**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** |
| 2/05/2023 | V0.1 | Moe Khant Kyaw | Initial Document and analysis |
| 7/05/2023 | V0.2 | Moe Khant Kyaw | Finalizing Investigation and documentation |

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 ***High Impact Quality*** type defect identified in the following CIDs:  
***1520682***

# 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

CID 1520682 has been marked as high impact issue, Out-of-bounds access type under Memory-Corruptions category by Coverity. This issue is marked as CWE-117: Improper restriction of Operations within the Bounds of a Memory Buffer, also known as buffer overflow, which happens when the program is trying to read or write outside of the limit of intended buffer.

## 

## Observations

This issue is found in “bsles\_sdr\_serialize\_buffer” function under “/bpv7/library/ext/bpsec/bpsec\_policy\_eventset.c” file path. The above function serializes a ‘BpSecEventSet’ data structure into a provided buffer. The function initially sets ‘max\_events’, ‘elt’, and ‘eventAddr’ to 0. The number of events is then determined by the function in the ‘eventSetPtr->events’ using ‘sm\_list\_length’ function. Then, it enters a loop that iterates over each element in ‘eventSetPtr->events’ by calling ‘sm\_list\_first’ and ‘sm\_list\_next’ functions. The function then returns the number of bytes written into the buffer, Error if it returns -1.

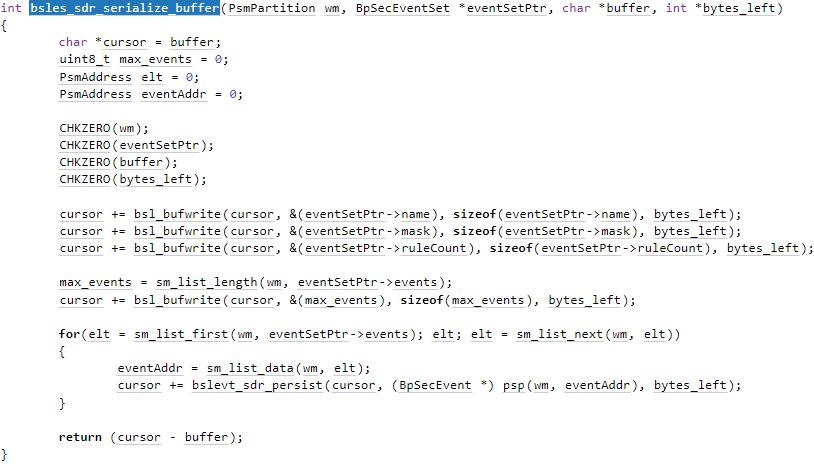


Fig 1: bsles\_sdr\_serialize\_buffer function

The error happens inside the for loop when going through the list, ‘eventSetPtr->events’. For each event, event address is retrieved using sm\_list\_data function which takes PsmPartition wm and PsmAddress elt as parameters. The loop begins by taking elt as the first element using ‘sm\_list\_first’ function and continuing until elt becomes invalid. The loop executes the next iteration by setting ‘elt’ to the next element using ‘sm\_list\_next’ function.

Graphical user interface, text

Description automatically generated

Fig 2: Out-of-bounds access error

Since elt variable has been assigned a value of ‘18446744073709551615’, (maximum value of a 64-bit unsigned integer) from sm\_list\_next(wm, elt), it is not a valid index for ‘sm\_list\_data’ function. Coverity is indicating that elt value is exceptionally large due to a negative parameter being interpreted as unsigned. The issue could cause the program to crash, generate incorrect results or expose to security risks. That is why it has been flagged as CWE-119 by Coverity.

# 

# Conclusions and Recommendations

This issue can expose to program failure, memory corruptions and other undefined behaviours.

To prevent this issue from happening, we can do the followings:

* A validation of elt variable can be made inside the for loop before executing ‘*sm\_list\_data**’,*
* Make sure let variable is within the acceptable index for ‘*sm\_list\_data**’,*
* Check variable elt after executing ‘*sm\_list\_data’* and terminates the for loop if it's out of bound.
* Alternatively, a function to check elt value can be included inside ‘*sm\_list\_data’* function to prevent this issue as the function only has null checker for elt.

Graphical user interface, text, application, email

Description automatically generated

Fig 3: sm\_list\_data function

References

*Common weakness enumeration* (no date) *CWE*. Available at: https://cwe.mitre.org/data/definitions/119.html (Accessed: May 4, 2023).

Appendix

Include additional information/documentation here to help the readers understand complex information.