F Y E O

Security Assessment of ICON Bridge Blockchain Transmission Protocol (BTP)

ICON Foundation

December 2022
Version 1.2

Presented by:
FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Strictly Confidential

Table of Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

Overview

ICON Foundation engaged FYEO Inc. to perform a Security Assessment of ICON Bridge Blockchain Transmission Protocol (BTP).

The assessment was conducted remotely by the FYEO Security Team. Testing took place on August 15 - October 18, 2022, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The following issues were identified during the testing period. Upon further review, FYEO-IB-02 has been upgraded to a High severity rating

- FYEO-IB-01 – Javascore - BTPTokenService - Transfer does not check for native coin
- FYEO-IB-02 – Solidity - Gas limit DoS possible in `transferBatch()` Function
- FYEO-IB-03 – Javascore - BTPMessageCenter - Reentrancy while claiming reward
- FYEO-IB-04 – Javascore - BTPTokenService - Fallback function does not check for negative value
- FYEO-IB-05 – Javascore - BTPTokenService - Fees are neither added nor refunded during `refund()`
- FYEO-IB-06 – Javascore - BTPTokenService - Fees are not checked during registration
- FYEO-IB-07 – Javascore - BTPTokenService - Same coin address can have different name
- FYEO-IB-08 – Relayer - BSC light client: Missing verification of message Next field (BTP address of the BMC to handle the message on the destination chain)
- FYEO-IB-09 – Relayer - BSC light client: Signatures are not verified

- FYEO-IB-10 – Relayer - Data from destination chain is not verified
- FYEO-IB-11 – Relayer - ICON light client: Duplicate votes are not checked
- FYEO-IB-12 – Relayer - ICON light client: Missing verification in `syncVerifier()` function
- FYEO-IB-13 – Solidity - Possible to register two tokens with same address, but different name
- FYEO-IB-14 – Javascore - BTPMessageCenter - FeeGathering optimizations
- FYEO-IB-15 – Javascore - BTPMessageCenter - Relayer bond can be 0
- FYEO-IB-16 – Javascore - BTPTokenService - Reentrancy in `refund()`
- FYEO-IB-17 – Relayer - ICON light client: Minimum number of votes is not enforced
- FYEO-IB-18 – Solidity - Outdated Solidity Version Specified in Multiple Contracts
- FYEO-IB-19 – Solidity - Use of Zero Address to Represent Native Token
- FYEO-IB-20 – Javascore - BTPMessageCenter - Sacking is not in use
- FYEO-IB-21 – Javascore - BTPMessageCenter - `Link.rotate` not in use
- FYEO-IB-22 – Javascore - BTPTokenService - Blacklist response code used for token limits
- FYEO-IB-23 – Javascore - BTPTokenService - Optimization in `balanceOf()`
- FYEO-IB-24 – Javascore - BTPTokenService - Optimization in `transferBatch()`
- FYEO-IB-25 – Javascore - BTPTokenService - Public function is not external
- FYEO-IB-26 – Javascore - BTPTokenService - Reclaiming sets usable amount to 0
- FYEO-IB-27 – Javascore - BTPTokenService - TokenLimits can be 0
- FYEO-IB-28 – Javascore - BTPTokenService - `responseError` is not used
- FYEO-IB-29 – Relayer - BSC light client: Wrong client name
- FYEO-IB-30 – Solidity - Misleading `require` Statement in `transfer()` Function
- FYEO-IB-31 – Solidity - Unnecessary `temp` Variable Inside Loop
- FYEO-IB-32 – Solidity - Use of `this.<>` notation for local function calls
- FYEO-IB-33 – Solidity - Widespread Use of Floating Pragmas
- FYEO-IB-34 – Solidity - `links` mapping currently set to internal for testing, should be set to private
- FYEO-IB-35 – Solidity - abicoderv2 is specified, but this is redundant as v2 is the default

Based on our review process, we conclude that the reviewed code implements the documented functionality.

Scope and Rules of Engagement

The FYEO Review Team performed a Security Assessment of ICON Bridge BTP. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a public repository at **https://github.com/icon-project/icon-bridge** with the commit hash c9415e889bee317ce4d8a78275bc594ac637da9f.

The re-review was carried out between the 10th and 20th of November using the commit hash 3d4e6840fca3f49a54cb22457baf92924e8cab86 with the following fork: **https://github.com/icon-project/icon-bridge-ghsa-pmf5-wx49-g3fg/pulls**

A further Re-review was conducted by FYEO upon the return of the amalgamated code from the client. This was under branch `consolidated-audit-review` with commit hash a0d2d742215e3c408e15445649d8ee14a1874e24

Files included in the code review

```
icon-bridge/
├── common/
│   ├── cli/
│   │   └── helper.go
│   ├── codec/
│   │   ├── bytes.go
│   │   ├── codec.go
│   │   ├── msgpack.go
│   │   └── rlp.go
│   ├── config/
│   │   └── fileconfig.go
│   ├── crypto/
│   │   ├── crypto_test.go
│   │   ├── hash.go
│   │   ├── key.go
│   │   └── signature.go
│   ├── db/
│   │   ├── badger_db.go
│   │   ├── badger_db_test.go
│   │   ├── bucket.go
│   │   ├── database.go
│   │   ├── go_level_db.go
│   │   ├── go_level_db_test.go
│   │   ├── layer_db.go
│   │   ├── map_db.go
```

Files included in the code review

```
│   │   └── map_db_test.go
│   ├── errors/
│   │   ├── errors.go
│   │   └── errors_test.go
│   ├── intconv/
│   │   ├── bigint.go
│   │   ├── bigint_test.go
│   │   ├── bytes.go
│   │   ├── bytes_test.go
│   │   ├── string.go
│   │   └── string_test.go
│   ├── jsonrpc/
│   │   ├── client.go
│   │   └── type.go
│   ├── log/
│   │   ├── filter.go
│   │   ├── formatter.go
│   │   ├── forwarder.go
│   │   ├── forwarder_test.go
│   │   ├── log.go
│   │   ├── slack_hook.go
│   │   └── writer.go
│   ├── mpt/
│   │   └── mpt.go
│   ├── mta/
│   │   ├── accumulator.go
│   │   ├── accumulator_test.go
│   │   ├── extaccumulator.go
│   │   └── extaccumulator_test.go
│   ├── wallet/
│   │   ├── encrypted.go
│   │   ├── keystore.go
│   │   ├── keystore_evm.go
│   │   ├── utils.go
│   │   ├── wallet.go
│   │   └── wallet_evm.go
│   ├── address.go
│   ├── hexbytes.go
│   ├── hexint.go
│   ├── http.go
│   └── string.go
├── javascore/
│   ├── api/
```

Files included in the code review

```
│   │   └── src/
│   │       └── main/
│   │           └── java/
│   │               └── foundation/
│   │                   └── icon/
│   │                       └── iip25/
│   │                           ├── Message.java
│   │                           ├── MessageVerifier.java
│   │                           └── ServiceHandler.java
│   ├── bmc/
│   │   └── src/
│   │       ├── main/
│   │       │   └── java/
│   │       │       └── foundation/
│   │       │           └── icon/
│   │       │               ├── btp/
│   │       │               │   └── bmc/
│   │       │               │       ├── BMCException.java
│   │       │               │       ├── BMCMessage.java
│   │       │               │       ├── BMCMock.java
│   │       │               │       ├── BMCProperties.java
│   │       │               │       ├── BTPMessage.java
│   │       │               │       ├── BTPMessageCenter.java
│   │       │               │       ├── ErrorMessage.java
│   │       │               │       ├── EventDataBTPMessage.java
│   │       │               │       ├── FeeGatheringMessage.java
│   │       │               │       ├── ICONSpecific.java
│   │       │               │       ├── InitMessage.java
│   │       │               │       ├── Link.java
│   │       │               │       ├── LinkMessage.java
│   │       │               │       ├── Links.java
│   │       │               │       ├── ReceiptProof.java
│   │       │               │       ├── Relay.java
│   │       │               │       ├── RelayMessage.java
│   │       │               │       ├── Relayer.java
│   │       │               │       ├── Relayers.java
│   │       │               │       ├── RelayersProperties.java
│   │       │               │       ├── Relays.java
│   │       │               │       ├── Routes.java
│   │       │               │       ├── SackMessage.java
│   │       │               │       ├── ServiceCandidate.java
│   │       │               │       ├── Services.java
│   │       │               │       └── UnlinkMessage.java
```

