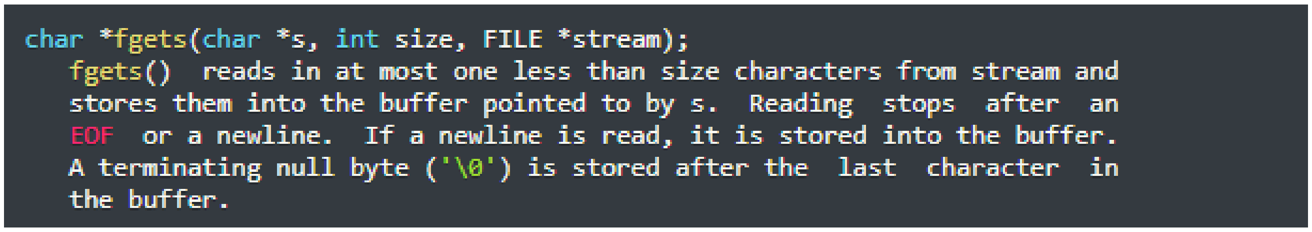


```
root@kali:~/Downloads/bigbangtheory-master# ./sheldon2
-------------------------------------------------------------

     HOURS       MINUTES      SECONDS
+-------------------------------------------------------------+
|                                                             |
|         DR. VON NOIZEMAN'S NUCLEAR BOMB                     |
|   /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\                 |
|                                                             |
|     [1] YELLOW [2] GREEN [3] BLUE [4] RED                   |
|                                                             |
+-------------------------------------------------------------+
 MENU SELECTION: 2
 ENTER UNLOCK PASSWORD 2: dcaotdae
 UNLOCK PASSWORD 2 ACCEPTED, LOCK DISENGAGED
 ACTION OVERRIDDEN FROM USER NOIZEV, LOCK RE-ENGAGED
 PRESS ENTER TO RETURN TO MENU
-------------------------------------------------------------

     HOURS       MINUTES      SECONDS
+-------------------------------------------------------------+
|                                                             |
|         DR. VON NOIZEMAN'S NUCLEAR BOMB                     |
|   /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\                 |
|                                                             |
|     [1] YELLOW [2] GREEN [3] BLUE [4] RED                   |
|                                                             |
+-------------------------------------------------------------+
 MENU SELECTION: 2
```

The system accepts that word as the password but it won't clear the green like previous yellow line. So there must be another passphrase.

Line 55 of green must be reached with [ebp-0x08] at an even value. Since [ebp-0x08] is initialized to 1 on line 9 and its parity is changed twice (lines 27-32 and 40-45), the only way to do this is to overwrite ebp-0x08 through ebp-0x05 with our input string.

The input string is read into ebp-0x14 using fgets().

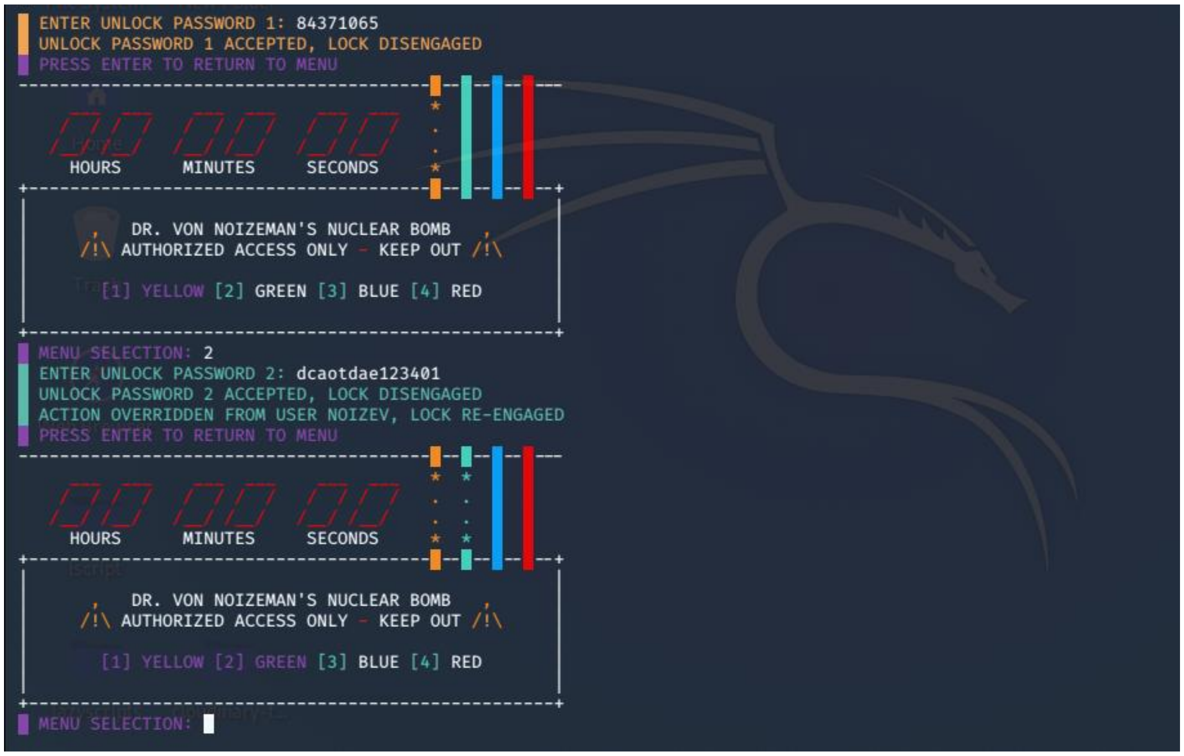


```
char *fgets(char *s, int size, FILE *stream);
   fgets()  reads  in at most one less than size characters from stream and
   stores them into the buffer pointed to by s.  Reading  stops  after  an
   EOF  or a newline.  If a newline is read, it is stored into the buffer.
   A terminating null byte ('\0') is stored after the  last  character  in
   the buffer.
```

The stored string starting at ebp-0x14 should be 16 characters long in order to overwrite ebp-0x08 through ebp-0x05 but not ebp-0x04. Because it will be terminated by both a newline (0x0a) and a null byte (0x00), we should supply 14 characters such that the first eight are "dcaotdae".

