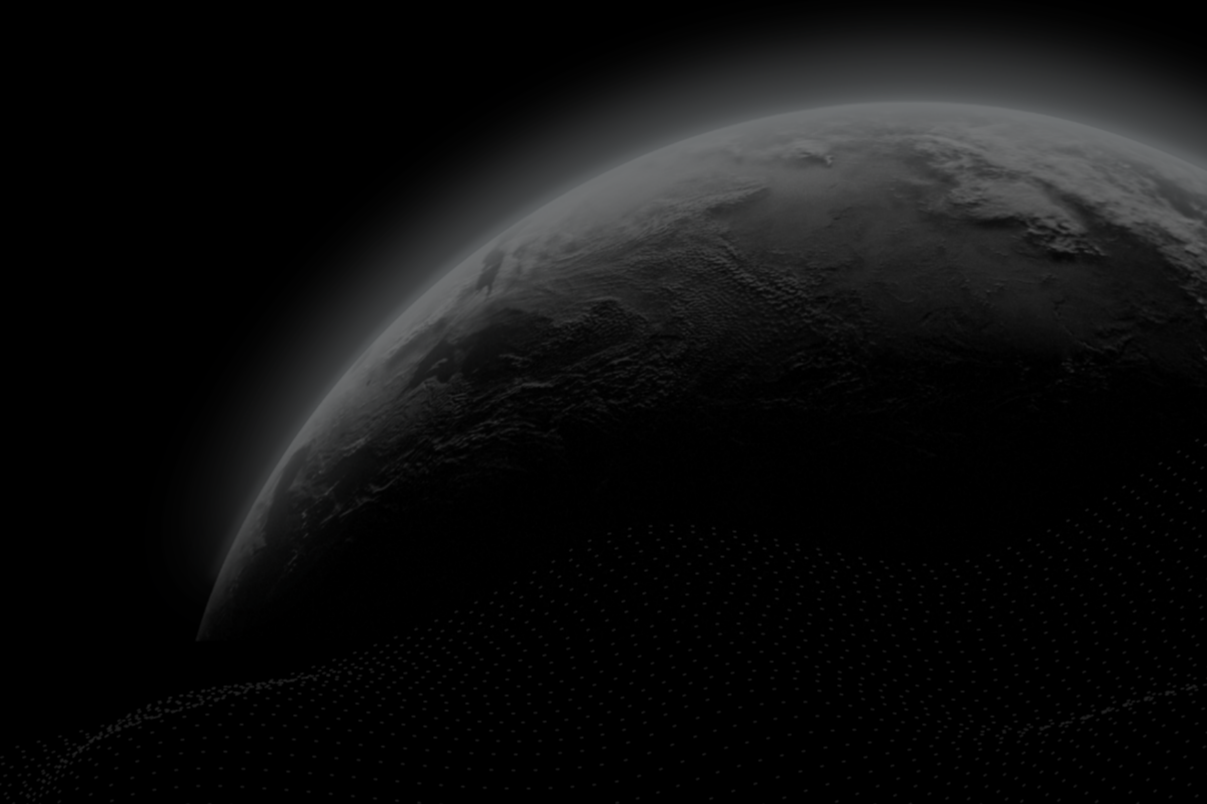
Security Assessment

Empire v3

CertiK Verified on Mar 1st, 2023



CertiK Verified on Mar 1st, 2023

Empire v3

The security assessment was prepared by CertiK, the leader in Web3.0 security.

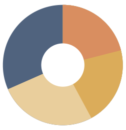
Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	BSC | Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 03/01/2023	N/A

CODEBASE	COMMITS
https://bscscan.com/address/0x51A183d8D79df6892Ab7b8f57b33ba70599515d4#code https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80 ...View All	Vault V2, EmpireToken V3, Bridge ...View All

Vulnerability Summary



19	5	0	0	14	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥 0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧 4	Major	4 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨 4	Medium	1 Resolved, 3 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 5	Minor	1 Resolved, 4 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦 6	Informational	3 Resolved, 3 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | EMPIRE V3

