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** |
| 30/04/2023 | V1 | Jesse Ludeman | Initial document and investigation |
| 1/05/2023 | V1.2 | Jesse Ludeman | Finalize investigation and submit SAR report |

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: 1520727

# 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 1520727 has been flagged by Coverity as a High impact issue of type Out-of-bounds access. This type of defect typically means that the program is trying to read or write to a memory location that is outside of the boundary of the intended buffer.

## Observations

### Introduction

This defect has been flagged in the /bpv7/utils/bpadmin.c file, and exists in the listEndpoints function. This function is part of the Bundle Protocol (BP) administration utility, and is used to display the endpoints currently configured in the BP agent.

When listEndpoints is called, it retrieves a list of all the endpoints currently registered with the BP agent and prints them to the console. The endpoint is a communication channel that can be used to send and receive bundles in the BP angent, and is identified by a combination of an IP address and a service number.

This function is mostly useful for debugging and monitoring the BP agent since it provides a way to see which endpoints are currently active and which ones might be causing issues or errors in the system.

The first occurrence of the error is inside a for loop, then inside the second switch statement on line 1091, which has been demonstrated in Figure 1. We first notice that a call to sm\_list\_data() is being made, which returns the PsmAddress that is the data value for elt, or returns 0 on error. We note that the elt variable has been assigned 18446744073709551615 from the above for loop on line 1089 in a call to sm\_list\_first(). This function returns the first element in the list, which in this case is PsmAddress schemes.

Due to this, when a call to sm\_list\_data() is made, the elt variable is passed in as the second argument to the function. This is a problem given that the value for elt is now 18446744073709551615, which is not a valid PsmAddress for the function.

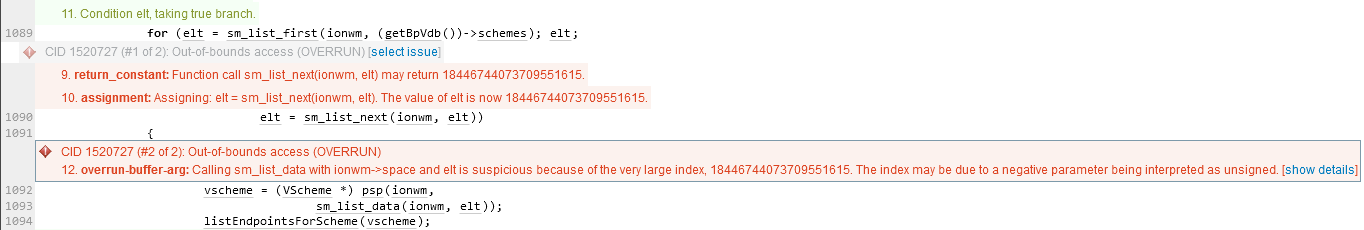


Figure 1 - Out-of-bounds access error

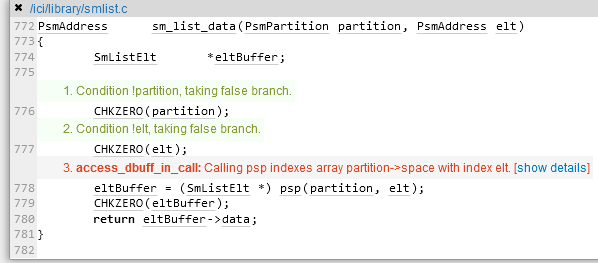
Furthermore, inspecting the sm\_list\_data() function shows that partition->space is being indexed with the elt variable.

Figure - sm\_list\_data() function

### The defect

The defect has been flagged because the elt variable has been assigned a very large index [18446744073709551615]. If we assume that this is a valid value, then this should not cause the program to crash, or introduce a security risk to the program. However, if we assume that it is indeed an incorrect value, then the program could crash, or potentially cause a risk to the security posture of the program.

## Supporting Evidence

# Conclusions and Recommendations

Hardhat Enterprises made the following discoveries, and suggest the following changes be implemented:

* Add a validation check to the sm\_list\_data() function that ensures only a valid PsmAddress value is returned. This will prevent it from returning a very large result such as 18446744073709551615.
* No security risk or vulnerability exists with this CID

References

Appendix