**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** |
|  | V0.1 | Jaspriya |  |
|  |  |  |  |
|  |  |  |  |

Contents

Table of Content

[Introduction 3](#_bookmark0)

[Objective 3](#_bookmark1)

[Scope 3](#_bookmark2)

[Acronyms and Abbreviations 3](#_bookmark3)

[Code Review and Analysis 4](#_bookmark4)

[Outcomes 4](#_bookmark5)

[Observations 4](#_bookmark6)

[Supporting Evidence 4](#_bookmark7)

[Conclusions and Recommendations 4](#_bookmark8)

[References 5](#_bookmark9)

[Appendix 6](#_bookmark10)

# 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  ***‘ Untrusted value as argument’*** type defect identified in the following CIDs:

1520707

1520850

# 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 |
| CID | Coverity Issue Identification Number |
| CWE | Common Weakness Enumeration |

# Code Review and Analysis

Overview

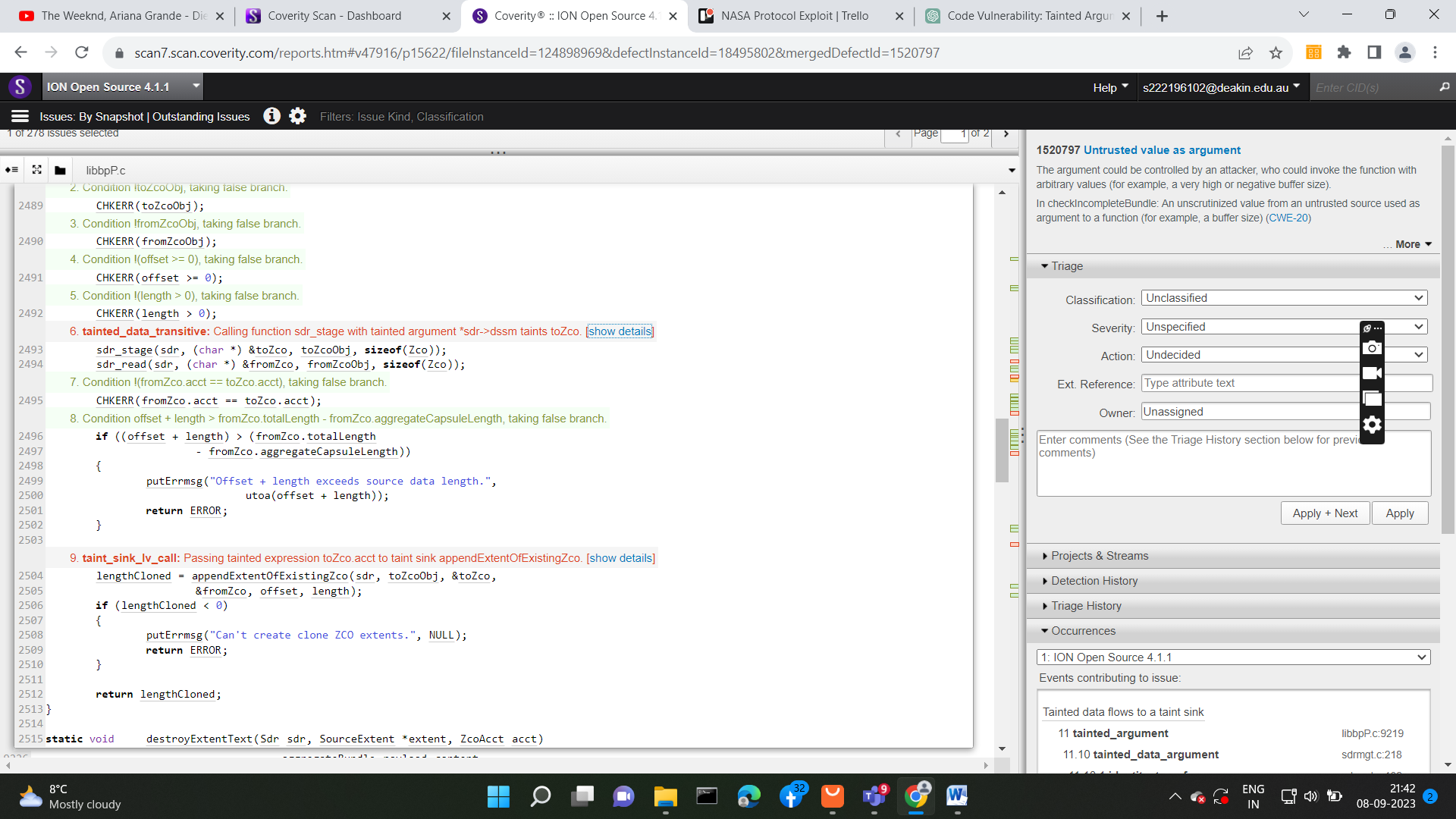
Coverity has flagged CID-1520797, CID-157850 as a Medium-severity vulnerability in the `serializeDataPDU` function, categorized under insecure data handling. Specifically, it highlights the presence of an unvalidated input value originating from an untrusted source. This vulnerability, classified under CWE-20 (Improper Input Validation), implies that an attacker might manipulate the argument, potentially invoking the function with arbitrary and malicious values, posing a security risk.

