

PasswordStore Audit Report

Prepared by: Payal Lead Auditors:

- [Payal](#)

Assisting Auditors:

- None

Table of contents

► Details

Contents

- [PasswordStore Audit Report](#)
- [Table of contents](#)
- [About me](#)
- [Disclaimer](#)
- [Risk Classification](#)
- [Audit Details](#)
 - [Scope](#)
- [Protocol Summary](#)
 - [Roles](#)
- [Executive Summary](#)
 - [Issues found](#)
- [Findings](#)
 - [High](#)
 - [\[H-1\] Passwords stored on-chain are visable to anyone, not matter solidity variable visibility](#)
 - [\[H-2\] PasswordStore::setPassword is callable by anyone](#)
 - [\[I-1\] The PasswordStore::getPassword natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect](#)

About me

As a new security researcher exploring Web3 protocols, I'm committed to securing decentralized systems. This report highlights my efforts to improve security practices in digital ecosystems.

Disclaimer

The team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the 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
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

Audit Details

The findings described in this document correspond the following commit hash:

```
2e8f81e263b3a9d18fab4fb5c46805ffc10a9990
```

Scope

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

Protocol Summary

PasswordStore is a protocol dedicated to storage and retrieval of a user's passwords. The protocol is designed to be used by a single user, and is not designed to be used by multiple users. Only the owner should be able to set and access this password.

Roles

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

For this contract, only the owner should be able to interact with the contract.

Executive Summary

Issues found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	1
Gas Optimizations	0
Total	0

Findings

High

[H-1] Passwords stored on-chain are visable to anyone, not matter solidity variable visibility

Description: All data stored on-chain is visible to anyone, and can be read directly from the blockchain. The `PasswordStore::s_password` variable is intended to be a private variable, and only accessed through the `PasswordStore::getPassword` function, which is intended to be only called by the owner of the contract.

However, anyone can directly read this using any number of off chain methodologies

Impact: The password is not private.

Proof of Concept: The below test case shows how anyone could read the password directly from the blockchain. We use `foundry's cast` tool to read directly from the storage of the contract, without being the owner.

- 1. Create a locally running chain

```
make anvil
```

- 2. Deploy the contract to the chain

```
make deploy
```

- 3. Run the storage tool

We use `1` because that's the storage slot of `s_password` in the contract.

```
cast storage <ADDRESS_HERE> 1 --rpc-url http://127.0.0.1:8545
```

[illegible][illegible]

myPassword

[H-2] PasswordStore::setPassword is callable by anyone

```
function setPassword(string memory newPassword) external {
@>    // @audit - There are no access controls here
    s_password = newPassword;
    emit SetNetPassword();
}
```

Proof of Concept:

```
function test_anyone_can_set_password(address randomAddress) public {
    vm.prank(randomAddress);
    string memory expectedPassword = "myNewPassword";
    passwordStore.setPassword(expectedPassword);
    vm.prank(owner);
    string memory actualPassword = passwordStore.getPassword();
    assertEq(actualPassword, expectedPassword);
}
```

Recommended Mitigation: Add an access control modifier to the `setPassword` function.

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

[I-1] The `PasswordStore::getPassword` natspec indicates a parameter that doesn't exist, causing the natspec to be incorrect

Description:

```
/*  
 * @notice This allows only the owner to retrieve the password.  
@> * @param newPassword The new password to set.  
 */  
function getPassword() external view returns (string memory) {
```

The natspec for the function `PasswordStore::getPassword` indicates it should have a parameter with the signature `getPassword(string)`. However, the actual function signature is `getPassword()`.

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

```
- * @param newPassword The new password to set.
```