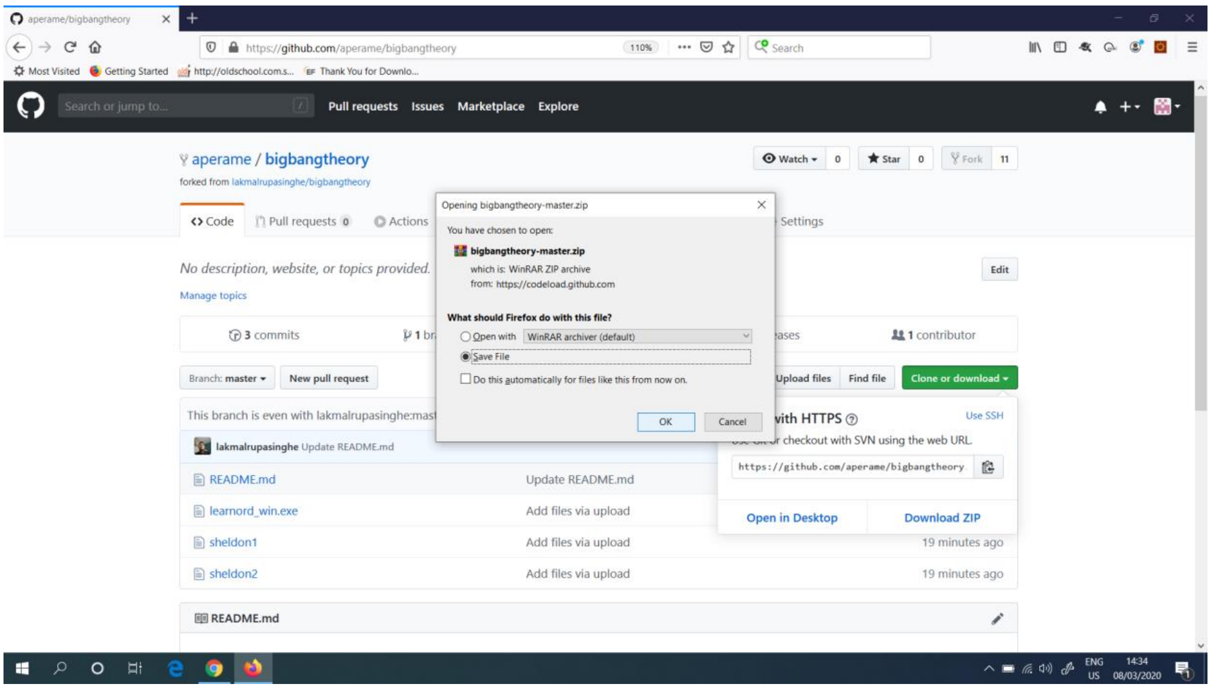
IT17157674 – V.Aperame

OHTS – Big bang Theory

First, I opened the "Github" account and forked the account, which is shown below (Figure 1).



aperame / bigbangtheory
forked from lakmalrupasinghe/bigbangtheory

Opening bigbangtheory-master.zip

You have chosen to open:
bigbangtheory-master.zip
which is: WinRAR ZIP archive
from: https://codeload.github.com

What should Firefox do with this file?
Open with WinRAR archiver (default)
[✓ Save File]
Do this automatically for files like this from now on.

OK Cancel

No description, website, or topics provided.

