METATRUST

Draft

Security Assessment for

jetonBridgeV2

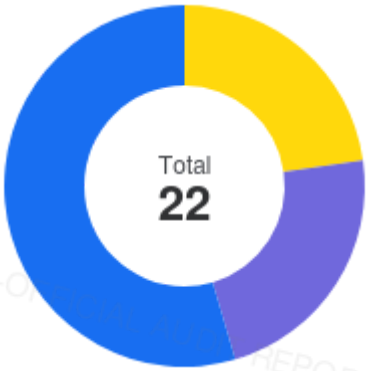
November 15, 2023

Executive Summary

Overview	
Project Name	jetonBridgeV2
Codebase URL	-
Scan Engine	Security Analyzer
Scan Time	2023/11/15 10:50:18
Commit Id	-

Total	
Critical Issues	0
High risk Issues	0
Medium risk Issues	5
Low risk Issues	5
Informational Issues	12

Critical Issues	The issue can cause large economic losses, large-scale data disorder, loss of control of authority management, failure of key functions, or indirectly affect the correct operation of other smart contracts interacting with it.
High Risk Issues	The issue puts a large number of users' sensitive information at risk or is reasonably likely to lead to catastrophic impacts on clients' reputations or serious financial implications for clients and users.
Medium Risk Issues	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk Issues	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational Issue	The issue does not pose an immediate risk but is relevant to security best practices or Defence in Depth.



Critical Issues	0%	0
High risk Issues	0%	0
Medium risk Issues	23%	5
Low risk Issues	23%	5
Informational Issues	55%	12

Summary of Findings

MetaScan security assessment was performed on **November 15, 2023 10:50:18** on project **jetonBridgeV2** with the repository **jetonBridgeV2** on branch **default branch**. The assessment was carried out by scanning the project's codebase using the scan engine **Security Analyzer**. There are in total **22** vulnerabilities / security risks discovered during the scanning session, among which **0** critical vulnerabilities, **0** high risk vulnerabilities, **5** medium risk vulnerabilities, **5** low risk vulnerabilities, **12** informational issues.

ID	Description	Severity
MSA-001	MWE-113: For continue increment in contracts/facets/DexManagerFacet.sol	Medium risk
MSA-002	MWE-113: For continue increment in contracts/facets/ContextFacet.sol	Medium risk
MSA-003	MWE-124: Inconsistent transferFrom Logic in Smart Contracts in node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	Medium risk
MSA-004	MWE-124: Inconsistent transferFrom Logic in Smart Contracts in contracts/libraries/LibAssets.sol	Medium risk
MSA-005	MWE-125: Lack of Check in Setter Function in contracts/facets/AccessManagerFacet.sol	Medium risk
MSA-006	MWE-088: Unused Return Value in contracts/base/JetonSwapper.sol	Low risk
MSA-007	MWE-105: Missing Zero Address Check in contracts/JetonDiamond.sol	Low risk
MSA-008	MWE-105: Missing Zero Address Check in contracts/facets/DexManagerFacet.sol	Low risk
MSA-009	MWE-105: Missing Zero Address Check in contracts/facets/ContextFacet.sol	Low risk
MSA-010	MWE-105: Missing Zero Address Check in contracts/facets/FeesFacet.sol	Low risk
MSA-011	MWE-018: Incorrect ERC20 Interface in contracts/interfaces/IERC20Usdt.sol	Informational
MSA-012	MWE-087: Uninitialized Local Variables in contracts/libraries/LibFee.sol	Informational
MSA-013	MWE-087: Uninitialized Local Variables in contracts/libraries/LibDiamond.sol	Informational
MSA-014	MWE-110: Missing Event Setter in contracts/libraries/LibAccess.sol	Informational
MSA-015	MWE-110: Missing Event Setter in contracts/facets/FeesFacet.sol	Informational
MSA-016	MWE-110: Missing Event Setter in contracts/libraries/LibAssets.sol	Informational
MSA-017	MWE-110: Missing Event Setter in contracts/libraries/LibContext.sol	Informational
MSA-018	MWE-110: Missing Event Setter in contracts/libraries/LibDiamond.sol	Informational
MSA-019	MWE-009: Unnecessary Boolean Comparison in contracts/libraries/LibAccess.sol	Informational
MSA-020	MWE-108: Centralized Risk With Other Variable Read in contracts/facets/ContextFacet.sol	Informational
MSA-021	MWE-076: DoS with Block Gas Limit in contracts/libraries/LibAssets.sol	Informational

