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.
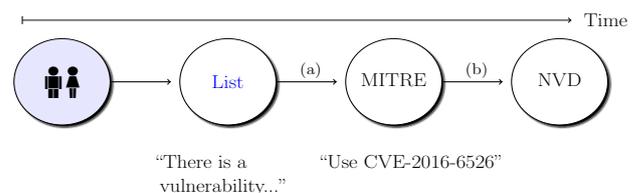

Figure 1: Coordination via `oss-security` (2008 – 2016)

---

*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.

Coordination is one of the major inhibitors of software development [38]. In the ideal case, the timeline from the leftmost circle would be short in terms of the timestamp that is recorded for a reported vulnerability to appear in NVD. In the future, robots might do the required coordination and evaluation work, but, recently, the time lags have allegedly been long. These delays have also fueled criticism about the whole tracking infrastructure maintained by MITRE and associated parties (for a recent media take see [55]). The alleged delays exemplify a basic characteristic of coordination: the presence of coordination often appears invisible to outsiders until coordination problems make it visible [62]. The recent criticism intervenes with other transformations. These include the continuing prevalence of so-called vulnerability markets [92], including the increasingly popular crowd-sourcing bug bounty platforms [52, 89]. Governmental interests would be another example [1]. The CVEs assigned are commonly used also to track exploitable vulnerabilities on criminal underground platforms [4]. A further example would be the use of social media and the resulting information leakages of the sensitive information coordinated [55, 97, 104]. These and other factors have increased the volume of discovered vulnerabilities for which CVEs are requested.

Reflecting these challenges, the coordination of CVEs through the mailing list ended in February 2017. To some extent, it is reasonable to conclude that `oss-security` lost its appeal as an efficient coordination and communication medium for CVE identifiers. To answer to the challenges, new options were provided for public CVE tracking [79], and larger reforms were implemented at MITRE.

Motivated by these practical challenges, the paper examines the CVE coordination through the `oss-security` mailing list between 2008 and 2016. The paper continues the earlier case study [95] by explicitly focusing on the timeline (b) in Fig. 1. In other words, the present paper observes the time between CVE assignments on the mailing list and the later publication of these identifiers in NVD. Although the previous study revealed interesting aspects about social networks for open source CVE coordination, it also hinted that the coordination delays might be difficult to model and predict. Modeling of the delays is the goal of the present paper. To empirically model the delays, nearly fifty metrics are derived in this paper for proxying three dimensions of coordination: (i) social networks and communication practices, (ii) tracking infrastructures within which the vulnerabilities had often already appeared prior to `oss-security`, and (iii) the technical characteristics of the vulnerabilities coordinated. Before elaborating these dimensions in detail, the opening Section 2 discusses the background and introduces the case studied. The research approach used to study the case is described in Section 3. Results are presented in Section 4 and further discussed in Section 5.

2. Background

Software engineering coordination is a fundamental part of the grand theories in the discipline [38]. The following discussion will briefly motivate the general background by considering some of the basic coordination themes in terms of software vulnerabilities and their coordination. After this motivation, the case studied is elaborated from a theoretical point of view. The discussion ends to a formulation of three research questions for the empirical analysis.

2.1. Vulnerability Coordination

Coordination can be defined as a phenomenon that originates from *dependencies* between different *activities*, and as a way to *manage* these dependencies between the activities [36]. Software engineering is team work. As team work implies social dependencies between engineers, there is always some amount of coordination in all software engineering projects involving two or more engineers. There are also technical dependencies in all software products. Early work on software engineering coordination focused on the reduction of such technical dependencies in order to improve task allocation and work parallelism [14]. However, not all technical dependencies can be eliminated. For this reason, later work has often adopted a *socio-technical* perspective for studying software engineering coordination.

The augmentation of technical characteristics with social tenets is visible at a number of different fronts. One relates to the research questions examined. Examples include synchronization of work and milestones, incremental integration of activities, arrangement of globally distributed teams, frequent deliveries, and related project management processes [53, 63, 81]. Another relates to the research methodologies used. While interviews and surveys are still often used [14, 53, 81], techniques such as social network analysis have become increasingly common for examining software engineering coordination [12, 101]. Despite of these changes, one fundamental theoretical premise has remained more or less constant over the years.

Coordination requires *communication*, and coordination failures are often due to communication problems [14, 115]. The consequences from coordination failures vary. Typical examples include schedule slips, duplication of work, build failures, and software bugs [12, 14]. Communication obstacles, coordination failures, and the resulting problems are well-known also in the security domain.

A good example would be incident and abuse notifications for vulnerable Internet domains: it is notoriously difficult to communicate the security issues discovered to the owners or maintainers of such vulnerable domains [16]. Analogous problems have been prevalent in *vulnerability disclosure*, which refers to the practices and processes via which the vulnerability discoverers make their discoveries known to the vendors whose products are affected either directly or via a third-party coordinator. Although there have long been recommendations and guidelines [20], it is still today often difficult to communicate vulnerability

discoveries to vendors [85]. A substantial amount of effort may be devoted to coordinate the disclosure and remediation of high-profile vulnerabilities, but difficulties often occur for more mundane but no less important vulnerabilities. Some vendors are reluctant to participate in vulnerability disclosure—and some vendors even avoid patching their software products altogether. Even legal threats are still today not unheard of. While vulnerability disclosure is a prime example of coordination (failures) in the software security context, it has only a narrow scope.

As Steven M. Christey—the primary architect behind the current CVE tracking—has argued, the term *vulnerability coordination* is preferable because vulnerability disclosure only captures a small portion of the activities required to handle and archive vulnerabilities [19]. In fact, vulnerability coordination is a relatively frequent activity among the integrative software engineering activities carried out by open source developers [2]. Bug reports must be handled in a timely manner for security issues. Evaluation is needed to assess the versioned products affected. Of course, also fixes must be written, but fixes may further require careful backtracking within version control systems, debugging, reviews from peers, testing done by peers and users alike, and coordination between business partners or third-party open source software projects. Then, erratas, security advisories, or other notes must be written, identifiers must be allocated to uniquely track the notes and to map these to other identifiers, preparations may be required for answering to user feedback, and so forth. These are all examples of the software engineering activities that are typically present in the software vulnerability context.

For empirical software engineering research, the identifiers used to track vulnerabilities are particularly relevant. The most important identifier is CVE, in terms of both research and practice. For open source projects, having a single canonical identifier helps developers and users to coordinate their efforts. Given the inconsistency issues affecting open bug trackers [86, 108], CVEs help at addressing questions such as: "well it sounds like this one but maybe it's that other one?" [100]. For a security professional, CVEs are ubiquitous entries in a curriculum vitae. For research, CVEs provide the starting point for connecting the distinct engineering activities into a coherent schema. While there is an abundant amount of existing research operating with CVE-based research schemas, thus far, only one attempt [95] has been made to better understand the primary schema; software and security engineering coordination is required for CVEs themselves.

2.2. CVE Coordination Through a Mailing List

Software development teams tend to produce software products that reflect the communication structure of the teams, to paraphrase the famous law formulated by Melvin E. Conway [23]. Although the law is not directly applicable to the context of CVE coordination, the `oss-security` case supports a weaker variant of the same basic assertion.

This variant might be defined as a statement that a communication structure tends to reflect the *type* of software engineering work and the *medium* used for communication.

The type of work done and the medium both characterize also the social network structure of the CVE coordination through the mailing list. Both limit also the applicability of common theoretical interpretations. For instance, knowledge sharing is a typical way to motivate research on open source social networks [50, 57, 109], but sharing of *knowledge* was not the main purpose of the `oss-security` mailing list during the period studied.

The primary purpose of the list was to coordinate the assignment of CVE identifiers for discovered and usually already disclosed software vulnerabilities. This coordination was done by explicitly requesting CVE identifiers, which were then assigned by MITRE affiliates based on a brief evaluation. In terms of communication practices, this coordination practice often culminated in short replies, such as "use CVE-2016-6526" [78], made by MITRE affiliates to previously posted CVE requests. Participants typically kept MITRE's `cve-assign@mitre.org` email address in the carbon copy field when they posted a request, further using a subject line that identified the message as a request. While longer discussions were not unheard of, these communication practices indicate that knowledge sharing has only limited appeal for framing the case theoretically. Exchanging information about abstract identifiers does not necessarily imply thorough discussions about the technical details of the vulnerabilities coordinated. In other words, data does not equate to information, and information does not equal knowledge.

The coordination work through the list produced clear core social network components [95]. This observation is classical in the open source context [22, 24, 57, 110], but the contextual interpretation is still relevant. Due to the way CVEs are assigned, the cores centered to MITRE affiliates. By using a network representation described later in detail, the core social network components are illustrated in Fig. 2. Excluding the use of MITRE's email address without an explicit sender, there are two identifiable individual participants with degrees higher than 800, meaning that these two participants both coordinated at least eight hundred CVEs each. Both participants are MITRE affiliates. What is more, there are only a few CVEs that link the three core participants to each other. This observation indicates that task allocation and related coordination techniques apply also to the case studied.

These core social network components are a good example of so-called communication brokers who help at resolving coordination obstacles in software engineering [116]. Such brokers integrate information from multiple sources. This *integrative* engineering work often leads to a social network structure with relatively high level of centrality, low level of clustering, and star-like centers within which the integrative work is located [82]. This theoretical characterization applies well also to the social networks for the CVE coordination through the mailing list.
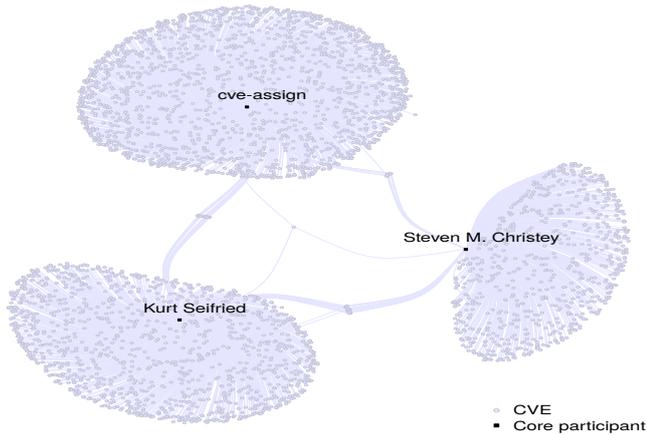
Figure 2: The Core Participants (one hop from the labeled vertices)

Thus, the type of software engineering work carried out is an important factor characterizing the corresponding social network structures. In terms of development mailing lists, non-core participants may post as many messages as core participants [25], but this observation does not generalize to all cases [34], including the case studied. Although coordination requires communication, coordination tends to result in different network structures than communication required for other work (or leisure) activities. A communication medium used for coordination presumably also affects the emerging network structure. For instance, social networks constructed from bug tracking systems indicate that many individuals may report bugs even though the actual development may be concentrated to a small core group [120]. A communication medium also places restrictions over who can participate, how much can be communicated, and what is communicated [12, 84]. While `oss-security` is open for anyone to participate, the communication volume and the content of messages exchanged are both important for further elaborating the case.

The integrative coordination work done on the list reflected not only social but also loose technical dependencies between different tracking infrastructures. The classical consumer-producer abstraction for software engineering coordination [62, 63] is useful for framing these technical dependencies theoretically. According to this abstraction, there exists a *prerequisite constraint*: the work activity of a producer must be usually completed before work can start on the consumer side [36, 62]. In terms of the case studied, the participants (producers) who requested CVEs provided sufficient technical information to justify the requests for the MITRE affiliates (consumers). If the information was sufficient for a request, the prerequisite constraint was satisfied, and the vulnerability in question later appeared in NVD with the CVE requested. Instead of in-depth knowledge sharing, the sufficient information was usually delivered via hyperlinks to other open source tracking infrastructures within which a given vulnerability had already been discussed. Consequently, the discovery,

disclosure, and patching of the vulnerabilities coordinated had almost always occurred before information appeared on the mailing list.

2.3. Research Questions

The preceding discussion motivates three questions worth asking about the delays between CVE requests on the mailing list and the later appearance of these identifiers in the central tracking database. Given the medium used and the type of software engineering coordination done, these questions can be framed by separating the two terms in the concept of socio-technical coordination. The first question addresses the social dimension:

RQ$_1$ *Have the CVE coordination delays been affected by the social networks and communication practices between the participants on the `oss-security` mailing list?*

The remaining two questions address the technical dimension in socio-technical coordination. Given the integrative work done and the consumer-producer abstraction, it is worth asking the following question about the delays:

RQ$_2$ *Did traces to other tracking infrastructures affect the coordination delays during the period observed?*

The third and final research question approaches the technical dimension from a more direct perspective:

RQ$_3$ *Were the coordination delays affected by the technical characteristics of the vulnerabilities coordinated?*

The analytical meaning behind the three questions is illustrated in Fig. 3. It should be noted that causal inference is not attempted; hence, no arrows are drawn in the figure between the three explanatory dimensions.

3. Approach

In what follows, the research approach taken to study the CVE coordination delays is elaborated by introducing the empirical dataset, the operationalization of the delays, and the construction of the social networks observed. After this machinery has been installed, the approach is further elaborated by describing the explanatory metrics and the statistical methodology used to model the delays.

3.1. Data

The dataset is compiled from two sources. All email messages posted on `oss-security` were obtained from the online Openwall archive [77]. These messages are cross-referenced with CVE identifiers to the second data source, NVD [74]. The sampling period runs from the first welcome message in February 2008 to the emails posted in December 31 2016. This sampling interval corresponds with the historical period during which the mailing list was used for CVE assignments.
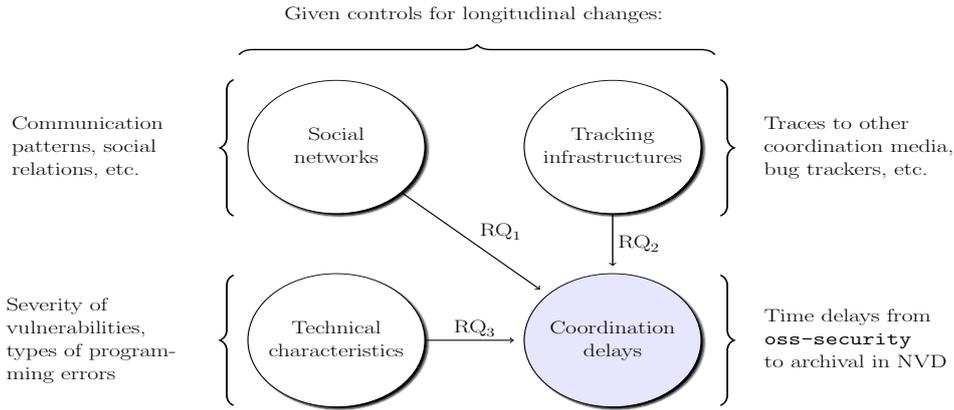
4

Figure 3: Hypothetical Relations and the Corresponding Research Questions

There were a little over sixteen thousand messages posted during this time interval. For each message delivered in the hypertext markup language (HTML) format, a twofold routine is used for pre-processing the archival data (see Fig. 4). The first pre-processing subroutine extracts the actual messages from the HTML markup, further omitting all lines referring to quotations from previous messages posted on the mailing list. To some extent, also forwarded messages are excluded based on simple but previously used heuristics [118]. Due to the markup language, this pre-processing routine is likely to yield some inconsistencies. However, the consequences for the empirical analysis should be relatively small because no attempts are made to pre-process email threads. This choice is largely imposed by the use of the online archive. In particular, the fields `Message-ID`, `In-Reply-To`, and `References` are not delivered via the online interface, which prevents the use of common (meta-data) techniques for parsing email threads [34, 112]. Although content-based alternatives exist for reconstructing threaded email structures [28, 98], only the senders of emails are considered in order to improve data quality and to simplify the data processing.
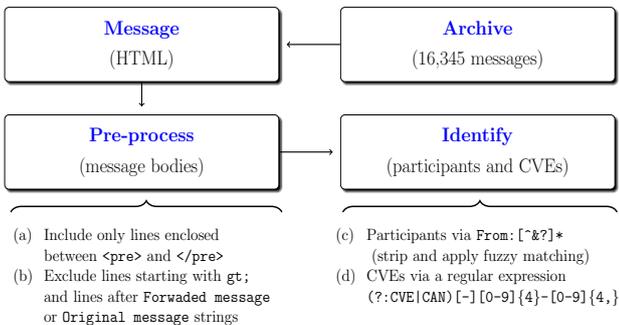
The second subroutine is used for identification. The `From` email header field is used for identifying the individual participants by using the first match from a simple Python regular expression. Although the matching itself is simple, it should be emphasized that all senders are identified according to their names rather than their addresses. This option is often preferred because individuals' email addresses tend to vary [12, 73, 109]. The choice is again also imposed by the online archive, which obfuscates the email addresses for spam prevention and related reasons. Consequently, additional pre-processing techniques [71] for mapping identified names to addresses cannot be used.

