**T1** 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** |
| 15/05/2023 | V1.0 | Adharshaan Devaraj | Initial analysis Document |
|  |  |  |  |
|  |  |  |  |

Table of Content

Contents

[Introduction 3](#_Toc133624419)

[Objective 3](#_Toc133624420)

[Scope 3](#_Toc133624421)

[Acronyms and Abbreviations 3](#_Toc133624422)

[Code Review and Analysis 4](#_Toc133624423)

[Overview 4](#_Toc133624424)

[Observations 4](#_Toc133624425)

[Supporting Evidence 4](#_Toc133624426)

[Conclusions and Recommendations 4](#_Toc133624427)

[References 5](#_Toc133624428)

[Appendix 5](#_Toc133624429)

# 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 ***High impact quality*** type defect identified in the following CIDs:  
***1520661***

# 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

## Overview

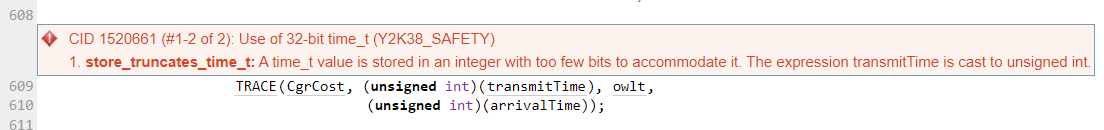
Coverity discovered a "**high impact quality**" vulnerability with the usage of casting **unsigned** INT to the 32-bit time\_t data type inside lines 609 and 610 of the **ComputeDistanceToTerminus ()** function by examining the "**libcgr.c**" file under the "**cgr**" folder of the **Bundle Protocol v7** directory. The fault being reported is a usage of 32-bit time issue, which is defined as CWE-197. The issue, also known as a Numeric Truncation Error, which happens when a mathematical operation is done with a restricted number of digits and the result is rounded to accommodate that limited number of digits, preventing the result from being completely correct.

## Observations

CGR is an important routing technique used in DTN networks that calculates the optimal path required for a bundle to reach its destination based on network topology and available links between nodes. The algorithm generates a network of alternate routes based on node contacts and connections, and optimizes routes based on various parameters such as route length, age of data to be sent, and route available bandwidth. Select a route.

## In the “**libcgr.c”** code, the **ComputeDistanceToTerminus** () function calculates the distance based on the number of hops necessary to reach the final destination node. This method requires two parameters: the current node's endpoint id and the ultimate destination node's endpoint id. This method uses a recursive strategy to search the list of endpoint IDs for the target node's endpoint ID.

## Supporting Evidence



On line 609, we see the occurrence of this issue. The error shows that the code is converting an int to an unsigned int. To be more specific, the **transmitTime** to an unsigned int. In C programming language, the unsigned int data type may only hold values in the range [0 - 65,535] or [0 - 4,294,967,295]. Both int and unsigned int types are commonly 2 or 4 bytes in size. The fundamental distinction between the two is that an int can hold both positive and negative values, but an unsigned int can only hold non-negative integers.

# Conclusions and Recommendations

This is not an exploitable vulnerability. The solution to that issue would be to convert the “**transmitTime**” and “**arrrivalTime”** variables to a signed integer data type. A signed integer can include both positive and negative integers. This must then be verified to confirm that the code is still valid. Another solution to rectify the issue would be to use a function such as gmtime().general

# References

Appendix

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