WiCyS CTF 2023 AES+ Writeup

Manav Malik

We begin, as always, by examining the challenge description:

> I wrote a function that uses AES to encrypt information, but I have a feeling you can break it and get the flag.

The first thing we should do here is take a look at the source code. After examining `function`, we see that the program simply takes the data inputted into the function and appends the flag to it before encrypting the result using AES.

Let's start by thinking about what would happen if we gave the function 15 bytes of data. Because the block size here is 16 bytes, the first block would contain the entirety of the input plus the first byte of the flag. Furthermore, we know that AES is deterministic so any given plaintext will always map to the same ciphertext. Now that we know what some random 15 bytes plus the first byte of the flag is, we can brute force possible 16th bytes until we find the byte that yields the same first block of the ciphertext.

We will continue this process by decrementing the number of bytes in the input each time to get the flag. Note that instead of using 15 bytes of data initially, we must use a number that is 15 (mod 16) because the length of the appended bytes is almost certainly longer than 16 bytes.

Here's an example of code we can use to determine the flag. All that's left to do from here is adapt the code to input it into the hosted program.

```
def solve():
    b_size = 16
    flag_len = 144 # random large multiple of 16
    flag = b''
    for byte_i in range(flag_len):
        test_str = b'x' * (flag_len - byte_i - 1)
        lookup_dict = {
            function(test_str + flag + bytes([i]))[128:144]: i
            for i in range(256)
        }
        actual = function(test_str)
        block_i = flag_len - byte_i - 1
        block_i += (-block_i) % b_size

        try:
            flag += bytes([lookup_dict[actual[128:144]]])
        except KeyError: # no more flag to lookup
            return flag
```

Running this yields the flag: WCS{sn34kY_Ae5}