When ebp-0x08 is read as a 4-byte integer, our 13th character will decide its parity because data is stored in little-endian format. Since we want [ebp-0x08] to be even, our 13th character should have an even ASCII value, and one valid solution would be "dcaotdae123401".

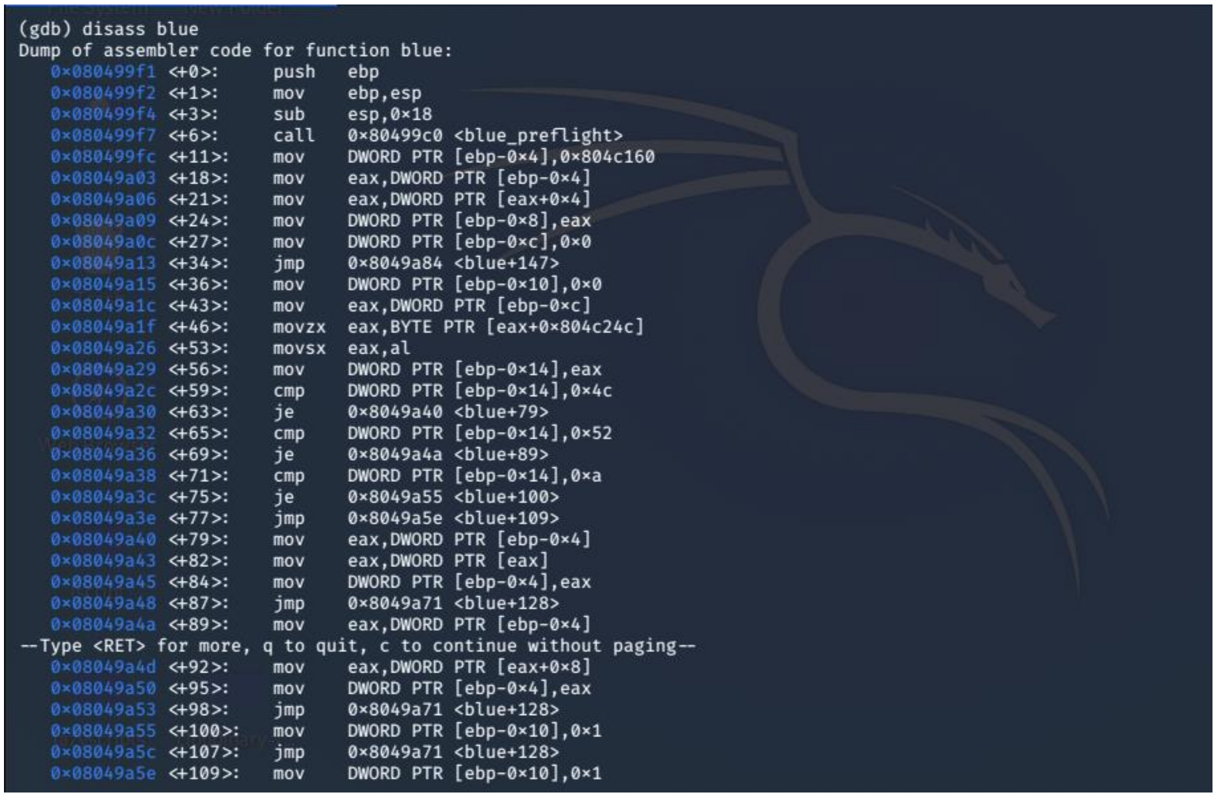
Passphrase for green : dcaotdae123401



```
ENTER UNLOCK PASSWORD 1: 84371065
UNLOCK PASSWORD 1 ACCEPTED, LOCK DISENGAGED
PRESS ENTER TO RETURN TO MENU
          HOURS      MINUTES     SECONDS

       DR. VON NOIZEMAN'S NUCLEAR BOMB
  /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

     [1] YELLOW [2] GREEN [3] BLUE [4] RED

MENU SELECTION: 2
ENTER UNLOCK PASSWORD 2: dcaotdae123401
UNLOCK PASSWORD 2 ACCEPTED, LOCK DISENGAGED
ACTION OVERRIDDEN FROM USER NOIZEV, LOCK RE-ENGAGED
PRESS ENTER TO RETURN TO MENU
          HOURS      MINUTES     SECONDS

       DR. VON NOIZEMAN'S NUCLEAR BOMB
  /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

     [1] YELLOW [2] GREEN [3] BLUE [4] RED

MENU SELECTION:
```

Phase Blue

Assembly code for function blue:



```
(gdb) disass blue
Dump of assembler code for function blue:
   0x080499f1 <+0>:     push   ebp
   0x080499f2 <+1>:     mov    ebp,esp
   0x080499f4 <+3>:     sub    esp,0x18
   0x080499f7 <+6>:     call   0x80499c0 <blue_preflight>
   0x080499fc <+11>:    mov    DWORD PTR [ebp-0x4],0x804c160
   0x08049a03 <+18>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08049a06 <+21>:    mov    eax,DWORD PTR [eax+0x4]
   0x08049a09 <+24>:    mov    DWORD PTR [ebp-0x8],eax
   0x08049a0c <+27>:    mov    DWORD PTR [ebp-0xc],0x0
   0x08049a13 <+34>:    jmp    0x8049a84 <blue+147>
   0x08049a15 <+36>:    mov    DWORD PTR [ebp-0x10],0x0
   0x08049a1c <+43>:    mov    eax,DWORD PTR [ebp-0xc]
   0x08049a1f <+46>:    movzx  eax,BYTE PTR [eax+0x804c24c]
   0x08049a26 <+53>:    movsx  eax,al
   0x08049a29 <+56>:    mov    DWORD PTR [ebp-0x14],eax
   0x08049a2c <+59>:    cmp    DWORD PTR [ebp-0x14],0x4c
   0x08049a30 <+63>:    je     0x8049a40 <blue+79>
   0x08049a32 <+65>:    cmp    DWORD PTR [ebp-0x14],0x52
   0x08049a36 <+69>:    je     0x8049a4a <blue+89>
   0x08049a38 <+71>:    cmp    DWORD PTR [ebp-0x14],0xa
   0x08049a3c <+75>:    je     0x8049a55 <blue+100>
   0x08049a3e <+77>:    jmp    0x8049a5e <blue+109>
   0x08049a40 <+79>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08049a43 <+82>:    mov    eax,DWORD PTR [eax]
   0x08049a45 <+84>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049a48 <+87>:    jmp    0x8049a71 <blue+128>
   0x08049a4a <+89>:    mov    eax,DWORD PTR [ebp-0x4]
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049a4d <+92>:    mov    eax,DWORD PTR [eax+0x8]
   0x08049a50 <+95>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049a53 <+98>:    jmp    0x8049a71 <blue+128>
   0x08049a55 <+100>:   mov    DWORD PTR [ebp-0x10],0x1
   0x08049a5c <+107>:   jmp    0x8049a71 <blue+128>
   0x08049a5e <+109>:   mov    DWORD PTR [ebp-0x10],0x1
```

