Coverity Scan Static Analysis Report

Hardhard Enterprises

**T3** 2022

**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** |
| 07/04/2023 | V0.1 | John-Eddie Cubis | Investigation |
| 20/04/2023 | V0.2 | John-Eddie Cubis | Finalising Report |
|  |  |  |  |

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 **Error! Bookmark not defined.**](#_Toc119848729)

[Observations 4](#_Toc119848730)

[Supporting Evidence 5](#_Toc119848731)

[Conclusions and Recommendations 5](#_Toc119848732)

[References 6](#_Toc119848733)

[Appendix 6](#_Toc119848734)

# Introduction

The Coverity report has identified an out-of-bounds access error within CID 1520659, the following outlines the issues identified in the function bpsec\_instr\_get\_srcnames.

## 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:  
1520659

# 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 |
| ION | Interplanatary Overlay Network |
| PSM | PersonalSpace Management – Memory Management inside pre allocated partition |
| SDR | Spacecraft Data Recorder: Persistent object database in shared memory using PSM and SmList |
| SmList | Linked list in shared mermory |
| ZCO | Zero-Copy Objects capability – minimize data copying up and down the stack |

# Code Review and Analysis

## Observations

This bpsec\_instr\_get\_srcnames function, is used to retrieve the sources known to BPSEC as a comma-separated string.

* The function will calculate the size of the list by the total size of the source, plus 1 character per key for a comma to separate values, plus a null terminator.
  + ***This function in itself is expensive on system memory and is inefficient.***

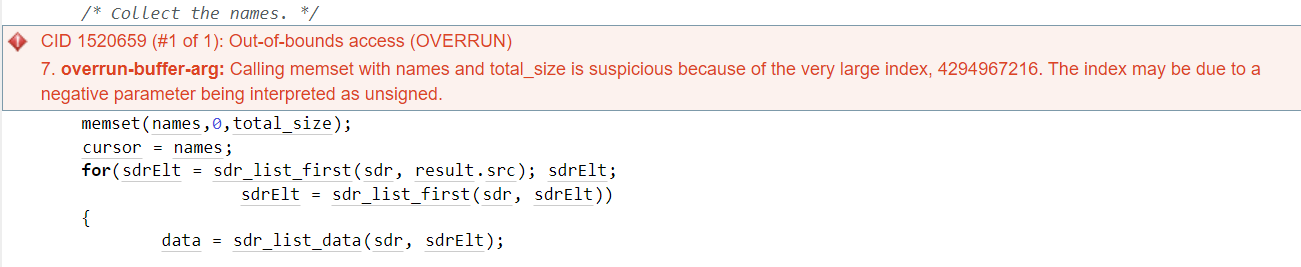
*Text

Description automatically generated*

* If the source check is unknown the function, the function will catch the error and return.

Graphical user interface, text, application

Description automatically generated

* The function collects the names in the function however comes to an overrun error.
* The calculation to find the ‘total\_size’, is the factor that produces this error.
* When the value is initialised to 0 in the linked list, this makes the list invalid which unintentionally assigns the list to a large number which the picture above shows this error.
* Consequences of an overrun error include Integrity Confidentiality and Availability
  + Buffer overflows can be used to execute arbitrary code, that is outside the scope of a program’s implicit security policy.
  + Buffer overflows also lead to crashes leading to lack of availability and putting the program into an infinite loop.
* This error has a possibility of an exploit given the attacker knows the source code and the problems that arise when calculations of this function activate, this must be fixed immediately.

## 

## Supporting Evidence

We can refer to ‘Report 1520883’ from Connie Cox, that also comes across a similar error that we can refer to.

# Conclusions and Recommendations

Possible mitigations include:

Requirements:

* Use language that does not allow weakness or provides constructs to avoid issues.
* Code is still subject to overflows even if the language is theoretically safe.

Architecture and Design:

* Use vetted library or framework.

When elements are used to access linked lists, we must validate values within the list and that they exist. This is to ensure the list is not risk of being overrun.

References

MITRE Corporation. (2023, January 31). CWE - CWE-805: Buffer Access with Incorrect Length Value. Retrieved March 21, 2023, from <https://cwe.mitre.org/data/definitions/805.html>

MITRE Corporation. (2023, January 31). CWE - CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer. Retrieved March 21, 2023, from <https://cwe.mitre.org/data/definitions/119.html>

Appendix