**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** |
| 5/05/2023 | V0.1 | John-Eddie Cubis | Iteration One |
| 6/05/2023 | V0.2 | John-Eddie Cubis | Iteration Two |
| 14/05/23 | V0.3 | John-Eddie Cubis | Iteration Three |
| 15/05/23 | V1 | John-Eddie Cubis | Completion |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Content

Contents

[Introduction 4](#_Toc119848724)

[Objective 4](#_Toc119848725)

[Scope 4](#_Toc119848726)

[Acronyms and Abbreviations 4](#_Toc119848727)

[Code Review and Analysis 5](#_Toc119848728)

[Outcomes 5](#_Toc119848729)

[Observations 5](#_Toc119848730)

[Supporting Evidence 5](#_Toc119848731)

[Conclusions and Recommendations 6](#_Toc119848732)

[References 6](#_Toc119848733)

[Appendix 7](#_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 ***Out-of-bounds access*** type defect identified in the following CIDs:  
[***1520838***](https://scan7.scan.coverity.com/doc/en/cov_checker_ref.html#static_checker_OVERRUN)

[***1520682***](https://scan7.scan.coverity.com/doc/en/cov_checker_ref.html#static_checker_OVERRUN)

# 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 |
| PSM | Personal, Space Managment |
| SDR | Space Data Repository |
|  |  |
|  |  |
|  |  |

# Code Review and Analysis

## Overview

Coverity reports errors detected within ION 4.1.1, ION is a list of network protocols that may be used for communication in space exploration and other mission critical applications. These detections are [*Out-of-bounds*](https://scan7.scan.coverity.com/doc/en/cov_checker_ref.html#static_checker_OVERRUN) access errors within **‘Sdr\_hash\_revise’** & **‘bsles\_sdr\_serialise\_buffer’.**

The ‘Sdr\_hash\_revise()’ function is part of the SDR module in ION open source. Its purpose is to revise the value associated with a given key in the hash table, stored in the SDR.

The ‘bsles\_sdr\_serialise\_buffer’ function is part of the SDR module in ION open source. Its purpose is to handle serialization of ‘BpSecEventSet’.

## Observations

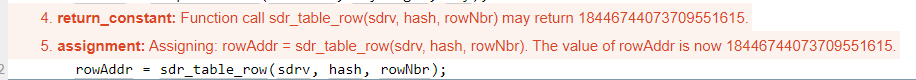
Overrun instances are used to validate buffers that are out of bounds, improper access may corrupt, crash, create security vulnerabilities, in memory, within a system.

* + Buffer overflows may be used to execute arbitrary code, that is outside the scope of a program’s implicit security policy.
  + Buffer overflows may also lead to crashes leading to lack of availability and putting the program into an infinite loop.

This error has a possibility of an exploit given the attacker knows the source code and the direct bugs that cause these problems calculations of this function activate, this must be fixed immediately.

CID 1520838 is an out-of-bounds error captured in Coverity Static Analysis , out of bounds errors are errors that attackers may exploit memory corruption issues. The issue occurs when ‘sdr\_read()’ function is called with an address that is out of bounds ‘rowAddr’ value, Coverity had detected the function because the value of ‘rowAddr’ is larger than the expected size of the buffer.

* Potential causes in /ici/sdr/sdrhash.c
* Calculation when hashing, causing overrun errors.



CID 1520682 is an out of bound error captured in Coverity Static Analysis, out of bounds errors are errors that attackers may exploit memory corruption issues. The issue occurs when calling ‘sm\_list\_data’, value ‘elt’ used to retrieve the data from its storage list, this value is suspiciously large which is an out-of-bounds access error. This may result in system errors, unintended memory access, instability, and program crashes.

* Documentation: ION 4.1.1, (/bpv7/library/ext/bpsec/bpsec\_policy\_eventset.c)
* Line 12 & 14, ‘elt’ produces value of 18446744073709551615.

A picture containing text, screenshot, font

Description automatically generated

When the value is initialised to 0 in the linked list, this makes the list invalid which unintentionally assigns the list to a large number which the picture above shows this error.

Consequences of an overrun error include Integrity Confidentiality and Availability

* Buffer overflows may be used to execute arbitrary code, that is outside the scope of a program’s implicit security policy.
* Buffer overflows may also lead to crashes leading to lack of availability and putting the program into an infinite loop.

This error has a possibility of an exploit given the attacker knows the source code and the problems that arise when calculations of this function activate, this must be fixed immediately.

## Supporting Evidence

SAR\_1520817: Author John-Eddie Cubis

# Conclusions and Recommendations

It's important to investigate the root cause of these issues, such as verifying the input parameters and ensuring that the indices used in list operations are valid and within the expected range. Additionally, reviewing the context and logic of the surrounding code could help in resolving these out-of-bounds access errors.

Possible mitigations include:

Requirements:

* Use language that does not allow weakness or provides constructs to avoid issues.
* Code is still subject to overflows even if the language is theoretically safe.

Architecture and Design:

* Use vetted library or framework.

When elements are used to access linked lists, we must validate values within the list and that they exist. This is to ensure the list is not risk of being overrun.

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.