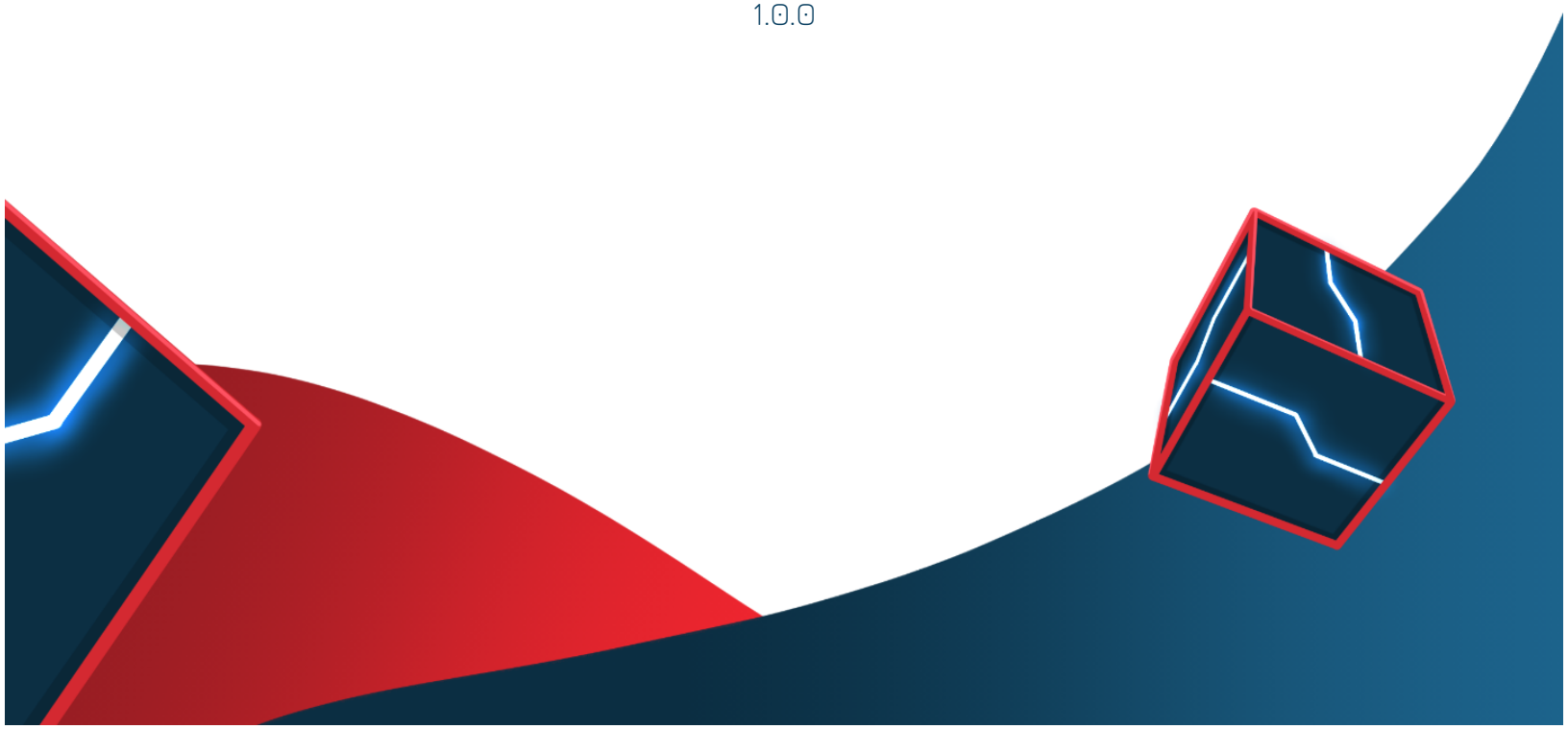
22nd July 2023

BlockApex WhiteBox Code Review Report for Lightlink Bridge

Version

1.0.0



1. Executive Summary

1.1 Report Objectives

This document details the source code review of Lightlink Bridge Validator and Keeper. The purpose of the assessment was to perform the whitebox testing of the Bridge's validator and Keeper before going into production and identify potential threats and vulnerabilities.

