NFTProxyWith ERC20Fee

Smart Contract Audit Report Prepared for Token X

TOKEN X

Date Issued:	Oct 17, 2023
Project ID:	AUDIT2023018
Version:	v1.0
Confidentiality Level:	Public

inspeX
CYBERSECURITY PROFESSIONAL SERVICE

Report Information

Project ID	AUDIT2023018
Version	v1.0
Client	Token X
Project	NFTProxyWithERC20Fee
Auditor(s)	Phitchakorn Apiratisakul Kongkit Chatchawanhirun
Author(s)	Phitchakorn Apiratisakul
Reviewer	Natsasit Jirathammanuwat
Confidentiality Level	Public

Version History

Version	Date	Description	Author(s)
1.0	Oct 17, 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 NFTProxyWithERC20Fee smart contracts on Oct 3, 2023. During the audit, Inspex team examined all smart contracts and the overall operation within the scope to understand the overview of NFTProxyWithERC20Fee 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 low, and 1 info-severity issues. With the project team's prompt response, 1 low and 1 info-severity issues were acknowledged by the team. Therefore, Inspex trusts that NFTProxyWithERC20Fee smart contracts have sufficient protections to be safe for public use. However, as the source code is currently not publicly available, there is a potential risk that the smart contracts deployed on the blockchain may not be identical to the audited smart contracts. This discrepancy could result in introducing security vulnerabilities or unintended behaviors that were not identified during the audit process. It is of importance to recognize that interacting with an unverified smart contract may lead to the potential loss of funds. In this case, the hash of the deployed smart contract bytecode should be compared with the hash of the audited smart contract bytecode to ensure that the deployed smart contract is identical to the audited smart contract before interacting with them. In the long run, Inspex suggests resolving all issues found in this report.

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 NFTProxyWithERC20Fee contract is responsible for collecting fees from users when they transfer NFTs. The collected fee token and amount can be set individually for each NFT. Additionally, the owner has the privilege to withdraw the collected fees from the FeeVault contract.

Scope Information:

Project Name	NFTProxyWithERC20Fee
Website	https://tokenx.finance/
Smart Contract Type	Ethereum Smart Contract
Chain	Token X (TKX)
Programming Language	Solidity
Category	NFT

Audit Information:

Audit Method	Whitebox
Audit Date	Oct 3, 2023
Reassessment Date	Oct 17, 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 smart contracts with the following bytecodes were audited and reassessed by Inspex in detail:

Initial Audit:

Contract	Bytecode SHA256 Hash
NFTProxyWithERC20Fee	8cde7b8e6f29b6da1defd13f74358290f566cd92130f3eec2f03472a14d46d13
FeeVault	17a395b3f5253059565efd8cf570712846f4cbcbb95788ac824b05b14807cd0c

Reassessment Audit:

Contract	Bytecode SHA256 Hash
NFTProxyWithERC20Fee	8cde7b8e6f29b6da1defd13f74358290f566cd92130f3eec2f03472a14d46d13
FeeVault	17a395b3f5253059565efd8cf570712846f4cbcbb95788ac824b05b14807cd0c

compiler_config.json

```
{
    "language": "Solidity",
    "settings": {
        "optimizer": {
            "enabled": true,
            "runs": 200
        },
    }
}
```

As the Token X team has decided not to publish the source code to protect their intellectual property, the users should compare the bytecode hashes with the smart contracts deployed before interacting with them to make sure that they are the same with the contracts audited.

2.3. Security Model

2.3.1 Trust Modules

The NFTProxyWithERC20Fee has a privileged role with the authority to mutate the critical state variables of the contract. Changes to these state variables significantly impact the contract's functionality. The privileged roles and their corresponding privileged functions are enumerated as follows:

- The `owner` address of the `NFTProxyWithERC20Fee` contract can perform the following actions:
  - Set the fee amount and fee token for a specific NFT; this privileged operation could be used to gain more fees depending on the user's approved token amount
  - Set the fee receiver to any address; all fees are transferred directly to the fee receiver address
  - Ownership management; transfer ownership to any address, set **owner** address to zero
- The **owner** address of the `FeeVault` contract can perform the following actions:
  - Withdraw fee from the contract
  - Set the fee receiver to any address
  - Ownership management; transfer ownership to any address, set `owner` address to zero
- The `feeReceiver` address of the `FeeVault` contract can withdraw fees from the contract.

