CERTIK

Security Assessment

TAP FANTASY

Dec 27th, 2021

Table of Contents

Summary

This report has been prepared for TAP FANTASY to discover issues and vulnerabilities in the source code of the TAP FANTASY project 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.

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

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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	TAP FANTASY
Platform	BSC
Language	Solidity
Codebase	https://github.com/tapfantasy/fantasyprotocol
Commit	613b3ef0e99445b22f6deaea15cccf103c142e90 9ca4873237925545b32da11b3daf313cc9b62daa

Audit Summary

Delivery Date	Dec 27, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	3	0	0
Medium	0	0	0	0	0	0
Minor	4	0	0	0	0	4
Informational	12	0	0	0	2	10
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
OCK	contracts/access/Operators.sol	b8d6d33bf9dc95da4a5fb6029d3b9dc5741a347109e01092b0c8eab3a3bb8b11
OUC	contracts/access/OperatorsUpgradeable.sol	4bdc0bd0f4970a54e35315101aacc80c5aa2a78e306e5b2a76ceefb4ea06e7f9
ICC	contracts/interfaces/ICompCToken.sol	d98a65343358c38956d4fef4e48ca408bad4a94c3f70f983957b1cb953a155aa
IGT	contracts/interfaces/IGoldToken.sol	860a806583f0b7d0639759d061fe3538b9318e7ff93f7fd644b42aa6272b8e18
IGC	contracts/interfaces/IGoldTreasury.sol	66333b38b7f0888fb082de7fa74d591ec8538394666e50b6cafceb314af277ea
INF	contracts/interfaces/INFTAssets.sol	1bb0f72bc0df5212b62f5e3c52099ac3e739b615dcaf8783153cc25b672f30b4
INI	contracts/interfaces/INFTInGame.sol	d1c19acff4c33f5e3e7138f8b631309e99d7af548db3906925191786fbdd371d
INM	contracts/interfaces/INFTMarket.sol	9622c43dcf37422a97e99902b62d0d4523761cfce55810f4c2ba605f78ac2246
INS	contracts/interfaces/INFTShop.sol	d877dcc3ffd86a20c3662deb4d0ab4de320b94433a0845ff76009a8ba5f2bbf0
IOC	contracts/interfaces/IOperators.sol	191cc75e12965a051beb385530556108b6552eb996f4011b336680eb5aac078f
ITI	contracts/interfaces/ITokenInGame.sol	b7ac81f2d555064bf1454b1885e267d80e13401d4c8ffd5391f082f348c413c1
ITL	contracts/interfaces/ITokenLocker.sol	eb00743637da138a277372df51e4255be3cc12075ac78da93cf1d91b668d9a9c
IWE	contracts/interfaces/IWETH.sol	ac6a54f366be1a5ac49eb2d929951419882f22867acf3a9909d60fcea6689b30
NFA	contracts/nftassets/NFTAssets.sol	94f1a59a765eb4d3967a9ce6d4b705505922180c4d1b4af46599b971832690d7

