CTF Write-up: Crypto Vault

Challenge Type: Web Exploitation (SQL Injection) + Cryptography (XOR Cipher) **Target:** `http://75.101.155.130/` **Flag Format:** `SECE{...}`

1. Understanding the Challenge

The "Crypto Vault" challenge was divided into two main objectives:

1. **Web Exploitation:** Gaining unauthorized access to the web application via SQL Injection.
2. **Cryptography:** Decrypting a set of encrypted messages to reconstruct the final flag.

The application, named CryptoVault, featured a login page that was immediately identified as a classic target for SQL Injection.

2. Bypassing Login Using SQL Injection

To bypass the authentication mechanism, a basic SQL Injection payload was employed in the username field.

Payload Used in Username Field: ```sql ' OR 1=1 – ```

The password field was populated with an arbitrary value.

Mechanism of the Bypass

This payload is designed to manipulate the backend SQL query, which is typically structured as:

`SELECT * FROM users WHERE username='[USER_INPUT]' AND password='[PASS_INPUT]'`

By inserting the payload, the query is transformed into a logically true statement:

```sql SELECT * FROM users WHERE username=" OR 1=1 – ' AND password='xyz' ```

- The single quote (`'`) closes the opening quote of the username string.
- The `OR 1=1` condition ensures the `WHERE` clause always evaluates to true.
- The double dash (`--`) acts as a comment, causing the rest of the original query (including the password check) to be ignored by the database.

This results in the database returning the first user record, granting instant administrative access.

3. Finding the Encrypted Data

Upon successful login, the administrative dashboard provided an option titled **“Admin: View All Notes”**.

This section displayed five Base64-encoded encrypted messages, each corresponding to a different user, which were clearly the fragmented components of the final flag.

User	Ciphertext (Base64)
admin	`EDc6NQ8YZQ0WXBkBbw==`
alice	`N0lmBBxcCQIHWwRGAG0=`
bob	`NUYMHAAwZgcqH0dRQgFBIDo=`
carol	`dxwdLxxeMgVGAitfA0FGZwJQATo=`
dave	`LkYKB EcdCQIBDx8BQk8=`

4. Cracking the XOR Encryption

The encryption method was identified as a simple **repeating-key XOR cipher**. The known flag format, `SECE{...`, was utilized for a **known-plaintext attack**.

XOR Logic for Key Recovery

The XOR operation’s reversible property allows for key recovery when both the plaintext and ciphertext are known:

- `Plaintext ⊕ Ciphertext = Key`

The first ciphertext (`EDc6NQ8YZQ0WXBkBbw==`) was Base64-decoded and XORed with the known plaintext prefix, which was extended to `SECE{w3lc0m3_` to identify the full repeating key.

Recovered Repeating Key: ``` CryptoVault2025 ```

5. Final Decryption

The key `CryptoVault2025` was used to decrypt each of the five Base64-decoded messages individually (the key was reapplied from the start for each message).

User	Decrypted Part
admin	`SECE{w3lc0m3_`
alice	`t0_th3_cr7pt0_`
bob	`v4ult_0f_s3cr3t_`
carol	`4nd_h1dd3n_m3ss4g3s_`
dave	`m4st3r_h4ck3r}`

Final Flag

Concatenating the decrypted parts yields the complete flag:

```
SECE{w3lc0m3_t0_th3_cr7pt0_v4ult_0f_s3cr3t_4nd_h1dd3n_m3ss4g3s_m4st3r_h4ck3r}
```