**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** |
| 11/05/2023 | V0.1 | Anthony Scantsonihas | Beginning of Investigation |
| 12/05/2023 | V0.2 | Anthony Scantsonihas | Investigation |
| 13/05/2023 | V1.0 | Anthony Scantsonihas | Report Finalization |

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:  
***CID1520680***

# 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 |
| CID | Coverity Issue Identification Number |
| CWE | Common Weakness Enumeration |

# Code Review and Analysis

## Overview

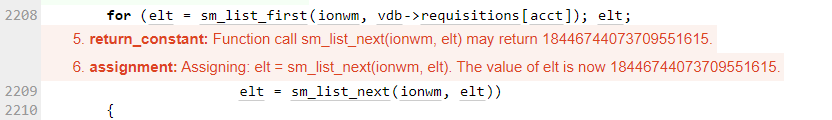
The Coverity report for the CID 1520680 has discovered errors within the code base ion.c:2211. The issue for this code block is an Out-of-bounds access which is described under CWE-119. This error indicates that the code is reading information and writing it from outside the intended boundary of the buffer.

## Observations

The Out-of-bounds access error can be traced to the function called “ionProvideZcoSpace”, which is responsible for retrieving the necessary information about requested space and available space from the SDR (Shared Data Repository) of the ION program.



The issue appears to arise from a loop variable called ‘elt’ as it is set to an initial value causing the loop to proceed along and continue even after ‘sm\_list\_next’ returns its maximum value. As the loop continues it causes a buffer overrun when sm\_list\_data is called.

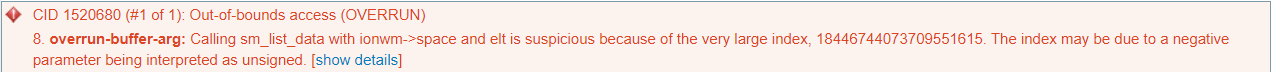


**-Potential Vulnerabilities**

There is a potential vulnerability with this block of code that could be exploited by potential adversaries. It is possible for the ‘elt’ variable to be exploited by an attacker to induce a buffer overflow error within the program, and therefore conducting a denial-of-service attack causing the application to crash.

## Supporting Evidence

The error flag for the block of code can be seen below. As can be observed within the screenshot the error is caused by the suspicious variable ‘elt’ due to it not being set to the correct value.



# Conclusions and Recommendations

My recommendation to resolve this issue would be to implement a method within the code that checks if the ‘elt’ variable is set to a safe value that won’t cause the buffer overflow error. If the value is safe the method can allow for the code to proceed as normal, otherwise if the value is capable of crashing the application it should be prevented from running the loop.

References

MITRE Corporation. (2023, January 31). CWE - CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer. Retrieved March 21, 2023, from <https://cwe.mitre.org/data/definitions/119.html>