Vault

Smart Contract Audit Report
Prepared for Coin98

Date Issued: Apr 29, 2022
Project ID: AUDIT2022027
Version: v1.0
Confidentiality Level: Public

Report Information

Project ID	AUDIT2022027
Version	v1.0
Client	Coin98
Project	Vault
Auditor(s)	Peeraphut Punsuwan Ronnachai Chaipha Fungkiat Phadejtaku
Author(s)	Peeraphut Punsuwan Ronnachai Chaipha Fungkiat Phadejtaku
Reviewer	Patipon Suwanbol
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.0	Apr 29, 2022	Full report	Peeraphut Punsuwan Ronnachai Chaipha Fungkiat Phadejtaku

Contact Information

Company	Inspex
Phone	(+66) 90 888 7186
Telegram	t.me/inspexco
Email	audit@inspex.co

Table of Contents

1. Executive Summary

As requested by Coin98, Inspex team conducted an audit to verify the security posture of the Vault smart contracts in two phases: from Apr 19, 2022 to Apr 21, 2022 and from Apr 28, 2022 to Apr 29,2022. During the audit, Inspex team examined all smart contracts and the overall operation within the scope to understand the overview of Vault smart contracts. Static code analysis, dynamic analysis, and manual review were done in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

1.1. Audit Result

In the initial audit, Inspex found 1 high, 1 medium, and 1 very low-severity issues. With the project team's prompt response in resolving the issues found by Inspex, all issues were resolved or mitigated in the reassessment. Therefore, Inspex trusts that Vault smart contracts have high-level protections in place to be safe from most attacks.

1.2. Disclaimer

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), Inspex suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.

2. Project Overview

2.1. Project Introduction

Coin98 is a Financial Services builder that creates and develops an ecosystem of DeFi protocols, applications, and NFTs on multiple blockchains. The platform can help people to access DeFi services effortlessly.

The `coin98_vault` program is the reward distributor on Solana for users who use the Coin98 Wallet application. They can redeem the reward token in the vault by joining the activity, as an example.

Scope Information:

Project Name	Vault
Website	https://coin98.com/
Smart Contract Type	Solana Program
Chain	Solana
Programming Language	Rust
Category	Vault

Audit Information (Round 1):

Audit Method	Whitebox
Audit Date	Apr 19, 2022 - Apr 21, 2022
Reassessment Date	Apr 28, 2022

Audit Information (Round 2):

Audit Method	Whitebox
Audit Date	Apr 28, 2022 - Apr 29, 2022

The audit method can be categorized into two types depending on the assessment targets provided:

1. **Whitebox**: The complete source code of the smart contracts are provided for the assessment.
2. **Blackbox**: Only the bytecodes of the smart contracts are provided for the assessment.

2.2. Scope

The following smart contracts were audited and reassessed by Inspex in detail:

Initial Audit (Round 1): (Commit: 8f941fa90db1229fdba0f37bea7207c523f5c207)

Contract	Location (URL)
coin98_vault	https://github.com/coin98/coin98-vault/tree/8f941fa90d/solana/programs/vault

Reassessment (Round 1): (Commit: e05c703fb4b6f81f523a682246db98261eaba6ee)

Contract	Location (URL)
coin98_vault	https://github.com/coin98/coin98-vault/tree/e05c703fb4/solana/programs/vault

Initial Audit (Round 2): (Commit: e05c703fb4b6f81f523a682246db98261eaba6ee)

Contract	Location (URL)
coin98_vault	https://github.com/coin98/coin98-vault/tree/e05c703fb4/solana/programs/vault

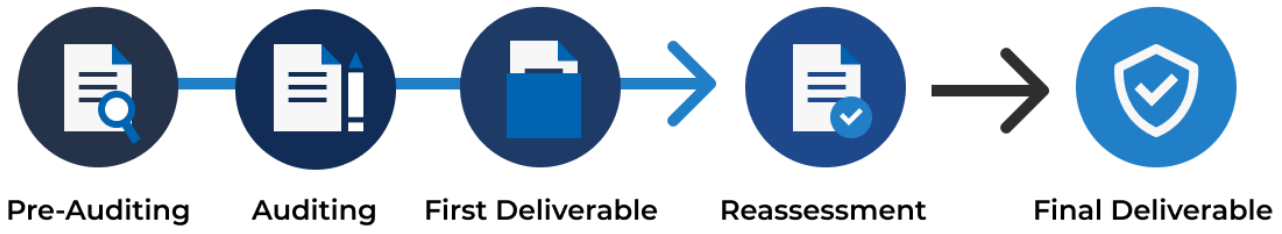
Reassessment (Round 2):
There is no issue that needed the reassessment activity.