Summary

Findings

Optimizations

Appendix

Disclaimer

CODEBASE | EMPIRE V3

Repository

https://bscscan.com/address/0x51A183d8D79df6892Ab7b8f57b33ba70599515d4#code

https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code

https://bscscan.com/address/0x62AB5437563fC655226239cA8146F727E1D28BF4#code

Commit

Vault V2,

EmpireToken V3,

Bridge

AUDIT SCOPE | EMPIRE V3

11 files audited ● 2 files with Acknowledged findings ● 9 files without findings

ID	File	SHA256 Checksum
● ETC	contracts/EmpireToken.sol	0afafe3af46bedbc17b05ee59d2b8c73cfcc2ac2d69f5be05f7ed77aeaa55cf7
● BRI	contracts/Bridge.sol	96b045e566392d28d3fc4400247daea58da4bb707cdbe7a8fbee6a52f9017659
● OWN	@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
● IER	@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
● SMC	@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca0dfe5ec14916369a2935d888ff257a
● COE	@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
● EBV	EmpireBridgeVault.sol	0b21c173e384196a288398d707677a2ee0f4b68988699b0d1ff9ac88bcd9fe1c
● OWA	@openzeppelin/contracts/access/Ownable.sol	75e3c97011e75627ffb36f4a2799a4e887e1a3e27ed427490e82d7b6f51cc5c9
● PAU	@openzeppelin/contracts/security/Pausable.sol	5b6abc290190f46b9941c674594eee083a3fe6b92d1828d0cfefacc94d1cac9a
● RGC	@openzeppelin/contracts/security/ReentrancyGuard.sol	aa73590d5265031c5bb64b5c0e7f84c44cf5f8539e6d8606b763adac784e8b2e
● COU	@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a

APPROACH & METHODS | EMPIRE V3

This report has been prepared for Empire v3 to discover issues and vulnerabilities in the source code of the Empire v3 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

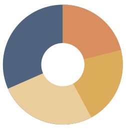
The auditing process pays special attention to the following considerations:

- 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:

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

FINDINGS | EMPIRE V3



19	0	4	4	5	6
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Empire v3. Through this audit, we have uncovered 19 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
BRI-01	**Centralization Risks In Bridge.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**
BRI-02	Cross Chain Swap Dependencies	Logical Issue	Major	● Acknowledged
ETC-01	**Centralized Risk In `addLiquidity`**	**Centralization / Privilege**	**Major**	● **Acknowledged**
ETC-02	**Centralization Risks In EmpireToken.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**
BRI-03	Ineffective `isContract()` Check	Volatile Code	Medium	● Acknowledged
ETC-03	Pancake Pair Should Be Excluded From Rewards	Logical Issue	Medium	● Acknowledged
ETC-14	Variable `_rOwned[account]` Not Updated In Function `includeInReward()`	Logical Issue	Medium	● Acknowledged
MAI-01	Lack Of Reasonable Boundary	Logical Issue	Medium	● Resolved
ETC-04	Need Max Transaction Check	Logical Issue	Minor	● Acknowledged
ETC-05	Proper Usage Of “Pure” And “View”	Coding Style	Minor	● Resolved

ID	Title	Category	Severity	Status
ETC-06	Potential Sandwich Attacks	Logical Issue	Minor	● Acknowledged
ETC-07	Third Party Dependencies	Volatile Code	Minor	● Acknowledged
ETC-08	Unused Return Value	Volatile Code	Minor	● Acknowledged
BRI-04	Missing Error Messages	Coding Style	Informational	● Acknowledged
ETC-09	The Purpose Of Function `deliver`	Control Flow	Informational	● Acknowledged
ETC-10	Typos In The Contract	Coding Style	Informational	● Resolved
ETC-11	Redundant SafeMath Usage	Language Specific	Informational	● Acknowledged
ETC-12	Unused Event	Coding Style	Informational	● Resolved
GLOBAL-01	Unlocked Compiler Version	Language Specific	Informational	● Resolved