Files included in the code review

```
│   │   │   │   │   └── score/
│   │   │   │   │       └── data/
│   │   │   │   │           └── EnumerableDictDB.java
│   │   │   │   └── test/
│   │   │   │       └── java/
│   │   │   │           └── foundation/
│   │   │   │               └── icon/
│   │   │   │                   └── btp/
│   │   │   │                       └── bmc/
│   │   │   │                           └── BTPMessageCenterTest.java
│   │   ├── bsr/
│   │   │   └── src/
│   │   │       ├── main/
│   │   │       │   └── java/
│   │   │       │       └── foundation/
│   │   │       │           └── icon/
│   │   │       │               └── btp/
│   │   │       │                   ├── irc2/
│   │   │       │                   │   ├── IRC2.java
│   │   │       │                   │   └── IRC2Basic.java
│   │   │       │                   └── restrictions/
│   │   │       │                       ├── Restrictions.java
│   │   │       │                       ├── RestrictionsException.java
│   │   │       │                       ├── RestrictionsManager.java
│   │   │       │                       ├── RestrictionsScoreInterface.java
│   │   │       │                       └── TokenLimit.java
│   │   │       └── test/
│   │   │           └── java/
│   │   │               └── foundation/
│   │   │                   └── icon/
│   │   │                       └── btp/
│   │   │                           └── restrictions/
│   │   │                               └── RestrictionsTest.java
│   │   ├── bts/
│   │   │   └── src/
│   │   │       ├── main/
│   │   │       │   └── java/
│   │   │       │       └── foundation/
│   │   │       │           └── icon/
│   │   │       │               └── btp/
│   │   │       │                   └── bts/
│   │   │       │                       ├── irc2/
│   │   │       │                       │   ├── IRC2ScoreInterface.java
```

Files included in the code review

```
│   │   │   │   │   ├── IRC2Supplier.java
│   │   │   │   │   └── IRC2SupplierScoreInterface.java
│   │   │   │   ├── utils/
│   │   │   │   │   └── EnumerableSet.java
│   │   │   │   ├── Asset.java
│   │   │   │   ├── AssetTransferDetail.java
│   │   │   │   ├── BTPTokenService.java
│   │   │   │   ├── BTS.java
│   │   │   │   ├── BTSEvents.java
│   │   │   │   ├── BTSException.java
│   │   │   │   ├── BTSMessage.java
│   │   │   │   ├── Balance.java
│   │   │   │   ├── BlacklistDB.java
│   │   │   │   ├── BlacklistResponse.java
│   │   │   │   ├── BlacklistTransaction.java
│   │   │   │   ├── Coin.java
│   │   │   │   ├── TokenLimitRequest.java
│   │   │   │   ├── TokenLimitResponse.java
│   │   │   │   ├── TokenLimitTransaction.java
│   │   │   │   ├── TransferRequest.java
│   │   │   │   ├── TransferResponse.java
│   │   │   │   └── TransferTransaction.java
│   │   │   └── test/
│   │   │       └── java/
│   │   │           └── foundation/
│   │   │               └── icon/
│   │   │                   └── btp/
│   │   │                       └── bts/
│   │   │                           ├── AbstractBTPTokenService.java
│   │   │                           ├── AssertBTS.java
│   │   │                           ├── AssertBTSException.java
│   │   │                           ├── BTPTokenServiceTest.java
│   │   │                           ├── BTSIntegrationTest.java
│   │   │                           ├── BTSTest.java
│   │   │                           ├── OwnershipTest.java
│   │   │                           ├── TransferEndEventLog.java
│   │   │                           ├── TransferStartEventLog.java
│   │   │                           └── UnknownResponseEventLog.java
│   │   ├── integration-tests/
│   │   │   └── src/
│   │   │       └── test/
│   │   │           └── java/
│   │   │               └── foundation/
```

Files included in the code review

```
│   │                   └── icon/
│   │                       └── btp/
│   │                           └── test/
│   │                               ├── RlpUtil.java
│   │                               ├── SampleData.java
│   │                               └── ServiceHandlerTest.java
│   ├── irc2Tradeable/
│   │   └── src/
│   │       └── main/
│   │           └── java/
│   │               └── foundation/
│   │                   └── icon/
│   │                       └── btp/
│   │                           └── irc2Tradeable/
│   │                               ├── IRC2.java
│   │                               ├── IRC2Basic.java
│   │                               └── IRC2Tradeable.java
│   └── token-bsh/
│       └── src/
│           ├── main/
│           │   └── java/
│           │       └── foundation/
│           │           └── icon/
│           │               └── btp/
│           │                   ├── bsh/
│           │                   │   ├── types/
│           │                   │   │   ├── Asset.java
│           │                   │   │   ├── BTPAddress.java
│           │                   │   │   ├── Balance.java
│           │                   │   │   ├── ErrorCodes.java
│           │                   │   │   ├── Token.java
│           │                   │   │   └── TransferAsset.java
│           │                   │   ├── BMCMock.java
│           │                   │   ├── HashMock.java
│           │                   │   └── ServiceHandler.java
│           │                   └── irc2/
│           │                       ├── IRC2.java
│           │                       └── IRC2Basic.java
│           └── test/
│               └── java/
│                   └── foundation/
│                       └── icon/
│                           └── btp/
```

Files included in the code review

```
│                                   └── bsh/
│                                       └── test/
│                                           ├── HashTest.java
│                                           ├── IRC2BasicTest.java
│                                           └── ServiceHandlerTest.java
└── solidity/
    ├── TokenBSH/
    │   └── contracts/
    │       ├── BEP20/
    │       │   ├── BEP20.sol
    │       │   ├── Context.sol
    │       │   └── IBEP20.sol
    │       ├── Interfaces/
    │       │   ├── IBMC.sol
    │       │   ├── IBMCManagement.sol
    │       │   ├── IBMCPeriphery.sol
    │       │   ├── IBMV.sol
    │       │   ├── IBSH.sol
    │       │   ├── IBSHImpl.sol
    │       │   └── IBSHProxy.sol
    │       ├── Libraries/
    │       │   ├── DecodeBase64.sol
    │       │   ├── EncodeBase64.sol
    │       │   ├── Helper.sol
    │       │   ├── Owner.sol
    │       │   ├── ParseAddress.sol
    │       │   ├── Precompiles.sol
    │       │   ├── RLPDecodeStruct.sol
    │       │   ├── RLPEncode.sol
    │       │   ├── RLPEncodeStruct.sol
    │       │   ├── RLPReader.sol
    │       │   ├── Strings.sol
    │       │   └── Types.sol
    │       ├── Mock/
    │       │   ├── BMC.sol
    │       │   ├── BMCMock.sol
    │       │   └── PrecompilesMock.sol
    │       ├── Upgradable/
    │       │   ├── BSHImplUpdate.sol
    │       │   └── BSHProxyUpdate.sol
    │       ├── BEP20TKN.sol
    │       ├── BSHImpl.sol
    │       ├── BSHProxy.sol
```

Files included in the code review

```
│       └── ERC20TKN.sol
├── bmc/
│   └── contracts/
│       ├── interfaces/
│       │   ├── IBMCManagement.sol
│       │   ├── IBMCPeriphery.sol
│       │   └── IBSH.sol
│       ├── libraries/
│       │   ├── DecodeBase64.sol
│       │   ├── EncodeBase64.sol
│       │   ├── ParseAddress.sol
│       │   ├── RLPDecode.sol
│       │   ├── RLPDecodeStruct.sol
│       │   ├── RLPEncode.sol
│       │   ├── RLPEncodeStruct.sol
│       │   ├── String.sol
│       │   ├── Types.sol
│       │   └── Utils.sol
│       ├── test/
│       │   ├── BMCManagementV2.sol
│       │   ├── MockBMCManagement.sol
│       │   ├── MockBMCPeriphery.sol
│       │   ├── MockBSH.sol
│       │   └── TestLibRLP.sol
│       ├── BMCManagement.sol
│       └── BMCPeriphery.sol
└── bts/
    └── contracts/
        ├── interfaces/
        │   ├── IBMCPeriphery.sol
        │   ├── IBMV.sol
        │   ├── IBSH.sol
        │   ├── IBTSCore.sol
        │   ├── IBTSPeriphery.sol
        │   └── IERC20Tradable.sol
        ├── libraries/
        │   ├── DecodeBase64.sol
        │   ├── EncodeBase64.sol
        │   ├── ParseAddress.sol
        │   ├── RLPDecode.sol
        │   ├── RLPDecodeStruct.sol
        │   ├── RLPEncode.sol
        │   ├── RLPEncodeStruct.sol
```

Files included in the code review
│ ├── String.sol │ └── Types.sol ├── test/ │ ├── BMC.sol │ ├── BTSCoreV1.sol │ ├── BTSCoreV2.sol │ ├── CheckParseAddress.sol │ ├── EncodeMessage.sol │ ├── Holder.sol │ ├── MockBMC.sol │ ├── MockBTSCore.sol │ ├── MockBTSPeriphery.sol │ ├── NonRefundable.sol │ ├── NotPayable.sol │ └── Refundable.sol ├── tokens/ │ ├── ERC20TKN.sol │ └── HRC20.sol ├── BTSCore.sol ├── BTSPeriphery.sol └── ERC20Tradable.sol

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Security Assessment of ICON Bridge BTP, we discovered:

- 2 findings with HIGH severity rating.
- 11 findings with MEDIUM severity rating.
- 6 findings with LOW severity rating.
- 16 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

