deBridge - Node

Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 26th, 2021 - December 10th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/10/2021	Gabi Urrutia
0.2	Draft Review	12/11/2021	Gabi Urrutia
1.0	Remediation Plan	05/16/2022	Afaq Abid
1.1	Remediation Plan Review	05/17/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Afaq Abid	Halborn	Afaq.Abid@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

deBridge engaged Halborn to conduct a security assessment on their Node beginning on November 26th, 2021 and ending December 10th, 2021. deBridge is a cross-chain interoperability and liquidity transfer protocol that allows truly decentralized transfer of assets between various blockchains. The cross-chain intercommunication of deBridge programs is powered by the network of independent oracles/validators that are elected by the deBridge governance.

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned one full-time security engineer to audit the security of the assets in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to achieve the following:

- Ensure that deBridge Node functions are intended.
- Identify potential security issues with the deBridge Node.

In summary, Halborn identified few security risks that were addressed by deBridge team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual view of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of programs and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Storing private keys and assets securely
- Application Logic Flaws
- Fuzzing of all input parameters
- Areas where insufficient validation allows for hostile input
- Research into architecture and purpose.
- Static Analysis of security for scoped program and imported functions. (nodejsscan, Dependency-Check, eslint)
- Manual Assessment for discovering security vulnerabilities.
- Ensuring correctness of the codebase. (eslint)
- Dynamic Analysis on Node functions and data types.
- Known vulnerabilities in 3rd party / OSS dependencies.

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to debridge-finance/debridge-launcher repository.

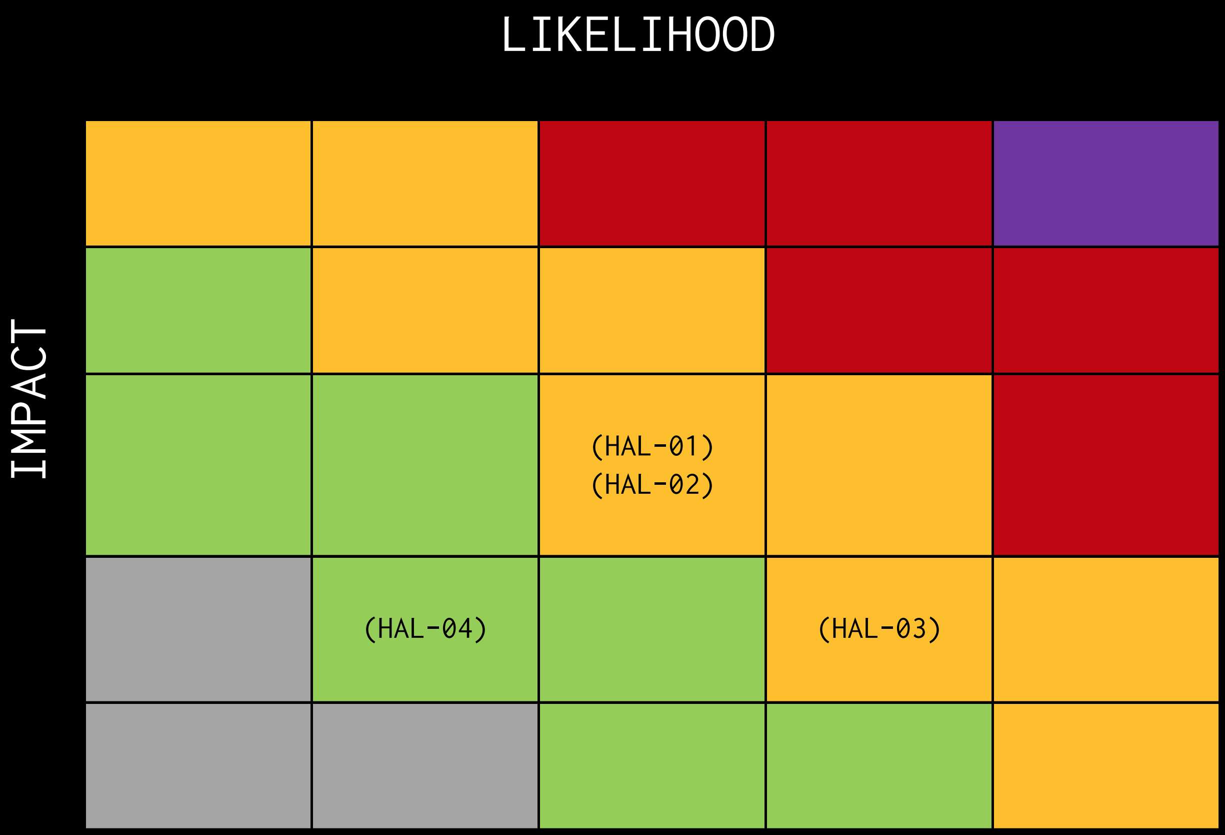
Commit ID: 785901ea4d15c232444d98ee361c885468a49cae