The assessment scope covers only the in-scope smart contracts and the smart contracts that they inherit from.

3. Methodology

Inspex conducts the following procedure to enhance the security level of our clients' smart contracts:

1. **Pre-Auditing**: Getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing
2. **Auditing**: Inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals
3. **First Deliverable and Consulting**: Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation
4. **Reassessment**: Verifying the status of the issues and whether there are any other complications in the fixes applied
5. **Final Deliverable**: Providing a full report with the detailed status of each issue



3.1. Test Categories

Inspex smart contract auditing methodology consists of both automated testing with scanning tools and manual testing by experienced testers. We have categorized the tests into 3 categories as follows:

1. **General Smart Contract Vulnerability (General)** - Smart contracts are analyzed automatically using static code analysis tools for general smart contract coding bugs, which are then verified manually to remove all false positives generated.
2. **Advanced Smart Contract Vulnerability (Advanced)** - The workflow, logic, and the actual behavior of the smart contracts are manually analyzed in-depth to determine any flaws that can cause technical or business damage to the smart contracts or the users of the smart contracts.
3. **Smart Contract Best Practice (Best Practice)** - The code of smart contracts is then analyzed from the development perspective, providing suggestions to improve the overall code quality using standardized best practices.

3.2. Audit Items

The following audit items were checked during the auditing activity:

General
Program with Unpublished Source Code
Integer Overflows and Underflows
Bad Randomness
Use of Known Vulnerable Component
Use of Deprecated Component
Solana Account Confusions
Missing Rent Exemption Checking
Use of Upgradable Contract Design
Advanced
Business Logic Flaw
Ownership Takeover
Broken Access Control
Broken Authentication
Denial of Service
Improper Oracle Usage
Memory Corruption
Best Practice
Implicit Type Inference
Function Declaration Inconsistency
Best Practices Violation

3.3. Risk Rating

OWASP Risk Rating Methodology (https://owasp.org/www-community/OWASP_Risk_Rating_Methodology) is used to determine the severity of each issue with the following criteria:

- **Likelihood**: a measure of how likely this vulnerability is to be uncovered and exploited by an attacker
- **Impact**: a measure of the damage caused by a successful attack

Both likelihood and impact can be categorized into three levels: **Low**, **Medium**, and **High**.

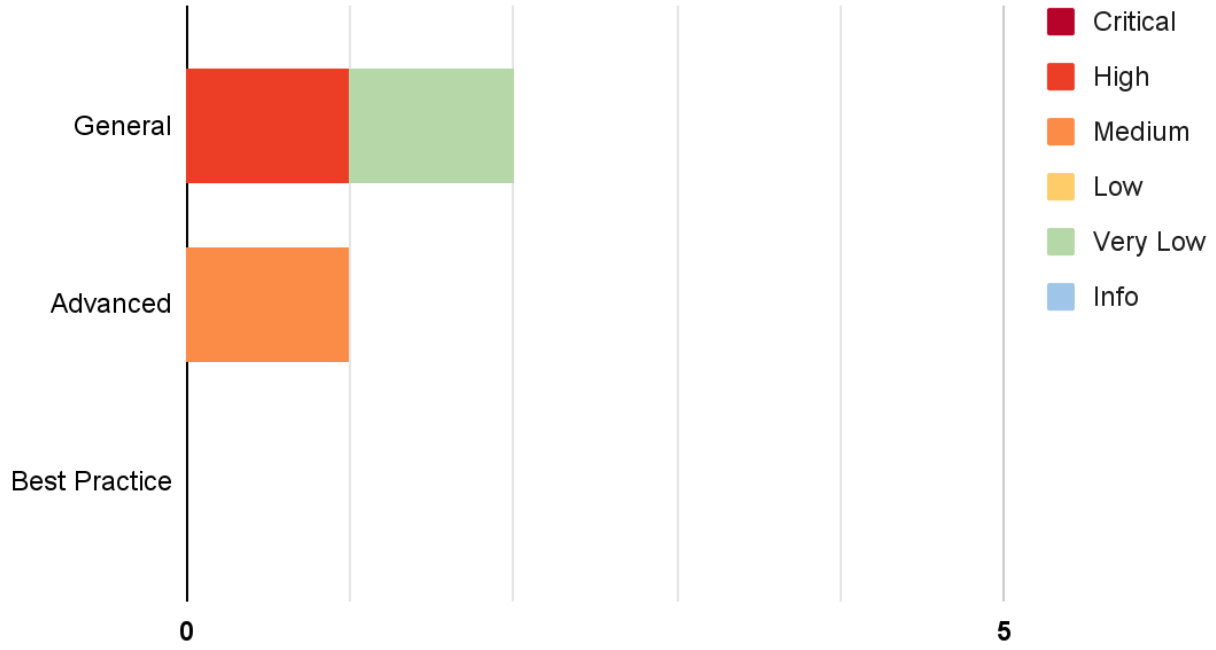
Severity is the overall risk of the issue. It can be categorized into five levels: **Very Low**, **Low**, **Medium**, **High**, and **Critical**. It is calculated from the combination of likelihood and impact factors using the matrix below. The severity of findings with no likelihood or impact would be categorized as **Info**.

