BOSCH CodeRace 2023

- Round 1 -

Team: Ternary Stars

Members:

1. Nguyen Ho Tien Dat
2. Nguyen Sy Thanh
3. Pham Huynh Quoc Thanh

Challenge 1

We use the IDA tool for Reverse Engineering in this challenge. We check the function **check_password** and get the rule that the password is built by getting the digits from position 85 to position 85+11 from a random digits sequence.

––> The flag is: **301349920110**

Challenge 2

As the AES-ECB algorithm divides the plaintext into 128-byte blocks (16 characters) and encrypts each block independently, we can follow these step to get the result:

First, we can run get-flag.exe with username = "aaaa", then the message will be:

"Hello aaaa, the |**flag is letter X**| (uppercase)."

(the symbol | here just for separating the blocks)

We get the output:

05c83517e6dbf57e6b1dc855eb411795|**bdae8f4d417e9056d85500686c712402**|72faa70835b1be7f9977589c7aa84532

Hence the block "flag is letter X" is encrypted to:

bdae8f4d417e9056d85500686c712402

Second, we run get-flag.exe with username = "aaaaaaaaaaflag is letter *i*" with *i* in turn to be each of the letters from A to Z (i.e. "aaaaaaaaaaflag is letter A", "aaaaaaaaaaflag is letter B”, etc.). The message now will be:

“Hello aaaaaaaaaa|**flag is the letter *i***|, the flag is le|tter X (uppercas|e).”

Extract the hexadecimal characters from position 33 to position 64 (the first character of the output is at position 1, counting from left to right), which is also the second block, and compare to the encrypted block above, we will get the solution when these hexadecimal characters are exactly the same.

——> The flag is: **P**

Challenge 3

The given code is vulnerable to a buffer overflow. By providing a command-line argument longer than the size of the ‘name’ buffer, we can overwrite the ‘magic_word’ variable and change its value to ‘**0x69696969**’, which triggers the printing of the flag. To solve this challenge we can run get-flag.exe with input: **AAAAAAAAAiiii**

——> The flag is: **M4G1CW0RD**

Challenge 4

In recorded_encryptions.json, we can see a pair of plaintext-ciphertext generated using the same IV as the IV for the encrypted flag:

"plaintext": "**001333b95c1edb3ef145a4a9f52f7b9a**",
"ciphertext": "**b28b7e3f49355b7a236ad2745d69cb25**",
"iv": "**70359350f329603f0da99a1f9151d844**"

Since AES-CTR is a stream cipher, encryption and decryption can be conducted using XOR operation of the plaintext/ciphertext and a key stream. With the same secret key and IV, the same key stream is also generated. Then we can XOR the above plaintext and ciphertext to get the key stream:

b2984d86152b8044d22f76dda846b0bf

Finally, we use this key stream to XOR with the encrypted flag to get the decrypted flag and convert to string type as the solution.

Encrypted flag: **fdfa29ef6547ef37a64a1bb2c629c5cc**

Decrypted flag: **4f626469706c6f7374656d6f6e6f7573**

——> The flag is: **Obdiplostemonous**

Challenge 5

SHA-1 digest of the flag is: **0x071deccefe90336bc00e3fca11263ac97287b980**. There is a way to decrypt a SHA-1 hash, using a dictionary populated with strings and their SHA-1 counterpart.

——> The flag is: **weakpassword**

Challenge 6

The most popular substitution cipher is the Caesar cipher. In this challenge, we need to shift 9 characters (Ex: a -> j) to receive the answer. Our ciphertext is: **kxblqpuxkjubxocfjancnlqwxuxprnb**.

——> The flag is: **boschglobalsoftwaretechnologies**