This branch is even with lakmalrupasinghe:master

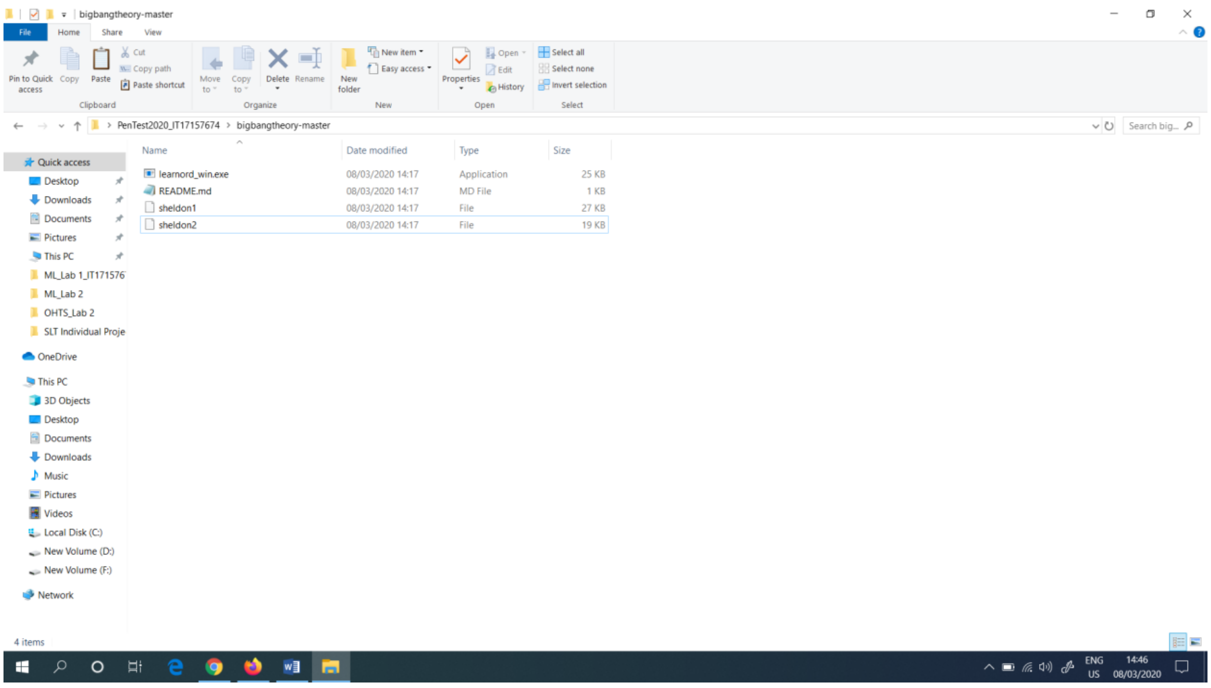
lakmalrupasinghe Update README.md

README.md	Update README.md	
learnord_win.exe	Add files via upload	
sheldon1	Add files via upload	19 minutes ago
sheldon2	Add files via upload	19 minutes ago

Clone with HTTPS
Use Git or checkout with SVN using the web URL.
https://github.com/aperame/bigbangtheory

Figure 1:Fork and get the account

Next, I unzipped the downloaded file, and saw the files (Figure 2).



PenTest2020_IT17157674 > bigbangtheory-master

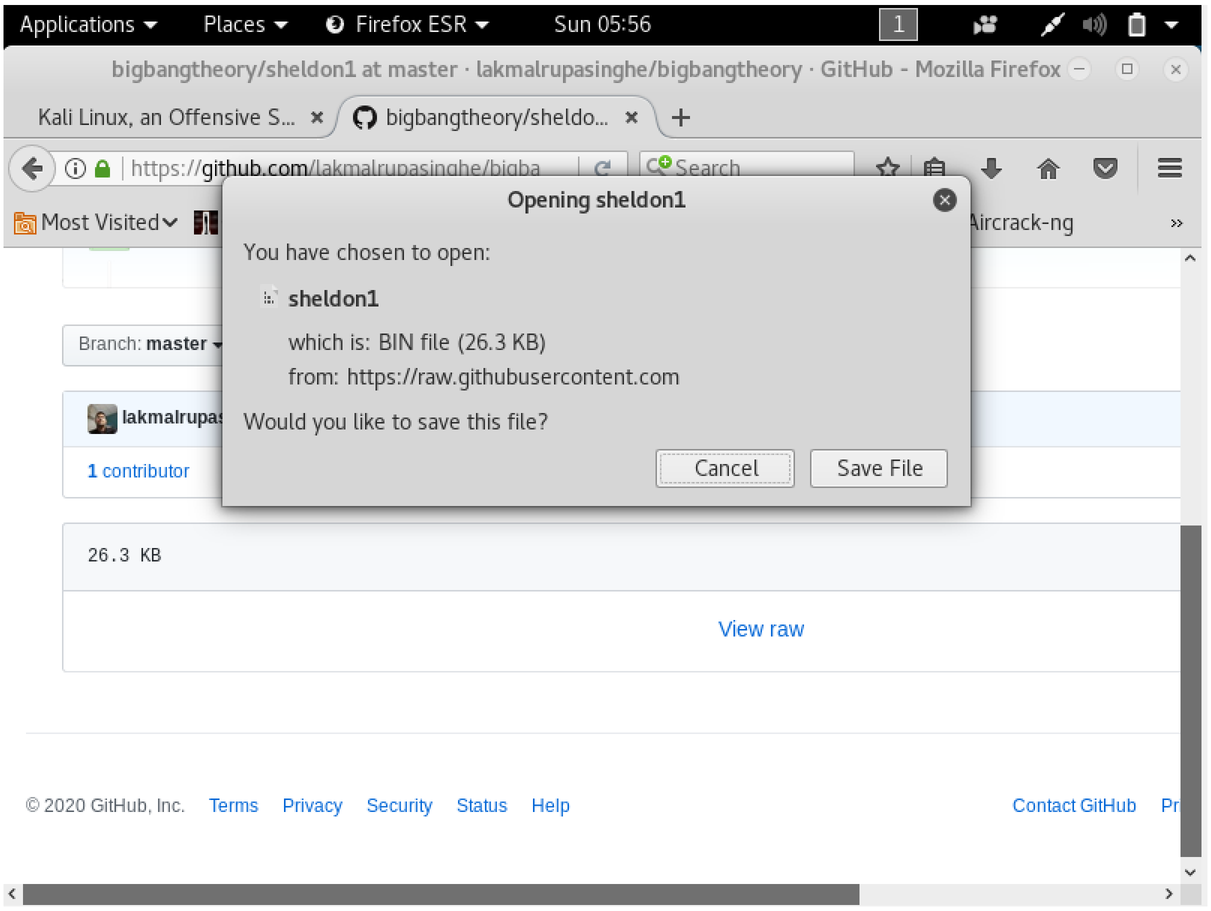
Name	Date modified	Type	Size
learnord_win.exe	08/03/2020 14:17	Application	25 KB
README.md	08/03/2020 14:17	MD File	1 KB
sheldon1	08/03/2020 14:17	File	27 KB
sheldon2	08/03/2020 14:17	File	19 KB