Furthermore, Levenshtein's [54] classical distance metric between two names, say $L(x, y)$, is a decent choice for accounting small inconsistencies [122]. If s_1 and s_2 denote two full names with lengths l_1 and l_2, a similarity score is computed through $\delta = 1.0 - L(s_1, s_2)/\max\{l_1, l_2\}$ [18]. If the scalar δ then exceeds a threshold value 0.8, two names are taken to refer to the same individual. Although the threshold is subjective, it captures some typical cases, such as "John Doe" who occasionally writes his name as "John, Doe" when sending emails. The few outlying cases that exceeded the $\delta = 0.8$ threshold were further evaluated manually. This additional check revealed no obvious approximation mistakes. In addition, "Christey, Steven M." and "Steven M. Christey" were merged manually.

Also CVEs are identified with a regular expression, which is a typical approach for searching vulnerabilities from heterogeneous sources [4, 58]. The expression takes into account both the old and deprecated candidate (CAN) syntax as well as the recently made syntax change that allows an arbitrary amount of digits in the second part of CVE identifiers [66, 67]. Even though forwarded messages and direct quotations to previous emails are approximately excluded, it should be noted that there can be many-to-many relations between messages and CVEs. For instance, a lengthy security advisory posted on the list may contain a large amount of individual CVE-referenced vulnerabilities.

Figure 4: Pre-processing in a Nutshell

Such cases are included in the sample. Finally, only those CVEs are qualified that have also valid entries in NVD. Most of the invalid entries in the database refer to identifiers that were assigned but which were later rejected from inclusion to the database. Given that these disqualified CVE assignments cause different biases [19], all rejected identifiers were excluded from the empirical sample. The matching of these invalid entries was done by searching for the string REJECT in the summary field provided by NVD.

3.2. Coordination Delays

There are multiple ways to measure the efficiency of software engineering coordination [53]. In the context of software bugs in general, a good example would be the validity of bug reports. Bug tracking infrastructures typically reserve multiple categories for classifying invalid reports. These categories include classes for already fixed bugs, irreproducible bugs, and duplicate reports, among other groups. If a large amount of bug reports end up into such classes, coordination would be generally inefficient.

As coordination deals with dependencies, it is no surprise that also social network structures of bug reporters have been observed to affect the probability of valid bug reports [120]. Although the generalizability of this observation seems limited [10], there are good reasons to suspect that an analogous effect is present in the software vulnerability context. Because attribution is particularly important for vulnerabilities and monetary compensations are relatively common, it is no wonder that invalid—or even outright fake—reports have been a typical menace affecting vulnerability coordination [19, 52]. For this reason, vulnerability reports from established discoverers likely increase the validity of the reports as well as the trust placed on the reports. This trust provision is also reflected on the CVE assignment authorities granted for a few individual security researchers.

Due to the sensitiveness of the information coordinated, the availability of open data is limited about the internal vulnerability tracking systems used by software vendors, MITRE, and other actors. Many open source projects also limit the visibility of security bugs, or otherwise try to constrain the exposure of public information.

These data limitations have affected also the ways to quantify longitudinal vulnerability information. For instance, the efficiency of vulnerability disclosure can be approximately measured with a time difference between a disclosure notification sent to a software vendor and the vendor's reply [7, 90]. Analogously, patch release delays can be measured by fixing the other endpoint to the dates on which vendors released patches for the vulnerabilities disclosed to the vendors [64, 107]. Similar delays can be measured also in terms of initial bug reports and later CVE assignments based on the reports [113]. Further examples include the timing of security advisories [93], the dates on which signatures were added to intrusion detection and related systems [11], and exploit release dates for known

vulnerabilities [1, 4, 15]. All these different dates convey different viewpoints on the coordination of vulnerabilities.

In this paper, analogously, the empirical interest relates to the following per-CVE time differences (in days):

$$y_i = T_{\text{NVD}_i} - T_{\text{oss-security}_i}, \qquad (1)$$

given

$$T_{\text{NVD}_i} \geq T_{\text{oss-security}_i} \quad \text{for all} \quad i = 1, \ldots, n \qquad (2)$$

vulnerabilities observed. The timestamp T_{NVD_i} records the date on which the i:th CVE was first stored to NVD. The second timestamp $T_{\text{oss-security}_i}$ refers to the *earliest* date on which this CVE was posted on the mailing list. The restriction in (2) excludes already archived CVEs that were later discussed on the mailing list. Thus, the integer $y_i \geq 0$ approximates the length of the path (b) in Fig. 1. In theory, also the path (a) in Fig. 1 could be measured, but the data from the online archive makes it difficult to identify the initial CVE requests. This limitation does not affect the validity of the delay metric, but it does affect the interpretation given for it.

The metric in (1) approximates the time delays between the initial CVE assignments done through the oss-security mailing list and the usually later appearance of the requested CVEs in NVD. Therefore, y_i focuses on the *internal* coordination done by MITRE affiliates, the NVD team, and other actors involved in the coordination. One way to think about this internal coordination is to consider the delay metric as an *indirect* quantity for measuring how fast work items in a *backlog* or an *inventory* are transferred to complete deliveries [62, 88]. Due to data limitations, however, it is currently neither possible to observe the internal within-MITRE (or within-NVD) coordination directly nor to explicitly measure the work items in the CVE backlog. Therefore, the delay metric used provides a sensible but not entirely reliable way to approximate the typical coordination delays that affected one particular CVE coordination channel.

3.3. Bipartite Email and Infrastructure Networks

The socio-technical coordination and communication characteristics are proxied by observing so-called task-based [116] or person-task [101] social networks. Thus, the underlying social network structure is bipartite, meaning that there are two types of vertices. An edge connecting any two vertices always contains both types; there are no edges that would connect a vertex of one type to a vertex of the same type. In addition, the network structure is unweighted and undirected. More formally:

$$G_s = (P \cup A, E), \qquad (3)$$

where G_s denotes an observed social network constructed from the messages that were posted on the mailing list, from the first message posted in 2008 to the last message in 2016. The two disjoint vertex sets P and A refer to

participants and CVE identifiers, respectively. If a participant $p \in P$ has sent a message containing a CVE identifier $a \in A$, an undirected edge, $(p, a) = (a, p)$, is present in the edge set E. Therefore, individual participants are linked together through CVE identifiers (see Fig. 5). Due to the bipartite structure, it holds that $P \cap A = \emptyset$ and

$$|E| = \sum_{p \in P} \mathrm{Deg}(p) = \sum_{a \in A} \mathrm{Deg}(a), \qquad (4)$$

where

$$\mathrm{Deg}(x) = \sum_{(x,y) \in E} y. \qquad (5)$$

Furthermore, another network is constructed from the uniform resource locators (URLs) embedded in the hyperlinks present in the emails that contained also CVEs. The structure is again undirected, unweighted, and bipartite:

$$G_d = (D \cup B, F), \qquad (6)$$

where D denotes a set of domain names extracted from the URLs sent by participants in P, B denotes another set of CVEs, and F is an edge set containing edges from the domain names to the CVEs. If a participant $p \in P$ sent an email containing a $b \in B$ and a domain name $d \in D$ extracted from a URL in a hyperlink, an undirected edge $(b, d) = (d, b)$ is present in the set F. Thus, CVEs are linked also to domain names through the participants who posted messages containing both CVE identifiers and hyperlinks. Therefore, $B \subseteq A$ and $n = |A| \geq |B|$ because the subset of CVE identifiers stored to B are required to also have mappings to domain names.
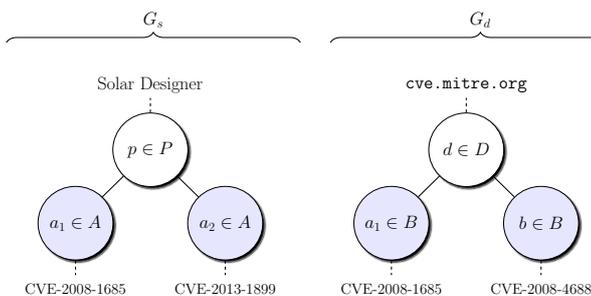

Figure 5: Example Networks

The network G_s is a *social* network in the traditional sense; even though participants are linked to each other through abstract identifiers, the participants are still *human* beings. The network G_d, in contrast, resembles more the so-called domain name system graphs within which domain names are connected to each other via Internet protocol (IP) addresses or by other technical relations [94, 103]. Consequently, it would be possible to manipulate G_d by resolving the addresses of the domain names or by considering only the second-level domain names [96]. Given the historical context, however, no attempts are made to

resolve the domain names, many of which are nonexistent today. Instead, only three semantic validation checks are enforced: (a) the length of each $d \in D$ is asserted to be at least three characters; (b) each d is required to contain a dot character; (c) and no entry in D is allowed to refer to a semantically valid IPv4 address.

3.4. Explanatory Metrics

The three research questions are evaluated by regressing the coordination delays $y_1, \ldots, y_n$ against the metrics enumerated in Table 1. Six *models* (M) are used for the statistical computations; the integer k denotes the cumulative number of metrics included in the consecutively estimated models, including the intercept. The table shows also the scale of the metrics; there are a few *continuous* ($\mathcal{C}$) metrics but most of the metrics are *dichotomous* ($\mathcal{D}$) dummy variables. The metrics enumerated can be further elaborated according to the models within which these first appear.

3.4.1. Control Metrics

Temporal aggregation of social network data should be done only after a careful consideration [73]. Previous work in the `oss-security` context also indicates that the social networks for the open source CVE coordination changed over the years [95]. In contrast to what has been claimed to characterize open source projects [71], the coordination effort did not diminish over time. In fact, the list became more popular, which resulted in more participants and more coordinated CVEs. These transformations also changed the social network structure, although the core of the network structure remained centered to MITRE affiliates. A notable change occurred also in the structure of this network core: Kurt Seifried from Red Hat joined the CVE editorial board [68] and took an active role also on `oss-security`. This activity reduced the reliance on a single MITRE affiliate for the CVE coordination through the list, resulting in the network cores illustrated in Fig. 2.

Instead of explicitly modeling these changes through separate annual social networks—as has been typical in applied social network research [90, 95, 110, 121], the longitudinal dimension is approximately controlled with eight dummy variables. Each of these is zero for the i:th identifier unless the CVE identifier was assigned on a given year between 2009 and 2016 according to the corresponding $T_{\mathrm{oss\text{-}security}_i}$ timestamp used in (1). The initial year 2008 acts as the reference category against which the effects of these annual dummy variables are compared against.

Given the fairly complex time series dynamics arising from the archiving of vulnerabilities to NVD and related tracking infrastructures [37, 106], a further set of eleven dummy variables is included for controlling potential monthly variation in the coordination delays. The $T_{\mathrm{oss\text{-}security}_1}, \ldots, T_{\mathrm{oss\text{-}security}_n}$ timestamps are again used for computation, and January acts as the reference month.

The third and final longitudinal control metric is named WEEKEND. It scores a value one for a CVE posted to

Table 1: Explanatory Metrics

RQ$_i$	M$_j$	Name	Scale	Description
–	M$_1$	2009, ..., 2016	$\mathcal{D}$	True for CVEs assigned on a given year according to $T_{\texttt{oss-security}}$.
	($k = 21$)	Feb, ..., Dec	$\mathcal{D}$	True for CVEs assigned on a given month according to $T_{\texttt{oss-security}}$.
		WEEKEND	$\mathcal{D}$	True for CVEs assigned on Saturday or Sunday according to $T_{\texttt{oss-security}}$.
RQ$_1$	M$_2$	SOCDEG	$\mathcal{C}$	The degree of all CVE-labeled vertices in G_s.
	($k = 25$)	MITREDEV	$\mathcal{D}$	True for CVEs in the neighborhood of the three labeled vertices in Fig. 2.
		MSGSLEN	$\mathcal{C}$	The amount of characters divided by 100 in the emails mentioning a CVE.
		MSGSENT	$\mathcal{C}$	For a given CVE, the Shannon entropy of the emails mentioning the CVE.
RQ$_2$	M$_3$	INFDEG	$\mathcal{C}$	The degree of CVE-labeled vertices in G_d (zero for any $b \in B$ but $b \notin A$).
	($k = 31$)	NVDREFS	$\mathcal{C}$	Number of reference URLs given in NVD for the CVEs observed.
		VULNINF	$\mathcal{D}$	True for CVEs linked to vulnerability infrastructures via G_d.
		BUGS	$\mathcal{D}$	True for CVEs linked to bug tracking and related systems via G_d.
		REPOS	$\mathcal{D}$	True for CVEs linked to version control and related systems via G_d.
		SUPPORT	$\mathcal{D}$	True for CVEs linked to vendors' support channels via G_d.
RQ$_3$	M$_4$	IMPC	$\mathcal{D}$	True for CVEs having a partial or a complete impact on confidentiality
	($k = 34$)	IMPI	$\mathcal{D}$	True for CVEs having a partial or a complete impact on integrity
		IMPA	$\mathcal{D}$	True for CVEs having a partial or a complete impact on availability
	M$_5$	EXPNET	$\mathcal{D}$	True for CVEs that may be exploited only with a network access.
	($k = 37$)	EXPCPLX	$\mathcal{D}$	True for CVEs with a high or a medium access complexity for exploitation.
		EXPAUTH	$\mathcal{D}$	True for CVEs that can be exploited only through authentication.
	M$_6$	CWE-264	$\mathcal{D}$	True for CVEs in the domain of permissions, privileges, and access controls.
	($k = 47$)	CWE-119	$\mathcal{D}$	True for CVEs in the domain of buffer-related bugs.
		CWE-79	$\mathcal{D}$	True for CVEs in the domain of cross-site scripting (XSS).
		CWE-20	$\mathcal{D}$	True for CVEs in the domain of input validation.
		CWE-200	$\mathcal{D}$	True for CVEs in the domain of information leaks.
		CWE-399	$\mathcal{D}$	True for CVEs in the domain of resource management bugs.
		CWE-189	$\mathcal{D}$	True for CVEs in the domain of numeric bugs.
		CWE-352	$\mathcal{D}$	True for CVEs in the domain of cross-site request forgery (CSRF).
		CWE-89	$\mathcal{D}$	True for CVEs in the domain of structured query language (SQL) injection.
		CWE-310	$\mathcal{D}$	True for CVEs in the domain of cryptographic bugs.

the mailing list on Saturday or Sunday according to the coordinated universal time, taking a value zero otherwise. The rationale relates to observations that the days of week may affect the likelihood of introducing bugs during software development [102]. Although the empirical evidence is mixed regarding this assertion [29], it is reasonable to extend it toward vulnerability coordination. For instance: if the i:th requested CVE would have otherwise ended up to NVD rapidly after two days, it may be that an additional delay, say $\epsilon > 0$, is present in case the request was posted on a weekend, such that $y_i = 2 + \epsilon$. The same rationale applies to the monthly effects. In other words, annual holidays presumably taken by the MITRE affiliates and others participants may well affect the coordination delays.