OUT-OF-SCOPE:

External libraries.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	3	1	0



🟨	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥	🟥
🟩	🟩	🟨 (HAL-01) (HAL-02)	🟨	🟥
⬜	🟩 (HAL-04)	🟩	🟨 (HAL-03)	🟨
⬜	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - USE OF POTENTIALLY INSECURE FUNCTION	Medium	NOT APPLICABLE
HAL-02 - DOCKER COMPOSE SECURITY MISCONFIGURATION	Medium	SOLVED - 05/11/2022
HAL-03 - LACK OF RESOURCES AND RATE LIMITING	Medium	SOLVED - 05/11/2022
HAL-04 - USING PACKAGES WITH KNOWN VULNERABILITIES	Low	SOLVED - 05/11/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) USE OF POTENTIALLY INSECURE FUNCTION - MEDIUM

Description:

The deBridge Node invokes a potentially insecure function findOne() that could introduce NoSQL injection vulnerabilities. As a result, this web application security issue allows a malicious party to extract data from the application.

Code Location:

Listing 1: debridge-launcher/debridge_node/src/subscribe/SubscribeHandler.ts (Line 32)

```
30 private async uploadConfig() {
31 for (const config of chainConfigs) {
32     const configInDd = await this.supportedChainRepository.findOne({
33     chainId: config.chainId,
34     });
```

Listing 2: debridge-launcher/debridge_node/src/subscribe/actions/CheckAssetsEventAction.ts (Line 47)

```
47 const confirmNewAction = await this.confirmNewAssetEntityRepository.findOne({
48     debridgeId: submission.debridgeId,
49 });
```

Listing 3: debridge-launcher/debridge_node/src/subscribe/actions/AddNewEventsAction.ts (Line 58)

```
56 const submissionId = sendEvent.returnValues.submissionId;
57 const submission = await this.submissionsRepository.findOne({
58     submissionId,
59 });
```

Listing 4: debridge-launcher/debridge_node/src/subscribe/actions/AddNewEventsAction.ts (Line 116)

```
114 async process(chainId: number, from: number = undefined, to: number = undefined) {
115     this.logger.verbose(`checkNewEvents ${chainId}`);
116     const supportedChain = await this.supportedChainRepository.findOne({
117         chainId,
118     });
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

To prevent NoSQL injections, you must always treat user input as untrusted. Use a sanitization library. For example, mongo-sanitize.

Listing 5: example.ts (Line 3)

```
1 const sanitize = require('mongo-sanitize')
2 const submission = await this.submissionsRepository.findOne({
3     sanitize(submissionId),
4 });
```

Remediation Plan:

NOT APPLICABLE: The deBridge team provided feedback that they are using the TypeORM for all postress database requests. So this is not vulnerable in that case.

3.2 (HAL-02) DOCKER COMPOSE SECURITY MISCONFIGURATION - MEDIUM

Description:

Security Misconfiguration vulnerabilities are configuration weaknesses that exist in software due to a lack of proper security hardening. By adding an environment variable to the .env file and defining it in the docker-compose.yml file, it will be added to **all containers**. Therefore, all environment variables are accessible to all running containers. Besides the fact that it is unnecessary, it can introduce vulnerabilities.

Code Location:

Listing 6: debridge-launcher/docker-compose.yml (Line 8)

```
 1 version: "3.6"
 2 services:
 3   postgres:
 4     image: postgres
 5     container_name: postgres${DOCKER_ID}
 6     restart: on-failure
 7     env_file:
 8       - .env
 9     volumes:
10       - ./pgdata:/var/lib/postgresql/data
11       - ./pg-init-scripts:/docker-entrypoint-initdb.d
12     networks:
13       - debridge-node-network
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Docker-compose allows us to define environment variables to pass to running containers with the environment option. It is recommended to use this config instead of the env_file option.

