**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** |
| 20/05/23 | V1.0 | Damon Willmott | Write up |
| 21/05/23 | V1.1 | Damon Willmott | Completed |

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:  
***CID-1520619***

# Acronyms and Abbreviations

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

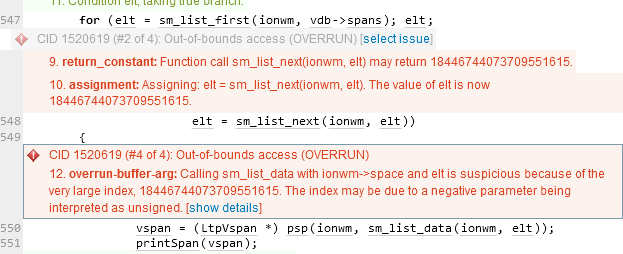
|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| DTN | Delay/Disruption Tolerant Network |
| ION | Interplanetary Overlay Network |
| LTP | Linux Test Project suite |

# Code Review and Analysis

## Overview

Coverity identifies CID-1520619 as a **High Impact** quality vulnerability, and is labelled as an **Out of Bounds** access error, also known as a buffer overflow error. This implies the that the code is trying to access memory outside of an allocated buffer. This usually occurs when the data is somehow given the ability to read and write data from a memory location outside of its intended boundaries/buffer.

## Observations

The issue can be found in the listSpans function in the /ltp/utils/ltpadmin.c file directory of the Linux test Project suite (LTP). Within this function, a value is being passed without being properly validated, and in this case the value being passed (18446744073709551615) is larger than the specifications of the buffer and is too large to be handled by an element index, which is causing the overflow of the allocated memory.  
  
The issue begins at line 547, which indicates that the function **sm\_list\_next** will return a value of 18446744073709551615. This is a result of the following for loop, which assigns the variable elt the value from the sm\_list\_first function. Within this loop, the elt variable is also reassigned the value from the sm\_list\_next function with each iteration of the loop. Furthermore, because the sm\_list\_next function contains a PsmAddress parameter, and the value of elt is quite high, Coverity will identify this as an error as the value is too high.

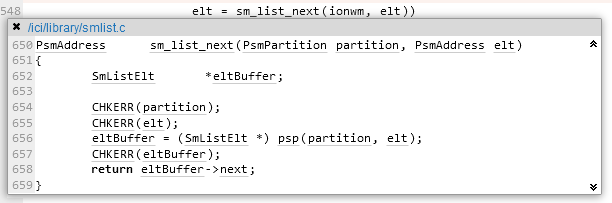
Fig. 1. Coverity Static analysis results

Fig. 2. Sm\_list\_next function

## Supporting Evidence

# Conclusions and Recommendations

Whilst this particular defect does not produce any security vulnerabilities, in general out-of-bounds access erros do have the potential to be exploited by attackers, for example they can bring rise to program failure, memory corruptions and denial-of-service attacks.

Whilst the sm\_list\_next function is working properly, it is recommened that it is updated to include validation on the elt variable. For example, the function could include a check on elt by including a maximum value for PsmAddress, to see whether the value of the PsmAddress is too high, and if it is, prevent it from being used and produce an error instead of the value continuing to be used and causing complications.

See also SAR-1520702 by Damon Willmott

References  
Common Weakness Enumeration (n.d.) *CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer*, <https://cwe.mitre.org/data/definitions/119.html>

Appendix

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