3.4.2. Social Network and Communication Metrics

Four metrics are used for soliciting an answer to RQ$_1$. The first is the amount of participants linked to CVEs:

$$\text{SOCDEG} = [\text{Deg}(a_1 \in A), \ldots, \text{Deg}(a_n \in A)], \quad (7)$$

where $n = |A|$ and the set A is assumed to be ordered. In other words, the metric equals the degree of the CVE-labeled vertices in G_s. This degree centrality conveys a clear theoretical rationale. In the software engineering context this rationale relates to the saying "too many cooks spoil the broth". The essence behind the saying is that increasing number of participants increases the coordination requirements, which translate into delays in completing software engineering tasks [13, 38]. There exists also some evidence for an assertion that bug resolution delays increase with increasing number of participants in the resolution processes [10]. Although the existing empirical evidence seems weak, the same dictum can be extended to a further hypothesis that "too many developers" increase the probability of introducing vulnerabilities during software development [65]. Given analogous reasoning, SOCDEG can be expected to lengthen the coordination delays. If a given $a \in A$ has a high degree, meaning that many participants posted messages containing the CVE,

it may be that the vulnerability in question was particularly interesting or controversial. Either way, a longer delay could be expected for such a vulnerability.

In theory, also other vertex-specific centrality metrics could be used for modeling the delays. There are a couple of reasons to avoid additional centrality metrics, however. The first reason relates to interpretation ambiguities. For instance, the so-called closeness centrality is often used for quantifying information flows among a group of human participants [8]. In the context of sender-receiver type of email networks [59, 105], this quantification rests on the assumption that a reply to an email constitutes an information flow. In reality, however, the lack of a reply does not imply the absence of an information flow; a participant may read an email without replying [73]. In addition to these theoretical limitations, the bipartite structure of G_s makes the interpretation of many vertex centrality metrics challenging. For instance, the so-called betweenness centrality is often interpreted to reflect communication gatekeepers and information brokers [13, 84], but it is difficult to theorize how a CVE identifier would be a gatekeeper. The second, more practical reason stems from multicollinearity issues induced by the inclusion of additional centrality metrics. As is typical [96, 99], many of the centrality metrics are correlated. In particular, SOCDEG is highly correlated with the betweenness centrality values.

Instead of explicitly computed centrality, the second social network metric takes a simpler approach to quantify the concept of core developers in the open source context. The definitions for such developers differ. For instance, some authors have identified core developers with a cutoff point for vertex degrees [57, 110], while others have relied on documents about developer responsibilities and commit accesses [21, 25]. In the present context the relevant social network core is composed by the MITRE affiliates. Thus, MITREDEV is a dummy variable that scores one for all CVEs linked to the three labeled participants in Fig. 2.

The two remaining metrics are not explicitly related to social networks *per se*, although both of these still proxy communication practices. Namely: the metric MSGSLEN counts the length of strings in all emails posted with a given CVE identifier divided by one hundred, while the metric MSGSENT records the Shannon entropy of these emails. Both metrics have been used previously in the software engineering context [8]. The effect of both metrics upon y_i can be also expected to be positive. Given the rationale of the mailing list for coordinating CVE identifiers, lengthy email exchanges and increasing entropy are both likely to increase the coordination delays. In other words, high values for these two metrics both run counter to the short "use CVE-2016-6527" [78] communication patterns preferred on the list during the period observed.

3.4.3. Infrastructure Metrics

Six metrics are used for soliciting an answer to RQ_2. The first of these, INFDEG, is defined analogous to (7) but by using the vertex set B present in the network G_d.

Thus, this metric counts the number of semantically valid domain names in the adjacency of the CVE identifiers observed. The analytical meaning is similar to the number of hyperlinks in reports posted within bug tracking systems, which have been hypothesized to reflect bugs that are particularly difficult to remedy [8]. By translating the same hypothesis to the vulnerability coordination context, the number of domain names extracted from the hyperlinks could be expected to increase the coordination delays. Accordingly, a CVE that accumulates many hyperlinks may correspond with a vulnerability that is particularly difficult to interpret. Another explanation may be that the signal of relevant information is lost to the noise of numerous hyperlinks. However, the effect of INFDEG could be alternatively speculated to shorten the delays. The rationale for this alternative speculation relates to the prerequisite constraints in typical software engineering coordination.

A requested CVE identifier is likely to end up in NVD in case it is backed by sufficient and valid technical information. In addition to INFDEG, these prerequisite constraints can be also retrospectively approximated through the references provided in NVD to the primary information sources. Like the historical [15] and contemporary [87] databases, NVD maintains a list of reference sources that is frequently polled for new vulnerability information [19]. If a vulnerability appears in multiple sources, it is probable that a CVE for the vulnerability appears rapidly in NVD due to information gains about the technical details. Thus, the metric NVDREFS counts the number of NVD's reference sources for all CVEs observed. Even when a CVE was coordinated through `oss-security`, shorter delays can be expected for an identifier with multiple alternative sources for confirming the corresponding vulnerability.

Table 2: Infrastructure Regular Expressions

Metric	Regular expression for all $d \in D$
VULNINF	`cert.`, `exploit-db.com`, `first.org`, `mitre.org`, `nist.gov`, `osvdb.org`
BUGS	`bugs.`, `bugzilla.`, `gnats.`, `issues.`, `jira.`, `redmine.`, `trac.`, `tracker.`
REPOS	`code.`, `cvs.`, `cvsweb.`, `download.`, `downloads.`, `ftp.`, `git.`, `gitweb.`, `hg.`, `packages.`, `svn.`, `webcvs.`, `websvn.`
SUPPORT	`blog.`, `blogs.`, `dev.`, `doc.`, `docs.`, `forum.`, `forums.`, `help.`, `info.`, `lists.`, `support.`, `wiki.`

The same rationale can be used for justifying a more fine-grained look at the hyperlinks explicitly posted on the mailing list. The four remaining infrastructure metrics are dummy variables that approximate the content behind the domain names stored to the set D in G_d. Given a domain name neighborhood of each CVE identifier in the vertex
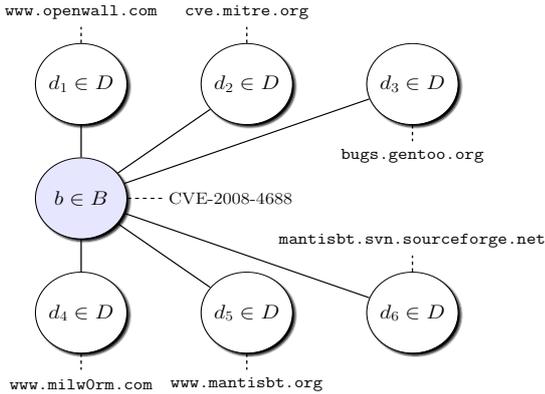
Figure 6: An Example Domain Name Neighborhood

set B, the value of these metrics is zero unless at least one domain name in the neighborhood matches the regular expressions listed in Table 2.

To further elaborate the operationalization of these metrics, Fig. 6 shows the neighborhood of a vertex labeled as CVE-2008-4688 in $B \in G_d$. For this CVE, the INFDEG metric takes a value six because there are six domain names linked to the identifier. Given the regular expressions, also the metrics VULNINF, BUGS, and REPOS each score a value one due to the vertices d_2, d_3, and d_6 (for the corresponding email see [76]). Of course, the expressions used are only approximations for automatically probing the nature of the primary tracking infrastructures. For instance, hosting services such as GitHub are mostly (but not entirely) excluded due to the matching primarily based on subdomains. Nevertheless, these infrastructure metrics convey a clear theoretical expectation. For instance, patches are often provided faster for high-profile vulnerabilities that affect multiple vendors, particularly in case computer emergency response teams are involved [90, 107]. For analogous reasons, VULNINF could be expected to decrease the coordination delays observed. Given the open source way of coordinating defects in open bug trackers, also BUGS can be expected to show a significant negative impact upon the delays. In other words, a good bug report often goes a long way in satisfying a prerequisite constraint for a CVE assignment in the open source context.

3.4.4. Vulnerability Metrics

The remaining metrics are used for evaluating whether the severity and technical characteristics of the vulnerabilities coordinated affect the delays. Given the existing bug tracking research, there are good reasons to expect a correlation. Although the empirical evidence seems to be again somewhat mixed [10], the severity of bugs have been observed to correlate with resolution delays in bug tracking systems [124, 125]. Furthermore, the complexity of source code has been observed to correlate with bug resolution times [51]. In terms of the metrics used, however, the bug tracking literature differs from vulnerability research.

Most bug tracking systems contain categories for assessing the severity and impact of bugs. The severity assignments within these systems are typically done either by bug reporters or by the associated developers. Due to the human work involved, the severity assignments have been observed to be relatively inconsistent and unreliable [108]. In contrast to bug tracking systems, the severity of vulnerabilities archived to NVD are evaluated by experts by using the Common Vulnerability Scoring System (CVSS). Although it remains debatable whether the severity assignments are suitable for assessing the security risks involved [6, 119], the assignments themselves are highly consistent across different databases and evaluation teams [40]. Therefore, in technical terms, CVSS provides a good and reliable framework for seeking an answer to the question RQ_3.

The metrics in the models M_4 and M_5 are all based on the second version of the CVSS standard [31]. (It is worth also remarking that only data based on this version is provided in NVD for the historical archival material relevant to the case studied.) This CVSS version classifies the severity of the vulnerabilities archived according to two dimensions: *exploitability* possibilities and *impact* upon successful exploitation. The impact dimension contains three metrics (confidentiality, integrity, and availability). Each of these can take a value from three options (NONE, PARTIAL, or COMPLETE). For instance, successfully exploiting the recently discovered and disclosed high-profile Meltdown vulnerability results in complete loss of confidentiality, although integrity and availability remain unaffected [75]. Due to multicollinearity issues, the three impact metrics IMPC, IMPI, and IMPA are based on collapsing the three value categories according to:

$$g(x) = \begin{cases} 0 & \text{if } x \text{ is NONE,} \\ 1 & \text{if } x \text{ is PARTIAL or COMPLETE.} \end{cases} \quad (8)$$

An analogous operationalization is used for the three dummy variables EXPNET, EXPCPLX, and EXPAUTH (see Table 1). These approximate the exploitability dimension based on the access *vector* (whether exploitation requires a network or a local access), access *complexity* (how complex are the access conditions for exploitation), and *authentication* (whether exploitation requires prior authentication) metrics in the CVSS v.2 standard.

The CVSS standard and the associated data from NVD provide a wealth of information for approaching different security questions, but it is unclear how the re-coded impact and exploitability metrics may affect the coordination delays. There exists some evidence for a conjecture that the difficulty of reliable severity assessments vary in terms of what is being assessed and who is doing the assessments. This conjecture applies both to the CVSS framework [5] and to quantitative security assessments in general [41]. However, analogous reasoning does not seem to hold in the context of NVD and the second version of the CVSS

standard; the content of the standard does not notably affect the delays for CVSS assignments [88]. In addition to these empirical observations, it is difficult to speculate why some particular CVSS metric would either increase or decrease the coordination and related delays [93]. For these reasons, the CVSS metrics are used in the models without prior theoretical expectations. The assumption merely is that the severity of the vulnerabilities coordinated is statistically associated with the delays observed.

The final set of metrics is based on the Common Weakness Enumeration (CWE) framework maintained and developed by MITRE in cooperation with the associated governmental partners and volunteers [69]. In essence, this comprehensive framework is used to catalog the typical "root causes" (weaknesses) that may lead to exploitable vulnerabilities in software products. The examples range from buffer and integer overflows to race conditions and software design flaws. By using a subset of frequent CWE identifiers [70], NVD maintainers derive the underlying weaknesses behind the vulnerabilities archived either during the CVE coordination or in retrospect.

Currently, the CWE framework covers over 700 distinct weaknesses. Partially due to this large amount, some of the CWEs are difficult to evaluate in practice [33, 117]. This difficulty does not necessarily imply that the framework itself would be problematic. Rather, many vulnerabilities chain a lot of distinct weaknesses together. For instance, CWEs can be used to exemplify that buffer overflow vulnerabilities posit a complex "mental model" for developers due to the tangled web of distinct programming mistakes involved [117]. Given analogous reasoning, it can be hypothesized that NVD-based CWE information can also explain a portion of the variation in the CVE coordination delays. As an example: when compared to XSS and CSRF vulnerabilities (as identified with CWE-79 and CWE-352), it may be that buffer overflow vulnerabilities (CWE-119) result in slower coordination because such vulnerabilities are usually technically more complex than simple web vulnerabilities. In other words, the effort required for interpretation may vary from a vulnerability to another, and such variation may show also in the coordination delays. To probe this general assumption, the final model M_6 includes the ten most frequent CWEs in the sample. Given that about 90% of the CVEs in the sample have valid CWE entries in NVD, together these top-10 entries account for about 79% of the CWEs available.

3.5. Estimation

The coordination delay vector $\mathbf{y} = [y_1, \ldots, y_n]'$ represent count data: the observations count the days between CVE assignments on the mailing list and the publication of the CVEs assigned within the primary global tracking database. Therefore, Poisson regression [88, 93] and regression estimators for survival analysis [1, 107] have been typical statistical estimation strategies in the research domain. Previous work in comparable settings [7, 26, 88, 90]

indicates that the ordinary least squares (OLS) regression often works also sufficiently well when the dependent metric is passed through a transformation function $f(x) = \log(x + 1)$. This simple OLS approach is taken as the baseline for the empirical analysis. For all regression models estimated, the explanatory metrics with continuous scale are also transformed via the same function in order to lessen skew.

Although frequently used in applied research, the OLS approach contains also problems. In a sense, the transformation function is redundant and should be avoided in order to maintain the statistical properties of count data. It also adds small but unnecessary complexity for interpreting the parameter estimates. Furthermore, the residual vector $\boldsymbol{\epsilon}$ from the OLS model is assumed to be independent and identically distributed from the normal distribution with a mean of zero and variance σ^2. This basic assumption can be written as $E(\boldsymbol{\epsilon} \mid \mathbf{X}_j) = \mathbf{0}$ and $E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}' \mid \mathbf{X}_j) = \sigma^2\mathbf{I}$, where $E(\cdot)$ denotes the expected value, $\mathbf{X}_j$ the j:th model matrix for a given M_j from Table 1, and $\mathbf{I}$ an identity matrix. If the assumption is satisfied,

$$\boldsymbol{\beta} \mid \mathbf{X}_j \sim N(\boldsymbol{\beta}, \sigma^2[\mathbf{X}_j'\mathbf{X}_j]^{-1}), \tag{9}$$

where $\boldsymbol{\beta}$ is a regression coefficient vector and $N(\cdot)$ refers to the multivariate normal distribution. This distributional assumption allows exact inference based on t and F distributions. In many empirical software engineering applications the problem is not as much about the (asymptotic) normality assumption as it is about the unreliable variance patterns typically present in the typically messy datasets [44]. Thus, often, $E(\boldsymbol{\epsilon}\boldsymbol{\epsilon}' \mid \mathbf{X}_j) = \sigma^2\mathbf{V}$, where $\mathbf{V}$ is a generally unknown non-diagonal matrix that establishes some systematic pattern in the residuals. Such tendency is generally known as heteroskedasticity.

