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 formatted 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** |
| 11/12/2022 | 0.1 | Connie Cox | 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 ***Out-of-bounds access*** type defect identified in the following CIDs:  
**1520885**

# Acronyms and Abbreviations

Please keep an updated list of acronyms and abbreviations used throughout the report.

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| CID | Coverity Issue Identification Number |
| DTN | Delay/Disruption Tolerant Network |
| ION | Interplanetary Overlay Network |
| PSM | Personal Space Management – memory management within a pre-allocated memory partition |
| SDR | Spacecraft Data Recorder: persistent object database in shared memory using PSM and SmList |
| SmList | Linked lists in shared memory using PSM |
| ZCO | Zero-Copy Objects capability – minimize data copying up and down the stack |

# Code Review and Analysis

## Introduction

Please provide an in-depth report on the analysis performed here.

The Coverity report has identified numerous Memory Corruption issues within the libbpP.c code base within the Bundle Protocol v7 Library [/bpv7/library]. The following analysis outlines the Out-of-bounds access issues identified in the bpStart function.

Both functions use doubly linked lists as a buffer to hold the address of the element at the index address of the buffer. In bpStart(), the buffer is partition->space. The buffer is accessed using double linked list functions sm\_list\_first and sm\_list\_next.

## Observations

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

This bpStart function is the main function that is invoked to start the bundle protocol. The first offending code in this function is where the forwarders and admin endpoints for all schemes are started.

1. /\* Start forwarders and admin endpoints for all schemes. \*/

2.

3. for (elt = sm\_list\_first(bpwm, bpvdb->schemes); elt;

4. elt = sm\_list\_next(bpwm, elt))

5. {

6. startScheme((VScheme \*) psp(bpwm, sm\_list\_data(bpwm, elt)));

7. }

8.

In the above code, there is no validation on the element index returned in sm\_list\_next(). In the event that a negative parameter is treated as unsigned is assigned to elt as a returned value from sm\_list\_first(bpwm, bpvdb->schemes), elt will be an extremely large value. Accordingly, there is a possibility that the element index far exceeds the number of elements in the buffer. When this occurs, the buffer is overrun.

The above issue occurs in all instances in the bpStart() where the sm\_list\_next() is used to access the buffer. Examples include where the convergence-layer managers for all egress plans are started, and where all ducts for all convergence-layer adapters are started. These are highlighted below.

1. /\* Start convergence-layer managers for all egress plans. \*/

2.

3. for (elt = sm\_list\_first(bpwm, bpvdb->plans); elt;

4. elt = sm\_list\_next(bpwm, elt)

)

5. {

6. startPlan((VPlan \*) psp(bpwm, sm\_list\_data(bpwm, elt)));

7. }

8.

9. /\* Start all ducts for all convergence-layer adapters. \*/

10.

11. for (elt = sm\_list\_first(bpwm, bpvdb->inducts); elt;

12. elt = sm\_list\_next(bpwm, elt))

13. {

14. startInduct((VInduct \*) psp(bpwm, sm\_list\_data(bpwm, elt)));

15. }

16.

17. for (elt = sm\_list\_first(bpwm, bpvdb->outducts); elt;

18. elt = sm\_list\_next(bpwm, elt)

)

19. {

20. startOutduct((VOutduct \*) psp(bpwm, sm\_list\_data(bpwm, elt)));

21. }

22.

23.

### What is an out-of-bounds access defect?

An out-of-bounds access error occurs when the buffer accesses memory and/or stores data beyond its original memory allocation. In this context, the index used in the buffer storing PsmAddresses for span elements could potentially be larger than the max number of element indexes that was allocated to the buffer holding all the element addresses. This has the potential to:

* Cause adjacent storage to store overflowed data
* Crash the node(s)
* Create an entry point for a cyber exploit

## Supporting Evidence

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

# Conclusions and Recommendations

Please provide any conclusions and recommendations here. Include references to supporting evidence and any other required information in the appendix.

To ensure that buffers are not over run, there needs to be bounds checking on the index being used. This can be done by verifying the parameter used. That is, if the parameter is negative, it should not be treated as unsigned. Secondly, validation of the element index returned. The element index should never be larger than the number of bundle elements in the configuration. To streamline this process, a function could be created to perform the element index value validation.

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

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