AllowlistRegistryProxy

Smart Contract Audit Report
Prepared for Token X

Date Issued:	Sep 12, 2023
Project ID:	AUDIT2023013
Version:	v1.1
Confidentiality Level:	Public

CYBERSECURITY PROFESSIONAL SERVICE

Report Information

Project ID	AUDIT2023013
Version	v1.1
Client	Token X
Project	AllowlistRegistryProxy
Auditor(s)	Wachirawit Kanpanluk Phitchakorn Apiratisakul
Author(s)	Phitchakorn Apiratisakul
Reviewer	Natsasit Jirathammanuwat
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.1	Sep 12, 2023	Update issue information	Wachirawit Kanpanluk
1.0	Jul 26, 2023	Full report	Phitchakorn Apiratisakul

Contact Information

Company	Inspex
Phone	(+66) 90 888 7186
Telegram	t.me/inspexco
Email	audit@inspex.co

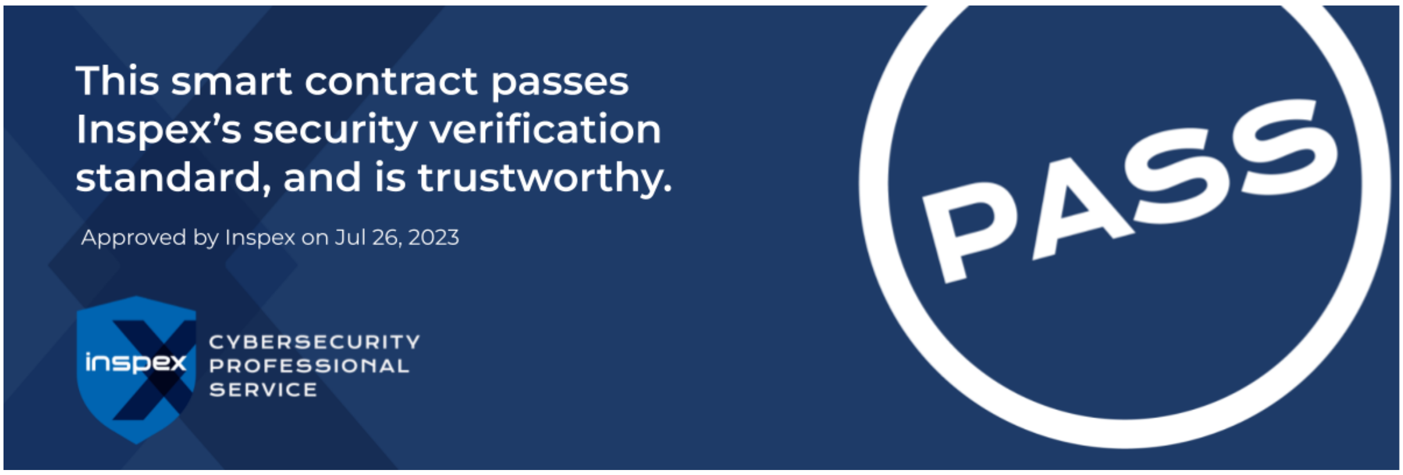
Table of Contents

1. Executive Summary

As requested by Token X, Inspex team conducted an audit to verify the security posture of the AllowlistRegistryProxy smart contracts on Jul 20, 2023. During the audit, Inspex team examined all smart contracts and the overall operation within the scope to understand the overview of AllowlistRegistryProxy smart contracts. Static code analysis, dynamic analysis, and manual review were done in conjunction to identify smart contract vulnerabilities together with technical & business logic flaws that may be exposed to the potential risk of the platform and the ecosystem. Practical recommendations are provided according to each vulnerability found and should be followed to remediate the issue.