For instance, some vulnerability time series are known to exhibit time-dependent heteroskedastic patterns [91, 106]. In the present context heteroskedasticity is to be expected due to the count data characteristics, and possibly also due to the heavy use of dichotomous variables. Either way, it is important to emphasize that asymptotic inference is still possible under heteroskedasticity; the consistency of $\boldsymbol{\beta}$ remains unaffected but the estimates are inaccurate. An analogous consequence results from multicollinearity; the stronger the correlation between a variable and other variables, the higher the variance of the regression coefficient for the variable.

When heteroskedasticity is an issue, more accurate statistical significance testing can be done by adjusting the variance-covariance matrix, $\mathrm{Var}(\boldsymbol{\beta} \mid \mathbf{X}_j) = \sigma^2[\mathbf{X}_j'\mathbf{X}_j]^{-1}$, with well-known techniques based on the unknown $\mathbf{V}$ estimated from data. The OLS results reported use the MacKinnon-White adjustment [61] conveniently available from existing implementations [123]. Another point is that rather analogous problems are often encountered with count data regressions. For instance, the negative binomial distribution is often preferable over the Poisson distribu-

tion [88]. Also the gamma distribution is known to characterize related difference-based vulnerability datasets [39]. Furthermore, conventional methods for survival analysis face some typical problems. Although not worth explicitly reporting, it can be shown that the Cox regression's so-called proportional hazards assumption fails for most of the metrics in $M_1, \ldots, M_6$, for instance. Due to these and other reasons, it beneficial to use a regression estimator that makes no distributional assumptions. Quantile regression (QR) is one of such estimators.

Ordinary least squares provides estimates for the conditional mean. Quantile regression estimates quantiles. The τ:th quantile is defined by

$$x_\tau = \inf\{x \mid F(x) \geq \tau\}, \quad 0 \leq \tau \leq 1, \qquad (10)$$

where $F(\cdot)$ denotes a cumulative distribution function, while the infimum operator is used to denote the smallest real number for which the condition $F(x) \geq \tau$ is satisfied. If $\tau = 0.5$, for instance, QR provides a solution $\hat{\boldsymbol{\beta}}$ for the conditional median of $\mathbf{y}$ conditional on $\mathbf{X}_j$. Estimation minimizes the sum of absolute residuals:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \rho_\tau(y_i - \beta_0 - \mathbf{x}'_{i(j)}\boldsymbol{\beta}), \qquad (11)$$

where β_0 is the intercept, $\mathbf{x}'_{i(j)}$ is the i:th row vector from the j:th model matrix $\mathbf{X}_j$, and $\rho_\tau(x) = x[\tau - I(x < 0)]$ with $0 < \tau < 1$ and $I(\cdot)$ denoting an indicator function [3, 47]. The optimization of (11) is computed with a so-called Frisch-Newton algorithm available in existing implementations [48]. The benefits from QR are well-known and well-documented. Among these are the lack of distributional assumptions and the robustness against outliers. Although asymptotic assumptions are still required for significance testing under heteroskedasticity, no parametric assumptions are made regarding the residual vector [45]. For applied research, quantile regression has also an immediate appeal in that the whole range in the conditional distribution of the explained variable can be observed. This potential is also relevant for vulnerability delay metrics, which typically (but not necessarily) tend to be distributed from a long-tailed distribution.

There are a couple of additional points that still warrant brief attention. First, potential non-linearities should be accounted for also with QR. As the dataset contains many variables but only five of these have a $\mathcal{C}$ scale (see Table 1), non-linear modeling of the explanatory metrics can be reasonably left for further work, however. The $\log(x+1)$ transformation applied to these five variables also lessens some of these concerns (particularly regarding MSGSLEN and MSGSENT). Second, multicollinearity is always a potential issue with typical empirical software engineering datasets. Although the concern is not as pressing as with software source code metrics [30], analogous datasets containing social network and communication metrics allow to also expect potential multicollinearity issues [8]. Instead of adopting techniques such as principal component

regression—which would make the interpretation of Fig. 3 difficult—the QR models are re-checked with the least absolute shrinkage and selection operator (LASSO). For quantile regression, LASSO amounts to optimizing

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \rho_\tau(y_i - \beta_0 - \mathbf{x}'_{i(j)}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1, \qquad (12)$$

where $\|\cdot\|_1$ denotes the L_1-norm and λ is a non-negative tuning parameter [3]. Although cross-validation and other techniques can be used for selecting the tuning parameter [88], the QR-LASSO regressions are estimated with $\lambda \in [1, 2, \ldots, 100]$ using the full M_6. This range captures most of the regularization applicable to the dataset.

The rationale behind (12) follows the rationale of LASSO in general: when λ increases, the quantile regression coefficients shrink toward zero. This regularization makes LASSO useful as a variable selection tool for high-dimensional and ill-conditioned datasets. When λ is sufficiently large, some of the coefficients shrink to zero, leaving a group of more relevant coefficients. The model selection properties are not entirely ideal, however. When a coefficient for a correlated variable is regularized to zero, the coefficients of the other correlated variables are affected; that is, LASSO tends to select one correlated variable from a group of highly correlated variables [17]. It is worth to remark that also the common alternatives contain problems. In addition to issues related to multiple comparisons, the expected heteroskedasticity presumably causes problems for a stepwise variable selection algorithm particularly in case the algorithm uses statistical significance to make decisions. Instead of relying on such algorithms, a more traditional is adopted for the modeling and inference.

3.6. Modeling

Applied regression modeling has two essential functions. It can be used to make predictions based on explanatory metrics, or it can be used to examine "the strength of a theoretical relationship" between the explained metric and the explanatory metrics [32]. Already because prediction is not a sensible research approach for a historical case study, this paper leans toward the examination of theoretical relationships based on classical statistical inference. The two functions cannot be arguably separated from each other, however. Given the notorious data limitations affecting vulnerability archiving [19], the inevitable noise introduced by the collection of online data, and many related reasons, a watchful eye should be kept for asserting the presence of a signal in terms of prediction. The magnitudes of the regression coefficients, the effect sizes, are used to balance the final subjective judgment calls regarding the research questions RQ_1, RQ_2, and RQ_3.

The six models $M_1, \ldots, M_6$ are fitted consecutively. This hierarchical modeling approach [8] is also known as a bottom-up or specific-to-general modeling strategy [60]. For comparing the six OLS models, the adjusted coefficients of determination (adj. R^2) and Akaike's information criterion (AIC) values are used. Higher and lower

values, respectively, indicate better performance according to these two common evaluation statistics. Both penalize the performance by the number of parameters. Although pseudo-R^2 measures are available for quantile regression [47], AIC is used to summarize the general signal-to-noise-type of performance of four quantile regressions estimated for each of the six models. The following quantiles are used in order to have probes across a wide range of the conditional distribution of the coordination delays:

$$\tau = [0.25, 0.50, 0.75, 0.90]. \tag{13}$$

It should be noted that neither censoring nor transformations are applied for the delays when estimated with QR. Consequently, the predicted values may be also negative, which is theoretically impossible. As prediction is not a goal, this limitation can be accepted.

The specific-to-general strategy is used also for formal testing with two analytical dimensions. First, the nested structure is exploited to test parameter restrictions between models. The testing is done by comparing (restricted) M_{j-1} against (unrestricted) M_j for $1 < j \leq 6$. The procedure follows the standard (Wald's) logic to compare nested models [47]. In fact, also the test statistics are delivered via F-like statistics [45]. Second, QR provides also means to test whether the coefficients are equal for a given M_j across the whole (13) or subsets thereof. For instance, to evaluate whether the effect of WEEKEND remains constant across the proxied conditional range of the coordination delays, the twentieth coefficients from $\{\hat{\boldsymbol{\beta}}_{\tau_1}, \hat{\boldsymbol{\beta}}_{\tau_2}, \hat{\boldsymbol{\beta}}_{\tau_3}, \hat{\boldsymbol{\beta}}_{\tau_4}\}_j$ for the j:th model would be used to test that $\hat{\beta}_{20j\tau_1} \simeq \hat{\beta}_{20j\tau_2} \simeq \hat{\beta}_{20j\tau_3} \simeq \hat{\beta}_{20j\tau_4}$. As it is reasonable to speculate that the effect of the explanatory metrics differ particularly at the tails, the coefficients in the following sets are used for the between-quantile testing:

$$
\begin{aligned}
S_1 &= \{\ \hat{\boldsymbol{\beta}}_{\tau_1}, \hat{\boldsymbol{\beta}}_{\tau_2}\}_j, \\
S_2 &= \{\ \hat{\boldsymbol{\beta}}_{\tau_1}, \hat{\boldsymbol{\beta}}_{\tau_2}, \hat{\boldsymbol{\beta}}_{\tau_3}\}_j, \\
S_3 &= \{\ \hat{\boldsymbol{\beta}}_{\tau_1}, \hat{\boldsymbol{\beta}}_{\tau_2}, \hat{\boldsymbol{\beta}}_{\tau_3}, \hat{\boldsymbol{\beta}}_{\tau_4}\}_j.
\end{aligned}
\tag{14}
$$

The bootstrap procedure available from the implementation used [48] is used to compute the statistical significance for the between-model tests. To accompany the MacKinnon-White adjustment for the OLS estimates, the same bootstrap procedure is used also for reporting the statistical significance of the coefficients from the QR regressions. Unfortunately, analogous procedure has not been implemented for the between-quantile tests. The conventional approach [45] is thus used instead.

Finally, the regression analysis is accompanied with a brief classification experiment to further assess the general performance. Following the existing bug tracking research [35, 124], this experiment is conducted by splitting the coordination delays into low-delay and high-delay groups according to median. Although this sample splitting is a good example of data manipulation that can be avoided with quantile regression [46], it provides a simple additional assertion regarding the overall performance across the six models. Classification accuracy is a sufficient evaluation metric for this simple purpose. Estimation is done with a readily available and well-known random forest classifier [49, 56]. Ten-fold cross-validation is used during training. Given that prediction of new data is not a realistic scenario for the historical `oss-security` case studied, testing is done with a randomly picked set containing ten percent of the CVEs observed. The same test set is used for all six classification models.

4. Results

The results are disseminated by first presenting a few relevant descriptive statistics. A summary of the regression analysis follows. In addition, four computational checks are presented about the statistical performance.

4.1. Descriptive Statistics

The sample contains $n = 5,780$ identifiers once the exclusion criteria in (2) is enforced (see Table 3). These were discussed by about five hundred unique participants who posted hyperlinks containing 4,642 unique domains. The mean and median delays were 77 and 15 days, respectively. The values for (13) are 4, 15, 92, and 226 days. As can be seen from the histogram in the outer plot of Fig. 7, the delay distribution indeed has a long tail. This tail contributes to the large standard deviation of 147 days. Thus, there is a large majority group of CVEs that were coordinated rapidly and a small but important group of CVEs for which the coordination was significantly delayed. For a few outlying CVEs, the coordination has taken even over four years. To balance this remark, it can be noted that about 10% of the cases observed attain a value zero, meaning that these CVEs appeared in NVD during the same day when these were requested on the mailing list.

Table 3: Sample Characteristics

Quantity	Value
$|A| =$ Number of CVEs	5,780
$|P| =$ Number of participants	496
$|D| =$ Number of domain names	4,642

As can be seen from the inner plot in Fig. 7, the transformation function $f(x) = \log(x + 1)$ does not yield normally distributed delays, although the shape of the delay distribution is still better suited for OLS regression after the transformation. To examine whether immediate multicollinearity issues are also visible, Fig. 9 displays the absolute correlation coefficients between all explanatory variables. (As with the regression analysis, the transformation function is used also for the continuous variables marked with the symbol $\mathcal{C}$ in Table 1.) The interpretation is simple: the bigger a circle and the darker a color, the
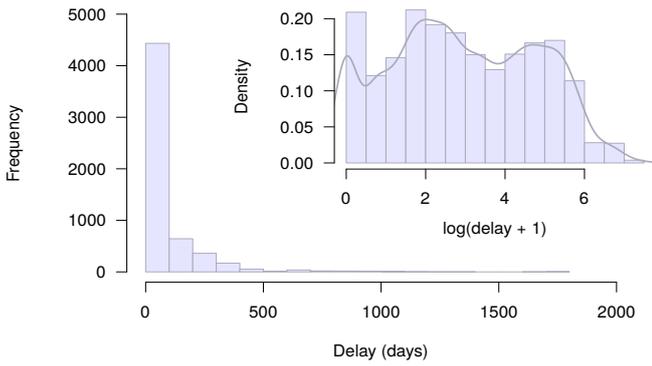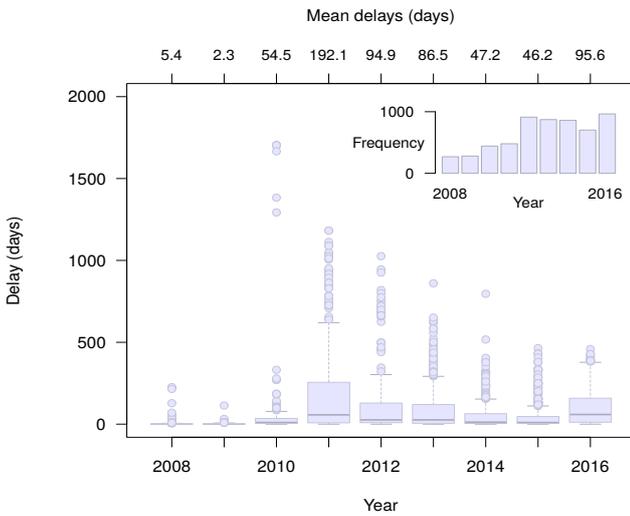
Figure 7: Coordination Delays

Figure 8: Annual Coordination Delays

stronger the correlation between any two metrics. Given this simple visual decoding guide, there are two notable correlations that warrant a multicollinearity concern.

The first is between MITREDEV and MSGSENT. The correlation between these two metrics reflects the typical "use this identifier" replies made by the MITRE affiliates. In other words, such replies tend to result in a lower entropy for the messages referencing CVEs. The second notable correlation is between INFDEG and MSGSLEN. Also this correlation is logical: posting many hyperlinks increases the length of the messages. Despite of these correlations, the regression estimates do not change notably when only MITREDEV and INFDEG (or, equivalently, MSGSENT and MSGSLEN) are included in the models. The same applies to the few visible correlations between the CVSS and CWE metrics. For these reasons, the results reported in the subsequent section are based on the original model specifications summarized in Table 1.

To examine the longitudinal variation, the outer plot in Fig. 8 shows the delays across the period observed. This illustration clearly shows that the delays started to increase from 2010 onward, but the increases were mostly caused by

outlying CVEs. The arithmetic mean delays shown on the upper x-axis indicate that the average annual delays have been below one hundred days. The annual median delays are all below 60 days. These averages align roughly with the so-called *grace periods* (the time vendors are given to patch their products before public release of information) typically used in the security industry during vulnerability disclosure. Although the lengths of these grace periods vary, an upper limit of about three months seems to capture most explicitly reinforced policies [64, 90]. Finally, the inner plot in Fig. 8 shows that the increasing delays and the increasing variation corresponded with the increasing amount of CVEs coordinated. The increased coordination volume in turn corresponded with the increased number of participants [95]. These longitudinal changes allow to expect that the baseline model M_1 explains a relatively large share of the total variation in the delays. This expectation provides a good way to start the dissemination of the results from the regression analysis.

