Xtransfer Network

Smart Contract Audit Report

IMMUNE BYTES

Audits

January 1, 2021

IMMUNE BYTES

Introduction

1. About Xtransfer Network

Xtransfer Network is one of a kind Payment and fund transfer system that permits advanced resources for collaboration with the actual world. Xtransfer is something other than just another swap for crypto and non-crypto related.

Visit https://xtransfernetwork.com/ to know more.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 15+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

Xtransfer team has provided documentation for the purpose of conducting the audit. The documents are:

1. https://xtransfernetwork.com/WhitePaper_xTransfer.pdf

IMMUNE BYTES

Audit Process & Methodology

ImmueBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: Xtransfer Network
- Languages: Solidity(Smart contract)
- Github commit hash for audit: 04fbd88c70456797d957c9f762c3bf0e57e3abe7

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include:
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level References

Every issue in this report was assigned a severity level from the following:

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	4
Closed	-	-	-

High severity issues

No Issues Found

Medium severity issues

No Issues Found

Low severity issues

1. **Workaround to skip Transfer Fee.**
   The **XTRANSFER** Token has a fee deduction mechanism implemented which deducts **0.9%** fee on every token transfer. However, there is a workaround to skip the fee deduction on token transfer. The fee deduction mechanism is only present in the **transfer**() function and not in **transferFrom**(). So a user can first give himself sufficient enough allowance and then can use the **transferFrom**() function to work the same as **transfer**() function but with No Fee.

   Recommendation:
   Consider adding a fee mechanism to **transferFrom**() as well if you do not want the above-mentioned scenario. But be aware that other Ethereum projects expect that the token they are interacting with must not deduct any fee on token transfers.

2. **Use *_msgSender*() instead of *msg.sender***
   Throughout the XTRANSFER Token contract **_msgSender**() is used to access the caller's address instead of **msg.sender** except one place, in the **constructor**() **Line 229 msg.sender** is used. To maintain consistency it is always recommended to use exact same conventions throughout the contract.

   Recommendation:
   Consider replacing **msg.sender** with **_msgSender**() inside the **constructor**().

3. ***addressTreasury* should be declared as a *public constant* variable.**
   In **XTRANSFER** Token contract **addressTreasury** variable is declared which collects the fee from token transfers. Since the variable is declared with no visibility specifier, it is marked as a **private** variable. There is no easy way for off-chain elements (like frontends or etherscan) to read **private** variables of a smart contract.

Recommendation:
Consider marking the variable as **public constant** or create a getter function for the same.

4. **Use *indexed* parameters in events.**
   The **AddToWhitelist** and **RemoveFromWhitelist** events in **XTRANSFER** Token contract use an address parameter that is not marked as **indexed**. Indexed variable comes handy in filtering out events in the off-chain elements.
   More details on the use of **indexed** variables can be found here https://docs.soliditylang.org/en/v0.7.4/contracts.html?highlight=indexed#events.

Unit Test

No unit tests were provided by the XTRANSFER team.

Recommendation:
Our team suggests that the developers should write extensive test cases for the contracts.

Coverage Report

Coverage report cannot be generated without unit test cases.

Recommendation:
We recommend 100% line and branch coverage for unit test cases.

Recommendations

1. **The contract should follow the proper Solidity Style Guide.**
   Consider following the Solidity guide. It is intended to provide coding/naming conventions for writing solidity code.

   Recommendation:
   Solidity style guide can be found using the following link https://docs.soliditylang.org/en/v0.6.12/style-guide.html

Concluding Remarks

While conducting the audits of the Xtransfer smart contract, it was observed that the contracts contain only Low severity issues, along with a few areas of recommendations.

Our auditors suggest Low severity issues should be resolved by Xtransfer developers. Resolving the areas of recommendations are up to Xtransfer's discretion. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Xtransfer Network platform or its product neither this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes Pvt Ltd.