Protocol Audit Report

Azriel

April 21, 2025

Protocol Audit Report

Version 1.0

Cyfrin.io

April 21, 2025

Protocol Audit Report

Azriel

April 21, 2025

Prepared by: Azriel

Table of Contents

Protocol Summary

Stores passwords

Disclaimer

The Azriel(me) team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
	High	H	H/M	M
Likelihood	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Scope

`Commit Hash: 7d55682ddc4301a7b13ae9413095feffd9924566`

In Scope:

```
./src/
-- PasswordStore.sol
```

Solc Version: 0.8.18
Chain(s) to deploy contract to: Ethereum

Roles

Owner: The user who can set the password and read the password.
Outsiders: No one else should be able to set or read the password.

Executive Summary

the audit went great

Issues found

Severity	Number of Issues Found
High	2
Medium	0
Low	0
Info	1

Findings

High

[H-1] Variables on chain are not private, allowing anyone to view the password

Description: There is a vulnerability with `PasswordStore::s_password` due to the nature of the blockchain as storing private variables on chain is still readable by the public. This allows attackers to bypass the ownership checks within `PasswordStore::getPassword()` as well since the variable can be read directly.

Impact: Secrets can be read and stolen by anyone

Proof of Concept:

1. Create a local chain

```
make anvil
```

2. Deploy the scripts. The address of the contract will be returned.

```
make deploy
```

3. Run the following command with the contract address

```
cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1 --rpc-url http://127.0.0.1:8545
```

This would give the following output:

```
0x6d7950617373776f726400000000000000000000000000000000000000000014
```

4. Convert the value from hexadecimal to string

```
cast parse-bytes32-string 0x6d7950617373776f72640000000000000000000000000000000000000000001
```

Recommended Mitigation: It is inherently insecure to store private values on chain due to the public nature of the blockchain. So there is no reason a contract like this should even be deployed and used in the first place. Consider using off-chain solutions such as password managers instead.

[H-2] Lack of Access Control checks for setting passwords allows anyone to set the password

Description: There is a vulnerability with `PasswordStore::setPassword` due to missing Access Control Checks, allowing anyone to set passwords. This contradicts the goal of only allowing the owner to set the password.

Impact: Password can be set by any user, overriding the values stored by the owner

Proof of Concept: To demonstrate this we have included an additional test suite to test if calling `PasswordStore::setPassword` would revert if the caller is not the owner. The following test case was added:

```
function test_non_owner_set_password_reverts() public {
    vm.startPrank(address(1));

    vm.expectRevert(PasswordStore.PasswordStore__NotOwner.selector);
    passwordStore.setPassword("newPassword");
}
```

After running `forge test`, the test failed as this function does not revert as expected, showing that it is possible for non-owners to set passwords.

Recommended Mitigation: Include additional access control checks before being allowed to make this call. The following check should be included in the function:

```
if (msg.sender != s_owner) {
    revert PasswordStore__NotOwner();
}
```

Informational

[I-1] Inaccurate documentation of function `PasswordStore::getPassword`

Description: The documentation of the function `PasswordStore::getPassword` includes an additional `newPassword` variable that is not used.

Impact: May cause confusion for future auditors and developers

Proof of Concept: NA

Recommended Mitigation: Remove this parameter as it is not needed.