4.2. Model Performance

The statistical performance is somewhat modest regardless of the model. Analogous to interpreting effect sizes [42], adjectives such as modest are subject to interpretation, of course. Values $R^2 < 0.3$ are neither atypical in the vulnerability research domain [7] nor uncommon in empirical software engineering experiments in general. Such values are commonly seen also in social sciences, including economics [43]. In this sense, the about 28% of the total variation explained by the OLS model M_6 indicates modest but typical performance for regressions involving human beings. This observation can be seen from Table 4, which shows the adjusted coefficients of determination, the AIC values, and the differences between the AIC values of the consecutively estimated models. The latter two are shown also for each of the four per-model QR regressions.

Four further points can be made about the performance. First, as expected, the longitudinal control variables provide most of the explanatory power. When compared to M_1, the social network and communication metrics increase the performance by about one percentage point according to the OLS estimates. The infrastructure metrics subsequently increase the performance by about three percentage points according to the adjusted R^2 values. Second, the conditional median ($\tau = 0.5$) and the conditional log-transformed mean (OLS) models indicate comparable behavior in terms of ΔAIC. Third, the QR estimates indicate that the longer the delays, the bigger the performance gains from M_2 and M_3. Particularly when the tail is probed with $\tau = 0.9$, the social network, communication, and infrastructure metrics reduce the AIC values substantially. In contrast, when short delays are proxied with $\tau = 0.25$, these metrics do not bring performance gains. Last: at best, there are only small improvements after M_3, and there are also cells with positive ΔAIC increments.

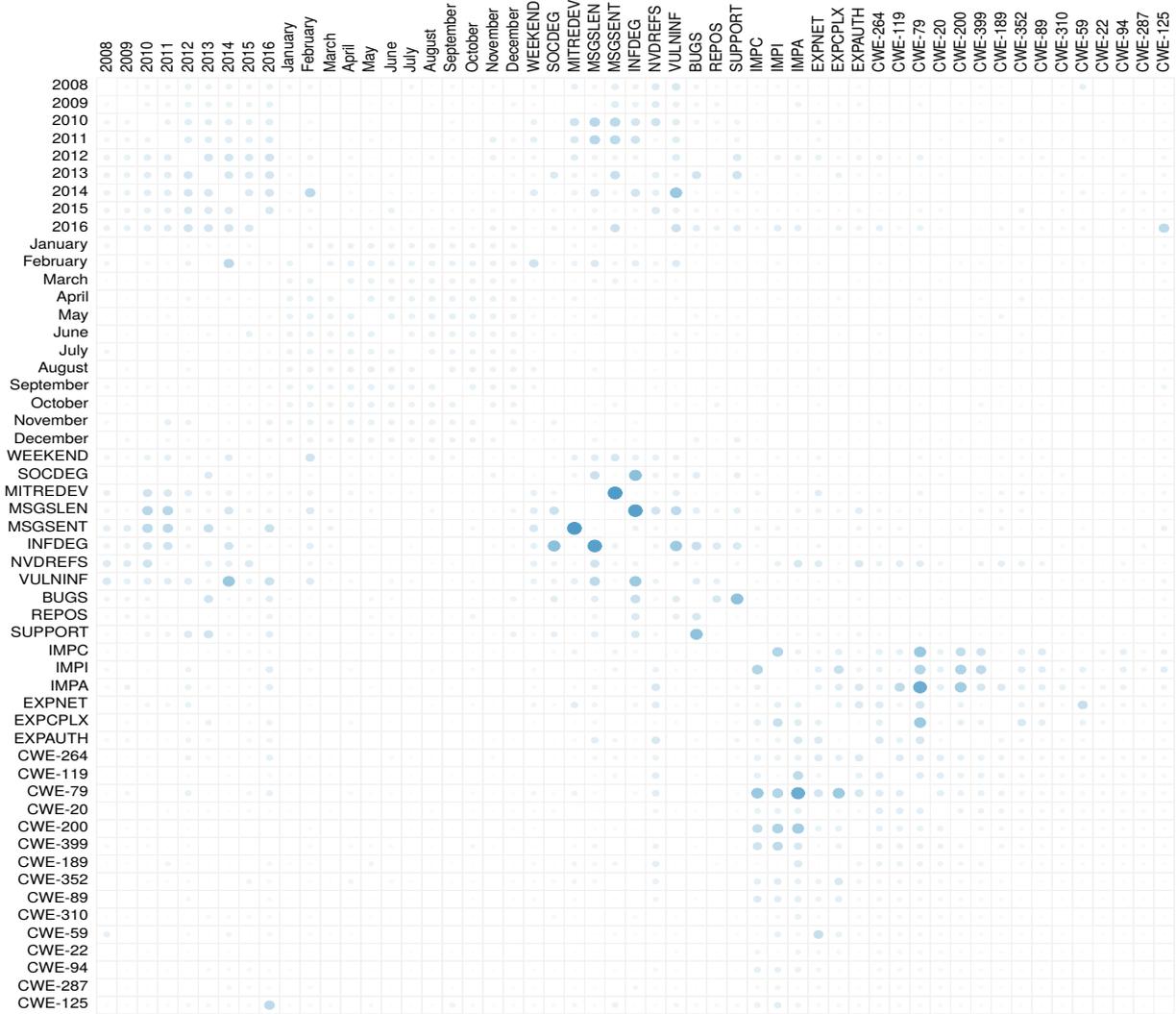However, all but one of the consecutively estimated nested models yield statistically significant rejections of

Figure 9: Correlations Between Explanatory Metrics (absolute values of Spearman's rank correlation coefficients)

Table 4: Model Performance

| | OLS | | | QR | | | | | | | | |
| | (mean) | | | $\tau = 0.25$ | | $\tau = 0.50$ | | $\tau = 0.75$ | | $\tau = 0.90$ | |
M_k	Adj. R^2	AIC	ΔAIC	AIC	ΔAIC	AIC	ΔAIC	AIC	ΔAIC	AIC	ΔAIC
M_1	0.240	21841	–	64775	–	68490	–	73136	–	77920	–
M_2	0.250	21774	-67	64772	-3	68472	-18	72956	-180	77032	-888
M_3	0.276	21572	-202	64771	-1	68428	-44	72757	-199	76561	-471
M_4	0.278	21563	-9	64776	+5	68426	-2	72740	-17	76544	-17
M_5	0.277	21567	+4	64781	+5	68431	+5	72741	+1	76545	+1
M_6	0.281	21552	-15	64796	+15	68433	+2	72721	-20	76504	-41

the between-model parameter restrictions. In other words, 95% of the hypotheses that the smaller (restricted) models would be adequate compared to the larger models are rejected according to the bootstrapped computations. It may also be that information criteria measures such as AIC have problems in differentiating between models when the overall performance is modest [43]. The turn to statistical significance motivates to also take a look at the residuals from the full M_6 in the form of Fig. 10. The residuals are rather randomly scattered across the x-axes only for the conditional median regression. For $\tau = 0.25$, there is a ∩-shaped pattern. For $\tau = 0.75$ and particularly for
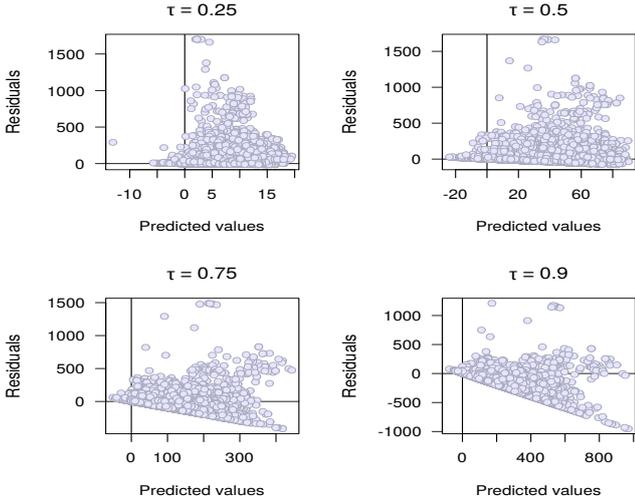
Figure 10: Residuals and Fitted Values (QR, M_6)

$\tau = 0.9$, there exists a more visible linear heteroskedasticity pattern; the longer the delays estimated, the larger the residuals. It is beyond the scope of this paper to review and evaluate how the QR-based inference performs under the heteroskedasticity patterns observed. Nevertheless, it seems reasonable to prefer the conditional median regressions and proceed with caution particularly when interpreting the $\tau > 0.5$ quantile regressions.

4.3. Regression Estimates

The regression estimates from the full unrestricted model are shown in Fig 11. The accompanying Fig. 12 shows the results from the between-quantile tests. As noted in the previous section, these tests as well as the statistical significance of the coefficients should be interpreted tentatively. While keeping this point in mind, the following enumeration summarizes the key observations.

- The effects of the longitudinal control metrics are consistently strong. When compared to 2008, the increased delays particularly in 2011, 2012, and 2016 are visible especially with respect to the $\tau = 0.9$ quantile regression. The same information is conveyed in Fig. 8. There exists also monthly variation in the delays. In addition to annual holidays, another reason may relate to security conferences and related events that tend to spike the public disclosure of new vulnerabilities [37]. In any case, it is simpler to interpret the positive effect of WEEKEND. When compared to other days of the week, the median delays have been about two weeks longer for requests made on weekends. The effect of WEEKEND also increases at the tails of the conditional delay distribution.

- The social network and communication metrics exhibit strong effects. In particular, the assumption

	OLS	$\tau = 0.25$	$\tau = 0.5$	$\tau = 0.75$	$\tau = 0.9$
(Intercept)	-3.2	-62.2	-54.9	78.7	617.4
2009	-0.1	-0.2	-3.2	-13.6	-23.9
2010	1.8	3.3	13.8	23.9	43.6
2011	3.0	7.6	48.1	241.2	437.6
2012	2.4	5.9	18.6	86.2	148.6
2013	2.0	2.9	15.5	46.2	86.0
2014	1.4	1.8	4.1	3.2	28.5
2015	1.4	1.8	1.9	5.9	88.9
2016	2.5	9.1	46.0	103.9	156.4
February	0.8	3.1	20.9	37.0	53.2
March	0.5	2.4	10.1	25.1	4.5
April	0.2	0.9	4.0	32.5	0.2
May	0.5	2.3	11.6	27.3	1.5
June	0.2	2.7	4.0	0.7	-38.3
July	0.3	1.3	6.5	9.7	-32.2
August	0.1	0.7	3.9	5.9	-47.9
September	0.4	3.0	8.1	8.4	-40.8
October	0.5	4.5	14.4	0.5	-34.3
November	0.4	2.4	11.4	2.0	4.9
December	0.2	1.5	5.0	-13.3	-62.3
WEEKEND	0.5	2.1	13.7	42.8	63.3
SOCDEG	1.0	1.5	19.0	148.9	372.2
MITREDEV	-0.2	-2.0	1.1	-0.2	-5.6
MSGSLEN	< -0.01	0.9	0.3	-14.1	-45.3
MSGSENT	2.5	34.5	34.4	-21.0	-298.2
INFDEG	0.1	-1.0	4.1	23.2	76.6
NVDREFS	-0.6	-1.5	-11.1	-36.9	-56.1
VULNINF	< 0.01	0.4	-0.4	4.7	-12.7
BUGS	-0.3	-1.5	-5.5	-21.3	-29.3
REPOS	-0.1	-0.2	-2.2	-6.0	-22.7
SUPPORT	< -0.01	-0.2	-0.4	3.7	-14.3
IMPC	< -0.01	0.4	-1.1	-7.1	-19.6
IMPI	-0.1	-0.7	-3.9	-7.8	2.0
IMPA	-0.1	-0.4	-0.9	-11.6	-10.5
EXPNET	< 0.01	< -0.01	-0.9	-2.3	-4.4
EXPCPLX	< -0.01	< -0.01	1.0	6.7	-2.9
EXPAUTH	< -0.01	-0.5	< -0.01	4.1	12.8
CWE-264	-0.2	-0.6	-4.4	-13.4	-36.5
CWE-119	-0.2	-0.8	-2.7	-7.6	-16.3
CWE-79	-0.3	-0.5	-7.4	-32.7	-48.2
CWE-20	-0.2	-1.2	-5.0	-6.9	-13.7
CWE-200	-0.1	-1.0	-3.3	-9.2	1.2
CWE-399	-0.4	-1.6	-7.1	-17.4	-17.8
CWE-189	< 0.01	1.3	-0.2	-0.5	-13.0
CWE-352	-0.3	-0.3	-5.5	-25.6	-40.4
CWE-89	-0.4	-1.6	-7.6	-11.6	-29.2
CWE-310	< 0.01	-0.5	0.2	0.1	9.8

Figure 11: Regression Coefficients (M_6, darker blue values denote $p < 0.05$, MacKinnon-White standard errors for the OLS regression and bootstrapped standard errors for the quantile regressions)

	S_1	S_2	S_3
2009	True	False	False
2010	False	False	False
2011	False	False	False
2012	False	False	False
2013	False	False	False
2014	True	True	False
2015	True	True	False
2016	False	False	False
February	False	False	False
March	False	False	False
April	True	False	False
May	False	False	False
June	True	True	True
July	False	False	False
August	True	True	False
September	False	False	False
October	False	False	False
November	False	False	False
December	True	False	False
WEEKEND	False	False	False
SOCDEG	False	False	False
MITREDEV	False	True	True
MSGSLEN	True	False	False
MSGSENT	True	True	False
INFDEG	False	False	False
NVDREFS	False	False	False
VULNINF	True	True	False
BUGS	False	False	False
REPOS	True	True	False
SUPPORT	True	True	True
IMPC	True	True	False
IMPI	False	True	True
IMPA	True	True	True
EXPNET	True	True	True
EXPCPLX	True	True	False
EXPAUTH	True	True	True
CWE-264	False	False	False
CWE-119	True	True	True
CWE-79	False	False	False
CWE-20	False	True	True
CWE-200	True	True	True
CWE-399	False	False	False
CWE-189	True	True	True
CWE-352	False	False	False
CWE-89	False	False	False
CWE-310	True	True	True

Figure 12: Between-Quantile Tests (given the full unrestricted M_6, the cells show whether $p \geq 0.05$ for testing the null hypotheses that the slope coefficients are equal with respect to the sets in Eq. 14)

about "too many cooks" seem to hold well; the effect of $\log(\text{SOCDEG}+1)$ is large particularly for long coordination delays. Also the communication surrogates $\log(\text{MSGSLEN}+1)$ and $\log(\text{MSGSENT}+1)$ show large effects. The signs vary, however. Because the coefficients are positive for $\tau \leq 0.5$ and negative for $\tau \geq 0.75$, the prior theorization in Subsection 3.4.2 should not be as unequivocal as was presented.

- From the infrastructure metrics, the coefficients for INFDEG, NVDREFS, and BUGS are statistically significant for each regression. The coefficient magnitudes are also notable. If a CVE request was accompanied with a hyperlink to a bug tracking system, the median delay was about five to six days shorter, for instance. As was expected (see Subsection 3.4.3), the signs are also negative for NVDREFS and BUGS but positive for INFDEG. It seems that hyperlinks can also increase the noise, which tends to increase the CVE coordination delays.