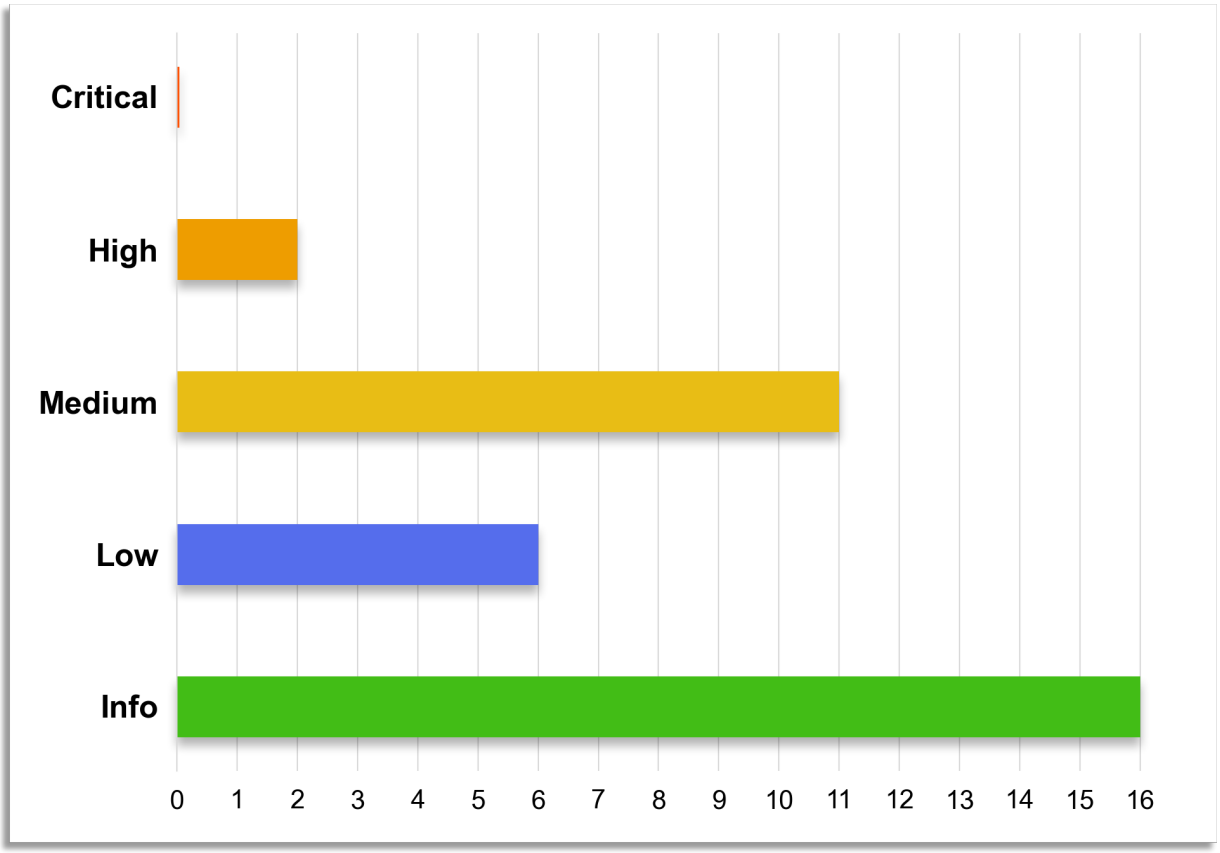

Figure 1: Findings by Severity

FINDINGS

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-IB-01	High	Javascore - BTPTokenService - Transfer does not check for native coin
FYEO-IB-02	High	Solidity - Gas limit DoS possible in `transferBatch()` Function
FYEO-IB-03	Medium	Javascore - BTPMessageCenter - Reentrancy while claiming reward
FYEO-IB-04	Medium	Javascore - BTPTokenService - Fallback function does not check for negative value
FYEO-IB-05	Medium	Javascore - BTPTokenService - Fees are neither added nor refunded during `refund()`
FYEO-IB-06	Medium	Javascore - BTPTokenService - Fees are not checked during registration
FYEO-IB-07	Medium	Javascore - BTPTokenService - Same coin address can have different name
FYEO-IB-08	Medium	Relayer - BSC light client: Missing verification of message Next field (BTP address of the BMC to handle the message on the destination chain)
FYEO-IB-09	Medium	Relayer - BSC light client: Signatures are not verified
FYEO-IB-10	Medium	Relayer - Data from destination chain is not verified
FYEO-IB-11	Medium	Relayer - ICON light client: Duplicate votes are not checked
FYEO-IB-12	Medium	Relayer - ICON light client: Missing verification in `syncVerifier()` function

FYEO-IB-13	Medium	Solidity - Possible to register two tokens with same address, but different name
FYEO-IB-14	Low	Javascore - BTPMessageCenter - FeeGathering optimizations
FYEO-IB-15	Low	Javascore - BTPMessageCenter - Relayer bond can be 0
FYEO-IB-16	Low	Javascore - BTPTokenService - Reentrancy in `refund()`
FYEO-IB-17	Low	Relayer - ICON light client: Minimum number of votes is not enforced
FYEO-IB-18	Low	Solidity - Outdated Solidity Version Specified in Multiple Contracts
FYEO-IB-19	Low	Solidity - Use of Zero Address to Represent Native Token
FYEO-IB-20	Informational	Javascore - BTPMessageCenter - Sacking is not in use
FYEO-IB-21	Informational	Javascore - BTPMessageCenter - `Link.rotate` not in use
FYEO-IB-22	Informational	Javascore - BTPTokenService - Blacklist response code used for token limits
FYEO-IB-23	Informational	Javascore - BTPTokenService - Optimization in `balanceOf()`
FYEO-IB-24	Informational	Javascore - BTPTokenService - Optimization in `transferBatch()`
FYEO-IB-25	Informational	Javascore - BTPTokenService - Public function is not external
FYEO-IB-26	Informational	Javascore - BTPTokenService - Reclaiming sets usable amount to 0
FYEO-IB-27	Informational	Javascore - BTPTokenService - TokenLimits can be 0
FYEO-IB-28	Informational	Javascore - BTPTokenService - `responseError` is not used
FYEO-IB-29	Informational	Relayer - BSC light client: Wrong client name

FYEO-IB-30	Informational	Solidity - Misleading `require` Statement in `transfer()` Function
FYEO-IB-31	Informational	Solidity - Unnecessary `temp` Variable Inside Loop
FYEO-IB-32	Informational	Solidity - Use of `this.<>` notation for local function calls
FYEO-IB-33	Informational	Solidity - Widespread Use of Floating Pragmas
FYEO-IB-34	Informational	Solidity - `links` mapping currently set to internal for testing, should be set to private
FYEO-IB-35	Informational	Solidity - abicoderv2 is specified, but this is redundant as v2 is the default

Table 2: Findings Overview

Technical Analysis

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

Technical Findings

General Observations

ICON Bridge is an early iteration of ICON's cutting-edge interoperability product, BTP (Blockchain Transmission Protocol), which allows cross-chain transfers and integration with any blockchain that supports smart contracts.

This audit is focused on the relayer code (GoLang), bridge contracts on Binance Smart Chain (Solidity) and ICON chain (Javascore). We performed the following

Phase 1: Static code analysis using Gosec and Slither and collaborate with developer to better understand the system.

Phase 2: Initial Review

- Review message delivery flow,
- Light client implementations, and error handling.
- Review critical functions such as ones that handle messages/transfers coming from other chains.
- Ensure Inputs are validated properly.
- Verify third party dependencies.

Phase 3: Deep review - verify bridge logics and arithmetic/math implementation.

Phase 4: Peer review

Code quality is good; most operations carried out carefully. The ICON Bridge development team was very communicative, quickly providing responses to the auditing team.

JAVASCORE - BTPTOKENSERVICE - TRANSFER DOES NOT CHECK FOR NATIVE COIN

Finding ID: FYEO-IB-01
Severity: **High**
Status: **Remediated**

Description

Users can send native coins to another blockchain via the ICON bridge. To do so, they should first pay the amount they want to send to `BTPTokenService`. Non-payble external function `transfer()`, aimed to transfer non-native coins, does not sufficiently check the type of coin transferred. A user could use it transfer a native coin without actually paying for it.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 465

```
@External
public void transfer(String _coinName, BigInteger _value, String _to) {
    require(_value != null && _value.compareTo(BigInteger.ZERO) > 0, "Invalid
amount");
    checkUintLimit(_value);
    require(isRegistered(_coinName), "Not supported Token");

    Address owner = Context.getCaller();
    BTPAddress to = BTPAddress.valueOf(_to);
    checkRestrictions(_coinName, Context.getCaller().toString(), to, _value);
    // only for wrapped coins
    transferFrom(owner, Context.getAddress(), _coinName, _value);
    sendRequest(owner, to, List.of(_coinName), List.of(_value));
}
```

Severity and Impact Summary

Users can transfer a native coin without actually paying for it.

Recommendation

Require that `_coinName != this.name`.

