**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** |
| 16/05/2023 | V1.0 | Druween Perera | Initial Document |
|  |  |  |  |
|  |  |  |  |

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:  
***1520786***

# 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

Coverity has identified **CID 1520786** as a **High impact** problem of type **Out-of-bounds access**. This sort of flaw often shows that the product is performing operations on a memory buffer, yet it has the capacity of accessing or modifying a memory location that extends outside the buffer's intended limitations. This has been marked in the **ltpStart()** function. This function is part of the Licklider Transmission Protocol, a point-to-point protocol designed for usage in deep space networks.

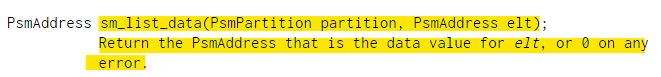
## Observations

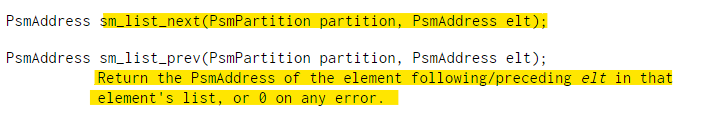
On futher investigation into the above mentioned error, it was observed that the high impact out-of-bounds access problem occurs across 4 different location, which together as a whole related to the ltpStart() function and generates CID 1520786.

The first of four errors initially occurs within line 1180 (figure 1) which advices that calling the **sm\_list\_data** function may return a large value of 18446744073709551615, this is due to the term “elt” being called as a parameter. Furthermore, This occurs because inside the for loop, the elt loop variable is being assigned the value from the sm\_list\_first() function. Additionally, inside the body of the for loop, we notice that the elt variable is then reassigned the value from the sm\_list\_next() function during each iteration.

Hence, similar instances of the same above mentioned error can be observed within the remaining 3 CID codes that are placed across the ltpStart() function.(figures 2, 3, and 4).

## 





The second parameter of the sm\_list\_next(), sm\_list\_first(), and sm\_list\_data() functions is a PsmAddress, according to the API description. (smlist - shared memory list management library, 2014) Coverity flags this as an error owing to the big index since the elt variable has been assigned the value 18446744073709551615 which can be seen in the above highlighted screenshots.

## Supporting Evidence

# 

Figure 1

# 

Figure 2

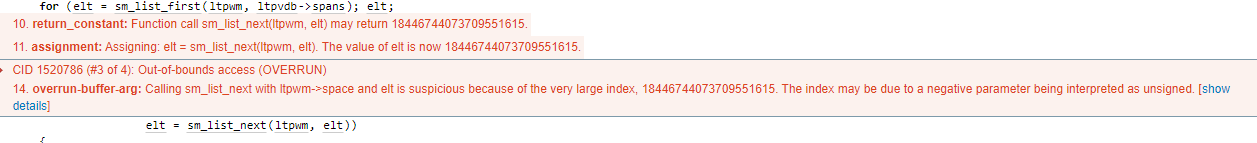


Figure 3

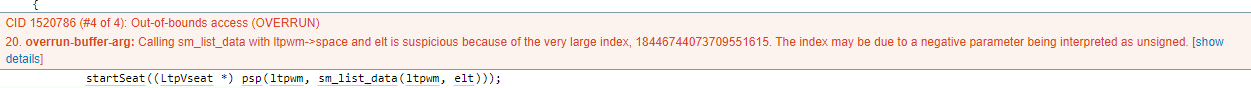


Figure 4

# Conclusions and Recommendations

While no security vulnerability has been detected for this flaw, it is crucial to note that out-of-bounds access mistakes can be exploited. For this sort of vulnerability, buffer overflow and denial-of-service attacks are both viable. However, given the present codebase's context, we are unable to detect any significant hazards.

# References

*smlist - shared memory list management library*. (2014, 07 08). Retrieved from Debian manpage: https://manpages.debian.org/jessie/ion-doc/smlist.3.en.html

Appendix

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