1.2 Scope of Work

Type	Details
Target System Name	LightLink Validator LightLink Keeper
Source Code URL(s)/	https://github.com/pellartech/ll-bridge-validator/commit/34977df32ecd8079701181cb632d4dbb2b119aac https://github.com/pellartech/ll-bridge-keeper/commit/17a7d47e8b8473d8e34b44681f8a0d35bdb6aa47
Type of Test	White-Box Testing
Language	TypeScript

1.3 Project Objectives:

Conduct a thorough code-review and vulnerability research on Lightlink Bridge's Validator and keeper code.

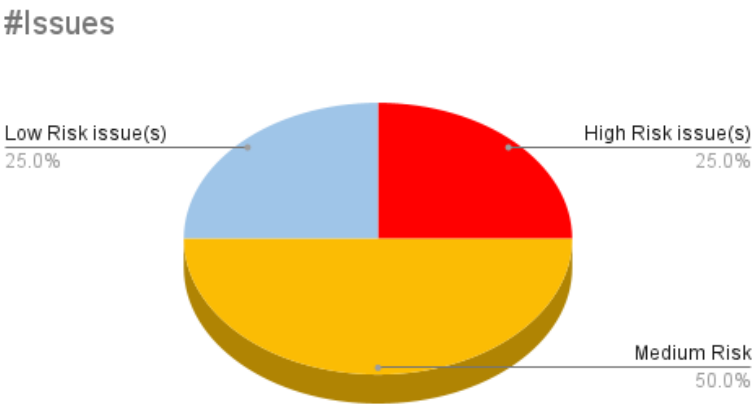
1.4 Results

The codebase is written in good quality. No significant issue was identified which can affect the functionality of the bridge. Error handling should be improved. For overall understanding of the codebase, proper commenting should be applied.

1.5 Summary of Findings

Our team found:

#Issues	Severity Level
0	Critical
1	High Risk issue(s)
2	Medium Risk issue
1	Low Risk issue(s)



2 Methodology

2.1 Planning

Our general approach during this code audit was as follows:

1. Code review

The first step of our audit was a thorough code review. We read and go through the code, ensuring a comprehensive understanding of the bridge's architecture not limited to the superficial aspects of the code.

2. Static analysis

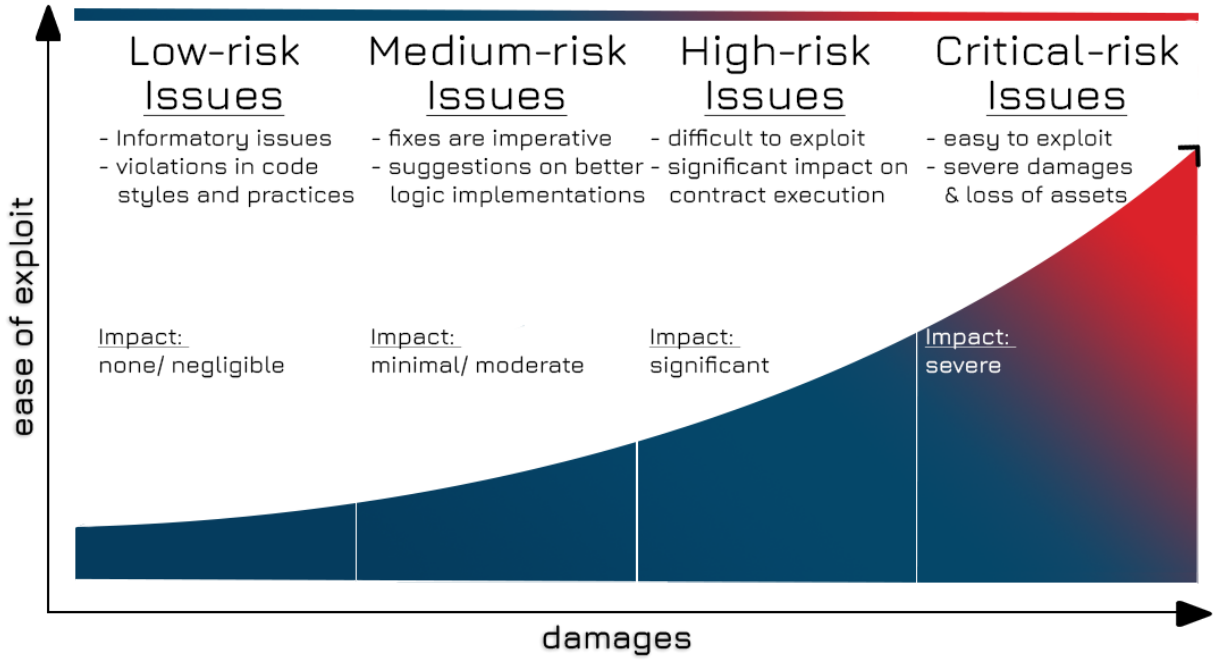
The next stage in our audit involved conducting a rigorous static analysis using SonarQube. This powerful tool allowed us to analyze the codebase in a non-runtime environment. With SonarQube, we were able to detect & verify bugs, vulnerabilities, and code smells to mitigate any risks and improve overall code quality.

