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** |
| 4/05/2023 | V1.0 | Jesse Ludeman | Initial document and investigation |
| 6/05/2023 | V1.1 | Jesse Ludeman | Final investigation complete |

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 ***High impact quality*** type defect identified in the following CIDs: 1520798

# 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 1520798 has been flagged by Coverity as a High impact issue of type *Use of 32-bit time\_t*. This type of defect typically indicates that a date time value has been stored in an integer which may store a valid, albeit different time. This function makes use of the fprintf() function, which formats program data into a json file.

## Observations

We notice that the error is first identified on line 526, which occurs inside the call to the fprintf() function. Inspecting both the fromTime and deliveryTime variables confirms they are indeed valid time\_t data types. However, note that both of them are being cast to an unsigned int data types. The main difference between an integer and unsigned integer in C is that the former can hold both negative and positive numbers, while the latter can only hold positive numbers.

Whilst we cannot inspect and confirm the actual values of these date variables, if we assume that one of these dates has a negative value assigned to it, and is then cast to an unsigned integer then this will inevitably cause problems for the program. For example, if deliveryTime has a negative date time value, and is then cast to an unsigned integer, then this would have caused Coverity to flag this as a possible issue.



Figure - First occurence of the offending error

Whilst not immediate security risk is associated with this defect, it does imply that a bad actor can potentially exploit these types of issues. One such example would be where a large positive value is misinterpreted as valid input. For example, if an unsigned int value is represented as a memory address or even a buffer size, a large positive value could cause a buffer overflow. Consider Figure 2, where the values are printed to the console, the output is: x = -1, y = 4294967295.

|  |
| --- |
| #include <stdio.h>  int main()  {      int x = -1;      unsigned int y = (unsigned int)x;      printf("x = %d, y = %u\n", x, y);      return 0;  } |

Figure 2 - Example type conversion

## Supporting Evidence

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

# Conclusions and Recommendations

In terms of avoiding these types of risks, Hardhat Enterprises suggest to carefully consider using appropriate data types and type conversions. This will ensure that valid program data is never malformed or interpreted incorrectly when cast to a different type. Furthermore, we also recommend ensuring that data is validated before being returned by the calling function.

References  
Please keep an updated references list in APA7; The Deakin referencing guide can be found [here](https://www.deakin.edu.au/__data/assets/pdf_file/0009/2236752/Deakin-guide-to-APA7.pdf).

Appendix

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