SOLIDITY - GAS LIMIT DOS POSSIBLE IN `TRANSFERBATCH()` FUNCTION

Finding ID: FYEO-IB-02
Severity: **High**
Status: **Remediated**

Description

Similar to the vulnerability in the `balanceOfBatch()` function, the `transferBatch()` function takes an array of token names as argument that are passed in by the user. Although this function will revert in the case that an unregistered token is passed in, it is possible to pass a valid token limitless times.

Proof of Issue

File Name: BTSCore.sol

Line Number: 549

```
function transferBatch(
    string[] calldata _coinNames,
    uint256[] memory _values,
    string calldata _to
) external payable override {
    require(_coinNames.length == _values.length, "InvalidRequest");
    require(_coinNames.length > 0, "Zero length arguments");
    uint256 size = msg.value != 0
        ? _coinNames.length.add(1)
        : _coinNames.length;
    string[] memory _coins = new string[](size);
    uint256[] memory _amounts = new uint256[](size);
    uint256[] memory _chargeAmts = new uint256[](size);
    Coin memory _coin;
    string memory coinName;
    uint value;

    for (uint256 i = 0; i < _coinNames.length; i++) {
        address _erc20Addresses = coins[_coinNames[i]];
        //  Does not need to check if _coinNames[i] == native_coin
        //  If _coinNames[i] is a native_coin, coins[_coinNames[i]] = 0
        require(_erc20Addresses != address(0), "UnregisterCoin");
        coinName = _coinNames[i];
        value = _values[i];
        require(value > 0,"ZeroOrLess");

        btsPeriphery.checkTransferRestrictions(
            coinName,
            msg.sender,
            value
        );
```

```
        IERC20Tradable(_erc20Addresses).transferFrom(
            msg.sender,
            address(this),
            value
        );

        _coin = coinDetails[coinName];
        //  _chargeAmt = fixedFee + msg.value * feeNumerator / FEE_DENOMINATOR
        //  Thus, it's likely that _chargeAmt is always greater than 0
        //  require(_chargeAmt > 0) can be omitted
        _coins[i] = coinName;
        _chargeAmts[i] = value
            .mul(_coin.feeNumerator)
            .div(FEE_DENOMINATOR)
            .add(_coin.fixedFee);
        _amounts[i] = value.sub(_chargeAmts[i]);

        //  Lock this requested _value as a record of a pending transferring
transaction
        //  @dev Note that: _value is a requested amount to transfer from a
Requester including charged fee
        //  The true amount to receive at a destination receiver is calculated
by
        //  _amounts[i] = _values[i].sub(_chargeAmts[i]);
        lockBalance(msg.sender, coinName, value);
    }

    if (msg.value != 0) {
        //  _chargeAmt = fixedFee + msg.value * feeNumerator / FEE_DENOMINATOR
        //  Thus, it's likely that _chargeAmt is always greater than 0
        //  require(_chargeAmt > 0) can be omitted
        btsPeriphery.checkTransferRestrictions(
            nativeCoinName,
            msg.sender,
            msg.value
        );


        _coins[size - 1] = nativeCoinName; // push native_coin at the end of
request
        _chargeAmts[size - 1] = msg
            .value
            .mul(coinDetails[nativeCoinName].feeNumerator)
            .div(FEE_DENOMINATOR)
            .add(coinDetails[nativeCoinName].fixedFee);
        _amounts[size - 1] = msg.value.sub(_chargeAmts[size - 1]);
        lockBalance(msg.sender, nativeCoinName, msg.value);
    }

    //  @dev `_amounts` is true amounts to receive at a destination after
deducting charged fees
    btsPeriphery.sendServiceMessage(
```

```
            msg.sender,
            _to,
            _coins,
            _amounts,
            _chargeAmts
        );
    }
```

Severity and Impact Summary

This could possibly lead to denial of service attack.

Recommendation

Set limit on different coins that can be transferred in a batch.

JAVASCORE - BTPMESSAGECENTER - REENTRANCY WHILE CLAIMING REWARD

Finding ID: FYEO-IB-03
Severity: **Medium**
Status: **Remediated**

Description

Reentrancy occurs when native coin is transferred before changing the state on which the transfer depends. This can happen in refund as the balance of the account is updated after the amount has been transferred. To exploit this, a relayer would need to consistently trigger `claimRelayerReward`.

Proof of Issue

Line number: 1125

```
@External
public void claimRelayerReward() {
    Address addr = Context.getCaller();
    if (!relayers.containsKey(addr)) {
        throw BMCException.unknown("not found registered relayer");
    }
    Relayer relayer = relayers.get(addr);
    BigInteger reward = relayer.getReward();
    if (reward.compareTo(BigInteger.ZERO) < 1) {
        throw BMCException.unknown("reward is not remained");
    }
    Context.transfer(addr, reward);
    relayer.setReward(BigInteger.ZERO);
    relayers.put(addr, relayer);
    RelayersProperties properties = relayers.getProperties();
    properties.setDistributed(properties.getDistributed().subtract(reward));
    relayers.setProperties(properties);
}
```

Severity and Impact Summary

If a malicious relayer is registered, it could use reentrancy to drain rewards.

Recommendation

Change the relayer's properties before the reward is transferred.

JAVASCORE - BTPTOKENSERVICE - FALLBACK FUNCTION DOES NOT CHECK FOR NEGATIVE VALUE

Finding ID: FYEO-IB-04
Severity: **Medium**
Status: **Remediated**

Description

The `tokenFallback()` function is used to receive tokens from the coin contracts. Although it is unlikely, the contract allows for negative values which would reduce the balance of the contract for that coin.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 419

```
    @External
    public void tokenFallback(Address _from, BigInteger _value, byte[] _data)
{
        checkUintLimit(_value);
        String _coinName = coinAddressName.get(Context.getCaller());
        if (_coinName != null && !Context.getAddress().equals(_from)) {
            Context.require(coinAddresses.get(_coinName) != null,
"CoinNotExists");
            Balance _userBalance = getBalance(_coinName, _from);
            _userBalance.setUsable(_userBalance.getUsable().add(_value));
            setBalance(_coinName, _from, _userBalance);
        } else {
            throw BTSException.unknown("Token not registered");
        }
    }
```

Severity and Impact Summary

IRC2 contracts can reduce `BTPTokenService` user balance.

Recommendation

Check that that value is a positive number, `> 0`.

JAVASCORE - BTPTOKENSERVICE - FEES ARE NEITHER ADDED NOR REFUNDED DURING `REFUND()`

Finding ID: FYEO-IB-05
Severity: **Medium**
Status: **Remediated**

Description

Failed requests are detected using error responses. When handled, these refund the original amount. Fees are only accumulated if the request is successful. In case of failure, it is not clear what should happen with the fee as the code neither adds nor refunds the fee deducted. The amount unlocked is also not equal to the amount that is refunded.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 769

```
private void refund(String coinName, Address owner, BigInteger locked,
BigInteger fee) {
    logger.println("refund", "coinName:", coinName, "owner:", owner,
"locked:", locked, "fee: ", fee);
    // unlock and add refundable
    Balance balance = getBalance(coinName, owner);
    BigInteger value = locked.subtract(fee);
    balance.setLocked(balance.getLocked().subtract(locked));
    try {
        if (name.equals(coinName)) {
            Context.transfer(owner, value);
        } else {
            _transferBatch(Context.getAddress(), owner, List.of(coinName),
List.of(value));
        }
    } catch (Exception e) {
        if (!owner.equals(Context.getAddress())) {
            balance.setRefundable(balance.getRefundable().add(value));
        }
    }
    setBalance(coinName, owner, balance);
}
```

- The amount that is unlocked is `locked`
- The amount that is transfer is `locked - fee`
- Fee is not added to `feeBalances`

Line number: 853

```
 private void handleResponse(BigInteger sn, TransferResponse response) {
        // ...
        if (TransferResponse.RC_OK.equals(code)) {
                addFee(coinName, fee);
        } else {
            // ...
                refund(coinName, owner, locked, fee);
        }
}
```

Line number: 789

```
private void addFee(String coinName, BigInteger amount) {
    BigInteger fee = feeBalances.getOrDefault(coinName, BigInteger.ZERO);
    feeBalances.set(coinName, fee.add(amount));
}
```

The fee is only added if the response is successful.

Severity and Impact Summary

The amount that is refunded does not equal the amount that is unlocked, while fees remain unchanged.

Recommendation

Depending on whether fees should be collected, either refund the full amount or refund the amount after fees, while also increasing the fee balances.

