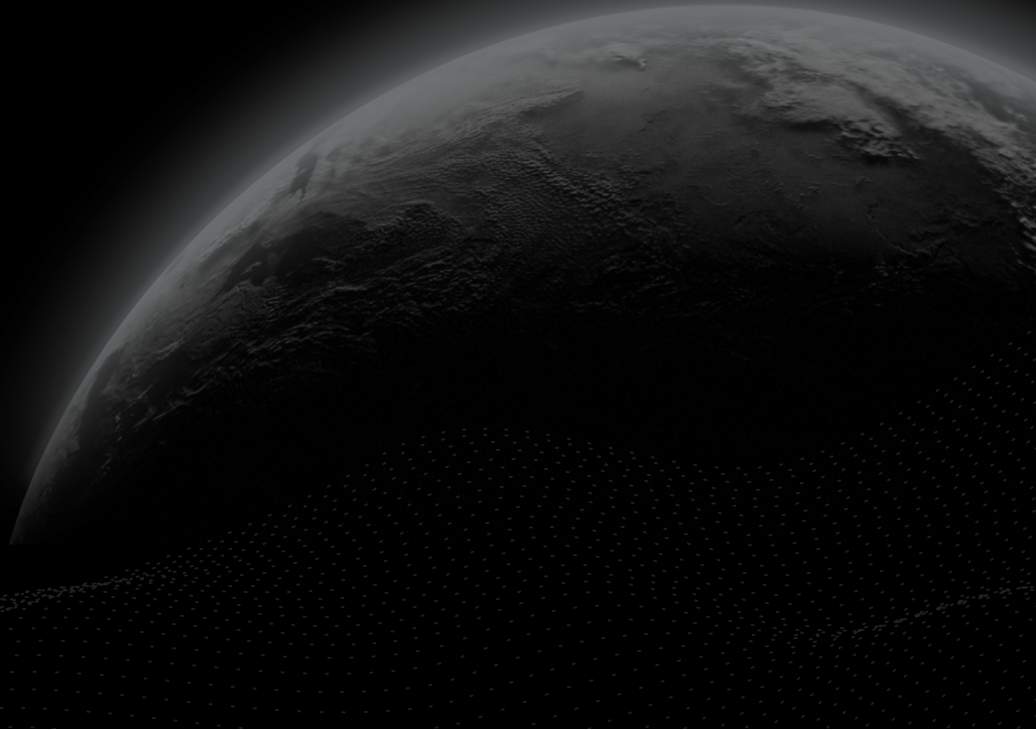




Security Assessment

commonwealth - audit

CertiK Assessed on Aug 30th, 2024





CertiK Assessed on Aug 30th, 2024

commonwealth - audit

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

Executive Summary

TYPES

Others

ECOSYSTEM

Binance Smart Chain (BSC)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 08/30/2024

KEY COMPONENTS

N/A

CODEBASE

[CommonWealth](#)

[View All in Codebase Page](#)

COMMITTS

[Initial commit](#)

[Remediation commit](#)

[View All in Codebase Page](#)

Highlighted Centralization Risks

- ⚠ Contract upgradeability
- ⚠ Privileged role can mint tokens
- ⚠ Fees are bounded by 8%
- ⚠ Initial owner token share is 100%

Vulnerability Summary



12

Total Findings

6

Resolved

0

Mitigated

0

Partially Resolved

6

Acknowledged

0

Declined

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.

3 Major

3 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.

0 Medium

Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

7 Minor

5 Resolved, 2 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.

■ 2 Informational

1 Resolved, 1 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 | COMMONWEALTH - AUDIT

■ **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

■ **Findings**

[ATL-01 : Initial Token Distribution](#)

[GLOBAL-01 : Centralization Related Risks](#)

[SRC-01 : Centralized Control of Contract Upgrade](#)

[ATL-02 : Missing Zero Address Validation](#)

[ATL-03 : Unreachable Code due to uninitialized mapping](#)

[ATL-04 : Inappropriate Token Name Identifier](#)

[OBB-01 : Unchecked ERC-20 `transfer\(\)`/`transferFrom\(\)` Call](#)

[SRC-02 : `initialize\(\)` Is Unprotected](#)

[SRC-03 : Third-Party Dependency Usage](#)

[SRC-04 : Potential Cross-Chain Replay Attack](#)

[ATL-05 : Missing Emit Events](#)

[PGB-01 : Redundant Check](#)

■ **Appendix**

■ **Disclaimer**

CODEBASE | COMMONWEALTH - AUDIT

■ Repository

CommonWealth






■ Commit

Initial commit

Remediation commit

AUDIT SCOPE | COMMONWEALTH - AUDIT

5 files audited ● 5 files with Acknowledged findings

ID	Repo	File	SHA256 Checksum
● OBB	memeboxes/four-contracts-certik	 airdrop/OpenBox.sol	25ea0e560dd4e0096da73b2d82acca4242e5612a5bc0856c30bdeb4cf9bb0e7a
● SCB	memeboxes/four-contracts-certik	 libs/SignatureChecker.sol	f709e53eb7a361010f88eccc6f3fd9473df502450a71dc38f1257f2bf1454408
● PGB	memeboxes/four-contracts-certik	 nft/PostGameBox.sol	8cc21aae99ebd0e3f31525a91dfe82c5533f76b91113b87d804b0ee7b24b3e2f
● PRE	memeboxes/four-contracts-certik	 nft/PreGameBox.sol	03c7ff07b0b30b67738314b617ce78b79d6e3b7fb21f14df69cba381878aca2f
● ATL	memeboxes/four-contracts-certik	 token/Atlantis.sol	03d09ecfd9387ee3d0f7da64cd7ab17aa59bb8871716fcbacf6e00ce030208da