ID	File	SHA256 Checksum
NFI	contracts/nftassets/NFTInGame.sol	90fc950e130b83c0c2a9f49e20584919084e158f12231b95d93313583126eeb4
NFM	contracts/nftassets/NFTMarket.sol	61bc356ab074d3b7b9921a51d866f60cb237a549f5c80fad59883f26b3fd7d61
NFB	contracts/nftassets/NFTMysteryBox.sol	9c18c4a03974b21bc5b079542727816ad8fc7bfa6158ce6dcf377ba228b959f1
NFS	contracts/nftassets/NFTShop.sol	9ef72f2bbada53d2563b27afc2739480b00f807ec65f3bcb0c130b6dcaa5acc6
GTC	contracts/tokens/GoldToken.sol	ebfd4e70909f959c440da3b0eb3373599f9a1201a40e602ce25fadd67a5dca34
GTK	contracts/tokens/GoldTreasury.sol	aa38ca089be3233fe69f2900aae5d47da7e072d242a3268b33b06be877ae8742
MCT	contracts/tokens/MCToken.sol	c4bfe067dbeb32071eec7e2efacb529fc189e5c59d96cd9baa5338bc1b3191d4
TTC	contracts/tokens/TapToken.sol	e6581471d22f63528b27c889e11653bf60c905d47f0462ca9eff5562cf01de69
TIG	contracts/tokens/TokenInGame.sol	69acd75ce4f6891a0b6ec145dd6adf556549fe08c839b2469e7a7d7da965ad78
TLC	contracts/tokens/TokenLocker.sol	534c1c43117de05a976c57d5ee9799bad3130dbcd690811e444a767dfb952138
BMC	contracts/utils/BatchMint.sol	453269363dad84cf57384d29bfa533b1aed8f453c287f752238c31eb0bb7897a
PAC	contracts/utils/ProxyAdmin.sol	0ce4b0207a858559a7cdd12a38cdbf1167ab7944d7770c560be6801c5260ddce
TUP	contracts/utils/TransparentUpgradeableProxy.sol	ea16e0639ddf1e0ed1dc5ca3f3d11b60c9922b05e3a5c8ca3095326108e92e8e
TDC	contracts/TapDesktop.sol	4554943860c0e70c8b9703cdc318847029b7534c354c3bcbbbec2f0968bee65f

Understandings

Overview

Privileged Functions

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

The `onlyOwner` modifier:

Contract `Operators`:

- setOper(address _a, bool _b)

Contract `OperatorsUpgradeable`:

- setOper(address _a, bool _b)

Contract `NFTAssets`:

- setBaseURI(string memory _baseUri)
- pause()
- unpause()

Contract `NFTInGame`:

- setBlacklist(uint[] memory _idlist, bool _enable)
- setCoolTime(uint _cooltime)
- setFeeGather(address payable _feeGather, uint _feeAmount)

Contract `NFTMarket`:

- setFeeGather(address _feegather)
- addGoods(address _payToken, address _itemNFT, address _itemToken, uint _feeRate)
- setGoods(uint _goodsid, uint _feeRate)
- setGoodsAvailable(uint _goodsid, bool _available)

Contract `NFTMysteryBox`:

- setFeeGather(address _feegather)
- setTokens(address _payToken, address _obtainToken)

Contract `NFTShop`:

- setFeeGather(address _feegather)
- setTokens(address _payToken, address _obtainToken)

Contract `GoldTreasury`:

- setFeeGather(address _feegather)
- setFeeRate(uint _feerate)
- setInvest(address _invest)
- setTokens(address _tokenCash, address _tokenGold)

Contract `TokenLocker`:

- setLockTimeRate(uint _feeLockTime, uint _feeRate)

Contract `BatchMint`:

- mint(address _token, uint[] memory _itemId, address _to)

Contract `ProxyAdmin`:

- changeProxyAdmin(TransparentUpgradeableProxy proxy, address newAdmin)
- upgrade(TransparentUpgradeableProxy proxy, address implementation)
- upgradeAndCall(TransparentUpgradeableProxy proxy, address implementation, bytes memory data)

The `onlyOper` modifier:

Contract `NFTAssets`:

- safeMint(address to, uint256 tokenId)

Contract `NFTInGame`:

- unlockFromGame(uint _tokenId, address _user)
- unlockFromGameBatch(uint[] memory _tokenId, address[] memory _user)
- resetCoolTime(uint _tokenId)
- nftLevelUpBatch(uint[] memory _tokenId, uint[] memory _tokenNewId)
- nftLevelUp(uint _tokenId, uint _tokenNewId)

Contract `NFTMarket`:

- onCancel(uint _orderid, address _user)

Contract `NFTMarket`:

- addItem(uint typeid, uint price, uint stock)
- enableItem(uint itemId, bool open)
- onObtain(uint flowid, address _user, uint256 _tokenId)