BRI-01 | CENTRALIZATION RISKS IN BRIDGE.SOL

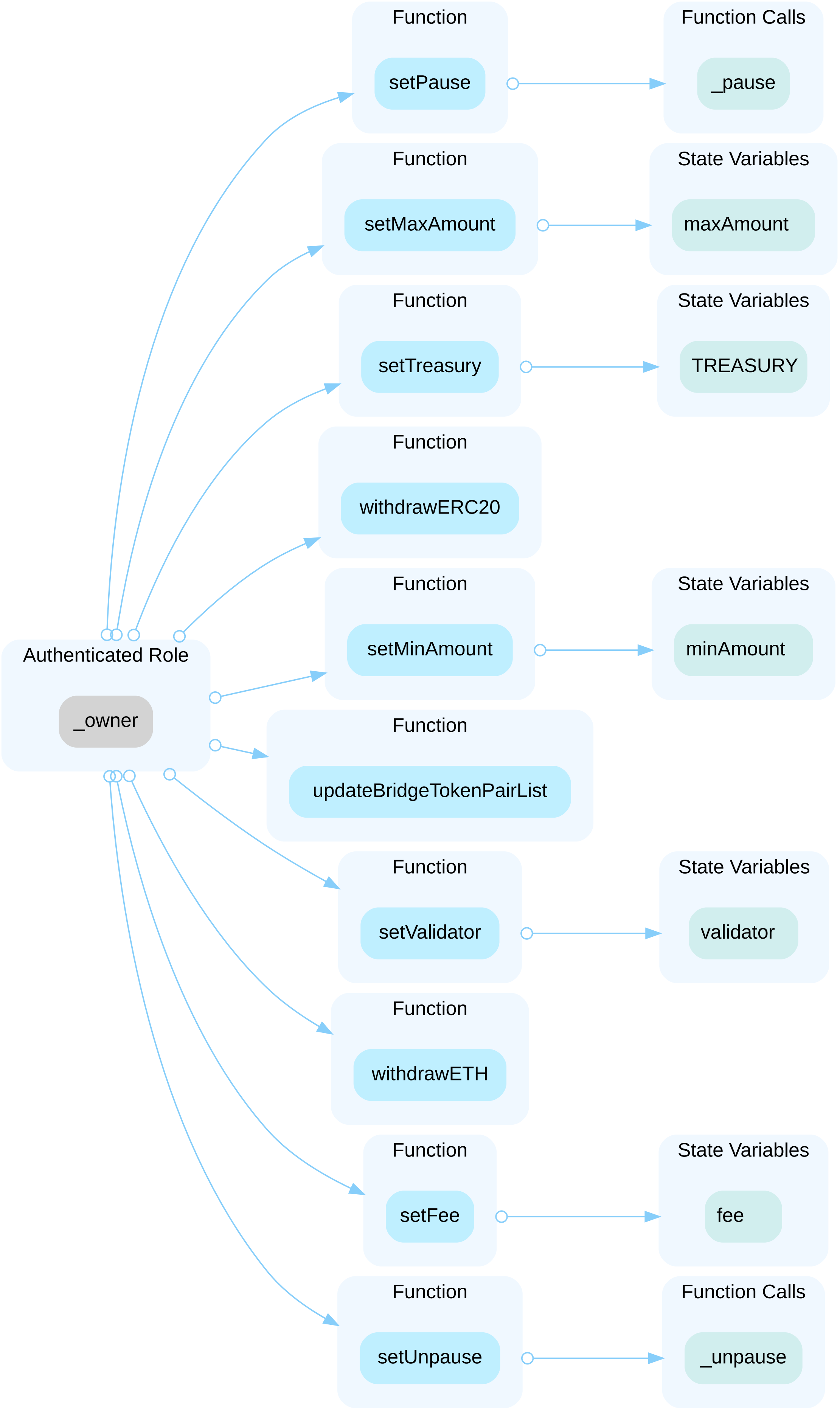
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/Bridge.sol (Bridge): 122~143, 176, 184, 192, 202, 206, 210, 217, 224, 231, 244**	● **Acknowledged**

Description

In the contract `Bridge` the role `validator` has authority over the functions below:

- function `redeem()` : to transfer tokens to another chain. Any compromise to the `validator` account may allow the hacker to take advantage of this authority.

In the contract `Bridge` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.


Authenticated Role
_owner
Function
setPause
Function Calls
_pause
Function
setMaxAmount
State Variables
maxAmount
Function
setTreasury
State Variables
TREASURY
Function
withdrawERC20
Function
setMinAmount
State Variables
minAmount
Function
updateBridgeTokenPairList
Function
setValidator
State Variables
validator
Function
withdrawETH
Function
setFee
State Variables
fee
Function
setUnpause
Function Calls
_unpause

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol 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 a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

The team acknowledged this issue and stated that they will use timelock + multi-sig wallet in the future.

