Coverity Scan Static Analysis Report

Hardhard Enterprises

**T1** 2023

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** |
| 10/04/2023 | V0.1 | Callam Besley | 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 ***Memory corruptions*** type defect identified in the following CIDs:  
***1520842***

# 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

## Outcomes

This C function bslpol\_scparms\_size calculates the size of a list of bpsecctx\_parm structures in a memory partition. First it initializes the variable size to 2 bytes (since the first item in the list is a uint16\_t that represents the number of items in the list). It then iterates through the list through use of a ‘for’ loop with variable ‘elt’ to point to each element. While in the loop it checks if the data at the current element is not null and if it passes this check, it increments the size variable by the size of the id field, length field and the length if the data

## Observations

From scanning through the code, it is clear that the program has encountered an **out of bounds access** (Overrun). Sm\_list\_next (fig 1) is called with elt as a parameter which is a very large index (18446744073709551615) which may be due to a negative parameter being interpreted as unsigned. This is the maximum value representable by a 64 bit unsigned integer. Looking into the error, there is an attempt to access an element in the buffer partition->space at an index greater than the size of the buffer (fig 2). This is obviously due to the extremely large elt index which can be traced to the address argument (fig 3), the error reveals that the address argument is greater than the size of the buffer partition-> space meaning the resulting pointer points to an address outside of the buffer causing out of bounds access.

There are also a few logic issues present in the code (fig 4) such as the pointer parm being checked for a non-null value when it is assigned null on line 1108.

Sm\_list\_data is called with elt as a parameter which may access memory beyond the allocated list causing serious issues (fig 5)

## Supporting Evidence



Figure 2: index elt was attempting to access an element outside of the buffer partition->space

Figure 1: Sm\_list\_next is called with elt as a parameter returning the large integer

# A picture containing text Description automatically generated

Figure : the address arguement is pointing to an element outside of the buffer

# A picture containing timeline Description automatically generated

Figure 5: sm\_list\_data calls the high integer elt value which may result in catastrophic results such as data leakage.

Figure 4: Logic issue in code, palm is assigned null value then checked for null value later on

# Conclusions and Recommendations

This code can be exploited. The errors suggest that the program is trying to access memory it is not supposed to, which can lead to memory corruption or other significant issues. Out of bounds access occurs when a program attempts to access memory outside of its allocated memory block resulting in unpredictable behaviour in the program. Attackers can inject malicious code into the programs execution code using a buffer overflow (when the program writes code beyond the allocated buffer), or **Stack Smashing**, allowing the attacker to execute arbitrary code through overwriting the return address, or other stack variables. This malicious code can give an attacker access to the ION, or even crash the ION.

Some solutions include:

* Alter the address argument to be within the bounds of the buffer partition-> space
* Create a check to check the return value of sm\_list\_data before calling psp() to avoid a segmentation fault or undefined behavour as a result of NULL value or an invalid address. This check would ensure sm\_list\_data is not NULL and points to a valid memory location
* Create a check to see if the value of the elt parameter is less than the size of the partitions space before calling psp such as adding this check before line 656 in sm\_list\_next

if (elt >= partition->size) {

return 0;

}

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

***Out of bounds access –*** When the index used to address array items exceeds its specified value, it is attempting to access an area outside of the arrays bounds.

***Stack smashing -*** when a program overwrites the stack frame beyond its intended bounds