1.1. Audit Result

In the initial audit, Inspex found 1 medium, and 2 info-severity issues. With the project team's prompt response, 1 medium, and 1 info-severity issue were resolved or mitigated in the reassessment, while 1 info-severity issue was mitigated by the team. Therefore, Inspex trusts that AllowlistRegistryProxy smart contracts have sufficient protections to be safe for public use. However, in the long run, Inspex suggests resolving all issues found in this report.



This smart contract passes Inspex's security verification standard, and is trustworthy.

Approved by Inspex on Jul 26, 2023

inspex
CYBERSECURITY PROFESSIONAL SERVICE
PASS

1.2. Disclaimer

This security audit is not produced to supplant any other type of assessment and does not guarantee the discovery of all security vulnerabilities within the scope of the assessment. However, we warrant that this audit is conducted with goodwill, professional approach, and competence. Since an assessment from one single party cannot be confirmed to cover all possible issues within the smart contract(s), Inspex suggests conducting multiple independent assessments to minimize the risks. Lastly, nothing contained in this audit report should be considered as investment advice.

2. Project Overview

2.1. Project Introduction

The AllowlistRegistryProxy contract is responsible for managing and storing the registry data of platforms. It includes a list of users who are whitelisted to access the platform, as well as security-related privilege functions such as pausing mechanisms and a blacklist feature. The blacklist feature is designed to prevent malicious users from gaining access to the platform. Additionally, the contract facilitates fetching the relevant data from the registry to ensure proper platform management.

Scope Information:

Project Name	AllowlistRegistryProxy
Website	https://tokenx.finance/
Smart Contract Type	Ethereum Smart Contract
Chain	Token X (TKX)
Programming Language	Solidity
Category	Token

Audit Information:

Audit Method	Whitebox
Audit Date	Jul 20, 2023
Reassessment Date	Jul 25, 2023

The audit method can be categorized into two types depending on the assessment targets provided:

1. **Whitebox**: The complete source code of the smart contracts are provided for the assessment.
2. **Blackbox**: Only the bytecodes of the smart contracts are provided for the assessment.

2.2. Scope

The following smart contracts were audited and reassessed by Inspex in detail:

Initial Audit: (Commit: f6177a1fd4dc33d69bf91a910957e75fa9b66122)

Contract	Location (URL)
AllowlistRegistryProxy	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/f6177a1fd4/contracts/AllowlistRegistryProxy.sol
Ownable	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/f6177a1fd4/extensions/Ownable.sol
Storage	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/f6177a1fd4/contracts/Storage.sol
Blacklistable	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/f6177a1fd4/extensions/Blacklistable.sol

Reassessment: (Commit: 8aa1f63767c40ca8e9f1b94dee51d0c5ea06a082)

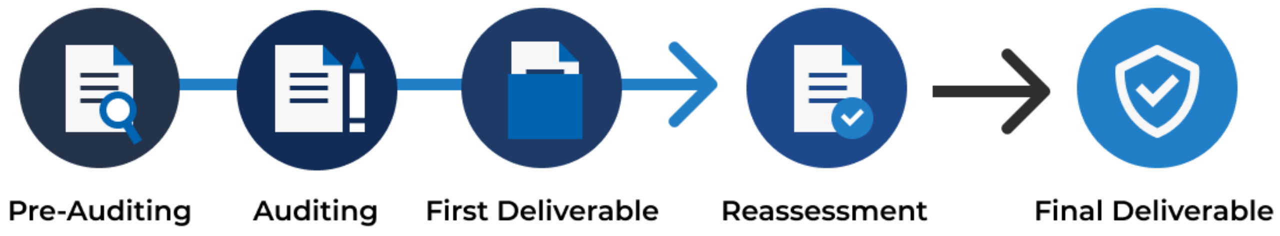
Contract	Location (URL)
AllowlistRegistryProxy	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/8aa1f63767/contracts/AllowlistRegistryProxy.sol
Ownable	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/8aa1f63767/extensions/Ownable.sol
Storage	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/8aa1f63767/contracts/Storage.sol
Blacklistable	https://github.com/tokenx-finance/allowlist-registry-core-contracts/blob/8aa1f63767/extensions/Blacklistable.sol

The assessment scope covers only the in-scope smart contracts and the smart contracts that they inherit from.