BRI-02 | CROSS CHAIN SWAP DEPENDENCIES

Category	Severity	Location	Status
Logical Issue	● Major	contracts/Bridge.sol (Bridge): 122~143	● Acknowledged

Description

The logic ensures the cross-chain transaction atomicity is not implemented in the contract. The `validator` could be a message server host by the owner or an intermediate 3rd party application, hence the parameters in the `redeem()` function could not be guaranteed to correspond to the `swap()` .

The scope of the audit treats the above-mentioned application entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised, which may lead to lost or stolen assets. Suppose the cross-chain transaction atomicity is not guaranteed properly. In that case, the user deposits tokens into the source chain, but not be able to redeem the correct token amount from the target chain.

Recommendation

We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and stated that they will constantly monitor the 3rd parties for security.

ETC-01 | CENTRALIZED RISK IN `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/EmpireToken.sol (EmpireToken V3): 806~816**	● **Acknowledged**

Description

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `liquidityWallet` for acquiring the generated LP tokens from the `EmpireToken-BNB` pool. As a result, over time the `liquidityWallet` address will accumulate a significant portion of LP tokens. If the `liquidityWallet` is an EOA (Externally Owned Account), mishandling its private key can have devastating consequences for the project.

```
    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            liquidityWallet,
            block.timestamp
        );
    }
```

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` 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 `liquidityWallet` 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.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness of 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

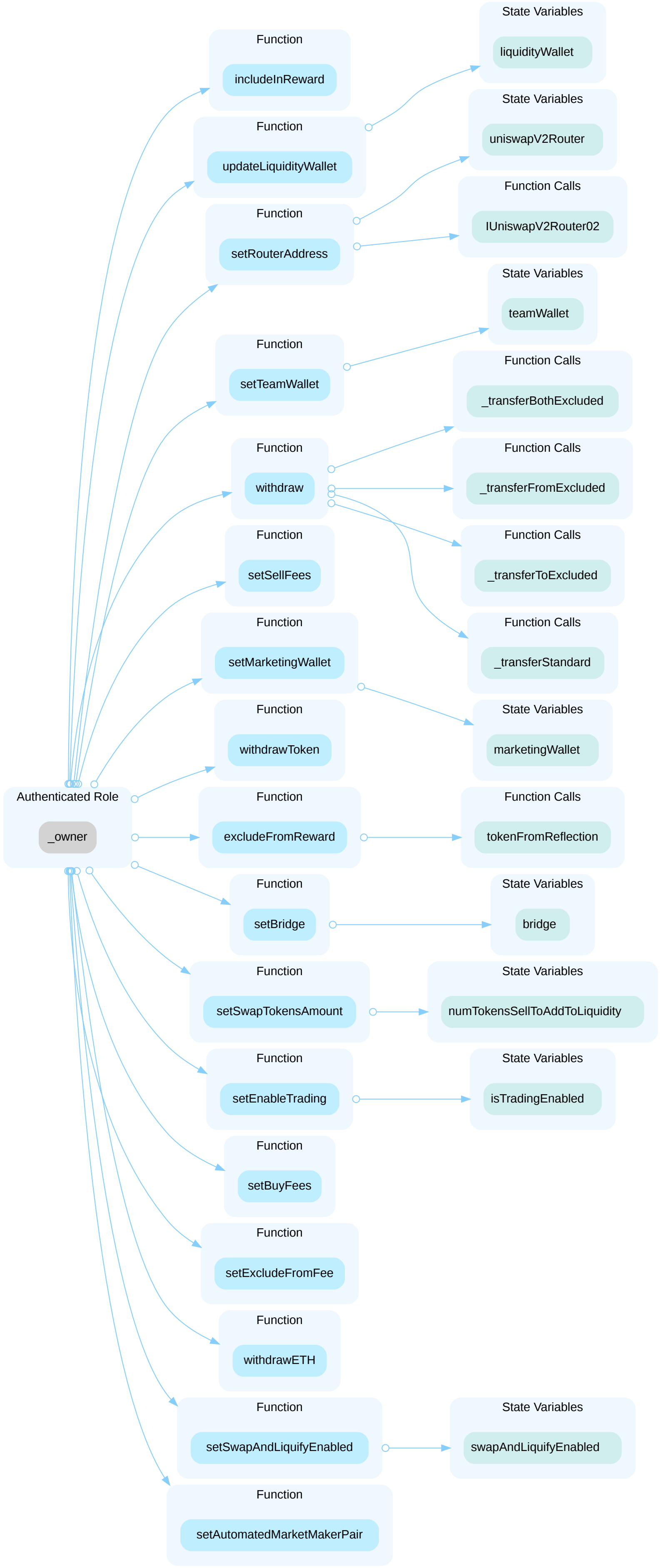
The team acknowledged this issue and stated that they will use multi-sig wallet in the future.