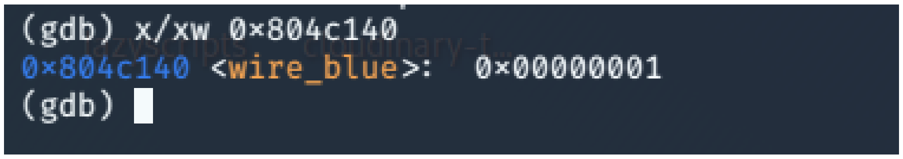
Dump of assembler code for function blue:

```
   0x080499f1 <+0>:     push   ebp
   0x080499f2 <+1>:     mov    ebp,esp
   0x080499f4 <+3>:     sub    esp,0x18
   0x080499f7 <+6>:     call   0x80499c0 <blue_preflight>
   0x080499fc <+11>:    mov    DWORD PTR [ebp-0x4],0x804c160
   0x08049a03 <+18>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08049a06 <+21>:    mov    eax,DWORD PTR [eax+0x4]
   0x08049a09 <+24>:    mov    DWORD PTR [ebp-0x8],eax
   0x08049a0c <+27>:    mov    DWORD PTR [ebp-0xc],0x0
   0x08049a13 <+34>:    jmp    0x8049a84 <blue+147>
   0x08049a15 <+36>:    mov    DWORD PTR [ebp-0x10],0x0
   0x08049a1c <+43>:    mov    eax,DWORD PTR [ebp-0xc]
   0x08049a1f <+46>:    movzx  eax,BYTE PTR [eax+0x804c24c]
   0x08049a26 <+53>:    movsx  eax,al
   0x08049a29 <+56>:    mov    DWORD PTR [ebp-0x14],eax
   0x08049a2c <+59>:    cmp    DWORD PTR [ebp-0x14],0x4c
   0x08049a30 <+63>:    je     0x8049a40 <blue+79>
   0x08049a32 <+65>:    cmp    DWORD PTR [ebp-0x14],0x52
   0x08049a36 <+69>:    je     0x8049a4a <blue+89>
   0x08049a38 <+71>:    cmp    DWORD PTR [ebp-0x14],0xa
   0x08049a3c <+75>:    je     0x8049a55 <blue+100>
   0x08049a3e <+77>:    jmp    0x8049a5e <blue+109>
   0x08049a40 <+79>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08049a43 <+82>:    mov    eax,DWORD PTR [eax]
   0x08049a45 <+84>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049a48 <+87>:    jmp    0x8049a71 <blue+128>
   0x08049a4a <+89>:    mov    eax,DWORD PTR [ebp-0x4]
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049a4d <+92>:    mov    eax,DWORD PTR [eax+0x8]
   0x08049a50 <+95>:    mov    DWORD PTR [ebp-0x4],eax
   0x08049a53 <+98>:    jmp    0x8049a71 <blue+128>
   0x08049a55 <+100>:   mov    DWORD PTR [ebp-0x10],0x1
   0x08049a5c <+107>:   jmp    0x8049a71 <blue+128>
   0x08049a5e <+109>:   mov    DWORD PTR [ebp-0x10],0x1
   0x08049a65 <+116>:   mov    DWORD PTR [esp],0x804a3bb
   0x08049a6c <+123>:   call   0x80487b4 <puts@plt>
   0x08049a71 <+128>:   cmp    DWORD PTR [ebp-0x10],0x0
   0x08049a75 <+132>:   jne    0x8049a8a <blue+153>
   0x08049a77 <+134>:   mov    eax,DWORD PTR [ebp-0x4]
   0x08049a7a <+137>:   mov    eax,DWORD PTR [eax+0x4]
   0x08049a7d <+140>:   xor    DWORD PTR [ebp-0x8],eax
   0x08049a80 <+143>:   add    DWORD PTR [ebp-0xc],0x1
   0x08049a84 <+147>:   cmp    DWORD PTR [ebp-0xc],0xe
   0x08049a88 <+151>:   jle    0x8049a15 <blue+36>
   0x08049a8a <+153>:   mov    DWORD PTR [esp],0x804a3c0
   0x08049a91 <+160>:   call   0x8048744 <printf@plt>
   0x08049a96 <+165>:   mov    eax,ds:0x804c240
   0x08049a9b <+170>:   mov    DWORD PTR [esp],eax
   0x08049a9e <+173>:   call   0x8048734 <fflush@plt>
   0x08049aa3 <+178>:   mov    DWORD PTR [esp],0x1
   0x08049aaa <+185>:   call   0x80487a4 <sleep@plt>
   0x08049aaf <+190>:   mov    DWORD PTR [esp],0x804a3eb
   0x08049ab6 <+197>:   call   0x80487b4 <puts@plt>
   0x08049abb <+202>:   mov    DWORD PTR [esp],0x7a120
   0x08049ac2 <+209>:   call   0x8048724 <usleep@plt>
   0x08049ac7 <+214>:   mov    eax,ds:0x804a384
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049acc <+219>:   cmp    DWORD PTR [ebp-0x8],eax
   0x08049acf <+222>:   jne    0x8049aec <blue+251>
   0x08049ad1 <+224>:   mov    DWORD PTR [esp],0x804a3fc
   0x08049ad8 <+231>:   call   0x80487b4 <puts@plt>
   0x08049add <+236>:   mov    eax,ds:0x804c140
   0x08049ae2 <+241>:   sub    eax,0x1
   0x08049ae5 <+244>:   mov    ds:0x804c140,eax
   0x08049aea <+249>:   jmp    0x8049af9 <blue+264>
   0x08049aec <+251>:   mov    eax,ds:0x804c140
   0x08049af1 <+256>:   add    eax,0x1
   0x08049af4 <+259>:   mov    ds:0x804c140,eax
   0x08049af9 <+264>:   leave
   0x08049afa <+265>:   ret
End of assembler dump.
```