3. Dynamic Analysis

Our final audit stage consisted of dynamic analysis. This process involved integrating test functions within the code and running the program with a range of inputs. Our goal was to trigger and observe any suspicious or unexpected behaviors that might emerge during runtime. This approach helped us to identify potential issues that only manifest themselves under specific conditions or when the system is operational.

2.2 Risk Classification

This report is adhering to the classification standards defined as per OWASP. The summarized version is presented below.

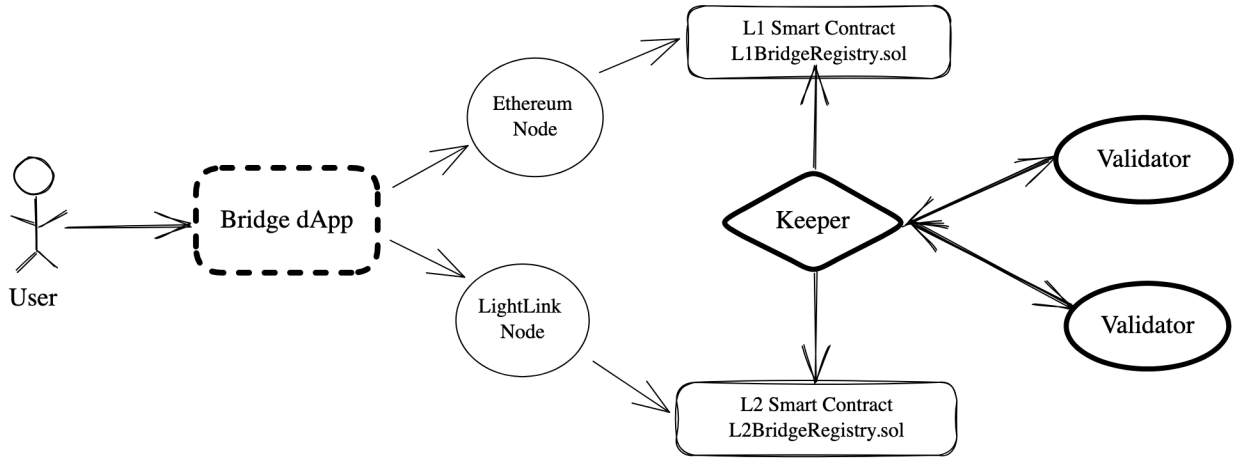


3 Light Link Bridge Architecture

3.1 System Architecture

The LightLink Bridge system architecture is based on a Proof-of-Authority mechanism with an external validator set that stakes its individual reputation in order to earn incentives for the trades on the LightLink bridge. This ensures a robust and interconnected framework enabling seamless asset transfers between Ethereum's Layer 1 and Layer 2. The bridge architecture consists of several key components, each playing a crucial role in the overall operation of the bridge. The bridge architecture consists of three main components, the front-end dApp, validator nodes and a single keeper node.

