CERTIK

Preliminary Comments

AFT

Oct 20th , 2021

Table ofContents

Summary

This report has been prepared for AFT smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

ProjectSummary

Project Name	AFT
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0x49f965fdd87e9ef1083e8581cadd6fc9880b4052#code
Commits	Deployed contract address: 0xA2A1ADBaA25285C85515E0de5c1872d8e80740d2

AuditSummary

Delivery Date	Oct 20,2021
Audit MBNBodology	Static Analysis, Manual Review
Key Components	

VulnerabilitySummary

Total Issues	0
Critical	0
Major	0
Medium	0
Minor	0
Informational	0
Discussion	0

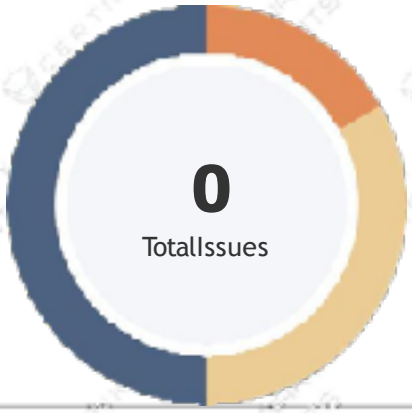
AuditScope

ID	file	SHA256 Checksum
SMC	SafeMars.sol	f90164092172ae6aea6d665923a4e897933c8258739e0ac604be73e5eb9afd1e

CheckResult

Tpye	Security level
smar-contract	High

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**0** (0.00%)
Informational	**0** (0.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
SMC-01	Typos in the contract	Coding Style	Informational	Finished
SMC-02	Incorrect error message	Logical Issue	Minor	Finished
SMC-03	Contract gains non-withdrawable AFT via the `AForceToken` function	Logical Issue	Major	Finished
SMC-04	Return value not handled	Volatile Code	Informational	Finished
SMC-05	Centralized risk in `addLiquidity`	**Centralization / Privilege**	**Critical**	**Finished**
SMC-06	Redundant code	Logical Issue	Informational	Finished
SMC-07	Variable could be declared as `constant`	Gas Optimization	Informational	Finished
SMC-08	3rd party dependencies	Control Flow	Minor	Finished
SMC-09	Missing event emitting	Coding Style	Informational	Finished
SMC-10	Privileged ownership	**Centralization / Privilege**	**Minor**	**Finished**
SMC-11	The purpose of function `deliver`	Control Flow	Informational	Finished
SMC-12	Possible to gain ownership after renouncing the contract ownership	**Logical Issue, Centralization / Privilege**	**Minor**	**Finished**

SMC-01 | Typos in the contract

Category	Severity	Location	Status
Coding Style	Informational	SafeMars.sol: 937, 1177	Finished

Description

No found

Recommendation

SMC-02 | Incorrect error message

Category	Severity	Location	Status
Logical Issue	Minor	SafeMars.sol: 1118	Finished

Description

No found

```
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
}
```

Recommendation

SMC-03 | Contract gains non-withdrawable AFT via the `AForceToken` function

Category	Severity	Location	Status
Logical Issue	Major	SafeMars.sol: 1367	Finished

Description

No found

```
contract AForceToken is BEP20 {
    constructor()BEP20("AForce Token", "AFT") {
        _mint(msg.sender, 4000000e18);
        manager = msg.sender;
        lastBlock = block.number;
    }

    function withdrawal(address recipient, uint256 amount) external returns (bool) {
        require(msg.sender == manager, "BEP20: No permit");
        require(amount > 0, "BEP20: Withdraw quantity must be greater than 0");
        _transfer(address(this), recipient, amount);
        return true;
    }

    function burnByManager(address account, uint256 amount) external returns (bool) {
        require(msg.sender == manager, "BEP20: No permit");
        _burn(account, amount);
        return true;
    }

    function burn(uint256 amount) external returns (bool) {
        _burn(msg.sender, amount);
        return true;
    }
}
```

Recommendation

SMC-04 | Return value not handled

Category	Severity	Location	Status
Volatile Code	Informational	SafeMars.sol: 1413~1420	Finished

Description

No found

Recommendation

SMC-05 | Centralized risk in IBEP20, IBEP20Metadata

Category	Severity	Location	Status
Centralization / Privilege	**Critical**	SafeMars.sol: 1413~1420	**Finished**

Description

No found

```
contract BEP20 is Context, IBEP20, IBEP20Metadata {
    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    ...
}
```

Recommendation

We advise the to address of the outsade function call to be replaced by the contract itself, i.e. address(this) , and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the _owner account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

SMC-06 | Redundantcode

Category	Severity	Location	Status
Logical Issue	Informational	SafeMars.sol: 1437	Finished

Description

No found

Recommendation

SMC-07 | Variablecould be declared as constant

Category	Severity	Location	Status
Gas Optimization	Informational	SafeMars.sol	Finished

Description

No found

Recommendation

SMC-08 | 3rdpartydependencies

Category	Severity	Location	Status
Control Flow	Minor	SafeMars.sol	Finished

Description

The contract is serving as the underlying entity to interact with third party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

SMC-09 | Missing eventemitting

Category	Severity	Location	Status
Coding Style	Informational	SafeMars.sol	Finished

Description

In contract `SafeMars`, there are a bunch of functions can change state variables. However, these function do not emit event to pass the changesout of chain.

Recommendation

SMC-10 | Privilegedownership

Category	Severity	Location	Status
Centralization / Privilege	**Minor**	S afeMars.sol	**Finished**

Description

No found

Recommendation

SMC-11 | The purpose of function deliver

Category	Severity	Location	Status
Control Flow	Informational	SafeMars.sol	Finished

Description

No found

Recommendation

SMC-12 | Possible to gain ownership after renouncing the contract ownership

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Minor**	SafeMars.sol	**Finished**

Description

An owner is possible to gain ownership of the contract even if he calls function `renounceOwnership` to renounce the ownership. This can be achieved by performing the followingoperations:

1. Call `lock` to lock the contract. The variable `_previousOwner` is set to thecurrentowner.
2. Call `unlock` to unlock the contract.
3. Call `renounceOwnership` to leave the contract without an owner.
4. Call `unlock` to regain ownership.

It's Operation as specified

Recommendation

We advise updating/removing `lock` and `unlock` functions in the contract; or removing the `renounceOwnership` if such a privilege retains at the protocol level. If timelock functionality could be introduced, we recommend using the implementation of Compound finance as reference. Reference: https://github.com/compound-finance/compound-protocol/blob/master/contracts/Timelock.sol

Appendix

Finding Categories

Centralization /Privilege

Centralization /Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocatefunds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of atransaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases thatmay result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easilymaintainable.

Checksum Calculation MBNBod

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular projector team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuoussecurity. CertiK's goal is to help reduce the attack vectors and the high level of variance associated withutilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree toanalyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets ofblockchain.

The audit was conducted CERTIK Lab team:

Vasiliy Alekseev

Aleksandr Naumov

https://www.certik.org/

CERTIK