Noname manuscript No.
(will be inserted by the editor)

On the Coordination of Vulnerability Fixes

An Empirical Study of Practices from 13 CVE Numbering Authorities

Jiahuei Lin · Bram Adams · Ahmed E. Hassan

Received: date / Accepted: date

Abstract The Common Vulnerabilities and Exposures (CVE) program is dedicated to analyzing vulnerabilities, then to assigning a unique ID to them and disclosing the vulnerabilities to affected software vendors. A CVE Numbering Authority (CNA) is a key partner in the CVE program responsible for assigning an official ID to a CVE and registering a description of the vulnerability in order to communicate it to the other CNAs and the affected software vendors. To avoid the disclosure of vulnerabilities before the development of a fix, the CNAs and the affected vendors need to coordinate a proper schedule for the disclosure of vulnerabilities and the release of their fixes through multi-party coordination. This paper analyzes the practices used by CNAs to coordinate on vulnerability fix releases and disclosure by empirically studying the 13 CNAs that assigned the most CVEs from 2010 to 2020 and are also software vendors. Our results show that the studied CNAs discover and assign CVE IDs for the majority of vulnerabilities that affect their own products, which we refer to as self-assigned vulnerabilities. While the vulnerabilities that are assigned for other CNAs' products, which we refer to as delegated vulnerabilities, tend to be more severe than the self-assigned vulnerabilities, (median Common Vulnerability Scoring System score of 7.5), we observe that their fixes are released at a slower pace. Moreover, when such a delegated vulnerability affects several CNAs' products, the fixes are released a median of 4 days after the disclosure date, with a median delay between the first and last

Jiahuei Lin, Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
Queen's University
Kingston, Ontario, Canada
E-mail: {jhlin,ahmed}@cs.queensu.ca

Bram Adams
Lab on Maintenance, Construction, and Intelligence of Software (MCIS)
Queen's University
Kingston, Ontario, Canada
E-mail: bram.adams@queensu.ca

patch releases of those products of 35 days up to more than one year, which corresponds to a large window of exploitation.

Keywords Developer coordination · Vulnerabilities · Patch release speed · Software ecosystems

1 Introduction

The surge of vulnerabilities in the past few years indicates the high risk of keeping software systems operational, since software vendors cannot guarantee a patch for all vulnerabilities before those are exploited by attackers. In fact, the number of exploits on zero-day software vulnerabilities has doubled from 2020 to 2021.[1] Releasing a timely fix becomes vital to reduce exploits of such vulnerabilities and save the losses of vulnerable software. In other words, a delay of fix releases until after disclosure implies higher risks to users. For example, millions of users waited for at least 6 months to get fixes of a particular vulnerability in Microsoft Word.[2] As a result, this delay led to a large number of attacks on victim systems and a substantial amount of losses.

To deal with vulnerabilities, organizations and companies often rely on vulnerability databases to become aware of vulnerabilities and get their fixes. The CVE program [2] is designed to link such vulnerability databases and other tools together by assigning a unique identifier (CVE ID) to a given vulnerability, which software vendors often include in the relevant mitigation plans (e.g., security advisories, fixes). Users leverage the CVE ID to find all the information (e.g., fixes) related to that vulnerability.

CVE IDs are assigned by a diverse group of organizations, companies and researchers, acting as CVE Numbering Authorities (CNAs) [1]. There are over two hundred CNAs, each of which are security experts within their own specific scopes of coverage. They evaluate a vulnerability before assigning an identifier, then control the disclosure process. That is, the CNAs, the discoverer, and the affected software vendors coordinate to develop a vulnerability fix and schedule a proper disclosure time, i.e., multi-party coordination.

There are many challenges for parties involved in multi-party vulnerability coordination and disclosure [4]. For example, the number of affected vendors is typically too large for the discoverer to deal with, thus involving a coordinator (CNA) to cope with this. However, the higher the number of involved parties in fixing a vulnerability, the higher the complexity of coordination between the involved parties. In such a case, the affected vendors are unlikely to release their fix in a coordinated fashion. In addition, when an affected vendor does not consider a report to be a vulnerability, the CNA is forced to disclose the vulnerability to nudge the vendor to release a fix, implying that the vulnerable products become the target of attackers. Furthermore, cloud

[1] `https://www.technologyreview.com/2021/09/23/1036140/2021-record-zero-day-hacks-reasons/`

[2] `https://www.businessinsider.com/hackers-microsoft-word-flaw-reuters-2017-4`

computing services often use customized versions of open-source projects like distributions or middleware. As such, the fix for a particular vulnerability in, say, a customized Debian version on OpenStack is likely to be released later than the fix for Debian itself.

