ZIPTOS

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Mon Aug 19 2024

ZIPTOS Audit Report

1 Executive Summary

1.1 Project Information

Description	The original bot suite for APTOS
Type	DeFi
Auditors	MoveBit
Timeline	Wed Aug 07 2024 - Mon Aug 19 2024
Languages	Move
Platform	Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/Ziptos-Inc/Ziptos-Vault https://github.com/Ziptos-Inc/Ziptos-Framework.git
Commits	https://github.com/Ziptos-Inc/Ziptos-Vault: a3ef4b47d6163ffd3e215ba71c0550c8aa67143d aca8b06a1f7ac160c2177edfca6f2c875b9bae08 https://github.com/Ziptos-Inc/Ziptos-Framework.git: 22b1b5063e78b10891743fc6ec9b04c4c89dd5cd 154acbffd8963683f2a0fa49d5163f6f5271bc1c

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
B64	sources/base64.move	b926f61b8c52a59a3454b051e61d98730bf234ee
DST	sources/dao_storage.move	cbb29b1b93c1b0b4b6ae4bff972f1e3dce00c338
CON	sources/config.move	afa2d1d35fdb341d382f543d3218aac777a320bf
UTI	sources/utils.move	c75b77121bed91389a356312895ea78336e2f755
ZCO	facoin/sources/zip_coin.move	fc46c41a4a1d061f9916a757476bb1f06949571e
ZIP	zipdrip_framework/sources/zipdrip.move	9c1edcc382ec5fd3ccd2b1922990b1804e9dd3d6
VES	sources/vesting.move	534a12bc99c2e3f3ac41f1550effe2f6a50df021
LOC	sources/locker.move	56b80373d2fc3d0fcdadcc16b576eea767160fee
DEP	ziployer_framework/sources/deployer.move	58242465ae70b8f7bac7949474ccd5fa9ea736ab

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	4	0
Informational	0	0	0
Minor	3	3	0
Medium	1	1	0
Major	0	0	0
Critical	0	0	0

1.4 MoveBit Audit Breakdown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by ZIPTOS to identify any potential issues and vulnerabilities in the source code of the ZIPTOS smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.
During the audit, we identified 4 issues of varying severity, all have been successfully fixed, and no issues remain, listed below.

ID	Title	Severity	Status
DEP-1	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed
DEP-2	Unused Constants	Minor	Fixed
LOC-1	The `set_price()` and `set_price_new()` Functions Perform the Same Function	Minor	Fixed
VES-1	Missing parameter validation	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the ZIPTOS Smart Contract :
Owner

- The owner can initialize the module and mint tokens to a specified address through `my_init_module()`
- The owner can mint additional tokens to specified addresses through `mint()`
- The owner can transfer tokens from one address to another through `transfer()`
- The owner can burn tokens from a specified address through `burn()` The owner can freeze an account to prevent any transfers from that account through `freeze_account()`
- The owner can unfreeze an account allowing transfers from that account through `unfreeze_account()`
- The owner can withdraw tokens from an address through `withdraw()`
- The owner can deposit tokens into a specified address through `deposit()`
- The owner can register new coin types through `register()`
- The owner can deposit coins into the reserve through `deposit()`
- The owner can retrieve the fee collector's address through `get_fee_collector()`
- The owner can withdraw coins from the reserve through `withdraw()`
- The owner can set a new fee collector through `set_fee_collector()`
- The owner can update the fee for new locks through `set_price_new()`
- The owner can update the fee for new locks through `set_price()`
- The owner can configure module initialization through `init_module()`
- The owner can set new vesting fees through `set_price()`

User

- The user can create a new lock with specified coins and unlock time through `lock()`
- The user can withdraw coins from an expired lock through `withdraw()`

- The user can transfer the ownership of a lock to another address through `transfer_ownership()`
- The user can deposit funds and create a vesting schedule through `deposit()`
- The user can claim funds according to the vesting schedule through `claim()`
- The user can withdraw unclaimed funds from their own vesting schedule through `withdraw()`