Similar to green_preflight(), blue_preflight() reads at most 16 characters using fgets() from stdin into buffer, located at 0x804c24c. Afterthat, blue() validates the input:

Here whatever happens we will always return to blue();

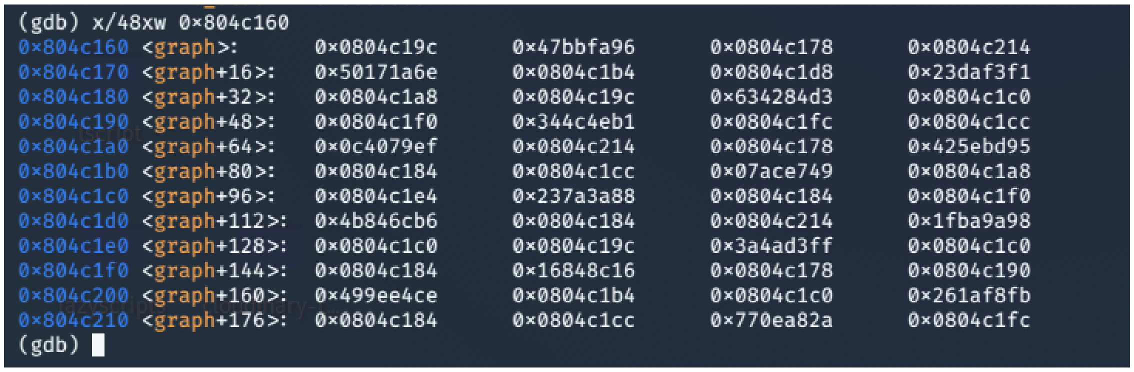
But the most important thing here is the value at memory location 0x804c140. This value represents whether the blue wire is cut or not, and while disassembling menu() proves this, it's simpler to get the variable name through GDB:



```
(gdb) x/xw 0x804c140
0x804c140 <wire_blue>:  0x00000001
(gdb)
```

In order to exit with wire_blue == 0 we must pass the check on line 71 that var2 == 0x40475194. To reach line 71, either var3 > 15 or var4 != 0. It turns out that both of these conditions are equivalent to reaching the end of our input string. If you look at lines 14-27, you can see that var5 = char(buffer[var3]) is the current character of our input string. If var5 is not 'L', 'R', or '\n', the bomb goes off. If either we reach the end of the input string (var3 > 15) or a line feed (var5 == 0x0a), we jump to line 71.

Now we must figure out which combination of 'L' and 'R' characters will result in var2 == 0x40475194 when we reach line 71. To start, let's examine the memory location loaded into var1:

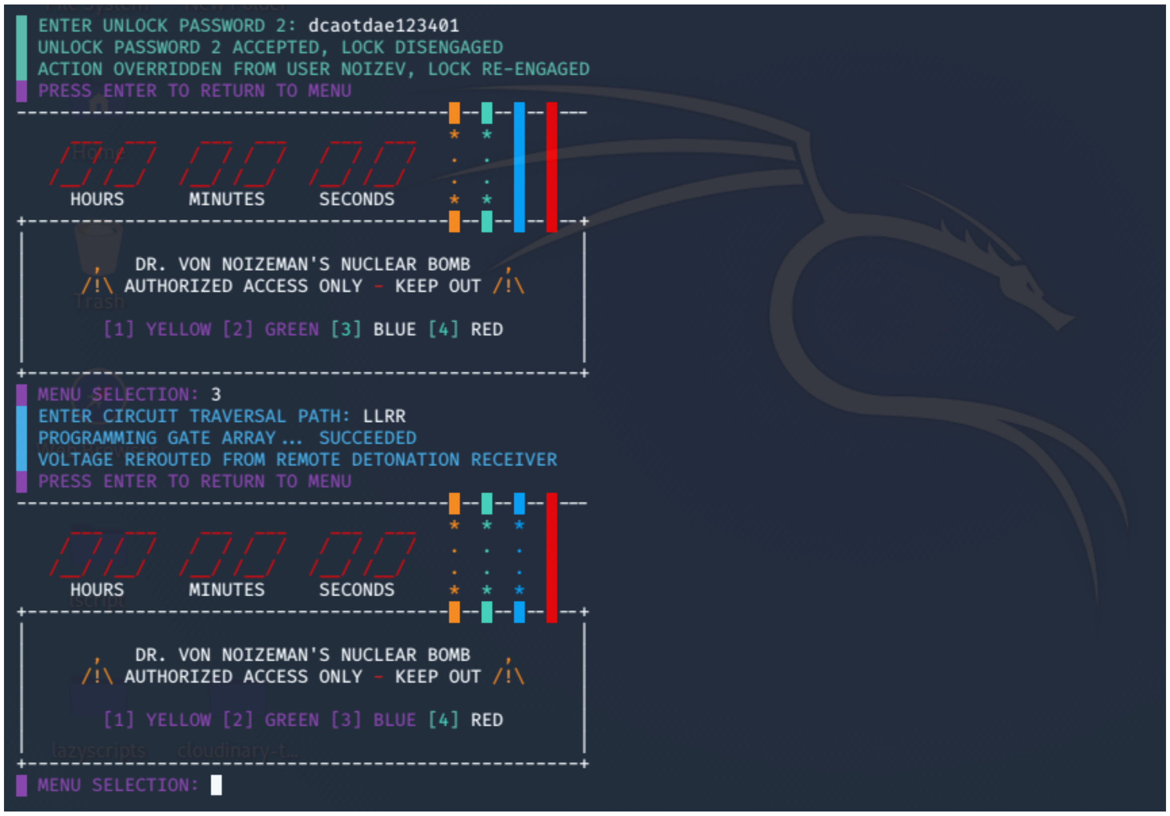


```
(gdb) x/48xw 0x804c160
0x804c160 <graph>:      0x0804c19c      0x47bbfa96      0x0804c178      0x0804c214
0x804c170 <graph+16>:   0x50171a6e      0x0804c1b4      0x0804c1d8      0x23daf3f1
0x804c180 <graph+32>:   0x0804c1a8      0x0804c19c      0x634284d3      0x0804c1c0
0x804c190 <graph+48>:   0x0804c1f0      0x344c4eb1      0x0804c1fc      0x0804c1cc
0x804c1a0 <graph+64>:   0x0c4079ef      0x0804c214      0x0804c178      0x425ebd95
0x804c1b0 <graph+80>:   0x0804c184      0x0804c1cc      0x07ace749      0x0804c1a8
0x804c1c0 <graph+96>:   0x0804c1e4      0x237a3a88      0x0804c184      0x0804c1f0
0x804c1d0 <graph+112>:  0x4b846cb6      0x0804c184      0x0804c214      0x1fba9a98
0x804c1e0 <graph+128>:  0x0804c1c0      0x0804c19c      0x3a4ad3ff      0x0804c1c0
0x804c1f0 <graph+144>:  0x0804c184      0x16848c16      0x0804c178      0x0804c190
0x804c200 <graph+160>:  0x499ee4ce      0x0804c1b4      0x0804c1c0      0x261af8fb
0x804c210 <graph+176>:  0x0804c184      0x0804c1cc      0x770ea82a      0x0804c1fc
(gdb)
```