Impact \ Likelihood	Low	Medium	High
Low	Very Low	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical

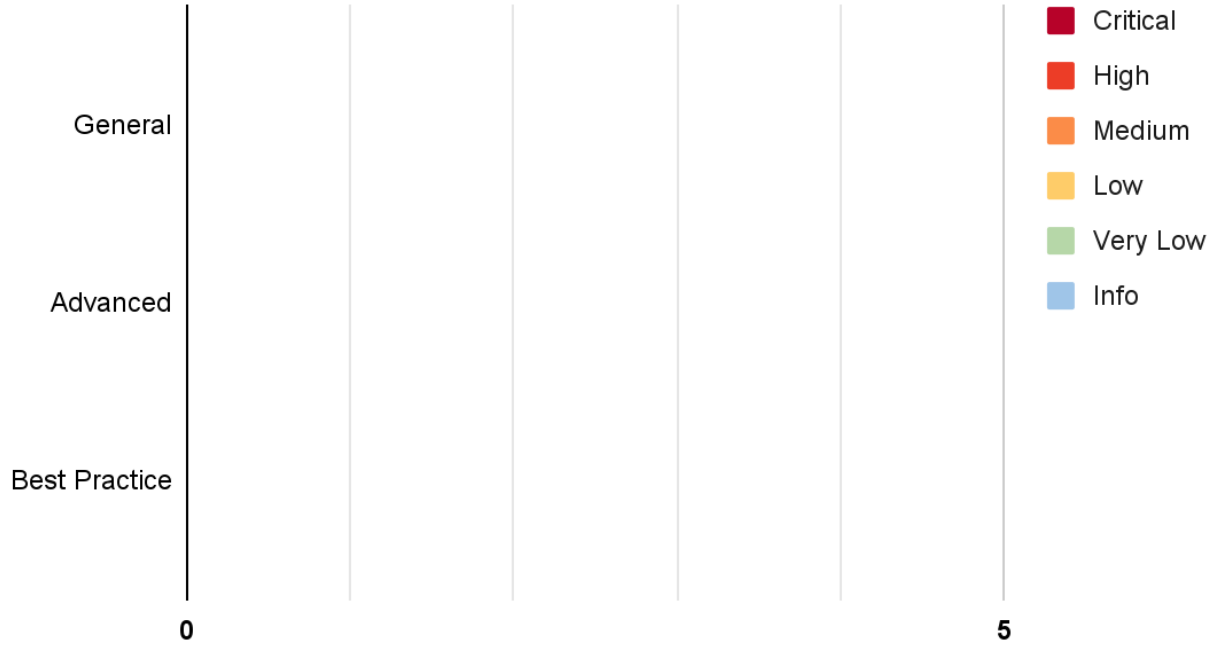
4. Summary of Findings

The following charts show the number of the issues found during the assessment and the issues acknowledged in the reassessment, categorized into three categories: **General**, **Advanced**, and **Best Practice**.

Assessment:



Reassessment:



The statuses of the issues are defined as follows:

Status	Description
Resolved	The issue has been resolved and has no further complications.
Resolved *	The issue has been resolved with mitigations and clarifications. For the clarification or mitigation detail, please refer to Chapter 5.
Acknowledged	The issue's risk has been acknowledged and accepted.
No Security Impact	The best practice recommendation has been acknowledged.

The information and status of each issue can be found in the following table:

Audit Round 1

ID	Title	Category	Severity	Status
IDX-001	Upgradability of Solana Program	General	High	Resolved *
IDX-002	Missing Account Signer Validation	Advanced	Medium	Resolved
IDX-003	Use of Outdated Dependency	General	Very Low	Resolved

Audit Round 2

There is no additional issue found during the assessment.

* The mitigations or clarifications by Coin98 can be found in Chapter 5.

