**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** |
| 6/5/2023 | 1.0 | Dean Scanlon | 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 5](#_Toc119848731)

[Conclusions and Recommendations 5](#_Toc119848732)

[References 6](#_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 ***Out-of-bounds access*** type defect identified in the following CIDs:  
***1520747 and 1520738***

# 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

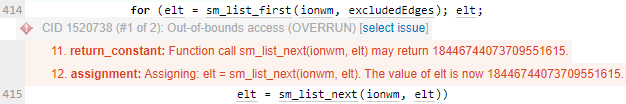
These combined issues occur in the *libcgr.c* code segment within the *cgr* directory of Bundle Protocol 7. Coverity Static Analysis Tool was able to detect a **High Impact Quality** vulnerability relating to **Memory – corruptions**. CID1520747 occurs within the disabledRoute function and CID1520738 within the edgelsExcluded function.

The value of the element index elt can potentially be passed a negative value. If this occurs, there is the possibility that it is interpreted as unsigned in which case the software may respond in unpredictable ways that are not well defined in the c language.

## 

## Observations

In these CIDs and many others within the NASA ION DTN protocol, Coverity has flagged that the assignment of the elt element index variable as a negative number may return ‘18446744073709551615’ which would be outside the boundaries of the partition->space memory buffer. This element index is a 64 bit data tpe, and as such the largest numerical value it can hold is 18446744073709551615. As specified in C standard (2011 6.3.1.3p2), when a variable assumes the value of -1 and is interpreted as an unsigned integer, the value is converted by repeatedly adding or subtracting one more than the maximum value that can be represented by the type, therefore elt is assigned as the large number as the computation is (-1 + 18446744073709551616). Examples are shown in Fig.1 and Fig.2.



*Fig.1. CID1520738 Out-of-bounds access issue*

Text

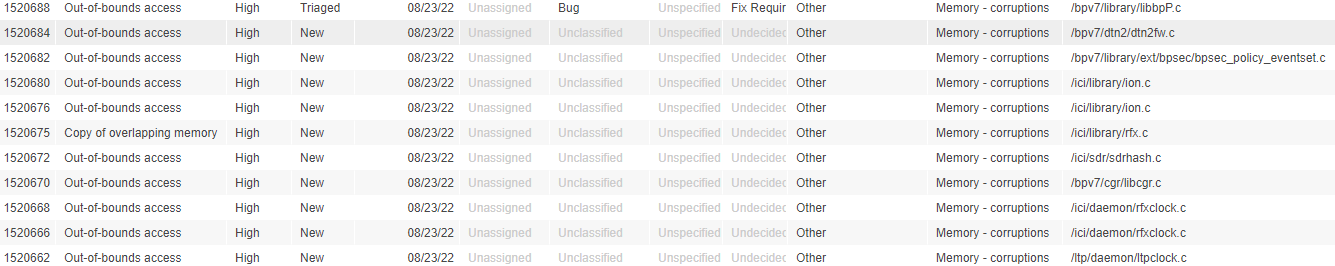
Description automatically generated with medium confidence

*Fig.2. CID1520747 Out-of-bounds access issue*

Both CIDs exist within the same library and are susceptible to the same vulnerability, with the large erroneous number pointing to an element in the linked list sm\_list\_data which contains the addresses SMs (state machines) which correspond to bundles in storage. The number of elements in this linked list is dependant on the number of bundles that are currently stored in the system. When such a large index is pointed to in this protocol, Coverity flags a potential error as the buffer partition->space is not allocated enough memory to handle such a large number. This results in a memory buffer overflow/Out-of-Bounds access issue.

Issue recurrence throughout Bundle Protocol 7 in ION DTN

Bundle Protocol 7 within NASA Interplanetary Overlay Network (ION) project contains no less than ***91*** instances in which this issue occurs, as identified by Coverity Static Analysis tool. Fig.3 demonstrates the frequency of recurrences of this issue. Note: list is non-inclusive.



*Fig.1. Non-inclusive list of instances of issue as flagged by Coverity Static Analysis Tool.*

The elt function is used to retrieve an element at a specific index from linked list sm\_list\_data in CID1520747 and CID1520738 which are being analysed in this report, however the elt function is used to access elements in several linked lists throughout the code such as:

bundle\_list - linked list containing bundles that are ready for transmission and is used extensively throughout Bundle Protocol 7 to manage bundles

xmit\_blocks - contains transmission blocks that are used to store outgoing transmission blocks for bundles.

sm\_list\_data – contains information about each bundle such as destination node, priority, destination node and transmission status

Given the frequency of occurrences of this issue in the DTN protocol and the potential for it to cause problems, efforts should be made to correct the issue as a priority. The use of elt is recurring and central to the management of bundle transmission so any errors within it’s application should be addressed.

Out-of-bounds access errors

Out-of-bounds access errors can be considered a vulnerability in that they can result in system crashes or unauthorized access to private information as documented in CWE-119 (CWE – CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer). In a worst-case scenario, attackers could potentially exploit buffer overflow vulnerabilities such as the one present in ION DTN BPv7 to change execution paths and execute code that can be tailored to damage applications, crash systems or expose private information. A best-case scenario for such code errors is a crash of the system which could negatively affect NASA projects that use the protocol.

# 

# Conclusions and Recommendations

The C language does not automatically validate the locations to which memory buffers are assigned and lacks a ‘garbage collection’ type memory management scheme as in Java or Python. As such, efforts need to be made to validate the values being passed to the elt variable. A function should be included in the code which validates the elt function returns 1 or no greater than the number of elements in the linked list sm\_list\_data. This would avoid elt taking a negative value and as a result being interpreted as an unsigned integer.

An example, albeit out of context, of a function that could be used to validate the values of elt is shown in Fig.4.

Text

Description automatically generated

Alternatively, consideration could be made for the NASA DTN protocol to be written in another programming language such as Python or Java.

References

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer (2023 January) *Common Weakness Enumeration* <https://cwe.mitre.org/data/definitions/119.html>

ISO/IEC 9899:201x. (2011 April) *International Standard: Programming languages – C* <https://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf> (6.3.1.3p2)

Appendix

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