Given the variable name graph, the fact that our two valid characters are 'L' and 'R', and that two out of every three dwords in graph is a valid pointer, we appear to be dealing with a graph composed of nodes, each with two pointers to neighboring nodes.

Lines 29-37 show that based on each input character, either the left or right pointer is followed before var2 (starting at 0x477bbfa96) is xored with the node's value (lines 49-51). A quick brute-force search using paths of increasing length shows that 'LLRR' is a valid solution.

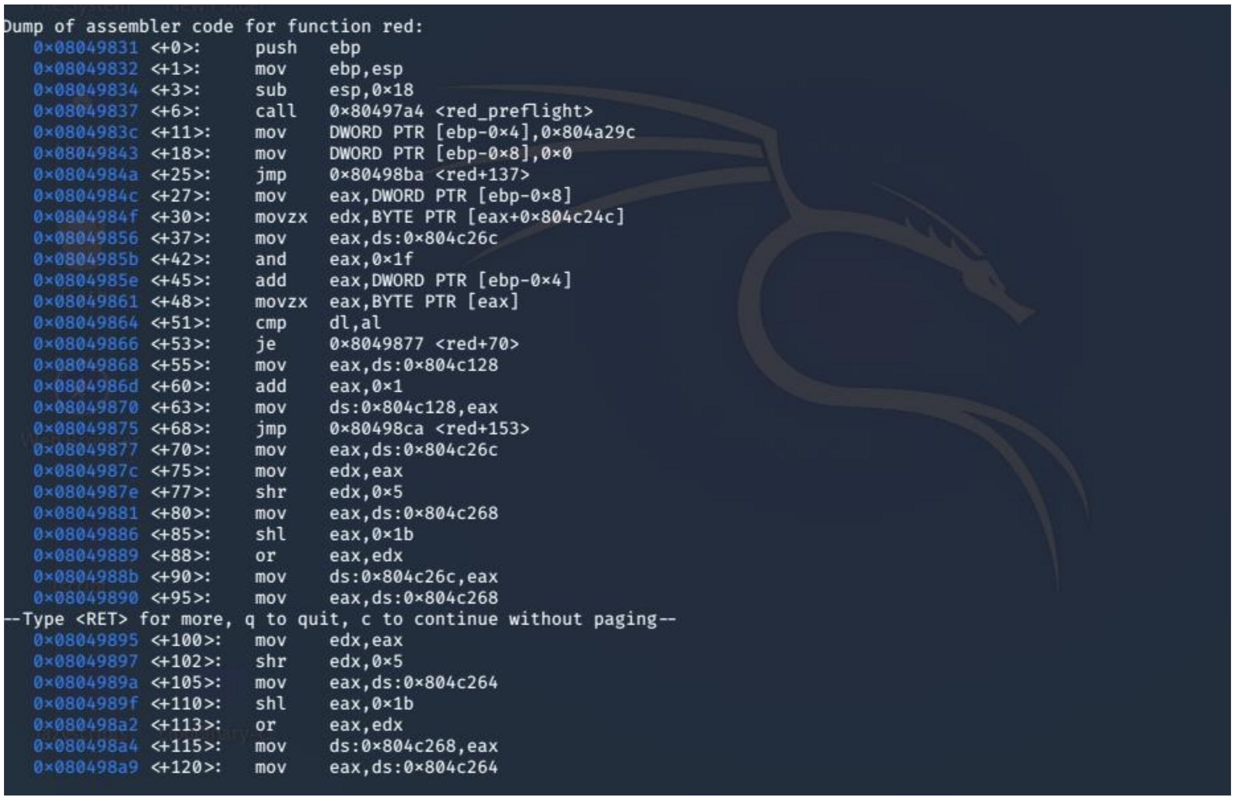
Passphrase for blue : LLRR



```
ENTER UNLOCK PASSWORD 2: dcaotdae123401
UNLOCK PASSWORD 2 ACCEPTED, LOCK DISENGAGED
ACTION OVERRIDDEN FROM USER NOIZEV, LOCK RE-ENGAGED
PRESS ENTER TO RETURN TO MENU

  HOURS      MINUTES      SECONDS

        DR. VON NOIZEMAN'S NUCLEAR BOMB
    /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

      [1] YELLOW [2] GREEN [3] BLUE [4] RED

MENU SELECTION: 3
ENTER CIRCUIT TRAVERSAL PATH: LLRR
PROGRAMMING GATE ARRAY... SUCCEEDED
VOLTAGE REROUTED FROM REMOTE DETONATION RECEIVER
PRESS ENTER TO RETURN TO MENU

  HOURS      MINUTES      SECONDS

        DR. VON NOIZEMAN'S NUCLEAR BOMB
    /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

      [1] YELLOW [2] GREEN [3] BLUE [4] RED

MENU SELECTION:
```