- The CVSS and CWE metrics show diverging results. On one hand, only three of the coefficients for the CVSS metrics are statistically significant in the five M_6 regressions. The effect sizes are also small, and mostly equal according to the between-quantile tests. On the other hand, many of the CWE metrics attain statistically significant coefficients with large magnitudes. All of the CWE metrics that are statistically significant have negative signs. As these observations apply also for the predominantly web-related weaknesses (CWE-79, CWE-89, and CWE-352; to some extent, also CWE-20), it seems that mundane low-profile vulnerabilities are generally coordinated faster. As was discussed in Subsection 3.4.4, interpretation is not easy, however. The difficulty of interpretation further increases because some of the CWE metrics likely proxy the effects of the CVSS metrics due to multi-collinearity (see Fig. 9). All in all, it can be concluded that the coordination delays vary also in terms of the technical characteristics of the vulnerabilities coordinated. As will be shown in the next section, performance can be also slightly increased with additional weaknesses, although the signals from the CVSS and CWE metrics still remain somewhat weak.

4.4. Computational Checks

Four computational checks are made to assess the robustness of the conclusions. The first check relates to the number of CWEs included; the model M_6 includes the ten most frequent CWEs in the sample, but there are 55 unique CWEs in total. The summary shown in Fig. 13 displays the performance when separate models are estimated by adding $5, 10, 15, \ldots, 55$ weaknesses to the fifth model specification. The case with ten weaknesses thus equals M_6. The OLS performance increases up to around 25 weaknesses. For the median QR regression, all of the

fifty-five CWEs seem to steadily reduce the AIC values. The negative ΔAIC increments are also relatively large compared to those in Table 4. These improvements should not be exaggerated, however. The adjusted R^2 values from the OLS regressions still do not reach the 0.3 threshold, for instance.
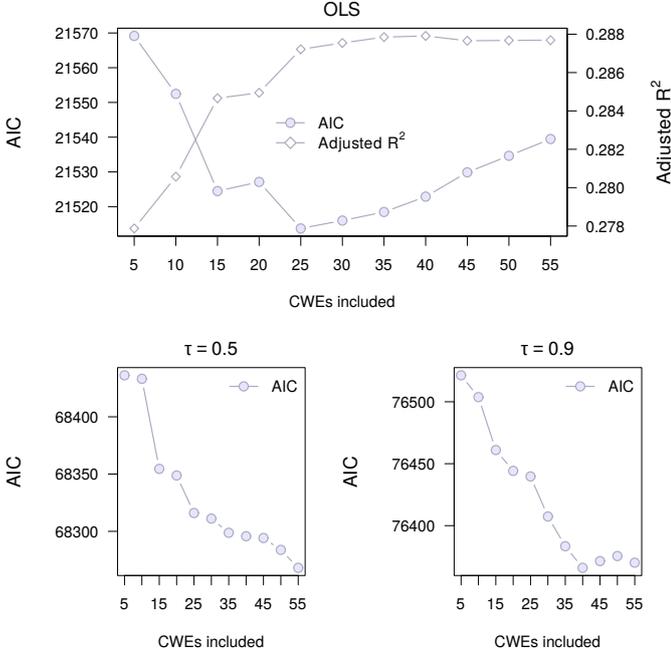

Figure 13: Model Performance with Varying Number of CWEs

The second check relates to the classification of low-delay and high-delay groups (see Subsection 3.5). The split according to median results in a roughly balanced metric for classification: $2,904$ of the CVEs observed attain a delay less than or equal to the median delay and $2,876$ a delay higher than the median delay of 15 days. The accuracy of classifying these two groups is shown in Table 5. The accuracy rates increase steadily up to the fourth model and decrease thereafter. These classification results agree with the regression results shown in Table 4.

Table 5: Classification Performance (median split)

| | Model | | | | | |
	M_1	M_2	M_3	M_4	M_5	M_6
Accuracy	0.62	0.67	0.71	0.75	0.73	0.73

The third check is about the annual variation. In contrast to subsetting the explained metric, segmenting a model matrix into subsets according to the conditioning explanatory metrics is a sensible regression modeling approach [46]. Thus, to examine whether also the model performance varies annually, all six models are estimated with OLS in annual subsets, omitting the eight yearly dummy variables present in the original model specifications. For instance: if $\tilde{M}_1$ denotes the first subset-model without the

Table 6: OLS Performance in Annual Subsets (adj. R^2)

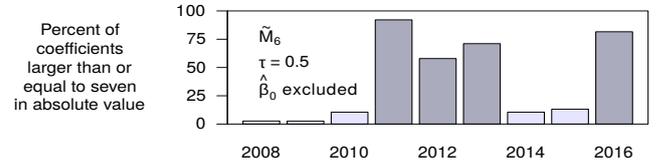| | Subset Model | | | | | |
Year	$\tilde{M}_1$	$\tilde{M}_2$	$\tilde{M}_3$	$\tilde{M}_4$	$\tilde{M}_5$	$\tilde{M}_6$
2008	0.10	0.32	0.40	0.40	0.40	0.40
2009	0.21	0.45	0.48	0.48	0.48	0.48
2010	0.11	0.15	0.20	0.21	0.22	0.22
2011	0.15	0.22	0.34	0.34	0.38	0.38
2012	0.28	0.31	0.32	0.33	0.34	0.34
2013	0.07	0.32	0.42	0.42	0.44	0.45
2014	0.23	0.22	0.25	0.25	0.25	0.26
2015	0.15	0.17	0.18	0.18	0.19	0.21
2016	0.03	0.05	0.16	0.16	0.16	0.20
k	13	17	23	26	29	39

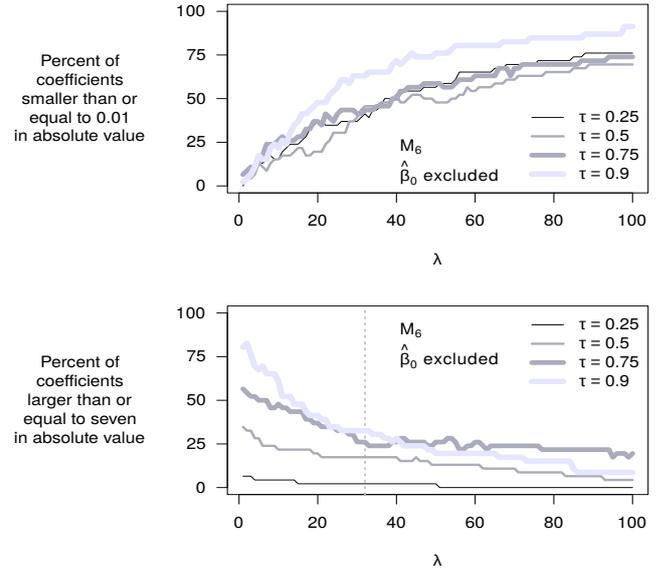Figure 14: Large Effects in Annual Subsets (QR, $\tilde{M}_6$, $\tau = 0.5$)

Figure 15: QR Effects with LASSO (M_6)

annual dummy variables, there are $k = 21 - 8 = 13$ parameters in the model and its regression coefficient vector. The results from these subset OLS regressions are summarized in Table 6. The table should be only read horizontally by comparing values in a given row across the columns; the different sample sizes in the annual subsets (see Fig. 8) do not allow to make sound comparisons across the years. Given this interpretation guide, the results are clear. For the majority of years, there are no notable performance

gains after the model $\tilde{M}_3$. Another way to look at the annual variation is to check how well the large effect sizes of the annual dummy variables (see Fig. 11) balance the effects of the other metrics. For this purpose, the median QR regression for the subset model $\tilde{M}_6$ can be briefly examined. Even when keeping in mind that the coefficients are not directly comparable as some of the metrics have different scales, the summary shown in Fig. 14 roughly tells that the large effects are indeed pronounced during those years when the CVE coordination was delayed. Given that the majority of metrics show strong effects during these four years, the annual dummy variables seem to control relatively well the coefficient magnitudes reported in Fig. 11.

The fourth and final check is about (12). The four hundred QR-LASSO regressions estimated are summarized in Fig. 15 for M_6. The upper plot indicates that even with a modest regularization such as $\lambda = 20$, about a quarter of the coefficients are close to zero. Given that a week seems like a reasonably strong effect size with practical relevance, the lower plot indicates that the regularization seems to stabilize around $\lambda \leq 40$ for $\tau < 0.9$. For the $\tau = 0.9$ regressions, which generally gather the strongest effects (see Fig. 11), the applicable regularization seems to continue further. For the conditional median QR-LASSO regressions, only eight coefficients are larger than or equal to seven in absolute value at $\lambda = 30$. These are: the annual dummy variables for 2011, 2012, 2013, and 2016, the monthly February dummy variable, WEEKEND, SOCDEG, and NVDREFS. These belong to the longitudinal control metric group, the group of social network and communication metrics, and the infrastructure metric group. None of the CVSS and CWE metrics pass this subjective threshold.

5. Discussion and Conclusions

The results presented allow to answer positively to each three research questions RQ_1, RQ_2, and RQ_3. If the answers should be ordered in terms of importance of the corresponding statistical effects, the order would follow the numbering of the questions. The remainder of this paper summarizes the main findings, discusses the limitations, and points out a couple of prolific but challenging research paths for further work.

5.1. Main Findings

The main empirical findings can be summarized and theorized with the following five points.

1. The first point is clear: even though nearly fifty explanatory metrics were considered, the statistical performance was relatively modest for explaining the CVE coordination delays. Less than one third of the total variation in the delays is explained by the OLS regression models estimated. The median quantile regression results largely agree. Analogous results were also obtained by classifying CVEs with a low-delay and high-delay split according to median. The adjusted coefficient of determination is not an ideal statistic to make comparisons (especially when OLS is not used), but it is still worth remarking that the bug tracking research frequently attains values around 0.5 or even more [10, 26]. The modest performance should not be interpreted as poor performance, however. Given the typical effect sizes seen in empirical software engineering research [42], some of the regression coefficients and their standard errors indicate consistent, accurate, and large effects upon the CVE coordination delays.

2. The second point is likewise clear: most of the explanatory power comes from metrics used to proxy longitudinal variation. This result is not surprising. In contrast, it would be surprising if a software engineering study would not reveal longitudinal effects in a period covering almost a decade. The result is also familiar from comparable studies about vulnerability coordination [88] and time series aspects of vulnerability archiving [37, 106]. If avoiding delays is important, the results can be also used to conjecture that "do not request CVEs during weekends". The longitudinal variation implies a further important takeaway message. Given that the efficiency of software engineering coordination is always dependent also on the *volume* of items being coordinated, it seems that the `oss-security` case studied reflects the larger coordination issues that have gained publicity in recent years. By assumption, delays for CVE assignments are largely explained by the mere amount of these vulnerability identifiers requested. By implication, a good theoretical metric for prediction would simply be the amount of identifiers in an abstract CVE backlog.

3. The third point is about *noise* that tends to lengthen the coordination delays. Both the social network (RQ_1) and the infrastructure (RQ_2) effects manifest themselves through different abstractions for such noise. In terms of the former, noise increases when there are "too many cooks" posting emails with high-entropy content. Both factors tend to increase also the CVE coordination delays. In the terms of the latter, noise increases with emails containing multiple hyperlinks to distinct tracking infrastructures within which the vulnerabilities have already been discussed or to which these have already been archived.

4. The fourth point relates to prerequisite constraints. When satisfied, these constraints tend to balance the positive effect of noise upon the delays by shortening the coordination delays to some extent. For instance, traces to bug tracking systems show a small but visible negative effect. The same applies regarding traces about the information sources regularly polled by the parties involved in the CVE tracking. For develop-

ers requesting CVE identifiers for their projects, the practical takeaway message is conveyed by an advice such as: "write good bug reports to backup CVE requests".

5. The fifth point relates to the technical characteristics of the vulnerabilities coordinated. The social network, communication, and infrastructure metrics (RQ_1 and RQ_2) explain a few percentage points of the total variation in the delays. Given roughly comparable observations about the impact of social interactions on software quality [8], these effects seem reasonable. Although the results are somewhat mixed, the technical characteristics (RQ_3) do not generally explain the delays well. When backtracking to Fig. 2, the explanation may be simple: for participants who have assigned nearly thousand CVEs via `oss-security` alone, it may be irrelevant whether a request is about XSS or about buffer overflows. As there are still some signals about the statistical relevance of the CVSS and CWE metrics, another plausible theoretical explanation may be that it is mentally easier to handle vulnerabilities that fall explicitly into the scopes of some particular CWEs. Given the common problem of overloading a single vulnerability report with multiple distinct (security) issues [19], the final practical takeaway message could be that: "try to avoid ambiguities when requesting CVE identifiers".

All in all, software vulnerability coordination can be concluded to exhibit typical characteristics of software engineering coordination in general. Dependencies, social relations, communication, and technical software elements are all present. These are also the factors that with varying degrees correlate with different coordination problems, including the CVE coordination delays observed.

5.2. Threats to Validity

Some limitations must be mentioned. According to a common taxonomy, the validity of the results reported are potentially exposed to threats to external validity, construct validity, and internal validity [114]. The discussion that follows is structured according to this taxonomy.

5.2.1. External Validity

External validity relates to generalizability. Insofar as CVE assignments are considered, `oss-security` is a unique case that was limited to one particular way of assigning CVE identifiers during one particular historical period. Thus, neither the results nor the conclusions necessarily apply to other ways to assign CVEs, including the contemporary practices. While acknowledging this limitation, it is important to underline that analogous threats to external validity presumably continue to constrain also further research on vulnerability coordination.

On one hand, research in this domain is dependent on the openness (or lack thereof) of the internal tracking infrastructures used by MITRE and related parties. While keeping in mind the sensitivity of the information tracked, one question for practitioners to consider is whether data could be partially opened by using embargo periods akin to the grace periods used during vulnerability disclosure. The so-called distributed weakness filing project is a good step toward this direction [79]. On the other hand, empirical research is more generally constrained by the lack of robust open data on the software engineering activities related to vulnerability coordination. The point applies also in the open source context. For instance, invitation-only coordination media [80] and other types of information hiding restrict the possibility of continuing the work [2] on security-related integration work done in open source projects. These constraints imply that further software engineering research in the domain is presumably as much about "reverse engineering" as it is about software and security engineering processes and coordination practices.

5.2.2. Construct Validity

The reverse engineering tenet is visible also in terms of construct validity. This type of validity relates to questions about how well the metrics used and the questions asked reflect the theoretical ideas and research goals. The most notable construct validity threat is directly related to the operationalization in (1). Because CVE requests were not explicitly modeled due to data limitations, even with the restriction in (2), it is impossible to say whether a CVE that appeared on the list was already disseminated to MITRE via other channels. While the delays observed should still provide a reasonable approximation, it is perhaps more important to note that the metric used provides only a limited viewpoint on the coordination. Unfortunately, the lack of open data makes it difficult to study more nuanced coordination aspects such as task allocation and work parallelism related to CVE coordination. Bayesian methods and expert opinion [40, 41] may help at resolving this constraint and related limitations.