ETC-02 | CENTRALIZATION RISKS IN EMPIRETOKEN.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/EmpireToken.sol (EmpireToken V3): 232, 416, 427, 643, 960, 970, 977, 984, 1001, 1018, 1026, 1033, 1040, 1051, 1065, 1081, 1103, 1114, 1133; EmpireToken.sol (fix_ETH): 1083**	● **Acknowledged**

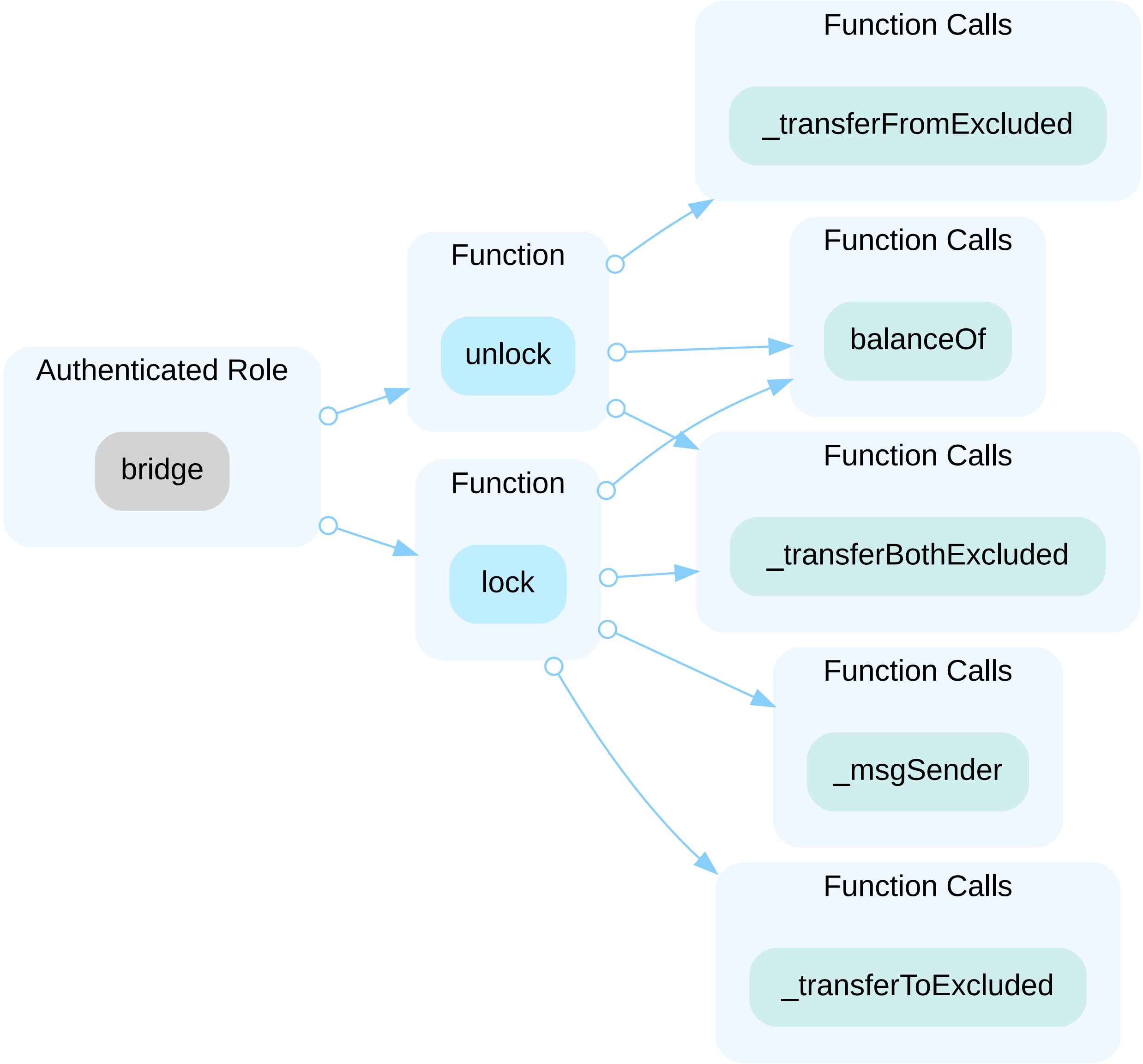
Description

In the contract `EmpireToken` the role `_owner` has authority over the functions shown in the diagram below.