Phase Red

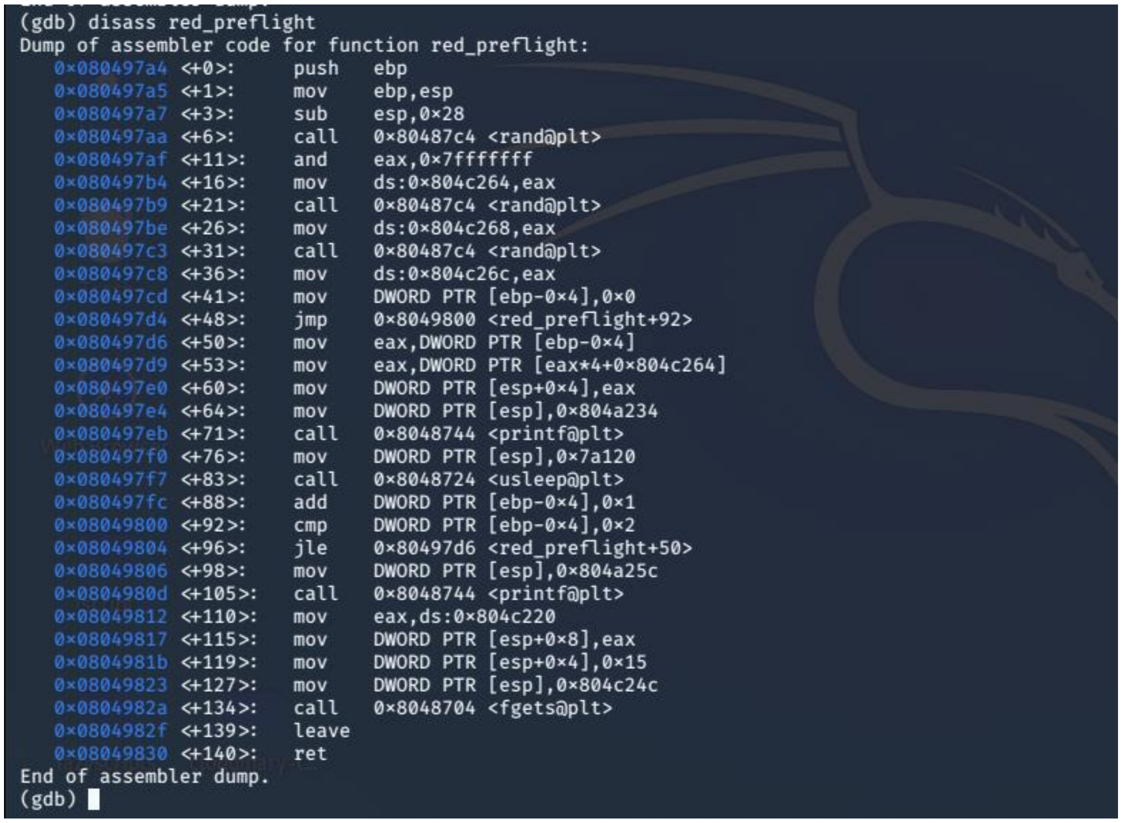
Assembly code for function red:



```
Dump of assembler code for function red:
   0x08049831 <+0>:     push   ebp
   0x08049832 <+1>:     mov    ebp,esp
   0x08049834 <+3>:     sub    esp,0x18
   0x08049837 <+6>:     call   0x80497a4 <red_preflight>
   0x0804983c <+11>:    mov    DWORD PTR [ebp-0x4],0x804a29c
   0x08049843 <+18>:    mov    DWORD PTR [ebp-0x8],0x0
   0x0804984a <+25>:    jmp    0x80498ba <red+137>
   0x0804984c <+27>:    mov    eax,DWORD PTR [ebp-0x8]
   0x0804984f <+30>:    movzx  edx,BYTE PTR [eax+0x804c24c]
   0x08049856 <+37>:    mov    eax,ds:0x804c26c
   0x0804985b <+42>:    and    eax,0x1f
   0x0804985e <+45>:    add    eax,DWORD PTR [ebp-0x4]
   0x08049861 <+48>:    movzx  eax,BYTE PTR [eax]
   0x08049864 <+51>:    cmp    dl,al
   0x08049866 <+53>:    je     0x8049877 <red+70>
   0x08049868 <+55>:    mov    eax,ds:0x804c128
   0x0804986d <+60>:    add    eax,0x1
   0x08049870 <+63>:    mov    ds:0x804c128,eax
   0x08049875 <+68>:    jmp    0x80498ca <red+153>
   0x08049877 <+70>:    mov    eax,ds:0x804c26c
   0x0804987c <+75>:    mov    edx,eax
   0x0804987e <+77>:    shr    edx,0x5
   0x08049881 <+80>:    mov    eax,ds:0x804c268
   0x08049886 <+85>:    shl    eax,0x1b
   0x08049889 <+88>:    or     eax,edx
   0x0804988b <+90>:    mov    ds:0x804c26c,eax
   0x08049890 <+95>:    mov    eax,ds:0x804c268
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049895 <+100>:   mov    edx,eax
   0x08049897 <+102>:   shr    edx,0x5
   0x0804989a <+105>:   mov    eax,ds:0x804c264
   0x0804989f <+110>:   shl    eax,0x1b
   0x080498a2 <+113>:   or     eax,edx
   0x080498a4 <+115>:   mov    ds:0x804c268,eax
   0x080498a9 <+120>:   mov    eax,ds:0x804c264
```

```
Dump of assembler code for function red:
   0x08049831 <+0>:     push   ebp
   0x08049832 <+1>:     mov    ebp,esp
   0x08049834 <+3>:     sub    esp,0x18
   0x08049837 <+6>:     call   0x80497a4 <red_preflight>
   0x0804983c <+11>:    mov    DWORD PTR [ebp-0x4],0x804a29c
   0x08049843 <+18>:    mov    DWORD PTR [ebp-0x8],0x0
   0x0804984a <+25>:    jmp    0x80498ba <red+137>
   0x0804984c <+27>:    mov    eax,DWORD PTR [ebp-0x8]
   0x0804984f <+30>:    movzx  edx,BYTE PTR [eax+0x804c24c]
   0x08049856 <+37>:    mov    eax,ds:0x804c26c
   0x0804985b <+42>:    and    eax,0x1f
   0x0804985e <+45>:    add    eax,DWORD PTR [ebp-0x4]
   0x08049861 <+48>:    movzx  eax,BYTE PTR [eax]
   0x08049864 <+51>:    cmp    dl,al
   0x08049866 <+53>:    je     0x8049877 <red+70>
   0x08049868 <+55>:    mov    eax,ds:0x804c128
   0x0804986d <+60>:    add    eax,0x1
   0x08049870 <+63>:    mov    ds:0x804c128,eax
   0x08049875 <+68>:    jmp    0x80498ca <red+153>
   0x08049877 <+70>:    mov    eax,ds:0x804c26c
   0x0804987c <+75>:    mov    edx,eax
   0x0804987e <+77>:    shr    edx,0x5
   0x08049881 <+80>:    mov    eax,ds:0x804c268
   0x08049886 <+85>:    shl    eax,0x1b
   0x08049889 <+88>:    or     eax,edx
   0x0804988b <+90>:    mov    ds:0x804c26c,eax
   0x08049890 <+95>:    mov    eax,ds:0x804c268
--Type <RET> for more, q to quit, c to continue without paging--
   0x08049895 <+100>:   mov    edx,eax
   0x08049897 <+102>:   shr    edx,0x5
   0x0804989a <+105>:   mov    eax,ds:0x804c264
   0x0804989f <+110>:   shl    eax,0x1b
   0x080498a2 <+113>:   or     eax,edx
   0x080498a4 <+115>:   mov    ds:0x804c268,eax
   0x080498a9 <+120>:   mov    eax,ds:0x804c264
   0x080498ae <+125>:   shr    eax,0x5
   0x080498b1 <+128>:   mov    ds:0x804c264,eax
   0x080498b6 <+133>:   add    DWORD PTR [ebp-0x8],0x1
   0x080498ba <+137>:   cmp    DWORD PTR [ebp-0x8],0x12
   0x080498be <+141>:   jle    0x804984c <red+27>
   0x080498c0 <+143>:   mov    DWORD PTR ds:0x804c128,0x0
   0x080498ca <+153>:   leave
   0x080498cb <+154>:   ret
End of assembler dump.
```

