Secret Vault - Findings Report

Table of contents

Contest Summary

Sponsor: First Flight #46

Dates: Aug 14th, 2025 - Aug 21st, 2025

See more contest details here (https://codehawks.cyfrin.io/c/2025-07-secret-vault)

Results Summary

Number of findings:

- High: 2
- Medium: 0
- Low: 0

High Risk Findings

H-01. Plaintext Secret Exposure via UTF-8 Encoding in Vault

Root + Impact

Secret here is set as UTf-8 which is not encrypting data and will be decode automatically on blockchain explorer,where any one can see it as plain text,so any one can see the secret of any user in plain text on chain.

Description

- A “secret vault” should not expose raw sensitive data to the public. On-chain data is readable by anyone, so secrets must be encrypted or stored as hashes/commitments.
- The Vault stores secret in **UTF-8 encode** format**.**UTF-8 is a character encoding, not an encryption method. It merely defines how text is represented as bytes. If you store secret data in UTF-8 on a blockchain, it remains **plaintext** and fully readable to anyone.

- Secrets are stored using `string::utf8(secret)` which **encodes data as UTF-8**, not encrypts it. This results in **plaintext storage** on-chain, fully visible to anyone via:
  - Blockchain explorers
  - SDKs
  - Node queries

```
// Root cause in the codebase with @> marks to highlight the relevant section
 public entry fun set_secret(caller:&signer,secret:vector<u8>){
  @> let secret_vault = Vault{secret: string::utf8(secret)};
     move_to(caller,secret_vault);
     event::emit(SetNewSecret {});
 }
```

Risk

Likelihood:

- This occurs every time `set_secret` is used UTF-8/plaintext is committed to state.
- When `get_secret` is used plain text will be logged on the blockchain as plain text.
- Blockchain transparency ensures any node or indexer can read the data.
- Message or transaction done on chain is visible to the public over blockchain explorer.

Impact:

- Complete loss of confidentiality for all “secrets” stored.
- Undermines the purpose of having a “secret vault” entirely.
- Any one/Everyone can see other user's secret stored in the Smartcontract.

Proof of Concept

```
This is how our test secret will be shown on blockchain explorer


{
  "resources": [
    {
      "type": "0xcc::vault::Vault",
      "data": {
        "secret": "my_password_123"
      }
    }
  ]
}
```

Recommended Mitigation

- Do not store any secret/sensetive data on chain.
- Encrypt and the Secret with strong algorith and than store that encrypted values on chain.
- Implement Move Commitment scheme for better security as shown below.

```
let commitment = hash::sha3_256(b"my_password_123");
let vault = Vault { secret: string::utf8(commitment) };
```

H-02. Global Resource Exposure Leading to Confidential Data Leakage

Root + Impact

Sensitive data is stored in plaintext within Move’s global storage, making it publicly accessible despite function-level access control. This violates confidentiality principles and exposes secrets to unauthorized actors.

Description

- In Move, resources stored using move_to are placed in global storage under the caller’s account. This allows any external observer to query the blockchain and inspect the stored data.
- The module stores a secret string using move_to(caller, secret_vault);, but this secret is not encrypted. Although access to the get_secret function is restricted, the actual data remains readable via standard tooling, violating data confidentiality.

```
// Root cause in the codebase with @> marks to highlight the relevant section
public entry fun set_secret(caller: &signer, secret: vector<u8>) {
    let secret_vault = Vault { secret: string::utf8(secret)};
    @> move_to(caller, secret_vault); // Secret stored in plaintext under caller's account
}
```

Risk

Likelihood:

- Any actor with access to a full node, indexer, or blockchain explorer can query global storage at any time.
- The Move language does not provide native encryption or obfuscation for stored resources, making plaintext data trivially accessible.

Impact:

- Secrets or private messages can be harvested by adversaries.
- No authentication or authorization is required to access this data.
- The `get_secret` function enforces access control, but it is irrelevant because the data is already exposed in global storage.

- This violates **OWASP A02:2021 – Cryptographic Failures** and **SWC-136 – Unencrypted Secrets**, as sensitive data is stored without any form of encryption or obfuscation.

Proof of Concept

Below shown output is accessible without invoking `get_secret`, bypassing all access control logic with mentoned query.

```
aptos view-resource --address 0xcc --resource-type secret_vault::vault::Vault


Output
{
  "secret": "i'm a secret"
}
```

Recommended Mitigation

- Encrypt Secret off-chain and store it on chain
- Use a Commit-Reveal Scheme,as shown in below code snippet

```
- move_to(caller, Vault { secret: string::utf8(secret) });
+ let commitment = hash::sha3_256(secret); // or blake2b, keccak256
+ move_to(caller, Vault { secret: string::utf8(commitment) });
```