SUPREMACY

Smart Contract Security Audit Report

Prepared for Fusionist

Prepared by Supremacy

August 26, 2024

Contents

1 Introduction

Given the opportunity to review the related codebase of the Fusionist’s Cross-chain bridge, we outline in the report our systematic approach to evaluate potential security issues in the smart contract(s) implementation, and provide recommendations for improvement. Our results show that the given version of smart contracts can be further improved due to the presence of several issues related to either security. This document outlines our audit results.

1.1 About Client

Fusionist is a game universe with collectable NFTs comprising of three types of gameplay: Colonize (building simulation), Conquer (turn-based tactics) and Unite (explore, expand, exploit, exterminate). Players take on the role of Mech Commanders, running their own planet(s), collecting rare resources, upgrading technology, scanning blueprints to manufacture Mechs, and constructing production pipelines. They battle through PvP and PvE battles, build starship fleets for inter-planetary warfare, and conquer the galaxy. Endurance is Fusionist's mainnet, social & game oriented infrastructure layer.

Item	Description
Client	Fusionist
Type	Smart Contract
Languages	Solidity
Platform	EVM-compatible

1.2 Audit Scope

In the following, we show the on-chain smart contract address used in this security audit:

- OriginalTokenBridge: 0xf3310e3f0D46FF5EE7daB69C73452D0ff3979Bed
- RemoteTokenBridge: 0x9d0698D9a6D01f1409AC56c5aDF62547b0055915

1.3 Changelogs

Version	Date	Description
0.1	August 23, 2024	Initial Draft
1.0	August 26, 2024	Final Release

1.4 About Us

Supremacy is a leading blockchain security firm, composed of industry hackers and academic researchers, provide top-notch security solutions through our technology precipitation and innovative research.

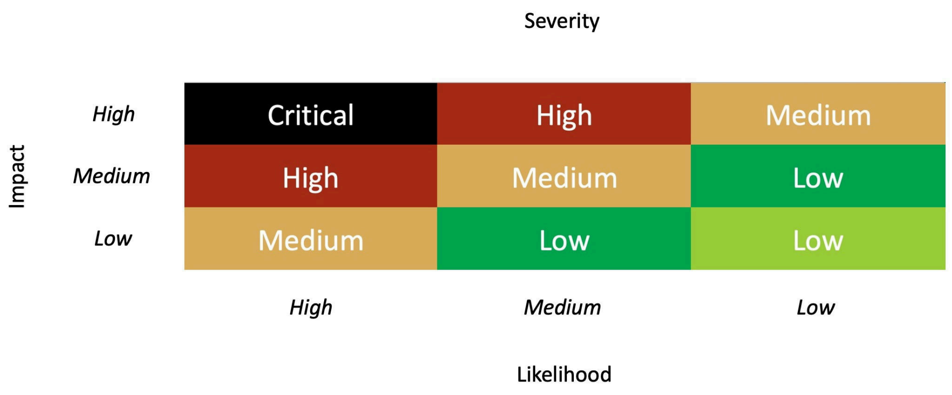
We are reachable at Twitter (https://twitter.com/SupremacyHQ), or Email (contact@supremacy.email).

1.5 Terminology

For the purpose of this assessment, we adopt the following terminology. To classify the severity of our findings, we determine the likelihood and impact (according to the CVSS risk rating methodology).

- Likelihood represents the likelihood of a finding to be triggered or exploited in practice
- Impact specifies the technical and business-related consequences of a finding
- Severity is derived based on the likelihood and the impact

We categorize the findings into four distinct categories, depending on their severity. These severities are derived from the likelihood and the impact using the following table, following a standard risk assessment procedure.



Severity

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

As seen in the table above, findings that have both a high likelihood and a high impact are classified as critical. Intuitively, such findings are likely to be triggered and cause significant disruption. Overall, the severity correlates with the associated risk. However, every finding's risk should always be closely checked, regardless of severity.