APPROACH & METHODS | COMMONWEALTH - AUDIT

This report has been prepared for commonwealth to discover issues and vulnerabilities in the source code of the commonwealth - audit 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 | COMMONWEALTH - AUDIT



12

Total Findings

0

Critical

3

Major

0

Medium

7

Minor

2

Informational

This report has been prepared to discover issues and vulnerabilities for commonwealth - audit. Through this audit, we have uncovered 12 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
ATL-01	Initial Token Distribution	Centralization	Major	● Acknowledged
GLOBAL-01	Centralization Related Risks	Centralization	Major	● Acknowledged
SRC-01	Centralized Control Of Contract Upgrade	Centralization	Major	● Acknowledged
ATL-02	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
ATL-03	Unreachable Code Due To Uninitialized Mapping	Inconsistency	Minor	● Resolved
ATL-04	Inappropriate Token Name Identifier	Logical Issue	Minor	● Resolved
OBB-01	Unchecked ERC-20 <code>transfer()</code> / <code>transferFrom()</code> Call	Volatile Code	Minor	● Resolved
SRC-02	<code>initialize()</code> Is Unprotected	Logical Issue	Minor	● Resolved
SRC-03	Third-Party Dependency Usage	Design Issue	Minor	● Acknowledged
SRC-04	Potential Cross-Chain Replay Attack	Logical Issue	Minor	● Acknowledged
ATL-05	Missing Emit Events	Coding Style	Informational	● Acknowledged

ID	Title	Category	Severity	Status
PGB-01	Redundant Check	Logical Issue	Informational	● Resolved

ATL-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	● Major	token/Atlantis.sol: 71~73	● Acknowledged

Description

All of the `Atlantis` tokens are sent to the contract deployer on contract deployment. This is a centralization risk because the deployer can distribute tokens without obtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature ($2/3$, $3/5$) wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

Alleviation

[CommonWealth Team, 08/30/2024]: For agile development, team is managing ownership and proxy with admin accounts. But team will work with community to move to DAO.

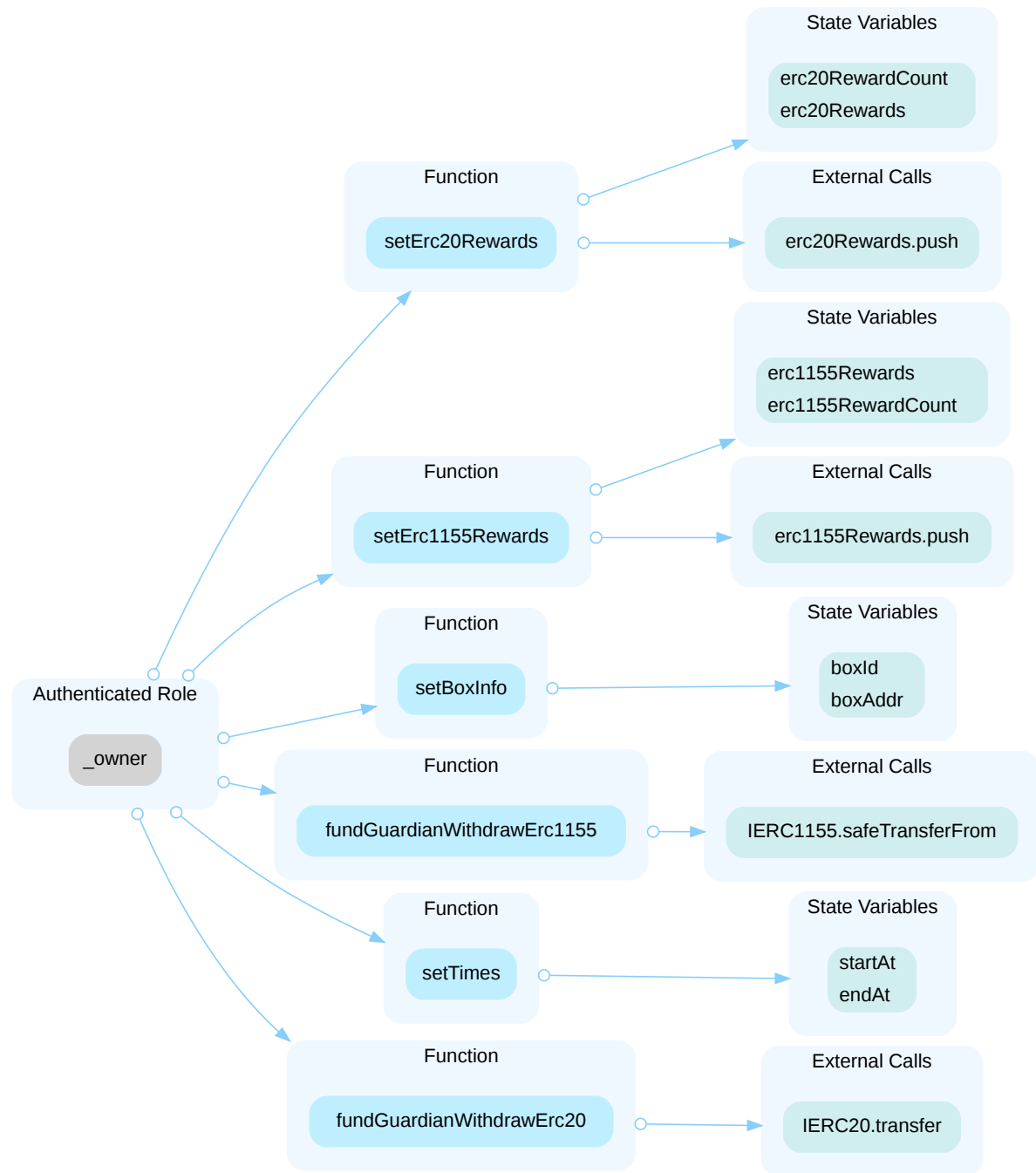
GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● Major		● Acknowledged

