Jumpgate Smart Contracts Security Audit Report

May 4, 2022

0XORIO

List of contents

1. Introduction **3**

1.1. Disclaimer 3

1.2. Security Assessment Methodology 3

1.2.1 Severity Level Reference 4

1.2.2 Status Level Reference 4

1.3. Project overview 4

2. Audit Scope **5**

3. Report **6**

3.1. CRITICAL 6

3.2. MAJOR 6

3.2.1 recoverERC20 will revert for some tokens 6

3.3. WARNING 7

3.3.1 Compiler version is not fixed 7

3.3.2 Unexpected behavior for tokens with non-standard decimals value 7

3.4. INFO 8

3.4.1 Repetitive constant for nonce value 8

3.4.2 Comment doesn't match the code 9

3.4.3 SafeERC20 import is unused 9

3.4.4 Several public functions may be declared external 10

3.4.5 Owner has ability to get all the tokens 10

3.4.6 Owner can renounce ownership disabling recover functions 11

3.4.7 Recover functions allow sending to address(0) 11

3.4.8 Unnecessary nonce variable in TokensBridged event 11

3.4.9 Lack of checking approve return value harder errors understanding 12

4. Conclusion **13**

5. About Oxorio **14**

1 Introduction

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on this audit. Each of them check the provided source code independently of each other in accordance with the security assessment methodology described below:

1. Project architecture review:

Manually code study of the architecture of the code based on the source code only to find out the errors and bugs.

2. Check the code against the list of known vulnerabilities

Verification process of the code against the constantly updated list of already known vulnerabilities maintained by the company.

3. Architecture and structure check of the security model

Study project documentation and its comparison against the code including the study of the comments and other technical papers.

4. Result's cross-check by different auditors

Normally the research of the project is made by more than two auditors. After that, there is a step of the mutual cross-check process of audit results between different task performers.

5. Report consolidation

Consolidation of the audited report from multiple auditors.

6. Reaudit of new editions

After the client's review and fixes, the founded issues are being double-checked. The results are provided in the new audit version.

7. Final audit report publication

The final audit version is prepared and provided to the client and also published on the official website of the company.

1.2.1 Severity Level Reference

Findings discovered during the audit are classified as follows: Every issue in this report was assigned a severity level from the following:

- **CRITICAL**: A bug leading to assets theft, fund access locking, or any other loss of funds due to transfer to unauthorized parties.
- **MAJOR**: A bug that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
- **WARNING**: A bug that can break the intended contract logic or expose it to DDoS attacks.
- **INFO**: Minor issue or recommendation reported to / acknowledged by the client's team.

1.2.2 Status Level Reference

Based on the feedback received from the client's team regarding the list of findings discovered by the contractor, the following statuses were assigned to the findings:

- **NEW**: Waiting for the project team's feedback.
- **FIXED**: Recommended fixes have been made to the project code and the identified issue no longer affects the project's security.
- **ACKNOWLEDGED**: The project team is aware of this finding. Recommended fixes for this finding are planned to be made. This finding does not affect the overall security of the project.
- **NO ISSUE**: Finding does not affect the overall security of the project and does not violate the logic of its work
- **DISMISSED**: The issue or recommendation was dismissed by the client.

1.3 Project overview

Jumpgate is a smart contract whose purpose is to transfer available tokens to a predefined recipient on another blockchain via the Wormhole Token Bridge. Jumpgate facilitate cross-chain token transfers under the Lido DAO incentive programs.

2 Audit Scope

The scope of the audit includes those contracts from project's repo:

- AssetRecoverer.sol
- Jumpgate.sol

The audited commit identifier is 0802ae168c81c9a4aa32a1b3f1ac0099cd090428

It's assumed that the nonce does not affect the logic of the bridge and it's safe to set it to 0.

3 Report

3.1 CRITICAL

No issues found

3.2 MAJOR

3.2.1 `recoverERC20` will revert for some tokens

Severity	MAJOR
Status	FIXED

Description

Not every token returns `bool` on transfer. Some returns nothing, e.g. USDT.

So you won't be able to recover USDT and similar tokens, the code will revert AssetRecoverer.sol#L53

```
bool success = IERC20(_token).transfer(_recipient, _amount);
require(success);
```

TetherToken#L340

```
function transfer(address _to, uint _value) public whenNotPaused {
```

Additionally, if the main token (set in constructor) has this peculiarity there is another threat:

`Bridge.sol` has a `require` Bridge.sol#L330

```
require(outstanding + normalizedAmount <= type(uint64).max,
"transfer exceeds max outstanding bridged token amount");
```

called in `transferTokens` Bridge.sol#L139

If you already sent > `type(uint64).max` it will revert. So in this case funds would be locked because you will be unable neither send nor recover the tokens.

Recommendation

Use `safeTransfer` in `recoverERC20` function.

Update

Fixed as recommended.

3.3 WARNING

3.3.1 Compiler version is not fixed

Severity	WARNING
Status	FIXED

Description

AssetRecoverer.sol#L2 Jumpgate.sol#L2

```
pragma solidity ^0.8.0;
```

Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs.

Recommendation

Use fixed compiler version, e.g.

```
pragma solidity 0.8.13;
```

Update

Fixed as recommended.

