**T2** 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** |
| 16/08/2023 | V1.0 | Adharshaan Devaraj | Initial Document |
|  |  |  |  |
|  |  |  |  |

Table of Content

Contents

[Introduction 3](#_Toc145894876)

[Objective 3](#_Toc145894877)

[Scope 3](#_Toc145894878)

[Acronyms and Abbreviations 3](#_Toc145894879)

[Code Review and Analysis 4](#_Toc145894880)

[Overview 4](#_Toc145894881)

[Observations 4](#_Toc145894882)

[Supporting Evidence 4](#_Toc145894883)

[Conclusions and Recommendations 4](#_Toc145894884)

[References 5](#_Toc145894885)

[Appendix 5](#_Toc145894886)

# 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 ***Dereference before null check*** type defect identified in the following CIDs:  
***1520869.***

# 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 |
| CWE | Common Weakness Enumeration |
| BIB | Bundle Integrity Block |

# Code Review and Analysis

## Overview

By analyzing “**bpsec\_util.c”** code base within the ‘**library’** folder of the Bundle Protocol v7 directory. Coverity highlighted a “***Dereference before null check*”** error that highlights the unnecessary comparison against null values hence a null pointer dereference is detected. (CWE-476, 2023)

## Observations

The "**bpsec\_util.c**" is a source code file that contains utility functions and functions related to the implementation of BPsec (Bundle Protocol Security) within ION such as implementing access control mechanisms to ensure that only authorized entities can access and manipulate bundles or keys.. The function ‘**bpsec\_getOutboundSecuritySource’** retrieves the EID of the security source for an outbound abstract security block.

## Supporting Evidence

A screenshot of a computer

Description automatically generated

Figure 1- CID 1520869

The above screenshot represents the error that was identified in the code within the “**bpsec\_util.c” file.** The reason for this error is because the code is attempting to access an object without confirming if it is null or not. In line 1333 we can see that ‘**fromEid’** variable is being assigned as NULL, hence dereferencing.

# Conclusions and Recommendations

The error indicates that the code is attempting to dereference a pointer or access a member of an object without first determining if it is null. If the pointer is null, this might result in undefined behavior or crashes. To correct this mistake, reverse the sequence of the null check and dereference operations.

To fix the code use a validation like this:

if (fromEid != NULL) {

// Dereference fromEid here

// This code block will only execute if fromEid is not null

} else {

// Handle the case where fromEid is null

}

# References

CWE-476, 2023. *CWE-476: NULL Pointer Dereference.* [Online]   
Available at: https://cwe.mitre.org/data/definitions/476.html  
[Accessed 2023].

Appendix

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