Description

In the contract `openBox` 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 and

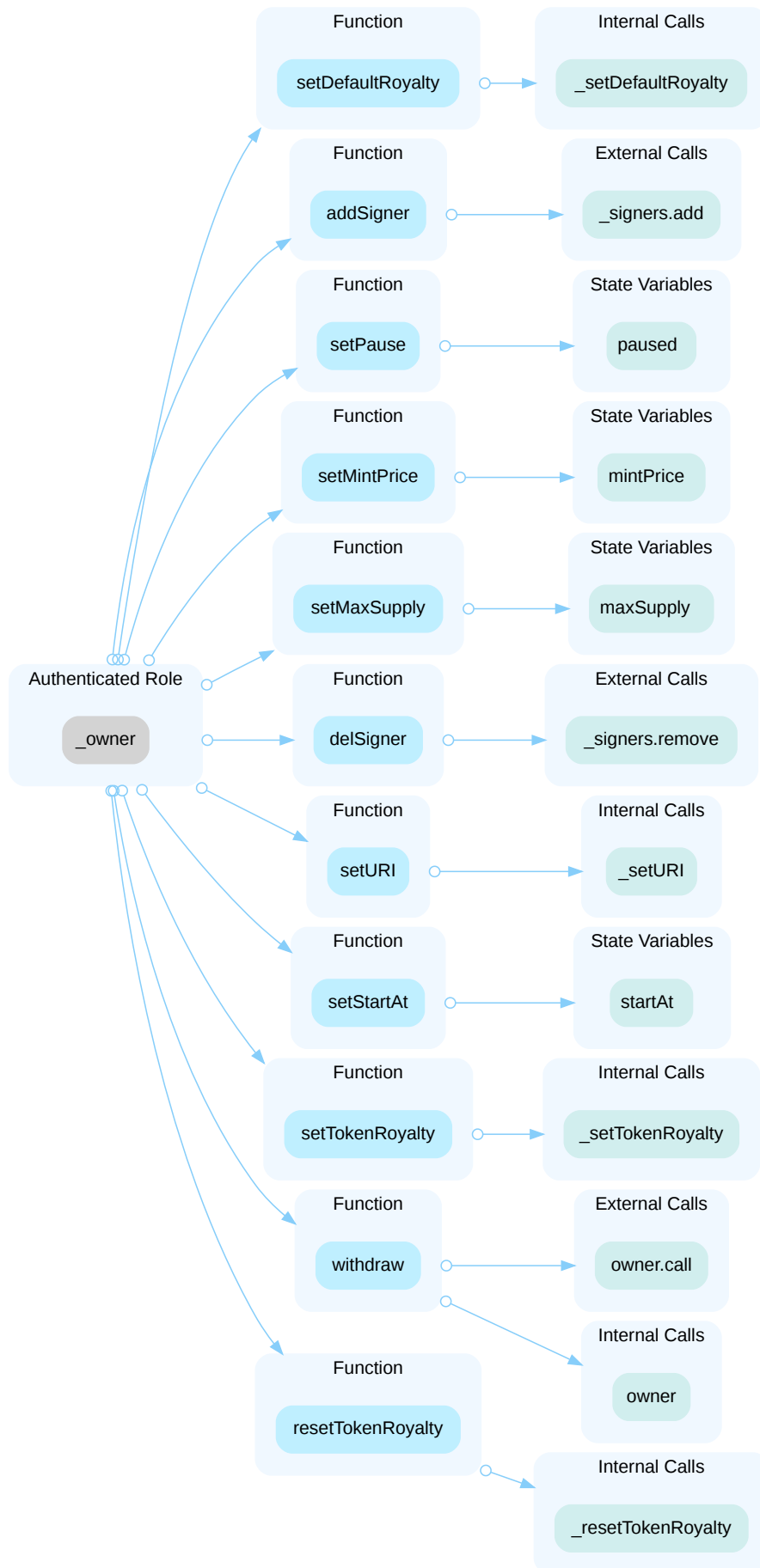
- Set Airdrop rewards token and amount
- Set `boxId` and `boxAddr` through function `setBoxInfo`
- Set Airdrop start/end timestamps
- Withdraw reward ERC-20/ERC-1155 to the `Fund_Guard` address



In the contract `PreGameBox` 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 and

