



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 ***untrusted loop bound*** type defect identified in the following CIDs:  
***1520808***

# 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

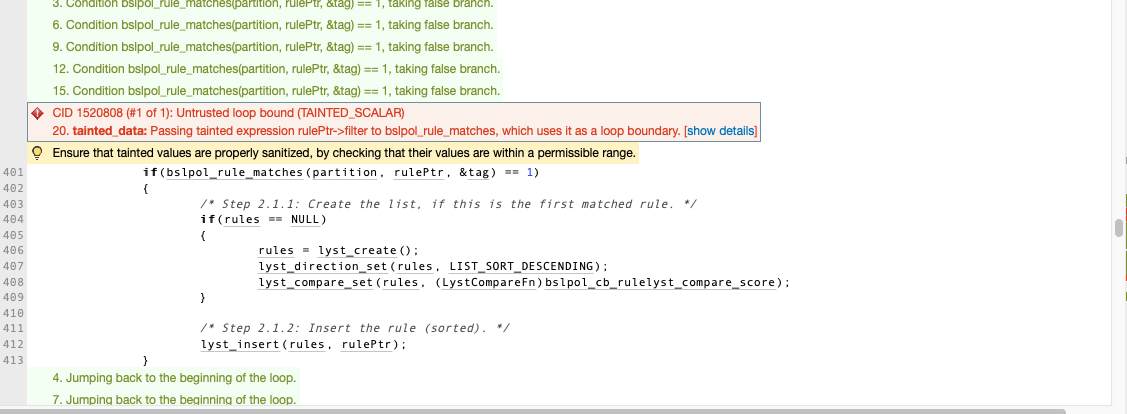
## Overview

The code snippet involves a loop that iterates over a set of conditions to determine if a certain rule matches. If the rule matches, it proceeds to create a list and insert the matched rule into it. The loop repeats until all conditions have been evaluated.

## Observations

The code snippet contains a loop that checks conditions using the function bslpol\_rule\_matches(partition, rulePtr, &tag) == 1. These conditions seem to represent a rule-matching logic.The code initializes a list called rules and sets its sorting and comparison functions if it's the first time a rule matches.When a rule matches, it inserts the rule into the rules list.The loop appears to continue until all conditions have been evaluated, and then it returns the rules list.There is a warning (CID 1520808) related to a potentially tainted value (rulePtr->filter) being used as a loop boundary. It suggests ensuring that tainted values are properly sanitized.

## Supporting Evidence

**

# Conclusions and Recommendations

The code is intended to process rules and generate a sorted list of rules that correspond to particular situations. However, CID 1520808 raises a potential security problem with the use of a tainted value as a loop boundary. If the contaminated data is not properly cleaned, this could lead to vulnerabilities.

Validate and sanitize the rulePtr->filter value before using it as a loop boundary. To avoid security risks, ensure that it fits inside an acceptable range.

Consider including comments or documentation to clarify the code's purpose and anticipated behavior, especially if it's part of a bigger program.

Examine the loop's general logic and conditions to ensure they accurately represent the desired rule-matching criterion.

Test thoroughly, including edge cases and boundary conditions, to ensure the code's correctness and the rule-matching process's behavior.

Addressing the potential security risk and validating the code will assist in guaranteeing that it works as intended and does not add vulnerabilities to the system.

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.*