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 formatted 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** |
| 2/12/2022 | 0.1 | Connie Cox | Initial Investigation |
|  |  |  |  |

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 to fix.
* Any exploit for consideration.

## Scope

This static code analysis is limited to the ***Resource Leak*** type defect identified in the following CIDs:  
**1520893**

# Acronyms and Abbreviations

Please keep an updated list of acronyms and abbreviations used throughout the report.

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| CID | Coverity Issue Identification Number |
| DTN | Delay/Disruption Tolerant Network |
| ION | Interplanetary Overlay Network |
|  |  |

# Code Review and Analysis

## Outcomes

Please provide an in-depth report on the analysis performed here.

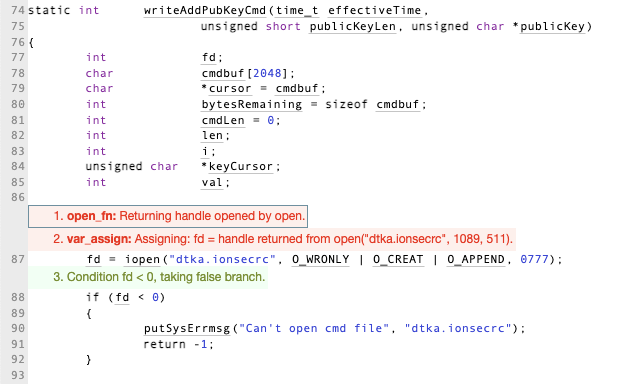
The intention of the writeAddPubKeyCmd function is to write an “add public key” command to the dtka.ionsecrc file. When the function is unable to add the public key to the file, it displays an error and returns.

## Observations

Please provide any observations, and feel free to make references to documents in the appendix.

The function attempts to open dtka.ionsecrc file in read/write mode. If it encounters an error, the function returns with an error message and no file is created or modified. If the return value is non-negative, it is assigned to variable representing the file descriptor.

There are several issues being raised by Coverity regarding the file descriptor variable. In C programming, functions are pass by value. When passing a file handle in/to a function, the function receives a copy of it allowing the function to modify the file handle locally. Once the function completes, whatever was done to the file locally is lost to outside functions. See screenshot below.



## Supporting Evidence

Please provide any supporting evidence, and feel free to make references to documents in the appendix.

According to GNU Org(*Opening and Closing Files*), file handling uses a structure pointer of *file type* should be used when declaring a file descriptor.

# Conclusions and Recommendations

Please provide any conclusions and recommendations here. Include references to supporting evidence and any other required information in the appendix.

Whilst this issue isn’t necessarily exploitable, this is a code bug that should be fixed. The recommendation is the following:

* Declare a File type descriptor 🡪 FILE \*fd rather than int fd on line 4
* Pass the pointer \*fd as &fd for use in functions on line 15 and 38

1. static int writeAddPubKeyCmd(time\_t effectiveTime,

2. unsigned short publicKeyLen, unsigned char \*publicKey)

3. {

4. int fd;

5. char cmdbuf[2048];

6. char \*cursor = cmdbuf;

7. int bytesRemaining = sizeof cmdbuf;

8. int cmdLen = 0;

9. int len;

10. int i;

11. unsigned char \*keyCursor;

12. int val;

13.

14. fd = iopen("dtka.ionsecrc", O\_WRONLY | O\_CREAT | O\_APPEND, 0777);

15. if (fd < 0)

16. {

17. putSysErrmsg("Can't open cmd file", "dtka.ionsecrc");

18. return -1;

19. }

20.

21. len = \_isprintf(cmdbuf, sizeof cmdbuf, "a pubkey " UVAST\_FIELDSPEC

22. " %d %d %d ", getOwnNodeNbr(), effectiveTime,

23. getCtime(), publicKeyLen);

24. cursor += len;

25. bytesRemaining -= len;

26. cmdLen += len;

27. for (i = 0, keyCursor = publicKey; i < publicKeyLen; i++, keyCursor++)

28. {

29. val = \*keyCursor;

30. isprintf(cursor, bytesRemaining, "%02x", val);

31. cursor += 2;

32. bytesRemaining -= 2;

33. cmdLen += 2;

34. }

35.

36. \*cursor = '\n';

37. cmdLen += 1;

38. if (write(fd, cmdbuf, cmdLen) < 0)

39. {

40. putSysErrmsg("Can't write command to add key", "dtka.ionsecrc");

41. return -1;

42. }

43.

44. close(fd);

45. return 0;

46. }

47.

* 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).
* *Opening and Closing Files*. GNU Org. <https://www.gnu.org/software/libc/manual/html_node/Opening-and-Closing-Files.html>

Appendix

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

Other websites that provide information on how C handles files are:

* [Free Code Camp’s File Handling in C – How to Open, Close, and Write to Files](https://www.freecodecamp.org/news/file-handling-in-c-how-to-open-close-and-write-to-files/)
* [Linux manual page](https://man7.org/linux/man-pages/man2/open.2.html)