**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** |
| 6/05/2023 | V1.0 | Druween Perera | 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 ***High Impact quality*** type defect identified in the following CIDs:  
***1520645***

# 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 in the "**cgrfetch.c"** file under the **"utils"** folder of the Bundle Protocol v7 directory by converting unsigned INT to the 32-bit time\_t data type over many lines (lines 713 and 714) of the **usage ()** function. The stated issue is a 32-bit time consumption concern known as CWE-197. The issue, also known as a Numeric Truncation Error, happens when a mathematical operation with a restricted number of digits is performed.

## Observations

## This problem may be found on lines 713 and 714. The error indicates that the code is attempting to convert an int to an unsigned int. In particular, the "dispatchOffset" and "expirationOffset" to an unsigned int. The unsigned int data type in C programming language can only carry numbers in the range [0 - 65,535] or [0 - 4,294,967,295]. Integer and unsigned int types are typically 2 or 4 bytes in size. The primary difference between the two is that an int can store both positive and negative values, but an unsigned int can only store non-negative numbers.

## Supporting Evidence

# 

# 

A non-negative number in the range [0 - 4294967295] is stored as a 32-bit unsigned integer. The issue here is that if a data structure is given a value that exceeds its capacity, the application may lose data. As a result, the program's execution may fail if either the dispatchOffset or expirationOffset functions returns a number that the unsigned integer data type cannot store.

# Conclusions and Recommendations

This is not an exploitable vulnerability. The solution to that issue would be to convert the “DispactchOffset” and “ExpirationOffset” 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.

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.