Reference: Why you should split your env file

Remediation Plan:

SOLVED: The deBridge team fixed the issue by adding the appropriate checks.

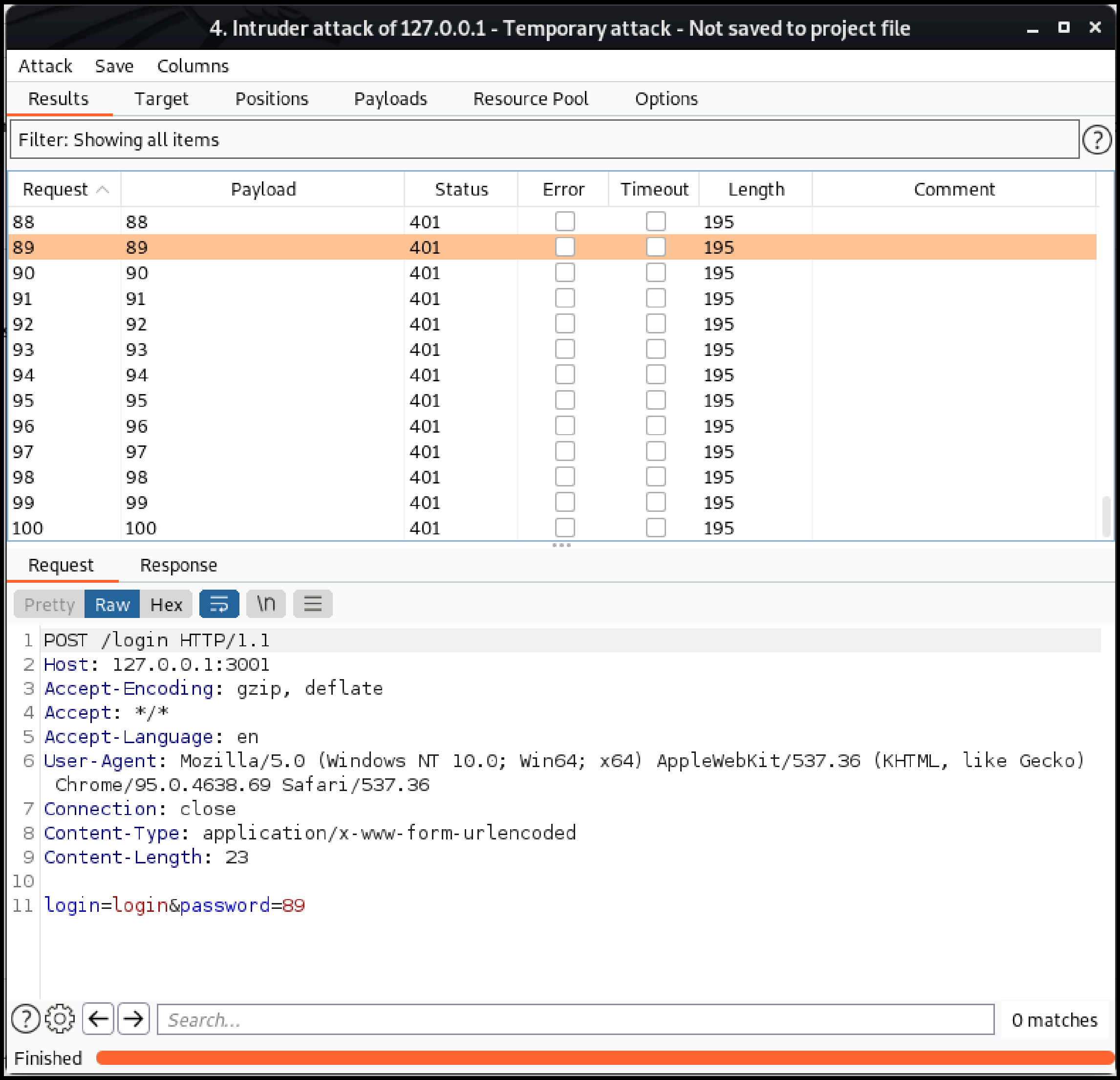
3.3 (HAL-03) LACK OF RESOURCES AND RATE LIMITING - MEDIUM

