**T3** 2022

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** |
| 21/03/2023 | V0.1 | John-Eddie Cubis | 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 ***Out-of-bounds access*** type defect identified in the following CIDs:  
1520817

# 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 | Interplanetary Overlay Network |
| PSM | Personal Space Management – Memory Management inside pre allocated partition |
| SDR | Spacecraft Data Recorder: Persistent object database in shared memory using PSM (Process Safety Management) and SmList |
| SmList | Linked list in shared memory |
| ZCO | Zero-Copy Objects capability – minimize data copying up and down the stack |

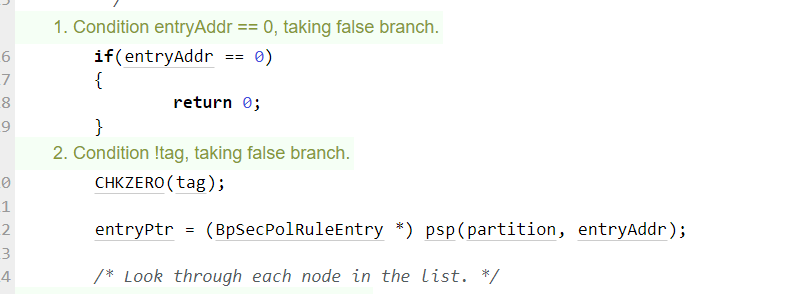
# Code Review and Analysis

# Introduction

Coverity has identified this CID as an Out of bounds access error defect. Also known as a buffer overflow, this means data stored in this buffer is out of the range of its allocated memory. Buffer Overflows have the possibility to read or write outside its intended boundary.

## Observations

The return\_constant function, ‘entryAddr’ returns the value that is out of the boundaries set in the code.



The code adds a check if the value is equal to zero, and if it is not 0, it will return a large value, forcing it to loop. This is a potential exploit if attackers know of this code bug.

This will make the for-loop overload, creating an out of bounds access defect.

A picture containing text

Description automatically generated

* **It is possible for attackers to execute code between the check and within the overload.**
* The attack can script an address of a large integer and overload the buffer.

Out of bounds errors occur, when memory stores that exist in buffer are beyond its memory allocation and has the potential to store overflowed data, crash the node and create an entry point for attackers.

Consequences also include:

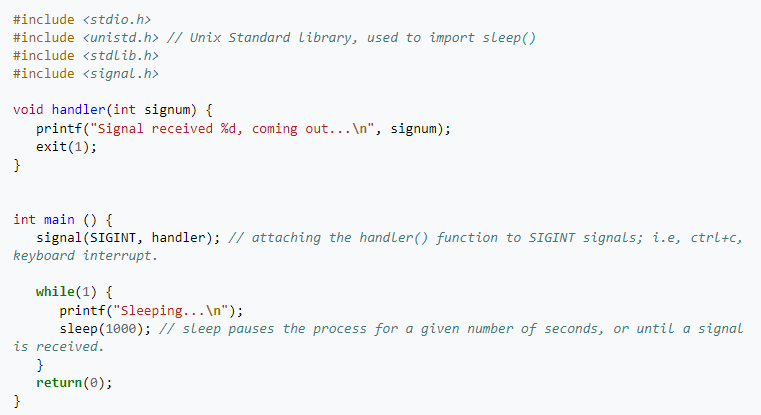
Integrity, Confidentiality, Availability

* Buffer overflows can be used to execute arbitrary code 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.

**However, proper error checking will fix this code bug and allow for values to not be executed** **such as:**

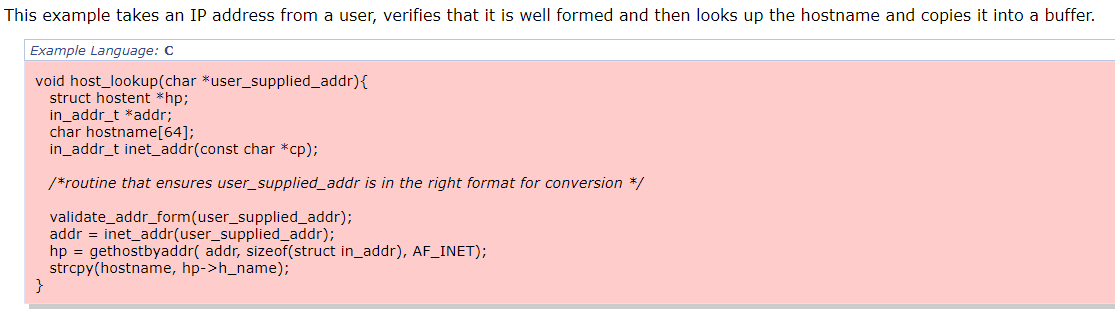
Sourced from [C Programming/Error Handling](https://en.wikibooks.org/wiki/C_Programming/Error_handling):

One solution may be Signals, Signals can be used to raise events in the host environment or OS, to indicate a specific error or critical event. Signals are not used as a means of implicit error catching, but indicate interference of normal program flow, which network administrators can act upon and monitor during the uptime of the network. Signals can be used to validate data that is within the scope of the program and as well as be used to alert system administrators of potential issues.

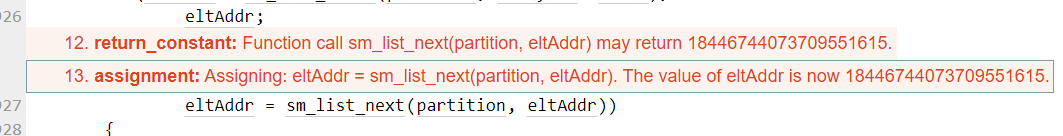


## Supporting Evidence

This error is a bug, however, can lead to an exploit, and must be fixed at once. For example this code sample:



The code snippet above, shows the attacker generates a value resolve to a large integer, the function has the possibility of overwriting sensitive data or even relinquish control flow to the attacker, similar to the error in Coverity where the index returned is a large value.



# 

# Conclusions and Recommendations

Mitigations include:

Requirements:

* Use language that does not allow weakness or supplies constructs to avoid issues
  + Validation of the index
  + Bound Checks
* Code is still subject to overflows even if the language is theoretically safe

Architecture and Design:

* Use vetted library or frameworks

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