**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 | V0.1 | Callam | 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 ***{Issue Category}*** type defect identified in the following CIDs:  
***{Coverity Issue CID}***

# 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

This code is responsible for parsing topics and payloads and is a part of a larger system related to handling data communication protocols. Some aspects of the code include ‘parsetopic’ taking various parameters, including a buffer that contains data to be passed. It also checks if there are enough unparsed bytes in the buffer to proceed with parsing. Using ‘decodeSdnv’ the code extracts the topic ID and the number of records from the buffer. If the selected topic is not flagged for skipping, memory is allocated to the payload and payload data is written to allocated memory. Payload records are parsed until the desired number is reached/no more data to parse

## Observations

On lines 2476 - 2490 (fig 1) bytesUnparsed is incremented by bytesRecieved after receiving data into the buffer, although it fails to check if the value of this increment exceeds the buffer size which may lead to a **buffer overflow**

On line 2508 (fig 2) the loop iterates over vsap list but fails to include a termination condition, instead relying on loop condition ‘elt’ to evaluate to ‘null’ when there are no more elements. When elt is assigned an extremely large integer the code encounters an issue.

On line 2512 (fig 3) if topicID is located in the vsap the code breaks out of the loop, although there is no error handling if topicID is not matched.

More specifically, when examining the error, (fig 4), sm\_list\_data is attempting to access the data associated with the given ‘elt’ in the list. However loop termination is not handled correctly causing the loop to continue beyond the last element in the list. When the loop reaches the last invalid element, an out-of-bounds error occurs.

Furthermore, there exist errors on numerous lines where ‘memmove’ function is called to refill the buffer. This ‘memove’ function copies data from one memory location to another. However, buffer size (‘buflen’) is used as the length argument for ‘memmove’ is fixed at ‘BUFMAXSIZE’ assumed to be 65536 bytes (default value). This fixed buffer size may cause a buffer overflow if ‘bytesRecieved’ is larger than the remaining space in the buffer.

## A picture containing text, screenshot, font, number Description automatically generatedSupporting Evidence

Figure : No check if bytesunparsed incremented by bytesrecieved exceeds buffer size

# A screenshot of a computer program Description automatically generated with low confidence

Figure : No termination condition for vsap list

# A picture containing text, font, screenshot, line Description automatically generated

Figure : no error handling if topicID is not matched

# A screenshot of a computer program Description automatically generated with low confidence

Figure : loop termination is not handled correctly causes out of bounds error

# Conclusions and Recommendations

Code should be reviewed to fix loop termination conditions to ensure loop ends at correct point.

‘bytesRecieved’ should be used as the length argument for ‘memmove’ to prevent buffer overflow. Buffer size should be properly defined and parsed as an argument to ‘parseTopic’ function.

Since a buffer overflow error has occurred, the code is exploitable.

Out of bounds access allows an attacker to modify/access memory outside of the buffer/array which can lead to buffer overflows that can execute malicious code or crash the program. Can also include memory leaks that reveal sensitive information such as passwords, cryptographic keys or other sensitive data. Through exploiting, an attacker can modify the return address on the stack to redirect program execution to their malicious code leading to more serious issues such as crashing the ION/returning sensitive data.

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

**Buffer overflow:** When a program writes data beyond the bounds of an allocated buffer