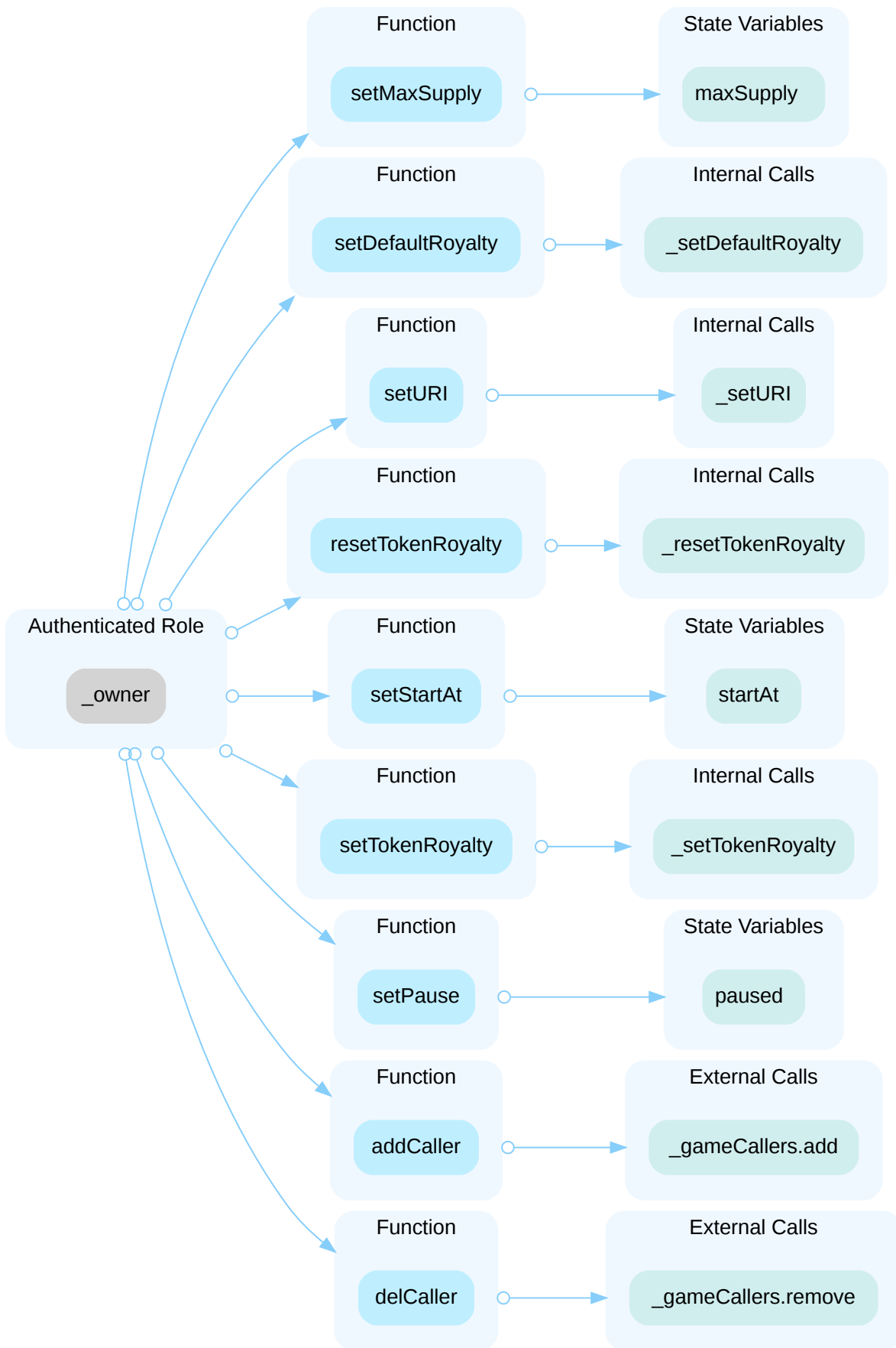
- Set mint start timestamp and contract pause status
- Set ERC-1155 `PreGameNFT` token URI
- Set `PreGameNFT` mint price
- Set max supply for `PreGameNFT` token mint
- **Add/Delete `signer` role, who is responsible to sign mint messages. If the `signer` or `owner` role get compromised, the attacker can mint `PreGameNFT` tokens without limitation.**
- Set ERC2981 royalty information for any address

- Withdraw mint fee from the contract



In the contract `PostGameBox` 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 and

- Set mint start timestamp and contract pause status
- Set ERC-1155 `PostGameNFT` token URI
- Set max supply for `PostGameNFT` token mint
- **Add/Delete `caller` role, who has access to mint `PostGameNFT` tokens. If the `caller` or `owner` role get compromised, the attacker can mint `PostGameNFT` tokens without limitation.**
- Set ERC2981 royalty information for any address