3. Methodology

Inspex conducts the following procedure to enhance the security level of our clients' smart contracts:

1. **Pre-Auditing**: Getting to understand the overall operations of the related smart contracts, checking for readiness, and preparing for the auditing
2. **Auditing**: Inspecting the smart contracts using automated analysis tools and manual analysis by a team of professionals
3. **First Deliverable and Consulting**: Delivering a preliminary report on the findings with suggestions on how to remediate those issues and providing consultation
4. **Reassessment**: Verifying the status of the issues and whether there are any other complications in the fixes applied
5. **Final Deliverable**: Providing a full report with the detailed status of each issue



3.1. Test Categories

Inspex smart contract auditing methodology consists of both automated testing with scanning tools and manual testing by experienced testers. We have categorized the tests into 3 categories as follows:

1. **General Smart Contract Vulnerability (General)** - Smart contracts are analyzed automatically using static code analysis tools for general smart contract coding bugs, which are then verified manually to remove all false positives generated.
2. **Advanced Smart Contract Vulnerability (Advanced)** - The workflow, logic, and the actual behavior of the smart contracts are manually analyzed in-depth to determine any flaws that can cause technical or business damage to the smart contracts or the users of the smart contracts.
3. **Smart Contract Best Practice (Best Practice)** - The code of smart contracts is then analyzed from the development perspective, providing suggestions to improve the overall code quality using standardized best practices.

3.2. Audit Items

The testing items checked are based on our Smart Contract Security Testing Guide (SCSTG) v1.0 (https://github.com/InspexCo/SCSTG/releases/download/v1.0/SCSTG_v1.0.pdf) which covers most prevalent risks in smart contracts. The latest version of the document can also be found at https://inspex.gitbook.io/testing-guide/.

The following audit items were checked during the auditing activity:

Testing Category	Testing Items
1. Architecture and Design	1.1. Proper measures should be used to control the modifications of smart contract logic 1.2. The latest stable compiler version should be used 1.3. The circuit breaker mechanism should not prevent users from withdrawing their funds 1.4. The smart contract source code should be publicly available 1.5. State variables should not be unfairly controlled by privileged accounts 1.6. Least privilege principle should be used for the rights of each role
2. Access Control	2.1. Contract self-destruct should not be done by unauthorized actors 2.2. Contract ownership should not be modifiable by unauthorized actors 2.3. Access control should be defined and enforced for each actor roles 2.4. Authentication measures must be able to correctly identify the user 2.5. Smart contract initialization should be done only once by an authorized party 2.6. tx.origin should not be used for authorization
3. Error Handling and Logging	3.1. Function return values should be checked to handle different results 3.2. Privileged functions or modifications of critical states should be logged 3.3. Modifier should not skip function execution without reverting
4. Business Logic	4.1. The business logic implementation should correspond to the business design 4.2. Measures should be implemented to prevent undesired effects from the ordering of transactions 4.3. msg.value should not be used in loop iteration
5. Blockchain Data	5.1. Result from random value generation should not be predictable 5.2. Spot price should not be used as a data source for price oracles 5.3. Timestamp should not be used to execute critical functions 5.4. Plain sensitive data should not be stored on-chain 5.5. Modification of array state should not be done by value 5.6. State variable should not be used without being initialized

