Coverity Scan Static Analysis Report

Hardhard Enterprises

**T3** 2022

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** |
| 14/04/2023 | V1 | Jesse Ludeman | Initial document and investigation |
| 18/04/2023 | V1.1 | Jesse Ludeman | Finalize investigation and complete 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 ***Memory – corruptions*** type defect identified in the following CIDs: 1520860

# 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

## Introduction

CID 1520860 has been flagged by Coverity as a High impact issue of type Out-of-bounds access. This type of issue generally means that the program can read or write to a memory location that is outside of the intended boundary of the given buffer.

## Observations

### Introduction

This issue occurs in the bsl\_remove\_sop\_target\_at\_sender function in the /bpv7/library/ext/bpsec/bpsec\_policy.c file. As per the commentary in the file, this function’s purpose is to remove the security target of the security block provided.

The initial error that Coverity raises is for the function bsl\_findOutboundTarget, which calculates and assigns the return value into the tgt variable. Note that this function call assigns 18446744073709551615 into the tgt variable.

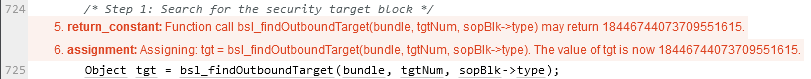


Figure Initial issues raised in Coverity

Whilst there’s not a lot of documentation or commentary available for the bsl\_findOutboundTarget function, it appears that this is designed to find an outbound security block in a given target bundle protocol packet.

Reviewing the logic in the function we have confirmed there is no presence of input validation for parameters [Bundle, int, and BpBlockType]. However, if we assume that the given parameters are valid *and* that a valid BPsec block is provided, then the elt2 value is calculated and returned.

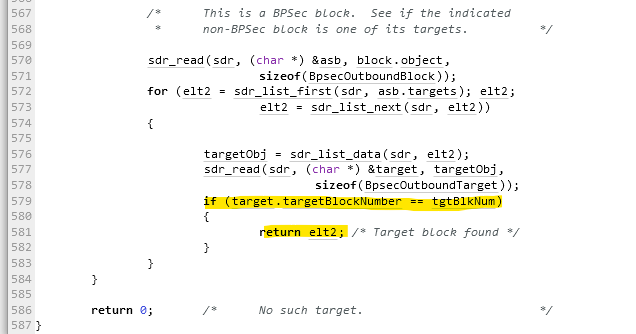


Figure - If statement for bsl\_findOutboundTarget function

The next error we need to investigate occurs on line 729 inside the if statement, for the sdr\_list\_delete() function. As the name suggests, this deletes an elt object from the list if it’s present. Additionally, on line 475 a call to the sdrFetch() function is made with the index set to elt, which we previously know was set to 18446744073709551615.