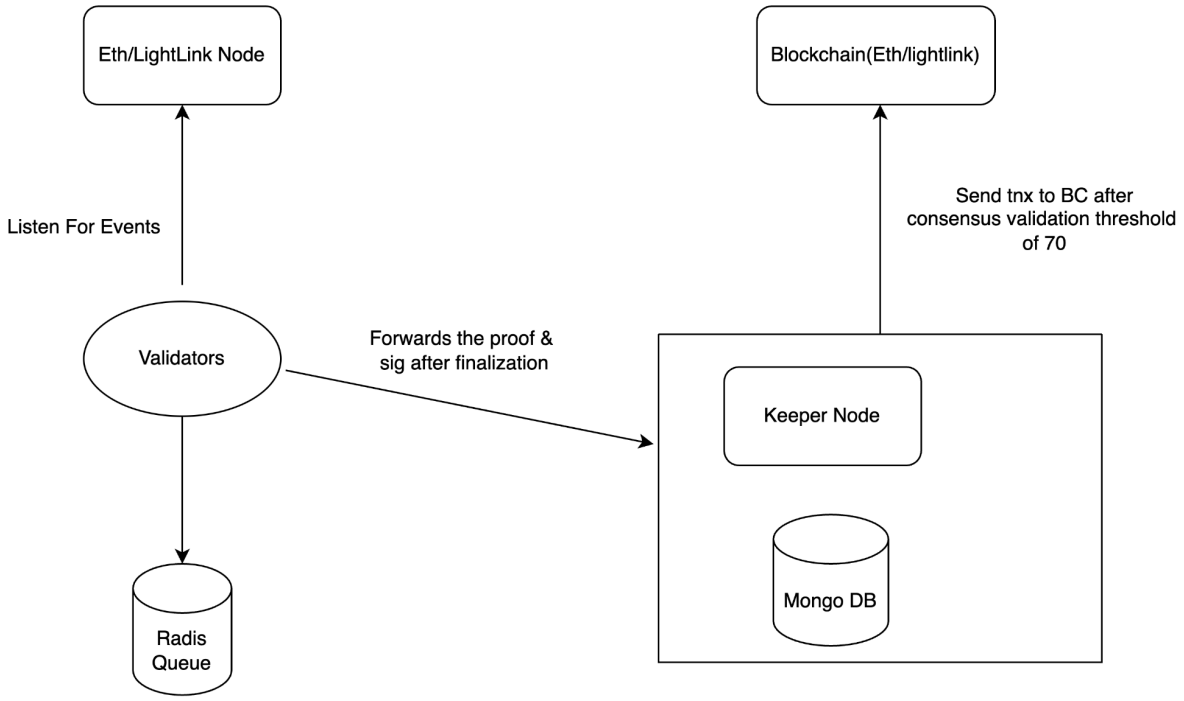
3.2 Architecture Diagram



Note:

While the provided documentation for the LightLink architecture is commendable and provides an essential overview, it could benefit from further detailed elaboration. A comprehensive breakdown of each component's functionality, interactions, and the specific roles they play in the broader architecture could enhance the depth and clarity of the document. Additionally, providing more detailed technical insights, workflows, and possible edge cases could facilitate a more robust understanding of the system. This will not only aid in improved transparency but also contribute to more effective and inclusive future audits, and security assessments.

3.3 Workflow of Keeper and Bridge



4. Findings

4.1 Potential Risks of Unauthorized Access on Redis

Severity	Likelihood	Affected Files
High	Medium	*src/config/environments/production.ts*

Description:
Redis is used by both Validator and Keeper. Under current implementation of the validator and keeper code, there was no authentication applied for redis connection. If the same code is deployed on production without using proper authentication mechanism, it can lead to potential RCE on the redis server. Open redis servers are prone to Remote Code Execution issues.

```
export default class RedisProvider {
@inject(LLogger.name) private _logger: LLogger

protected redis: RedisClientType
public readonly REDIS_PORT = CONFIG.REDIS.PORT
public readonly REDIS_HOST = CONFIG.REDIS.HOST
public readonly REDIS_PREFIX = CONFIG.REDIS.PREFIX

constructor(@inject(LLogger.name) logger: LLogger) {
this._logger = logger
this.redis = createClient({
url: `redis://${CONFIG.REDIS.HOST}:${CONFIG.REDIS.PORT}`
})

this.redis.on('error', err => {
this.disconnect()
this._logger.info(`Redis Client Error: ${err}`)
})
this.connect().then(() => {
this._logger.info('Redis Client Connected')
})
}
```

/ll-bridge-keeper-17a7d47e8b8473d8e34b44681f8a0d35bdb6aa47/src/providers/redis.ts

Proof Of Concept (POC)

File:
“ll-bridge-keeper-17a7d47e8b8473d8e34b44681f8a0d35bdb6aa47/src/config/environments/production.ts”

```
REDIS: {
HOST: String(process.env.REDIS_HOST || '127.0.0.1'),
PORT: Number(process.env.REDIS_PORT || 6379),
PREFIX: String(process.env.REDIS_PREFIX || '__ll_bridge_keeper__')
},
```

Mitigation:
The production code must incorporate the use of a Redis instance with the appropriate credentials in order to address potential unauthorized access to Redis.

Developer Response:
All validators are private and run via a docker package prepared by the development team. There is no way to access the packages in an unauthorized manner and all validators may only be run via a whitelisting process via the distributed docker solution.
Redis also does not store any security data.

