OpenSSL:

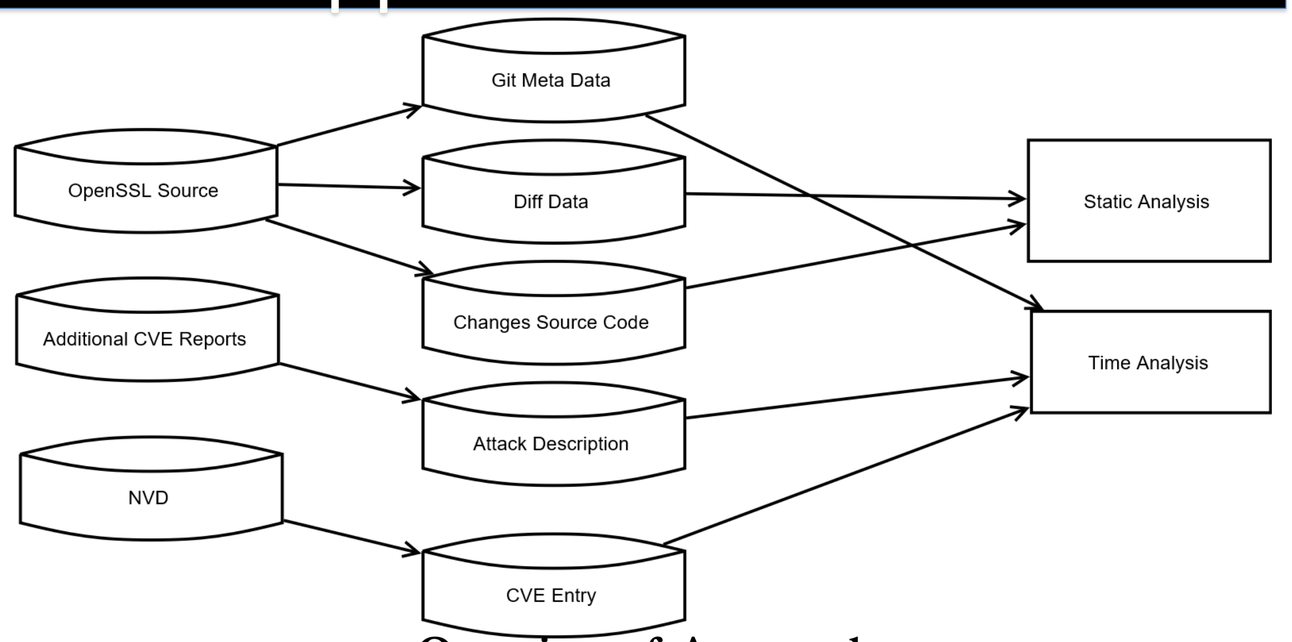
Diving Deeper in Vulnerability Causing Patterns and Reporting Practices using Static Analysis

Quentin Mayo, Dr. Tao Xie

Introduction

- It is hard to generalize and determine different patterns in coding practices that leads to major vulnerabilities in an crypto-tool. Individual vulnerabilities are often viewed as domain specific and mutual exclusive [1].
- Using OpenSSL , we were able to map 90 of the CVE reports to the corresponding GIT repository commits. We found that around 20% of all the listed vulnerabilities were related to DTLS. 66% of all attacks involved a denial of service (DOS) based attack and 42% of attacks had the risk of crashing the application. The majority of the other current attacks are memory related. 54% vulnerabilities repaired involved modification or the inclusion of explicit exception handling and subroutine exiting.

