**T1** 2023

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** |
| 15/09/2023 | V0.1 | Nicholas Mag |  |
|  |  |  |  |
|  |  |  |  |

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 ***{Issue Category}*** type defect identified in the following CIDs:  
***{Coverity Issue CID}***

# 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

All code related to this resource leak seems to be contained within the dccplsi.c file within the filepath

/ltp/dccp/dccplsi.c

The code relates to the connection handling within the main loop for DCCP while it listens for connections.

## Observations

This leak seems to be of moderate severity, as it is contained to one file. The variables have seemingly gone out of the scope when the freshly assigned memory has been assigned. This has caused a small memory leak within the code.

If the leak was to remain it could cause the program to run for long periods of time without restarting and potentially lead to further crashes and errors within the program, even if it was a minor leak.

The fact the leak is within the listening loop for the program indicates that if it goes unfixed, there may also be errors when it is run as the memory allocated may not be valid within the rest of the program.

## Supporting Evidence

# <https://scan7.scan.coverity.com/doc/en/cov_checker_ref.html#static_checker_RESOURCE_LEAK> Conclusions and Recommendations

The recommendation here is to try and fix the memory leak. There is a file descriptionleak in the consock file descriptor that is opened but never closed. So for this specific leak it should be closed before the loop continues.

The other memory leak identified is in the ‘list’ and ‘elk’ variables. There appears to be no calls to release the resources used when the thread exits, thus causing the leak. The recommendation is to destroy the ‘list’ and ‘elk’ variables at the end of the function.

With these recommendations the leak should be sealed and no further errors in these functions should occure.

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.