PasswordStore 監査 レポート

Version 1.0

Jeremy Bru ・ジェレミー　ブルー

February 17, 2024

PasswordStore 監査 レポート

Jeremy Bru ・ジェレミー　ブルー

2024 年 2 月 16 日

最小限の監査レポート・PasswordStore

作成者: Jeremy Bru (Link)

主任監査役・リードセキュリティ担当:

- Jeremy Bru

Contact: 一

目次

- 目次
- プロトコル概要
- 免責事項
- リスク分類
- 監査の詳細
  - スコープ
  - 役割
- エグゼクティブサマリー
  - 発見された問題
- 問題の発見
  - 高
  - 中
  - 低
  - 情報系
  - ガス

プロトコル概要

- ユーザーが登録した任意のパスワードを保存します。
- ユーザーはパスワードを保存し、後でそれを取得できるようにする必要があります。
- パスワードを登録したユーザー以外の他者は、パスワードにアクセスできないようにする必要があります。

免責事項

私、Jeremy Bru は、与えられた期間内にコードの脆弱性をできるだけ多く見つけるために全力を尽くしましたが、本書類に提供された発見に対しては一切の責任を負いません。チームによるセキュリティ監査は、基盤となるビジネスや製品の推薦ではありません。監査は時間を区切って行われ、コードのレビューは契約の Solidity 実装のセキュリティ側面にのみ焦点を当てて行われました。

リスク分類

		インパクト		
		高	中	低
可能性	高	高	高/中	中
	中	高/中	中	中/低
	低	中	中/低	低

CodeHawks (Link) の重大度マトリックスを使用して重大度を判断します。詳細については、ドキュメンテーションを参照してください。

監査の詳細

Commit Hash: 7d55682ddc4301a7b13ae9413095feffd9924566

スコープ

```
./src/
|___ PasswordStore.sol
```

役割

- オーナー・所有者：パスワードを設定し、読み取ることができるユーザー。
- オーナー以外のユーザー：他の誰もパスワードを設定または読み取ることができないようにすべきです。

エグゼクティブサマリー

発見された問題

深刻度	見つかった問題の数
高	3
中	0
低	0
情報系	2
——-	————————
合計	4

問題の発見

- S = Severity: 深刻度
- クリティカル・クリット (Crit)= 非常に高い
- 情報系　= お知らせ事項
- 例：S-低＋番号 = 順番的に並んでいる。

高

[S-高 1 ・クリティカル] 可視性キーワードに関わらず,PasswordStore:: s_password は誰にでも公開されて見えます。

説明:

`PasswordStore:: private` キーワードはコントラクト内の変数の可読性に関するものですが, それでも誰にでも見えます。これはプロトコルの目的を完全に破壊します。

```
PasswordStore:: string private s_password;
```

https://docs.soliditylang.org/en/v0.8.24/cheatsheet.html#function-visibility-specifiers

影響:

コントラクト内のパスワードは安全ではありません。

概念実証:

コントラクト内のプライベート変数を読む方法を示す証拠コード:

ローカルチェーンを実行

```
1 anvil
```

コントラクトを導入する

```
1 make deploy
```

`s_password`のストレージ値を取得

```
1 cast storage 0x5FbDB2315678afecb367f032d93F642f64180aa3 1
```

返されたバイト:

```
1 0x6d7950617373776f726400000000000000000000000000000000000000000014
```

バイトを文字列に

```
1 cast parse-bytes32-string 0
    x6d7950617373776f726400000000000000000000000000000000000000000014
```

返された文字列:

```
1 myPassword
```

推奨される軽減策:

その他の質問:

- // q なぜパスワードは単一の変数に格納されていますか？
- // q 複数のパスワードを格納するためのアドレスから文字列へのマッピングはどこにありますか？
- // q なぜパスワードは暗号化されていませんか？

問題を解決する方法のアイデア:

アーキテクチャは再検討されるべきです。パスワードはハッシュ化され, 暗号化されるべきです。マッピングは, できるだけ多くのユーザーからの複数のパスワードを格納するために使用されるべきです。

オフチェーンで暗号化し, オンチェーンにハッシュを格納することができます。

- 別のパスワードを覚える必要があります。
- 復号化されたパスワードを表示するビュー関数を削除してください。

これはアーキテクチャの全面的な再検討であるため, この問題に対する修正は書いていません。

[S-高２] アクセス制御攻撃。誰でも新しいパスワードを設定できます。

説明:

- // q なぜ誰でも新しいパスワードを設定できますか？
- // @audit アクセス制御攻撃。高い。

```
/*
 * @notice この関数は,所有者のみが新しいパスワードを設定できるようにします。
 * @param newPassword 設定する新しいパスワード。
 */
function setPassword(string memory newPassword) external {
    s_password = newPassword;
    emit SetNetPassword();
}
```

影響:

以前選択されたパスワードは上書きされ, 誰でも変更することができます。

一瞬でパスワードの制御を失い, その結果に対処することになります。

概念実証:

テストーケース `PasswordStore.t.sol`:

- コントラクトが導入される時に, パスワードは`myPassword`。
- 所有者以外のアドレスを使って,`PasswordStore::setPassword`の関数に別なパラメーター内容を入力する。`myNewPassword`
- そのあとは, 所有者として,`PasswordStore::getPassword`の関数からパスワードを読み取る。
- もし, パスワードが`myNewPassword`となっている場合は, テストが成功です。↓(成功)

Code

```
    function test_anyone_can_change_password(address randomAddress)
        public {
        vm.assume(randomAddress != owner);
        vm.startPrank(randomAddress);
        string memory expectedPassword = "myNewPassword";
        passwordStore.setPassword(expectedPassword);

        vm.startPrank(owner);
        string memory actualPassword = passwordStore.getPassword();
        console.log(actualPassword);
        assertEq(actualPassword, expectedPassword);
    }
```

推奨される軽減策:

アクセス制御を管理するために,OpenZeppelin の Ownable のようなライブラリを使用することができます。

緊急時にコントラクトを一時停止するために,OpenZeppelin の Pausable のようなライブラリを使用することができます。

`modifier`または所有者のみのチェックを追加します。

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

```
/*
 * @notice この関数は,所有者のみが新しいパスワードを設定できるようにします。
 * @param newPassword 設定する新しいパスワード。
 */
function setPassword(string memory newPassword) external {
    if (msg.sender != s_owner) {
        revert PasswordStore__NotOwner();
    }
    s_password = newPassword;
    emit SetNetPassword();
}
```

[S-高３] 所有権/アクセス制御攻撃。パスワードを見ることができるのは所有者だけで,ユーザーはできません。

説明:

コードドキュメンテーションには, 次のように記載されています:

```
PasswordStore:: @notice これにより,所有者のみがパスワードを取得できます。
```

プロジェクトドキュメンテーションには, 次のように記載されています:

```
ユーザーはパスワードを保存し,後でそれを取得できるようにする必要があります。他の人がパスワードにアクセスすることはできません。
```

- したがって, 所有者はユーザーのパスワードを見ることができません。**影響:**

他のユーザーによるパスワードのアクセスはできません。所有者による可能なパスワードの漏洩につながります。

```
    function getPassword() external view returns (string memory) {
        if (msg.sender != s_owner) {
            revert PasswordStore__NotOwner();
        }
        return s_password;
    }
```

推奨される軽減策:

アーキテクチャを再検討するか, 明確にドキュメンテーションを変更してください。ユーザーにとって安全ではありません。

中

- None

低

- None

情報系

[S-情報系] プラグマのバージョンが古いです。その後に破壊的な変更がないか確認してください。

説明:

プラグマのバージョンは最新の安定版ではありません:

// これは正しいバージョンですか？

影響:

古いバージョンは, 最新バージョンで修正されたエクスプロイトや脆弱性につながる可能性があります。

コントラクト:`PasswordStore.sol`

概念実証:

0.8.18 -> 0.8.24

推奨される軽減策:

0.8.18 -> 0.8.24

[S-低 1・情報系] getPassword 関数にパラメーターが欠けているか, 説明が間違っています。それを追加するか, または削除してください。

説明:

// @audit パラメータとして設定するパスワードはありません。以下が必要ですか？

```
PasswordStore:: @param newPassword 新しく設定するパスワード
```

影響:

説明が間違っていても影響は小さい。それ以外の場合は, 関数の修正が必要です。

概念実証:

この行を削除,

```
@param newPassword 新しく設定するパスワード
```

```
/*
* @notice これにより,所有者のみがパスワードを取得できます。
* @param newPassword 新しく設定するパスワード。
*/

function getPassword() external view returns (string memory) {
if (msg.sender != s_owner) {
// q 所有者のみがパスワードを読むことができますか？
revert PasswordStore__NotOwner();
}
return s_password;
}
```

推奨される軽減策:

パラメータ `PasswordStore:: newPassword` を追加するか, 説明からそれを削除してください。

```
- * @param newPassword 新しく設定するパスワード
```

ガス

- None