Also some of the explanatory metrics are exposed to modest construct validity threats. Without attempting to participate in the current debates [6, 40, 119], it is reasonable to assert that the CVSS and CWE data from NVD is robust enough for the purposes of this paper. Construct validity is a bigger issue for the few custom metrics derived. Most of the social network and communication metrics have been successfully used previously [8, 12]. The operationalization was also deliberately restricted to metrics that are easy to compute and interpret. Therefore, the approximations used for the infrastructure metrics are more noteworthy. In particular, the regular expressions in Table 2 are inadequate for explicitly linking different tracking infrastructures together. While a subset of the CVEs discussed on `oss-security` could be explicitly linked to bug tracking and related systems via repository mining techniques [27, 72, 86], the historical period studied largely prevents such linking because many of the trackers are nonexistent today (see Subsection 3.4.3). In this sense, the approximations are a necessary evil for this paper.

5.2.3. Internal Validity

Internal validity relates to questions about computational and statistical biases particularly when causal relations are postulated. Three internal validity threats are worth pointing out. First, even when causal inference is not attempted, the data availability issues translate to potential problems in terms of omitted variables and confounding factors. Second, not all options were examined for thoroughly assessing the reasons behind the modest performance. For instance, clustering could be used in conjunction with regression analysis [9, 90]. Another option might be to examine different CWE-based ontologies [111, 117]. The third and final threat to internal validity relates to the statistical issues reported.

As is typical to software engineering datasets [30, 44], non-normality, heteroskedasticity, and multicollinearity were all present with varying degrees of severity. Further work is required to examine these issues with respect to generally more robust but still not invulnerable techniques such as quantile regression and LASSO. Quantile regression addresses the distributional assumption, while the heteroskedasticity patterns observed (see Fig. 10) would be easy to explicitly adjust for in further work. The multicollinearity issue is more interesting. Given that the CVSS (v. 2) metrics are alone challenging to adequately handle in empirical research due to the various plausible combinations between the variables, further work is also required on variable selection in the software engineering context. The potential solutions should be systematic and algorithmic to ensure consistency of theoretical arguments—otherwise the so-called researcher bias is inevitable for regression analysis with interacting variables. By using combinations of CVSS metrics, combinations of CWE metrics, combinations of CVSS and CWE metrics, and other combinations, it would be possible to assert almost an unlimited amount of theoretical propositions.

Finally, the real issue is more subtle than what might be remedied by merely changing a statistical modeling approach. There have been historical delays also for CVSS assignments [88], there are delays between reporting bugs in tracking systems, fixing these in version control systems, and integrating the fixes to releases [26, 83], and so forth. All such delays may affect also CVE coordination. While statistical modeling is one thing, analytical understanding of the complexities involved is another.

5.3. Moving Forward

The limitations discussed prompt a couple of points about potential means for moving forward. First, it is sensible to recommend a closer marriage between the software vulnerability and bug tracking research domains. The intersection between the domains is large but seldom explicitly articulated. Such a marriage might also remedy the somewhat modest statistical performance reported. In this regard, a sensible conclusion is that the information used was too limited for capturing the truly relevant elements affecting the particular type of coordination observed. When looking at the bug tracking research domain, some of the frequently incorporated dimensions (such as the reputation of bug reporters and the text mining of bug reports) seem prolific for pursuing further also in the context of software vulnerabilities. These and related dimensions are relevant for studying also more contemporary questions such as those related to bug bounties.

Second, a marriage between two domains alone seems insufficient for making more practical advances. Mining of software repositories frequently provides valuable insights about software engineering coordination through case studies. Yet, as was briefly demonstrated, the practical problem is that there are hundreds (if not thousands) of relevant repositories to mine. For advances with practical relevance, affairs are required also with other computer science domains such as information retrieval and web crawling. If one is to believe the current "mega trends", maybe some day, perhaps in a galaxy far away, also the coordination of abstract identifiers could be done by robots.

References

[1] Ablon, L. and Bogart, A. (2017). Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits. RAND Corporation, Santa Monica. Available online in September 2017: https://www.rand.org/content/dam/rand/pubs/research_reports/RR1700/RR1751/RAND_RR1751.pdf.

[2] Adams, B., Kavanagh, R., Hassan, A. E., and German, D. M. (2016). An Empirical Study of Integration Activities in Distributions of Open Source Software. *Empirical Software Engineering*, 21(3):960–1001.

[3] Alhamzawi, R., Yu, K., and Benoit, D. F. (2012). Bayesian Adaptive Lasso Quantile Regression. *Statistical Modelling*, 12(3):279–297.

[4] Allodi, L. (2017). Economic Factors of Vulnerability Trade and Exploitation: Empirical Evidence from a Prominent Russian Cybercrime Market. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS 2017)*, pages 1483–1499, Dallas. ACM.

[5] Allodi, L., Biagioni, S., Crispo, B., Labunets, K., Massacci, F., and Santos, W. (2017). Estimating the Assessment Difficulty of CVSS Environmental Metrics: An Experiment. In Dang, T. K., Wagner, R., Küng, J., Thoai, N., Takizawa, M., and Neufhold, E. J., editors, *Proceedings of the International Conference on Future Data and Security Engineering (FDSE 2017), Lecture Notes in Computer Science (Volume 10646)*, pages 23–39. Springer.

[6] Allodi, L. and Massacci, F. (2014). Comparing Vulnerability Severity and Exploits Using Case-Control Studies. *ACM Transactions on Information and System Security*, 17(1):1:1–1:20.

[7] Arora, A., Forman, C., Nandkumar, A., and Telang, R. (2010). Competition and Patching of Security Vulnerabilities: An Empirical Analysis. *Information Economics and Policy*, 22(2):164–177.

[8] Bettenburg, N. and Hassan, A. E. (2013). Studying the Impact of Social Interactions on Software Quality. *Empirical Software Engineering*, 18(2):375–431.

[9] Bettenburg, N., Nagappan, M., and Hassan, A. E. (2014). Towards Improving Statistical Modeling of Software Engineering Data: Think Locally, Act Globally! *Empirical Software Engineering*, 20(2):294–335.

[10] Bhattacharya, P. and Neamtiu, I. (2011). Bug-Fix Time Prediction Models: Can We Do Better? In *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR 2011)*, pages 207–210, Waikiki. ACM.

[11] Bilge, L. and Dumitras, T. (2012). Before We Knew It: An Empirical Study of Zero-Day Attacks in the Real World. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS 2012)*, pages 833–844, Raleigh. ACM.

[12] Bird, C. (2011). Sociotechnical Coordination and Collaboration in Open Source Software. In *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance (ICSM 2011)*, pages 568–573, Williamsburg. IEEE.

[13] Bird, C., Gourley, A., Devanbu, P., Gertz, M., and Swaminathan, A. (2006). Mining Email Social Networks. In *Proceedings of the 2006 International Workshop on Mining Software Repositories (MSR 2006)*, pages 137–143, Shanghai. ACM.

[14] Blincoe, K., Valetto, G., and Damian, D. (2015). Facilitating Coordination between Software Developers: A Study and Techniques for Timely and Efficient Recommendations. *IEEE Transactions on Software Engineering*, 41(10):969–985.

[15] Bozorgi, M., Saul, L. K., Savage, S., and Voelker, G. M. (2010). Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pages 105–114, London. ACM.

[16] Cetin, O., Gañán, C., Korczyński, M., and van Eeten, M. (2017). Make Notifications Great Again: Learning How to Notify in the Age of Large-Scale Vulnerability Scanning. In *Proceedings of the 16th Workshop on the Economics of Information Security (WEIS 2017)*, San Diego. Available online in January 2018: http://weis2017.econinfosec.org/wp-content/uploads/sites/3/2017/05/WEIS_2017_paper_17.pdf.

[17] Chia-Yen Lee, B.-S. C. (2017). Mutually-Exclusive-and-Collectively-Exhaustive Feature Selection Scheme. *Applied Soft Computing*, 68:961–971.

[18] Christen, P. (2012). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer, Berlin.

[19] Christey, S. and Martin, B. (2013). Buying Into the Bias: Why Vulnerability Statistics Suck. In *Presentation at Black Hat 2013*, Las Vegas. available online in January 2017: https://media.blackhat.com/us-13/US-13-Martin-Buying-Into-The-Bias-Why-Vulnerability\-Statistics-Suck-Slides.pdf.

[20] Christey, S. and Wysopal, C. (2002). Responsible Vulnerability Disclosure Process. Internet Engineering Task Force (IETF), INTERNET-DRAFT, available online in December 2017: https://tools.ietf.org/html/draft-christey-wysopal-vuln-disclosure-00.

[21] Conaldi, G. and Lomi, A. (2013). The Dual Network Structure of Organizational Problem Solving: A Case Study on Open Source Software Development. *Social Networks*, 35(2):237–250.

[22] Conaldi, G., Lomi, A., and Tonellato, M. (2012). Dynamic Models of Affiliation and the Network Structure of Problem Solving in Open Source Software Projects. *Organizational Research Methods*, 15(3):385–412.

[23] Conway, M. E. (1968). How Do Committees Invent? *Datamation*, 14(5):28–31.

[24] Crowston, K. and Howison, J. (2006). Hierarchy and Centralization in Free and Open Source Software Team Communications. *Knowledge, Technology & Policy*, 18(4):65–85.

[25] Crowston, K. and Shamshurin, I. (2017). Core-Periphery Communication and the Success of Free/Libre Open Source Software Projects. *Journal of Internet Services and Applications*, 8(10):1–11.

[26] da Costa, D. A., McIntosh, S., Kulesza, U., Hassan, A. E., and Abebe, S. L. (2018). An Empirical Study of the Integration Time of Fixed Issues. *Empirical Software Engineering*, 23(1):334–383.

[27] Dashevskyi, S., Brucker, A. D., and Massacci, F. (2019). A Screening Test for Disclosed Vulnerabilities in FOSS Components. *IEEE Transactions on Software Engineering*, 45(10):945–966.

[28] Dehghani, M., Asadpour, M., and Shakery, A. (2012). An Evolutionary-Based Method for Reconstructing Conversation Threads in Email Corpora. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 1132–1137, Istanbul. IEEE.

[29] Eyolfson, J., Tan, L., and Lam, P. (2011). Do Time of Day and Developer Experience Affect Commit Bugginess? In *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR 2011)*, pages 153–162, Honolulu. ACM.

[30] Fenton, N. E. and Neil, M. (1999). A Critique of Software Defect Prediction Models. *IEEE Transactions on Software Engineering*, 25(5):675–689.

[31] FIRST (2007). A Complete Guide to the Common Vulnerability Scoring System Version 2.0, FIRST.ORG. Available online in June 2015: https://www.first.org/cvss/cvss-v2-guide.pdf.

[32] Giacalone, M., Panarello, D., and Mattera, R. (2017). Multicollinearity in Regression: An Efficiency Comparison Between L_p-Norm and Least Squares Estimators. *Quality & Quantity*, 52(4):1831–1859.

[33] Goseva-Popstojanova, K. and Perhinschi, A. (2015). On the Capability of Static Code Analysis to Detect Security Vulnerabilities. *Information and Software Technology*, 68:17–33.

[34] Guzzi, A., Bacchelli, A., Lanza, M., Pinzger, M., and van Deursen, A. (2013). Communication in Open Source Software Development Mailing List. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR 2013)*, pages 277–286, San Francisco. IEEE.

[35] Habayeb, M., Murtaza, S. S., Miranskyy, A., and Bener, A. B. (2018). On the Use of Hidden Markov Model to Predict the Time to Fix Bugs. *IEEE Transactions on Software Engineering*, 44(12):1224–1244.

[36] Howison, J. and Crowston, K. (2014). Collaboration Through Open Superposition: A Theory of the Open Source Way. *MIS Quarterly*, 38(1):29–50.

[37] Joh, H. and Malaiya, Y. K. (2017). Periodicity in Software Vulnerability Discovery, Patching and Exploitation. *International Journal of Information Security*, 16(6):673–690.

[38] Johnson, P. and Ekstedt, M. (2016). The Tarpit – A General Theory of Software Engineering. *Information and Software Technology*, 70:181–203.

[39] Johnson, P., Gorton, D., Langerström, R., and Ekstedt, M. (2016). Time Between Vulnerability Disclosures: A Measure of Software Product Vulnerability. *Computers & Security*, 62:278–295.

[40] Johnson, P., Lagerström, R., Ekstedt, M., and Franke, U. (2017). Can the Common Vulnerability Scoring System be Trusted? A Bayesian Analysis. *IEEE Transactions on Dependable and Secure Computing*, 15(6):1002–1015.

[41] Johnston, R., Sarkani, S., Mazzuchi, T., Holzer, T., and Eveleigh, T. (2018). Multivariate Models Using MCMCBayes for Web-Browser Vulnerability Discovery. *Reliability Engineering & System Safety*, 176:52–61.

[42] Kampenes, V. B., Dybå, T., Hannay, J. E., and Sjøberg, D. I. (2008). A Systematic Review of Effect Size in Software Engineering Experiments. *Information and Software Technology*, 49(11–12):1073–1086.

[43] Karlsson, P. S., Behrenz, L., and Shukur, G. (2019). Performance of Model Selection Criteria When Variables are Ill Conditioned. *Computational Economics*, 54:77–98.

[44] Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., and Pohthong, A. (2017). Robust Statistical Methods for Empirical Software Engineering. *Empirical Software Engineering*, 22(2):579–630.

[45] Koenker, R. and Bassett, G. (1982). Robust Tests for Heteroscedasticity Based on Regression Quantiles. *Econometrica*, 50(1):43–61.

[46] Koenker, R. and Hallock, K. F. (2001). Quantile Regression. *Journal of Economic Perspectives*, 15(4):143–156.

[47] Koenker, R. and Machado, J. A. F. (1999). Goodness of Fit and Related Inference Processes for Quantile Regression. *Journal of the American Statistical Association*, 94(448):1296–1310.

[48] Koenker, R., Portnoy, S., Ng, P. T., Zeileis, A., Grosjean, P., and Ripley, B. D. (2018). quantreg: Quantile Regression. R package version 5.3.5, available online in April 2018: https://cran.r-project.org/web/packages/quantreg/index.html.

[49] Kuhn, M. (2008). Building Predictive Models in R Using the

caret Package. *Journal of Statistical Software*, 28(5):1–26.

[50] Kuk, G. (2006). Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science*, 52(7):1031–1042.

[51] Kula, R. G., Fushida, K., Kawaguchi, S., and Iida, H. (2010). Analysis of Bug Fixing Processes Using Program Slicing Metrics. In Babar, M. A., Vierimaa, M., and Oivo, M., editors, *Proceedings of the International Conference on Product Focused Software Process Improvement (PROFES 2010), Lecture Notes in Computer Science (Volume 6156)*, pages 32–46, Limerick. Springer.

[52] Laszka, A., Zhao, M., and Grossklags, J. (2016). Banishing Misaligned Incentives for Validating Reports in Bug-Bounty Platforms. In Askoxylakis, I., Ioannidis, S., Katsikas, S., and Meadows, C., editors, *Proceedings of the European Symposium on Research in Computer Security (ESORICS 2016), Lecture Notes in Computer Science (Volume 9879)*, pages 161–178, Heraklion. Springer.

[53] Lee, G., Espinosa, J. A., and DeLone, W. H. (2013). Task Environment Complexity, Global Team Dispersion, Process Capabilities, and Coordination in Software Development. *IEEE Transactions on Software Engineering*, 39(12):1753–1771.

[54] Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710.

[55] Leyden, J. (2017). Most Vulnerabilities First Blabbed About Online or on the Dark Web: Official Bug Notice? Sure, but not Before I Get Cred and LOLs. The Register. Available online in December 2017: `http://www.theregister.co.uk/2017/06/08/vuln_disclosure_lag/`.