3.3.2 Unexpected behavior for tokens with non-standard `decimals` value

Severity	WARNING
Status	FIXED

Description

In `bridgeTokens` you assume that main token has `18` decimals Jumpgate.sol#L79

```
require(amount >= 10**10, "Amount too small for bridging!");
```

but it's not always the case. There are no checks for this in code.

- If `decimals` > 18 you may burn up to 10**10, depending on price it may not be acceptable
- If `decimals` < 18 you don't allow to send tokens even so Wormhole allows it

Recommendation

Use formula used in Wormhole Bridge.sol#L146

```
function normalizeAmount(uint256 amount, uint8 decimals) internal
pure returns(uint256){
        if (decimals > 8) {
            amount /= 10 ** (decimals - 8);
        }
        return amount;
    }
```

or add a check in constructor that token.decimals() is 18.

Update

Fixed as recommended.

3.4 INFO

3.4.1 Repetitive constant for nonce value

Severity	INFO
Status	FIXED

Description

The nonce of the transfer is hardcoded as `0` and it used twice - at Jumpgate.sol#L91 and Jumpgate.sol#L109 which contradicts of DRY principle ("Don't repeat yourself").

Recommendation

We recommend to add constant `nonce` and use it as an argument when calling `TokensBridged` and `bridge.transferTokens`.

Update

Fixed as recommended.

3.4.2 Comment doesn't match the code

Severity	INFO
Status	FIXED

Description

Comment at AssetRecoverer.sol#L44 refers to `SafeERC20.safeTransfer` but actual code doesn't use it.

```
/// @dev SafeERC20.safeTransfer doesn't return a bool as it performs
an internal `require` check
```

Recommendation

We recommend to keep comments in sync with the code.

Update

Fixed as recommended, `SafeERC20.safeTransfer` is used in the function.

3.4.3 `SafeERC20` import is unused

Severity	INFO
Status	FIXED

Description

AssetRecoverer.sol#L5

```
import "OpenZeppelin/openzeppelin-contracts@4.5.0/contracts/token/
ERC20/utils/SafeERC20.sol";
```

`SafeERC20` is imported but is never used.

Recommendation

In general, we advise removing unused imports however in this case we recommend using `SafeERC20`, see the major issue in this report.

Update

Fixed as recommended, `SafeERC20` is used for `IERC20`.

3.4.4 Several `public` functions may be declared `external`

Severity	INFO
Status	FIXED

Description

`recoverEther`, `recoverERC20`, `recoverERC721`, `bridgeTokens`

`public` functions that are never called by the contract should be declared `external` to save gas.

Recommendation

Use the `external` visibility for functions never called from the contract.

Update

Fixed as recommended.

3.4.5 Owner has ability to get all the tokens

Severity	INFO
Status	ACKNOWLEDGED

Description

Owner can steal all the tokens on the contract, including the main one that is set in constructor, using recover functions. It make the contract very centralized because one actor has control over all the tokens.

Recommendation

Make sure the `owner` is either multisig or DAO.

Update

Issue acknowledged by the team, does not present a security concern because each jumpgate must be verified and whitelisted with the Lido DAO.

3.4.6 Owner can renounce ownership disabling recover functions

Severity	INFO
Status	FIXED

Description

It may happen by mistake or with bad intents.

Recommendation

Consider disabling `renounceOwnership`.

Update

Fixed by overriding `renounceOwnership` with a reverting function.

3.4.7 Recover functions allow sending to `address(0)`

Severity	INFO
Status	FIXED

Description

If `recoverEther` function is called without an argument it will translate `_recipient` to `address(0)` and burn ethers. Setting recipient to `address(0)` is also possible for other function even so less likely.

Recommendation

Consider adding zero address checks for recover functions.

Update

Fixed by using a modifier that checks for zero address.

3.4.8 Unnecessary `nonce` variable in `TokensBridged` event

Severity	INFO
Status	FIXED

Description

Nonce is always `0` but it's emitted in an event Jumpgate.sol#L30 which costs some gas: :

```
event TokensBridged(
    ...
        uint32 _nonce,
    ...
    );
```

Recommendation

Remove a variable that is actually a constant from the TokensBridged event.

Update

Fixed as recommended.

3.4.9 Lack of checking `approve` return value harder errors understanding

Severity	INFO
Status	FIXED

Description

`token.approve` return value is not checked at Jumpgate.sol#L81:

```
token.approve(address(bridge), amount);
```

If it returns false the execution will revert inside Wormhole bridge which may be confusing.

Recommendation

Consider using `safeApprove` to increase UX and DX.

Update

Fixed as recommended using `SafeERC20.safeApprove`.

4 Conclusion

The following table contains the total number of issues that were found during audit:

Level	Amount
CRITICAL	0
MAJOR	1
WARNING	2
INFO	9
Total	9

As stated in each particular issue, all issues identified have been correctly fixed or acknowledged by the client by commit with hash `40f7e708b585238025abd5844a428f231bf3b464`, so contracts are assumed as secure to use according to our security criteria and ready to deploy to mainnet.

5 About Oxorio

Oxorio is a young but rapidly growing audit and consulting company in the field of the blockchain industry, providing consulting and security audits for organizations from all over the world. Oxorio has participated in multiple blockchain projects where smart contract systems were designed and deployed by the company.

Oxorio is the creator, maintainer, and major contributor of several blockchain projects and employs more than 5 blockchain specialists to analyze and develop smart contracts.

Contacts:

- oxor.io
- ping@oxor.io
- github
- linkedin