



Smart contracts security assessment

Final report

Tariff: Standard

Printer Financial

March 2022



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	3
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	5
7. Conclusion	8
8. Disclaimer	9
9. Slither output	10

Introduction

The report has been prepared for Printer Financial team.

The audited code has md5 hash-sum 3d9a07d59e2b9cd95ffae0b1a725808b. Users should check if they are interacting with the audited contract.

The audited contract is the Bridge contract helping to exchange a token between networks by burning a token on the first network and minting it on the second network.

Name	Printer Financial
Audit date	2022-03-21 - 2022-03-22
Language	Solidity
Platform	Binance Smart Chain, Avalanche Network, Fantom Network, Cronos Network

Contracts checked

Name	Address
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	passed
DoS With Block Gas Limit	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	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Outdated Compiler Version	passed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

No issues were found

Medium severity issues

No issues were found

Low severity issues

1. Lack of events on important value changes (Bridge.sol)

Many important setter functions don't emit events on important value changes.

```
function setProcessedStatus(uint256 _txId, bool _status) external onlyBridgeAdmins

function setBridgeOperator(address _account, bool _operator) external onlyBridgeAdmins

function setPaper(address _paper) external onlyBridgeAdmins

function setTreasury(address _treasury) external onlyBridgeAdmins

function setBridgeStatus(bool _status) external onlyBridgeAdmins

function setChainStatus(uint256 _chain, bool _status) external onlyBridgeAdmins

function setMinTokenForChain(uint256 _chain, uint256 _amount) external onlyBridgeAdmins

function setNextTxId(uint256 _txId) external onlyBridgeAdmins
```

Recommendation: We recommend adding events to make tracking changes easier.

2. Return value of transferFrom() not checked (Bridge.sol)

The function sendRequest() does not check return value of the transferFrom() call.

```
function sendRequest(address _recipient, uint256 _chain, uint256 _amount) external {
    ...
    IERC20(paper).transferFrom(msg.sender, address(this), _amount);
    ...
}
```

Recommendation: Use OpenZeppelin library SafeERC20.

Conclusion

Printer Financial Bridge.sol contract was audited. 2 low severity issues were found.

It must be noted that the architecture of the bridge is highly centralised. The contract is dependant on the bridgeAdmins and the bridgeOperators accounts. bridgeAdmins can set bridgeOperators, if a bridgeOperator is compromised attacker can mint any amount of the paper tokens. These accounts must be properly secured.

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.

Slither output

Bridge.sendRequest(address,uint256,uint256) (Bridge.sol#50-66) ignores return value by IERC20(paper).transferFrom(msg.sender,address(this),_amount) (Bridge.sol#62)
 Bridge.governanceRecoverUnsupported(IERC20,uint256,address) (Bridge.sol#119-121) ignores return value by _token.transfer(_to,_amount) (Bridge.sol#120)
 Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer>

Bridge.constructor(uint256,address,address)._paper (Bridge.sol#39) lacks a zero-check on :

- paper = _paper (Bridge.sol#42)

Bridge.constructor(uint256,address,address)._treasury (Bridge.sol#39) lacks a zero-check on :

- treasury = _treasury (Bridge.sol#43)

Bridge.setPaper(address)._paper (Bridge.sol#95) lacks a zero-check on :

- paper = _paper (Bridge.sol#96)

Bridge.setTreasury(address)._treasury (Bridge.sol#99) lacks a zero-check on :

- treasury = _treasury (Bridge.sol#100)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation>

Reentrancy in Bridge.bridgeMint(uint256,address,address,uint256) (Bridge.sol#70-83):

External calls:

- ITreasury(treasury).bridgeMint(_recipient,_amount) (Bridge.sol#80)

Event emitted after the call(s):

- SendProcessed(_txId,_sender,_recipient,chain,_amount) (Bridge.sol#82)

Reentrancy in Bridge.sendRequest(address,uint256,uint256) (Bridge.sol#50-66):

External calls:

- IERC20(paper).transferFrom(msg.sender,address(this),_amount) (Bridge.sol#62)

- ERC20Burnable(paper).burn(_amount) (Bridge.sol#63)

Event emitted after the call(s):

- SendRequested(_txId,msg.sender,_recipient,_chain,_amount) (Bridge.sol#65)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3>

Bridge.sendRequest(address,uint256,uint256) (Bridge.sol#50-66) compares to a boolean constant:

- require(bool,string)(enabledChains[_chain] == true,Bridge: chain is not

enabled/available) (Bridge.sol#52)

Bridge.bridgeMint(uint256,address,address,uint256) (Bridge.sol#70-83) compares to a boolean constant:

```
-require(bool,string)(bridgeEnabled == true,Bridge: bridge is not enabled)
(Bridge.sol#72)
```

Bridge.bridgeMint(uint256,address,address,uint256) (Bridge.sol#70-83) compares to a boolean constant:

```
-require(bool,string)(processedTransfers[_txId] != true,Bridge: transaction
already processed) (Bridge.sol#75)
```

Bridge.onlyBridgeOperators() (Bridge.sol#26-29) compares to a boolean constant:

```
-require(bool,string)(bridgeOperators[msg.sender] == true,Bridge: caller is not
a bridge operator) (Bridge.sol#27)
```

