verichains

SECURITY AUDIT OF

FUNARCADE TOKEN SMART CONTRACT

Public Report

Aug 29, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Aug 29, 2023. We would like to thank the FunArcade Token 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 FunArcade Token Smart Contract. 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 1 vulnerable issue in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About FunArcade Token Smart Contract

FAT is a ERC-20 token on the Arbitrum blockchain. FAT can be staked to earn a portion of the platform profits or used for play in their exciting games.

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the FunArcade Token Smart Contract. It was conducted on commit `74bc519f01dce9eb6c99aaa0d1cfcf97efe1f789` from git repository *https://github.com/FunAsiaSolution/FunAsia-Crypto-Token*.

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
`0c84746317ca155e53e43c2c5f48af4863c0be985d74bcee5fb24b42189bf8a1`	`esFATToken.sol`
`b47e7d7c59f143c4f385427231ee5d49ef6daeb67b7105ca9fc66deea883e0da`	`FATToken.sol`

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:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy

- Explicit visibility of functions state variables (external, internal, private and public)
- 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 FunArcade Token Smart Contract was written in `Solidity` language, with the required version to be `^0.8.9`.

2.1.1. FATToken.sol

The `FATToken` token contract extends `ERC20`, `ERC20Permit`, `ERC20Votes` and `Ownable` contracts. With `Ownable`, by default, the contract Owner is the contract deployer, but he can transfer ownership to another address at any time.

The address `_checker` is capable of transferring funds from any user as long as the owner has not turned off `_isChecker`.

Since we do not control the logic of the `_checker` contract, there is no guarantee that `_checker` will not contain any security related issues.

Table 2 lists some properties of the audited Token Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	Funarcade Token
Symbol	FAT
Decimals	18
Total Supply	100,000,000 (x10^{18}) Note: the number of decimals is 18, so the total representation token will be 100,000,000 or 100 million.

Table 2. The Token Smart Contract properties

2.1.2. esFATToken.sol

The `esFATToken` token contract extends `ERC20`, `ReentrancyGuard` and `Ownable` contracts. With `Ownable`, by default, the contract Owner is the contract deployer, but he can transfer ownership to another address at any time.

The main functions within the contract are:

- `convert()`: Converts `FAT` tokens to receive an equivalent amount of `esFATToken`.

- `redeem()`: Users redeem `esFATToken` to retrieve `FAT` tokens, and must wait until the `endTime` to execute `claimRedemption()`.
- `claimRedemption()`: Claims the amount of FAT tokens that have been redeemed.
- `cancelRedemption()`: Allows users to cancel their redemption request.

The contract owner can modify the value of the `duration` variable (the waiting period for users to call the `claimRedemption()` function after redeeming). This variable has no limit.

Table 3 lists some properties of the audited Token Smart Contract (as of the report writing time).

PROPERTY	VALUE
Name	Escrowed Funarcade Token
Symbol	esFAT
Decimals	18

Table 3. The Token Smart Contract properties

2.2. Findings

During the audit process, the audit team found one vulnerability in the given version of FunArcade Token Smart Contract.

2.2.1. esFATToken.sol - The confusion between `owner` and `msg.sender` in the `_removeUserRedemptionId()` function LOW

In cases where there are code changes and `owner` is not equal to `msg.sender`, this function might operate incorrectly and deviate from its intended purpose.

```
function _removeUserRedemptionId(address owner, uint256 redemptionId) internal {
    uint256 length = userRedemptionIds[owner].length;

    for (uint i = 0; i < length; i++) {
        if (userRedemptionIds[owner][i] == redemptionId) {
            for (uint j = i; j < length-1; j++){
                userRedemptionIds[msg.sender][j] = userRedemptionIds[msg.sender][j+1];
            }
            userRedemptionIds[msg.sender].pop();

            break;
        }
    }
}
```

RECOMMENDATION

It is recommended to use the `owner` parameter exclusively.

```
function _removeUserRedemptionId(address owner, uint256 redemptionId) internal {
    uint256 length = userRedemptionIds[owner].length;

    for (uint i = 0; i < length; i++) {
        if (userRedemptionIds[owner][i] == redemptionId) {
            for (uint j = i; j < length-1; j++){
                userRedemptionIds[owner][j] = userRedemptionIds[owner][j+1];
            }
            userRedemptionIds[owner].pop();

            break;
        }
    }
}
```

UPDATES

- *Aug 29, 2023*: This issue has been acknowledged and fixed by the FunArcade Token team.

2.3. Additional notes and recommendations

2.3.1. esFATToken.sol - Optimizing gas efficiency for the `_removeUserRedemptionId()` function INFORMATIVE

Should replace the element to be deleted with the last element and then delete the last element instead of shifting each element upwards.

```
function _removeUserRedemptionId(address owner, uint256 redemptionId) internal {
    uint256 length = userRedemptionIds[owner].length;

    for (uint i = 0; i < length; i++) {
        if (userRedemptionIds[owner][i] == redemptionId) {
            for (uint j = i; j < length-1; j++){
                userRedemptionIds[msg.sender][j] = userRedemptionIds[msg.sender][j+1];
            }
            userRedemptionIds[msg.sender].pop();

            break;
        }
    }
}
```

RECOMMENDATION

Code example:

```
function _removeUserRedemptionId(address owner, uint256 redemptionId) internal {
    uint256 length = userRedemptionIds[owner].length;

    for (uint i = 0; i < length; i++) {
        if (userRedemptionIds[owner][i] == redemptionId) {
            userRedemptionIds[owner][i] = userRedemptionIds[owner][length - 1] //replace
the element to be deleted with the last element
            userRedemptionIds[owner].pop(); //delete the last element

            break;
        }
    }
}
```

UPDATES

- *Aug 29, 2023*: This issue has been acknowledged by the FunArcade Token team.

2.3.2. esFATToken.sol - Missing necessary events INFORMATIVE

Functions such as `convert()`, `redeem()`, `cancelRedemption()`, `claimRedemption()`, and `changeDuration()` are missing essential events. Should add them and emit accordingly.

UPDATES

- *Aug 29, 2023*: This issue has been acknowledged and fixed by the FunArcade Token team, but the `changeDuration()` event is still missing.

2.3.3. FATToken.sol - Consider converting variables `_cap` and `_checker` into constants INFORMATIVE

Position:

- `_cap`#L9
- `_checker`#L12

Should convert the variables `_cap` and `_checker` into constants for improved contract clarity.

UPDATES

- *Aug 29, 2023*: This issue has been acknowledged and fixed by the FunArcade Token team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Aug 25, 2023*	Private Report	Verichains Lab
1.1	*Aug 29, 2023*	Public Report	Verichains Lab

Table 4. Report versions history