Auditor Response:
The response from developers is acceptable and acknowledged. Open Redis instance are prone to RCE issues. We reported the issue in that sense. It will be a good practice to use credentials on redis too.

4.2 Potential DOS Risk due to whitelisted validators

Severity	Likelihood	Affected Files
Medium	Low	*src/modules/v1/chain/chain.indexer.ts*

Description:
A temporary DOS can happen if following scenarios are carried out

1. User submits a transaction for bridging
2. Validator submits a pending proof in DB
3. Validators are updated on chain by admin
4. Keeper will submit transaction on chain but as validators are updated hence validation will not be successful
5. Keeper will repeatedly send request until it is successful
6. If there are 10 such proofs in DB then it can cause temporary DOS as in the `produceValidatedProofs()` method , `getVerifiedProofs()` is taking a limit of 10 , and it will fetch 10 such proofs which can cause issues.

```
public async produceValidatedProofs() {
const requests = await Promise.all([
this._v1BridgeRepository.getVerifiedProofs({
chain: ChainSupported.Lightlink,
consensusPowerThreshold:
CONFIG.MODULES.V1.BRIDGE.CONTRACTS.LIGHTLINK.CONSENSUS_POWER_THRESHOLD,
limit: 10
})
// this._v1BridgeRepository.getVerifiedProofs({
// chain: ChainSupported.Ethereum,
// consensusPowerThreshold:
CONFIG.MODULES.V1.BRIDGE.CONTRACTS.ETHEREUM.CONSENSUS_POWER_THRESHOLD,
// limit: 10 // })
])
// const items = [...requests[0], ...requests[1]]
const items = [...requests[0]]

for (const item of items) {
await this.submitProofToChain(item)
}}
```

src/modules/v1/chain/chain.service.ts

Mitigation:
It is important to include a mechanism that ensures when onchain validators are updated, the corresponding whitelisted validators should also be updated to prevent any potential scenarios.

Developer Response:
Configuration operating with validators is as follows:
1 x Validator with 40 voting power run by Pellar
2 x Validator with 20 voting power run by Pellar
Voting power required for authorisation of bridge transfers = 80
At the moment in the unlikely case that a validator is removed from the pool then we can monitor any proofs that are invalid due to this circumstance and process the withdrawal.
To mitigate the chance of us not being able to reach 80 voting power we are onboarding partners who will be running validators each with 20 power of their own.

Auditor Response:
The response from developers is acceptable and acknowledged.

4.3 Potential Risk of Price Manipulation Due to single source

Severity	Likelihood	Affected Files
Medium	High	src/config/environments/production.ts

Description:
Keeper is fetching prices from a single oracle source i.e gate.io , which is defined in `src/config/environments/production.ts`

```
EXTERNAL_PARTIES: {
GATE_IO: {
API_ROOT_V4: 'https://api.gateio.ws/api/v4'
}}}
```

In case of a price manipulation on gate.io , bridge can be affected.

Mitigation:
It is advised to define multiple price oracles and aggregate the price or use Chainlink's price oracle as it aggregate price from multiple sources.

Developer Response:
We are not currently fetching prices from any oracle so this is a non-issue. If/when we do then we will make sure to either use an aggregated source such as Chainlink or aggregate sources on our own.

Auditor Response:
The response from developers is acceptable and acknowledged.

4.4 Information Disclosure due to Improper Error Handling

Severity	Likelihood	Affected Files
Low	High	*src/core/apiError.ts*

Description:
When a Keeper node is deployed it deploys certain API endpoints to which the validator can see the proofhashes & signatures. If the data is not sent in proper format then the endpoint returns the full stack error. This stack shows the full path of the deployed keeper, which is a bad practice & leaks information.