ID	Description	Severity
MSA-022	MWE-073: Floating Pragma in contracts/base/JetonValidator.sol	Informational

Findings

Critical (0)

No Critical vulnerabilities found here

High risk (0)

No High risk vulnerabilities found here

Medium risk (5)

1. MWE-113: For continue increment in contracts/facets/DexManagerFacet.sol

Medium risk | Security Analyzer

A continue statement before an unchecked index increment can turn into an infinite loop

File(s) Affected

contracts/facets/DexManagerFacet.sol #39-57

Examples

```
39      function batchAddDex(address[] calldata _dexs) external {
40          if (msg.sender != LibDiamond.contractOwner()) {
41              LibAccess.enforceAccessControl();
    •••
55              }
56          }
57      }
```

Recommendation

Increment the loop index before the continue statement

2. MWE-113: For continue increment in contracts/facets/ContextFacet.sol

Medium risk | Security Analyzer

A continue statement before an unchecked index increment can turn into an infinite loop

File(s) Affected

contracts/facets/ContextFacet.sol #63-81

Examples

```
63      function batchAddTrustedForwarder(
64          address[] calldata _forwarders
65      ) external onlyAuthorized{
    •••
79              }
80          }
81      }
```

Recommendation

Increment the loop index before the continue statement

3. MWE-124: Inconsistent transferFrom Logic in Smart Contracts in node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol

Medium risk | Security Analyzer

In certain smart contracts, especially in token and financial ones, the consistency of 'transferFrom' logic is crucial. An inconsistent logic could lead to unforeseen behaviors, including fund loss or locked assets.

File(s) Affected

node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol #34-36

Examples

```
34      function safeTransferFrom(IERC20 token, address from, address to, uint256 value) internal {
35          _callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from, to, value))
36      }
```

Recommendation

Always ensure that 'transferFrom' and related functions maintain consistent logic. It's recommended to have these functions reviewed by professionals and covered by unit tests.

4. MWE-124: Inconsistent transferFrom Logic in Smart Contracts in contracts/libraries/LibAssets.sol

Medium risk | Security Analyzer

In certain smart contracts, especially in token and financial ones, the consistency of 'transferFrom' logic is crucial. An inconsistent logic could lead to unforeseen behaviors, including fund loss or locked assets.

File(s) Affected

contracts/libraries/LibAssets.sol #85-102

Examples

```
85      function transferFromERC20(
86          address token,
87          address from,
...
100             revert InvalidAmountError();
101         }
102     }
```

Recommendation

Always ensure that 'transferFrom' and related functions maintain consistent logic. It's recommended to have these functions reviewed by professionals and covered by unit tests.

5. MWE-125: Lack of Check in Setter Function in contracts/facets/AccessManagerFacet.sol

Medium risk | Security Analyzer

In the function with setter logic, there must be a check for msgsender to ensure that only authorized users can change values. Without this check, unauthorized users might be able to manipulate contract data or execute functions they shouldn't.

File(s) Affected

contracts/facets/AccessManagerFacet.sol #20-37

Examples

```
20      function setCanExecute(
21          bytes4 _selector,
22          address _executor,
   •••
35              emit ExecutionDenied(_executor, _selector);
36          }
37      }
```

Recommendation

Always add a check for msg.sender in functions with setter logic to ensure that only authorized users can make changes. It's recommended to have these functions reviewed by professionals and covered by unit tests.

Low risk (5)

1. MWE-088: Unused Return Value in contracts/base/JetonSwapper.sol

Low risk | Security Analyzer

Either the return value of an external call is not stored in a local or state variable, or the return value is declared but never used in the function body.