Function
includeInReward
State Variables
liquidityWallet
Function
updateLiquidityWallet
State Variables
uniswapV2Router
Function Calls
IUniswapV2Router02
Function
setRouterAddress
State Variables
teamWallet
Function
setTeamWallet
Function Calls
_transferBothExcluded
Function
withdraw
Function Calls
_transferFromExcluded
Function
setSellFees
Function Calls
_transferToExcluded
Function
setMarketingWallet
Function Calls
_transferStandard
Function
withdrawToken
State Variables
marketingWallet
Authenticated Role
_owner
Function
excludeFromReward
Function Calls
tokenFromReflection
Function
setBridge
State Variables
bridge
Function
setSwapTokensAmount
State Variables
numTokensSellToAddToLiquidity
Function
setEnableTrading
State Variables
isTradingEnabled
Function
setBuyFees
Function
setExcludeFromFee
Function
withdrawETH
Function
setSwapAndLiquifyEnabled
State Variables
swapAndLiquifyEnabled
Function
setAutomatedMarketMakerPair

Besides, the role `_owner` also has authority over the function `setBridgeVault()` , which is used to set the `bridgeVault` by the owner. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.

In the contract `EmpireToken` the role `bridge` has authority over the functions shown in the diagram below.



Any compromise to the `bridge` account may allow the hacker to take advantage of this authority.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol 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 a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

The team acknowledged this issue and stated that they will use multi-sig wallet in the future.

BRI-03 | INEFFECTIVE `isContract()` CHECK

Category	Severity	Location	Status
Volatile Code	● Medium	contracts/Bridge.sol (Bridge): 154~156	● Acknowledged

Description

The implementation of the `isContract` check can not cover all scenarios. The check can be bypassed if the call is from the constructor of a smart contract or when the contract is destroyed. Because, in that case, the code size will also be zero.

Recommendation

It is recommended to add the additional `msg.sender == tx.origin` check to cover all the scenarios. Do note that the check still works for the current EVM (London) version, but future updates to the EVM or EIP (ex. EIP-3074) might cause the check to become ineffective.

Alleviation

The team acknowledged this issue and stated that they will implement the suggested code in the future.

ETC-03 | PANCAKE PAIR SHOULD BE EXCLUDED FROM REWARDS

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/EmpireToken.sol (EmpireToken V3)	● Acknowledged

Description

Generally, deflationary tokens are incompatible with DEX and special rules need to be coded for DEX addresses eg. excluding them from reward. Otherwise, a hacker can exploit the protocol using the reflection mechanism.

The balance of accounts that include rewards is calculated by `rAmount/rate` , where the rate is determined base on the total supply. If the `deliver()` function is executed with significant input, it can significantly decrease `rTotal` , thereby allowing for manipulation of the rate.

Scenario

If the pair is not excluded from rewards and a large portion of the token supply is added as the liquidity of a WBNB-EMPIRE pair, it becomes vulnerable to a flash loan attack.

1. Flash loan WBNB to buy most of EMPIRE in the Pancake pair.
2. Call `deliver()` function to burn attacker's tokens `_rOwned[attacker]` , thereby the rate is significantly reduced. This will result in an increase in the EMPIRE balance of the Pancake pair.
3. Utilize the `skim()` function of the pair to acquire the increased EMPIRE amount in the pair.
4. Repeat step 2 to increase the EMPIRE balance of the Pancake pair dramatically.
5. With the extra EMPIRE tokens, swap for WBNB without transferring any EMPIRE to the pair to drain the pool.
6. Repay flash loan.

Recommendation

We recommend excluding the dex pair from rewards.

Alleviation

The team acknowledged this issue and decided to leave it as it is for now.

