Smart contracts
security assessment

Final report

Tariff: Simple

Toro

November 2021

Oxguard.com

hello@Oxguard.com
Contents

1. Introduction  
2. Contracts checked  
3. Procedure  
4. Known vulnerabilities checked  
5. Classification of issue severity  
6. Issues  
7. Conclusion  
8. Disclaimer
Introduction

The report has been prepared for the Toronet team. The code was audited after commit d8cdd0.

<table>
<thead>
<tr>
<th>Name</th>
<th>Toro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit date</td>
<td>2021-11-09 - 2021-11-09</td>
</tr>
<tr>
<td>Language</td>
<td>Solidity</td>
</tr>
<tr>
<td>Platform</td>
<td>Toronet Chain</td>
</tr>
</tbody>
</table>

Contracts checked

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>EthBridge</td>
<td><code>https://github.com/Toronet/SmartContracts/blob/d8cdd0ded7419cb4424d15e0b10bade55cb761f6/eth_bridge/eth_bridge.sol</code></td>
</tr>
</tbody>
</table>

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyse smart contracts for security vulnerabilities
- Smart contracts' logic check
### Known vulnerabilities checked

<table>
<thead>
<tr>
<th>Title</th>
<th>Check result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unencrypted Private Data On-Chain</td>
<td>passed</td>
</tr>
<tr>
<td>Code With No Effects</td>
<td>passed</td>
</tr>
<tr>
<td>Message call with hardcoded gas amount</td>
<td>passed</td>
</tr>
<tr>
<td>Typographical Error</td>
<td>not passed</td>
</tr>
<tr>
<td>DoS With Block Gas Limit</td>
<td>not passed</td>
</tr>
<tr>
<td>Presence of unused variables</td>
<td>passed</td>
</tr>
<tr>
<td>Incorrect Inheritance Order</td>
<td>passed</td>
</tr>
<tr>
<td>Requirement Violation</td>
<td>passed</td>
</tr>
<tr>
<td>Weak Sources of Randomness from Chain Attributes</td>
<td>passed</td>
</tr>
<tr>
<td>Shadowing State Variables</td>
<td>passed</td>
</tr>
<tr>
<td>Incorrect Constructor Name</td>
<td>passed</td>
</tr>
<tr>
<td>Block values as a proxy for time</td>
<td>passed</td>
</tr>
<tr>
<td>Authorization through tx.origin</td>
<td>passed</td>
</tr>
<tr>
<td>DoS with Failed Call</td>
<td>passed</td>
</tr>
<tr>
<td>Delegatecall to Untrusted Callee</td>
<td>passed</td>
</tr>
<tr>
<td>Use of Deprecated Solidity Functions</td>
<td>passed</td>
</tr>
<tr>
<td>Assert Violation</td>
<td>passed</td>
</tr>
<tr>
<td>State Variable Default Visibility</td>
<td>passed</td>
</tr>
<tr>
<td>Reentrancy</td>
<td>passed</td>
</tr>
<tr>
<td>Unprotected SELFDESTRUCT Instruction</td>
<td>passed</td>
</tr>
<tr>
<td>Unprotected Ether Withdrawal</td>
<td>passed</td>
</tr>
<tr>
<td>Unchecked Call Return Value</td>
<td>passed</td>
</tr>
</tbody>
</table>
Floating Pragma: not passed
Outdated Compiler Version: passed
Integer Overflow and Underflow: passed
Function Default Visibility: passed

Classification of issue severity

High severity
High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity
Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity
Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. The credited flag of auto withdrawal may not be updated after ETH was sent (EthBridge)

The function processPendingAutoWithdrawals() requires the success of the two send() calls to set the withdrawal as credited. Ether may be sent to an eth_address but not update the auto_withdrawals structure that the withdrawal was credited. This may occur if the second send call at L288 fails.
### Medium severity issues

1. **Floating pragma (EthBridge)**

The pragma statement in the code accepts a wide range of Solidity versions.

```solidity
pragma solidity >=0.6.0 <0.9.0;
```

Solidity versions under 0.8 don't have built-in safemath checks and if the contract is compiled with such version integer overflow/underflow may occur.

**Recommendation:** Raise the pragma’s lower bound up to 0.8 to ensure that the contract will be compiled with built-in safemath checks or use SafeMath library from OpenZeppelin for calculations.
2. Withdrawals processing functions can exceed gasLimit (EthBridge)

The functions processPendingAutoWithdrawals() and processPendingManualWithdrawals() iterate over arrays with an unlimited number of elements. After reaching a certain amount of pending withdrawals, functions (L221 and L289) may exceed the block gas limit.

Low severity issues

1. Extra computations for timestamp (EthBridge)

Gas can be saved by removing unnecessary computations. Constant value \( \text{required\_NumberOfMins} \times 1\ \text{minutes} \) is calculated for each pending transaction while processing (L226 and L295).

**Recommendation:** Make \( \text{required\_NumberOfMins} \) (L119) immutable and equal to \( \_\text{Required\_NumberOfMins} \times 1\ \text{minutes} \), remove \( 1\ *\ \text{minutes} \) multiplier from L226 and L295.

2. isOwners() can exceed gasLimit (EthBridge)

In the case of a big array’s length isOwners() L155 can exceed gasLimit.

**Recommendation:** Use a mapping (address => bool) to set flags if a user is an owner.

3. Unnecessary require (EthBridge)

Ownership check L:139 is redundant. If the address doesn’t have ownership status, removeOwner() will return false.

**Recommendation:** Remove the require statement.

4. Discrepancy in the contract documentation (EthBridge)

The documentation of the contract states that

> Changing owners and removing owners has been removed from this contract for security

However, there are still functions in the contract to add or remove an owner (addOwner() and...
removeOwner() functions).

**Recommendation:** Update documentation or remove the functions.
Conclusion

Toro EthBridge contract was audited. 1 high, 2 medium, 4 low severity issues were found.
Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.