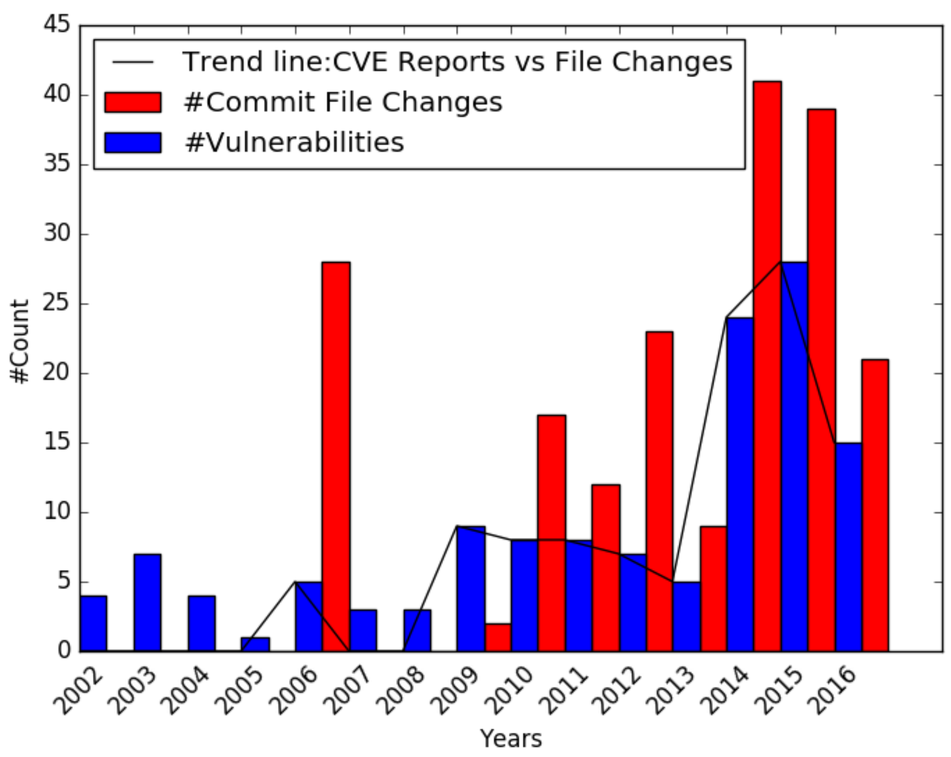
Approach Overview



Overview of Approach

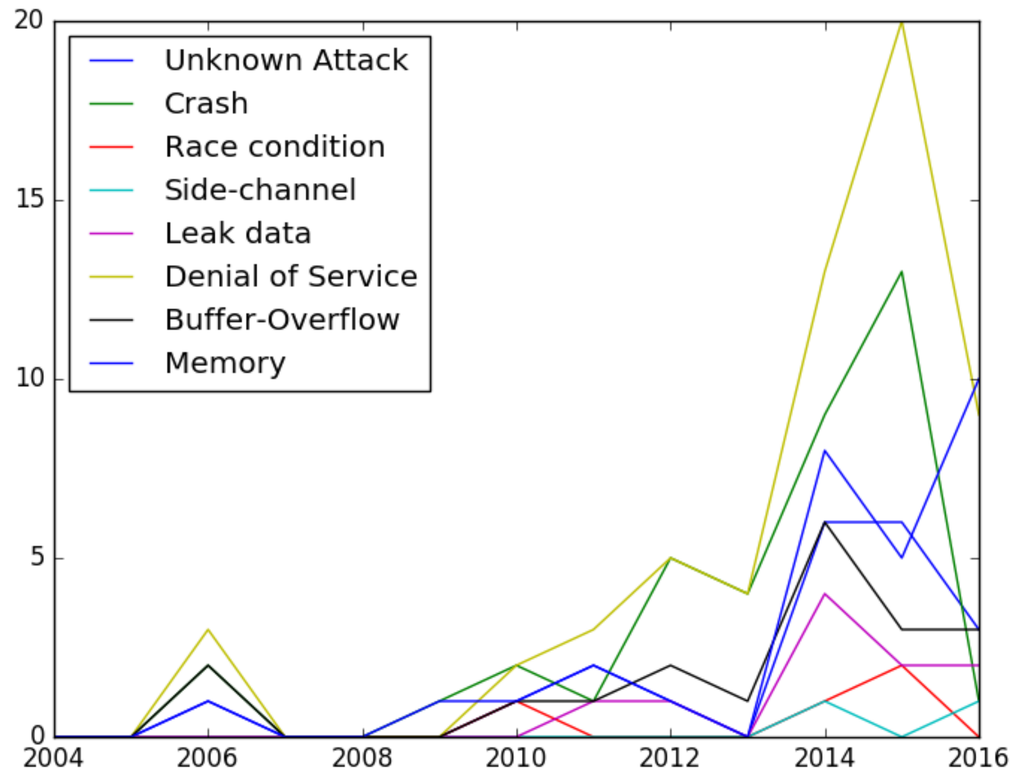
Findings

1. How fast are vulnerabilities repaired? *~2 days*
2. Who actually repairs the vulnerabilities? *~Internal Developers*
3. What are the characteristics of known OpenSSL vulnerabilities?
    1. *~ Some caused by code optimization*
    2. *~ Functions are often remove more while number of lines of code increases*
4. Does repairing vulnerabilities require major code overhaul?
    1. *~Minor code changes but large changes in the CFG*
5. Are vulnerabilities isolated or are they distributed throughout the code? *~Most isolated to a file or function*

