



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 ***Dereference null return value*** type defect identified in the following CIDs:

1520707

# 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 provided code snippet contains C code for three functions: ui\_str\_from\_op, ui\_str\_from\_rpt, and ui\_str\_from\_rpttpl. These functions are responsible for converting specific data structures into character strings. The code also includes comments with code analysis and a reference to a static code analysis tool (CID 1520707) with identified issues. A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit. NULL pointer dereference issues can occur through a number of flaws, including race conditions, and simple programming omissions.

## Observations

ui\_str\_from\_op: This function calls **ui\_str\_from\_ari** with the op structure's id field and returns the result. Apart from the function call, it doesn't do anything significant. **ui\_str\_from\_rpt**: A "TODO" note indicates that this function's implementation is incomplete. It returns NULL if no operations are performed.**ui\_str\_from\_rpttpl:** This function converts the rpttpl structure into a character string.It uses STAKE (memory allocation function) to allocate memory for the result.It stores the result of ui\_str\_from\_ari(rpttpl->id, NULL, 0) in a local variable tmp.If tmp is NULL, the code assigns the value "null" to fmt.It then uses strcat to concatenate strings and values into the result variable.

## Supporting Evidence

## 

## 

# Conclusions and Recommendations

Conclusion:

The ui\_str\_from\_op method appears to be complete and simple, transforming the id field of the op structure into a character string.

The ui\_str\_from\_rpt function is unfinished (marked with "TODO") and returns NULL with no operations performed. To be helpful, it must be implemented.

The ui\_str\_from\_rpttpl method returns a character string from a complex data structure. Dynamic memory allocation, string manipulation, and concatenation are employed.

When using strcat(result, fmt), the static code analysis tool (CID 1520707) discovered a potential issue with null pointer dereference in the ui\_str\_from\_rpttpl method.

Recommendation:

To guarantee that the ui\_str\_from\_rpt method performs its intended purpose, provide the missing functionality.

When using strcat(result, fmt), the static code analysis tool detected a null pointer dereference in the ui\_str\_from\_rpttpl method. Before concatenation, ensure that the result variable is properly created and allocated.

Consider adding comments and documentation to these methods to clarify their purpose and usage, particularly ui\_str\_from\_rpttpl, which requires more sophisticated processes. This will make future developers' code more maintainable and understandable.

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.*