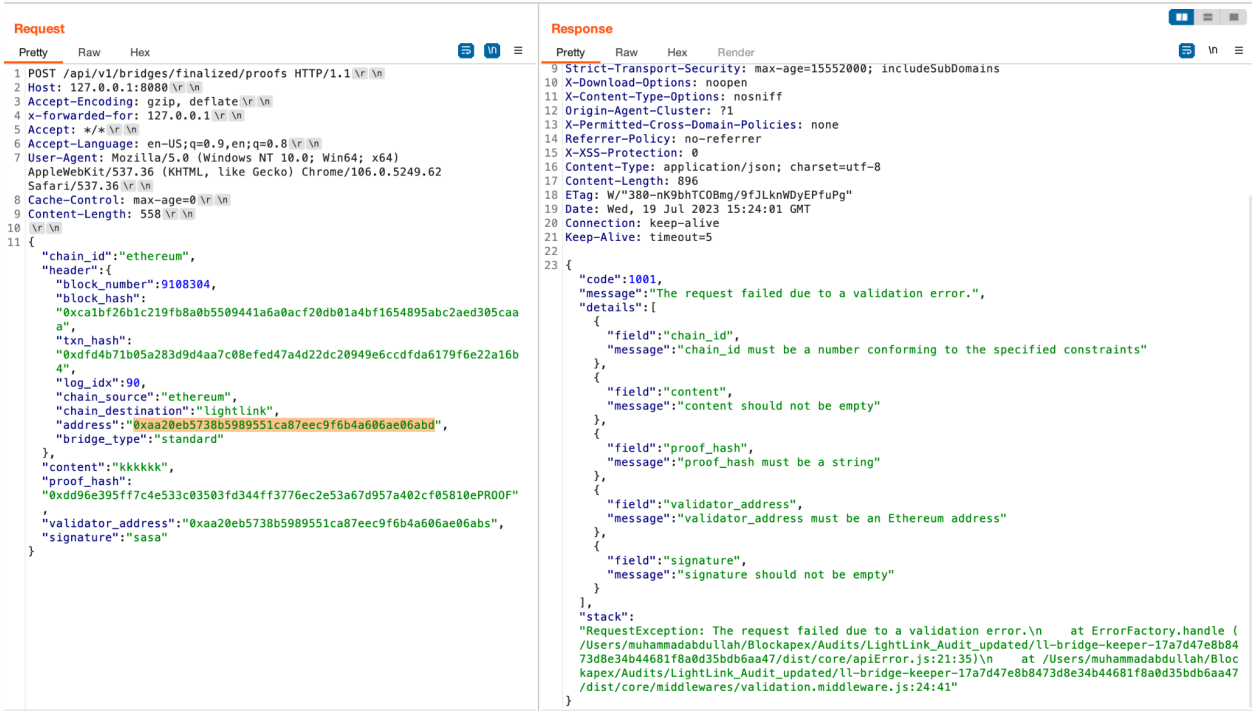
Technical Description:
In 'src/core/apiError.ts' if the type is 'RequestException' then it is returning exceptionInstance.resolve(). resolve() method is also returning the "this.stack".

```
export default class ErrorFactory {
static handle(type: Keys, error: IException, context?: IExceptionContext): IException {
const exceptionInstance = new exceptionMaps[type](error, context)
switch (type) {
case 'BizException':
case 'RequestException':
return exceptionInstance.resolve()
}
}
```

```
public resolve() {
return {
status: this.status,
code: this.code,
message: this.message,
details: this.details,
context: this.context,
stack: this.stack
}}}
```

Proof Of Concept (POC)

1. Make a bad parameterized request at the /api/v1/bridges/finalized/proofs
2. Analyze the response



Request

```
POST /api/v1/bridges/finalized/proofs HTTP/1.1
Host: 127.0.0.1:8080
Accept-Encoding: gzip, deflate
x-forwarded-for: 127.0.0.1
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62
Safari/537.36
Cache-Control: max-age=0
Content-Length: 558

{
  "chain_id":"ethereum",
  "header":{
    "block_number":9108304,
    "block_hash":
    "0xca1bf26b1c219fb8a0b5509441a6a0acf20db01a4bf1654895abc2aed305caa
    a",
    "txn_hash":
    "0xdfd4b71b05a283d9d4aa7c08efed47a4d22dc20949e6ccdfda6179f6e22a16b
    4",
    "log_idx":90,
    "chain_source":"ethereum",
    "chain_destination":"lightlink",
    "address":"0xaa20eb5738b5989551ca87eec9f6b4a606ae06abd",
    "bridge_type":"standard"
  },
  "content":"kkkkkk",
  "proof_hash":
  "0xdd96e395ff7c4e533c03503fd344ff3776ec2e53a67d957a402cf05810ePROOF"
  ,
  "validator_address":"0xaa20eb5738b5989551ca87eec9f6b4a606ae06abs",
  "signature":"sasa"
}
```