4 items

Figure 2: Unzipped and viewed the files

Next, I opened the Kali Linux and downloaded the file from the GitHub and saved it into the Downloads file. This is shown in (Figure 3).

In the same manner I have also downloaded the sheldon2 and learnord_win exe file.



Opening sheldon1

You have chosen to open:

sheldon1

which is: BIN file (26.3 KB)

from: https://raw.githubusercontent.com

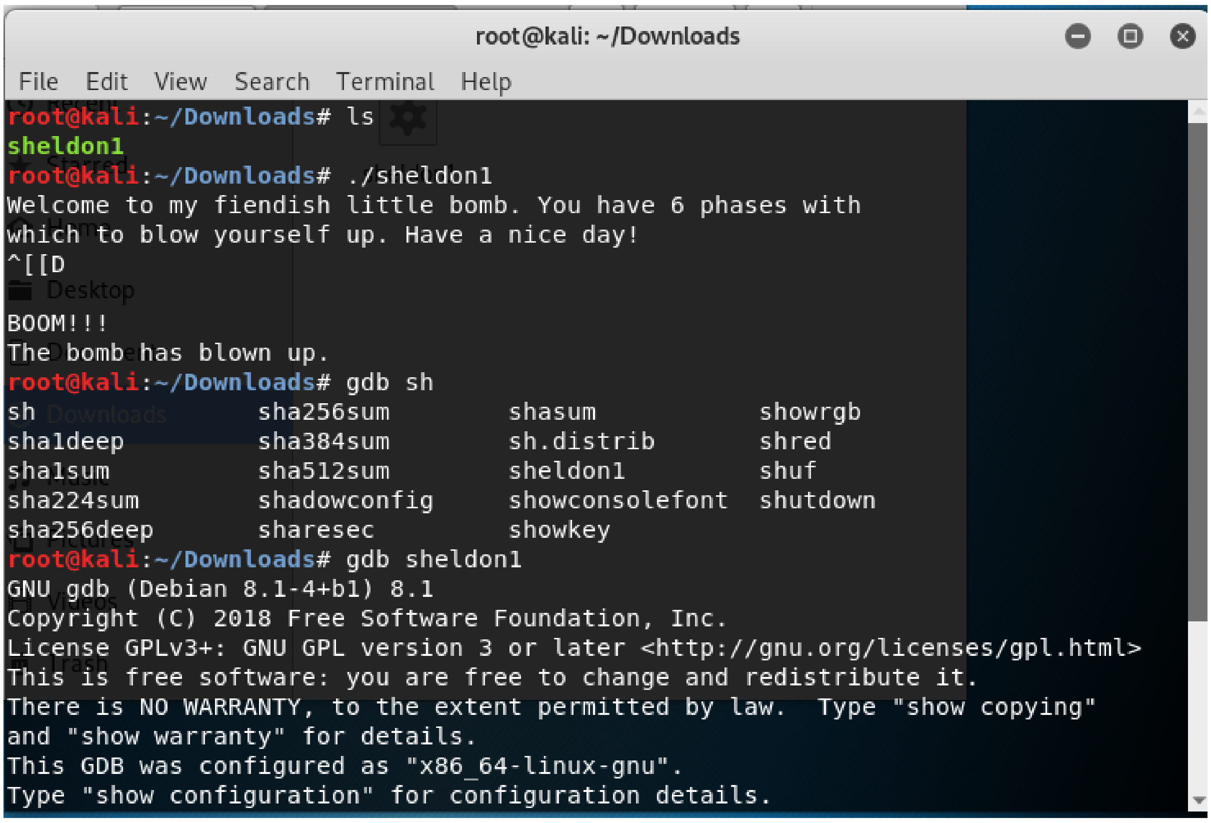
Would you like to save this file?

Cancel | Save File

Figure 3: Download and save the files

Next, I opened the terminal and typed as “ls” and other commands which is shown in (Figure 4).