ETC-14 | VARIABLE `_rOwned[account]` NOT UPDATED IN FUNCTION `includeInReward()`

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/EmpireToken.sol (EmpireToken V3): 427~438	● Acknowledged

Description

```
    function includeInReward(address account) external onlyOwner() {
        require(account != bridgeVault, "Bridge Vault can't receive reward");
        require(_isExcluded[account], "Account is already included");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                _excluded[i] = _excluded[_excluded.length - 1];
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
            }
        }
    }
```

Variable `_rOwned[account]` is not updated in the function `includeInReward()` , which will make the accounts included siphon off the tokens out of the balances of all token holders.

The Rate was higher at the moment of the `excludeFromReward(account)` call, so the `_rOwned[account] / _tOwned[account]` ratio is bigger than expected for accounts included in the reward.

Scenario

1. Let `_rTotal = 1000`
2. and `_tTotal = 100`
3. then `Rate = 1000 / 100 = 10` .
4. AccountA with a balance of 100R/10T (reflections/tokens) is `excludedFromReward` , then
5. `Rate = (1000 - 100) / (100 - 10) = 900 / 90 = 10` is unchanged.
6. Several transfers happen, and 90R are burned and subtracted from `_rTotal` . `_rTotal` is now 910.
7. The Rate drops `Rate = (910 - 100) / (100 - 10) = 810 / 90 = 9` .
8. All the rewarded accounts get extra 11.1% token balances, except AccountA - it is still 100R/10T.
9. Then AccountA is suddenly `includedToReward` . Since AccountA `_rOwned` was not updated, it unintentionally changes the Rate:

10. `Rate = (910 - 0) / (100 - 0) = 910 / 100 = 9.1` .
11. Since the Rate accidentally increased, all the rewarded accounts' token balances decreased by 1.1%.
12. Since AccountA reflection balance is still 100R, its token balance is `balance = rOwned / Rate = 100 / 9.1 = 11` . This is also undesired.

Recommendation

We recommend updating `_rOwned[account]` and `_rTotal` to keep the Rate unchanged:

```
function includeInReward(address account) external onlyOwner() {
        require(_isExcluded[account], "Account is not excluded");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                uint256 currentRate = _getRate();
                _rTotal = _rTotal.sub(_rOwned[account]);
                _rOwned[account] = _tOwned[account].mul(currentRate);
                _tOwned[account] = 0;
                _rTotal = _rTotal.add(_rOwned[account]);

                _isExcluded[account] = false;
                _excluded[i] = _excluded[_excluded.length - 1];
                _excluded.pop();
                break;
            }
        }
    }
```

Alleviation

The team acknowledged this issue and decided to leave it as it is for now.

MAI-01 | LACK OF REASONABLE BOUNDARY

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/EmpireToken.sol (EmpireToken V3): 984~1016; contracts/Bridge.sol (Bridge): 224~227	● Resolved

Description

The variables `fee` , `buyFee` , and `sellFee` do not have reasonable boundaries, so they can be given arbitrary values.

Recommendation

We recommend adding reasonable upper and lower boundaries to all the configuration variables.

Alleviation

The team resolved this issue in `https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code` , and set the max total fee when buy and sell as 50%.

ETC-04 | NEED MAX TRANSACTION CHECK

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/EmpireToken.sol (EmpireToken V3): 1	● Acknowledged

Description

It is recommended to add the max transaction amount check as many other deflation tokens(e.g. Safemoon) do to prevent the big whale.

Recommendation

We advise the client to modify the code as the aforementioned information.

Alleviation

The team acknowledged this and stated that this aligns with their original design.

ETC-05 | PROPER USAGE OF “PURE” AND “VIEW”

Category	Severity	Location	Status
Coding Style	● Minor	contracts/EmpireToken.sol (EmpireToken V3): 241~256	● Resolved

Description

Function state mutability should be restricted to `view` instead of `pure` for the reason that `_name` , `_symbol` , `_tTotal` , and `_decimals` are all state variables.

Recommendation

We advise the client to modify the code as the aforementioned information.

Alleviation

The team heeded our advice and resolved this issue in `https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code` .

ETC-06 | POTENTIAL SANDWICH ATTACKS

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/EmpireToken.sol (EmpireToken V3): 808	● Acknowledged

Description

Potential sandwich attacks could happen if calling `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens` and `uniswapV2Router.addLiquidityETH` without setting restrictions on slippage.

For example, when we want to make a transaction of swapping 100 A Token for 1 Eth, an attacker could raise the price of Eth by adding A Token into the pool before the transaction so we might only get 0.1 Eth. After the transaction, the attacker would be able to withdraw more than he deposited because the total value of the pool increases by 0.9 Eth.

