TACITVS

Protocol Security Audit Report

Sine ira et studio.
"WITHOUT ANGER OR BIAS."

Tacitus, Annals 1.1

TACITVS

Auctor Princeps

August 10, 2025

CONTENTS

Prepared by: TACITVS Lead Security Researcher: - I.Leskov(TACITVS)

I. Table of Contents

II. Protocol Summary

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

III. Disclaimer

The TACITVS 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.

IV. Risk Classification

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

Likelihood	Impact	High	Medium	Low
Low		M	M/L	L

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

V. Audit Details

A. Commit Hash

```
7d55682ddc4301a7b13ae9413095feffd9924566
```

B. Scope

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

C. Roles

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

VI. Executive Summary

Objective: Comprehensive security review of the PasswordStore v1.0 smart contract for access control vulnerabilities, data privacy issues, and code quality assessment.

Severity	Count
High	2
Medium	0
Low	0
Info	1
Total	**3**

Key Risk Areas: - **Critical Access Control Gaps:** Missing owner-only restrictions - **Data Privacy Violations:** On-chain password storage - **Documentation Inconsistencies:** Misleading NatSpec comments

Top 3 Recommendations: 1. Implement proper access control with `onlyOwner` modifier 2. Reconsider the fundamental architecture for password privacy 3. Update documentation to match actual function signatures

A. Issues Found

Severity	Number of issues found
High	2
Medium	0
Low	0
Info	1
Total	3

VII. Findings

A. High

[H-1] Password Storage on Public Blockchain
All data stored on-chain is visible to anyone. The `PasswordStore::s_password` variable can be read directly from blockchain storage, completely breaking the protocol's privacy.

Impact: Anyone can read the private password, severely breaking the functionality of the protocol.

Proof of Concept: The below test case shows how anyone can read the password directly from the blockchain.

1. Create a locally running chain:

```
make anvil
```

2. Deploy the contract to the chain:

```
make deploy
```

3. Run the storage tool (slot 1 for `s_password`):

```
cast storage <CONTRACT_ADDRESS_HERE> 1
```

Recommended Mitigation: Due to this, the overall architecture of the contract should be rethought. One could encrypt the password off-chain, and then store the encrypted password on-chain.

[H-2] Missing Access Control on setPassword()
The `PasswordStore::setpassword` function lacks `onlyOwner` modifier, allowing anyone to change the password and break the contract's intended functionality.

Impact: Anyone can set/change the password of the contract, severely breaking the contract intended functionality.

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

B. MEDIUM

No medium severity findings were identified during this audit.

C. LOW

No low severity findings were identified during this audit.

D. INFORMATIONAL

[I-1] Incorrect NatSpec Documentation
The `PasswordStore::getPassword` function documentation incorrectly specifies a parameter that doesn't exist in the function signature.

Description: The `PasswordStore::getPassword` function signature is `getPassword()` which the natspec say it should be `getPassword(string)`.

Impact: The natspec is incorrect.

Recommended Mitigation: Remove the incorrect natspec line.

PROOF & CONNECT

This report is cryptographically anchored on-chain. Scan the left QR to verify the exact SHA-256 fingerprint and blockchain transaction. Scan the right QR to explore my portfolio or book a call.

Scan to verify on-chain proof

Scan for portfolio & booking