It is important to note that, while CNAs correspond to security experts generating new CVE IDs, they also can play other roles in parallel, such as that of the discoverer, coordinator, and/or affected vendor. First, a CNA could be the discoverer of a vulnerability in its own products, and be responsible for assigning a CVE ID for the vulnerability, before, subsequently, having to develop a vulnerability fix in a timely manner. Second, the CNA could be responsible for both assigning a CVE ID and controlling the disclosure process, notifying the public of the vulnerability to remediate the vulnerable products. In this case, the CNA coordinates the schedule of releasing the fixes with the discoverer and the affected vendors, including the (upstream) vendor that developed the vulnerable product and those (downstream) that leveraged the vulnerable product as part of their products. As typically multiple downstream vendors rely on the fixes released by the upstream vendor, this increases the difficulty of coordination. Finally, the CNA could be one of the affected vendors receiving the notification of its vulnerable product, then has to release the fix according to the coordinated schedule.

The large threat of vulnerabilities led researchers to systematically analyze vulnerabilities from a wide range of aspects. Some prior studies analyzed vulnerability lifecycles to shed light on the vulnerability disclosure process [19, 40]. Other studies explored the disclosure process from the economy of vulnerabilities, such as vulnerability trading markets [11, 42], and evaluated the correlation with exploits [8]. Some studies worked on understanding the patch development process [23, 28, 34]. While providing insights of vulnerabilities for a certain aspect, these investigations have been limited to the software vendors' perspective. However, to the best of our knowledge, no study has focused on the special role of CNAs.

In this work, we conduct a large-scale empirical study of 39,409 vulnerabilities that affect products of the top 13 CNAs with the largest number of assigned CVE IDs, based on a dataset that merges vulnerability entries from the National Vulnerability Database (NVD) [5] and their associated security fixes. Due to the different responsibility of being a CNA and being a software vendor, we study the CNAs who have played both roles to gain an understanding of the efficiency of releasing vulnerability fixes w.r.t the disclosure date. We classify vulnerabilities into two groups: vulnerabilities whose CVE-IDs are assigned by the affected CNAs, i.e., *self-assigned vulnerabilities*, and those whose CVE IDs are not assigned by the affected CNAs, i.e., *delegated vulnerabilities*. We characterize in detail the delay of releasing vulnerability fixes in comparison to different affected product types and the number of vendors. We also explore the delay in each of the studied CNAs and summarize common practices. We structure our study by answering the following research questions, yielding the following key findings:

RQ1: What are the characteristics of self-assigned vs. delegated vulnerabilities?
The first RQ analyzes the prevalence of self-assigned vs. delegated vulnerabilities to measure the degree to which the CNAs focus on vulnerability discovery, and characterizes these two types of vulnerabilities. The majority of vulnerabilities affecting CNAs are self-assigned, though 3 out of the 13 CNAs have a similar percentage (≈50%) of self-assigned and delegated vulnerabilities. Application-type products have the largest proportion (47%) of self-assigned vulnerabilities across the CNAs, compared to the operating-system (35%) and hardware (18%) types of products. Furthermore, delegated vulnerabilities are more severe than self-assigned vulnerabilities, with a median Common Vulnerability Scoring System (CVSS) score of 7.5.

RQ2: How fast do the CNAs develop fixes for self-assigned vs. delegated vulnerabilities?
Due to the high severity level of delegated vulnerabilities, RQ2 measures how fast the studied CNAs release fixes for these vulnerabilities per product type, and compare the results with self-assigned vulnerabilities. In general, the studied CNAs release fixes for delegated vulnerabilities slower than self-assigned ones. For delegated vulnerabilities affecting the application type of product, while the studied CNAs release fixes in a median of -1 day (before the disclosure date), 2 of the CNAs (i.e., Apple and Oracle) release the majority of fixes at least 90 days after the disclosure date. For delegated vulnerabilities affecting the operating-system type of product, the delay of patch releases is a median of 3 days after the disclosure date. 3 out of the studied CNAs (i.e., Apple, Oracle, and Red Hat) release fixes for the majority of such vulnerabilities at least 90 days after the disclosure date.