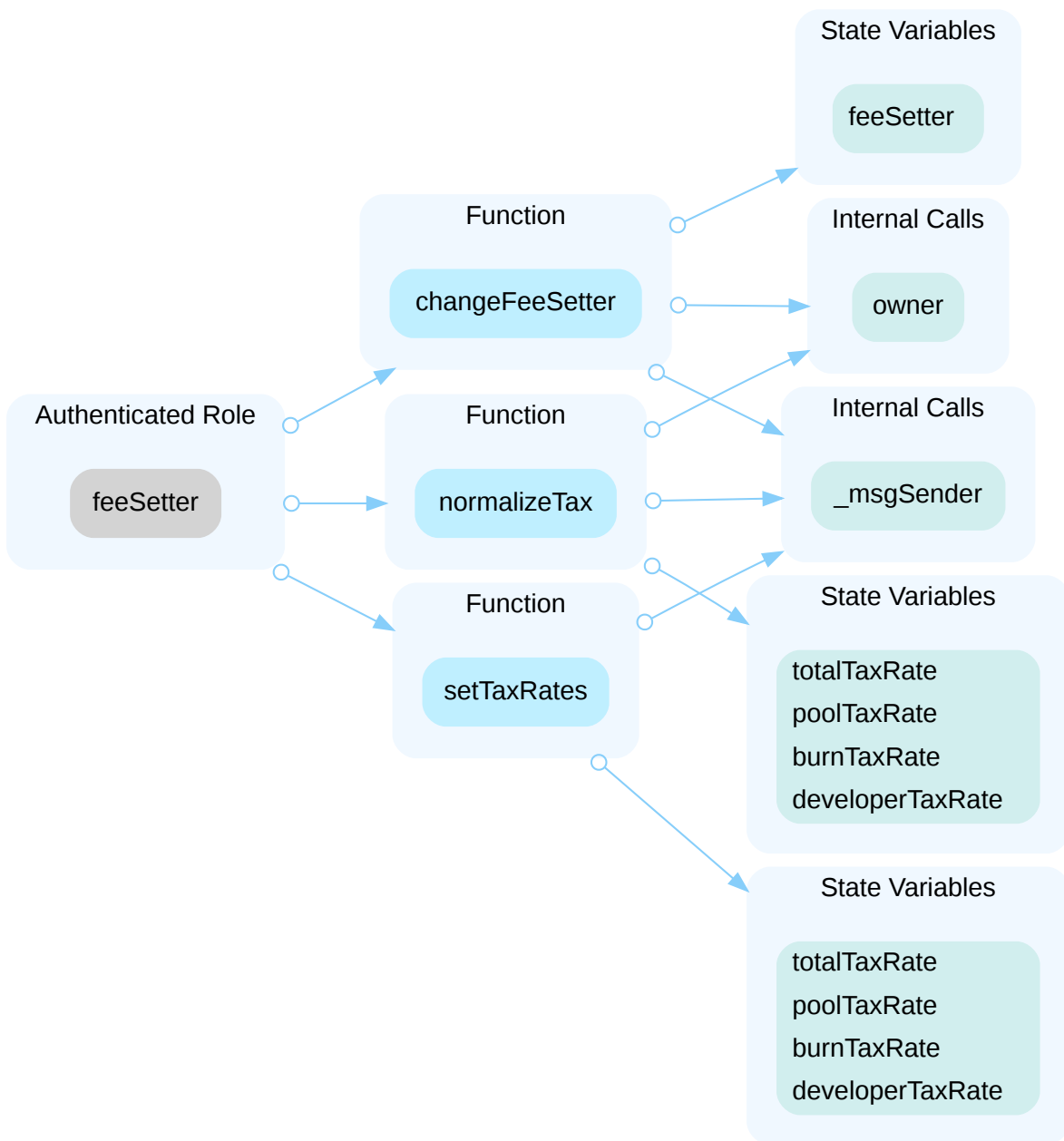


In the contract `AtlantisToken` the role `owner` has authority over the following functions:

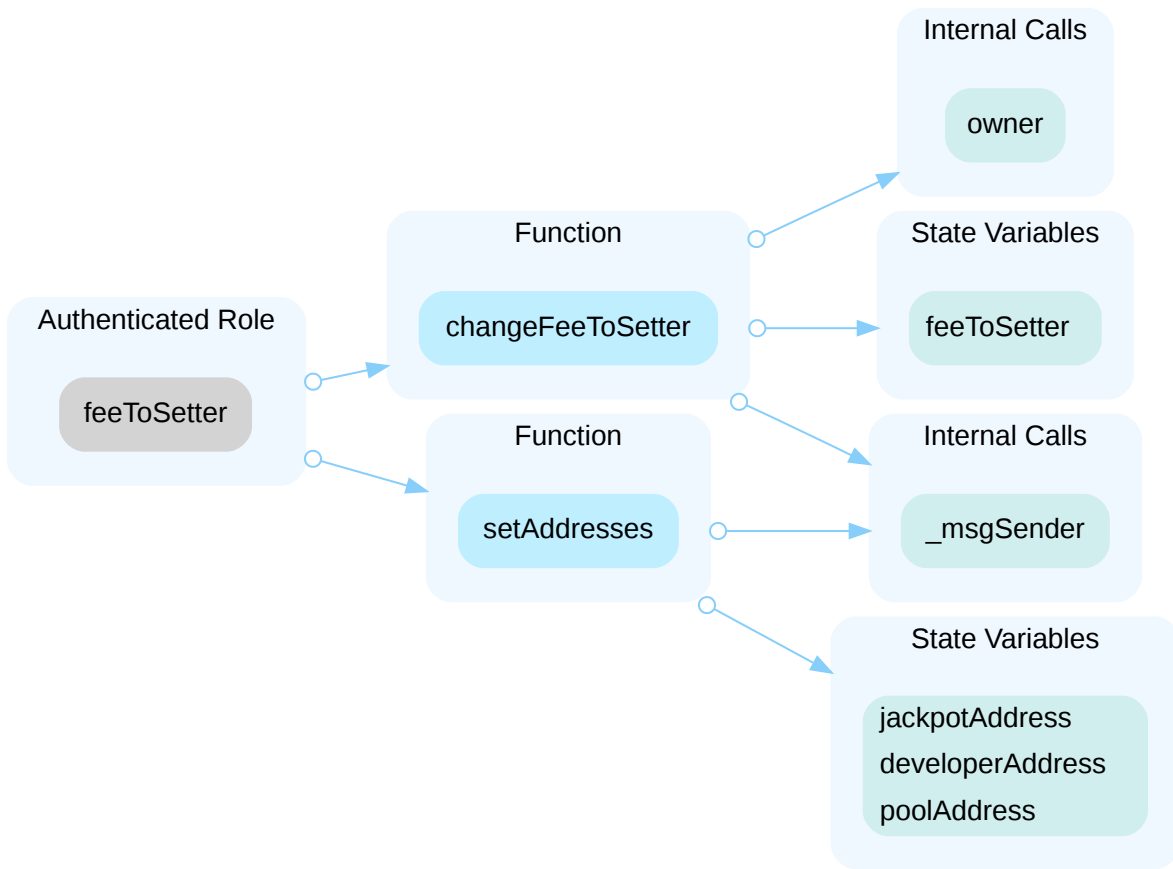
- changeFeeSetter()
- changeFeeToSetter()
- changeWhiteListSetter()
- normalizeTax()
- setLotteryEnabled()