JAVASCORE - BTPTOKENSERVICE - FEES ARE NOT CHECKED DURING REGISTRATION

Finding ID: FYEO-IB-06
Severity: **Medium**
Status: **Remediated**

Description

Fees in `BTPTokenService` can are set individually for each coin. The fixed fee is required to be be `>= 0`, while the fee numerator is additionally required to be `< FEE_DENOMINATOR`. This is not checked for during coin registration. This could lead to overcharging in case the numerator is `>= FEE_DENOMINATOR` or loss of funds in case either the numerator or fixed fee are negative.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 174

```
if (_addr == null || _addr.equals(ZERO_SCORE_ADDRESS)) {
    Address irc2Address = Context.deploy(serializedIrc2, _name, _symbol,
_decimals);
    coinAddresses.set(_name, irc2Address);
    coinAddressName.set(irc2Address, _name);
    coinDb.set(_name, new Coin(irc2Address, _name, _symbol, _decimals,
_feeNumerator, _fixedFee,
            NATIVE_WRAPPED_COIN_TYPE));
} else {
    coinAddresses.set(_name, _addr);
    coinDb.set(_name,
            new Coin(_addr, _name, _symbol, _decimals, _feeNumerator,
_fixedFee, NON_NATIVE_TOKEN_TYPE));
    coinAddressName.set(_addr, _name);
}
```

Severity and Impact Summary

Setting wrong fee ratios could lead to overcharging or loss of funds.

Recommendation

Fixed fee and fee numerator should be checked during coin registration.

JAVASCORE - BTPTOKENSERVICE - SAME COIN ADDRESS CAN HAVE DIFFERENT NAME

Finding ID: FYEO-IB-07
Severity: **Medium**
Status: **Remediated**

Description

Coins are indexed by their name. When registering a new coin, it is ensured that the coin name is not already in use. However, two coins with different names can have the same address. Although this will not affect balances, it leaves room for overcoming blacklist restrictions.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 166

```
@External

public void register(String _name, String _symbol, int _decimals, BigInteger
_feeNumerator, BigInteger _fixedFee,
        @Optional Address _addr) {
    requireOwnerAccess();

    require(!isRegistered(_name), "already existed");

    coinNames.add(_name);
    if (_addr == null || _addr.equals(ZERO_SCORE_ADDRESS)) {
        Address irc2Address = Context.deploy(serializedIrc2, _name, _symbol,
_decimals);
        coinAddresses.set(_name, irc2Address);
        coinAddressName.set(irc2Address, _name);
        coinDb.set(_name, new Coin(irc2Address, _name, _symbol, _decimals,
_feeNumerator, _fixedFee,
                NATIVE_WRAPPED_COIN_TYPE));
    } else {
        coinAddresses.set(_name, _addr);
        coinDb.set(_name,
                new Coin(_addr, _name, _symbol, _decimals, _feeNumerator,
_fixedFee, NON_NATIVE_TOKEN_TYPE));
        coinAddressName.set(_addr, _name);
    }
}
```

Severity and Impact Summary

Allowing the same address for different coin names can lead to inconsistencies when checking blacklist.

Recommendation

Check that `coinAddressName.get(_addr) === null`.

RELAYER - BSC LIGHT CLIENT: MISSING VERIFICATION OF MESSAGE NEXT FIELD (BTP ADDRESS OF THE BMC TO HANDLE THE MESSAGE ON THE DESTINATION CHAIN)

Finding ID: FYEO-IB-08
Severity: Medium
Status: Remediated

Description

In BSC's receiver, when filtering out receipts, `msg.Next` (BTP Address of the BMC to handle the message) is not validated.

Proof of Issue

File name: icon/receiver.go
Line number: 553-564

In ICON receiver, the source contract address `el.Addr`, event signature `el.Indexed[EventIndexSignature]`, and next field `el.Indexed[EventIndexNext]` are verified.

```
    if bytes.Equal(el.Addr, logFilter.addr) &&
bytes.Equal(el.Indexed[EventIndexSignature], logFilter.signature) &&
bytes.Equal(el.Indexed[EventIndexNext], logFilter.next) {
        var seqGot common.HexInt
        seqGot.SetBytes(el.Indexed[EventIndexSequence])
        evt := &chain.Event{
                Next:     chain.BTPAddress(el.Indexed[EventIndexNext]),
                Sequence: seqGot.Uint64(),
                Message:  el.Data[0],
        }
        receipt.Events = append(receipt.Events, evt)
    }
```

But in BSC, only the source contract address and event signature are verified.

The source contract address `sc` is compared against address in the log `log.Address.Bytes()`.

File name: bsc/receiver.go
Line number: 461-490

```
func (r receiver) getRelayReceipts(vBlockNotification) []chain.Receipt {
    sc := common.HexToAddress(r.src.ContractAddress())
    var receipts []chain.Receipt
    var events []chain.Event
    for i, receipt := range v.Receipts {
        events := events[:0]
        for _, log := range receipt.Logs {
            if !bytes.Equal(log.Address.Bytes(), sc.Bytes()) {
```

```
                continue
            }
            msg, err := r.bmcClient().ParseMessage(ethTypes.Log{
                Data: log.Data, Topics: log.Topics,
            })
            if err == nil {
                events = append(events, &chain.Event{
                    Next:     chain.BTPAddress(msg.Next),
                    Sequence: msg.Seq.Uint64(),
                    Message:  msg.Msg,
                })
            }
        }
        if len(events) > 0 {
            rp := &chain.Receipt{}
            rp.Index, rp.Height = uint64(i), v.Height.Uint64()
            rp.Events = append(rp.Events, events...)
            receipts = append(receipts, rp)
        }
    }
    return receipts
}
```

The message event is extracted in `ParseMessage()` function.

File name: bsc/bmc_abigen.go
Line number: 821-828

```
func (_BMCBMCFilterer) ParseMessage(log types.Log) (*BMCMessage, error) {
    event := new(BMCMessage)
    if err := _BMC.contract.UnpackLog(event, "Message", log); err != nil {
        return nil, err
    }
    event.Raw = log
    return event, nil
}
```

Severity and Impact Summary

Messages sending to incorrect BTP address are still valid.

Recommendation

We recommend checking the `msg.Next` field against the BTP address of the BMC to handle the message on the destination chain.

RELAYER - BSC LIGHT CLIENT: SIGNATURES ARE NOT VERIFIED

Finding ID: FYEO-IB-09
Severity: Medium
Status: Remediated

Description

The current BSC verifier only checks the headers against the parent hash from the verifier, but not signatures of the validators. Thus, a fraudulent header of a malicious or orphan block might still be accepted.

Proof of Issue

File name: bsc/verifier.go
Line number: 37-55

```
func (vr *Verifier) Verify(h *types.Header, newHeader *types.Header) error {
    vr.mu.Lock()
    defer vr.mu.Unlock()
    if newHeader.Number.Cmp((&big.Int{}).Add(h.Number, big1)) != 0 {
        return fmt.Errorf("Different height between successive header: Prev %v
New %v", h.Number, newHeader.Number)
    }
    if !bytes.Equal(h.Hash().Bytes(), newHeader.ParentHash.Bytes()) {
        return fmt.Errorf("Different hash between successive header: (%v):
Prev %v New %v", h.Number, h.Hash(), newHeader.ParentHash)
    }
    if vr.next.Cmp(h.Number) != 0 {
        return fmt.Errorf("Unexpected height: Got %v Expected %v", h.Number,
vr.next)
    }
    if !bytes.Equal(h.ParentHash.Bytes(), vr.parentHash.Bytes()) {
        return fmt.Errorf("Unexpected Hash(%v): Got %v Expected %v", h.Number,
h.ParentHash, vr.parentHash)
    }
    vr.parentHash = h.Hash()
    vr.next.Add(h.Number, big1)
    return nil
}
```

Severity and Impact Summary

Without proper signature verification, fraudulent headers might still be accepted.

Recommendation

We recommend additionally verifying the validator signatures to ensure the validity of new headers.

RELAYER - DATA FROM THE DESTINATION CHAIN IS NOT VERIFIED

Finding ID: FYEO-IB-10
Severity: Medium
Status: Open

Description

In the sender's implementations, receipts are read for each pending transaction, however it seems this is optional (since there are up to 30 retries) and their integrity is not verified.

This would allow attackers to potentially manipulate the flow of a transaction while allowing the relay to continue working normally.

Proof of Issue

File name: relay/relay.go
Line number: 225-244

```
    waitLoop:
    for blockHeight, err := tx.Receipt(ctx); retryCount < 30; _, err =
tx.Receipt(ctx) {
        switch {
        case err == nil:
            newMsg.From = srcMsg.From
            srcMsg = newMsg
            txBlockHeight = blockHeight
            break waitLoop
        case errors.Is(err, context.Canceled):
            r.log.WithFields(log.Fields{"error": err}).Error("tx.Receipt
failed")
            return err
        case errors.Is(err, chain.ErrGasLimitExceeded):
            // increase transaction gas limit
        case errors.Is(err, chain.ErrBlockGasLimitExceeded):
            // reduce batch size
        case errors.Is(err, chain.ErrBMCRevertInvalidSeqNumber):
            // messages skipped; refetch from source

        default:
            time.Sleep(relayTxReceiptWaitInterval) // wait before asking for
receipt
            if retryCount > retryWarnThreshold {
                r.log.WithFields(log.Fields{"error": err, "retry": retryCount
+ 1}).Warn("tx.Receipt: retry")
            } else {
                r.log.WithFields(log.Fields{"error": err, "retry": retryCount
+ 1}).Debug("tx.Receipt: retry")
            }
        }
```

```
            retryCount++
        }
```

Severity and Impact Summary

An attacker can make the relay think that the forwarded messages have been processed, which could cause a disruption in relay service and halt the relayer.