Testing Category	Testing Items
6. External Components	6.1. Unknown external components should not be invoked 6.2. Funds should not be approved or transferred to unknown accounts 6.3. Reentrant calling should not negatively affect the contract states 6.4. Vulnerable or outdated components should not be used in the smart contract 6.5. Deprecated components that have no longer been supported should not be used in the smart contract 6.6. Delegatecall should not be used on untrusted contracts
7. Arithmetic	7.1. Values should be checked before performing arithmetic operations to prevent overflows and underflows 7.2. Explicit conversion of types should be checked to prevent unexpected results 7.3. Integer division should not be done before multiplication to prevent loss of precision
8. Denial of Services	8.1. State changing functions that loop over unbounded data structures should not be used 8.2. Unexpected revert should not make the whole smart contract unusable 8.3. Strict equalities should not cause the function to be unusable
9. Best Practices	9.1. State and function visibility should be explicitly labeled 9.2. Token implementation should comply with the standard specification 9.3. Floating pragma version should not be used 9.4. Builtin symbols should not be shadowed 9.5. Functions that are never called internally should not have public visibility 9.6. Assert statement should not be used for validating common conditions

3.3. Risk Rating

OWASP Risk Rating Methodology (https://owasp.org/www-community/OWASP_Risk_Rating_Methodology) is used to determine the severity of each issue with the following criteria:

- **Likelihood**: a measure of how likely this vulnerability is to be uncovered and exploited by an attacker
- **Impact**: a measure of the damage caused by a successful attack

Both likelihood and impact can be categorized into three levels: **Low**, **Medium**, and **High**.

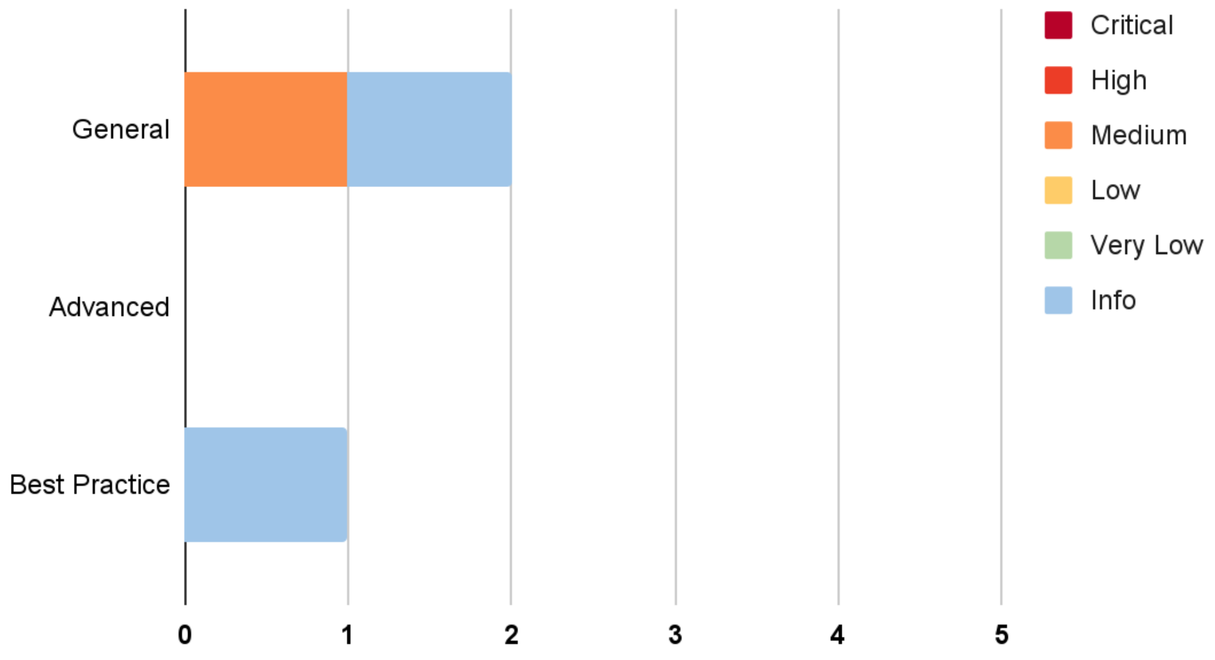
Severity is the overall risk of the issue. It can be categorized into five levels: **Very Low**, **Low**, **Medium**, **High**, and **Critical**. It is calculated from the combination of likelihood and impact factors using the matrix below. The severity of findings with no likelihood or impact would be categorized as **Info**.

