



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** |
| --- | --- | --- | --- |
|  | V0.1 | Kanad Dombhare |  |
|  |  |  |  |
|  |  |  |  |

Table of Content

Contents

[Introduction 3](#_gjdgxs)

[Objective 3](#_30j0zll)

[Scope 3](#_1fob9te)

[Acronyms and Abbreviations 3](#_3znysh7)

[Code Review and Analysis 4](#_2et92p0)

[Outcomes 4](#_tyjcwt)

[Observations 4](#_3dy6vkm)

[Supporting Evidence 4](#_1t3h5sf)

[Conclusions and Recommendations 4](#_4d34og8)

[References 5](#_2s8eyo1)

[Appendix 6](#_17dp8vu)

# 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 ***“ Expression Issues”***type defect identified in the following CIDs:  
***1520716***

# 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

The code excerpt supplied is a memory allocation function that allocates memory for a data block. Depending on whether RTEMS is defined, the code provides platform-specific conditional compilation for memory allocation using “ malloc” or “memalign”. The function's purpose is to allocate memory that is aligned on a boundary. There have been several observations about potential problems with the “memalign” call and the use of sizeof(void \*).

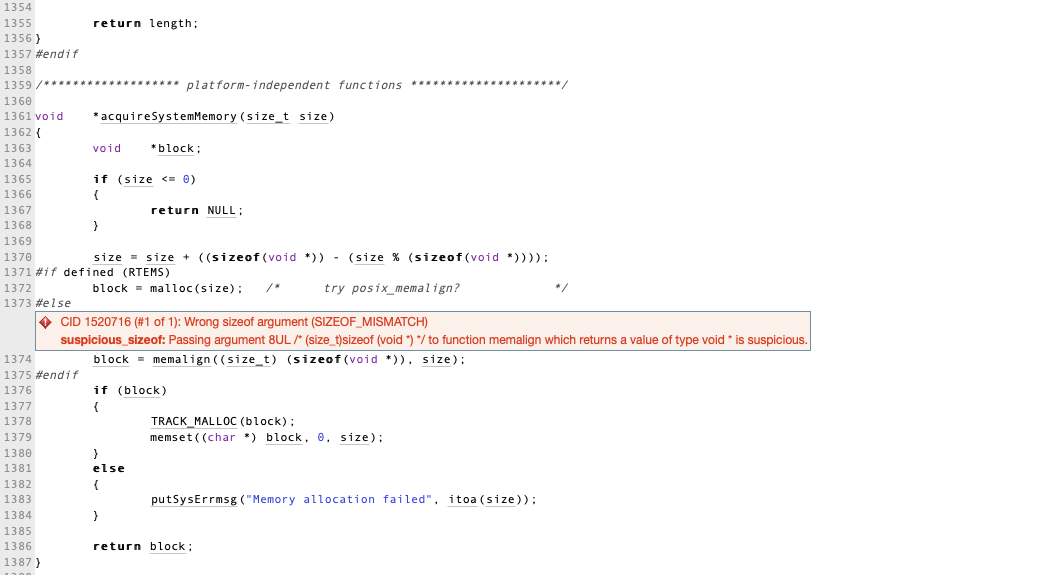
## Observations

The code snippet seeks to allocate memory for a data block and can be adapted to multiple platforms via conditional compilation. Specifically:

Lines 1372-1375: Memory allocation via memalign is performed when RTEMS is not defined.

Line 1373 raises a concern: the sizeof(void \*) parameter in the memalign call appears to be problematic.

## Supporting Evidence



# 

# Conclusions and Recommendations

Addressing the code's memory allocation strategy would involve correcting the usage of **memalign**. The second argument of **memalign** should be replaced with the appropriate value reflecting the desired memory alignment. For enhanced platform independence, a recommended approach is to consistently use **malloc** and manage memory alignment explicitly if necessary.

Furthermore, an opportunity for improvement lies in refining error handling techniques. Rather than a simple error message, considering mechanisms such as returning standardized error codes or utilizing exceptions would provide more structured feedback to the caller. In conclusion, while the code effectively allocates and initializes memory using **malloc** and **memset**, the misunderstanding of **memalign** arguments introduces potential risks. Enhancing platform independence and bolstering error handling procedures are advised. Rigorous testing and thorough review are paramount to ensure consistent memory allocation behavior across platforms and robust error management.

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.*