Description:

API requests consume resources such as network, CPU, memory, and storage. This vulnerability occurs when too many requests come in at the same time, and the API does not have enough compute resources to handle those requests.
An attacker could exploit this vulnerability to overload the API by sending more requests than it can handle. As a result, the API becomes unavailable or unresponsive to new requests.

Reference: CWE-770: Allocation of Resources Without Limits or Throttling

Proof-Of-Concept:



4. Intruder attack of 127.0.0.1 - Temporary attack - Not saved to project file
Attack Save Columns
Results Target Positions Payloads Resource Pool Options
Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
88	88	401	□	□	195	
89	89	401	□	□	195	
90	90	401	□	□	195	
91	91	401	□	□	195	
92	92	401	□	□	195	
93	93	401	□	□	195	
94	94	401	□	□	195	
95	95	401	□	□	195	
96	96	401	□	□	195	
97	97	401	□	□	195	
98	98	401	□	□	195	
99	99	401	□	□	195	
100	100	401	□	□	195	

Request Response
Pretty Raw Hex \n

```
POST /login HTTP/1.1
Host: 127.0.0.1:3001
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/95.0.4638.69 Safari/537.36
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 23

login=login&password=89
```

Risk Level:

Likelihood - 4

Impact - 2

Recommendation:

This vulnerability is due to the application accepting requests from users at a given time without performing request limitation checks. We recommend that you follow the following best practices:

- Implement a limit on how often a client can call the API within a defined timeframe.
- Notify the client when the limit is exceeded by providing the limit number and the time the limit will be reset.
- Define and enforce maximum data size on all incoming parameters and payloads, such as the maximum length of strings and the maximum number of elements in arrays.

Remediation Plan:

SOLVED: The deBridge team fixed the issue by adding the appropriate checks.

3.4 (HAL-04) USING PACKAGES WITH KNOWN VULNERABILITIES - LOW

Description:

The deBridge Node uses third-party dependencies to delegate handling of different types of operations, e.g. generation of documents in a specific format, HTTP communications, data parsing of a specific format, etc. However, the dependency has an expected downside where the actual application's security posture now rests on it.

Vulnerabilities List:

Title	**Package**	**Severity**
Inefficient Regular Expression Complexity	ansi-regex	Moderate
Inefficient Regular Expression Complexity	validator	Moderate
Prototype Pollution	json-schema	Moderate
SQL Injection and Cross-site Scripting	class-validator	Moderate

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is highly recommended to perform an automated analysis of the dependencies from the birth of the project and if they contain any security issues. The deBridge team needs to be aware of this and apply the required mitigation measures to secure the affected application.

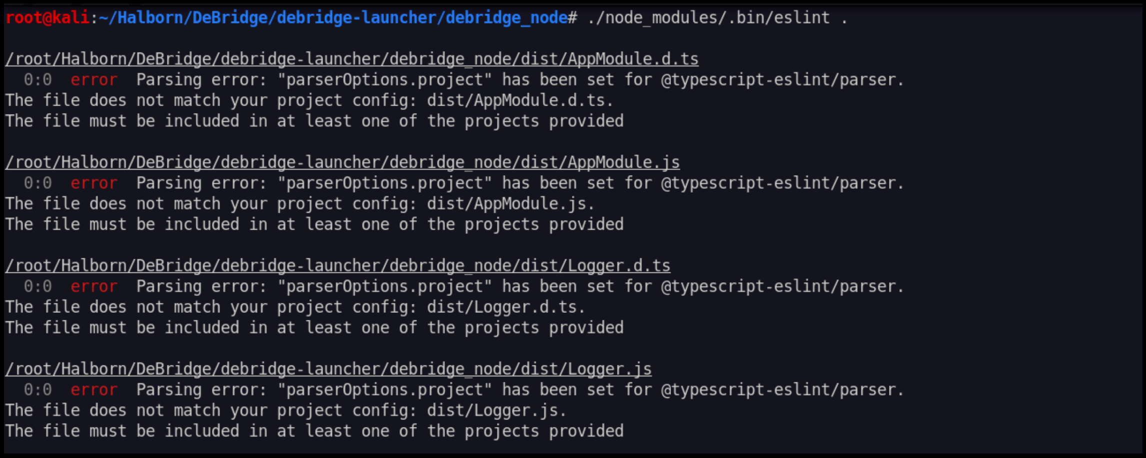
Remediation Plan:

SOLVED: The deBridge team fixed the issue by updating old packages to newer versions.

AUTOMATED TESTING

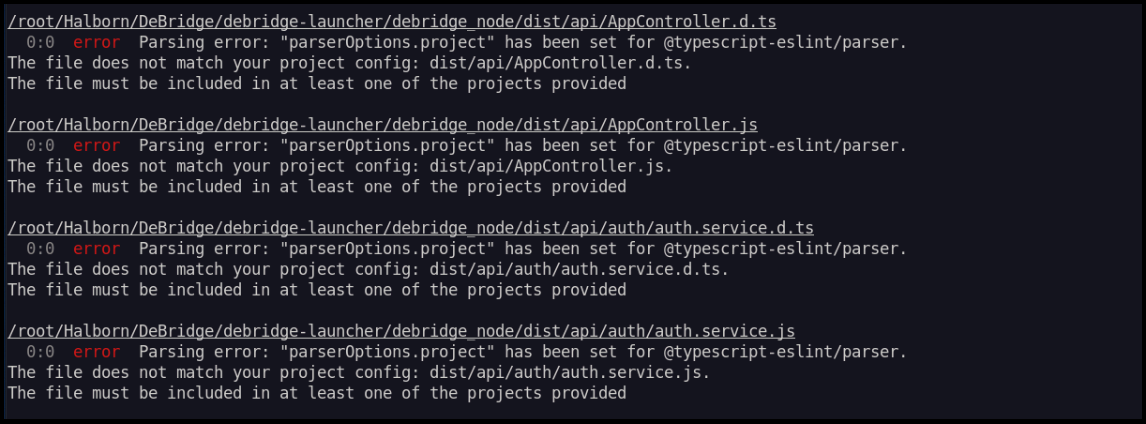
4.1 ESLint

According to the following screenshots, there could be multiple improvements on the codebase. It has been decided not to put these issues in the report in detail because these issues do not pose any security risk.



```
root@kali:~/Halborn/DeBridge/debridge-launcher/debridge_node# ./node_modules/.bin/eslint .

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/AppModule.d.ts
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/AppModule.d.ts.
The file must be included in at least one of the projects provided

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/AppModule.js
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/AppModule.js.
The file must be included in at least one of the projects provided

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/Logger.d.ts
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/Logger.d.ts.
The file must be included in at least one of the projects provided

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/Logger.js
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/Logger.js.
The file must be included in at least one of the projects provided
```

Figure 1: ESLint Results - 1

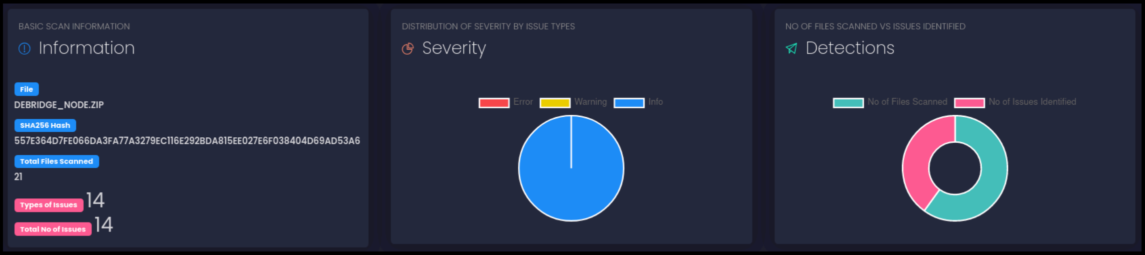


```
/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/api/AppController.d.ts
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/api/AppController.d.ts.
The file must be included in at least one of the projects provided

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/api/AppController.js
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/api/AppController.js.
The file must be included in at least one of the projects provided

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/api/auth/auth.service.d.ts
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/api/auth/auth.service.d.ts.
The file must be included in at least one of the projects provided

/root/Halborn/DeBridge/debridge-launcher/debridge_node/dist/api/auth/auth.service.js
  0:0  error  Parsing error: "parserOptions.project" has been set for @typescript-eslint/parser.
The file does not match your project config: dist/api/auth/auth.service.js.
The file must be included in at least one of the projects provided
```