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Alleviation

The team acknowledged this issue and decided to leave it as it is for now.

ETC-07 | THIRD PARTY DEPENDENCIES

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/EmpireToken.sol (EmpireToken V3): 792	● Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party protocol, including:

- `uniswapV2Router`

The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised that led to assets being lost or stolen.

Recommendation

We understand that the business logic requires interaction with third parties. We encourage the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and stated that they will monitor 3rd parties to secure investors.

ETC-08 | UNUSED RETURN VALUE

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/EmpireToken.sol (EmpireToken V3): 809~816	● Acknowledged

Description

The return value of an external call is not stored in a local or state variable.

```
809         uniswapV2Router.addLiquidityETH{value: ethAmount}(
810             address(this),
811             tokenAmount,
812             0,
813             0,
814             liquidityWallet,
815             block.timestamp
816         );
```

Recommendation

We recommend checking or using the return values of all external function calls.

Alleviation

The team acknowledged this issue and decide to leave it as it is for now.

BRI-04 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	● Informational	contracts/Bridge.sol (Bridge): 232	● Acknowledged

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

The team acknowledged this issue and stated that they will implement suggested code in the future.

ETC-09 | THE PURPOSE OF FUNCTION `deliver`

Category	Severity	Location	Status
Control Flow	● Informational	contracts/EmpireToken.sol (EmpireToken V3): 374~386	● Acknowledged

Description

The function `deliver` can be called by anyone. It accepts an uint256 number parameter `tAmount`. The function reduces the Empire token balance of the caller by `rAmount`, which is `tAmount` reduces the transaction fee. Then, the function adds `tAmount` to variable `_tFeeTotal`, which represents the contract's total transaction fee.

Recommendation

We wish the team could explain more on the purpose of having such functionality.

Alleviation

The team acknowledged this issue and stated that this function is made just in case somebody wants to burn their tokens and distribute it as reflections between all holders.

ETC-10 | TYPOS IN THE CONTRACT

Category	Severity	Location	Status
Coding Style	● Informational	contracts/EmpireToken.sol (EmpireToken V3): 125, 443	● Resolved

Description

There are several typos in the code and comments.

1. In the following code snippet, `tokensIntoLiqudity` should be `tokensIntoLiquidity` .

```
    event LogSwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
```

2. `recieve` should be `receive` in the line of comment `//to recieve ETH from uniswapV2Router when swapping.

Recommendation

We recommend correcting all typos in the contract.

Alleviation

The team heeded our advice and resolved this issue in `https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code` .

ETC-11 | REDUNDANT SAFEMATH USAGE

Category	Severity	Location	Status
Language Specific	● Informational	contracts/EmpireToken.sol (EmpireToken V3): 51	● Acknowledged

Description

Solidity version >=0.8.0 includes checked arithmetic operations and underflow/overflow by default, making SafeMath redundant.

Recommendation

We recommend removing the SafeMath library and use standard arithmetic operators to reduce code complexity.

Alleviation

The team acknowledged this issue and decided to leave it as it is for now.

ETC-12 | UNUSED EVENT

Category	Severity	Location	Status
Coding Style	● Informational	contracts/EmpireToken.sol (EmpireToken V3): 150	● Resolved

Description

```
150      event LogSetBurnWallet(address indexed setter, address burnWallet);
```

- `LogSetBurnWallet` is declared in `EmpireToken` but never emitted.

Recommendation

We advise removing the unused events or emitting them in the intended functions.

Alleviation

The team heeded our advice and resolved this issue in `https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code`.

GLOBAL-01 | UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	● Informational		● Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to different compiler versions. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

Alleviation

The team heeded our advice and resolved this issue in `https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code`.

OPTIMIZATIONS | EMPIRE V3

ID	Title	Category	Severity	Status
ETC-13	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Resolved

ETC-13 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/EmpireToken.sol (EmpireToken V3): 99, 104	● Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

The team heeded our advice and resolved this issue in `https://etherscan.io/token/0x9A2Af0AbB12bee5369B180976Be01E8c80D0e7B6#code`.

APPENDIX | EMPIRE V3

Finding Categories

Categories	Description
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.
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 that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
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 | CERTIK

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.

CertiK | **Securing** the **Web3** World

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.