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 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** |
| 25/04/2023 | V1 | Jesse Ludeman | Initial document and 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 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 ***Use of 32-bit time\_t*** type defect identified in the following CIDs: 1520792

# 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

## Introduction

CID 1520792 has been flagged by Coverity as a High impact issue of type Use of 32-bit time\_t. This type of defect typically means that a time value has been stored in an integer that may represent a different but valid time value.

## Observations

### Introduction

This defect has been flagged in the /bpv7/cgr/libcgr.c file, and exists in the computeRoute function. This function is responsible for computing the best possible route for the given destination based on the available network topology and other routing constraints. It computes the reachability of each next-hop node, which is a measure of the probability that the destination can be reached through that node. It then uses a scoring algorithm to evaluate the potential routes based on route quality, link quality, and hop count. The function will then select the best route from the list of candidate routes and returns it to the calling function.

### The defect

We notice the first and only occurrence of this error on line 901, inside the last else statement. The error indicates that the code is casting an int data-type to an unsigned int data-type. Specifically, the route->arrivalTime to an unsigned int. The unsigned int data-type in the C language can only store values in the range from [0 – 65,535] or [0 - 4,294,967,295].

Both int and unsigned int types are typically the same size, either 2 or 4 bytes. However, the main difference between the two is that an int can represent both positive and negative numbers, while an unsigned int can only store non-negative numbers. This can be a problem, as noted by the error that’s flagged: *“In computeRoute: A 64-bit t ime\_t value is stored in a smaller width integer”*. If we assume that a negative time value has been stored in an int and then cast to an unsigned int, then this value would sit outside the allowed range for the unsigned int type.

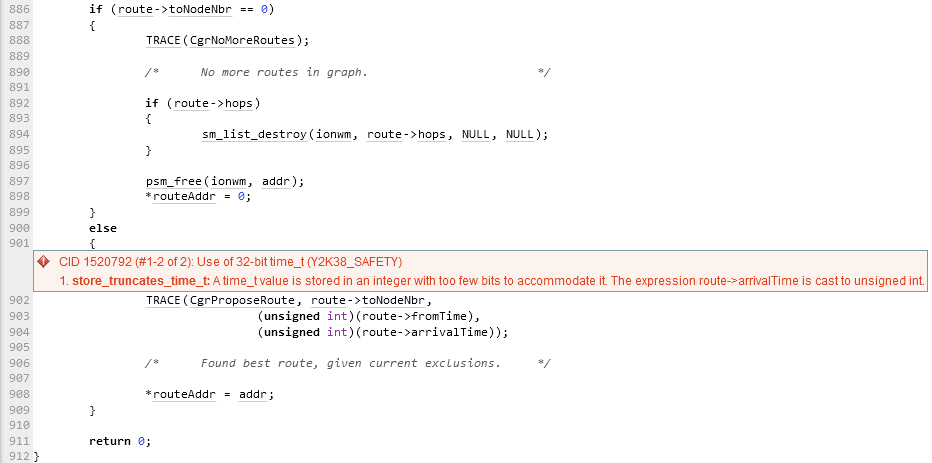


Figure 1 - store\_truncates\_time\_t error

## Supporting Evidence

# Conclusions and Recommendations

Hardhat Enterprises recommends the following changes be implemented:

* Add input validation in the bsl\_sdr\_list\_first() function to ensure that only non-negative numbers are being used for date/time values. This will prevent it from returning a very large result such as 18446744073709551615.

Furthermore, out-of-bounds access issues can introduce a risk to the security posture of the application. Whilst no immediate vulnerability has been found with this defect, it should be noted that these types of defects can be security risks. For example, they may be able to execute artbitrary code, read values from the application, or cause the system to crash.

References

Appendix