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** |
| 18/03/2023 | V1 | Jesse Ludeman | Initial document and investigation |
| 26/03/2023 | V1.1 | Jesse Ludeman | Continue analysis and investigation |
| 27/03/2023 | V1.2 | Jesse Ludeman | Finalize investigation and clean up 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 ***Memory – illegal accesses*** type defect identified in the following CIDs: ***1520638***

# 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

Coverity has identified this CID as an “Uninitialized pointer read” defect. This type of defect generally means that a variable has been initialized without a value. This can lead to unpredictable results or can cause an incorrect computation. This report details what this type of defect is, how it happens, whether it’s a security vulnerability, and how to remediate it.

## Observations

This issue originates in the addService function on line 665, which accepts two arguments: tokenCount and tokens. Additionally, on line 775 Coverity has flagged the dummyChild variable as not being initialized correctly.

A variable that has been declared in the C language without an initializer, depending on where it’s declared, can either be initialized to zero, or is best described as “indeterminate”. Trying to read a variable prior to it being assigned a value can result in undefined behaviour, which is undesirable for a C program. For this issue, the latter is the cause as the variable has been declared inside the scope of the addService function.

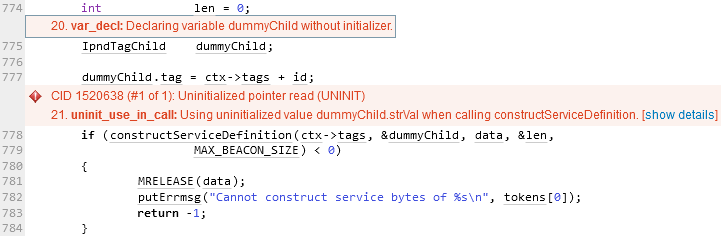


Figure 1 - dummyChild variable declared without initializer

Furthermore, there is an if statement that calls the constructServiceDefinition function, that Coverity has flagged as attempting to use the uninitialized dummyChild variable inside this conditional check. This function is designed to build the service definition inside ION and accepts two arguments: IpndTag and IpndTagChild. Inside this function is a switch statement that will execute if a condition is met based on the \*child variable of the latter argument that’s passed into this function. To be specific, this is checking if the child->tag->number is less than 64, and if it’s True, then the switch statement executes.

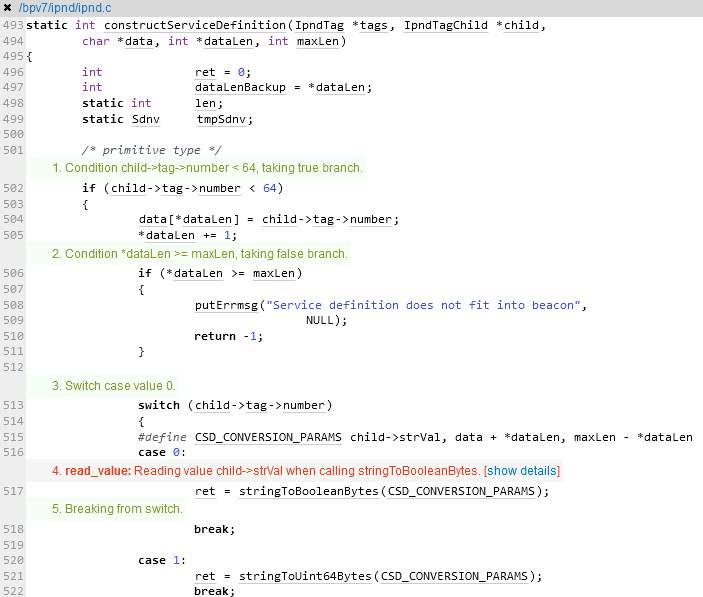


Figure 2 - constructServiceDefinition

The beginning of the switch statement starts on line 513 which executes based on previous if statement on line 502. For the purposes of this analysis, we assume that the if statement does in fact execute, as Coverity has already flagged this as noted in figure 1. We observe that on line 516 Coverity has a problem trying to read the value of child->strVal when calling the stringToBooleanBytes function.

## Supporting Evidence

This issue has been flagged by Coverity because dummyChild has been initialized without a value upon declaration and has then been passed to the constructServiceDefinition function, at which point it tries to read the value prior to calling the stringToBooleanBytes function.

# Conclusions and Recommendations

Based on this investigation, this issue appears to be a false-negative, and there is no immediate vulnerability associated with this finding. To fix this bug and remediate similar issues, ensure that variables are initialized with values when they are declared.

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

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