Then I typed as “./sheldon 1” in order to see whether the files are running or not.



```
root@kali:~/Downloads# ls
sheldon1
root@kali:~/Downloads# ./sheldon1
Welcome to my fiendish little bomb. You have 6 phases with
which to blow yourself up. Have a nice day!
^[[D

BOOM!!!
The bomb has blown up.
root@kali:~/Downloads# gdb sh
sh                sha256sum         shasum            showrgb
sha1deep          sha384sum         sh.distrib        shred
sha1sum           sha512sum         sheldon1          shuf
sha224sum         shadowconfig      showconsolefont   shutdown
sha256deep        sharesec          showkey
root@kali:~/Downloads# gdb sheldon1
GNU gdb (Debian 8.1-4+b1) 8.1
Copyright (C) 2018 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.
```

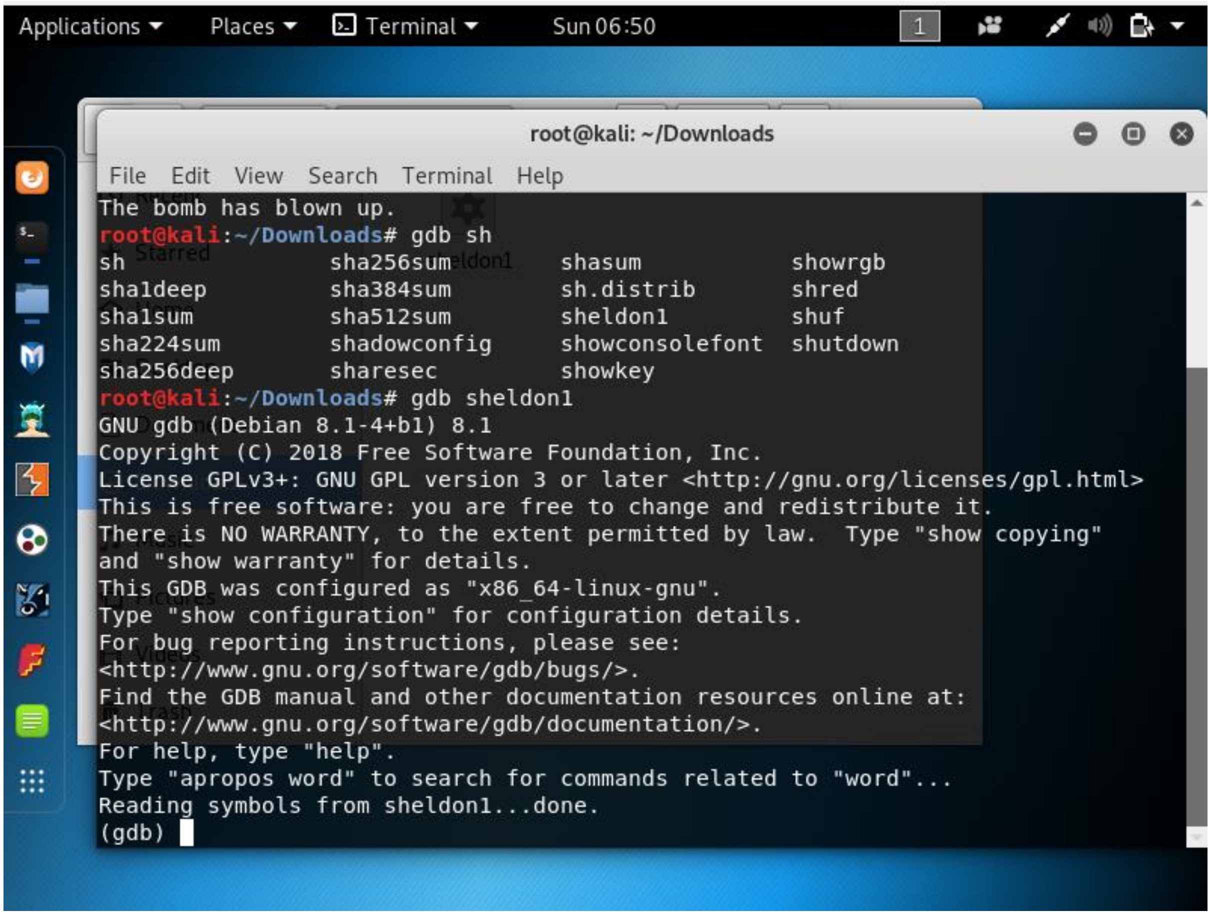
Figure 4: Typed the "ls" commands and other commands

Then I typed as ". /sheldon 1" command, "gdb sh" command and "gdb sheldon1" command as shown above.

Then I typed as "gdb sh" in order to analyse the Assembly codes.

Next, I typed as "gdb sheldon1" and analysed the sheldon1 file. Then the bomb will blow up as a result.