File(s) Affected

contracts/base/JetonSwapper.sol #123-138

Examples

```
134                 ))
135         ) revert ContractCallNotAllowedError();
136
137         LibSwap.swap(_bridgeData.transactionId, _swapData);
138     }
139
140
```

Recommendation

Ensure the return value of external function calls is used. Remove or comment out the unused return function parameters.

2. MWE-105: Missing Zero Address Check in contracts/JetonDiamond.sol

Low risk | Security Analyzer

This Function is lack of zero address check in important operation, which may cause some unexpected result.

File(s) Affected

contracts/JetonDiamond.sol #9-22

Examples

```
9       constructor(address _contractOwner, address _diamondCutFacet) payable {
10          LibDiamond.setContractOwner(_contractOwner);
11
...
20          });
21          LibDiamond.diamondCut(cut, address(0), "");
22      }
```

Recommendation

Add check of zero address in important operation.

3. MWE-105: Missing Zero Address Check in contracts/facets/DexManagerFacet.sol

Low risk | Security Analyzer

This Function is lack of zero address check in important operation, which may cause some unexpected result.

File(s) Affected

contracts/facets/DexManagerFacet.sol #61-67 #151-155

Examples

```
61      function removeDex(address _dex) external {
62          if (msg.sender != LibDiamond.contractOwner()) {
63              LibAccess.enforceAccessControl();
...
65          LibAllowList.removeAllowedContract(_dex);
66          emit DexRemoved(_dex);
67      }
```

Recommendation

Add check of zero address in important operation.

4. MWE-105: Missing Zero Address Check in contracts/facets/ContextFacet.sol

Low risk | Security Analyzer

This Function is lack of zero address check in important operation, which may cause some unexpected result.

File(s) Affected

contracts/facets/ContextFacet.sol #31-33 #87-90

Examples

```
31      function isTrustedForwarder(address _forwarder) public view returns (bool) {
32          return LibContext._isTrustedForwarder(_forwarder);
33      }
```

Recommendation

Add check of zero address in important operation.

5. MWE-105: Missing Zero Address Check in contracts/facets/FeesFacet.sol

Low risk | Security Analyzer

This Function is lack of zero address check in important operation, which may cause some unexpected result.

File(s) Affected

contracts/facets/FeesFacet.sol #59-64

Examples

```
59      function getBridgeFee(
60          address _token,
61          uint256 _bridgeAmount)
62          external view returns(uint256 bridgeFee){
63         bridgeFee = LibFee._getBridgeFee(_token, _bridgeAmount);
64      }
```

Recommendation

Add check of zero address in important operation.

Informational (12)

1. MWE-018: Incorrect ERC20 Interface in contracts/interfaces/IERC20Usdt.sol

Informational | Security Analyzer

A nonstandard ERC20 interface may halt interaction with a contract compiled with Solidity > 0.4.22 as the return value is missing.

File(s) Affected

contracts/interfaces/IERC20Usdt.sol #4-13

Examples

```
4  interface IERC20Usdt {
5      function totalSupply() external view returns (uint256);
6      function balanceOf(address account) external view returns (uint256);
7      function transfer(address recipient, uint256 amount) external;
8      function allowance(address owner, address spender) external view returns (uint256);
9      function approve(address spender, uint256 amount) external;
10     function transferFrom(address sender, address recipient, uint256 amount) external;
```

Recommendation

Set the appropriate return values and types for the defined `ERC20` functions.

2. MWE-087: Uninitialized Local Variables in contracts/libraries/LibFee.sol

Informational | Security Analyzer

A local variable is either never initialized or is initialized only under certain conditions, while the variable will be used regardless of its initialization. As a result, the default zero value is used, which is not desired.

File(s) Affected

contracts/libraries/LibFee.sol #74-74

Examples

```
74          for(uint i; i< _tokens.length ; i++){
```

Recommendation

Initialize the local variable to a reasonable value. Explicitly setting it to zero if it is meant to be initialized to zero.

