verichains

SECURITY AUDIT OF

BELAUNCH LAUNCHPAD

Public Report

Jun 06, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Sui Blockchain	Sui is an innovative, decentralized Layer 1 blockchain that redefines asset ownership. Sui Move feels like a paradigm change in web3 development. Treating objects as 1st class citizens brings composability to a whole new level. Polymedia. We are thrilled to be building on Sui.
Sui Object	The basic unit of storage in Sui is object. In contrast to many other blockchains where storage is centered around accounts and each account contains a key-value store, Sui's storage is centered around objects
Move	Move is a new programming language that implements all the transactions on the Aptos/Sui blockchain.
Move Module	A Move module defines the rules for updating the global state of the Aptos/Sui blockchain. In the Aptos/Sui protocol, a Move module is a smart contract.
IDO	An IDO is a crypto token offering run on a Decentralized Exchange (DEX). Liquidity pools (LP) play an essential role in IDO's by creating liquidity post-sale. A typical IDO lets users lock funds in exchange for new tokens during the token generation event. Some of the raised funds are then added with the new token to an LP before being returned later to the project.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Jun 06, 2023. We would like to thank the BeLaunch for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the BeLaunch Launchpad. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the smart contracts code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About BeLaunch Launchpad

BeLaunch is a decentralized launchpad that allows users to launch their own token and create their own initial token sale. No coding knowledge is required, just simply navigate through to our terminal and design your own token in just a few clicks. BeLaunch offers multiple other features to help you with the overall token launch.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the BeLaunch Launchpad.

SHA256 Sum	File
facb6bdd0df8e67cfe31c0d1f88e642169f401aad923c68173f31a9db0287305	belaunch.move

The issues identified in the file `belaunch.move` have been resolved in the renamed file `ido.move`. The updated scope audit includes:

SHA256 Sum	File
d7b0d90a8257a03e48bcc22abb0f0377ee27f794ee5a6fe4b077bddc597c6c1d	ido.move

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that were considered during the audit of the smart contract:

- Numerical precision errors
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- Gas Usage, Gas Limit and Loops
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The BeLaunch Launchpad was developed using the Move programming language and deployed on the Sui Blockchain.

2.1.1. Privileged Roles & Actors

- Owner of the contract: known as `Publisher`, possesses the authority to create sales.
- Owner of sales: know as `Fundraiser`, enable to modify whitelist, settle in their sales.
- User: Has the ability to buy sales, claim token sales, and claim affiliate rewards.

2.1.2. Important Notes

- Only owner of the contract can create sales.
- Time is expressed in milliseconds (`ms`).
- Sale time is between `start_sale_at` and `end_sale_at`. In this time, user can buy sale (must satisfy `raised_amount <= hard_cap`). If whitelist is available, users must be included. Specially, users can emergency withdraw the raised fund before 15 minutes.
- After sale time end, users can claim affiliate rewards (if available) or claim bought tokens:
  - If `raised_amount` smaller than `soft_cap`, all amount will be refunded.
  - If `tge_percent` greater than 0, the user can be claim percentage of the token amount for the first time. Subsequent claims, users receive a token amount calculated based on the `cycle_percent` and the number of cycles (`vesting_time`) that have passed. This process continues until the `released_amount` is equal to the `total_amount`, signifying the completion of the token claiming cycle.

2.2. Findings

During the audit process, the audit team found some vulnerability issues in the given version of BeLaunch Launchpad.

#	Issue	Severity	Status
1	Round up `time_pass` cause to always revert when calculate vesting amount	CRITICAL	FIXED
2	User can join sale before fundraiser update whitelist	MEDIUM	Acknowledged
3	Wrong timestamp unit	LOW	FIXED
4	`sale_type` must be less than or equal 2	LOW	FIXED
5	The variable name is incorrect	INFORMATIVE	FIXED

2.2.1. [CRITICAL] Round up `time_pass` cause to always revert when calculate vesting amount

Positions:

- `L610#belaunch.move`

2.2.1.1. Description

When a user claims an amount for the second or subsequent time, the code calculates the vesting amount based on the periods between the current time and the time of the first claim. This calculation involves using the `ceil_div()` function, which always rounds the result up by one. Unfortunately, this causes the `vesting_time` to be larger than the actual value, leading to a revert in the next assertion.

```
fun calculate_total_vesting_time(
    last_completed_period: u64,
    vesting_period_duration: u64,
    time_now: u64
): u64 {
    let time_pass: u64 = 0;
    let last_completed = last_completed_period;

    if (time_now > last_completed) {
        let time_day = (time_now - last_completed) / 86400000;
        time_pass = math::ceil_div(time_day, vesting_period_duration); // INCORRECT
CALCULATION
    };
```

```
    time_pass
}

fun calculate_vesting_amount<SaleCoinType, PurchaseCoinType>(
    ...
): u64 {
    let _amount_to_be_released = 0;

    // Distribute vesting
    if (pool.vesting.using_vesting) {
        // Compute claimable amount and check against the amount want to distribute
        ...
        // Short-circuit if vesting hasn't started yet.
        assert!(time_now > user_info.vesting_map.vesting_start, EVESTING_START_TOO_SOON);

        // Check TGE
        if (tge_percent > 0 && user_info.vesting_map.tge_status) {
            ...
        } else {
            ...

            let vesting_time = calculate_total_vesting_time(
                user_info.vesting_map.last_completed_period,
                period_duration,
                time_now
            );
            ...

            let vest_sec = (vesting_time * period_duration) * 86400000;
            let vest_completed_period = user_info.vesting_map.last_completed_period +
vest_sec;
            assert!(time_now > vest_completed_period, EVESTING_START_TOO_SOON); // THE CODE
ALWAYS REVERT AT THIS LINE

            ...
        };
    } else {
        ...
    };
    ...
}
```

