**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** |
| 28/03/23 | 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 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 ***Out-of-bounds access*** type defect identified in the following CIDs:  
***1520633***

# 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

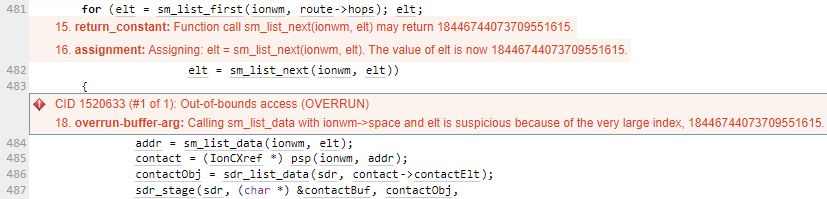
## Overview

When scanning ipnfw.c within the /bpv/ipn directory of Bundle Protocol 7, Coverity was able to identify a **High Impact Quality** vulnerability. In this case, the issue related to Out-of-Bounds access.

Within the enqueueToEntryNode function, a value is passed without being validated and as such is too large to be handled by an element index, thereby overflowing the allocated memory buffer.

## Observations

The element index elt is erroneously passed the value 18446744073709551615 when the sm\_list\_next function calls (ionwm, elt). The size of this element index is too large and as such encroaches beyond the allocated memory in the memory buffer. There is no validation when the value is passed and as such there is a possibility of a negative parameter being passed to the standard memory copy or allocation functions and being implicitly cast to a large unsigned value.



The function enqueueToEntryNode uses a linked list to hold the address of the element elt at the index address of the buffer and when being passed this large value has too many elements for the size of elements in the buffer.



Issues that can arise from Out-of-bounds access errors

Out-of-bounds access errors can be considered a vulnerability which can cause programs to crash or produce undesired results such as incorrect computations of values. Attackers can axploit this vulnerability with the aim to crash the program or execute arbitrary code. In the case of such errors within the DTN protocol across ION, the potential exists of disruption of the transmission of bundles at times when bundle transmission is possible. The risk of a system crash at a time when nodes are in physical proximity that allows transmission means that the DTN could be considered unworkable.

## Supporting Evidence

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

# 

# Conclusions and Recommendations

Values passed between functions need to be validated to avoid buffer overflow errors. C language allows direct addressing of memory locations and does not automatically validate the locations to which the memory buffer is assigned.

C language does not does not include ‘garbage collection’ type automatic memory management scheme as in Java or Python. The programming language that the DTN protocol is written in

The possibility of changing the programming language in which the DTN protocol is written should be considered. Failing that, proper validation of the parameters used should be implemented which includes bounds checking on the index being used and validation of the returned element index. For example a function could be created to perform the element index value validation.

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.