The Keeper of Secrets - Writeup

Challenge Name: The Keeper of Secrets

Category: Cryptography / Reverse Engineering

CTF Event: SeaTF 2025

1. Challenge Description:

A high-security authentication system stores its access flag using a simple, yet effective encryption method. A leaked transmission has revealed the session key and the encrypted flag, but without the right approach, it's just a jumble of bytes.

Your task is to reverse-engineer the encryption scheme and retrieve the original flag.

Hints:

- Understanding byte-level manipulations will be essential for decryption.
- The final flag follows the format SeaTF{...}.

Given Data:

- Encrypted Username Hint: VHJ1c3QgaXMgZ2l2ZW4sIGJ1dCBuZXZlciByZXR1cm5lZC4gT25jZSBpbnNpZGUsIGFsbCBkb29ycyBhcmUgb3Blbi4=
- challenge2_bin (executable file)

2. Files Provided:

challenge2_bin (executable binary file)

3. Approach & Solution:

Step 1: Decoding the Base64 Hint

When dealing with encoded data, it's common practice to check if it's in a recognizable encoding format. Base64 is one of the most frequently used

encoding schemes, often used in data transmission and cryptographic applications.
To test whether the given string is Base64 encoded, we can attempt decoding it. If successful, it should return a meaningful text rather than gibberish. We use the base64 command in Linux to decode the hint:

```
echo
"VHJ1c3QgaXMgZ2l2ZW4sIGJ1dCBuZXZlciByZXR1cm5lZC4gT25jZSBpbnNpZGUs
IGFsbCBkb29ycyBhcmUgb3Blbi4=" | base64 -d
```

The decoded hint reveals the message:

```
Trust is given, but never returned. Once inside, all doors are
open.
```

To deduce the correct username, we analyze the hint:
The phrase suggests a system where trust is granted once but not taken back, and once authenticated, access is granted universally. This aligns with Kerberos, a network authentication protocol that uses tickets to grant access without needing to re-enter credentials. Based on this reasoning, we infer that the username required is Kerberos.

Step 2: Running the Binary

Now that we have the correct username, we proceed with running the binary file. First, we grant execution permissions and execute challenge2_bin.

```
chmod +x challenge2_bin
```

```
./challenge2_bin
```

Entering "Kerberos" as the username provides a session key and an encrypted flag. The session key and encrypted flag change each time the binary is executed, meaning we need to dynamically decrypt the flag using the given session key.

```
Enter username: Kerberos
```

```
Your session key: session_key
```

```
Encrypted Flag: encryped_flag
```

Step 3: Reverse Bit Rotation & XOR Decryption

Understanding the Encryption Scheme

From analyzing the encryption process, we deduce that:

- Each byte of the flag has been **bit-rotated**.
- After rotation, the bytes have been **XOR**ed with a **session key** (which is incremented in a certain pattern).

To recover the original flag, we must:

1. Reverse the bit rotation to get the intermediate byte values.

2. Perform XOR decryption using the extracted session key.

Python Script for Decryption:

```
encrypted_hex = "encrypted_flag"
encrypted_bytes = bytes.fromhex(encrypted_hex)

def reverse_bit_rotation(byte):
      return ((byte >> 2) | (byte << 6)) & 0xFF

deobfuscated_bytes = bytes(reverse_bit_rotation(b) for b in
encrypted_bytes)

key = session_key
flag = "".join(chr(deobfuscated_bytes[i] ^ (key + i % 5)) for i
in range(len(deobfuscated_bytes)))
print("Decrypted Flag:", flag)
```

Step 4: Obtaining the Flag

Executing the decryption script with the extracted session key successfully reveals the flag:

```
python3 decrypt_script.py

Decrypted Flag: SeaTF{Kerberos_Ticket}
```

The final output correctly reveals the decrypted **flag: SeaTF{Kerberos_Ticket}.**

4. Flag Extraction

Final Flag: SeaTF{Kerberos_Ticket}

5. Learning Outcomes:

- When working with encoded data, test common encoding schemes (Base64, Hex, etc.).
- Understanding bitwise operations is essential for reversing encryption schemes.

6. References & Tools Used:

- base64 (Linux command for encoding/decoding)
- Python bitwise operations
- Binary analysis tools

*****