Likelihood / Impact	Low	Medium	High
Low	Very Low	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical

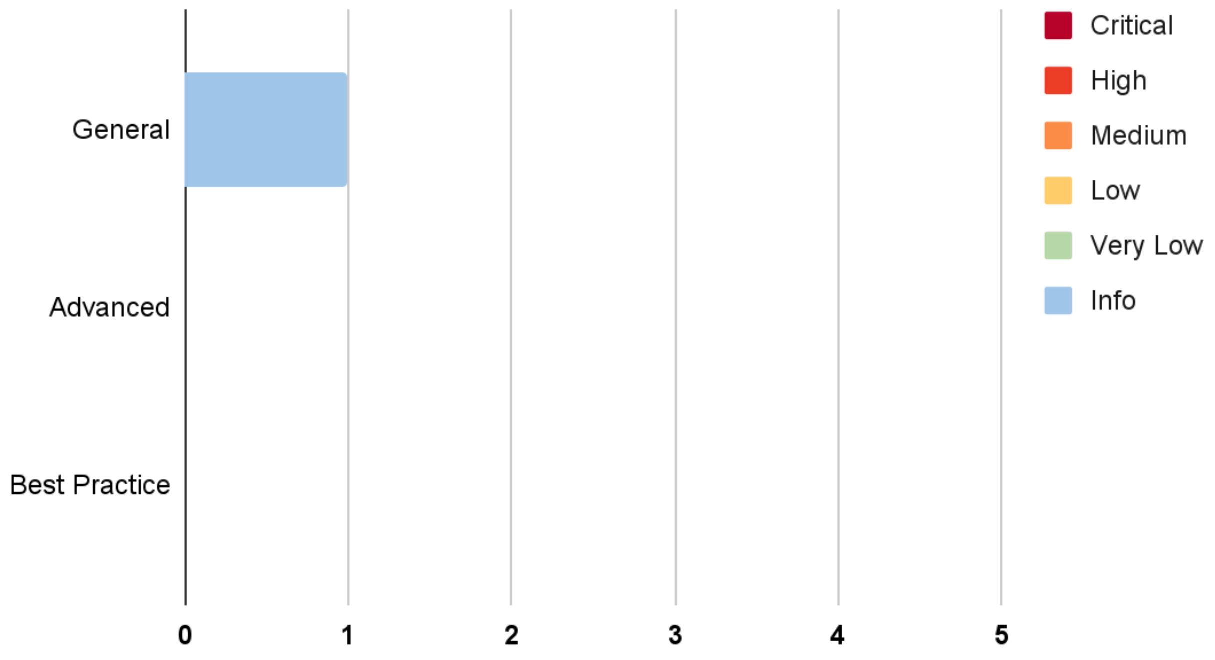
4. Summary of Findings

The following charts show the number of the issues found during the assessment and the issues acknowledged in the reassessment, categorized into three categories: **General**, **Advanced**, and **Best Practice**.

Assessment:



Reassessment:



The statuses of the issues are defined as follows:

Status	Description
Resolved	The issue has been resolved and has no further complications.
Resolved *	The issue has been resolved with mitigations and clarifications. For the clarification or mitigation detail, please refer to Chapter 5.
Acknowledged	The issue's risk has been acknowledged and accepted.
No Security Impact	The best practice recommendation has been acknowledged.

The information and status of each issue can be found in the following table:

ID	Title	Category	Severity	Status
IDX-001	Centralized Control of State Variable	General	Medium	Resolved *
IDX-002	Outdated Compiler Version	General	Info	No Security Impact
IDX-003	Improper Function Visibility	Best Practice	Info	Resolved

* The mitigations or clarifications by Token X can be found in Chapter 5.