Response

```
Strict-Transport-Security: max-age=15552000; includeSubDomains
X-Download-Options: noopen
X-Content-Type-Options: nosniff
Origin-Agent-Cluster: ?1
X-Permitted-Cross-Domain-Policies: none
Referrer-Policy: no-referrer
X-XSS-Protection: 0
Content-Type: application/json; charset=utf-8
Content-Length: 896
ETag: W/"380-nK9bhTCOBmg/9fJLknWDyEPfuPg"
Date: Wed, 19 Jul 2023 15:24:01 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "code":1001,
  "message":"The request failed due to a validation error.",
  "details":[
    {
      "field":"chain_id",
      "message":"chain_id must be a number conforming to the specified constraints"
    },
    {
      "field":"content",
      "message":"content should not be empty"
    },
    {
      "field":"proof_hash",
      "message":"proof_hash must be a string"
    },
    {
      "field":"validator_address",
      "message":"validator_address must be an Ethereum address"
    },
    {
      "field":"signature",
      "message":"signature should not be empty"
    }
  ],
  "stack":
  "RequestException: The request failed due to a validation error.\n    at ErrorFactory.handle (
  /Users/muhammadabdullah/Blockapex/Audits/LightLink_Audit_updated/ll-bridge-keeper-17a7d47e8b84
  73d8e34b44681f8a0d35bdb6aa47/dist/core/apiError.js:21:35)\n    at /Users/muhammadabdullah/Bloc
  kapex/Audits/LightLink_Audit_updated/ll-bridge-keeper-17a7d47e8b8473d8e34b44681f8a0d35bdb6aa47
  /dist/core/middlewares/validation.middleware.js:24:41"
}
```

Mitigation:

It's advised not to return the stack and detailed messages, rather it is recommended to only return generic error codes upon a bad request. This obscurity prevents attackers from gaining insights into the backend infrastructure from the error information, thereby limiting their scope for black box exploitation. It mitigates the potential reconnaissance performed by malicious actors.

Developer Response:

Access to the stack has been removed in production mode.

Auditor Response:

The response from developers is acceptable and acknowledged.

5. SonarQube Test Reports

We utilized SonarQube, a powerful tool designed to inspect and analyze the entire codebase for Keeper and Validator. The primary purpose of SonarQube is to conduct static code analysis, systematically examining the software without executing it, to identify potential vulnerabilities, bugs, and code smells. Code smells refer to certain characteristics or patterns in the code that could potentially indicate deeper problems. SonarQube provided us with several insightful suggestions for enhancing our code's quality and security. The following points highlight these suggested improvements and identified issues:

5.1 SonarQube Issues Report For Keeper

Rule	Severity	File	Line	Description
typescript:S4144	MAJOR	keeper:src/core/apiError.ts	24	Update this function so that its implementation is not identical to the one on line 15.
typescript:S4323	MINOR	keeper:src/helpers/utils.ts	4	Replace this union type with a type alias.
typescript:S4323	MINOR	keeper:src/modules/v1/bridge/bridge.repository.ts	38	Replace this union type with a type alias.
typescript:S1135	INFO	keeper:src/modules/v1/chain/chain.service.ts	537	Complete the task associated to this "TODO" comment.
typescript:S4822	MAJOR	keeper:src/modules/v1/system/system.indexer.ts	25	Consider using 'await' for the promise inside this 'try' or replace it with 'Promise.prototype.catch(...)' usage.
typescript:S4822	MAJOR	keeper:src/modules/v1/sys	37	Consider using 'await' for the promise inside this 'try' or replace it with

		tem/system.indexer.ts		'Promise.prototype.catch(...)' usage.
typescript:S4325	MINOR	keeper:src/providers/external.ts	71	This assertion is unnecessary since it does not change the type of the expression.

5.2 SonarQube Issues Report For Validator

Rule	Severity	File	Line	Description
typescript:S4144	MAJOR	validator:src/core/apiError.ts	24	Update this function so that its implementation is not identical to the one on line 15.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	72	Complete the task associated to this "TODO" comment.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	140	Complete the task associated to this "TODO" comment.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	213	Complete the task associated to this "TODO" comment.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	35	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S4323	MINOR	validator:src/modules/v1/chain/chain.service.ts	55	Replace this union type with a type alias.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	206	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.