Figure 2: ESLint Results - 2

4.2 NodeJSScan

As a result of the scans completed with NodeJSScan, many tests were carried out and some issue outputs were produced as a result of these tests. These generated issues were analyzed manually.



BASIC SCAN INFORMATION

Information

File
DEBRIDGE_NODE.ZIP

SHA256 Hash
557E364D7FE066DA3FA77A3279EC116E292BDA815EE027E6F038404D69AD53A6

Total Files Scanned
21

Types of Issues 14

Total No of Issues 14

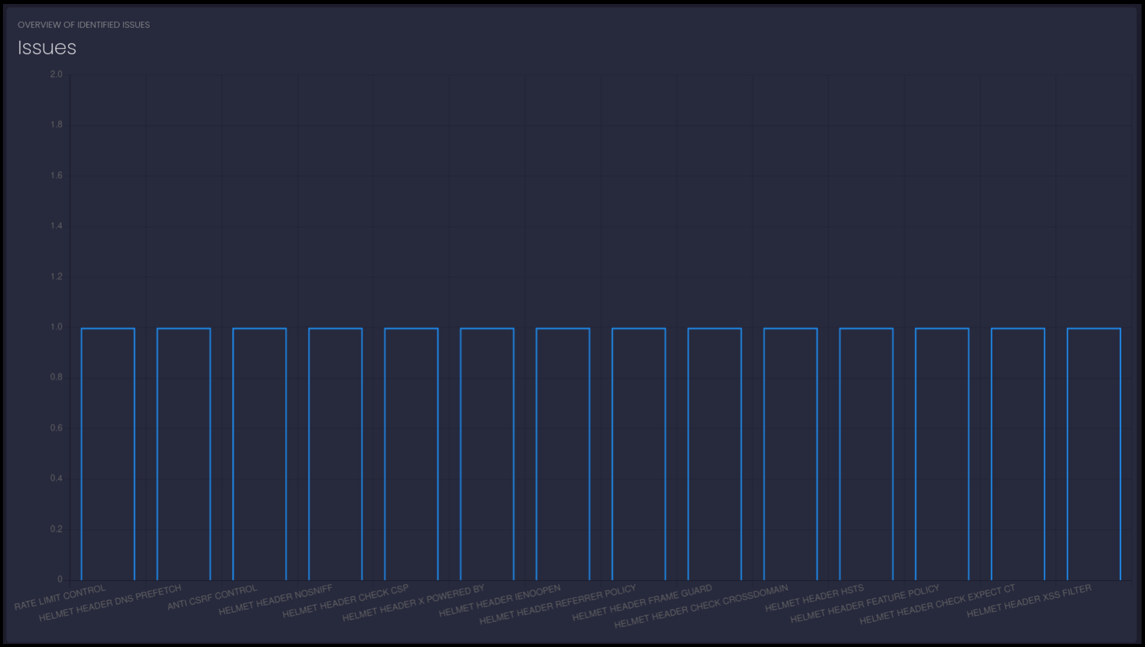
DISTRIBUTION OF SEVERITY BY ISSUE TYPES

Severity

NO OF FILES SCANNED VS ISSUES IDENTIFIED

Detections

Figure 3: NodeJSScan Results - 1



OVERVIEW OF IDENTIFIED ISSUES

Issues

Figure 4: NodeJSScan Results - 2

THANK YOU FOR CHOOSING

HALBORN