5. Detailed Findings Information

5.1. Centralized Control of State Variable

ID	IDX-001
Target	AllowlistRegistryProxy
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** Medium **Impact:** High The controlling authorities can change the critical state variables to gain additional profit. Thus, it is unfair to the other users. **Likelihood:** Low There is nothing to restrict the changes from being done; however, this action can only be done by the authorized party which is a multisig wallet.
Status	Resolved * The centralized functions are required to comply with the regulator's guidelines for protecting users' funds, also known as clawback mechanism. As a result, the Token X team has mitigated this issue by delegating control to a multi-sig wallet for those privileged functions to ensure that all the critical actions are properly reviewed according to the regulation.

5.1.1. Description

Critical state variables can be updated at any time by the controlling authorities. Changes in these variables can cause impacts to the users, so the users should accept or be notified before these changes are effective.

However, there is currently no constraint to prevent the authorities from modifying these variables without notifying the users. For example, the platform owner can call the `removeRegistry()` function to remove the `registry` address and then the function that gets information from the registry will not return allowed users in the registry. resulting in the users not being considered allowed anymore.

The controllable privileged state update functions are as follows:

File	Contract	Function	Modifier
AllowlistRegistryProxy.sol (L:171)	AllowlistRegistryProxy	addRegistry()	onlyOwner
AllowlistRegistryProxy.sol (L:187)	AllowlistRegistryProxy	removeRegistry()	onlyOwner

5.1.2. Remediation

In the ideal case, the critical state variables should not be modifiable to keep the integrity of the smart contract. However, if modifications are needed, Inspex suggests implementing a community-run smart contract governance to control the use of these functions.

If removing the functions or implementing the smart contract governance is not possible, Inspex suggests mitigating the risk of this issue by using a timelock mechanism to delay the changes for a reasonable amount of time, at least 24 hours.

However, if the timelock is used as a mitigation, the `onlyOwner` modifier should be changed to another operation role such as `onlyOperator` to prevent the timelock effect from applying to other functions with the `onlyOwner` modifier, The following function are listed in the table that are affected if the contract transfers ownership to timelock.

File	Contract	Function	Modifier
AllowlistRegistryProxy.sol (L:203)	AllowlistRegistryProxy	pauseRegistry()	onlyOwner
AllowlistRegistryProxy.sol (L:218)	AllowlistRegistryProxy	unpauseRegistry()	onlyOwner
AllowlistRegistryProxy.sol (L:238)	AllowlistRegistryProxy	addBlacklist()	onlyOwner
AllowlistRegistryProxy.sol (L:252)	AllowlistRegistryProxy	removeBlacklist()	onlyOwner

5.2. Outdated Compiler Version

ID	IDX-002
Target	AllowlistRegistryProxy Blacklistable Storage Ownable
Category	General Smart Contract Vulnerability
CWE	CWE-1104: Use of Unmaintained Third Party Components
Risk	**Severity:** Info **Impact: None** **Likelihood: None**
Status	No Security Impact Due to the limitation of the blockchain, which does not support the opcode PUSH0 (0x5f) in the Shanghai upgrade, the Token X team decided to use the Solidity compiler version 0.8.19. However, there is no security impact on the platform.

5.2.1. Description

The Solidity compiler versions specified in the smart contracts were outdated (https://soliditylang.org/blog/2023/07/19/solidity-0.8.21-release-announcement). As the compilers are regularly updated with bug fixes and new features, the latest stable compiler version should be used to compile the smart contracts for best practice.

AllowlistRegistryProxy.sol

```
// SPDX-License-Identifier: GPL-2.0-or-later
// TokenX Contracts v1.0.0 (contracts/AllowlistRegistryProxy.sol)
pragma solidity 0.8.19;
```

The table below represents the contracts that apply the outdated Solidity compiler version.

