**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** |
| 19/04/23 | 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:  
***1520708***

# 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

This issue occurs in the *bpsec\_policy rule.c* code segment within the *library/ext/bpsec/* directory of Bundle Protocol 7. Coverity Static Analysis Tool was able to detect a **High Impact Quality** vulnerability relating to **Memory – corruptions**.

An index value is passed to the buffer partition->spacewhich is larger than the number of elements in the buffer. This is classed as an Out-of-bounds access error which is documented as CWE-119 (CWE – CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer).

## Observations

The variable elt is given the value 0 on line 1010. Within the for loop on line 1021, the sm\_list\_next function is called to determine the next element index value and is supposed to do as such while cycling through the elements of sm\_list .

Graphical user interface

Description automatically generated with low confidence

Because of the lack of validation of the size of the elt variable, it is possible that it is being interpreted as unsigned and a such assigned the erroneous number. The number of elements in the partition->space buffer is recognized by Coverity to have a smaller number of elements than the large number. It can be assumed that the elt variable is a 64 bit type and as such the largest value it can hold is 18446744073709551615. As specified in C standard (2011 6.3.1.3p2), when an integer 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.

Out-of-bounds access errors

Out-of-bounds access errors are considered a vulnerability in that, if exploited, malicious actors could act to crash the system or access private information. Due to the delays in transmission of bundles across DTN and it’s reliance on physical proximity to function, any potential system crash could manifest into lengthy delays. The sensitive nature of work done by NASA means that it imperative for nodes to be able to communicate when they have an opportunity for bundle transmission.

The potential exists that attackers could also exploit buffer overflow issues to change execution paths and execute code. The result can be tailored to damage applications, crash systems or expose private information.

## 

## Supporting Evidence

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

# 

# 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 variable as 1 or no greater than the number of elements in the linked list. This would avoid elt being taking a negative value and as a result being interpreted as an unsigned integer.

The linked list in question is sm\_list and it holds the addresses of the nodes, so ensuring the value of elt is within an acceptable range is an important requirement to ensure the code functions correctly and communication between nodes is not disrupted.

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.