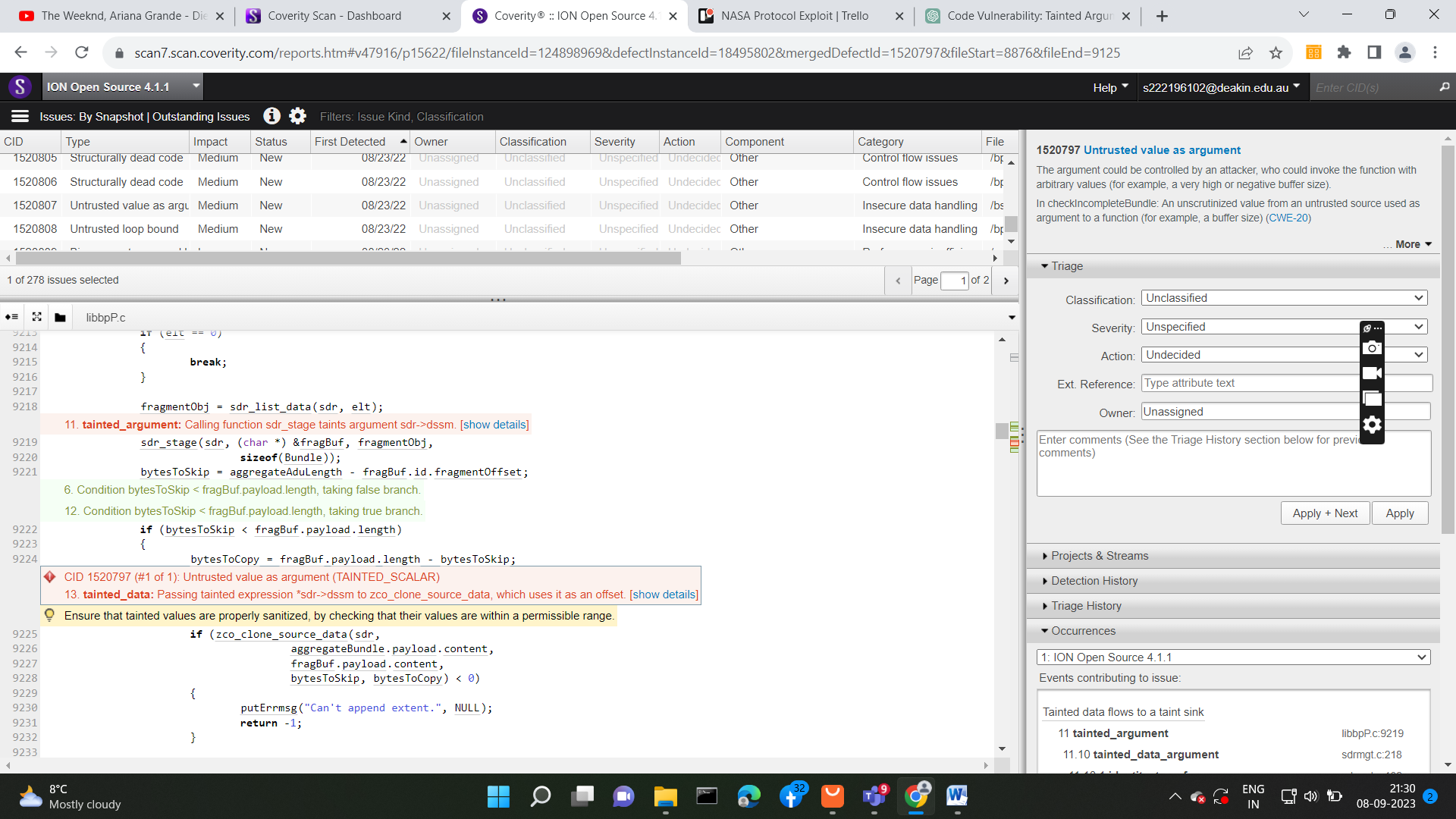
## Observations

CID 1520707:

The problem in line 9219 is that calling the `sdr\_stage` function taints the argument `sdr->dssm`. This indicates that the argument `sdr->dssm` is derived from an untrusted or potentially malicious source, which can lead to security vulnerabilities when used as input for database operations like `sdr\_stage`.