5. Detailed Findings Information

5.1. Upgradability of Solana Program

ID	IDX-001
Scope	Audit Round 1
Target	coin98_vault
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** High **Impact:** High The logic of the affected programs can be arbitrarily changed. This allows the upgrade authority to change the logic of the program in favor to the platform, e.g., transferring the users' funds to the platform owner's account. **Likelihood:** Medium Only the program upgrade authority can redeploy the program to the same program address; however, there is no restriction to prevent the authority from inserting a malicious logic.
Status	Resolved * Coin98 team has confirmed that they will use a multisig wallet as an upgrade authority. This will be controlled by multiple trusted parties to ensure the transparency of the platform. However, the contract has not been deployed yet, so the users should verify that the upgrade authority is controlled by trusted parties before using the platform.

5.1.1. Description

Programs on Solana can be deployed through the upgradable BPF loader to make them upgradable, allowing the program's upgrade authority to redeploy the program with the new logic, bug fixes, or upgrades to the same program address.

However, there is no restriction on how and when the program will be upgraded. This opens up an attack surface on the program, allowing the upgrade authority to redeploy the program with malicious logic and gain unfair benefits from the users, for example, transferring funds out from the users' accounts.

5.1.2. Remediation

Inspex suggests deploying the program as an immutable program to prevent the program logic from being modified.

However, if the upgradability is needed, Inspex suggests mitigating this issue by the following options:

- Using a multisig account controlled by multiple trusted parties as the upgrade authority
- Implementing a community-run governance to control the redeployment of the program

5.2. Missing Account Signer Validation

ID	IDX-002
Scope	Audit Round 1
Target	coin98_vault
Category	Advanced Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** Medium **Impact:** High The attacker can claim all the rewards of the specific user when the attacker knows the `proofs` value of that user. **Likelihood:** Low The proof hash is stored off-chain server, so it is unlikely that the attacker will know the poof hash of other users.
Status	Resolved Coin98 team has resolved this issue as suggested in commit `e05c703fb4b6f81f523a682246db98261eaba6ee`

5.2.1. Description

In the `coin98_vault` program, only users in the whitelist can claim their rewards which are defined in the `schedule` account. The whitelisted users can request their proof hash from the off-chain server and send it to the on-chain program to verify their proof hash, then the rewards will be sent to the user's wallet.

lib.rs

```
245 pub fn redeem_token(
246     ctx: Context<RedeemTokenContext>,
247     index: u16,
248     proofs: Vec<[u8; 32]>,
249     receiving_amount: u64,
250     sending_amount: u64,
251 ) -> ProgramResult {
252     msg!("Coin98Vault: Instruction_RedeemToken");
253 
254     let schedule = &ctx.accounts.schedule;
255     let root_signer = &ctx.accounts.root_signer;
256     let root_token0 = &ctx.accounts.root_token0;
257     let user = &ctx.accounts.user;
258     let user_token0 = &ctx.accounts.user_token0;
259     let clock = &ctx.accounts.clock;
```

```
260     let user_index: usize = index.into();
261
262     if !schedule.is_active {
263         return Err(ErrorCode::ScheduleUnavailable.into());
264     }
265     if clock.unix_timestamp < schedule.timestamp {
266         return Err(ErrorCode::ScheduleLocked.into());
267     }
268
269     let redemption_params = RedemptionParams {
270         index: index,
271         address: *user.key,
272         receiving_amount: receiving_amount,
273         sending_amount: sending_amount,
274     };
275     let redemption_data = redemption_params.try_to_vec().unwrap();
276     let leaf = hash(&redemption_data[..]);
277     let root: [u8; 32] = schedule.merkle_root.clone().try_into().unwrap();
278     if !shared::verify_proof(proofs, root, leaf.to_bytes()) {
279         return Err(ErrorCode::Unauthorized.into());
280     }
281     if schedule.redemptions[user_index] {
282         return Err(ErrorCode::Redeemed.into());
283     }
284     let inner_seeds: &[&[u8]] = &[
285         &[2, 151, 229, 53, 244, 77, 229, 7],
286         &[128, 1, 194, 116, 57, 101, 12, 92],
287     ];
288     let (signer_address, signer_nonce) = Pubkey::find_program_address(
289         &inner_seeds,
290         ctx.program_id,
291     );
292
293     if *root_signer.key != signer_address {
294         return Err(ErrorCode::InvalidSigner.into());
295     }
296     if *root_token0.key != schedule.receiving_token_account {
297         return Err(ErrorCode::InvalidTokenAccount.into());
298     }
299     if schedule.sending_token_mint != solana_program::system_program::ID &&
sending_amount > 0 {
300         return Err(ErrorCode::FeeRequired.into());
301     }
302
303     let schedule = &mut ctx.accounts.schedule;
304     schedule.redemptions[user_index] = true;
305
```

```
306     let seeds: &[&[_]] = &[
307         &inner_seeds[0],
308         &inner_seeds[1],
309         &[signer_nonce],
310     ];
311     let result = shared::transfer_token(&root_signer, &root_token0,
    &user_token0, receiving_amount, &[&seeds]);
312     if result.is_err() {
313         return Err(ErrorCode::TransactionFailed.into());
314     }
315
316     Ok(())
317 }
```