2.3.2 Trust Assumptions

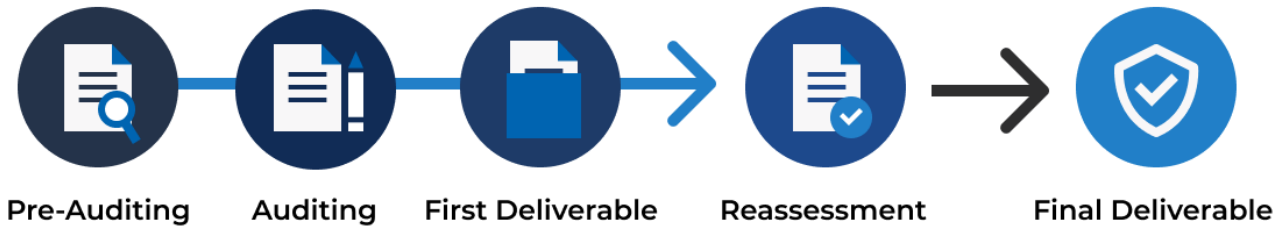
In the NFTProxyWithERC20Fee, the protocol's privileged role, has the ability to change the critical state variables of the contract, they were assumed to be trusted. Acknowledging these trust assumptions is important, as it introduces substantial risks to the platform. Trust assumptions include, but are not limited to:

- All privileged roles perform the privileged function with good will
- The `owner` address is trusted to set the critical state variables of the `NFTProxyWithERC20Fee` contract

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://docs.inspex.co/smart-contract-security-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**.

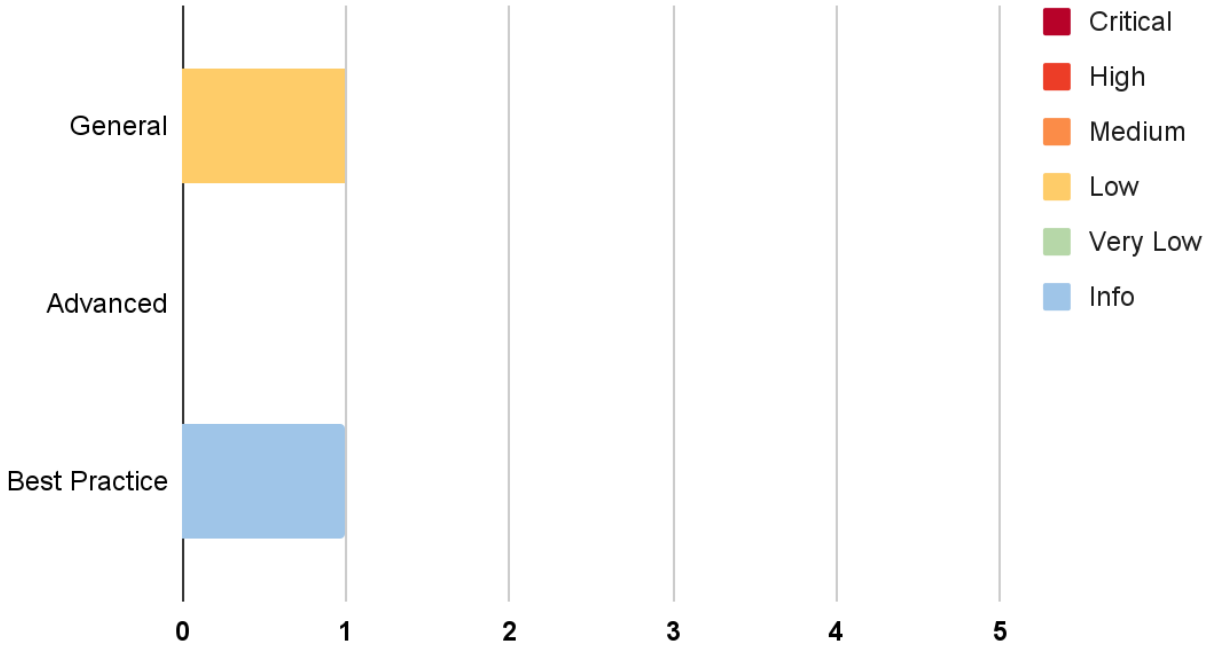
Severity is the overall risk of the issue. It can be categorized into five levels: **Very Low**, **Low**, **Medium**, **High**, and **Critical**. It is calculated from the combination of likelihood and impact factors using the matrix below. The severity of findings with no likelihood or impact would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

Impact \ Likelihood	Low	Medium	High
Low	Very Low	Low	Medium
Medium	Low	Medium	High
High	Medium	High	Critical