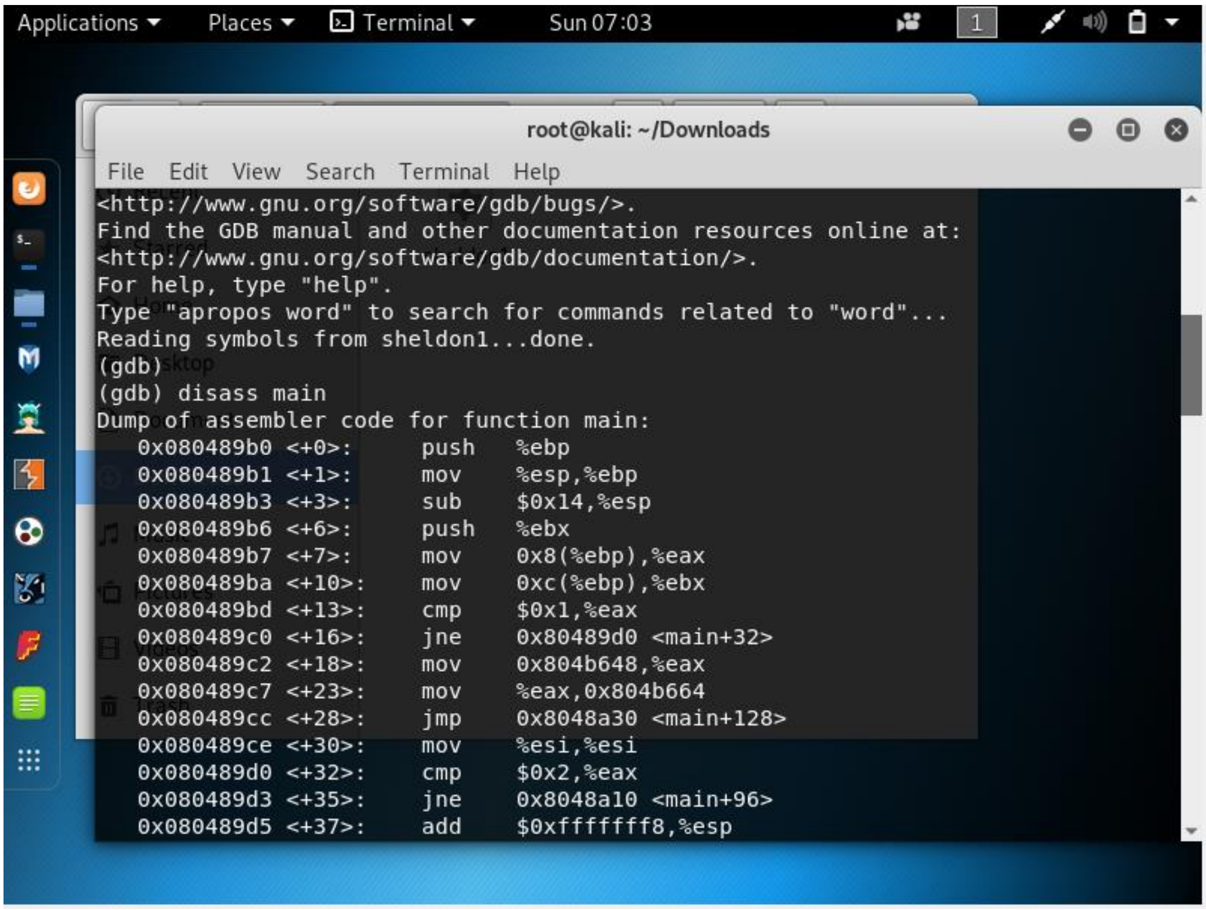
Next, I analysed the commands as shown below in Figure 5.



```
The bomb has blown up.
root@kali:~/Downloads# gdb sh
sh                sha256sum         shasum            showrgb
sha1deep          sha384sum         sh.distrib        shred
sha1sum           sha512sum         sheldon1          shuf
sha224sum         shadowconfig      showconsolefont   shutdown
sha256deep        sharesec          showkey
root@kali:~/Downloads# gdb sheldon1
GNU gdb (Debian 8.1-4+b1) 8.1
Copyright (C) 2018 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 sheldon1...done.
(gdb) 
```