[56] Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. *R News*, 2(3):18–22.

[57] Licorish, S. A. and MacDonell, S. G. (2014). Understanding the Attitudes, Knowledge Sharing Behaviors and Task Performance of Core Developers: A Longitudinal Study. *Information and Software Technology*, 56(12):1578–1596.

[58] Linares-Vásquez, M., Bavota, G., and Escobar-Velásquez, C. (2017). An Empirical Study on Android-Related Vulnerabilities. In *IEEE/ACM 14th International Conference on Mining Software Repositories (MSR 2017)*, pages 1–13, Buenos Aires. IEEE.

[59] Lubarski, P. and Morzy, M. (2012). Measuring the Importance of Users in a Social Network Based on Email Communication Patterns. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Network Analysis and Mining (ASONAM 2012)*, pages 86–90, Istanbul. IEEE.

[60] Lütkepohl, H. (2007). General-to-Specific or Specific-to-General Modelling? An Opinion on Current Econometric Terminology. *Journal of Econometrics*, 136(1):319–324.

[61] MacKinnon, J. G. and White, H. (1985). Some Heteroskedasticity-Consistent Covariance Matrix Estimators with Improved Finite Sample Properties. *Journal of Econometrics*, 29(3):305–325.

[62] Malone, T. W. and Crowston, K. (1994). The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26(1):87–119.

[63] McChesney, I. R. (1997). Effective Coordination in the Software Process – Historical Perspectives and Future Directions. *Software Quality Journal*, 6(3):235–246.

[64] McQueen, M. A., McQueen, T. A., Boyer, W. F., and Chaffin, M. R. (2009). Empirical Estimates and Observations of 0Day Vulnerabilities. In *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS 2009)*, pages 1–12, Honolulu. IEEE.

[65] Meneely, A. and Williams, L. (2010). Strengthening the Empirical Analysis of the Relationship Between Linus' Law and Software Security. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2010)*, pages 9:1–9:10, Bolzano-Bozen. ACM.

[66] MITRE (2015a). CVE-ID Syntax Change. Available online in December 2017: `https://cve.mitre.org/cve/identifiers/syntaxchange.html`.

[67] MITRE (2015b). Frequently Asked Questions. Available online in December 2017: `https://cve.mitre.org/about/faqs.html`.

[68] MITRE (2015c). Please welcome Kurt Seifried to the CVE Editorial Board. Appeared originally in *cve-editorial-board-list*. Available online in September 2016: `https://cve.mitre.org/data/board/archives/2015-11/msg00002.html`.

[69] MITRE (2018a). Common Weaknesses Enumeration. Available online in January 2018: `http://cwe.mitre.org/`.

[70] MITRE (2018b). CWE VIEW: Weaknesses Originally Used by NVD from 2008 to 2016. Available online in January 2018: `http://cwe.mitre.org/data/definitions/635.html`.

[71] Ngamkajornwiwat, K., Zhang, D., Koru, A. G., Zhou, L., and Nolker, R. (2008). An Exploratory Study on the Evolution of OSS Developer Communities. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*, pages 305–315, Waikoloa. IEEE.

[72] Nguyen, V. H. and Massacci, F. (2013). The (Un)Reliability of NVD Vulnerability Versions Data: An Empirical Experiment on Google Chrome Vulnerabilities. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS 2013)*, pages 493–498. ACM.

[73] Nia, R., Bird, C., Devanbu, P., and Filkov, V. (2010). Validity of Network Analyses in Open Source Projects. In *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 201–209, Cape Town. IEEE.

[74] NIST (2017). NVD Data Feed and Product Integration. National Institute of Standards and Technology (NIST), Annually Archived CVE Vulnerability Feeds: Security Related Software Flaws, NVD/CVE XML Feed with CVSS and CPE Mappings (Version 2.0). Retrieved in 23 September 2017 from: `https://nvd.nist.gov/download.cfm`.

[75] NIST (2018). Common Vulnerability Scoring System Calculator: Version 2 – CVE-2017-5754. National Institute of Standards and Technology (NIST), Available online in January 2018: `https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator?name=CVE-2017-5754&vector=(AV:L/AC:M/Au:N/C:C/I:N/A:N)`.

[76] Openwall (2008). CVE request: mantisbt <1.1.4: RCE. Available online in January 2018: `http://openwall.com/lists/oss-security/2008/10/19/1`.

[77] Openwall (2016a). Archive of *oss-security* Mailing List. Available online in September 2016: `http://www.openwall.com/lists/oss-security/`.

[78] Openwall (2016b). Fwd: CVE request - samsumg android phone SVE-2016-6244 Possible Privilege Escalation in telecom. Available online in September 2016: `http://www.openwall.com/lists/oss-security/2016/08/05/1`.

[79] Openwall (2017a). MITRE is adding data intake to its CVE ID process. Available online in December 2017: `http://www.openwall.com/lists/oss-security/2017/02/09/7`.

[80] Openwall (2017b). Re: linux-distros subscription. Available online in December 2017: `http://www.openwall.com/lists/oss-security/2017/01/15/1`.

[81] Paasivaara, M. and Lassenius, C. (2003). Collaboration Practices in Global Inter-Organizational Software Development Projects. *Software Process Improvement and Practice*, 8(4):183–199.

[82] Parraguez, P., Eppinger, S. D., and Maier, A. M. (2015). Information Flow Through Stages of Complex Engineering Design Projects: A Dynamic Network Analysis Approach. *IEEE Transactions on Engineering Management*, 62(4):604–617.

[83] Perez, G. R., Robles, G., and Barahona, J. M. G. (2017). How Much Time Did It Take to Notify a Bug? Two Case Studies: ElasticSearch and Nova. In *Proceedings of the IEEE/ACM 8th Workshop on Emerging Trends in Software Metrics (WETSoM 2017)*, pages 29–35, Buenos Aires. IEEE.

[84] Poo-Caamaño, G., Knauss, E., Singer, L., and German, D. M. (2017). Herding Cats in a FOSS Ecosystem: A Tale of Communication and Coordination for Release Management. *Journal of Internet Services and Applications*, 8(1):1–24.

[85] Ring, T. (2015). White Hats Versus Vendors: The Fight Goes On. *Computer Fraud & Security*, (10):12–17.

[86] Romo, B. A., Capiluppi, A., and Hall, T. (2014). Filling the Gaps of Development Logs and Bug Issue Data. In *Proceedings of The International Symposium on Open Collaboration (OpenSym 2014)*, pages 1–4, Berlin. ACM.

[87] Ruohonen, J. (2017). Classifying Web Exploits with Topic Modeling. In *Proceedings of the 28th International Workshop on Database and Expert Systems Applications (DEXA 2017)*, pages 93–97, Lyon. IEEE.

[88] Ruohonen, J. (2019). A Look at the Time Delays in CVSS Vulnerability Scoring. *Applied Computing and Informatics*, 15(2):129–135.

[89] Ruohonen, J. and Allodi, L. (2018). A Bug Bounty Perspective on the Disclosure of Web Vulnerabilities. In *Proceedings of the 17th Annual Workshop on the Economics of Information Security (WEIS 2018)*, pages 1–14, Innsbruck. Available online in June 2019: https://weis2018.econinfosec.org/wp-content/uploads/sites/5/2018/05/WEIS_2018_paper_33.pdf.

[90] Ruohonen, J., Holvitie, J., Hyrynsalmi, S., and Leppänen, V. (2016a). Exploring the Clustering of Software Vulnerability Disclosure Notifications Across Software Vendors. In *Proceedings of the 13th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2016)*, pages 1–8, Agadir. IEEE.

[91] Ruohonen, J., Hyrynsalmi, S., and Leppänen, V. (2015). The Sigmoidal Growth of Operating System Security Vulnerabilities: An Empirical Revisit. *Computers & Security*, 55:1–20.

[92] Ruohonen, J., Hyrynsalmi, S., and Leppänen, V. (2016b). Trading Exploits Online: A Preliminary Case Study. In *Proceedings of the IEEE Tenth International Conference on Research Challenges in Information Science (RCIS 2016)*, pages 1–12, Grenoble. IEEE.

[93] Ruohonen, J., Hyrynsalmi, S., and Leppänen, V. (2017a). Modeling the Delivery of Security Advisories and CVEs. *Computer Science and Information Systems*, 14(2):537–555.

[94] Ruohonen, J. and Leppänen, V. (2017). Investigating the Agility Bias in DNS Graph Mining. In *Proceedings of the 17th IEEE International Conference on Computer and Information Technology (IEEE CIT 2017)*, pages 253–260, Helsinki. IEEE.

[95] Ruohonen, J., Rauti, S., Hyrynsalmi, S., and Leppänen, V. (2017b). Mining Social Networks of Open Source CVE Coordination. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement (IWSM Mensura 2017)*, pages 176–188, Gothenburg. ACM.

[96] Ruohonen, J., Šćepanović, S., Hyrynsalmi, S., Mishkovski, I., Aura, T., and Leppänen, V. (2016c). Correlating File-Based Malware Graphs Against the Empirical Ground Truth of DNS Graphs. In *Proceedings of the 10th European Conference on Software Architecture Workshops (ECSAW 2016)*, pages 30:1 – 30:6, Copenhagen. ACM.

[97] Sabottke, C., Suciu, O., and Dumitraş, T. (2015). Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In *Proceedings of the 24th USENIX Security Symposium*, pages 1041–1056, Washington. USENIX.

[98] Schmid, M. R., Iqbal, F., and Fung, B. C. M. (2014). E-Mail Authorship Attribution Using Customized Associative Classification. *Digital Investigation*, 14(S1):S116–S126.

[99] Schoch, D., Valente, T. W., and Brandes, U. (2017). Correlations Among Centrality Indices and a Class of Uniquely Ranked Graphs. *Social Networks*, 50:46–54.

[100] Seifried, K. (2017). CVE-HOWTO. Available online in December 2017: https://github.com/RedHatProductSecurity/CVE-HOWTO.

[101] Sierra, J. M., Vizcaíno, A., Genero, M., and Piattini, M. (2018). A Systematic Mapping Study about Socio-Technical Congruence. *Information and Software Technology*, 94:111–129.

[102] Śliwerski, J., Zimmermann, T., and Zeller, A. (2005). When Do Changes Induce Fixes? (On Fridays.). In *Proceedings of the International Workshop on Mining Software Repositories (MSR 2005)*, pages 1–5, Saint Louis. ACM.

[103] Stevanovic, M., Pedersen, J. M., D'Alconzo, A., and Ruehrup, S. (2016). A Method for Identifying Compromised Clients Based on DNS Traffic Analysis. *International Journal of Information Security*, 16(2):115–132.

[104] Syed, R., Rahafrooz, M., and Keisler, J. M. (2018). What It Takes to Get Retweeted: An Analysis of Software Vulnerability Messages. *Computers in Human Behavior*, 80:207–215.

[105] Tang, G., Pei, J., and Luk, W. (2014). Email Mining: Tasks, Common Techniques, and Tools. *Knowledge and Information Systems*, 41(1):1–31.

[106] Tang, M., Alazab, M., and Luo, X. (2019). Big Data for Cybersecurity: Vulnerability Disclosure Trends and Dependencies. *IEEE Transactions on Big Data*, 5(3):317–329.

[107] Temizkan, O., Kumar, R. L., Park, S., and Subramaniam, C. (2012). Patch Release Behaviors of Software Vendors in Response to Vulnerabilities: An Empirical Analysis. *Journal of Management of Information Systems*, 28(4):305–337.

[108] Tian, Y., Ali, N., Lo, D., and Hassan, A. E. (2016). On the Unreliability of Bug Severity Data. *Empirical Software Engineering*, 21(6):2298–2323.

[109] Toral, S. L., Martínez-Torres, M. R., and Barrero, F. (2009). Virtual Communities as a Resource for the Development of OSS Projects: The Case of Linux Ports to Embedded Processors. *Behavior & Information Technology*, 28(5):405–419.

[110] Toral, S. L., Martínez-Torres, M. R., and Barrero, F. (2010). Analysis of Virtual Communities Supporting OSS Projects Using Social Network Analysis. *Information and Software Technology*, 52(3):296–303.

[111] Tsipenyuk, K., Chess, B., and McGraw, G. (2005). Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors. *IEEE Security & Privacy*, 3(6):81–84.

[112] Wang, Q. (2014). Link Prediction and Threads in Email Networks. In *Proceedings of the International Conference on Data Science and Advanced Analytics (DSAA 2014)*, pages 470–476, Shanghai. IEEE.

[113] Wijayasekara, D., Manic, M., Wright, J. L., and McQueen, M. (2012). Mining Bug Databases for Unidentified Software Vulnerabilities. In *Proceedings of the 5th International Conference on Human System Interactions (HSI 2012)*, pages 89–96, Perth. IEEE.

[114] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer, Heidelberg, revised edition.

[115] Wolf, T., Schroter, A., Damian, D., and Nguyen, T. (2009a). Predicting Build Failures Using Social Network Analysis on Developer Communication. In *Proceedings of the IEEE 31st International Conference on Software Engineering (ICSE 2009)*, pages 1–11, Vancouver. IEEE.

[116] Wolf, T., Schröter, A., Damian, D., Panjer, L. D., and Nguyen, T. H. (2009b). Mining Task-Based Social Networks to Explore Collaboration in Software Teams. *IEEE Software*, 26(1):58–66.

[117] Wu, Y., Gandhi, R. A., and Siy, H. (2010). Using Semantic Templates to Study Vulnerabilities Recorded in Large Software Repositories. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems (SESS 2010)*, pages 22–28, Cape Town. ACM.

[118] Wu, Y. and Oard, D. W. (2005). Indexing Emails and Email Threads for Retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*, pages 665–666, Salvador. ACM.

[119] Younis, A., Malaiya, Y. K., and Ray, I. (2016). Evaluating CVSS Base Score Using Vulnerability Rewards Programs. In Hoepman, J.-H. and Katzenbeisser, S., editors, *Proceedings of the 31st IFIP TC 11 International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2016)*, pages 62–75, Ghent. Springer.

[120] Zanetti, M. S., Scholtes, I., Tessone, C. J., and Schweitzer, F. (2013a). Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities. In *Proceedings of the 35th International Conference on Software Engineering (ICSE 2013)*, pages 1032–1041, San Francisco. IEEE.

[121] Zanetti, M. S., Scholtes, I., Tessone, C. J., and Schweitzer, F. (2013b). The Rise and Fall of a Central Contributor: Dynamics of Social Organization and Performance in the GENTOO Community". In *Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013)*, pages 49–56, San Francisco. IEEE.

[122] Zangerle, E., Gassler, W., and Specht, G. (2013). On the Impact of Text Similarity Functions on Hashtag Recommendations in Microblogging Environments. *Social Network Analysis and Mining*, 3(4):889–898.

[123] Zeileis, A. (2004). Econometric Computing with HC and HAC Covariance Matrix Estimators. *Journal of Statistical Software*, 11(10):1–17.

[124] Zhang, F., Khomh, F., Zou, Y., and Hassan, A. E. (2012). An Empirical Study on Factors Impacting Bug Fixing Time. In *Proceedings of the 19th Working Conference on Reverse Engineering (WCRE 2012)*, pages 225–234, Kingston. IEEE.

[125] Zhou, B., Neamtiu, I., and Gupta, R. (2015). Experience Report: How Do Bug Characteristics Differ Across Severity Classes: A Multi-Platform Study. In *Proceedings of the 26th International Symposium on Software Reliability Engineering (ISSRE 2015)*, pages 507–517, Gaithersbury. IEEE.