3. MWE-087: Uninitialized Local Variables in contracts/libraries/LibDiamond.sol

Informational | Security Analyzer

A local variable is either never initialized or is initialized only under certain conditions, while the variable will be used regardless of its initialization. As a result, the default zero value is used, which is not desired.

File(s) Affected

contracts/libraries/LibDiamond.sol #98-98 #145-145 #183-183 #216-216

Examples

```
98          for (uint256 facetIndex; facetIndex < _diamondCut.length; ) {
```

Recommendation

Initialize the local variable to a reasonable value. Explicitly setting it to zero if it is meant to be initialized to zero.

4. MWE-110: Missing Event Setter in contracts/libraries/LibAccess.sol

Informational | Security Analyzer

Setter-functions must emit events

File(s) Affected

contracts/libraries/LibAccess.sol #57-61

Examples

```
57      function enforceAccessControl() internal view {
58          AccessStorage storage accStor = accessStorage();
59          if (accStor.execAccess[msg.sig][msg.sender] != true)
60              revert UnAuthorizedError();
61      }
```

Recommendation

Emit events in setter functions

5. MWE-110: Missing Event Setter in contracts/facets/FeesFacet.sol

Informational | Security Analyzer

Setter-functions must emit events

File(s) Affected

contracts/facets/FeesFacet.sol #28-35 #38-40 #46-52

Examples

```
28      function setFeeConfigs(
29          address[] calldata _tokens,
30          uint256[] calldata _feeRates,
    •••
33          if (_maxFeeAmounts.length != _feeRates.length) revert LengthIsInconsistentError(_maxFeeAmounts.l
34          LibFee._setFeeConfigs(_tokens, _feeRates, _maxFeeAmounts);
35      }
```

Recommendation

Emit events in setter functions

6. MWE-110: Missing Event Setter in contracts/libraries/LibAssets.sol

Informational Security Analyzer

Setter-functions must emit events

File(s) Affected

contracts/libraries/LibAssets.sol #146-155 #163-171

Examples

```
    function transferAssetIn(address token, uint256 amount) internal {
        if (amount == 0) revert InvalidAmountError();
        if (isNativeToken(token)) {
...
            transferFromERC20(token, msg.sender, address(this), amount);
        }
    }
```

Recommendation

Emit events in setter functions

7. MWE-110: Missing Event Setter in contracts/libraries/LibContext.sol

Informational Security Analyzer

Setter-functions must emit events

File(s) Affected

contracts/libraries/LibContext.sol #71-81 #83-89

Examples

```
    function _msgSender() internal view returns (address sender) {
        if (_isTrustedForwarder(msg.sender) && msg.data.length >= 20) {
            // The assembly code is more direct than the Solidity version using `abi.decode`.
...
            return msg.sender;
        }
    }
```

Recommendation

Emit events in setter functions

8. MWE-110: Missing Event Setter in contracts/libraries/LibDiamond.sol

Informational Security Analyzer

Setter-functions must emit events

File(s) Affected

contracts/libraries/LibDiamond.sol #81-84

Examples

```
    function enforceIsContractOwner() internal view {
        if (msg.sender != diamondStorage().contractOwner)
            revert OnlyContractOwnerError();
    }
```

Recommendation

Emit events in setter functions

9. MWE-009: Unnecessary Boolean Comparison in contracts/libraries/LibAccess.sol

Informational | Security Analyzer

Boolean constants can be used directly and do not need to compare to true or false.

File(s) Affected

contracts/libraries/LibAccess.sol #57-61

Examples

```
    ///     has not been given permission to execute `msg.sig`
    function enforceAccessControl() internal view {
        AccessStorage storage accStor = accessStorage();
        if (accStor.execAccess[msg.sig][msg.sender] != true)
            revert UnAuthorizedError();
    }
}
```

Recommendation

Just use boolean constants directly.

10. MWE-108: Centralized Risk With Other Variable Read in contracts/facets/ContextFacet.sol

Informational | Security Analyzer

The contract has a centralized risk, which means that the contract is controlled by a single address. If the address is compromised, the contract will be compromised.