Assembly code for red_preflight function:



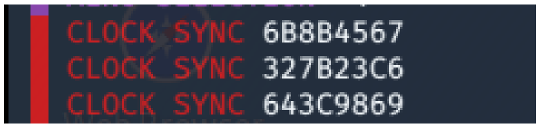
```
(gdb) disass red_preflight
Dump of assembler code for function red_preflight:
   0x080497a4 <+0>:     push   ebp
   0x080497a5 <+1>:     mov    ebp,esp
   0x080497a7 <+3>:     sub    esp,0x28
   0x080497aa <+6>:     call   0x80487c4 <rand@plt>
   0x080497af <+11>:    and    eax,0x7fffffff
   0x080497b4 <+16>:    mov    ds:0x804c264,eax
   0x080497b9 <+21>:    call   0x80487c4 <rand@plt>
   0x080497be <+26>:    mov    ds:0x804c268,eax
   0x080497c3 <+31>:    call   0x80487c4 <rand@plt>
   0x080497c8 <+36>:    mov    ds:0x804c26c,eax
   0x080497cd <+41>:    mov    DWORD PTR [ebp-0x4],0x0
   0x080497d4 <+48>:    jmp    0x8049800 <red_preflight+92>
   0x080497d6 <+50>:    mov    eax,DWORD PTR [ebp-0x4]
   0x080497d9 <+53>:    mov    eax,DWORD PTR [eax*4+0x804c264]
   0x080497e0 <+60>:    mov    DWORD PTR [esp+0x4],eax
   0x080497e4 <+64>:    mov    DWORD PTR [esp],0x804a234
   0x080497eb <+71>:    call   0x8048744 <printf@plt>
   0x080497f0 <+76>:    mov    DWORD PTR [esp],0x7a120
   0x080497f7 <+83>:    call   0x8048724 <usleep@plt>
   0x080497fc <+88>:    add    DWORD PTR [ebp-0x4],0x1
   0x08049800 <+92>:    cmp    DWORD PTR [ebp-0x4],0x2
   0x08049804 <+96>:    jle    0x80497d6 <red_preflight+50>
   0x08049806 <+98>:    mov    DWORD PTR [esp],0x804a25c
   0x0804980d <+105>:   call   0x8048744 <printf@plt>
   0x08049812 <+110>:   mov    eax,ds:0x804c220
   0x08049817 <+115>:   mov    DWORD PTR [esp+0x8],eax
   0x0804981b <+119>:   mov    DWORD PTR [esp+0x4],0x15
   0x08049823 <+127>:   mov    DWORD PTR [esp],0x804c24c
   0x0804982a <+134>:   call   0x8048704 <fgets@plt>
   0x0804982f <+139>:   leave
   0x08049830 <+140>:   ret
End of assembler dump.
(gdb)
```

```
Dump of assembler code for function red_preflight:
   0x080497a4 <+0>:     push   ebp
   0x080497a5 <+1>:     mov    ebp,esp
   0x080497a7 <+3>:     sub    esp,0x28
```

```
0x080497aa <+6>:     call   0x80487c4 <rand@plt>
0x080497af <+11>:    and    eax,0x7fffffff
0x080497b4 <+16>:    mov    ds:0x804c264,eax
0x080497b9 <+21>:    call   0x80487c4 <rand@plt>
0x080497be <+26>:    mov    ds:0x804c268,eax
0x080497c3 <+31>:    call   0x80487c4 <rand@plt>
0x080497c8 <+36>:    mov    ds:0x804c26c,eax
0x080497cd <+41>:    mov    DWORD PTR [ebp-0x4],0x0
0x080497d4 <+48>:    jmp    0x8049800 <red_preflight+92>
0x080497d6 <+50>:    mov    eax,DWORD PTR [ebp-0x4]
0x080497d9 <+53>:    mov    eax,DWORD PTR [eax*4+0x804c264]
0x080497e0 <+60>:    mov    DWORD PTR [esp+0x4],eax
0x080497e4 <+64>:    mov    DWORD PTR [esp],0x804a234
0x080497eb <+71>:    call   0x8048744 <printf@plt>
0x080497f0 <+76>:    mov    DWORD PTR [esp],0x7a120
0x080497f7 <+83>:    call   0x8048724 <usleep@plt>
0x080497fc <+88>:    add    DWORD PTR [ebp-0x4],0x1
0x08049800 <+92>:    cmp    DWORD PTR [ebp-0x4],0x2
0x08049804 <+96>:    jle    0x80497d6 <red_preflight+50>
0x08049806 <+98>:    mov    DWORD PTR [esp],0x804a25c
0x0804980d <+105>:   call   0x8048744 <printf@plt>
0x08049812 <+110>:   mov    eax,ds:0x804c220
0x08049817 <+115>:   mov    DWORD PTR [esp+0x8],eax
0x0804981b <+119>:   mov    DWORD PTR [esp+0x4],0x15
0x08049823 <+127>:   mov    DWORD PTR [esp],0x804c24c
0x0804982a <+134>:   call   0x8048704 <fgets@plt>
0x0804982f <+139>:   leave
0x08049830 <+140>:   ret
```

End of assembler dump.

