Coverity Scan Static Analysis Report

Hardhard Enterprises

**T3** 2022

Statement of Intent

Overview

This document aims to provide a record of static code analysis performed on a specific issue from the Coverity SAST scan for the NASA ION Open-Source code 4.1.1 project.

The primary purpose of this document is to validate the issue identified via the automated detection process to eliminate false positives.

Depending on findings, secondary purposes can include but are not limited to listing/providing recommended fixes alongside a list of attack vectors and potential exploits for consideration.

Reporting Best Practices

Please ensure best practices are kept when completing the document via regularly updating the Acronyms and Abbreviations table alongside any iterations made to the Document History table. This will allow other members to identify any updates and progress made across trimesters easily.

When using code snippets, please use screenshots that are clear and easy to read, alternatively, use words built-in code formatted found [here](https://appsource.microsoft.com/en-us/product/office/WA104382008?tab=Overview).

Document Naming Conventions

Naming conventions for this file are as follow; SAR\_{CID}. For example, when investigating issue 123456 the file name would be SAR\_123456.docx

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Dates** | **Version** | **Author** | **Comments** |
| 23/11/2022 | 1.0 | Jesse Ludeman | Initial document |
| 29/11/2022 | 1.1 | Jesse Ludeman | Update vulnerability investigation |
|  |  |  |  |

Table of Content

Contents

[Introduction 3](#_Toc119848724)

[Objective 3](#_Toc119848725)

[Scope 3](#_Toc119848726)

[Acronyms and Abbreviations 3](#_Toc119848727)

[Code Review and Analysis 4](#_Toc119848728)

[Outcomes 4](#_Toc119848729)

[Observations 4](#_Toc119848730)

[Supporting Evidence 4](#_Toc119848731)

[Conclusions and Recommendations 4](#_Toc119848732)

[References 4](#_Toc119848733)

[Appendix 4](#_Toc119848734)

# Introduction

## Objective

The primary objective of this analysis is to determine whether the defects identified in the Coverity Report for the ION Open Source 4.1.1 project are:

* Indeed, defects.
* Potentially exploitable.

The secondary objective of this analysis, where applicable, is to provide the following:

* Recommendation(s) to fix.
* Any exploit for consideration.

## Scope

This static code analysis is limited to the ***Performance inefficiencies*** type defect identified in the following CIDs: 1520896

# Acronyms and Abbreviations

Please keep an updated list of acronyms and abbreviations used throughout the report.

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| DTN | Delay/Disruption Tolerant Network |
| ION | Interplanetary Overlay Network |
|  |  |

# Code Review and Analysis

## Outcomes

When performing static code analysis using the ION Open Source 4.1.1 dashboard for CID 1520896, there is a low impact problem that involves a big parameter being passed into a function.

## Observations

A “big parameter passed by value” indicates that a parameter is being passed a value in a poor or inefficient manner. In the code snippet below, the jsonObject job is passed into the getNewRuleId function, which under normal circumstances is acceptable. However, this object has a size of 2064 bytes when passed in by value, which currently exceeds the high threshold of 512 bytes for this function. Because this is being passed by value, this typically results in a more expensive operation, depending on the size of the object.

## Supporting Evidence

|  |
| --- |
| static int getNewRuleId(jsonObject *job*, uint16\_t \**ruleId*)  {      CHKZERO(ruleId);      /\* Extract the ruleID from the JSON tokens. \*/      getRuleId(job, ruleId);  } |

# Conclusions and Recommendations

To resolve this low impact problem, I would suggest passing a pointer to the job object, instead of passing it by value. This means that when the program calls this function, the entire object isn’t passed to the function, only a reference to it.

An attacker could potentially exploit this issue by passing a large json object to the function, however the attacker would most likely need some level of visibility of the source code prior to exploiting the vulnerability. Additonallity, an unsuspecting user could inadvertently pass a large json object to the function, which could cause undesirable behaviour. Either of these scenarios could cause the function to fail, and subsequently cause the program’s execution to fail and stop.

|  |
| --- |
| static int getNewRuleId(jsonObject \**job*, uint16\_t \**ruleId*)  {      CHKZERO(ruleId);      /\* Extract the ruleID from the JSON tokens. \*/      getRuleId(job, ruleId);  } |

References

Appendix

Include additional information/documentation here to help the readers understand complex information.