File(s) Affected

contracts/facets/FeesFacet.sol #28-35 #38-40 #46-52
contracts/facets/ContextFacet.sol #47-57 #63-81 #87-90 #96-105

Examples

```
    function setFeeConfigs(
        address[] calldata _tokens,
        uint256[] calldata _feeRates,
    ...
        if (_maxFeeAmounts.length != _feeRates.length) revert LengthIsInconsistentError(_maxFeeAmounts.l
        LibFee._setFeeConfigs(_tokens, _feeRates, _maxFeeAmounts);
    }
```

Recommendation

Avoid using centralized risk contracts.

11. MWE-076: DoS with Block Gas Limit in contracts/libraries/LibAssets.sol

Informational Security Analyzer

When smart contracts are deployed, functions inside them are called, and the execution of these actions always requires a certain amount of gas based on how much computation is needed to complete them. The Ethereum network specifies a block gas limit, and the sum of all transactions included in a block can not exceed the threshold.\nProgramming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit. For example, modifying an array of unknown size that increases over time can lead to a Denial of Service condition.

File(s) Affected

contracts/libraries/LibAllowList.sol #40-60
contracts/libraries/LibContext.sol #42-61
contracts/libraries/LibFee.sol #68-84
contracts/base/JetonSwapper.sol #92-116
contracts/libraries/LibDiamond.sol #93-124 #126-162 #164-201 #203-229
contracts/libraries/LibAssets.sol #198-206

Examples

```

        uint256 length = als.contracts.length;
        // Find the contract in the list
        for (uint256 i = 0; i < length; i++) {
            if (als.contracts[i] == _contract) {
                // Move the last element into the place to delete
                als.contracts[i] = als.contracts[length - 1];
                // Remove the last element
                als.contracts.pop();
                break;
            }
        }
    }

    /// @dev Fetch contract addresses from the allow list
```

Recommendation

Caution is advised when you expect to have large arrays that grow over time. Actions that require looping across the entire data structure should be avoided.\n If you absolutely must loop over an array of unknown size, then you should plan for it to take multiple blocks and potentially require multiple transactions.

12. MWE-073: Floating Pragma in contracts/base/JetonValidator.sol

Informational Security Analyzer

An unlocked compiler version like ^0.8.0 in the contract's source code permits the user to compile it at or above a particular version, which leads to differences in the generated bytecode between compilations due to differing compiler version numbers. As a result, compiler-specific bugs may occur in the codebase that would be hard to identify throughout multiple compiler versions rather than a specific one, which can cause ambiguity. Moreover, the contracts may be at the risk of being accidentally deployed using an outdated compiler version which can introduce bugs to affect the contract system negatively.

File(s) Affected

contracts/interfaces/IUniswapV2Router01.sol #1-1
contracts/libraries/LibBytes.sol #2-2
contracts/facets/DiamondLoupeFacet.sol #2-2
contracts/facets/OwnershipFacet.sol #2-2
contracts/JetonDiamond.sol #2-2
contracts/facets/StargateFacet.sol #2-2
contracts/interfaces/IStargateRouter.sol #2-2
contracts/interfaces/IDiamondCut.sol #2-2
contracts/interfaces/IERC20Usdt.sol #2-2
contracts/base/ReentrancyGuard.sol #2-2
contracts/libraries/LibUtils.sol #2-2
contracts/facets/DiamondCutFacet.sol #2-2
contracts/facets/FeesFacet.sol #2-2
contracts/libraries/LibSwap.sol #2-2
contracts/facets/SwftFacet.sol #2-2
contracts/facets/ContextFacet.sol #2-2
contracts/libraries/LibContext.sol #2-2
contracts/interfaces/IJeton.sol #2-2
contracts/interfaces/IERC173.sol #2-2
contracts/interfaces/IUsdcRouter.sol #2-2
contracts/libraries/LibDiamond.sol #2-2
contracts/interfaces/INativeToken.sol #2-2
contracts/facets/UsdcFacet.sol #2-2
contracts/libraries/LibFee.sol #2-2
contracts/facets/DexManagerFacet.sol #2-2
contracts/errors/JetonErrors.sol #2-2
contracts/interfaces/IAccessManagerFacet.sol #2-2
contracts/interfaces/IERC20Permit.sol #2-2
contracts/interfaces/IERC165.sol #2-2
contracts/base/JetonSwapper.sol #2-2
contracts/libraries/LibAllowList.sol #2-2
contracts/interfaces/IDexManagerFacet.sol #2-2
contracts/interfaces/ISwftRouter.sol #2-2
contracts/interfaces/IDiamondLoupe.sol #2-2
contracts/libraries/LibAccess.sol #2-2
contracts/facets/AccessManagerFacet.sol #2-2
contracts/base/JetonValidator.sol #2-2
contracts/libraries/LibAssets.sol #3-3
node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Permit.sol #4-4
node_modules/@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol #4-4
node_modules/@openzeppelin/contracts/utils/Address.sol #4-4
node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol #4-4
node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol #4-4
node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol #4-4

