**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** |
| 21/04/2023 | V0.1 | Moe Khant Kyaw | Initial Document and analysis |
| 30/04/2023 | V0.2 | Moe Khant Kyaw | Finalizing Investigation and documentation |

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 ***High Impact Quality*** type defect identified in the following CIDs:  
***1520735***

# 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

CID 1520735 has been marked as high impact issue, Use of 32-bit time\_t type under High impact quality category by Coverity. This issue is marked as CWE-197: Numeric Truncation Error, which is the potential loss in bits of large value when converting primitive to a smaller primitive and thus resulting in an unexpected value.

## 

## Observations

This issue is found in “main” function under “/bpv7/brs/brsccla.c” filepath. Coverity indicated that the use of “ntohl(x)” and “hton(x)” functions to cast timeTag from network byte order to host byte order at line 343, 361 and 363 respectively. “ntohl(x)” and “hton(x)” functions takes an unsigned 32-bit integer and returns an unsigned 32-bit integer that has the byte reversed, using “\_\_bswap\_32” function. timeTag is cast to \_\_uint32\_t, unassigned 32\_bit integer. Because of this, Coverity is indicating this as an issue as time\_t value is stored in an integer with too few bits to accommodate it. This can lead to the issue with program execution.

## Supporting Evidence

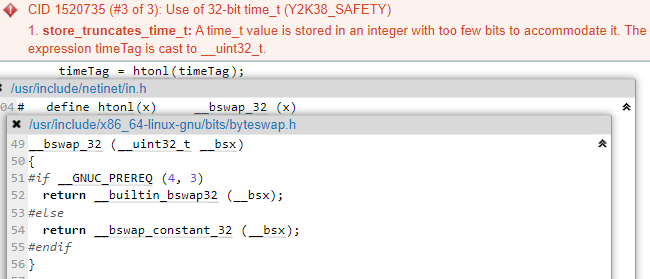


Figure 1: timeTag casted to \_\_unint32\_t by *htonl* function

timeTag is cast to an unassigned integer and it is marked as CWE-197 error by Coverity as it involves errors that occur when a primitive is cast to a smaller primitive and potential bits of data is lost in the process.

# 

# Conclusions and Recommendations

This is not an exploitable vulnerability.

Remedies to fix this issue:

* A larger integer type such as 64-bit integer can be used to store *time\_t* value and that will prevent any potential loss when storing “time\_t” value.
* Checking “time\_t” value before storing it so that it can fit in an integer with the available bits for “time\_t”.

References

*Common weakness enumeration* (2006) *CWE.* Available at: <https://cwe.mitre.org/data/definitions/197.html>, Retrieved: April 28, 2023.

Appendix

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