Poster: What Proportion of Vulnerabilities can be Attributed to Ordinary Coding Errors?

Rick Kuhn[1], Mohammad Raunak[2], Raghu Kacker[1]
kuhn@nist.gov, raghu.kacker@nist.gov raunak@loyola.edu
[1]National Institute of Standards and Technology [2]Loyola University Maryland

I. Introduction

The analysis reported in this poster developed from questions that arose in discussions of the Reducing Software Vulnerabilities working group, sponsored by the White House Office of Science and Technology Policy in 2016 [1]. The key question we sought to address is the degree to which vulnerabilities arise from ordinary program errors, which may be detected in code reviews and functional testing, rather than post-release.

The analysis used 2008 - 2016 data from the US National Vulnerability Database (NVD) [2]. NVD is the US government's repository of information system security vulnerabilities, which compiles nearly all publicly reported vulnerabilities using the Common Vulnerabilities and Exposures (CVE) dictionary [3]. Each reported CVE is assigned to one or more categories called the Common Weakness Enumeration (CWE) [4], which specifies categories that may include a number of subsidiary weaknesses. For example, CWE-119, Buffer errors, includes 14 subsidiary CWEs, such as out of bounds read (CWE-125), and untrusted pointer dereference (CWE-822).

We further grouped the NVD CWE categories into primary classes of Configuration, Design, and Implementation errors. In determining the class of each CWE category, we considered the common errors in each type. *Configuration* vulnerabilities result when a system is not set up correctly with respect to security goals. A simple example would be failure to enable password checking. *Design* related vulnerabilities are those that originate in the planning and design of the system, such as selecting an outdated or weak cryptographic algorithm. *Implementation* errors occur in program construction. One of the most common implementation vulnerabilities is simple buffer overflow. Failure to check that input size is within maximum buffer size is a simple error that should almost never occur, but continues to be a widespread problem. A wide variety of implementation related vulnerabilities also result from failure to properly validate input.

HoTSoS '18, April 10–11, 2018, Raleigh, NC, USA © 2018 Copyright is held by the owner/author(s). ACM ISBN 978-1-4503-6455-3/18/04. https://doi.org/10.1145/3190619.3191686

II. Analysis and Results

The poster includes analysis of the following data [5]:

- Severity trends - proportion of vulnerabilities designated *low*, *medium*, and *high* by year.
- Primary CWE type trends - direction of trend for 19 primary CWE types, further classed as Configuration, Design, or Implementation vulnerabilities.

Significant findings include:

- The proportion of *high* severity vulnerabilities trends downward, declining about 15 percentage points since 2008. About two-thirds of this fraction has shifted to *medium* severity vulnerabilities.
- Implementation or coding errors account for roughly two thirds of the total. We consider the proportion of implementation vulnerabilities, rather than absolute numbers, because the number of vulnerabilities is partially a function of the number of applications released, which has increased over time. The proportion of implementation vulnerabilities for 2008-2016 is close to the 64% reported for 1998 - 2003 in an analysis of an early version of NVD [6].

The high proportion of implementation errors suggests that little progress has been made in reducing these vulnerabilities that result from simple mistakes, but also that more extensive use of static analysis tools, code reviews, and testing could lead to signficant improvement. The poster also briefly summarizes data on effectiveness of approaches to preventing and detecting errors before release.

Products may be identified in this document, but such identification does not imply recommendation by the NIST, nor that the products identified are necessarily the best available for the purpose.

[1] Black, P. E., Badger, M. L., Guttman, B., & Fong, E. N. (2016). *Dramatically Reducing Software Vulnerabilities: Report to the White House Office of Science and Technology Policy.* NIST Interagency Report, NISTIR-8151.
[2] National Vulnerability Database, http://nvd.nist.gov 2017
[3] Common Vulnerabilities and Exposures, https://cve.mitre.org.
[4] Common Weakness Enumeration, https://cwe.mitre.org.
[5] Kuhn, D. R., Raunak, M. S., & Kacker, R. (2017, July). An Analysis of Vulnerability Trends, 2008-2016. *Software Quality, Reliability and Security* (QRS-C), 2017 IEEE International Conference on (pp. 587-588).
[6] Heffley, Jon, and Pascal Meunier. "Can source code auditing software identify common vulnerabilities and be used to evaluate software security?" *System Sciences, 37th Annual Hawaii Intl Conf*, IEEE, 2004.