Any compromise to the `owner` account may allow the hacker to take advantage of this authority and modify privileged role settings and set the `LotteryEnabled` flag.

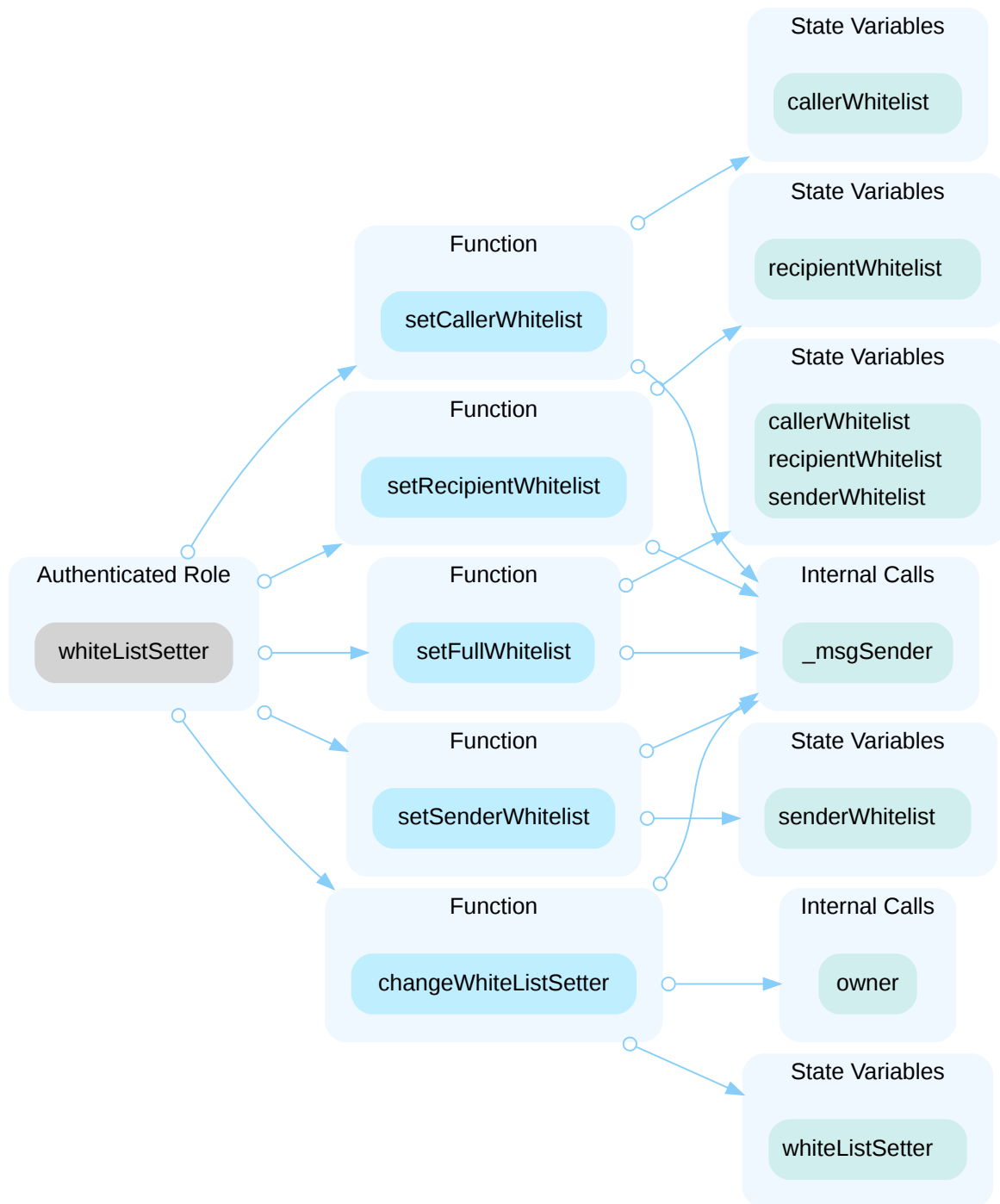
In the contract `AtlantisToken` the role `feeSetter` has authority over the functions shown in the diagram below. Any compromise to the `feeSetter` account may allow the hacker to take advantage of this authority and modify token transfer fee settings.