File	Version
AllowlistRegistryProxy.sol (L:3)	0.8.19
Blacklistable.sol (L:3)	0.8.19
Storage.sol (L:3)	0.8.19
Ownable.sol (L:3)	0.8.19

5.2.2. Remediation

Inspex suggests upgrading the Solidity compiler to the latest stable version (https://github.com/ethereum/solidity/releases). At the time of audit, the latest stable version of the Solidity compiler in major 0.8 is 0.8.21, for example:

AllowlistRegistryProxy.sol

```
// SPDX-License-Identifier: GPL-2.0-or-later
// TokenX Contracts v1.0.0 (contracts/AllowlistRegistryProxy.sol)
pragma solidity 0.8.21;
```

Please note that deploying the compiled contract with Solidity compiler version 0.8.20 or later could encounter some problems on the blockchain that do not support the opcode PUSH0 (0x5f) in the Shanghai upgrade. In this case, downgrade and use the Solidity compiler 0.8.19 version instead.

5.3. Improper Function Visibility

ID	IDX-003
Target	AllowlistRegistryProxy
Category	Smart Contract Best Practice
CWE	CWE-710: Improper Adherence to Coding
Risk	**Severity:** Info **Impact: None** **Likelihood: None**
Status	Resolved The Token X team has resolved this issue by changing the function's visibility to external in commit `8aa1f63767c40ca8e9f1b94dee51d0c5ea06a082`.

5.3.1. Description

Public functions that are never called internally by the contract itself should have external visibility. This improves the readability of the contract, allowing clear distinction between functions that are externally used and functions that are also called internally.

AllowlistRegistryProxy.sol

```
function pauseRegistry(address registry) public onlyOwner
existRegistry(registry) unpausedRegistry(registry) {
      Storage._registryInfo[registry].paused = true;

      emit RegistryPaused(registry);
}
```

The following table contains all functions that have public visibility and are never called from any internal function.

File	Contract	Function
AllowlistRegistryProxy.sol (L:94)	AllowlistRegistryProxy	version()
AllowlistRegistryProxy.sol (L:101)	AllowlistRegistryProxy	name()
AllowlistRegistryProxy.sol (L:114)	AllowlistRegistryProxy	registries()
AllowlistRegistryProxy.sol (L:122)	AllowlistRegistryProxy	totalRegistry()
AllowlistRegistryProxy.sol (L:131)	AllowlistRegistryProxy	getRegistryInfo()

AllowlistRegistryProxy.sol (L:141)	AllowlistRegistryProxy	getAccountProviderInfo()
AllowlistRegistryProxy.sol (L:171)	AllowlistRegistryProxy	addRegistry()
AllowlistRegistryProxy.sol (L:187)	AllowlistRegistryProxy	removeRegistry()
AllowlistRegistryProxy.sol (L:203)	AllowlistRegistryProxy	pauseRegistry()
AllowlistRegistryProxy.sol (L:218)	AllowlistRegistryProxy	unpauseRegistry()
AllowlistRegistryProxy.sol (L:274)	AllowlistRegistryProxy	isAllowlist()

5.3.2. Remediation

Inspex suggests changing all functions' visibility to external if they are not called from any internal function, as shown in the following example:

AllowlistRegistryProxy.sol

```
203 function pauseRegistry(address registry) external onlyOwner
    existRegistry(registry) unpausedRegistry(registry) {
204       Storage._registryInfo[registry].paused = true;
205 
206       emit RegistryPaused(registry);
207 }
```

6. Appendix

6.1. About Inspex

inspex CYBERSECURITY PROFESSIONAL SERVICE

Inspex is formed by a team of cybersecurity experts highly experienced in various fields of cybersecurity. We provide blockchain and smart contract professional services at the highest quality to enhance the security of our clients and the overall blockchain ecosystem.

Follow Us On:

Website	https://inspex.co
Twitter	@InspexCo
Facebook	https://www.facebook.com/InspexCo
Telegram	@inspex_announcement

inspex

CYBERSECURITY PROFESSIONAL SERVICE