Guard

Smart contracts security assessment

Final report

Tariff: Simple

Toro

November 2021

0xguard.com

hello@0xguard.com

Contents

Introduction

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

Name	Toro
Audit date	2021-11-09 - 2021-11-09
Language	Solidity
Platform	Toronet Chain

Contracts checked

Name	Address
EthBridge	https://github.com/Toronet/SmartContracts/blob/d8cdd0ded7419cb4424d15e0b10bade55cb761f6/eth_bridge/eth_bridge.sol

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

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

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Outdated Compiler Version	passed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 processPendiungAutoWithdrawals() 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.

```
    function processPendiungAutoWithdrawals() public onlyOwners returns (bool) {
        uint256 count = pending_auto_withdrawals.length;
        if (count > 0) {
            for (uint256 i = count; i > 0; i--) {
                bytes32 id = pending_auto_withdrawals[i - 1];
                if ((auto_withdrawals[id].date_Received + required_NumberofMins * 1
minutes) < block.timestamp) {
                    if (address(this).balance >= auto_withdrawals[id].amount) {
                        if
(auto_withdrawals[id].eth_address.send(auto_withdrawals[id].amount -
auto_withdrawals[id].fee) && owner.send(auto_withdrawals[id].fee)) {
                            auto_withdrawals[id].isCredited = true;
                            ...
                        }
                    }
                }
            }
        }
        return true;
    }
```

Medium severity issues

1. Floating pragma (EthBridge)

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

```
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 processPendiungAutoWithdrawals() and processPendiungManualWithdrawals() 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 *required_NumberofMins * 1 minutes* is calculated for each pending transaction while processing (**L226** and **L295**).

Recommendation: Make *required_NumberofMins* (**L119**) immutable *and* equal *to _Required_NumberofMins * 1* minutes, remove *1 * minutes* multiplier from **L226** and **L295**.

2. isOwners() can exceed gasILimit (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 secuirty
```

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.

0x Guard