Here before prompting us and loading our input string into buffer (lines 26-32), we initialize array r with three "random" numbers using rand() (lines 4-10) and display each number to the user (lines 11-24):



```
CLOCK SYNC 6B8B4567
CLOCK SYNC 327B23C6
CLOCK SYNC 643C9869
```

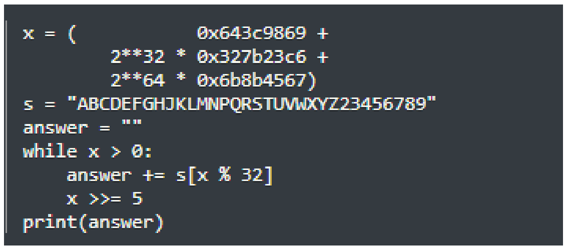
A time-based seed for rand() was never set.rand() is used with srand(time(0)) so that rand() is seeded with a different number each time and the results of the pseudorandom number generation will be different. Since our rand() is never seeded, it will generate the same random numbers in order every time: 0x6b8b4567, 0x327b23c6, and 0x643c9869.

In function red;

On line 6, the string "ABCDEFGHJKLMNPQRSTUVWXYZ23456789" is loaded into var1, which is exactly 32 characters long. Afterwards, var2 iterates over our input string, and some operations are performed on our "random" numbers.

On line 12, r[2] is anded with 0x0000001f. The character at corresponding index within var1 is then accessed and compared with the next character of our input string.

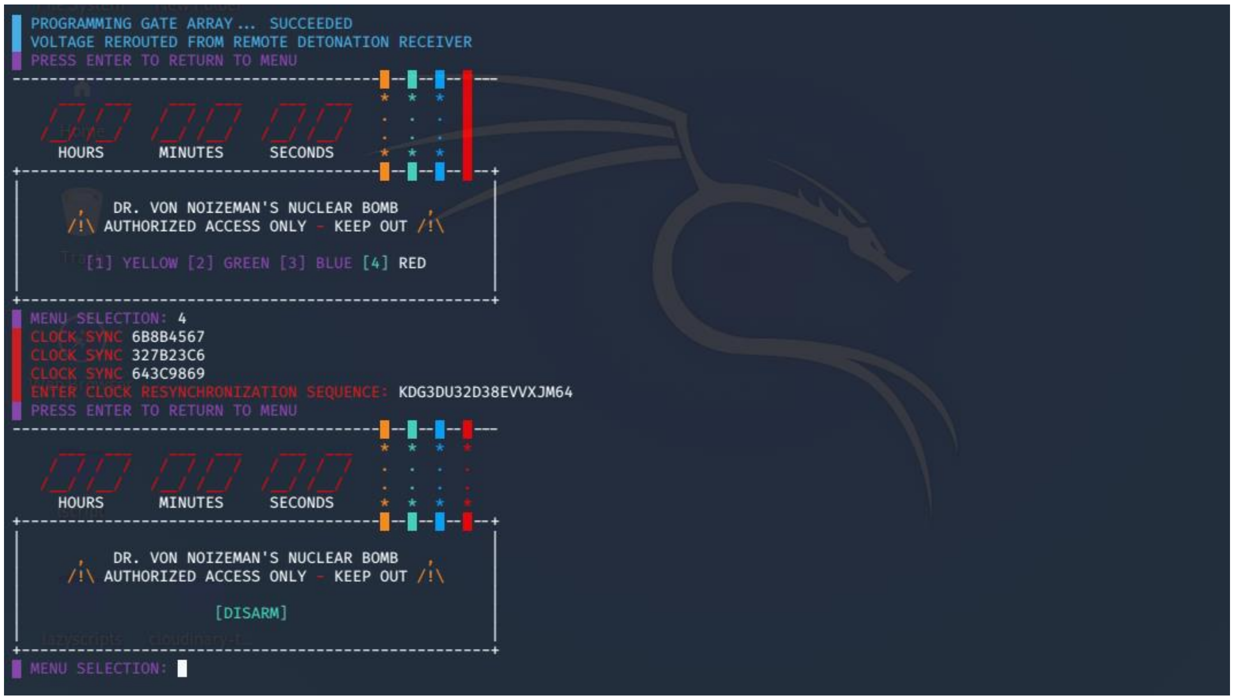
To move onto the next character, var2 is incremented and all bits in r[0] through r[2] are shifted over by 5 bits. In order to decide the proper input string, let's use a Python script



```python
x = (           0x643c9869 +
        2**32 * 0x327b23c6 +
        2**64 * 0x6b8b4567)
s = "ABCDEFGHJKLMNPQRSTUVWXYZ23456789"
answer = ""
while x > 0:
    answer += s[x % 32]
    x >>= 5
print(answer)
```

By running this script, we can get the answer as: "KDG3DU32D38EVVXJM64”

Passphrase for red: KDG3DU32D38EVVXJM64



```
PROGRAMMING GATE ARRAY... SUCCEEDED
VOLTAGE REROUTED FROM REMOTE DETONATION RECEIVER
PRESS ENTER TO RETURN TO MENU

    HOURS      MINUTES     SECONDS

      DR. VON NOIZEMAN'S NUCLEAR BOMB
 /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

   [1] YELLOW [2] GREEN [3] BLUE [4] RED

MENU SELECTION: 4
CLOCK SYNC 6B8B4567
CLOCK SYNC 327B23C6
CLOCK SYNC 643C9869
ENTER CLOCK RESYNCHRONIZATION SEQUENCE: KDG3DU32D38EVVXJM64
PRESS ENTER TO RETURN TO MENU

    HOURS      MINUTES     SECONDS

      DR. VON NOIZEMAN'S NUCLEAR BOMB
 /!\ AUTHORIZED ACCESS ONLY - KEEP OUT /!\

                [DISARM]

MENU SELECTION:
```