**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** |
| 22/04/2023 | V0.1 | Callam | Initial Investigation |
|  |  |  |  |
|  |  |  |  |

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 ***{Issue Category}*** type defect identified in the following CIDs:  
***{Coverity Issue CID}***

# 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

The code starts with a function called removeroute that removes a route from the **routing table**. The function takes two parameters ionwm partition and a PsMaddress pointing to the element of the routing table to be removed. The function retrieves CgR route from PsMaddress, checks to see if referenceELT field of the struct set. If it is set the function then deletes the element pointed to by referenceELT from the routing table. The function then checks to see if the PsmAddress passed as a parameter is different from referenceELT. If it is different, the function deletes the element pointed to by PsM address. The function then checks if the hops field is set, and if it is it iterates over each element in the list that is pointed to by hops.

Every element returns the address of the contact from the elements data field. If the address is NULL the function deletes the element from the list and continues to the next element. If it is not null the function retrieves IonCXref struct from the contact address and checks if the citations field of the struct is set. If it is set, the function iterates over every element in the list and for each element the function compares the element data field to the address of the current element in the hops list. If these addresses match the function deletes the element from the citations list and continues to the next element in hops.

## Observations

In figure 1 (lines 104-105) an **out of bounds** error has occurred, more specifically an OVERRUNN error. This error is occurring because the value of citationELT has become very large (maximum value that can be represented by an unsigned integer), suggesting that it has wrapped around to the maximum value of an unsigned integer due to a negative parameter being interpreted as unassigned which can happen if citationELT has been incorrectly initialized. This occurs if citationELT is already the last element in the list and sm\_list\_next() is called on it. If this occurs the loop at line 104-108 (figure 2) will continue indefinitely leading to a crash or hang.

This causes errors further on in the code as citationELT is passed to other functions.

## Supporting Evidence

Figure : Out of bounds error due to very large index

*Calendar

Description automatically generated with low confidence*.

Figure : loop goes through each element in the list until there is no more elements to process



Figure : citation ELT assigned large integer

# Conclusions and Recommendations

This code is exploitable. Attackers are able to overwrite crucial data, such as function pointers or return addresses. This can lead to execute malicious code of their choice that can allow them to gain control of the programs execution code and perform malicious acts. Attacks can also create inputs to trigger buffer overflows causing the program to crash or consume excessive resources rendering it unavailable to legititmate users. Secrets, passwords and other sensitive data can be obtained from overflowing a buffer.

Non-specific solutions to migitate risks of buffer overflows include: input validiation and bounds checking to validate input size and perform bounds checking to prevent buffer overflows. Consider safe string functions instead of unsafe string functions such as ‘strcpy’ or ‘gets’. Use secure coding guidelines such as CERT C. Avoid risky programming techniques such as direct memory manipulation, pointer arithmetic or unsafe string handling. Consider use of runtime protection mechanisms such as stack protection, integrity checks or runtime vulnerability scanners to detect and prevent buffer overflows during program execution.

The loop that iterates through contact -> citations uses sm\_list\_next to change to the next element, however it does not handle the case when sm\_list\_next returns an invalid or out of bounds address which results in an overrun. Include a condition that checks for the end of list and handle loop termination correctly.

Contact-> citations does not handle the case where contact -> citations may be NULL or unititialized which could lead to undefined behavour. Ensure contact -> citations is properly initialized to a valid list.

To fix the CitationELT infinite loop, the loop condition should be changed to check for the end of the list instead of relying on the return value from sm\_list\_next()

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

**Routing table –** A data table stored in a router or a network host that lists the routes to particular network host that lists the routes to particular network destinations and in some cases metrics associated with those routes