4 Findings

DEP-1 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

ziployer_framework/sources/deployer.move#70-80

Descriptions:

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract. The `deployer.update_owner()` function directly updates the new owner, which poses a security risk.

```
  entry public fun update_owner(ziptos_framework: &signer, new_owner: address)
acquires Config {
      assert!(
        signer::address_of(ziptos_framework) == @ziptos_framework,
        ERROR_INVALID_ZIPTOS_ACCOUNT
      );
      // only allowed after the deployer is initialized
      assert!(exists<Config>(@ziptos_framework),
ERROR_ERROR_INSUFFICIENT_APT_BALANCE);

      let config = borrow_global_mut<Config>(@ziptos_framework);
      config.owner = new_owner;
  }
```

Suggestion:

It is recommended to use a two-step process for transferring ownership to enhance security.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

DEP-2 Unused Constants

Severity: Minor

Status: Fixed

Code Location:

ziployer_framework/sources/deployer.move#38-41

Descriptions:

There are unused constants in the contract.

```
/// Only fungible asset metadata owner can make changes.
const ENOT_OWNER: u64 = 1;
/// The FA coin is paused.
const EPAUSED: u64 = 2;
```

```
/// Raised if the signer is not authorized to perform an action
const ENOT_AUTHORIZED: u64 = 1;
/// Raised if there is an invalid value for a configuration
const EINVALID_VALUE: u64 = 2;
```

Suggestion:

It is recommended to remove unused constants if there's no further design.

Resolution:

This issue has been fixed. The client has removed unused constants.

LOC-1 The `set_price()` and `set_price_new()` Functions Perform the Same Function

Severity: Minor

Status: Fixed

Code Location:

sources/locker.move#96-102

Descriptions:

The `locker.set_price()` function appears to have the same functionality as `set_price_new()`, making the `set_price()` function redundant.

```
  public entry fun set_price(account: &signer, new_fee_without_decimals: u8) acquires
AccountCapability{
      let cap = borrow_global<AccountCapability>(@coin_factory);
      let resource_account = account::create_signer_with_capability(&cap.signer_cap);

      assert!(signer::address_of(account) == @coin_factory, E_NO_PERMISSION);
      config::set_v1(&resource_account, utf8(CONFIG_PRICE), &(new_fee_without_decimals
as u64));
  }
```

Suggestion:

It is recommended to remove the `set_price()` function to streamline the code and avoid unnecessary duplication.

Resolution:

This issue has been fixed. The client has removed the unnecessary function.

VES-1 Missing parameter validation

Severity: Minor

Status: Fixed

Code Location:

sources/vesting.move#109-174

Descriptions:

The `deposit` function is missing parameter validation; `to_seconds` should be greater than `from_seconds`.

```
    public entry fun deposit<X>(account: &signer, target_addresses: vector<address>,
amounts: vector<u64>, cliffs: vector<u64>, from_seconds: u64, to_seconds: u64)
        acquires AccountCapability, Store, OwnerVestings, InfoStore
    {
        let cap = borrow_global<AccountCapability>(@coin_factory);
        let resource_account = account::create_signer_with_capability(&cap.signer_cap);
        let resource_address = signer::address_of(&resource_account);

        assert!(!vector::is_empty(&target_addresses), E_NO_ETNRIES);
        assert!(vector::length(&target_addresses) == vector::length(&amounts),
E_AMOUNTS_NE_ADDRESSES);
        assert!(vector::length(&amounts) == vector::length(&cliffs),
E_AMOUNTS_NE_ADDRESSES);
        assert!(from_seconds >= timestamp::now_seconds(), E_INVALID_FROM);
       // assert!(to_seconds > from_seconds + ONE_DAY_IN_SECONDS, E_INVALID_TO);
```

Suggestion:

It is recommended to add parameter validation for `to_seconds` and `from_seconds`.

Resolution:

This issue has been fixed. The client has added parameter validation for `to_seconds` and `from_seconds`.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.