Figure 5: Analysed the commands

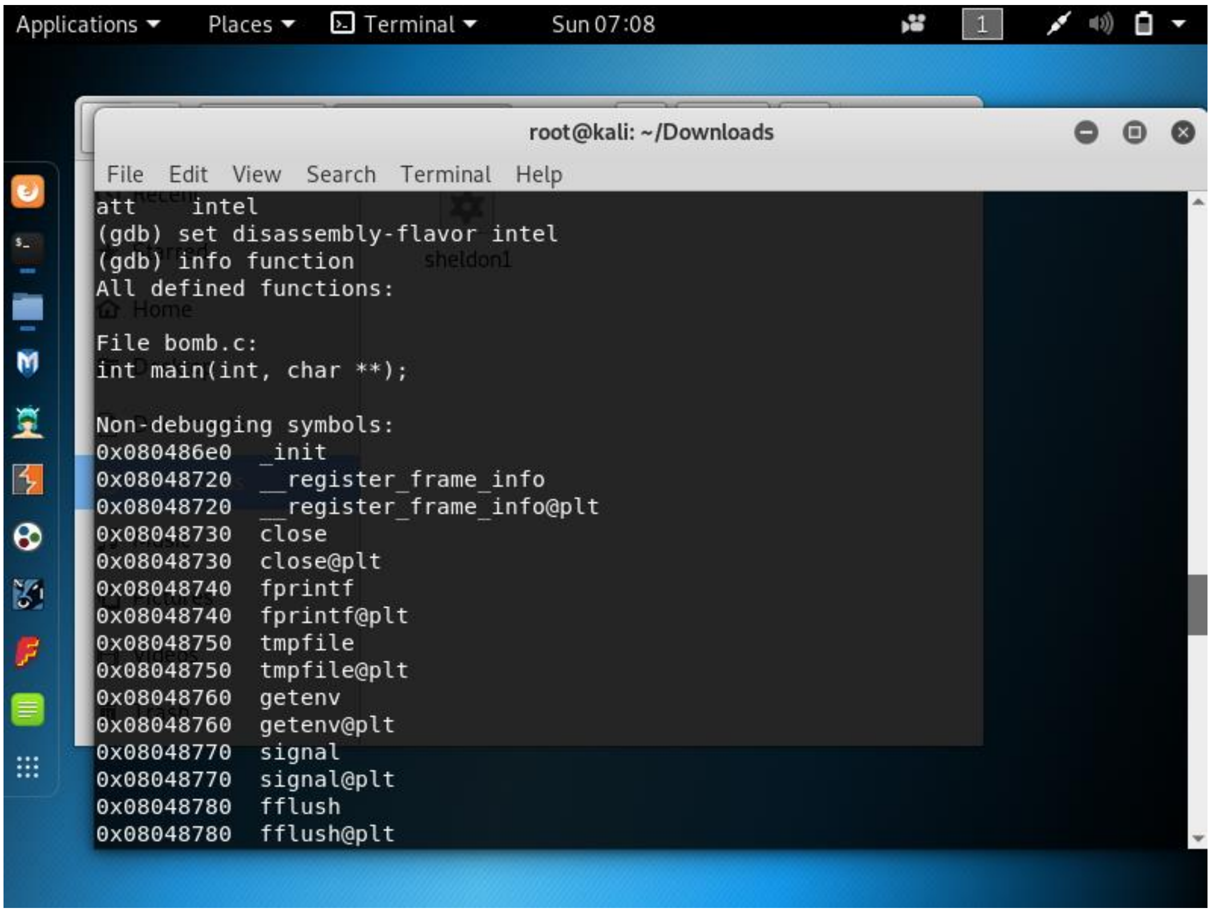
Next, I typed as “disass main” and analysed it. This is shown in (Figure 6).



```
<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 sheldon1...done.
(gdb)
(gdb) disass main
Dump of assembler code for function main:
   0x080489b0 <+0>:     push   %ebp
   0x080489b1 <+1>:     mov    %esp,%ebp
   0x080489b3 <+3>:     sub    $0x14,%esp
   0x080489b6 <+6>:     push   %ebx
   0x080489b7 <+7>:     mov    0x8(%ebp),%eax
   0x080489ba <+10>:    mov    0xc(%ebp),%ebx
   0x080489bd <+13>:    cmp    $0x1,%eax
   0x080489c0 <+16>:    jne    0x80489d0 <main+32>
   0x080489c2 <+18>:    mov    0x804b648,%eax
   0x080489c7 <+23>:    mov    %eax,0x804b664
   0x080489cc <+28>:    jmp    0x8048a30 <main+128>
   0x080489ce <+30>:    mov    %esi,%esi
   0x080489d0 <+32>:    cmp    $0x2,%eax
   0x080489d3 <+35>:    jne    0x8048a10 <main+96>
   0x080489d5 <+37>:    add    $0xfffffff8,%esp
```

Figure 6: Analysed the "disass main" command

Then, I typed as "disassembly-flavor intel" command and "into function" which is shown in (Figure 7).



