Coverity Scan Static Analysis Report

Hardhard Enterprises

**T3** 2022

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** |
| 27/03/2023 | V1 | Jesse Ludeman | Initial document and investigation |
| 30/03/2023 | V1.2 | Jesse Ludeman | Investigation continues |
| 1/04/2023 | V1.3 | Jesse Ludeman | Finalize 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: 1520736

# 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

## Introduction

CID 1520736 has been flagged by Coverity as a High impact issue of type Out-of-bounds access. This type of issue generally means that the program can read or write to a memory location that is outside of the intended boundary of the given buffer.

## Observations

This issue occurs in the bsles\_get\_event function in the [/bpv7/library/ext/bpsec/bpsec\_policy\_eventset.c] file. This function is designed to retrieve an event object associated with an eventset. It accepts three arguments: PsmPartition, BpSecEventSet, and BpSecEventId.

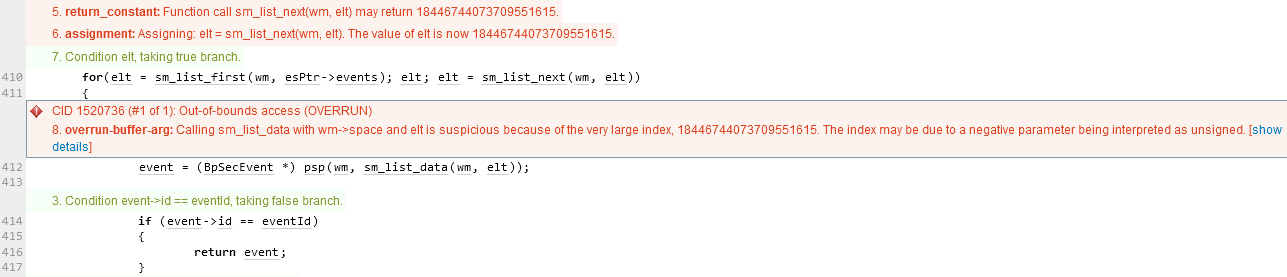


Figure 1 - Initial error in For loop

The initial error occurs inside the For loop on line 410, which iterates the sm\_list\_next variable. It then sets the event variable during each iteration. Furthermore, if the event->id matches the eventId, then it returns the event.

Additionally, we notice that in the for loop a call is made to sm\_list\_data, and passing in the elt variable causes another error to be flagged. Inspecting the elt variable shows that it’s initialized to 0, as demonstrated in figure 2.

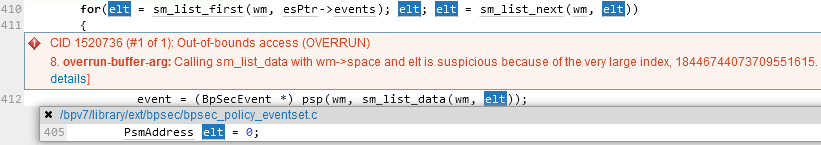


Figure 2 - elt is initialized to 0

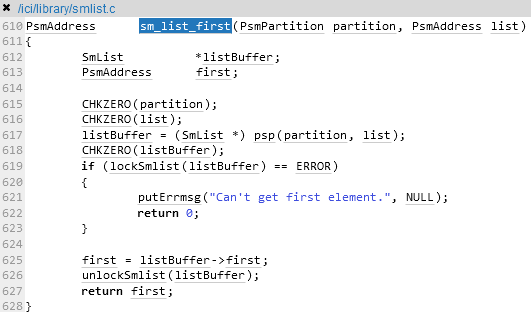


Figure 3 - sm\_list\_first function

Inspecting the sm\_list\_first() function in figure 3 indicates that there’s currently no validation on the return value for this function. According to the documentation this function indicates that it will return the PsmAdress of the first/last element in list, or 0 on error [1].wm is supposed to return a PSM address and elt is meant to return the number of nodes. Assuming that the linked-list is empty, the return value will be a null pointer. It’s possible that this return value is considered a negative number.

Alternatively, if the sm\_list\_first() function is being used incorrectly, it could return an unexpected value. For example, if the list is empty or if the function is being called on an invalid list, it might return a large number as an error code.

## Supporting Evidence

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

# Conclusions and Recommendations

There is no vulnerability associated with this CID. However, Hardhat Enterprises believes it’s a valid bug. To fix this, an update would need to be made to the [/ici/library/smlist.c] library which we ultimately have no control over.

References  
[1] Ubuntu Manpage: smlist – shared memory list management library (2019)

Appendix

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