Recommendation

We recommend verifying the transaction receipt of the relay message.

ICON team Response

In BTP, messages are always processed sequentially, and the sequence number is appropriately tracked by the Relayer to know which messages have been processed. The receipt is only read to know the status of the transaction, and there's no utility of the receipt beside that. Hence, it's not necessary to verify the integrity of the transaction receipt. Even if the receipt can be manipulated, the relay is not impacted much because it doesn't use the content of the receipt. And whenever this leads to missing BTP message, the Relayer thread restarts which helps recover from the issue.

RELAYER - ICON LIGHT CLIENT: DUPLICATE VOTES ARE NOT CHECKED

Finding ID: FYEO-IB-11
Severity: Medium
Status: Remediated

Description

ICON verifier does not check for duplicate votes in `cvl.Items`. It only checks if it is a vote from a valid validator. Thus, it's possible that a validator can vote multiple times and increase the vote count.

Proof of Issue

This function would check if the public key derived from the signature is a valid validator key in the verifier's validator set. It does not check if the public key was already counted or not.

File name: icon/helper.go
Line number: 34-41

```
    func listContains(list []common.HexBytes, data common.HexBytes) bool {
    for _, current := range list {
        if bytes.Equal(data, current) {
            return true
        }
    }
    return false
    }
```

File name: icon/verifier.go
Line number: 117-131

```
    votesCount := 0
    for _, item := range cvl.Items {
        vote.Timestamp = item.Timestamp
        pub, err :=
item.Signature.RecoverPublicKey(crypto.SHA3Sum256(codec.BC.MustMarshalToBytes(
vote)))
        if err != nil {
            continue
        }
        address := common.NewAccountAddressFromPublicKey(pub)
        if listContains(validators, address.Bytes()) {
            votesCount++
        }
    }
    if votesCount < (2*len(validators))/3 {
        return false, fmt.Errorf("insufficient votes")
    }
```

Severity and Impact Summary

Votes from the same validator can be counted multiple times and this would bypass the `2/3` threshold without actually having the majority valid votes.

Recommendation

We recommend filtering out duplications when counting the votes.

RELAYER - ICON LIGHT CLIENT: MISSING VERIFICATION IN `SYNCVERIFIER()` FUNCTION

Finding ID: FYEO-IB-12
Severity: **Medium**
Status: **Remediated**

Description

Function `syncVerifier()` is used to sync up the headers from the hashes in `bmr` config file. In BSC (Binance smart chain), its `syncVerifier()` function does verify the headers and update the verifier. However, in ICON, the function does not verify the headers or update the verifier directly. Moreover, votes are queried, but not used anywhere to verify.

Proof of Issue

Here, BSC verifies the headers then update the verifier

File name: bsc/receiver.go
Line number: 187-208

```
    if len(sres) > 0 {
            sort.SliceStable(sres, func(i, j int) bool {
                return sres[i].Height < sres[j].Height
            })
            for i := range sres {
                cursor++
                next := sres[i]
                if prevHeader == nil {
                    prevHeader = next.Header
                    continue
                }
                if vr.Next().Int64() >= height { // if height is greater than
targetHeight, break loop
                    break
                }
                err := vr.Verify(prevHeader, next.Header)
                if err != nil {
                    return errors.Wrapf(err, "syncVerifier: Update: %v", err)
                }
                prevHeader = next.Header
            }
            r.log.WithFields(log.Fields{"height": vr.Next().String(),
"target": height}).Debug("syncVerifier: syncing")
        }
```

But ICON updates but not verify the headers

File name: icon/receiver.go
Line number: 249-263

```
    if len(sres) > 0 {
            sort.SliceStable(sres, func(i, j int) bool {
                return sres[i].Height < sres[j].Height
            })
            for _, r := range sres {
                if vr.Next() == r.Height {
                    err := vr.Update(r.Header, r.NextValidators)
                    if err != nil {
                        return errors.Wrapf(err, "syncVerifier: Update: %v",
err)
                    }
                }
            }
            r.log.WithFields(log.Fields{"height": vr.Next(), "target":
height}).Debug("syncVerifier: syncing")
        }
```

File name: icon/receiver_core.go
Line number: 156-169

```
    if len(sres) > 0 {
            sort.SliceStable(sres, func(i, j int) bool {
                return sres[i].Height < sres[j].Height
            })
            for _, r := range sres {
                if vr.Next() == r.Height {
                    err := vr.Update(r.Header, r.NextValidators)
                    if err != nil {
                        return errors.Wrapf(err, "syncVerifier: Update: %v",
err)
                    }
                }
            }
            r.log.WithFields(log.Fields{"height": vr.Next(), "target":
height}).Debug("syncVerifier: syncing")
        }
```

Severity and Impact Summary

ICON light client may not receive the right headers, thus, many send incorrect message to a destination chain.

Recommendation

ICON should also verify the headers before updating.

SOLIDITY - POSSIBLE TO REGISTER TWO TOKENS WITH SAME ADDRESS, BUT DIFFERENT NAME

Finding ID: FYEO-IB-13
Severity: **Medium**
Status: **Remediated**

Description

The register function “registers a wrapped coin and id number of a supporting coin”. There are a few checks, but nothing that ensures the address has already been used. This means that the same address can be used to represent tokens with different names.

Proof of Issue

File Name: BTSCore.sol

Line Number: 197

```
    /**
        @notice Registers a wrapped coin and id number of a supporting coin.
        @dev Caller must be an Owner of this contract
        _name Must be different with the native coin name.
        _symbol symbol name for wrapped coin.
        _decimals decimal number
        @param _name    Coin name.
    */
    function register(
        string calldata _name,
        string calldata _symbol,
        uint8 _decimals,
        uint256 _feeNumerator,
        uint256 _fixedFee,
        address _addr
    ) external override onlyOwner {
        require(!_name.compareTo(nativeCoinName), "ExistNativeCoin");
        require(coins[_name] == address(0), "ExistCoin");
        require(_feeNumerator <= FEE_DENOMINATOR, "InvalidFeeSetting");
        require(_fixedFee >= 0 && _feeNumerator >= 0, "LessThan0");
        if (_addr == address(0)) {
            address deployedERC20 = address(
                new ERC20Tradable(_name, _symbol, _decimals)
            );
            coins[_name] = deployedERC20;
            coinsName.push(_name);
            coinDetails[_name] = Coin(
                deployedERC20,
                _feeNumerator,
                _fixedFee,
```

```
                NATIVE_WRAPPED_COIN_TYPE
            );
        } else {
            coins[_name] = _addr;
            coinsName.push(_name);
            coinDetails[_name] = Coin(
                _addr,
                _feeNumerator,
                _fixedFee,
                NON_NATIVE_TOKEN_TYPE
            );
        }
        string[] memory tokenArr = new string[](1);
        tokenArr[0] = _name;
        uint[] memory valArr = new uint[](1);
        valArr[0] = type(uint256).max;
        btsPeriphery.setTokenLimit(tokenArr, valArr);
    }
```

Severity and Impact Summary

It is necessary to ensure that a token can only be represented by one address in the mapping.

Recommendation

Add a duplicate address check.

JAVASCORE - BTPMESSAGECENTER - FEEGATHERING OPTIMIZATIONS

Finding ID: FYEO-IB-14
Severity: Low
Status: Remediated

Description

Fees are paid according to block heights. A loop to calculate the next height could be replaced by a simple calculation.

Proof of Issue

File name: bmc/src/main/java/foundation/icon/btp/bmc/BTPMessageCenter.java
Line number: 413

```
    // feeGathering
    BMCProperties properties = getProperties();
    Address feeAggregator = properties.getFeeAggregator();
    long feeGatheringTerm = properties.getFeeGatheringTerm();
    long feeGatheringNext = properties.getFeeGatheringNext();
    if (services.size() > 0 && feeAggregator != null &&
            feeGatheringTerm > 0 &&
            feeGatheringNext <= currentHeight) {
        String[] svcs = ArrayUtil.toStringArray(services.keySet());
        sendFeeGathering(feeAggregator, svcs);
        while (feeGatheringNext <= currentHeight) {
            feeGatheringNext += feeGatheringTerm;
        }
        properties.setFeeGatheringNext(feeGatheringNext);
        setProperties(properties);
    }
```

Severity and Impact Summary

Calculating the next fee height can be expensive.

Recommendation

Use `feeGatheringNext = feeGatheringTerm + remainder` where

```
remainder = feeGatheringTerm * (1 + (currentHeight - feeGatheringNext) /
feeGatheringTerm)
```

JAVASCORE - BTPMESSAGECENTER - RELAYER BOND CAN BE 0

Finding ID: FYEO-IB-15
Severity: Low
Status: Remediated

Description

Bonds are paid to the contract by relayers who wish to register. This bond is allowed to be 0. With a bond of 0, anyone can become a relayer, and thus the incentive for them to be well-behaved is removed. This can happen when the bond is set, or as soon as the contract is created.