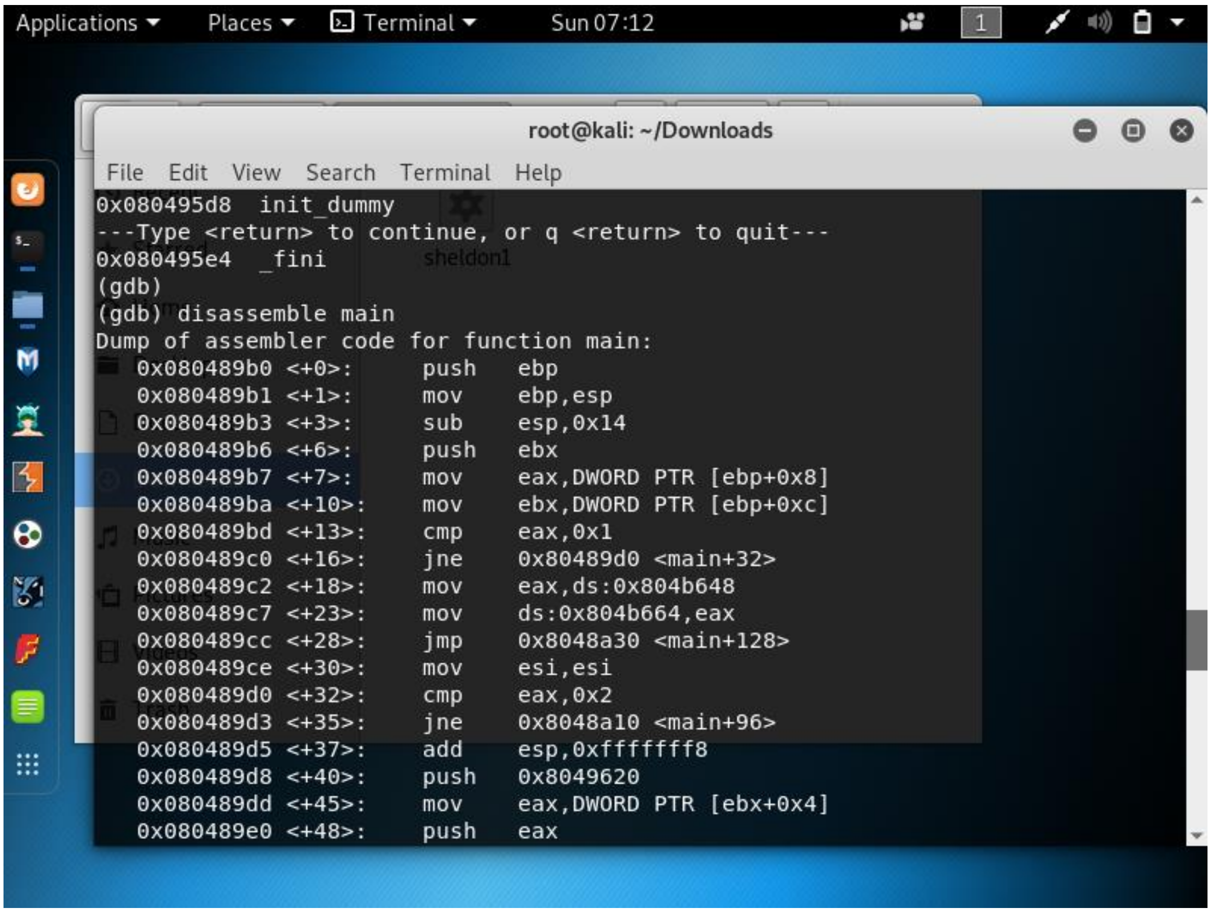
```
att    intel
(gdb) set disassembly-flavor intel
(gdb) info function
All defined functions:

File bomb.c:
int main(int, char **);

Non-debugging symbols:
0x080486e0  _init
0x08048720  __register_frame_info
0x08048720  __register_frame_info@plt
0x08048730  close
0x08048730  close@plt
0x08048740  fprintf
0x08048740  fprintf@plt
0x08048750  tmpfile
0x08048750  tmpfile@plt
0x08048760  getenv
0x08048760  getenv@plt
0x08048770  signal
0x08048770  signal@plt
0x08048780  fflush
0x08048780  fflush@plt
```