RECOMMENDATION

It is recommended to replace the ceil_div() function with the / operation. The code can be fixed as below:

```
fun calculate_total_vesting_time(
    last_completed_period: u64,
    vesting_period_duration: u64,
    time_now: u64
): u64 {
    let time_pass: u64 = 0;
    let last_completed = last_completed_period;

    if (time_now > last_completed) {
        let time_day = (time_now - last_completed) / 86400000;
        time_pass = time_day / vesting_period_duration; // FIXED
    };

    time_pass
}
```

UPDATES

Jun 06, 2023: This issue has been acknowledged and fixed by the BeLaunch team.

2.2.2. [MEDIUM] User can join sale before fundraiser update whitelist

Positions:

- L336#belaunch.move

2.2.2.1. Description

Fundraiser create a sale with whitelist is false by default. With the private sale only sale for whitelist, fundraiser must add whitelist after create. The user can call `mint_user_info()` before whitelist was added to join buy sale in private sale purpose.

```
public entry fun create_sale<SaleCoinType, PurchaseCoinType>(
    ...
) {
    ...

    let pool = Pool<SaleCoinType, PurchaseCoinType> {
        ...
        whitelist: Whitelist<SaleCoinType, PurchaseCoinType> {
            id: object::new(ctx),
            is_whitelist: false, // INCORRECT
            is_shutdown: false,
            list: object_table::new<address, WhitelistedUser<SaleCoinType,
PurchaseCoinType>>(ctx)
        },
        ...
    };
```

```
    ...
}
```

RECOMMENDATION

When create sale, the fundraiser must be allowed to control `is_whitelist`. The code can be fixed as below:

```
public entry fun create_sale<SaleCoinType, PurchaseCoinType>(
    ...
    is_whitelist: bool,
    ...
) {
    ...

    let pool = Pool<SaleCoinType, PurchaseCoinType> {
        ...
        whitelist: Whitelist<SaleCoinType, PurchaseCoinType> {
            id: object::new(ctx),
            is_whitelist, // FIXED
            is_shutdown: false,
            list: object_table::new<address, WhitelistedUser<SaleCoinType,
PurchaseCoinType>>(ctx)
        },
        ...
    };

    ...
}
```

UPDATES

Jun 06, 2023: This issue has been acknowledged.

2.2.3. [LOW] Wrong timestamp unit

Positions:

- `L936#belaunch.move`

2.2.3.1. Description

The timestamp unit used in the code is milliseconds, so failing to multiply by 1000 would result in an incorrect value

```
public entry fun emergency_withdraw<SaleCoinType, PurchaseCoinType>(
    ...
) {
    assert!(user_info.purchased_amount > 0, ELAUNCHPAD_NOT_JOIN);
    let timeend_emergency = pool.end_sale_at - 600; // INCORRECT
    let time_now = clock::timestamp_ms(clock);
```

```
    assert!(time_now < timeend_emergency, EROUND_IS_FINISHED);
    ...
}
```

RECOMMENDATION

The code can be fixed as below:

```
public entry fun emergency_withdraw<SaleCoinType, PurchaseCoinType>(
    ...
) {
    assert!(user_info.purchased_amount > 0, ELAUNCHPAD_NOT_JOIN);
    let timeend_emergency = pool.end_sale_at - 600 * 1000; // FIXED
    let time_now = clock::timestamp_ms(clock);
    assert!(time_now < timeend_emergency, EROUND_IS_FINISHED);
    ...
}
```

UPDATES

Jun 06, 2023: This issue has been acknowledged and fixed by the BeLaunch team.

2.2.4. [LOW] `sale_type` must be less than or equal 2

Positions:

- `create_sale()#belaunch.move`

RECOMMENDATION

It is recommended to add assertion:

UPDATES

Jun 06, 2023: This issue has been acknowledged and fixed by the BeLaunch team.

```
assert!(sale_type <= 2, EWRONG_SALE_TYPE);
```

2.3. Additional notes and recommendations

2.3.1. [INFORMATIVE] The variable name is incorrect

Positions:

- L691#belaunch.move

RECOMMENDATION

Rename `sale_coin_claimed` to `purchare_coin_claimed`

UPDATES

Jun 06, 2023: This issue has been acknowledged and fixed by the BeLaunch team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Apr 06, 2023*	Public Report	Verichains Lab

Table 2. Report versions history