Rule	Severity	File	Line	Description
typescript:S4144	MAJOR	validator:src/core/apiError.ts	24	Update this function so that its implementation is not identical to the one on line 15.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	72	Complete the task associated to this "TODO" comment.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	140	Complete the task associated to this "TODO" comment.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	379	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	556	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	730	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	884	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	1038	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S3776	CRITICAL	validator:src/modules/v1/chain/chain.service.ts	1197	Refactor this function to reduce its Cognitive Complexity from 18 to the 15 allowed.
typescript:S4822	MAJOR	validator:src/modules/v1/system/sy	18	Consider using 'await' for the promise inside this 'try' or replace it with

Rule	Severity	File	Line	Description
typescript:S4144	MAJOR	validator:src/core/apiError.ts	24	Update this function so that its implementation is not identical to the one on line 15.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	72	Complete the task associated to this "TODO" comment.
typescript:S1135	INFO	validator:src/modules/v1/bridge/bridge.service.ts	140	Complete the task associated to this "TODO" comment.
		stem.indexer.ts		'Promise.prototype.catch(...)' usage.

6. NPM Audit Report

Npm offers a dependency auditing option which should be run by the developer to ensure usage of secure dependencies and to mitigate the risk of supply chain attacks.

Vulnerable Dependency	Title	Severity	Via	Effects	Range
word-wrap	word-wrap vulnerable to Regular Expression Denial of Service	moderate	word-wrap	-	<1.2.4
semver	semver vulnerable to Regular Expression Denial of Service	moderate	simple-update-notifier	-	<=5.7.1
nodemon	-	moderate	simple-update-notifier	-	2.0.19 - 2.0.22
tough-cookie	tough-cookie Prototype Pollution vulnerability	moderate	tough-cookie	-	<4.1.3
minimist	Prototype Pollution in minimist	critical	-	optimist	<=0.2.3
mongoose	Mongoose Prototype Pollution vulnerability	critical	-	-	6.0.0 - 6.11.2
optimist	Prototype Pollution in minimist	critical	minimist	swig-templates	>=0.6.0

swig-templates	Arbitrary local file read vulnerability during template rendering	critical	swig-templates, optimist	swig-email-templates	*
class-validator	SQL Injection and Cross-site Scripting in class-validator	critical	-	-	<0.14.0
@aws-sdk/client-cognito-identity	-	high	@aws-sdk/client-sts	@aws-sdk/credential-provider-cognito-identity	3.12.0 - 3.54.1
@aws-sdk/client-sts	-	high	fast-xml-parser	@aws-sdk/client-cognito-identity, @aws-sdk/credential-providers	<=3.54.1
cheerio	-	high	css-select	swig-email-templates	0.19.0 - 1.0.0-rc.3
css-select	Inefficient Regular Expression Complexity in nth-check	high	nth-check	cheerio	<=3.1.0
fast-xml-parser	fast-xml-parser vulnerable to Regex Injection via Doctype Entities, fast-xml-parser vulnerable to Prototype Pollution through tag or attribute name	high	fast-xml-parser	@aws-sdk/client-sts	<=4.2.3

json5	Prototype Pollution in JSON5 via Parse Method	high	-	-	<1.0.2
luxon	Luxon Inefficient Regular Expression Complexity vulnerability	high	-	-	1.0.0 - 1.28.0
@aws-sdk/credential-provider-cognito-identity	-	high	@aws-sdk/client-cognito-identity	-	3.12.0 - 3.347.0
@aws-sdk/credential-providers	-	high	@aws-sdk/client-cognito-identity, @aws-sdk/client-sts, @aws-sdk/credential-provider-cognito-identity	-	<=3.347.0
swig-email-templates	-	high	cheerio, swig-templates	-	>=2.0.0

Mitigation

It is advised to run 'npm audit' before putting it into production and using 'npm audit fix' to mitigate the possible issues

Disclaimer: Source Code Review

The source code review is conducted for the purpose of providing constructive feedback and suggestions to enhance the quality and security of the software. This review is not meant to criticize or undermine the efforts of the developers or individuals involved in the project. The reviewer shall not be liable for any loss, damages, or issues resulting from the use or implementation of the feedback provided. The scope of this review is limited to the specific code presented, and any subsequent changes to the code may not be reflected in this evaluation. It is essential to understand that the reviewer cannot guarantee the identification of all potential issues or vulnerabilities. The developers are responsible for maintaining the software's security and quality. All information discovered during the review shall be treated as confidential. The use of third-party components is assumed to be compliant. The reviewer's assessment is independent, and there is no affiliation with any organization related to the software under review. The developers are encouraged to perform multiple audits,periodically reevaluate and update the code to address potential issues. By proceeding with the review results, all parties agree to accept the terms of this disclaimer.