Contract `NFTShop`:

- addItem(uint typeid, uint price, uint stock)
- enableItem(uint itemId, bool open)
- onObtain(uint flowid, address _user, uint256 _tokenId)

Contract `GoldToken`:

- mint(uint value)
- burn(uint value)

Contract `GoldTreasury`:

- mint(uint _value, address _to)
- burn(uint _value, address _to)
- claimRewards(address _user)
- claim(address _user)

Contract `MCToken`:

- mint(uint value)
- burn(uint value)

Contract `TapToken`:

- mint(uint value)
- burn(uint value)

Contract `TokenInGame`:

- setToken(address _token, bool _enable)
- gameOut(uint _serialid, address _token, address _user, uint _value)
- gameOutBatch(uint[] memory _serialid, address[] memory _token, address[] memory _user, uint[] memory _value)

Contract `TokenLocker`:

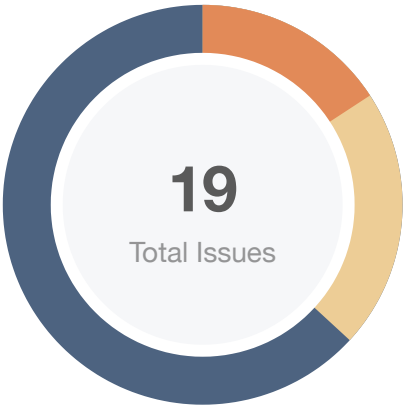
- claimBatch(uint[] memory lid, address _touser)
- gameOut(uint _serialid, address _user, uint _timestamp, uint _value)
- gameOutBatch(uint[] memory _serialid, address[] memory _user, uint[] memory _timestamp, uint[] memory _value)

The `ifAdmin` modifier:

Contract `TransparentUpgradeableProxy`:

- admin()
- implementation()
- changeAdmin(address newAdmin)
- upgradeTo(address newImplementation)
- upgradeToAndCall(address newImplementation, bytes calldata data)

Findings



Critical	**0** (0.00%)
Major	**3** (15.79%)
Medium	**0** (0.00%)
Minor	**4** (21.05%)
Informational	**12** (63.16%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
GLOBAL-02	Token Minted to Centralized Address	Logical Issue	Major	Acknowledged
GLOBAL-03	Function Visibility Optimization	Gas Optimization	Informational	Partially Resolved
GLOBAL-04	Missing Emit Events	Coding Style	Informational	Resolved
GTK-01	Incorrect Mint Token Amount	Logical Issue	Minor	Resolved
GTK-02	Incorrect Redeem Amount	Logical Issue	Minor	Resolved
GTK-03	Unused Variable	Coding Style	Informational	Resolved
GTK-04	Unfinished Function `claim()`	Logical Issue	Informational	Resolved
GTK-05	Missing Input Validation	Logical Issue	Informational	Resolved
GTK-06	Missing Input Validation	Logical Issue	Informational	Partially Resolved
NFB-01	Missing Input Validation	Logical Issue	Informational	Resolved
NFB-02	Central Server between Function `onBuy()` and `onObtain()`	Logical Issue	Major	Acknowledged
NFI-01	Missing Input Validation	Logical Issue	Informational	Resolved
NFM-01	Incorrect `tokenDecimals`	Logical Issue	Minor	Resolved
NFM-02	Incorrect Require Condition	Logical Issue	Minor	Resolved

ID	Title	Category	Severity	Status
NFM-03	Missing Input Validation	Logical Issue	Informational	Resolved
NFS-01	Missing Input Validation	Logical Issue	Informational	Resolved
TIG-01	Missing `array` Length Check	Logical Issue	Informational	Resolved
TLC-01	Missing `array` Length Check	Logical Issue	Informational	Resolved

GLOBAL-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

Description

In the contract `Operators`, the role `owner` has the authority over the following function:

- setOper()

In the contract `OperatorsUpgradeable`, the role `owner` has the authority over the following function:

- setOper()

In the contract `NFTAssets`, the role `owner/oper` has the authority over the following function:

- setBaseURI()
- pause()
- unpause()
- safeMint()

In the contract `NFTInGame`, the role `owner/oper` has the authority over the following function:

- setBlacklist()
- setCoolTime()
- setFeeGather()
- unlockFromGame()
- unlockFromGameBatch()
- resetCoolTime()
- nftLevelUpBatch()
- nftLevelUp()

In the contract `NFTMarket`, the role `owner/oper` has the authority over the following function:

- setFeeGather()
- addGoods()
- setGoods()
- setGoodsAvailable()
- onCancel()

In the contract `NFTMysteryBox`, the role `owner/oper` has the authority over the following function:

- setFeeGather()
- setTokens()
- addItem()
- enableItem()
- onObtain()

In the contract `GoldToken`, the role `oper` has the authority over the following function:

- mint()
- burn()

In the contract `MCToken`, the role `oper` has the authority over the following function:

- mint()
- burn()

In the contract `TapToken`, the role `oper` has the authority over the following function:

- mint()
- burn()

In the contract `BatchMint`, the role `owner` has the authority over the following function:

- mint()

In the contract `GoldTreasury`, the role `owner/oper` has the authority over the following function:

- setFeeGather()
- setFeeRate()
- setInvest()
- setTokens()
- mint()
- burn()
- claimRewards()
- claim()

In the contract `ProxyAdmin`, the role `owner` has the authority over the following function:

- changeProxyAdmin()
- upgrade()

- upgradeAndCall()

In the contract `TransparentUpgradeableProxy`, the role `admin` has the authority over the following function:

- admin()
- implementation()
- changeAdmin()
- upgradeTo()
- upgradeToAndCall()

In the contract `TokenInGame`, the role `owner/oper` has the authority over the following function:

- setToken()
- gameOut()
- gameOutBatch()

In the contract `TokenLocker`, the role `owner/oper` has the authority over the following function:

- setLockTimeRate()
- claimBatch()
- gameOutBatch()

Any compromise to these accounts may allow the hacker to manipulate the project through these functions.

Recommendation

We advise the client to carefully manage the `owner/oper/admin` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at the different levels in terms of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Team]:

1. `owner` will hand it over to Time-lock for management.
2. For different business contracts, `oper` will hand over to different operation management wallets, corresponding to different centralized business wallet operations

GLOBAL-02 | Token Minted to Centralized Address

Category	Severity	Location	Status
Logical Issue	Major	Global	Acknowledged

Description

In the following contracts:

1. `NFTInGame`
2. `NFTMarket`
3. `NFTMysteryBox`
4. `NFTShop`
5. `GoldTreasury`

the above contracts all have the behavior of transferring to the `feegather` address, which can be set by the owner role.

In the following contracts:

1. `GoldToken`
2. `MCToken`
3. `TapToken`

the above contracts all have the behavior of minting tokens to the operators‘ address and burning operators' tokens, these addresses can be set by the owner role.

Recommendation

We advise the client to carefully manage the `owner/feegather/operators` accounts' private keys and avoid any potential risks of being hacked. We also advise the client to adopt Multisig, Timelock, and/or DAO in the project to manage this specific account in this case.

Alleviation

[Team]:

1. `feegather` is a project wallet and will be used under supervision.
2. Other private key wallets will be kept and used by the project party in accordance with a high security level.

GLOBAL-03 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	Global	Partially Resolved

Description

The following functions are declared as `public` and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

In the contract `NFTAssets`:

- `setBaseURI()` in L23
- `pause()` in L28

In the contract `ProxyAdmin`:

- `changeProxyAdmin()` in L50
- `upgrade()` in L61
- `upgradeAndCall()` in L73

In the contract `TapDesktop`:

- `mysteryBoxOpen()` in L68
- `mysteryBoxOpenETH()` in L72

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GLOBAL-04 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	Global	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

contract `NFTInGame`

- `setBlacklist()`
- `setCoolTime()`
- `setFeeGather()`

contract `NFTMarket`

- `setFeeGather()`

contract `NFTMysteryBox`

- `setFeeGather()`
- `setTokens()`
- `enableItem()`

contract `NFTMysteryBox`

- `setFeeGather()`
- `setTokens()`
- `enableItem()`

contract `GoldTreasury`

- `setFeeGather()`
- `setFeeRate()`
- `setInvest()`
- `setTokens()`

- **contract** `TokenInGame`
- `setToken()`

Recommendation

We advise the client to add events for sensitive actions, and emit them in the function.

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GTK-01 | Incorrect Mint Token Amount

Category	Severity	Location	Status
Logical Issue	Minor	projects/fantasyprotocol/contracts/tokens/GoldTreasury.sol (9b340e4): 106	Resolved

Description

Missing multiply by `exchangeRate` when minting `tokenGold`.

Recommendation

We advise the client to correct the mint amount as below:

```
101  function claimRewards(address _user) external onlyOper nonReentrant returns (uint
value) {
102      uint amount = tokenCash.balanceOf(address(this));
103      uint total = tokenGold.totalSupply().div(exchangeRate);
104      value =
ICompCToken(invest).balanceOfUnderlying(address(this)).add(amount).sub(total);
105      if(value > 0) {
106          IGoldToken(address(tokenGold)).mint(value.mul(exchangeRate));
107          tokenGold.safeTransfer(_user, value.mul(exchangeRate));
108      }
109  }
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GTK-02 | Incorrect Redeem Amount

Category	Severity	Location	Status
Logical Issue	Minor	projects/fantasyprotocol/contracts/tokens/GoldTreasury.sol (9b340e4): 91	Resolved

Description

The redeeming underlying amount is not correct when the balance is not enough.

Recommendation

We advise the client to change the amount as below:

```
  91              uint code =
ICompCToken(invest).redeemUnderlying(cashAmount.sub(tokenCash.balanceOf(address(this))));
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GTK-03 | Unused Variable

Category	Severity	Location	Status
Coding Style	Informational	projects/fantasyprotocol/contracts/tokens/GoldTreasury.sol (9b340e4): 29~33, 34	Resolved

Description

The variables `LockItem` and `unlock` are declared but never used and updated.

Recommendation

We recommend removing the unused variable if it is not intended to be used.

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GTK-04 | Unfinished Function `claim()`

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/tokens/GoldTreasury.sol (9b340e4): 111	Resolved

Description

The code logic of the `claim()` function is incomplete.

Recommendation

We recommend to complete the `claim()` function before deploying the contract.

Alleviation

The development team removed this function in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GTK-05 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/tokens/GoldTreasury.sol (9b340e4): 50, 62	Resolved

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

setFeeGather():

```
require(_feeGather != address(0), "_feeGather can not be zero address.");
```

setTokens():

```
require(_tokenCash != address(0), "_tokenCash can not be zero address.");
require(_tokenGold != address(0), "_tokenGold can not be zero address.");
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

GTK-06 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/tokens/GoldTreasury.sol (9b340e4): 47, 54	Partially Resolved

Description

There is no validation to check whether `_feerate` is less than `1e9`.

Recommendation

We advise the client to add validation as below:

```
    require(_feeRate <= 1e9, 'rate!');
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

NFB-01 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/nftassets/NFTMysteryBox.sol (9b340e4): 37, 41	Resolved