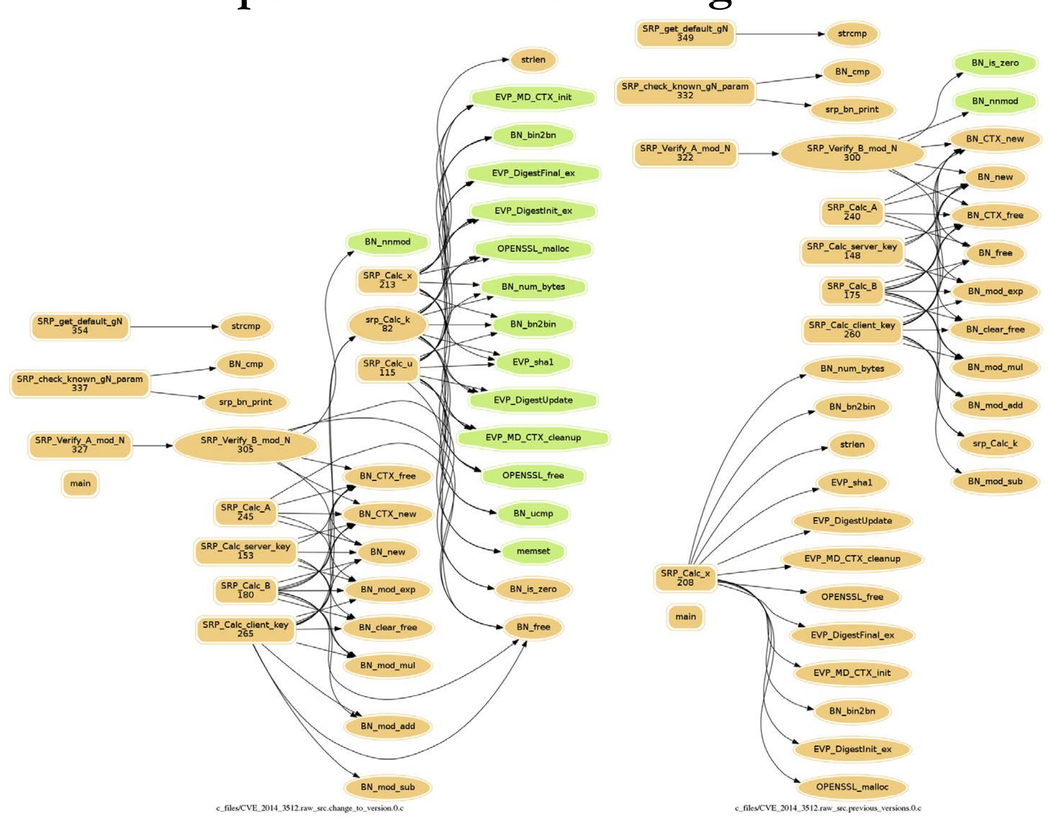


OpenSSL: Vulnerabilities throughout the years

Findings



OpenSSL: Attacks Through Time



CVE 2014 3512: CFG .C Commit Source Code Change

Conclusions

- Control flow graphs can give information on the vulnerabilities but they are better suited to be used as in part with additional static analysis such as statement maximum depth .
- Key characteristics
    - Most frequent attacks are DOS for OpenSSL.
    - Many of the vulnerabilities reviewed could also be can classified as bugs in the code.
    - The large percent of vulnerabilities seen in OpenSSL was based on exception handling and most of the vulnerabilities involved where contained in a single file.
    - Bit-wise operations, a basic operator, was the main area that was involved with vulnerabilities.

Future Work

- Compare the changes against none security related vulnerabilities.
- Use symbolic execution to see if we can predict the area in which the code is exploited.

References & Acknowledgements

1. Stephan Neuhaus, Thomas Zimmermann, Christian Holler, and Andreas Zeller. 2007. Predicting vulnerable software components. In Proceedings of the 14th ACM conference on Computer and communications security. ACM, 529–540.

"This material is based upon work supported by the Maryland Procurement Office under Contract No. H98230-14-C-0141."