Examples

```
pragma solidity >=0.6.2;
```

Recommendation

Lock the compiler version to the lowest version possible so that the contract can be compiled and consider known bugs for the chosen compiler version.

Disclaimer

This report is governed by the stipulations (including but not limited to service descriptions, confidentiality, disclaimers, and liability limitations) outlined in the Services Agreement, or as detailed in the scope of services and terms provided to you, the Customer or Company, within the context of the Agreement. The Company is permitted to use this report only as allowed under the terms of the Agreement. Without explicit written permission from MetaTrust, this report must not be shared, disclosed, referenced, or depended upon by any third parties, nor should copies be distributed to anyone other than the Company.

It is important to clarify that this report neither endorses nor disapproves any specific project or team. It should not be viewed as a reflection of the economic value or potential of any product or asset developed by teams or projects engaging MetaTrust for security evaluations. This report does not guarantee that the technology assessed is completely free of bugs, nor does it comment on the business practices, models, or legal compliance of the technology's creators.

This report is not intended to serve as investment advice or a tool for investment decisions related to any project. It represents a thorough assessment process aimed at enhancing code quality and mitigating risks inherent in cryptographic tokens and blockchain technology. Blockchain and cryptographic assets inherently carry ongoing risks. MetaTrust's role is to support companies and individuals in their security diligence and to reduce risks associated with the use of emerging and evolving technologies. However, MetaTrust does not guarantee the security or functionality of the technologies it evaluates.

MetaTrust's assessment services are contingent on various dependencies and are continuously evolving. Accessing or using these services, including reports and materials, is at your own risk, on an as-is and as-available basis. Cryptographic tokens are novel technologies with inherent technical risks and uncertainties. The assessment reports may contain inaccuracies, such as false positives or negatives, and unpredictable outcomes. The services may rely on multiple third-party layers.

All services, labels, assessment reports, work products, and other materials, or any results from their use, are provided "as is" and "as available," with all faults and defects, without any warranty. MetaTrust expressly disclaims all warranties, whether express, implied, statutory, or otherwise, including but not limited to warranties of merchantability, fitness for a particular purpose, title, non-infringement, and any warranties arising from course of dealing, usage, or trade practice. MetaTrust does not guarantee that the services, reports, or materials will meet specific requirements, be error-free, or be compatible with other software, systems, or services.

Neither MetaTrust nor its agents make any representations or warranties regarding the accuracy, reliability, or currency of any content provided through the services. MetaTrust is not liable for any content inaccuracies, personal injuries, property damages, or any loss resulting from the use of the services, reports, or materials.

Third-party materials are provided "as is," and any warranty concerning them is strictly between the Customer and the third-party owner or distributor. The services, reports, and materials are intended solely for the Customer and should not be relied upon by others or shared without MetaTrust's consent. No third party or representative thereof shall have any rights or claims against MetaTrust regarding these services, reports, or materials.

The provisions and warranties of MetaTrust in this agreement are exclusively for the Customer's benefit. No third party has any rights or claims against MetaTrust regarding these provisions or warranties. For clarity, the services, including any assessment reports or materials, should not be used as financial, tax, legal, regulatory, or other forms of advice.