**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** |
| 6/04/23 | V0.1 | Damon Willmott | Write-up |
| 8/04/23 |  | Damon Willmott | Write-up |
| 22/04/23 |  | Damon Willmott | Write-up |
| 23/04/23 |  | Damon Willmott | Write-up |
| 7/05/23 |  | Damon Willmott | Completion |

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-1520702***

# Acronyms and Abbreviations

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

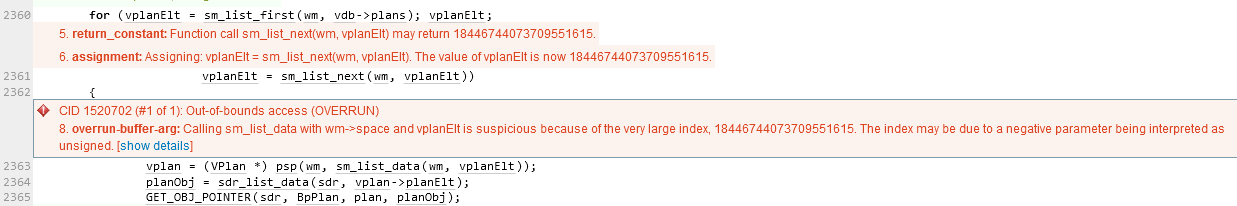
|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| DTN | Delay/Disruption Tolerant Network |
| ION | Interplanetary Overlay Network |

# Code Review and Analysis

## Overview

Coverity identifies CID-1520702 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 rescanPlans function in the /bpv7/tcp/tcpcli.c file directory of Bundle Protocol 7. Whilst there is little to no documentation indicating what this function does, it can be deduced it might play a role in scanning or reevaluating various plans or configurations within the TCP module client of the NASA ION software. 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 stems from the variable vplanElt being passed the large value of “18446744073709551615”, when it calls upon the function sm\_list\_next. Because the size of this element index is too large, it causes an overflow of the allocated memory beyond the buffer. This results from a lack of validation when the value is passed, which causes the passing of negative parameters which the code will interpret as unassigned. The sm\_list\_next function contains does contain some validation, through the use of two parameters, partition and let, which are verified by using the “CHKERR” macro, however this function results in the large of “18446744073709551615” being passed to let, and thus vplanELT, suggesting that the value of let exceeds the bounds outlined of the linked list, leading to out of bounds access.

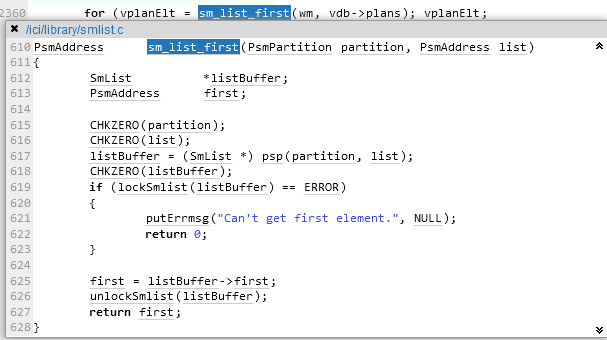
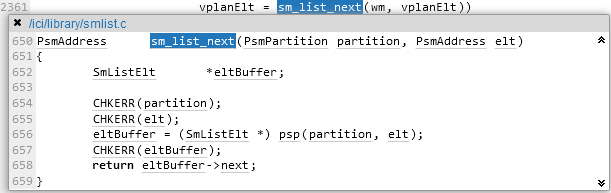
Fig. 1. Covereity Static Analysis results

Fig. 2. sm\_list\_first function

Fig. 3. Sm\_list\_next function

## Supporting Evidence

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

# Conclusions and Recommendations

Out-of-bounds errors can potentially cause program failure and memory corruptions, and if the issue it not fixed promptly, this could allow a potential attacker to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash. As a result, this error poses a risk to the software and can be considered a priority vulnerability that will need to be fixed (CWE – CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer). Unlike other languages that contain memory management systems, such as Python, the C language lacks the ability to automatically validate the locations of memory buffers. As a result, functions that ensure that the values that are being passed to vplanElt are validated should be implemented.  
  
The following could be implemented to rectify this issue:

* Implement validation for the value of vplanElt to ensure that the value produced for is within the buffer limits. This will ensure that large values such as 18446744073709551615 are not produced.
* Insert a function within vplanElt to check the value of vplanElt to ensure the value is within the limits of the buffer beforehand
* Utilize other programming languages that offer in-built memory management systems

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>

VeraCode (n.d.) *What Is a Buffer Overflow? Learn About Buffer Overrun Vulnerabilities, Exploits & Attacks*, VeraCode, <https://www.veracode.com/security/buffer-overflow>

Appendix

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