RQ3: How effectively do the CNAs coordinate to release fixes for shared vulnerabilities?
Since the delay of patch releases for delegated vulnerabilities is longer than self-assigned ones, this RQ studies potential collaboration between the studied CNAs on releasing fixes. We measure the delay of vulnerabilities that affect multiple CNAs' products, i.e., multi-affected vulnerabilities, vs. those that affect one CNA's products, i.e., single-affected vulnerabilities. 3 out of the studied CNAs have a majority of their vulnerabilities as multi-affected. The CVSS score of multi-affected vulnerabilities that affected both application and operating-system types of products reaches almost the full score of 10 (extremely severe), while the CVSS score of those that affected one product type is a median of 5.1 (medium-severity). In addition, the delay of multi-affected vulnerabilities that are delegated is a median of 4 days after the disclosure date. Moreover, 4 out of the studied CNAs (Apple, IBM, Oracle, and Red Hat) release the fixes of such vulnerabilities slower

than the other studied CNAs, with the majority of fixes being released at least 90 days after the disclosure date.

The studied CNAs release prompt fixes for self-assigned vulnerabilities, while they are relatively slower for delegated vulnerabilities, which are more severe. These results suggest that some CNAs coordinate on releasing fixes while some do not, ending up with a large window (of at least 90 days) for potential vulnerability exploits related to delegated vulnerabilities in the operating-system type of product or those that affect multiple CNAs' products. Researchers should focus on better understanding and analyzing the collaboration processes for such kinds of vulnerabilities to reduce their impact.

Paper organization. Section 2 provides the background of common security specifications and multi-party coordination on fixing vulnerabilities. Section 3 discusses prior related work to our study. Section 4 describes how we design our study and obtain our studied dataset. Section 5 answers our research questions. Section 6 discusses the implications of our study. Section 7 discusses the threats to the validity of our study. Finally, Section 8 concludes the paper.

2 Background

In this section, we briefly introduce the Common Vulnerabilities and Exposures (CVE) program and National Vulnerability Database (NVD), the Common Vulnerability Scoring System (CVSS), the Common platform enumeration (CPE) specification, and the multi-party coordination model on fixing vulnerabilities.

2.1 Common Vulnerabilities and Exposures (CVE) and National Vulnerability Database (NVD)

The CVE platform [2] is one of the most popular platforms storing lists of publicly disclosed computer security vulnerabilities. Since there are many databases storing information regarding security vulnerabilities, the CVE platform assigns a CVE ID to each vulnerability to give users a way to recognize unique vulnerabilities across various databases, and coordinates the development of fixes across software vendors. Such CVE IDs are assigned by CVE Numbering Authorities (CNAs), each of which represents a large software company, security company, or research organization. The diversity of CNAs provides varied yet specific areas of expertise for different types of vulnerabilities. After a CVE ID is assigned, the CNA records concise information (e.g., ID, description) about the vulnerability and includes references, while the full details (e.g., affected products) appear in other databases, such as the NVD. Then, the new CVE record will be posted on the CVE website.

The NVD [5] is tasked with analyzing CVE records after they have been synchronized from the CVE website to the NVD, typically within an hour. After a CVE record is in the NVD, NVD analysts analyze the vulnerability based on the relevant information provided with the CVE record, such as references. The analysts also manually search any publicly available information on the internet at the time of the analysis. The analysis process ensures that any publicly available information is used to document affected vendors and products for the CVE. This information also includes CVSS scores (Section 2.2) and a specification of the affected products using the CPE standard (Section 2.3).

