**T2** 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** |
| 21/08/2023 | V1.0 | Adharshaan Devaraj | Initial Document |
|  |  |  |  |
|  |  |  |  |

Table of Content

Contents

[Introduction 3](#_Toc133624419)

[Objective 3](#_Toc133624420)

[Scope 3](#_Toc133624421)

[Acronyms and Abbreviations 3](#_Toc133624422)

[Code Review and Analysis 4](#_Toc133624423)

[Overview 4](#_Toc133624424)

[Observations 4](#_Toc133624425)

[Supporting Evidence 4](#_Toc133624426)

[Conclusions and Recommendations 4](#_Toc133624427)

[References 5](#_Toc133624428)

[Appendix 5](#_Toc133624429)

# 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 value as argument*** type defect identified in the following CIDs:  
***1520678***

# 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 |
| CWE | Common Weakness Enumeration |
| BIB | Bundle Integrity Block |
| BCB | Bundle Confidentiality Block |

# Code Review and Analysis

## Overview

By analyzing “**libcfdp.c”** code base within the “library” folder of the CCSDS File Delivery Protocol directory. Coverity highlighted a “***untrusted value as argument*”** error that highlights the use of a variable (**cfdpConstants->maxFileDataLength)** to specify an offset in a read operation, and this variable might not be properly validated or sanitized, which can lead to vulnerabilities like buffer overflows.

## Observations

The error untrusted value as argument is caused due to improper input validation which means the product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.

The function “**cfdp\_read\_text\_lines**” essentially reads text lines from a file, calculates a checksum, and updates the file pointer for the next read operation. It also handles error conditions by returning -1. The error exists within the function and as a result could be controlled by the attacker who could invoke the function with arbitrary values (for example, a very high or negative buffer size).

## Supporting Evidence

A screenshot of a computer

Description automatically generated

Figure - CID 1520678

The above screenshot represents the error that was identified in the code within the “**libcfdp.c” file.** The reason for this error to be detected is due to improper input validation. (CWE, 2006)

# Conclusions and Recommendations

This is a vulnerability that can be exploited by an attacker and hence implementing input validation is required. Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications or transform it into something that does.

When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue."

Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

# References

CWE, 2006. *CWE-20: Improper Input Validation.* [Online]   
Available at: https://cwe.mitre.org/data/definitions/20.html  
[Accessed 2023].

Appendix

Include additional information/documentation here to help the readers understand complex information.