In the contract `AtlantisToken` the role `feeToSetter` has authority over the functions shown in the diagram below. Any compromise to the `feeToSetter` account may allow the hacker to take advantage of this authority and modify fee receiver address settings.



In the contract `AtlantisToken` the role `whiteListSetter` has authority over the functions shown in the diagram below. Any compromise to the `whiteListSetter` account may allow the hacker to take advantage of this authority and modify whitelist addresses that are waived from fees.



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 (2/3, 3/5) 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

[CommonWealth Team, 08/30/2024]: For agile development, team is managing ownership and proxy with admin accounts. But team will work with community to move to DAO.

[CertiK, 08/30/2024]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, it strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

SRC-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	● Major	airdrop/OpenBox.sol: 12; nft/PostGameBox.sol: 13; nft/PreGameBox.sol: 13	● Acknowledged

Description

In the linked contracts `OpenBox`, `PreGameBox` and `PostGameBox`, the role `admin` has the authority to update the implementation contract behind the proxy contract.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and therefore execute potential malicious functionality in the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (2/3, 3/5) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (2/3, 3/5) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.

- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

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

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

I Alleviation

[CommonWealth Team, 08/30/2024]: For agile development, team is managing ownership and proxy with admin accounts. But team will work with community to move to DAO.

ATL-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	token/Atlantis.sol: 61	● Resolved

Description

Under the current implementation, the `jackpot` address is not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior when calling

```
IJackpot(jackpotAddress).tradeEvent(sender, amount).
```

```
61     jackpotAddress = _jackpot;
```

```
136     function setAddresses(address _poolAddress, address _developerAddress,
address _jackpot) public {
137         require(_msgSender() == feeToSetter, "!allow");
138
139         poolAddress = _poolAddress;
140         developerAddress = _developerAddress;
141         jackpotAddress = _jackpot;
142     }
```

- `_jackpot` is not zero-checked before being used.

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[CommonWealth Team, 08/30/2024]: The team heeded the advice and resolved the issue in commit 724d4537533ff13670ef54611cfc0da7e5afb6fe.

ATL-03 | UNREACHABLE CODE DUE TO UNINITIALIZED MAPPING

Category	Severity	Location	Status
Inconsistency	● Minor	token/Atlantis.sol: 106~108	● Resolved

Description

The `pairs` mapping is utilized to determine whether the sender is a valid pair address in the context of lottery functionality. However, there's no indication from the current codebase that this mapping is properly initialized or updated before being used in condition checks.

As a result, the following code will never be executed, because the mapping `pairs[]` is not initialized or updated before being used and `pairs[sender]` will always be false.

```
186     if (lotteryEnabled && pairs[sender]) {  
187         try IJackpot(jackpotAddress).tradeEvent(sender, amount) {} catch {}  
188     }
```

Recommendation

It is recommended that the team initialize the mapping `pairs[]` before using it in condition checks.

Alleviation

[CommonWealth Team, 08/30/2024]: The team heeded the advice and resolved the issue in commit 724d4537533ff13670ef54611cfc0da7e5afb6fe.

ATL-04 | INAPPROPRIATE TOKEN NAME IDENTIFIER

Category	Severity	Location	Status
Logical Issue	● Minor	token/Atlantis.sol: 56	● Resolved

Description

The constructor for the AtlantisToken contract initializes the ERC20 token with the name and symbol both set to "TEST". This configuration is generally suitable for development or testing environments but is inappropriate for a production deployment. Using such placeholder values in a live setting can cause confusion and potentially undermine the credibility of the token, as it may not be taken seriously by users and investors.

```
55     constructor(address _poolAddress, address _developerAddress, address
    _jackpot, address _swapRouter)
56         ERC20("TEST", "TEST")
```

Recommendation

It is recommended that the team select appropriate names and symbols for the Atlantis Token during the ERC-20 initialization process

Alleviation

[CommonWealth Team, 08/30/2024]: This will be updated before we deploy to mainnet.

OBB-01 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	● Minor	airdrop/OpenBox.sol: 95, 116	● Resolved

Description

The return values of the `transfer()` and `transferFrom()` calls in the smart contract are not checked. Some ERC-20 tokens' transfer functions return no values, while others return a bool value, they should be handled with care. If a function returns `false` instead of reverting upon failure, an unchecked failed transfer could be mistakenly considered successful in the contract.

```
95      IERC20(erc20Rewards[i].token).transfer(msg.sender, rewardAmount);
```

```
116     IERC20(_token).transfer(FUND_GUARD, _amount);
```

Recommendation

It is advised to use the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if false is returned, making it compatible with all ERC-20 token implementations.

Alleviation

[CommonWealth Team, 08/30/2024]: The team heeded the advice and resolved the issue in commit 724d4537533ff13670ef54611cfc0da7e5afb6fe.

SRC-02 | `initialize()` IS UNPROTECTED

Category	Severity	Location	Status
Logical Issue	● Minor	airdrop/OpenBox.sol: 39, 39; nft/PostGameBox.sol: 37, 37; nft/PreGameBox.sol: 38, 38	● Resolved

Description

The `OpenBox`, `PostGameBox` and `PreGameBox` logic contracts do not protect the initializer functions. An attacker can call the `initialize` function directly to the implementation contract and assume ownership of the logic contract. Once in control, the attacker can perform privileged operations, misleading users into believing that they are interacting with the legitimate owner of the upgradeable contract.