However, at the `RedeemTokenContext` struct, the program does not verify the user who claims the reward is the owner of the `user` account (the account signer) or not. Hence, it results in any user being able to claim the rewards of others by entering the poof hash of other users who have not claimed the rewards yet.

lib.rs

```
532 #[derive(Accounts)]
533 pub struct RedeemTokenContext<'info> {
534
535       #[account(mut)]
536       pub schedule: Account<'info, Schedule>,
537
538       pub root_signer: AccountInfo<'info>,
539
540       #[account(mut)]
541       pub root_token0: AccountInfo<'info>,
542
543       pub user: AccountInfo<'info>,
544
545       #[account(mut)]
546       pub user_token0: AccountInfo<'info>,
547
548       pub token_program: AccountInfo<'info>,
549
550       pub clock: Sysvar<'info, Clock>,
551 }
```

5.2.2. Remediation

Inspex suggests adding the signer validation on the `user` account in the `RedeemTokenContext` struct.

lib.rs

```
532 #[derive(Accounts)]
```

```
533 pub struct RedeemTokenContext<'info> {
534
535       #[account(mut)]
536       pub schedule: Account<'info, Schedule>,
537
538       pub root_signer: AccountInfo<'info>,
539
540       #[account(mut)]
541       pub root_token0: AccountInfo<'info>,
542
543       #[account(signer)]
544       pub user: AccountInfo<'info>,
545
546       #[account(mut)]
547       pub user_token0: AccountInfo<'info>,
548
549       pub token_program: AccountInfo<'info>,
550
551       pub clock: Sysvar<'info, Clock>,
552 }
```

5.3. Use of Outdated Dependency

ID	IDX-003
Scope	Audit Round 1
Target	coin98_vault
Category	General Smart Contract Vulnerability
CWE	CWE-1104: Use of Unmaintained Third Party Components
Risk	**Severity:** Very Low **Impact:** Low Outdated dependencies have publicly known issues and bugs. It is possible that attackers can use those flaws to attack the program and cause monetary loss or business impact to the platform and its users. **Likelihood:** Low With the current dependency version, it is very unlikely that the publicly known bugs and issues will affect these programs.
Status	Resolved Coin98 team has resolved this issue by upgrading anchor-lang to latest version `0.24.2` and upgraded solana-program to version `1.9.13`.

5.3.1. Description

The dependency specified in the programs was outdated. `anchor-lang` version has publicly known inherent bugs that may potentially be used to cause damage to the program or the users of the program.

Cargo.toml

```
1  [package]
2  name = "coin98-vault"
3  version = "0.1.0"
4  authors = ["lukaz"]
5  license = "Apache-2.0"
6  edition = "2018"
7
8  [dependencies]
9  anchor-lang = "0.20.0"
10 solana-program = "1.8.5"
11
12 [features]
13 cpi = ["no-entrypoint"]
14 default = []
15 no-entrypoint = []
```

```
no-idl = []

[lib]
name = "vault"
crate-type = ["cdylib", "lib"]
```

5.3.2. Remediation

Inspex suggests updating the outdated dependency to the latest stable version. For example, at the time of the audit, the latest stable version of `anchor-lang` major 0.20 is 0.20.1 (https://github.com/project-serum/anchor/blob/master/CHANGELOG.md).

Cargo.toml

```
[package]
name = "coin98-vault"
version = "0.1.0"
authors = ["lukaz"]
license = "Apache-2.0"
edition = "2018"

[dependencies]
anchor-lang = "0.20.1"
solana-program = "1.8.5"

[features]
cpi = ["no-entrypoint"]
default = []
no-entrypoint = []
no-idl = []

[lib]
name = "vault"
crate-type = ["cdylib", "lib"]
```

6. Appendix

6.1. About Inspex

Inspex is formed by a team of cybersecurity experts highly experienced in various fields of cybersecurity. We provide blockchain and smart contract professional services at the highest quality to enhance the security of our clients and the overall blockchain ecosystem.

Follow Us On:

Website	https://inspex.co
Twitter	@InspexCo
Facebook	https://www.facebook.com/InspexCo
Telegram	@inspex_announcement

inspeX

CYBERSECURITY PROFESSIONAL SERVICE