Security Assessment

multiple

Aug 2nd, 2021

Table of Contents

Summary

This report has been prepared for multiple 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 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.

Additionally, this audit is based on a premise that all external smart contracts are implemented safely.

The security assessment resulted in 10 findings that ranged from minor 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

Project Summary

Project Name	multiple
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/multiple-finance/multiple-core
Commit	ec3688f7097b672b0894ff7e4dbcacae70cfc85c

Audit Summary

Delivery Date	Aug 02, 2021
Audit Methodology	Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	0	0	0	0	0	0
Minor	3	0	0	2	1	0
Informational	7	0	0	6	1	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	file	SHA256 Checksum
PCK	contracts/core/base/Permission.sol	9385a36327bd1e9344682d595a4fa86c483953a842fb96f0b4d5bf9868a3cee8
ICC	contracts/core/interfaces/ICompoundCERC20.sol	5d55f0fc211e1a22e021f54dfdd0461c36c906deda9a17d82306ac265d459ea2
ICE	contracts/core/interfaces/ICompoundCETH.sol	7a58a2c343cd8dfb451941173628a53125ed970035b8f323321a39a0a3fc9a9e
IMB	contracts/core/interfaces/IMulBank.sol	7290d1b6c70495d56d49b297b9b195283f8c7310a75dbfc71f45df821766c82b
IMW	contracts/core/interfaces/IMulWork.sol	139839a0d60ec0e6d33f60edc41e57e463b06eb8c65b524ff61af73c1cc77588
IPC	contracts/core/interfaces/IPayCallback.sol	5b712e4779a36abc605646ed3440056c5f360781c97e9f5791f1fa256255e844
IUV	contracts/core/interfaces/IUniswapV3Strategy.sol	049f60bfaad6d0915b385610d4ee7de79081f68fc8e751a4c9d1588e47c3b1b1
MAC	contracts/core/MulAuction.sol	045399df4f2c07849df0858716623f7e96ce80fdf9669cf328b940b33e30e57c
MBC	contracts/core/MulBank.sol	fb50d9602e0fd2904c6bafe73e0930fbb7ba32ac7c32d2758926e5dbc9286ea9
MER	contracts/core/MulERC20.sol	1eeda17bb3241e04841a85d7594707edd945679e8e0b542ce34f7dbd2ce2706d
MIC	contracts/core/MulInvest.sol	e1f0627b683ed51f0f0ee953a76a7572f4b18f560e06b148abcca35235348783
MWC	contracts/core/MulWork.sol	97d8aa9364a0bbbd89280c7d1fc2fd04c62c4dc3fe18455f7bcc843494062b34
PCP	contracts/core/Pop721.sol	4a62867aa9690f9b69125e3089a18c90d38ffb5fbef9639c1d2923b8edc8c7c7
UVS	contracts/core/UniswapV3Strategy.sol	5869df6bafa084affafc2819e13c9fbae6a656c5a35933e750f5f6cb62b09c15
THC	contracts/libraries/TransferHelper.sol	0571a5bc25e32f41bf8fe36cdebfdd94c57794dd74274cf7006ea9313525c455
UCK	contracts/libraries/Upgradable.sol	c11db30b13ef7503b8876eb607215bec7ce85476b209308b9bd5337797287a80

ID	file	SHA256 Checksum
WLC	contracts/libraries/WhiteList.sol	df8cb96ce83652ed0d6ef3b138b7a83f1e86b19a16d499184ceb7d325c9409d3

Privileged Functions

The project contains the following privileged functions that are restricted by the `onlyOwner` modifier.They are used to modify the contract configurations and address attributes. We grouped these functions below:

MulAuction

- function create(uint startTime, uint maxTime, uint plusTime, uint basePrice, uint addPrice, uint tokenId)
- function stop(uint pid)

MulBank

- function initPool(ERC20 supplyToken)

MulWork

- function setBaseQuota(address[] calldata tokens, uint[] memory amounts)
- function upgrade(address newContract, uint[] memory tokenIds)

Pop721

- function mint(address to, uint256 tokenId)

MulERC20

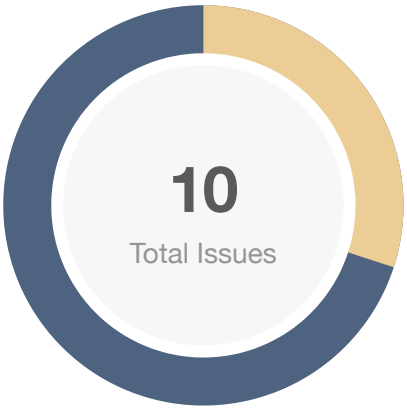
- function mint(address _to, uint256 _amount)
- function setDecimal(uint8 decimal)
- function burn(address _to, uint256 _amount)

Permission

- function addPermission(address _addPermission)
- function delPermission(address _delPermission)
- function getPermission(uint256 _index)

The dev team plans to hand over the `owner` to community governance or `TimeLock` in the future.

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**3** (30.00%)
Informational	**7** (70.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
MAC-01	Recommended Explicit Pool Validity Checks	Logical Issue	Informational	Resolved
MAC-02	Uninitialized State Variables	Coding Style	Informational	Resolved
MAC-03	Lack of Input Validation	Logical Issue	Informational	Resolved
MBC-01	Incorrect `shareToken` amount in Function `deposit()` and `withdraw()`	Logical Issue	Minor	Resolved
MBC-02	Duplicate Code	Coding Style	Informational	Resolved
MWC-01	Privileged Ownership	**Centralization / Privilege**	**Minor**	**Acknowledged**
MWC-02	Lack of Input Validation	Logical Issue	Informational	Resolved
MWC-03	`GPToken` in Contract `MulWork`	Logical Issue	Informational	Acknowledged
UVS-01	Lack of Input Validation	Logical Issue	Informational	Resolved
UVS-02	Strengthen Transfer Security	Logical Issue	Minor	Resolved

MAC-01 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	contracts/core/MulAuction.sol: 60, 78, 92	Resolved

Description

There's no sanity check to validate if a pool is existing.

Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `bid()`, `claim()` and `stop()`.

```
1 modifier validatePoolByPid(uint256 _pid) {
2     require (_pid < poolInfo.length , "Pool does not exist") ;
3     _;
4 }
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

MAC-02 | Uninitialized State Variables

Category	Severity	Location	Status
Coding Style	Informational	contracts/core/MulAuction.sol: 31~32	Resolved

Description

`cntOfPool` is uninitialized, but this variable is used in event `Create()`.

Recommendation

Consider making this variable increment in function `create()`.

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

MAC-03 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	contracts/core/MulAuction.sol: 35~36	Resolved

Description

Addresses should be checked before assigning to make sure they are not zero addresses.

Recommendation

Consider adding validation like bellow:

```
    constructor(IERC721 _GPToken, IERC20 _MulToken) {
        require(address(_GPToken) != address(0), "INVALID_ADDRESS");
        require(address(_MulToken) != address(0), "INVALID_ADDRESS");
        GPToken = _GPToken;
        MulToken = _MulToken;
    }
   constructor(IUniswapV3Factory _factory, IMulWork _work, IMulBank _bank, address
_rewardAddr) {
        require(address(_factory) != address(0), "INVALID_ADDRESS");
        require(address(_work) != address(0), "INVALID_ADDRESS");
        require(address(_bank) != address(0), "INVALID_ADDRESS");
        require(_rewardAddr != address(0), "INVALID_ADDRESS");
        factory = _factory;
        work = _work;
        bank = _bank;
        rewardAddr = _rewardAddr;
    }

    constructor(IERC721 _gpToken, IMulBank _bank) {
        require(address(_gpToken) != address(0), "INVALID_ADDRESS");
        require(address(_bank) != address(0), "INVALID_ADDRESS");
        GPToken = _gpToken;
        bank = _bank;
    }
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

MBC-01 | Incorrect `shareToken` amount in Function `deposit()` and `withdraw()`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/core/MulBank.sol: 169, 186	Resolved

Description

The amount of `shareToken` in the function `deposit()` and `withdraw()` is incorrect. It should be `share` instead of `amount`.

Recommendation

Consider changing to the correct logic like below:

```
    function deposit(address token, uint256 amount) external {
        ...
        uint totalShare = getTotalShare(address(pool.supplyToken));
        uint share = totalShare == 0 ? amount:
amount.mul(pool.shareToken.totalSupply()).div(totalShare);
        pool.shareToken.mint(msg.sender, share);
        ...
    }
    function withdraw(address token, uint256 share) external {
        ...
        uint amount = share.mul(totalShare).div(pool.shareToken.totalSupply());
        require(pool.supplyToken.balanceOf(address(this)) >= amount, "NO ENOUGH AMOUNT");

        pool.shareToken.burn(msg.sender, share);
        ...
    }
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

MBC-02 | Duplicate Code

Category	Severity	Location	Status
Coding Style	Informational	contracts/core/MulBank.sol: 11	Resolved

Description

`MulERC20.sol` is already imported in line 8, there is no need to import again.

Recommendation

Consider commenting the line 11 like below:

```
//import "./MulERC20.sol";
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

MWC-01 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Minor	contracts/core/MulWork.sol: 68	Acknowledged

Description

The owner of contract `MulWork` has the permission to transfer all `tokenIds` to another contract.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

The development team responded that they will change this part of the logic in the next version.

MWC-02 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	contracts/core/MulWork.sol: 41~42	Resolved

Description

Addresses should be checked before assigning to make sure they are not zero addresses.

Recommendation

Consider adding validation like bellow:

```
    constructor(IERC721 _GPToken, IERC20 _MulToken) {
        require(address(_GPToken) != address(0), "INVALID_ADDRESS");
        require(address(_MulToken) != address(0), "INVALID_ADDRESS");
        GPToken = _GPToken;
        MulToken = _MulToken;
    }
   constructor(IUniswapV3Factory _factory, IMulWork _work, IMulBank _bank, address
_rewardAddr) {
        require(address(_factory) != address(0), "INVALID_ADDRESS");
        require(address(_work) != address(0), "INVALID_ADDRESS");
        require(address(_bank) != address(0), "INVALID_ADDRESS");
        require(_rewardAddr != address(0), "INVALID_ADDRESS");
        factory = _factory;
        work = _work;
        bank = _bank;
        rewardAddr = _rewardAddr;
    }

    constructor(IERC721 _gpToken, IMulBank _bank) {
        require(address(_gpToken) != address(0), "INVALID_ADDRESS");
        require(address(_bank) != address(0), "INVALID_ADDRESS");
        GPToken = _gpToken;
        bank = _bank;
    }
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

MWC-03 | `GPToken` in Contract `MulWork`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/core/MulWork.sol	Acknowledged

Description

In this contract, user paid `GPToken` to create a worker account using the function `createAccount()`. But currently, user cannot redeem `GPToken`. Could you please tell us more detail about this?

Alleviation

The development team responded that `GPToken` is not redeemable, and redemption will not be provided in the later period. This is equivalent to proof of work. If the performance is good, the `GPToken` will be returned in other ways in the later period.

UVS-01 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	contracts/core/UniswapV3Strategy.sol: 65~68	Resolved

Description

Addresses should be checked before assigning to make sure they are not zero addresses.

Recommendation

Consider adding validation like bellow:

```
    constructor(IERC721 _GPToken, IERC20 _MulToken) {
        require(address(_GPToken) != address(0), "INVALID_ADDRESS");
        require(address(_MulToken) != address(0), "INVALID_ADDRESS");
        GPToken = _GPToken;
        MulToken = _MulToken;
    }
   constructor(IUniswapV3Factory _factory, IMulWork _work, IMulBank _bank, address
_rewardAddr) {
        require(address(_factory) != address(0), "INVALID_ADDRESS");
        require(address(_work) != address(0), "INVALID_ADDRESS");
        require(address(_bank) != address(0), "INVALID_ADDRESS");
        require(_rewardAddr != address(0), "INVALID_ADDRESS");
        factory = _factory;
        work = _work;
        bank = _bank;
        rewardAddr = _rewardAddr;
    }

    constructor(IERC721 _gpToken, IMulBank _bank) {
        require(address(_gpToken) != address(0), "INVALID_ADDRESS");
        require(address(_bank) != address(0), "INVALID_ADDRESS");
        GPToken = _gpToken;
        bank = _bank;
    }
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

UVS-02 | Strengthen Transfer Security

Category	Severity	Location	Status
Logical Issue	Minor	contracts/core/UniswapV3Strategy.sol: 128, 318	Resolved

Description

There are a lot of transfer operations in functions `_settle()` and `distributeFee()`, add a reentrant would be safer.

Recommendation

Consider adding a modifier as below:

```
bool private _status;
modifier nonReentrant() {
    require(!_status, 'reentrant call');
    _status = true;
    _;
   _status = false;
}
```

Alleviation

The development team resolved this issue in commit `d7a8bc27b0d70240fd6f50286c6a04d0f90fb410`.

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 relocate funds.

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.

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, easily maintainable.

Checksum Calculation Method

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 project or 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 continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

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 of blockchain.