PasswordStore Audit Report

Version 1.0

alphabuddha1357

October 20, 2023

PasswordStore Audit Report

alphabuddha1357

October 20, 2023

Prepared by: alphabuddha1357

Lead Auditors:

- alphabuddha1357

Table of Contents

Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release, and does not give any warranties on finding all possible security issues of the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contract(s). Last but not least, this security audit should not be used as investment advice.

Protocol Summary

The protocol is designed for securely storing passwords. Users can store a password and subsequently retrieve it, while preventing unauthorized access by others.

Audit Details

Scope

In the following, we show the Git repository of reviewed files and the commit hash value used in this audit. Note this audit only covers the following contacts: ./src/PasswordStore.sol.

https://github.com/Cyfrin/2023-10-PasswordStore (856ed94)

Severity Criteria

Summary of Findings

ID	Severity	Subject
1	High	private visibility

Tools Used

High

1. private visibility

In Ethereum, the private visibility for state variables in a contract means that they are not directly accessible from outside the contract. However, it's important to understand that Ethereum's design and architecture make it difficult for anyone to truly make a variable completely private.

While it's true that contract storage slots can be accessed using low-level functions like web3.eth.getStorageAt, it's not a straightforward process to access private variables, and it may require knowledge of the contract's storage layout. Even then, it's generally considered unethical and potentially illegal to access private contract storage without authorization.

The "s_password" within the contract can always be read through "vm.load" of Foundry or "web3.eth.getStorageAt" by bypassing slot 1 and the contract address.

Medium

Low

Informational

Gas