2 Findings

The table below summarizes the findings of the audit, including status and severity details.

ID	Severity	Description	Status
1	Medium	Improper forceWithdraw logic	Confirmed
2	Informational	Centralized risk	Confirmed
3	Informational	Lack of comments	Confirmed

2.1 Medium

1. Improper forceWithdraw logic [Medium]

Severity: Medium Likelihood: Medium Impact: Medium

Status: Confirmed

Description

In the `OriginalTokenBridge::forceWithdraw()` function, it will transfer native assets for emergency withdrawal of user assets. However, the user's assets transferred into `RemoteTokenBridge` are `ACE` token, and `forceWithdraw()` is still used to withdraw the native assets.

```
    //dev: It's only used in really urgent situations, like if the bridge gets hacked. We use it to prevent any further losses.
    function forceWithdraw(uint256 amount_) external nonReentrant onlyRole(DEFAULT_ADMIN_ROLE) {
        address payable to = payable(msg.sender);
        (bool result, ) = to.call{value: amount_}("");
        require(result, "Transfer failed.");
    }
```

OriginalTokenBridge.sol

```
    function forceWithdraw(uint256 amount_) external nonReentrant onlyRole(DEFAULT_ADMIN_ROLE) {
        address payable to = payable(msg.sender);
        (bool result, ) = to.call{value: amount_}("");
        require(result, "Transfer failed.");
    }
```

RemoteTokenBridge.sol

Recommendation: Revise the code logic as `_ace.transfer(to, amount_);`.

2.2 Informational

2. Centralized risk [Informational]

Status: Confirmed

Description

In cross-chain system logic, its key features are controlled by the EOA privilege account. Including `ORIGINAL_BRIDGE_VALIDATOR_ALICE`, `SIDE_BRIDGE_VALIDATOR_BOB`, `FINANCE_VALIDATOR_CHARLIE`, and `DEFAULT_ADMIN_ROLE`.

Our analysis shows that privileged accounts need to be scrutinized. In the following, we will examine privileged accounts and the associated privileged access in the current contract.

Note that if the privileged owner account is a plain EOA, this may be worrisome and pose counter-party risk to the protocol users. A multi-sig account could greatly alleviate this concern, though it is still far from perfect. Specifically, a better approach is to eliminate the administration key concern by transferring the role to a community-governed DAO. In the meantime, a timelock-based mechanism can also be considered as mitigation.

Recommendation: Initially onboarding could can use MPC wallets or multi-sig wallets to initially mitigate centralization risks, but as a long-running protocol, we recommend eventually transfer the privileged account to the intended DAO-like governance contract. All changed to privileged operations may need to be mediated with necessary timelocks.

Eventually, activate the normal on-chain community-based governance life-cycle and ensure the intended trustless nature and high-quality distributed governance.

3. Lack of comments [Informational]

`Status: Confirmed`

Description

Throughout the codebase there are numerous functions missing or lacking documentation. This hinders reviewers' understanding of the code's intention, which is fundamental to correctly assess not only security, but also correctness. Additionally, comments improve readability and ease maintenance. They should explicitly explain the purpose or intention of the functions, the scenarios under which they can fail, the roles allowed to call them, the values returned and the events emitted.

Recommendation: Consider thoroughly documenting all functions (and their parameters) that are part of the smart contracts' public interfaces. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing comments, consider following the Ethereum Natural Specification Format (NatSpec).

3 Disclaimer

This security audit report does not constitute investment advice or a personal recommendation. It does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Any entity should not rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. This security audit report is not an endorsement of any particular project or team, and the report does not guarantee the security of any particular project. This audit does not give any warranties on discovering all security issues of the smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues, also cannot make guarantees about any additional code added to the assessed project after the audit version. As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with independent audits and a public bug bounty program to ensure the security of smart contract(s). Unless explicitly specified, the security of the language itself (e.g., the solidity language), the underlying compiling toolchain and the computing infrastructure are out of the scope.