Proof of Issue

File name: bmc/src/main/java/foundation/icon/btp/bmc/BTPMessageCenter.java
Line number: 1150

```
@External
public void setRelayerMinBond(BigInteger _value) {
    requireOwnerAccess();
    if (_value.compareTo(BigInteger.ZERO) < 0) {
        throw BMCException.unknown("minBond must be positive");
    }
    RelayersProperties properties = relayers.getProperties();
    properties.setRelayerMinBond(_value);
    relayers.setProperties(properties);
}
```

Severity and Impact Summary

A bond of 0 allows for dis-incentivized relayers to register.

Recommendation

Use an absolute minimum value for the bond set during construction.

JAVASCORE - BTPTOKENSERVICE - REENTRANCY IN `REFUND()`

Finding ID: FYEO-IB-16
Severity: Low
Status: Remediated

Description

Rentrancy occurs when native coin is transferred before changing the state on which the transfer depends. This can happen in `refund()` as the balance of the account is updated after the amount has been transferred. To exploit this, one would have to consistently trigger `refund()`.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 853

```
private void handleResponse(BigInteger sn, TransferResponse response) {
                    // ....
                    refund(coinName, owner, locked, fee);
        }
        // ...
        transactions.set(sn, null);
```

Line number: 769

```
private void refund(String coinName, Address owner, BigInteger locked,
BigInteger fee) {
    logger.println("refund", "coinName:", coinName, "owner:", owner,
"locked:", locked, "fee: ", fee);
    // unlock and add refundable
    Balance balance = getBalance(coinName, owner);
    BigInteger value = locked.subtract(fee);
    balance.setLocked(balance.getLocked().subtract(locked));
    try {
        if (name.equals(coinName)) {
            Context.transfer(owner, value);
        } else {
            _transferBatch(Context.getAddress(), owner, List.of(coinName),
List.of(value));
        }
    } catch (Exception e) {
        if (!owner.equals(Context.getAddress())) {
            balance.setRefundable(balance.getRefundable().add(value));
        }
    }
    setBalance(coinName, owner, balance);
}
```

Severity and Impact Summary

Reentrancy can be exploited by privileged actors to repeat failed transactions.

Recommendation

Mark the transaction as completed before calling `refund()`.

RELAYER - ICON LIGHT CLIENT: MINIMUM NUMBER OF VOTES IS NOT ENFORCED

Finding ID: FYEO-IB-17
Severity: Low
Status: Remediated

Description

There is no minimum requirement for the number of validators. If there is only one validator, i.e., `len(validators) = 1`, then the verifier would still return true. If there is no valid vote, since `votesCount` is valid, is it is greater than or equal to `(2*len(validators))/3` which is `0` if `len(validators) = 1`.

Proof of Issue

File name: icon/verifier.go
Line number: 129-131

```
    if votesCount < (2*len(validators))/3 {
        return false, fmt.Errorf("insufficient votes")
    }
```

Severity and Impact Summary

No votes required if there is one validator.

Recommendation

We recommend requiring a minimum number of validators to be at least 2.

SOLIDITY - OUTDATED SOLIDITY VERSION SPECIFIED IN MULTIPLE CONTRACTS

Finding ID: FYEO-IB-18
Severity: Low
Status: Open

Description

Multiple contracts use outdated solidity version, prior to large 0.8.0 upgrade.

Proof of Issue

File Name: BEP20.sol

Line Number: 19

```
pragma solidity >=0.5.0 <=0.8.0;
```

Severity and Impact Summary

Using an outdated solidity version leaves code susceptible to many security vulnerabilities and breaking changes.

Recommendation

Pin this to a more recent version >=0.8.0

SOLIDITY - USE OF ZERO ADDRESS TO REPRESENT NATIVE TOKEN

Finding ID: FYEO-IB-19
Severity: Low
Status: Remediated

Description

The zero address is used for the address of the native coin. The zero address is also the default value of an uninitialized (, address) mapping. This means that any other uninitialized values also share this address.

Proof of Issue

File Name: BTSCore.sol

Line Number: 67

```
function initialize(
    string calldata _nativeCoinName,
    uint256 _feeNumerator,
    uint256 _fixedFee
) public initializer {
    owners[msg.sender] = true;
    listOfOwners.push(msg.sender);
    emit SetOwnership(address(0), msg.sender);
    nativeCoinName = _nativeCoinName;
    coins[_nativeCoinName] = address(0);
    coinsName.push(_nativeCoinName);
    coinDetails[_nativeCoinName] = Coin(
        address(0),
        _feeNumerator,
        _fixedFee,
        NATIVE_COIN_TYPE
    );
}
```

Severity and Impact Summary

Although coin name is used for most authority checks, it is still dangerous considering there are also no checks that any two coins can have the same address.

Recommendation

Change the address of the native coin to another value.

JAVASCORE - BTPMESSAGECENTER - SACKING IS NOT IN USE

Finding ID: FYEO-IB-20
Severity: **Informational**
Status: **Open**

Description

Links have assosiated sack properties. They can be set, but their usage is currently commented out. Their propagation is also not used.

Proof of Issue

File name: bmc/src/main/java/foundation/icon/btp/bmc/BTPMessageCenter.java
Line number: 627

```
    @External
    public void sendMessage(String _to, String _svc, BigInteger _sn, byte[]
_msg) {
        // ...

        // TODO (txSeq > sackSeq && (currentHeight - sackHeight) > THRESHOLD)
? revert
```

Line number: 226

```
@External(readonly = true)
public BMCStatus getStatus(String _link) {
    // ...
    // status.setRx_height_src(link.getRxHeightSrc());
    // status.setBlock_interval_dst(link.getBlockIntervalDst());
    // status.setBlock_interval_src(link.getBlockIntervalSrc());
    // status.setSack_term(link.getSackTerm());
    // status.setSack_next(link.getSackNext());
    // status.setSack_height(link.getSackHeight());
    // status.setSack_seq(link.getSackSeq());
```

Line number: 692

```
private void sendSack(BTPAddress link, long height, BigInteger seq)
```

Severity and Impact Summary

Sack sequence numbers and heights are not in use. Unused code should be refactored.

Recommendation

Either put the sack terms to use or remove them for efficiency.

JAVASCORE - BTPMESSAGECENTER - `LINK.ROTATE` NOT IN USE

Finding ID: FYEO-IB-21
Severity: **Informational**
Status: **Open**

Description

Rewards are distributed to a set of relayers. The code for rotating to the next relayer, as in the Solidity contract, is provided but never used.

Proof of Issue

File name: bmc/src/main/java/foundation/icon/btp/bmc/Link.java
Line number: 78

```
public Relay rotate(long currentHeight, long msgHeight, boolean hasMsg)
```

Severity and Impact Summary

Unused code increases gas costs during deployment and hinder code clarity.

Recommendation

Make use of this function or remove it.

JAVASCORE - BTPTOKENSERVICE - BLACKLIST RESPONSE CODE USED FOR TOKEN LIMITS

Finding ID: FYEO-IB-22
Severity: **Informational**
Status: **Remediated**

Description

Token limits messages are exchanged and handled between services. The token service checks the status code of a limit request using a blacklist response. Since the values are the same, this will not have undesired side effects, but these might occur if code changes in the future.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 954

```
private void handleChangeTokenLimit(String net, BigInteger sn,
TokenLimitResponse response) {
    // ...
        if (BlacklistResponse.RC_OK.equals(code)) {
```

Severity and Impact Summary

Code changes can have unwanted side effects.

Recommendation

Use `TokenLimitResponse` to check the response code.

JAVASCORE - BTPTOKENSERVICE - OPTIMIZATION IN `BALANCEOF()`

Finding ID: FYEO-IB-23
Severity: **Informational**
Status: **Remediated**

Description

The external function `balanceOf()` is used obtain the balance of an address for a specific coin. If the coin name corresponds to the native coin, the balance of the user's native coin is returned. This check could be performed first determining the balance of the user in case of other types of coins.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 376

```
Balance balance = getBalance(_coinName, _owner);
Address _addr = coinAddresses.get(_coinName);
if (_addr == null && !_coinName.equals(name)) {
    return balance.addUserBalance(BigInteger.ZERO);
}
Coin _coin = coinDb.get(_coinName);
if (_coinName.equals(name)) {
    BigInteger icxBalance = Context.getBalance(_owner);
    return balance.addUserBalance(icxBalance);
}
```

Severity and Impact Summary

Optimizing queries will lead to reduced gas cost

Recommendation

Immediately check

```
if (_coinName.equals(name)) {
    BigInteger icxBalance = Context.getBalance(_owner);
    return balance.addUserBalance(icxBalance);
}
```

upon the entrance of the function.