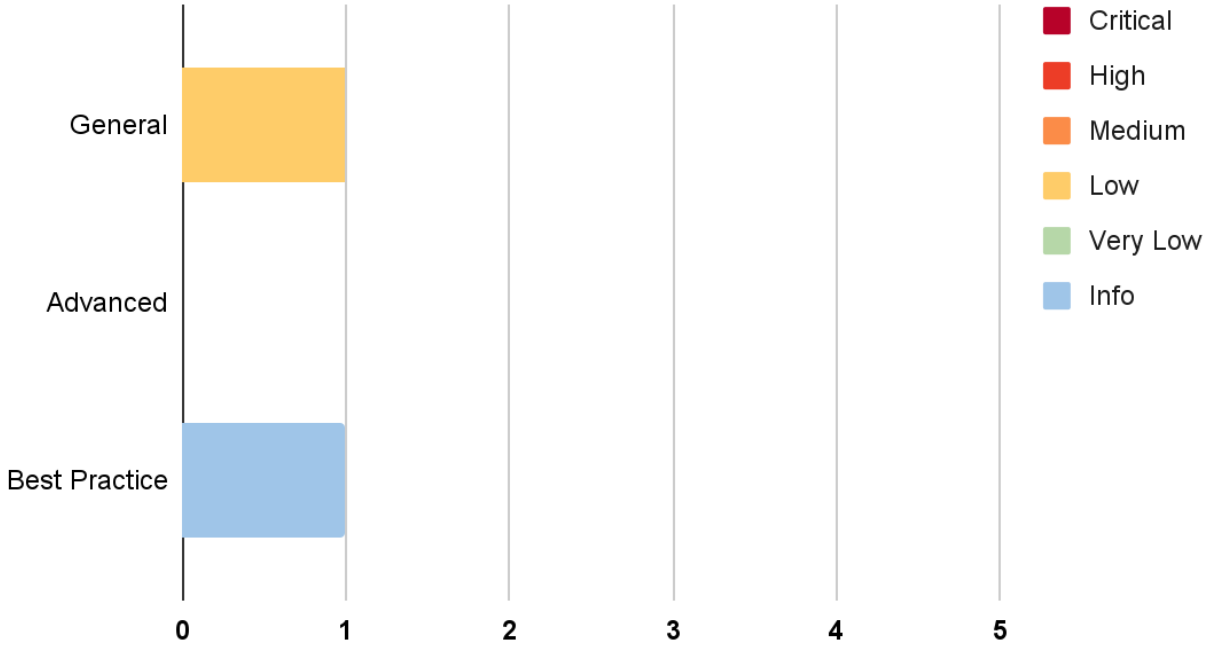
4. Summary of Findings

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 Authority Control	General	Low	Acknowledged
IDX-002	Inexplicit Solidity Compiler Version	Best Practice	Info	Acknowledged

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

5. Detailed Findings Information

5.1. Centralized Authority Control

ID	IDX-001
Target	NFTProxyWithERC20Fee
Category	General Smart Contract Vulnerability
CWE	CWE-284: Improper Access Control
Risk	**Severity:** Low **Impact:** Medium The controlling authorities have the ability to manipulate critical state variables, altering the contract's behavior and potentially increasing their profits. This manipulation can lead to malfunctions and render the system unreliable, which is unfair to platform users. **Likelihood:** Low The private key of the controlling authorities is unlikely to be compromised. Nevertheless, if it were to be compromised, there are no restrictions preventing unauthorized changes from being made.
Status	**Acknowledged** The Token X team has acknowledged this issue. However, we suggest the platform users monitor the state value of the contract before executing the platform functions.

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 mechanism in place to prevent the authorities from altering these variables without notifying the users. In the event that the private key of the controlling authorities is compromised by an attacker, they can manipulate the contract's behavior and profit without any restrictions.

The controllable privileged state update function is as follows:

File	Function	Modifier
NFTProxyWithERC20Fee.sol (L: 37)	setFeeAmount()	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 or using the multi-signature contract to reduce the risk of unauthorized or malicious alterations to the smart contract.

5.2. Inexplicit Solidity Compiler Version

ID	IDX-002
Target	NFTProxyWithERC20Fee FeeVault
Category	Smart Contract Best Practice
CWE	CWE-1104: Use of Unmaintained Third Party Components
Risk	**Severity:** Info **Impact: None** **Likelihood: None**
Status	No Security Impact The Token X team has acknowledged this issue. However, the inexplicit compiler version has no security impact.

5.2.1. Description

The Solidity compiler versions declared in the smart contracts were not explicit. Each compilation may be done using different compiler versions, which may potentially result in compatibility issues.

NFTProxyWithERC20Fee.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
```

The following table contains all targets which the Inexplicit compiler version is declared.

File	Version
NFTProxyWithERC20Fee.sol (L: 2)	^0.8.0
FeeVault.sol (L: 2)	^0.8.0

5.2.2. Remediation

Inspex suggests fixing the Solidity compiler to the latest stable version. At the time of the audit, the latest stable version of Solidity compiler in major 0.8 is v0.8.21. (https://github.com/ethereum/solidity/releases)

NFTProxyWithERC20Fee.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity 0.8.21;
```

For chains that may not be compatible with Solidity compiler version 0.8.21, Inspex suggests using Solidity compiler version 0.8.19 instead, as Solidity compiler version 0.8.20 or later introduces the PUSH0 (0x5f) opcode, which some chains have not yet included. (https://github.com/ethereum/solidity/releases/tag/v0.8.20)

NFTProxyWithERC20Fee.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity 0.8.19;
```

6. Appendix

6.1. About Inspex

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