Writeup for c2

August 9, 2025

1 Understanding the problem

For this challenge, I had been given some encrypting pattern in encrypt.py which was initially obtaining a random integer from 0-127 then performing haslib.sha256 algorithm on it and recursively appending the first character modulo 128 of this as a byte to the byte string key and performing it until its length equals that of plaintext. This acts as our key generation algorithm in this case. Now, for encrypting we simply add the integer for each position of bytes of key to the same for plaintext take modulo 128 and convert it into a byte string again finally giving us the ciphertext.

2 Issue with the encryption

The key generation algorithm in this case is not completely random and has only 128 possibilities as once you take the initial integer and perform the hashlib.sha26 the way it has been done in key generation you will always get the same key. It can be easily broken by checking for all values of this integer.

3 Solution

As we performed modular addition while encrypting, in decrypting process we need to subtract the value of the integer at that position of the key from value for ciphertext and add 128 and take modulo 128 to take care the value obtained lies in required range. Now, we can check for all keys with staring integer looping from 0 to 127 and run subtraction on them and print the plaintexts. All that remains is to search for a human readable kind of flag among all the printed 128 texts which can easily be performed.

4 Conclusion

The major issue in this encryption was that the key generation algorithm wasn't random but had only 128 possibilities which can easily be checked over.