0x Protocol

Governance

Security Assessment & Correctness

April 6th

Audited By:
Angelos Apostolidis
angelos.apostolidis@ourovoros.io
Georgios Delkos
georgios.delkos@ourovoros.io

Overview

Project Summary

Project Name	0x Protocol - Decentralized Governance
Website	0x Protocol
Description	Decentralized governance for the 0x Protocol and Treasury
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	Pre-audit: bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6 Post-audit: 3574ea5b27b9ccc4fca612246d0032178703eef3

Audit Summary

Delivery Date	April 6th
Method of Audit	Static Analysis, Manual Review

Vulnerability Summary

● **Total Issues**	7
● **Total Major**	0
● **Total Minor**	0
● **Total Informational**	7

Executive Summary

The primary objective of this system is to decentralize control over the `0x Protocol` and its Treasury through a series of interconnected smart contracts. These contracts facilitate the administration of both the decision-making process and the platform's financial resources.

The system consists of two distinct governors: the `ZeroExProtocolGovernor` and `ZeroExTreasuryGovernor`, each with specific responsibilities. The `ZeroExProtocolGovernor` manages the guidelines and enhancements for the `0x Protocol`, while the `ZeroExTreasuryGovernor` oversees the platform's financial assets. Both governors operate in conjunction with their respective time-lock contracts (`ZeroExTimelock`), ensuring that system changes are executed after a predefined delay.

Two key features of this system is delegation and quadratic voting for Treasury-related decisions. Delegation allows users to transfer their voting power to another party if desired. Quadratic voting ensures that as a voter accumulates more tokens, their influence gradually diminishes, preventing individuals or organizations from dominating the decision-making process.

Moreover, the system incorporates a Security Council, consisting of trusted members with the authority to cancel proposals related to the treasury or protocol governors and, if necessary, revert the Protocol to a previous version. The Security Council serves as a safety mechanism, maintaining the platform's security and preventing potentially harmful proposals from being executed.

The code quality and security of the involved smart contracts are essential aspects of this system. The development team has devoted considerable effort to ensure these contracts are designed and implemented following best practices, utilizing industry-standard tools and libraries such as `OpenZeppelin`. Regular due diligence and thorough testing are conducted to identify and address potential vulnerabilities, further enhancing the system's overall security. This rigorous approach to code quality and safety demonstrates the team's commitment to providing a reliable, trustworthy, and robust platform that instills confidence in its users and promotes the long-term success of the `0x Protocol` ecosystem.

It is crucial to acknowledge the `ZeroEx` smart contract, which primarily routes calls to the appropriate per-function implementation contracts through its fallback. The `ZeroEx` contract's upgradeable proxy mechanism plays a vital role in the system's flexibility and adaptability. This design choice allows individual functions or features to be upgraded without disrupting the overall system's stability. Employing a per-function proxy pattern makes the smart contract more efficient and easier to maintain over time. The rolling release model further fosters a dynamic and responsive system capable of swiftly adapting to evolving requirements or security considerations.

In conclusion, this system, built on a set of smart contracts, aims to decentralize the governance of the `0x Protocol` and its Treasury. It empowers token holders to actively participate in decision-making and implements safeguards to prevent a single party from exerting excessive control. By incorporating features like quadratic voting and the Security Council, the system maintains a balanced power distribution among community members, fostering a fair and secure environment for the ongoing development and management of the `0x Protocol`. Through these measures, the platform promotes transparency, inclusivity, and fairness, cultivating a robust and secure ecosystem that benefits all participants within the `0x Protocol` community.

Files In Scope

Contract	Location
CallWithGas.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/CallWithGas.sol
IZeroExGovernor.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/IZeroExGovernor.sol
IZeroExVotes.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/IZeroExVotes.sol
SecurityCouncil.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/SecurityCouncil.sol
ZeroExProtocolGovernor.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/ZeroExProtocolGovernor.sol
ZeroExTimelock.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/ZeroExTimelock.sol
ZeroExTreasuryGovernor.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/ZeroExTreasuryGovernor.sol
ZeroExVotes.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/ZeroExVotes.sol
ZRXWrappedToken.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/governance/src/ZRXWrappedToken.sol
ZeroEx.sol	https://github.com/0xProject/protocol/tree/bcbfbf

	a16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/zero-ex/contracts/src/ZeroEx.sol
SimpleFunctionRegistryFeature.sol	https://github.com/0xProject/protocol/tree/bcbfbfa16c2ec98e64cd1f2f2f55a134baf3dbf6/contracts/zero-ex/contracts/src/features/SimpleFunctionRegistryFeature.sol
DeployGovernance.s.sol	https://github.com/0xProject/protocol/tree/3574ea5b27b9ccc4fca612246d0032178703eef3/contracts/governance/script/DeployGovernance.s.sol
GovernanceE2E.t.sol	https://github.com/0xProject/protocol/tree/3574ea5b27b9ccc4fca612246d0032178703eef3/contracts/governance/test/integration/GovernanceE2E.t.sol

Findings

ID	Title	Type	Severity
F-1	Unlocked Compiler Version	Coding Style	informational
F-2	Unused Returned Value	Inconsistency	informational
F-3	Explicit Variable Return	Coding Style	informational
F-4	Unused Function	Dead Code	informational
F-5	`modifier` Optimization	Gas Optimization	informational
F-6	`return` Statement Optimization	Gas Optimization	informational
F-7	Input Sanity Check	Inconsistency	informational

Notes

ID	Title
N-1	System Upgradeability
N-2	System Migration

F-1: Unlocked Compiler Version

Type	Severity	Location
Coding Style	● informational	ZeroExProtocolGovernor L19, ZeroExTimelock L19, ZeroExTreasuryGovernor L19, ZeroExVotes L19, ZRXWrappedToken L19

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 differing compiler version numbers. 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 `v0.8.19` the contract should contain the following line:\n\n `Solidity\npragma solidity =0.8.19;\n`

Alleviation:

The development team acknowledges this exhibit and notes that, although the solc version has not been locked in the contracts, we have instead configured the Foundry settings to specify the version as solc = '0.8.19'.

F-2: Unused Returned Value

Type	Severity	Location
Inconsistency	informational	ZRXWrappedToken L74-L82, L94-L98, L154-L162

Description:

The linked invocations do not check or utilize the values returned by their respective function calls, leading to potential inconsistencies or inefficiencies in the code.

Recommendation:

It is recommended that the returned values are either appropriately utilized within the logic of the smart contracts or removed from the function declarations, ensuring cleaner and more efficient code execution. This will help maintain best practices and minimize any potential issues stemming from unused or unchecked return values.

Alleviation:

The development team acknowledges this exhibit, explaining that the intention behind this approach is to save an additional 100 gas in cases where a function has a return value, as opposed to returning nothing.

F-3: Explicit Variable Return

Type	Severity	Location
Coding Style	informational	ZeroExVotes L256

Description:

The linked statement returns a local variable explicitly, which could lead to decreased readability of the code.

Recommendation:

It is recommended to declare and utilize a named return variable at L255, which will improve code clarity and maintainability. By using a named return variable, developers can better understand the purpose and context of the returned value, leading to more efficient debugging and future code enhancements.

Alleviation:

The development team has addressed and resolved this issue.

F-4: Unused Function

Type	Severity	Location
Dead Code	informational	CallWithGas L27-L80, SecurityCouncil L56-L59

Description:

The linked functions are not being used or referenced throughout the entire project. This presence of dead code can lead to confusion and increase the difficulty of maintaining the code.

Recommendation:

It is recommended to remove the unused code to enhance code readability and maintainability. By eliminating dead code, developers can focus on the functionality that is actually relevant to the project, reducing the likelihood of introducing errors or overlooking important aspects during future updates.

Alleviation:

The development team has acknowledged this exhibit and provided the following comments: The `ejectSecurityCouncil` function is intended for potential future use, and a test for this function will be enabled concurrently with its logic implementation (see `testFailSecurityCouncilAreEjectedAfterCancellingAProposal`). As for the `CallWithGas`, the team accepts that it has an unused function, but they prefer to maintain a fully audited on-chain copy of the library.

F-5: `modifier` Optimization

Type	Severity	Location
Gas Optimization	informational	SecurityCouncil L28-L31, ZeroExVotes L46-L49

Description:

The linked `modifier`s present an opportunity for further optimization, which can lead to reduced gas consumption during contract execution.

Recommendation:

It is recommended to move the `require` statements from the modifier to a newly declared `private` function and then use that function within the `modifier`. By doing so, you reduce gas costs, making the smart contracts more efficient and cost-effective for users interacting with them.

Alleviation:

The development team has addressed and resolved this issue.

F-6: `return` Statement Optimization

Type	Severity	Location
Gas Optimization	informational	ZeroExVotes L75, L83

Description:

The linked `return` statements present an opportunity for further optimization, which can lead to reduced gas consumption during contract execution.

Recommendation:

It is recommended to wrap the linked statements in an `unchecked` block, considering that the local variable is bound within the values of the `uint96` type. By doing so, you can eliminate unnecessary overflow checks and reduce gas costs, making the smart contracts more efficient and cost-effective for users interacting with them.

Alleviation:

The development team has addressed and resolved this issue.

F-7: Input Sanity Check

Type	Severity	Location
Inconsistency	informational	ZeroExTimelock L43-L49

Description:

The linked function omits a check on the length of the input array. Addressing this issue can help ensure that the function handles edge cases appropriately and operates consistently under various conditions.

Recommendation:

It is recommended to add a `require` statement that checks whether the length of the `target` array is not zero. By implementing this check, you can prevent potential issues arising from an empty array and ensure that the function operates as expected.

Alleviation:

The development team has addressed and resolved this issue.

N-1: System Upgradeability

The `ZeroEx` smart contract employs an upgradeable proxy pattern, focusing on the extend and rollback functions for managing feature implementations. The `extend()` function allows registering a new function (`selector`) and implementation (`address`), while maintaining a history of past implementations. This mechanism provides the ability to upgrade individual functions or features without affecting the overall system's stability.

On the other hand, the `rollback()` function enables reverting a function implementation to a prior version in its history. This is particularly useful when addressing vulnerabilities or rolling back to a more stable version of the code. By combining both extend and rollback functions, the `ZeroEx` contract ensures a more flexible and adaptable system, capable of quickly adapting to changes in requirements or security considerations.

N-2: System Migration

The deployment process for the new version of the decentralized governance is designed with security as its top priority. The development team has utilized best practices and industry-standard tools and libraries to ensure the integrity and reliability of the contracts. The deployment process is well-structured and carefully crafted, implementing a series of initialization steps and role assignments to set up the contracts accurately and securely. This rigorous attention to detail and commitment to high standards in the deployment process is critical in providing a secure and trustworthy ecosystem.

Disclaimer

Reports made by Ourovoros are not to be considered as a recommendation or approval of any particular project or team. Security reviews made by Ourovoros for any project or team are not to be taken as a depiction of the value of the “product” or “asset” that is being reviewed.

Ourovoros reports are not to be considered as a guarantee of the bug-free nature of the technology analyzed and should not be used as an investment decision with any particular project. They represent an extensive auditing 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.

Each company and individual is responsible for their own due diligence and continuous security. Our goal is to help reduce the attack parameters and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claim any guarantee of security or functionality of the technology we agree to analyze.