[S-#] Anyone can read password and reset it by directly reading on-chain storage(Root Cause + Impact)

Description:

The `PasswordStore` contract attempts to protect a sensitive password by declaring it as a `private` state variable. However, in Ethereum, all contract storage is publicly readable, regardless of Solidity visibility modifiers.

An attacker can directly read the password from the contract's storage slot using RPC calls (e.g., `eth_getStorageAt`) or during local testing via `vm.load`. Once the password is obtained, the attacker can trivially reset it by calling `PasswordStore::setPassword()`.

The root cause is the incorrect assumption that `private` variables are hidden from on-chain observers, leading to sensitive data being stored in plaintext on-chain.

```
contract PasswordStore {
    error PasswordStore__NotOwner();

    address private s_owner;
@>  string private s_password;

    event SetNetPassword();
    ...
}
```

Impact:

- Loss of confidentiality: Any external party can read the stored password directly from contract storage.
- Unauthorized state modification: Since the password is readable, any attacker can call `setPassword()` with the correct value and reset the password.
- Broken access control guarantees: The contract provides no real protection against unauthorized password resets.
- Complete security failure of the password mechanism, rendering it ineffective.

This vulnerability fully compromises the intended security model of the contract.

Proof of Concept:

The following test demonstrates how an attacker can:

- Read the password directly from storage.
- Convert it back into a string.
- Use it to successfully call `setPassword()`.

```
function test_anyone_can_reset_password() public {
        bytes32 rawData = vm.load(address(passwordStore), bytes32(uint256(1)));

        uint256 len = uint8(rawData[31]) / 2;
        bytes memory out = new bytes(len);
```

```
        for (uint256 i = 0; i < len; i++) {
            out[i] = rawData[i];
        }
        assertEq(string(out), "myPassword");

        vm.expectEmit(false, false, false, false);
        emit SetNetPassword();
        passwordStore.setPassword(string(out));

    }
```

This proof confirms that any user can read and reset the password without authorization.

Recommended Mitigation:

- Never store sensitive information (e.g., passwords, secrets, private keys) directly on-chain, even if marked as private.
- If password-like verification is required:
    - Use hash-based verification (e.g., keccak256(password) comparisons), understanding that this still allows offline brute-force attacks.
    - Prefer signature-based authentication (e.g., ECDSA signatures).
    - Use off-chain secret management combined with on-chain verification.
- Reconsider whether a password mechanism is appropriate at all in a public blockchain environment.