Figure 7: Typed the above mentioned commands

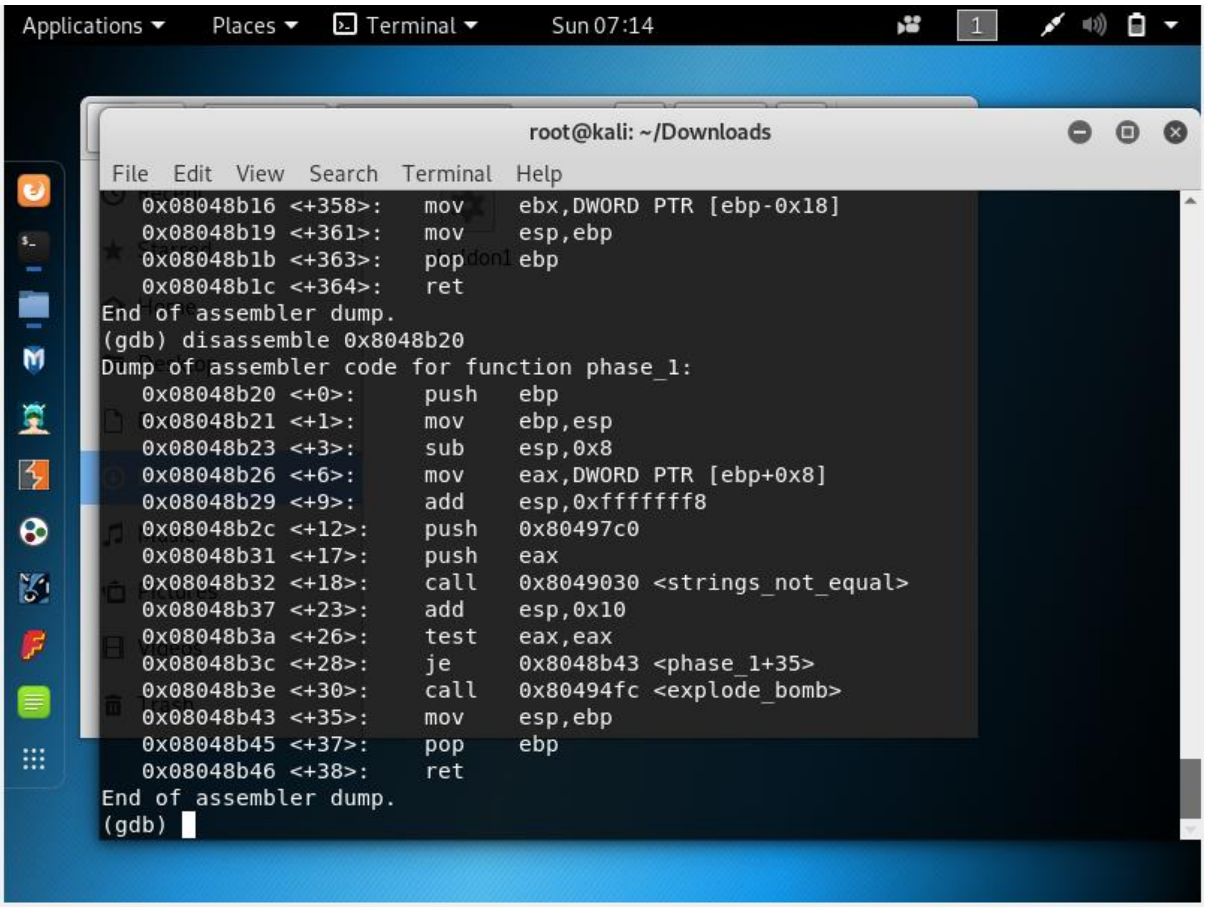
Next, I typed as “disassemble main” to analyse the main function (Figure 8).



```
0x080495d8  init_dummy
---Type <return> to continue, or q <return> to quit---
0x080495e4  _fini
(gdb)
(gdb) disassemble main
Dump of assembler code for function main:
   0x080489b0 <+0>:     push   ebp
   0x080489b1 <+1>:     mov    ebp,esp
   0x080489b3 <+3>:     sub    esp,0x14
   0x080489b6 <+6>:     push   ebx
   0x080489b7 <+7>:     mov    eax,DWORD PTR [ebp+0x8]
   0x080489ba <+10>:    mov    ebx,DWORD PTR [ebp+0xc]
   0x080489bd <+13>:    cmp    eax,0x1
   0x080489c0 <+16>:    jne    0x80489d0 <main+32>
   0x080489c2 <+18>:    mov    eax,ds:0x804b648
   0x080489c7 <+23>:    mov    ds:0x804b664,eax
   0x080489cc <+28>:    jmp    0x8048a30 <main+128>
   0x080489ce <+30>:    mov    esi,esi
   0x080489d0 <+32>:    cmp    eax,0x2
   0x080489d3 <+35>:    jne    0x8048a10 <main+96>
   0x080489d5 <+37>:    add    esp,0xfffffff8
   0x080489d8 <+40>:    push   0x8049620
   0x080489dd <+45>:    mov    eax,DWORD PTR [ebx+0x4]
   0x080489e0 <+48>:    push   eax
```

Figure 8: Typed the "disassemble main" commands

Then, I typed as shown in (Figure 9).



```
   0x08048b16 <+358>:   mov    ebx,DWORD PTR [ebp-0x18]
   0x08048b19 <+361>:   mov    esp,ebp
   0x08048b1b <+363>:   pop    ebp
   0x08048b1c <+364>:   ret
End of assembler dump.
(gdb) disassemble 0x8048b20
Dump of assembler code for function phase_1:
   0x08048b20 <+0>:     push   ebp
   0x08048b21 <+1>:     mov    ebp,esp
   0x08048b23 <+3>:     sub    esp,0x8
   0x08048b26 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x08048b29 <+9>:     add    esp,0xfffffff8
   0x08048b2c <+12>:    push   0x80497c0
   0x08048b31 <+17>:    push   eax
   0x08048b32 <+18>:    call   0x8049030 <strings_not_equal>
   0x08048b37 <+23>:    add    esp,0x10
   0x08048b3a <+26>:    test   eax,eax
   0x08048b3c <+28>:    je     0x8048b43 <phase_1+35>
   0x08048b3e <+30>:    call   0x80494fc <explode_bomb>
   0x08048b43 <+35>:    mov    esp,ebp
   0x08048b45 <+37>:    pop    ebp
   0x08048b46 <+38>:    ret
End of assembler dump.
(gdb) 
```

Figure 9: Typed the "disassemble 0x8048b20" commands

Then, in the same way I analysed for all the registers.