Recommendation

We recommend adding

```
/// @custom:oz-upgrades-unsafe-allow constructor
constructor() initializer {}
```

The addition will prevent the function `initialize()` from being called directly in the implementation contract, but the proxy will still be able to `initialize()` its storage variables.

Alleviation

[CommonWealth Team, 08/30/2024]: The team heeded the advice and resolved the issue in commit 724d4537533ff13670ef54611cfc0da7e5afb6fe.

SRC-03 | THIRD-PARTY DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	● Minor	nft/PreGameBox.sol: 156~160; token/Atlantis.sol: 39	● Acknowledged

Description

The linked contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
156     function _registerFilterer() internal {
157         if (address(OPERATOR_FILTER_REGISTRY).code.length > 0) {
158             OPERATOR_FILTER_REGISTRY.registerAndSubscribe(address(this),
CANONICAL_CORI_SUBSCRIPTION);
159         }
160     }
```

- The contracts `PreGameBox` and `PostGameBox` interact with third party contract with `IOperatorFilterRegistry` interface via `OPERATOR_FILTER_REGISTRY`.

```
39     address public jackpotAddress;
```

- The contract `AtlantisToken` interacts with third party contract with `IJackpot` interface via `jackpotAddress`.

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[CommonWealth Team, 08/30/2024]: The team acknowledged the finding and decided not to change the current codebase.

SRC-04 | POTENTIAL CROSS-CHAIN REPLAY ATTACK

Category	Severity	Location	Status
Logical Issue	● Minor	libs/SignatureChecker.sol: 32; nft/PreGameBox.sol: 112	● Acknowledged

Description

The `PreGameBox` smart contract utilizes EIP-712 style signatures for authorization in its `mintPreGameNft()` function. However, signed messages are not properly verified with the current chain ID, thus allowing attackers to perform replay attacks across chains.

Please also note that hardcoded or cached chain ID values are also vulnerable since a hard fork may occur and change the chain ID in the future.

```
111         bytes32 message = keccak256(abi.encode(address(this), _msgSender(), id,
112         _signers.requireValidSignature(message, signature);
```

- The linked code calls `requireValidSignature`, which eventually calls `ecrecover` with `chainid` not included in the message.

```
32         require(validSignature(signers, message, signature),
"SignatureChecker: Invalid signature");
```

- Calling `validSignature`, which eventually calls `ecrecover`.

```
142         address signer = ecrecover(hash, v, r, s);
```

- Calling `ecrecover` with a hash that does not properly include the chain ID.

Recommendation

We recommend modifying the message construction to include the chain ID explicitly and verifying signed messages against the current chain ID by using `block.chainid` or `chainid()` within the same transaction.

Alleviation

[CommonWealth Team, 08/30/2024]: The team acknowledged the finding and decided not to change the current codebase.

ATL-05 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	token/Atlantis.sol: 122~147	● Acknowledged

Description

It is important to emit events for sensitive actions, particularly those that can be executed by centralized roles or administrators. This ensures transparency and enables tracking of critical changes, which is essential for security and trust in the system. Missing event logs can indeed result in a lack of visibility and potential information loss.

- Following function calls do not emit events for sensitive actions:

```
122     function setTaxRates(uint256 _burnTaxRate, uint256 _poolTaxRate, uint256
    _developerTaxRate) public {
123         require(_msgSender() == feeSetter, "!allow");
124
125         burnTaxRate = _burnTaxRate;
126         poolTaxRate = _poolTaxRate;
127         developerTaxRate = _developerTaxRate;
128
129         totalTaxRate = _burnTaxRate + _poolTaxRate + _developerTaxRate;
130         require(totalTaxRate <= 800, "invalid tax rate");
131     }
132
133     /**
134     * @dev set the address of pool, developer and jackpot contract.
135     */
136     function setAddresses(address _poolAddress, address _developerAddress,
    address _jackpot) public {
137         require(_msgSender() == feeToSetter, "!allow");
138
139         poolAddress = _poolAddress;
140         developerAddress = _developerAddress;
141         jackpotAddress = _jackpot;
142     }
143
144     function setLotteryEnabled(bool _enabled) public {
145         require(_msgSender() == owner(), "!allow");
146         lotteryEnabled = _enabled;
147     }
```

Recommendation

We recommend adding events for state-changing actions, and emitting them in their relevant functions.

I Alleviation

[CommonWealth Team, 08/30/2024]: those two control events will not be called frequently and will perform read after write confirmation for each update.

PGB-01 | REDUNDANT CHECK

Category	Severity	Location	Status
Logical Issue	● Informational	nft/PostGameBox.sol: 97	● Resolved

Description

The function `mintPostGameNft()` verifies whether the current caller is contained in the EnumerableSet `gameCallers`. However, the same check has already been implemented in the modifier `postGameNftMintControl`, making the following check redundant.

```
97     require(!_gameCallers.contains(_msgSender()), "not game caller");
```

- Same check in modifier `postGameNftMintControl`:

```
30     modifier postGameNftMintControl() {
31         require(!paused, "paused");
32         require(block.timestamp >= startAt, "not started");
33         require(!_gameCallers.contains(_msgSender()), "not valid caller");
34         _;
35     }
36
```

Recommendation

We recommend removing the `_gameCallers` check in the function `mintPostGameNft()`.

Alleviation

[CommonWealth Team, 08/30/2024]: The team heeded the advice and resolved the issue in commit 724d4537533ff13670ef54611cfc0da7e5afb6fe.

APPENDIX | COMMONWEALTH - AUDIT

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

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.