Lines 6 to 9 in the code snippet reveal multiple critical errors. Line 6 taints `toZco` when calling `sdr\_stage` with a tainted argument. Subsequently, line 9 passes tainted `toZco.acct` to `appendExtentOfExistingZco`, a taint sink, compounding the issue. Additionally, the code lacks proper validation, potentially causing security vulnerabilities.

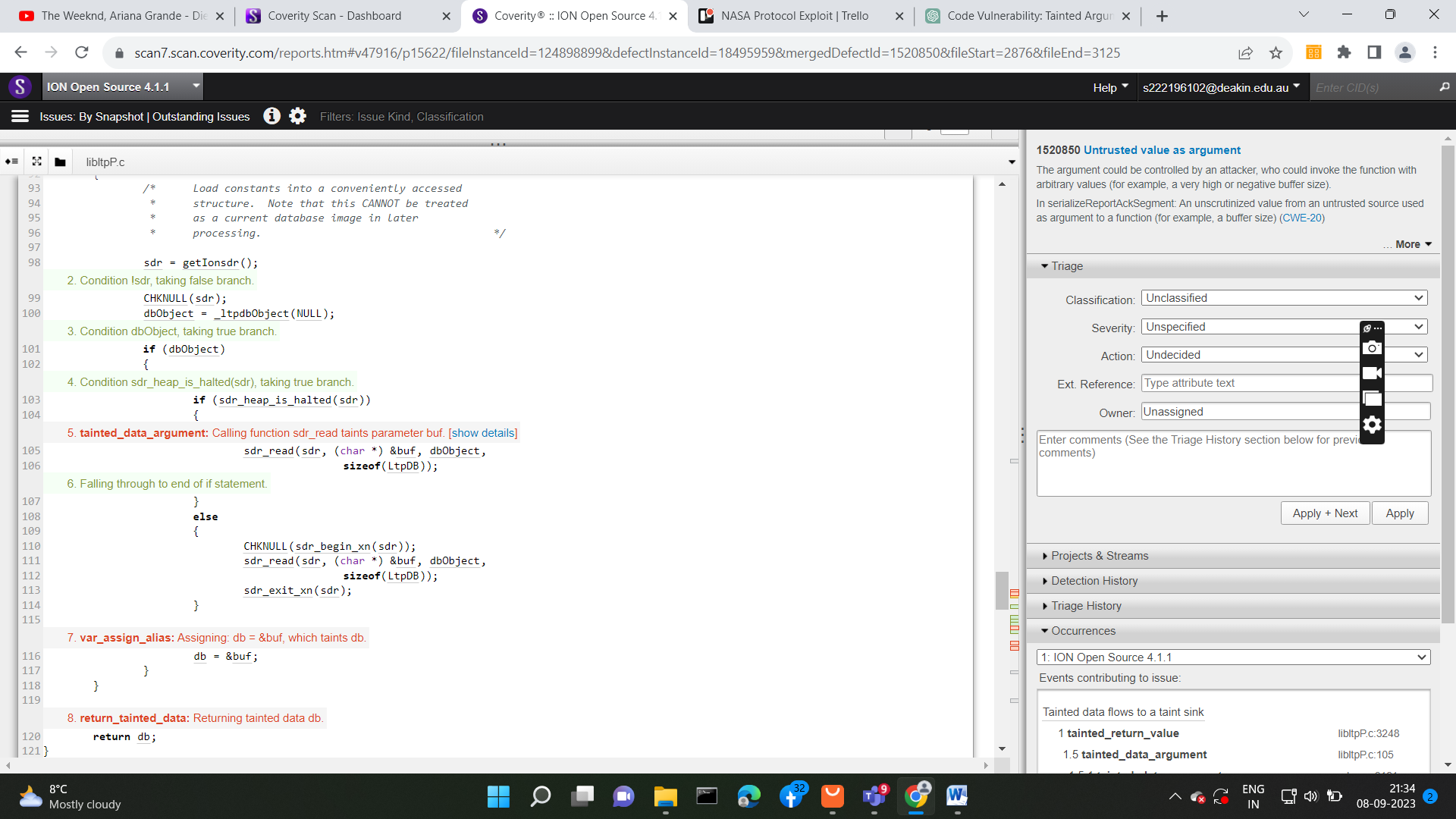


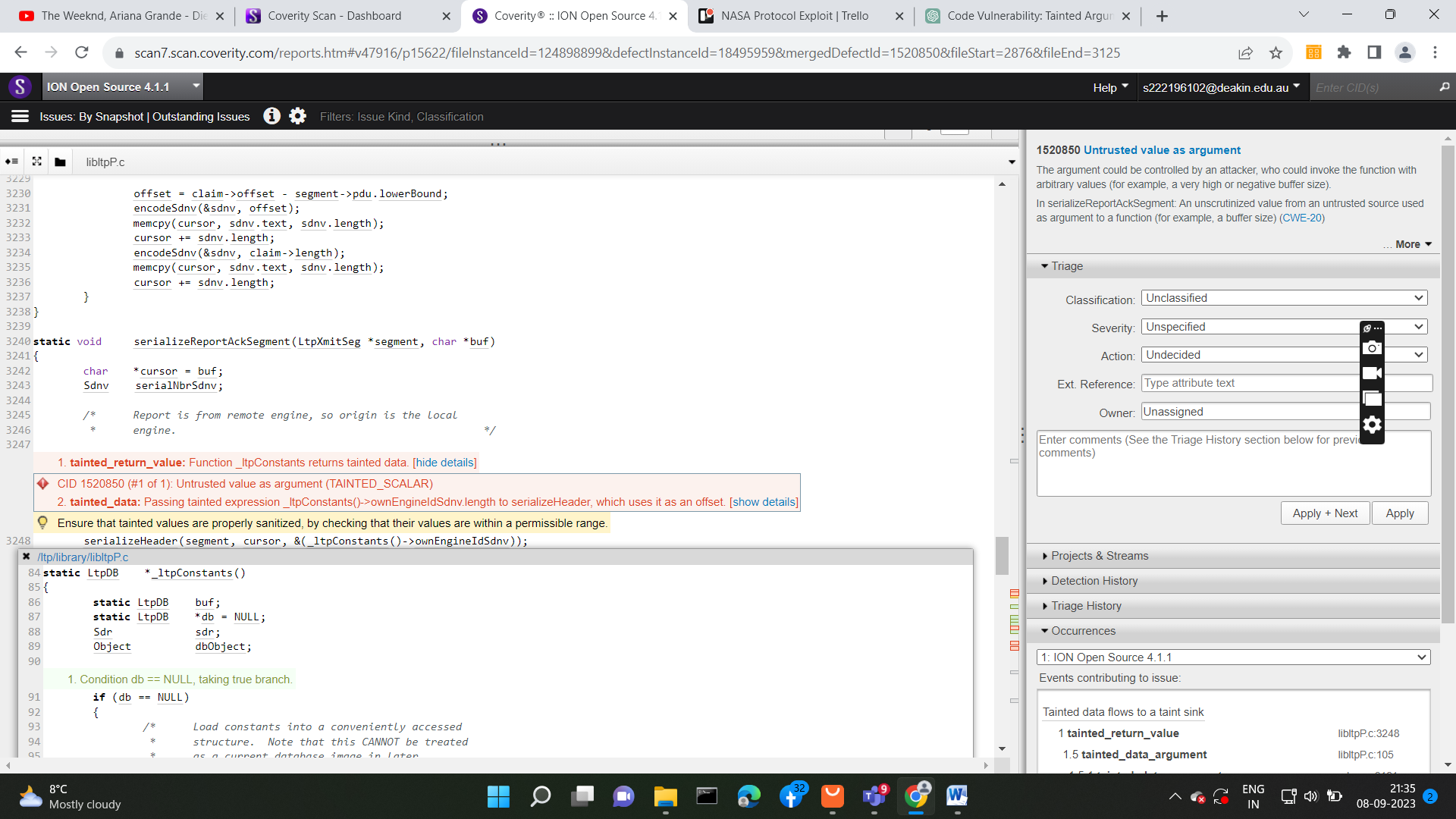


CID 1520797

CID 1520850:

In line 5, a security concern arises as the `sdr\_read` function taints the `buf` parameter, potentially introducing untrusted data. Line 7 compounds the issue by creating an alias for `db` linked to the tainted `buf`. Consequently, returning `db` in line 8 means the function returns tainted data, posing security risks due to potential untrusted sources. Addressing these problems requires proper validation, avoidance of unnecessary aliases, and careful handling of data before returning it to ensure code security.





## Supporting Evidence

# Conclusions and Recommendations

This security vulnerability is remediable, and it opens the door for potential attacks, allowing an attacker to invoke the function with arbitrary values, posing a significant security threat. To address this issue, it is highly advisable to sanitize the tainted inputs by implementing robust validation checks to ensure they fall within acceptable and safe ranges. This proactive approach will effectively bolster the system's security and protect against potential exploits.

# References

# Appendix

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