**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** |
| 29/04/2023 | V0.1 | 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 4](#_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 ***High Impact Quality*** type defect identified in the following CIDs:  
***1520720***

# 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

By analyzing “**adm\_bpsec\_impl.c”** code base within the **administration (adm)** folder of the Bundle Protocol v7 directory. Coverity highlighted a “**high impact quality”** vulnerability with the use of casting INT to the 32-bit time\_t data type at line 1631 in the **tnv\_t \*dtn\_bpsec\_get\_last\_reset(tnvc\_t \*parms)** function which collects the last BPSEC update from the source node.

## Observations

A 32-bit unsigned integer encodes a non-negative integer within the range 0 to 4294967295. The challenge here is that if a primitive data structure is given a value greater than what it can hold, the program may lose data. As a result, if the “**result->value.as\_uint = misc.last\_reset”** condition gets a value that the unsigned integer data type cannot hold, the program's execution may fail.

The "**adm\_bpsec\_impl.c**" file implements functions from the BPSEC Administration API in ION, which provides the setting and administration of security policies and bundle information. The function **tnv\_t \*dtn\_bpsec\_get\_last\_reset(tnvc\_t \*parms)** is used to acquire the timestamp of the most recent BPSEC protocol reset within the BPSEC Agent.

The **dtn\_bpsec\_get\_last\_reset()** function is part of the administrative API for the BPSEC Agent, which includes methods for controlling and monitoring the BPSEC protocol. External applications can use this function to retrieve information about the BPSEC Agent's state and behavior.

## Supporting Evidence

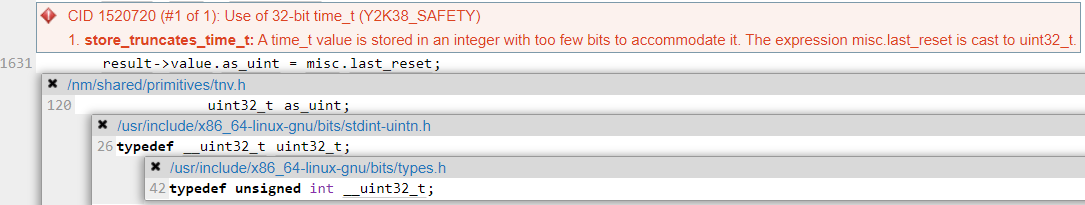


Figure - Value is converted to Unsigned INTEGER

The above screenshot represents that as\_uint is an unsigned integer which cannot hold a larger value than the maximum integer value can, hence the error being thrown which was identified by Coverity. (CWE-197: Numeric Truncation Error, 2023)

# Conclusions and Recommendations

This is not an exploitable vulnerability. The solution to that issue would be to convert the “**result->value.as\_uint = misc.last\_reset”** function to a signed integer data type. A signed integer can include both positive and negative integers. This must then be verified to confirm that the code is still valid. An example of a bigger integer would be to use **uint64\_t** to store the time\_t value therefore, you can ensure that the full range of time\_t values is accommodated and avoid any truncation errors.

# References

*CWE-197: Numeric Truncation Error*. (2023, April 20). Retrieved from Common Weakness Enumeration: https://cwe.mitre.org/data/definitions/197.html

Appendix

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