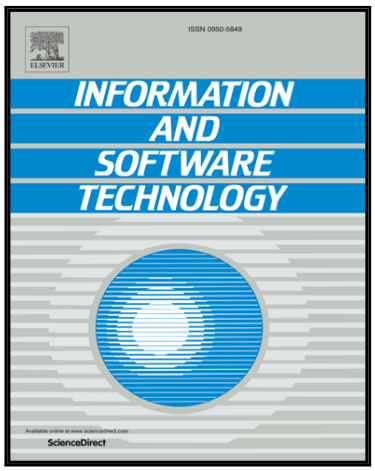

A Case Study on Software Vulnerability Coordination

Jukka Ruohonen, Sampsa Rauti, Sami Hyrynsalmi, Ville Leppänen

PII: S0950-5849(17)30511-6
DOI: 10.1016/j.infsof.2018.06.005
Reference: INFSOF 6003

To appear in: *Information and Software Technology*

Received date: 26 January 2018
Revised date: 10 June 2018
Accepted date: 16 June 2018

Please cite this article as: Jukka Ruohonen, Sampsa Rauti, Sami Hyrynsalmi, Ville Leppänen, A Case Study on Software Vulnerability Coordination, *Information and Software Technology* (2018), doi: 10.1016/j.infsof.2018.06.005

A Case Study on Software Vulnerability Coordination

Jukka Ruohonen[a,*], Sampsa Rauti[a], Sami Hyrynsalmi[a,b], Ville Leppänen[a]

[a] *Department of Future Technologies, University of Turku, FI-20014 Turun yliopisto, Finland*
[b] *Pori Department, Tampere University of Technology, P.O. Box 300, FI-28101 Pori, Finland*

Abstract

Context: Coordination is a fundamental tenet of software engineering. Coordination is required also for identifying discovered and disclosed software vulnerabilities with Common Vulnerabilities and Exposures (CVEs). Motivated by recent practical challenges, this paper examines the coordination of CVEs for open source projects through a public mailing list.

Objective: The paper observes the historical time delays between the assignment of CVEs on a mailing list and the later appearance of these in the National Vulnerability Database (NVD). Drawing from research on software engineering coordination, software vulnerabilities, and bug tracking, the delays are modeled through three dimensions: social networks and communication practices, tracking infrastructures, and the technical characteristics of the CVEs coordinated.

Method: Given a period between 2008 and 2016, a sample of over five thousand CVEs is used to model the delays with nearly fifty explanatory metrics. Regression analysis is used for the modeling.

Results: The results show that the CVE coordination delays are affected by different abstractions for noise and prerequisite constraints. These abstractions convey effects from the social network and infrastructure dimensions. Particularly strong effect sizes are observed for annual and monthly control metrics, a control metric for weekends, the degrees of the nodes in the CVE coordination networks, and the number of references given in NVD for the CVEs archived. Smaller but visible effects are present for metrics measuring the entropy of the emails exchanged, traces to bug tracking systems, and other related aspects. The empirical signals are weaker for the technical characteristics.

Conclusion: Software vulnerability and CVE coordination exhibit all typical traits of software engineering coordination in general. The coordination perspective elaborated and the case studied open new avenues for further empirical inquiries as well as practical improvements for the contemporary CVE coordination.

Keywords: vulnerability, open source, coordination, social network, CVE, CWE, CVSS, NVD, MITRE, NIST

1. Introduction

Software bugs have a life cycle.[1] In a relatively typical life cycle, a bug is first introduced during development with a version control system, then reported in a bug tracking system, and then again fixed in the version control system. Also software security bugs, or vulnerabilities, follow a similar life cycle. Unlike conventional bugs, however, vulnerabilities often require coordination between multiple parties. Coordination is visible also during the identification and archiving of vulnerabilities with unique CVEs.

There are four ways to obtain these universally recognized vulnerability identifiers. For obtaining a CVE, (a) an affiliation with an assignment authority (such as Mozilla or Microsoft) is required, but coordination may be done also by (b) contacting such an authority, making (c) a direct contact to the MITRE corporation, or (d) using alternative channels for public coordination [100]. During the period observed, the public channel referred to the `oss-security` mailing list. The typical workflow on the list resembled the simple communication pattern illustrated in Fig. 1.

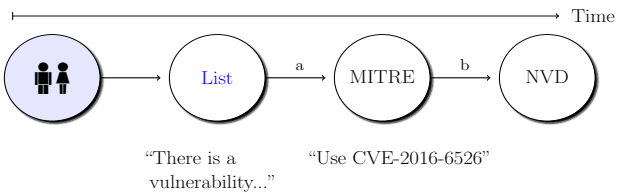

Figure 1: Coordination via `oss-security` (2008 – 2016)

*Corresponding author.
Email address: `juanruo@utu.fi` (Jukka Ruohonen)

[1] This paper is a rewritten and extended version of an earlier conference paper [95] presented at IWSM Mensura 2017.