Bridge.onlyBridgeAdmins() (Bridge.sol#31-34) compares to a boolean constant:

```
-require(bool,string)(bridgeAdmins[msg.sender] == true,Bridge: caller is not a
bridge admin) (Bridge.sol#32)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality>

Different versions of Solidity is used:

- Version used: ['^0.8.0', '^0.8.9']
- ^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#4)
- ^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#4)
- ^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\ERC20Burnable.sol#4)
- ^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.sol#4)
- ^0.8.0 (@openzeppelin\contracts\utils\Context.sol#4)
- ^0.8.9 (Bridge.sol#4)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used>

Context._msgData() (@openzeppelin\contracts\utils\Context.sol#21-23) is never used and should be removed

ERC20._mint(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#252-262) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code>

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\ERC20.sol#4) allows old versions

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\IERC20.sol#4) allows old versions

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions\ERC20Burnable.sol#4) allows old versions

Pragma version^0.8.0 (@openzeppelin\contracts\token\ERC20\extensions
\IERC20Metadata.sol#4) allows old versions

Pragma version^0.8.0 (@openzeppelin\contracts\utils\Context.sol#4) allows old versions

Pragma version^0.8.9 (Bridge.sol#4) necessitates a version too recent to be trusted.

Consider deploying with 0.6.12/0.7.6/0.8.7

solc-0.8.9 is not recommended for deployment

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

Parameter Bridge.sendRequest(address,uint256,uint256)._recipient (Bridge.sol#50) is not in mixedCase

Parameter Bridge.sendRequest(address,uint256,uint256)._chain (Bridge.sol#50) is not in mixedCase

Parameter Bridge.sendRequest(address,uint256,uint256)._amount (Bridge.sol#50) is not in mixedCase

Parameter Bridge.bridgeMint(uint256,address,address,uint256)._txId (Bridge.sol#70) is not in mixedCase

Parameter Bridge.bridgeMint(uint256,address,address,uint256)._sender (Bridge.sol#70) is not in mixedCase

Parameter Bridge.bridgeMint(uint256,address,address,uint256)._recipient (Bridge.sol#70) is not in mixedCase

Parameter Bridge.bridgeMint(uint256,address,address,uint256)._amount (Bridge.sol#70) is not in mixedCase

Parameter Bridge.setProcessedStatus(uint256,bool)._txId (Bridge.sol#87) is not in mixedCase

Parameter Bridge.setProcessedStatus(uint256,bool)._status (Bridge.sol#87) is not in mixedCase

Parameter Bridge.setBridgeOperator(address,bool)._account (Bridge.sol#91) is not in mixedCase

Parameter Bridge.setBridgeOperator(address,bool)._operator (Bridge.sol#91) is not in mixedCase

Parameter Bridge.setPaper(address)._paper (Bridge.sol#95) is not in mixedCase

Parameter Bridge.setTreasury(address)._treasury (Bridge.sol#99) is not in mixedCase

Parameter Bridge.setBridgeStatus(bool)._status (Bridge.sol#103) is not in mixedCase

Parameter Bridge.setChainStatus(uint256,bool)._chain (Bridge.sol#107) is not in mixedCase

Parameter Bridge.setChainStatus(uint256,bool)._status (Bridge.sol#107) is not in mixedCase

Parameter Bridge.setMinTokenForChain(uint256,uint256)._chain (Bridge.sol#111) is not in mixedCase

Parameter Bridge.setMinTokenForChain(uint256,uint256)._amount (Bridge.sol#111) is not

in mixedCase

Parameter Bridge.setNextTxId(uint256)._txId (Bridge.sol#115) is not in mixedCase

Parameter Bridge.governanceRecoverUnsupported(IERC20,uint256,address)._token (Bridge.sol#119) is not in mixedCase

Parameter Bridge.governanceRecoverUnsupported(IERC20,uint256,address)._amount (Bridge.sol#119) is not in mixedCase

Parameter Bridge.governanceRecoverUnsupported(IERC20,uint256,address)._to (Bridge.sol#119) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions>

name() should be declared external:

- ERC20.name() (@openzeppelin\contracts\token\ERC20\ERC20.sol#62-64)

symbol() should be declared external:

- ERC20.symbol() (@openzeppelin\contracts\token\ERC20\ERC20.sol#70-72)

decimals() should be declared external:

- ERC20.decimals() (@openzeppelin\contracts\token\ERC20\ERC20.sol#87-89)

totalSupply() should be declared external:

- ERC20.totalSupply() (@openzeppelin\contracts\token\ERC20\ERC20.sol#94-96)

balanceOf(address) should be declared external:

- ERC20.balanceOf(address) (@openzeppelin\contracts\token\ERC20\ERC20.sol#101-103)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#113-116)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#132-135)

transferFrom(address,address,uint256) should be declared external:

- ERC20.transferFrom(address,address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#150-164)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#178-181)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (@openzeppelin\contracts\token\ERC20\ERC20.sol#197-205)

burn(uint256) should be declared external:

- ERC20Burnable.burn(uint256) (@openzeppelin\contracts\token\ERC20\extensions\ERC20Burnable.sol#20-22)

burnFrom(address,uint256) should be declared external:

- ERC20Burnable.burnFrom(address,uint256) (@openzeppelin\contracts\token\ERC20\extensions\ERC20Burnable.sol#35-42)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external>

Bridge.sol analyzed (7 contracts with 77 detectors), 57 result(s) found

