**T2** 2023

Coverity Scan Static Analysis Report

Hardhard Enterprises

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 formatter 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** |
| 6/9/2023 | V0.1 | Chong Zhang | Initial document |
|  |  |  |  |
|  |  |  |  |

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 5](#_Toc119848733)

[Appendix 6](#_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 ***Untrusted divisor*** type defect identified in the following CIDs:  
***1520821***

# 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 |
| CID | Coverity Issue Identification Number |
| CWE | Common Weakness Enumeration |

# Code Review and Analysis

## Overview

Coverity identifies **CID-1520821 Untrusted divisor** as a Medium impact, and the category is insecure data handling. It appears in **generateKeyPair**: An unscrutinized value from an untrusted source used as a divisor (CWE-369). This means the divisor could be controlled by an attacker, who could cause a division by zero. This will cause the program to crash.

## Observations

The issue can be found in the generateKeyPair function in the /tc/dtka/dtka.c file. Within this function, An unscrutinized value from an untrusted source used as a divisor. In this case, it uses the tainted function getOwnNodeNbr as a divisor.

The problem comes from line 204. Using tainted expression getOwnNodeNbr() as the divisor in (unsigned int)currentTime / getOwnNodeNbr().

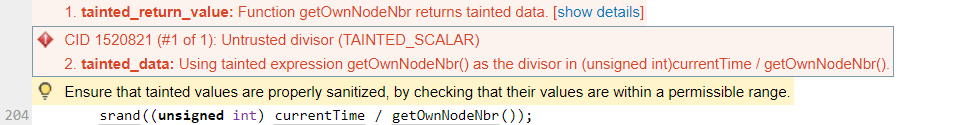


Fig. 1. Coverity Static analysis results

## Supporting Evidence

Please provide any supporting evidence, and feel free to make references to documents in the appendix.

# Conclusions and Recommendations

This is a vulnerability that can be exploited by attackers, the divisor could be controlled by an attacker, who could cause a division by zero. This will cause the program to crash.

It is recommended to repair the tainted values with proper sanitized by checking whether they are within the permissible range. In addition, you can add code to determine whether the divisor is 0 before running the division. This prevents the program from crashing.

References  
Common Weakness Enumeration (n.d.) CWE-369: Divide By Zero, <https://cwe.mitre.org/data/definitions/369.html>

Appendix

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