**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** |
| 14/9/23 | 1.0 | Dean Scanlon | Initial document |
|  |  |  |  |
|  |  |  |  |

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 ***Insecure data handling*** type defect identified in the following CIDs:  
***CID\_1520829***

# 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

This issue occurs in the libcfdp.c code segment within the cfdp/library/ directory of the NASA ION DTN protocol.

Coverity Static Analysis Tool was able to detect **Medium Impact Quality** vulnerabilities that relate to an **Untrusted value as argument** error.

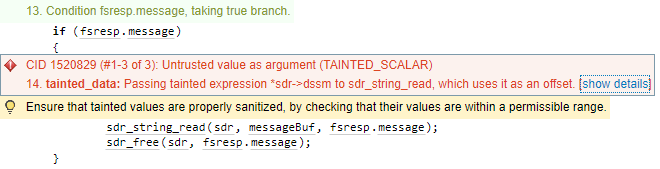
## Observations

This issue has to do with a code segment which is part of the CCSDS File Delivery Protocol (CFDP) within NASA ION DTN protocol. Ultimately it is a validation problem stemming from variables not being validated prior to use. Coverity defines this as a ‘tainted’ value, and as such flags any use of these tainted variables in further functions, and how each value can be passed from one function to another. If the value of the variable is outside of the bounds of what a function is expecting, in size or type, there is potential for erroneous computations to be made.

Ultimately the issue in question affects the memory management functions **memcpy** and ‘**memset**’. These are essential functions when it comes to managing the memory in NASA ION DTN, especially when considering the protocol’s ability to negate data loss when connections fail due to physical proximity or power losses. They are responsible for copying and filling memory buffers that are used to store data in the network and as such any value being utilized by either function should be validated to ensure ongoing functionality of the network.

On line 651 of the code segment, the expression ‘**sdr->dssm** is passed to **sdr\_string\_read** function which uses it as a memory offset as shown in Fig1. The dssm field of the sdr struct has not been validated and as such, if **sdr\_string\_read** uses this offset to read a string from sdr and the value of dssm is not within a permissible range, problems could arise for the functionality of the code.

Further along, **sdrFetch** function is called using the possibly tainted argument **dssm**, as is the **memcpy function** which sets the **into** variable’s value (Fig2). If an error has occurred during these computations and the value of the **into** variable is tainted, it could also have onflowing effects when **memset** function is called using that as an argument.



**Fig1 Coverity error CID 1520829**

A close-up of text

Description automatically generated

**Fig2 memcpy function called using tainted argument dssm**

A screenshot of a computer

Description automatically generated

**Fig3**

# Conclusions and Recommendations

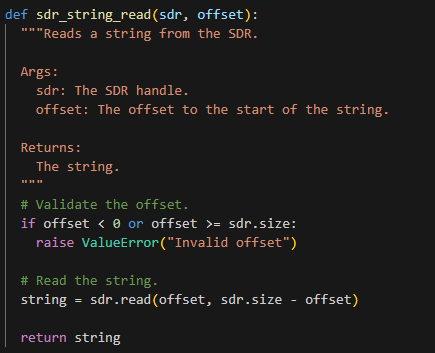
Within the NASA ION DTN protocol, a number of variables are not validated which can result in problems with the processing and allocation of memory within the code. NASA ION DTN should be altered to include such validation functions to ensure program correctness, security and stability. Without proper validation, programs can access unallocated memory which can lead to improper computations of values and to undefined behaviour within the program. A basic validation function is shown in Fig4. This example checks that the variable into is an integer of 4 bytes.

A computer screen shot of a code

Description automatically generated

**Fig4 Example of validation function**

The **dssm** field of the **sdr** struct is not validated before it is used as an argument by **sdr\_string\_read**. The code should be altered to include a function which checks if the offset is within the bounds of the **sdr** struct and hence can be safely used by the **memcpy** and **memset** functions that deal with memory allocation. Included below in Fig5 is an out of context code segment showing a validation function which would achieve this task

****

**Fig 6: Example code for sdr\_string\_read validation of offset**

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.