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

setFeeGather():

```
require(_feeGather != address(0), "_feeGather can not be zero address.");
```

setTokens():

```
require(_payToken != address(0), "_payToken can not be zero address.");
require(_obtainToken != address(0), "_obtainToken can not be zero address.");
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

NFB-02 | Central Server between Function `onBuy()` and `onObtain()`

Category	Severity	Location	Status
Logical Issue	Major	projects/fantasyprotocol/contracts/nftassets/NFTMysteryBox.sol (9b340e4): 63, 70	Acknowledged

Description

From the code logic of `onBuy()` and `onObtain()`, we believe that the `onBuy()` function is the entry point for purchase, and the `onObtain()` function is responsible for sending the NFT to the user who purchased the NFT. But the `onBuy()` function does not record the address of the purchasing user. How to ensure that the NFT can be sent to the orderer in the `onObtain()` function?

Alleviation

[Team]: When calling `onBuy()`, it triggers an event. The central server detects this event, the NFT will be sent to the user by `onObtain()` after business inspection is successful.

NFI-01 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/nftassets/NFTInGame.sol (9b340e4): 48	Resolved

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:
setFeeGather():

```
require(_feeGather != address(0), "_feeGather can not be zero address.");
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

NFM-01 | Incorrect `tokenDecimals`

Category	Severity	Location	Status
Logical Issue	Minor	projects/fantasyprotocol/contracts/nftassets/NFTMarket.sol (9b340e4): 70	Resolved

Description

When the item is `NFT`, the `tokenDecimals` should be zero instead of 18.

Recommendation

We advise the client to change `tokenDecimals` to zero as below:

```
70  tokenDecimals = 0;
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

NFM-02 | Incorrect Require Condition

Category	Severity	Location	Status
Logical Issue	Minor	projects/fantasyprotocol/contracts/nftassets/NFTMarket.sol (9b340e4): 90	Resolved

Description

The `require` condition is not correct, it should use `_feeRate <= 1e9`.

Recommendation

We recommended changing the require condition like below:

```
90  require(_feeRate <= 1e9, 'rate!');
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

NFM-03 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/nftassets/NFTMarket.sol (9b340e4): 36	Resolved

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:
setFeeGather():

```
require(_feeGather != address(0), "_feeGather can not be zero address.");
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

NFS-01 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/nftassets/NFTShop.sol (9b340e4): 38, 42	Resolved

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

setFeeGather():

```
require(_feeGather != address(0), "_feeGather can not be zero address.");
```

setTokens():

```
require(_payToken != address(0), "_payToken can not be zero address.");
require(_obtainToken != address(0), "_obtainToken can not be zero address.");
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

TIG-01 | Missing `array` Length Check

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/tokens/TokenInGame.sol (9b340e4): 52	Resolved

Description

The `array` arguments can be of different lengths.

Recommendation

We advise to `require` statements, ensuring that the input arrays `_user` and `_token` are of the same length as below:

```
function gameOutBatch(uint[] memory _serialid, address[] memory _token, address[] memory
_user, uint[] memory _value) external override onlyOper {
    require(_serialid.length == _token.length, 'length1!');
    require(_user.length == _value.length, 'length2!');
    require(_user.length == _token.length, 'length3!');
    for(uint i = 0; i < _serialid.length; i ++) {
        gameOut(_serialid[i], _token[i], _user[i], _value[i]);
    }
}
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

TLC-01 | Missing `array` Length Check

Category	Severity	Location	Status
Logical Issue	Informational	projects/fantasyprotocol/contracts/tokens/TokenLocker.sol (9b340e4): 82	Resolved

Description

The `array` arguments can be of different lengths.

Recommendation

We advise to `require` statements, ensuring that the input arrays `_user` and `_timestamp` are of the same length as below:

```
function gameOutBatch(uint[] memory _serialid, address[] memory _user, uint[] memory
_timestamp, uint[] memory _value) external override onlyOper {
    require(_serialid.length == _user.length, 'length1!');
    require(_timestamp.length == _value.length, 'length2!');
    require(_user.length == _timestamp.length, 'length3!');
    for(uint i = 0; i < _serialid.length; i ++) {
        gameOut(_serialid[i], _user[i], _timestamp[i], _value[i]);
    }
}
```

Alleviation

The development team resolved this issue in commit `9ca4873237925545b32da11b3daf313cc9b62daa`.

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.

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 a transaction.

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.