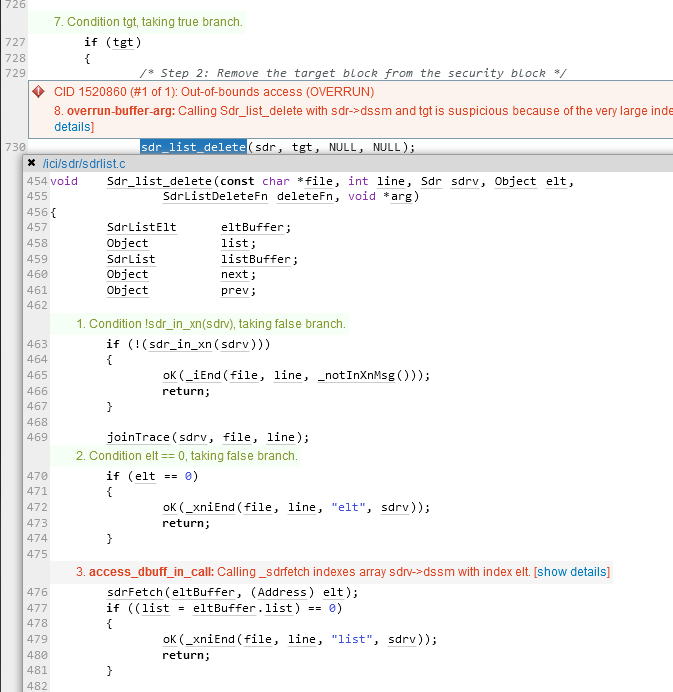


Figure - Sdr\_list\_delete function

Inspecting the \_sdrfetch() function appears to indicate that as the name suggests, will fetch an SDR (Simple Data Recorder) object from the application. Assuming that the result is True inside the if statement, then it uses the memcpy() function to copy the data that resides in the from and into variables into each other’s memory.

The main error in question is the *“…requires the indexes to be no more than the number of elements in the buffer”*. This means \_sdrfetch() is indexing the array sdrv->dssm with the elt value, which we know previously from the previous is 18446744073709551615. This value is far too large be to used as an index. This is the reason why Coverity has flagged this as an error.

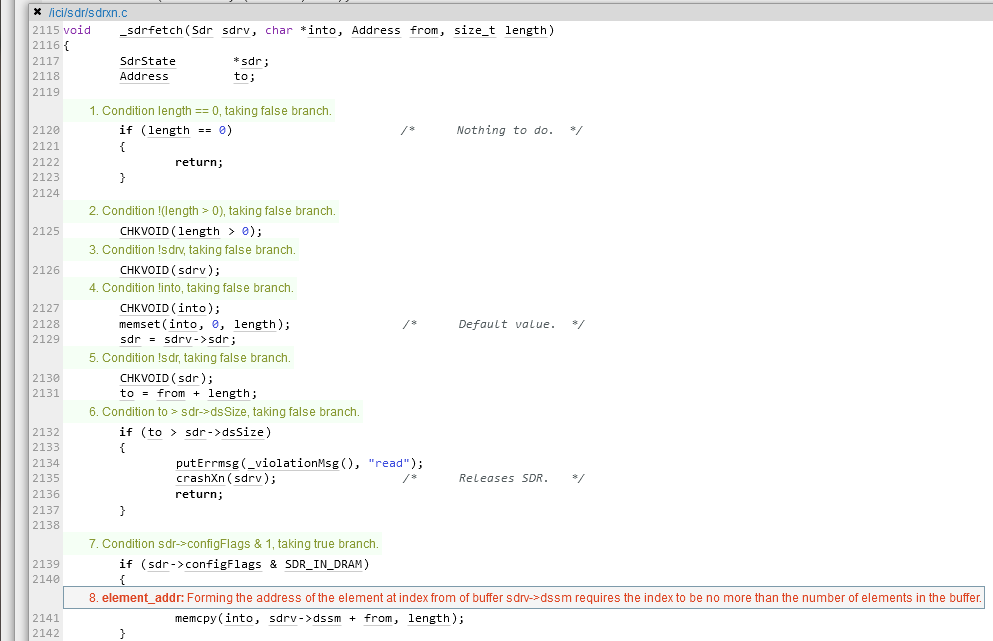


Figure - \_sdrfetch function

### What is an Out-of-bounds access issue?

This error indicates that the program has attempted to read or write to a block of memory or buffer outside of the valid boundary/size of the intended buffer. This is typically caused by the developer of the program, and will require mitigation to resolve.

This can pose a risk to the overall security posture of the application, given that reading or writing to a different memory location can expose different variables, data structures or even internal program data.

## Supporting Evidence

# Conclusions and Recommendations

Hardhat Enterprises recommends the following changes be implemented:

* Add input validation in the bsl\_sdr\_list\_first() function. This will prevent it from returning a very large result such as 18446744073709551615.

Furthermore, out-of-bounds access issues can introduce a risk to the security posture of the application. Should an attacker be made aware of this issue, they may be able to execute artbitrary code, read values from the application, or cause the system to crash.

References

Appendix