JAVASCORE - BTPTOKENSERVICE - OPTIMIZATION IN `TRANSFERBATCH()`

Finding ID: FYEO-IB-24
Severity: **Informational**
Status: **Remediated**

Description

Function `transferBatch()` is used to transfer multiple amounts across different coins. One of its requirements can be performed earlier in code to avoid a small overhead.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 479

```
@Payable
@External
public void transferBatch(String[] _coinNames, BigInteger[] _values, String
_to) {
    require(_coinNames.length == _values.length, "Invalid arguments");
    List<String> coinNameList = new ArrayList<>();
    List<BigInteger> values = new ArrayList<>();
    int len = _coinNames.length;
    require(len > 0, "Zero length arguments");
```

The final requirement could be performed at the beginning to avoid a few extra operations.

Severity and Impact Summary

Gas cost can be slightly larger than necessary in failed queries.

Recommendation

Fail as early as possible by performing the length check earlier.

JAVASCORE - BTPTOKENSERVICE - PUBLIC FUNCTION IS NOT EXTERNAL

Finding ID: FYEO-IB-25
Severity: **Informational**
Status: **Remediated**

Description

Public functions are called by external contracts. Function `getBalance()` is marked as public, but not used externally

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 732

```
public Balance getBalance(String coinName, Address owner)
```

Severity and Impact Summary

Code sanitization makes the codebase clear to maintain.

Recommendation

Mark the function as `private` if it is not intended to be used publicly.

JAVASCORE - BTPTOKENSERVICE - RECLAIMING SETS USABLE AMOUNT TO 0

Finding ID: FYEO-IB-26
Severity: **Informational**
Status: **Remediated**

Description

Users can reclaim coins after they have been refunded due to failed operations. For non-native coins, the `TokenService` adds the current usable amount to the amount that can be refunded. This can block the user from using a deposited balance and will require them to reclaim the full amount and transfer it again.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 432

```
    @External
    public void reclaim(String _coinName, BigInteger _value) {
        require(_value.compareTo(BigInteger.ZERO) > 0, "_value must be
positive");
        checkUintLimit(_value);

        Address owner = Context.getCaller();
        Balance balance = getBalance(_coinName, owner);

require(balance.getRefundable().add(balance.getUsable()).compareTo(_value) > -
1, "invalid value");
        require(isRegistered(_coinName), "Not registered");

balance.setRefundable(balance.getRefundable().add(balance.getUsable()));
        balance.setUsable(BigInteger.ZERO);
        balance.setRefundable(balance.getRefundable().subtract(_value));
        setBalance(_coinName, owner, balance);

        if (name.equals(_coinName)) {
            Context.transfer(owner, _value);
        } else {
            _transferBatch(Context.getAddress(), owner, List.of(_coinName),
List.of(_value));
        }
    }
```

Severity and Impact Summary

Reclaiming will force users to redeposit non native coins.

Recommendation

Subtract the minimum amount required from usable balance.

JAVASCORE - BTPTOKENSERVICE - TOKENLIMITS CAN BE 0

Finding ID: FYEO-IB-27
Severity: **Informational**
Status: **Remediated**

Description

Token limits can be set to 0, preventing trading of that token.

Proof of Issue

File name: bts/src/main/java/foundation/icon/btp/bts/BTPTokenService.java
Line number: 188

```
    @External
    public void setTokenLimit(String[] _coinNames, BigInteger[] _tokenLimits)
{
        //...
            require((_tokenLimits[i].compareTo(BigInteger.ZERO) >= 0),
                    "Invalid value");`
```

Severity and Impact Summary

Token limits can be set to 0, preventing trading of that token.

Recommendation

Require `> 0`.

JAVASCORE - BTPTOKENSERVICE - `RESPONSEERROR` IS NOT USED

Finding ID: FYEO-IB-28
Severity: **Informational**
Status: **Open**

Description

Function `responseError()` in BTPTokenService is not in use. Error handling is done by exception throwing.

Severity and Impact Summary

Unused code hinders code clarity.

Recommendation

Remove or make use of the unused function.

RELAYER - BSC LIGHT CLIENT: WRONG CLIENT NAME

Finding ID: FYEO-IB-29
Severity: **Informational**
Status: **Remediated**

Description

The client name in error output should be `bsc` instead of `hmny`.

Proof of Issue

File name: bsc/client.go
Line number: 24,30

```
    func newClients(urls []string, bmc string, l log.Logger) (cls []*Client,
bmcs []*BMC, err error) {
    for _, url := range urls {
        clrpc, err := rpc.Dial(url)
        if err != nil {
            l.Errorf("failed to create hmny rpc client: url=%v, %v", url, err)
            return nil, nil, err
        }
        cleth := ethclient.NewClient(clrpc)
        clbmc, err := NewBMC(common.HexToAddress(bmc), cleth)
        if err != nil {
            l.Errorf("failed to create bmc binding to hmny ethclient: url=%v,
%v", url, err)
            return nil, nil, err
        }
```

Severity and Impact Summary

Incorrect information.

Recommendation

We recommend changing `hmny` to `bsc`.

SOLIDITY - MISLEADING `REQUIRE` STATEMENT IN `TRANSFER()` FUNCTION

Finding ID: FYEO-IB-30
Severity: **Informational**
Status: **Remediated**

Description

A `require` statement checking whether a token is initialized within the `transfer()` function in BTSCore.sol is misleading. In the case `_erc20Address != address(0)` is false, the expression returns a `UnregisterCoin` message. However this could make it seem like the coin *needs to be unregistered*, opposed to something more clear.

Proof of Issue

File Name: BTSCore.sol

Line Number: 449

```
require(_erc20Address != address(0), "UnregisterCoin");
```

Severity and Impact Summary

This is an informational finding, with very small security impact.

Recommendation

Update the message to be more detailed/accurate.

SOLIDITY - UNNECESSARY `TEMP` VARIABLE INSIDE LOOP

Finding ID: FYEO-IB-31
Severity: **Informational**
Status: **Open**

Description

The loop is using the `temp` variable for no reason.

Proof of Issue

File Name: BSHProxy.sol

Line Number: 198

```
function tokenNames() external view returns (string[] memory _names) {
    _names = new string[](numOfTokens);
    uint256 temp = 0;
    for (uint256 i = 0; i < tokenNamesList.length; i++) {
        if (tokenAddr[tokenNamesList[i]] != address(0)) {
            _names[temp] = tokenNamesList[i];
            temp++;
        }
    }
    return _names;
}
```

Severity and Impact Summary

This has no security impact, but is redundant code that can be removed.

Recommendation

Remove the `temp` variable.

SOLIDITY - USE OF `THIS.<>` NOTATION FOR LOCAL FUNCTION CALLS

Finding ID: FYEO-IB-32
Severity: **Informational**
Status: **Open**

Description

This is a gas optimization opportunity. Using `this.<>` for local function calls is more expensive than normal local calls.

Proof of Issue

File Name: BTSCore.sol

Line Numbers: 373, 640, 736...

Severity and Impact Summary

There is no security implication, however this is unnecessarily expensive and prevalent throughout the codebase.

Recommendation

Do not use `this.<>` notation. Use normal local calls.

SOLIDITY - WIDESPREAD USE OF FLOATING PRAGMAS

Finding ID: FYEO-IB-33
Severity: **Informational**
Status: **Open**

Description

Floating pragmas (pragmas not tied to a specific version) are prominent throughout the codebase. It is recommended that solidity is pinned to a specific version for consistency in expected behavior.

Proof of Issue

File Name: BSHProxy.sol

Line Number: 19

```
pragma solidity >=0.5.0 <=0.8.0;
```

Severity and Impact Summary

This is an informational finding, due to unexpected behavior that can arise from different versions. In the example above, the solidity version is neither pinned, nor required to be up-to-date.

Recommendation

Pin to newer version.

SOLIDITY - `LINKS` MAPPING CURRENTLY SET TO INTERNAL FOR TESTING, SHOULD BE SET TO PRIVATE

Finding ID: FYEO-IB-34
Severity: **Informational**
Status: **Remediated**

Description

Intended visibility should be only the current contract. Currently set to current contract and all children.

Proof of Issue

File name: solidity/bmc/contracts/BMCManagement.sol

Line number: 33

```
mapping(string => Types.Link) internal links; // should be private,
temporarily set internal for testing
```

Severity and Impact Summary

There is no security impact since there are no other relevant contracts calling this function outside those used for testing.

Recommendation

Make sure to change this before deploying to mainnet.

SOLIDITY - ABICODERV2 IS SPECIFIED, BUT THIS IS REDUNDANT AS V2 IS THE DEFAULT

Finding ID: FYEO-IB-35
Severity: **Informational**
Status: **Open**

Description

`abicoderv2` is specified in BTSCore.sol, but this is not necessary because this is the default used. Note that abicoderv2 performs more sanity checks on inputs and supports more types. However it is also more expensive, and can make contract calls revert that did not revert with abicoderv1 when they contain data that does not conform to the parameter types.

Proof of Issue

File Name: BTSCore.sol

Line Number: 3

```
pragma abicoderv2
```

Severity and Impact Summary

This is an informational finding with no security implications.

Recommendation

Remove redundant code.

OUR PROCESS

METHODOLOGY

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Figure 2: Methodology Flow

KICKOFF

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

RAMP-UP

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

THE CLASSIFICATION OF VULNERABILITIES

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions
- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations