Coverity Scan Static Analysis Report

Hardhard Enterprises

**T1** 2023

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 |  |  |
|  |  |  |  |
|  |  |  |  |

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 ***Unchecked return value*** type defect identified in the following CIDs:  
**1520701**

# Acronyms and Abbreviations

Please keep an updated list of acronyms and abbreviations used throughout the report.

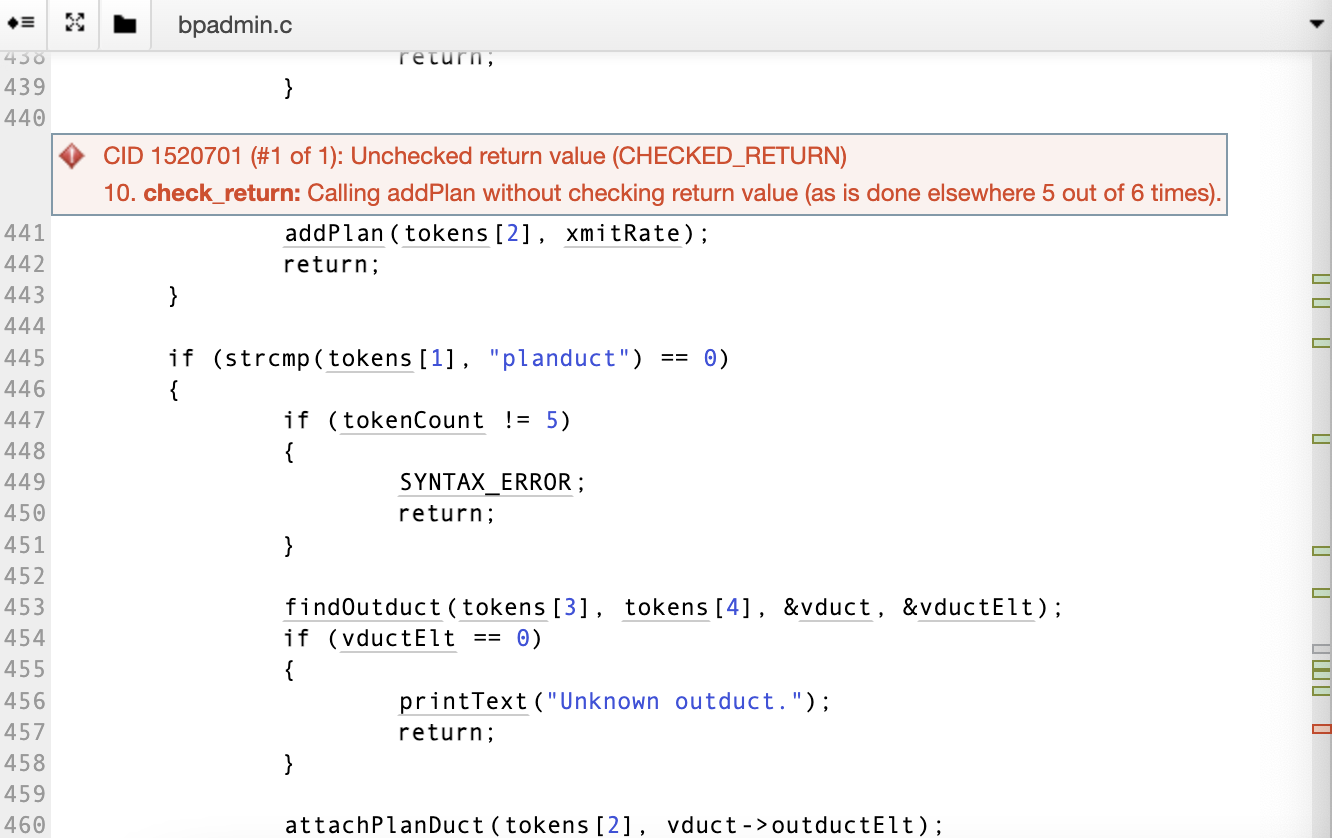
|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| DTN | Delay/Disruption Tolerant Network |
| ION | Interplanetary Overlay Network |
|  |  |

# Code Review and Analysis

## Outcomes

This report is looking at the unchecked return value defect for CID1520701.

## Observations

**

Coverity has discovered an unchecked return value effort for CID1520701. The warning is indiciating that the code has called a function – addPlan – but has not checked its return value, and this is inconsistent with the pattern found elsewhere in the code.

There is funcation call in the code – addPlan – which returns a value that is usually an error, because there has been no action taken to check or handle that return value.

The code is calling addPlan function without performing any type of validation or error-checking on the return value it provides.

The 5 out of 6 times part of the error message compares the current call to addPlan with the previous calls to the same function in your codebase. It is saying that in this instance, you’re not checking the return value, but in 5 out of 6 similar instances somewhere else in the code, the return value is checked.

The warning suggests that the issue is when the addPlan function is called, it returns a value, but that return value isn’t checked. This inconsistency in error handling can elad to unexpected behaviour or problems if the function encounters an error condition, and the error is not approporiately handled.

# Conclusions and Recommendations

To fix the issue, the code should be reviewed and the addPlan function should be checked to understand what values it can return and under what conditions it returns errors. Next, error-checking logic should be implemented using conditional statements or error codes to handle potential errors returned by addPlan appropriately. This ensures that your code behaves consistently and handles errors properly when they occur.