2.2 Common Vulnerability Scoring System (CVSS)

The CVSS [6] is an open framework for assigning scores to a vulnerability to indicate its threat level. CVSS scores are widely used by many prior studies [24, 40, 46] and large organizations and companies (e.g., Computer Emergency Response Team (CERT)) to communicate the severity of the vulnerabilities found in their products. A CVSS score is composed of three sets of metrics: base, temporal and environmental, each of which has its own underlying scoring component.

The base metrics represent the innate characteristics of a vulnerability to indicate its severity via two subscore groups: exploitability, i.e., how the vulnerability is exploited, and impact, i.e., the extent of victim system losses if the vulnerability is exploited. Computing all the subscores results into a threat score of a vulnerability. For example, a base score in the range of 4.0 to 6.9 indicates medium severity.

The temporal metrics represent metrics that can change over time. These metrics measure the current exploitability of the vulnerability, as well as the availability of mitigation plans, such as a patch.

Finally, the environmental metrics reflect additional attributes of a vulnerability in a specific environment, in terms of potential losses of a brand or a product, the proportion of vulnerable systems and the extent of impact. Note that we use CVSS 2.0 as CVSS 3.0 (released in 2015) and 3.1 (released in 2019) are inadequate for the majority of our data as we study vulnerabilities whose CVE ID was issued between 2010 and 2019.

2.3 Common platform enumeration (CPE) specification

The CPE specification[3] defines standardized methods for assigning names to software vendors and their products. For example,

"*cpe:2.3:a:microsoft:internet_explorer:8.0.6001:beta:*:*:*:*:*:**"

represents a CPE string following the CPE 2.3 specification. In this case, the vendor and its product are Microsoft Internet Explorer 8.0.6001, Beta version. The CPE specification classifies products into three types: applications (a), operating systems (o), and hardware devices (h). The "a" before "microsoft" in the CPE string indicates that Internet Explorer is an application. The string

[3] `https://cpe.mitre.org/specification/`

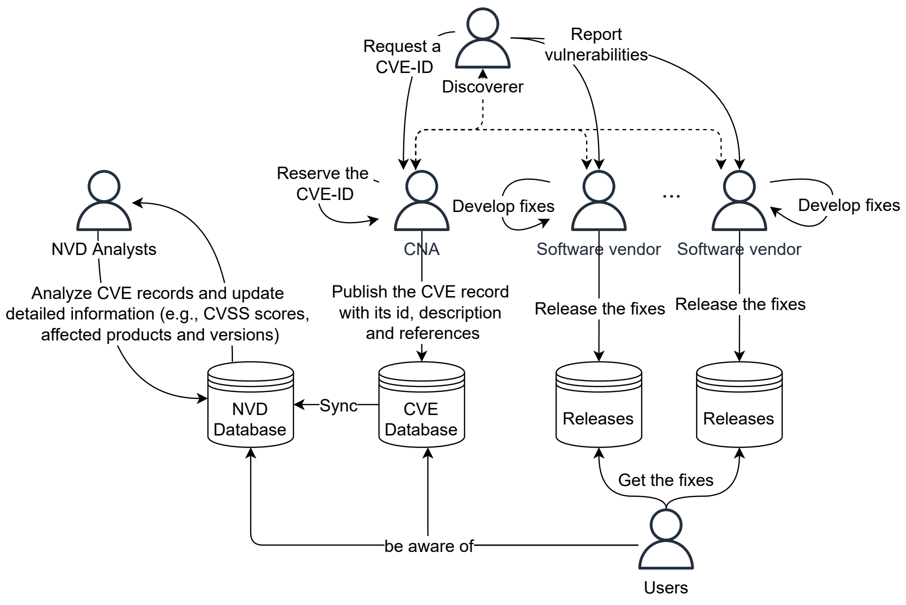

Fig. 1: Overview of multi-party coordination on fixing vulnerabilities. The dashed lines indicate possible coordination between the parties involved in fixing a vulnerability. The coordination activities include collecting information of a vulnerability, investigating root causes, discussing and developing a fix, and testing the fix. Note that software vendors can be a CNA and/or the discoverer and vice versa.

after "Beta" is a formatted string to reflect additional attributes (e.g., target hardware version is x86 or x64) of the product.

2.4 Multi-party coordination on fixing vulnerabilities

Figure 1 presents the multi-party coordination involved with fixing vulnerabilities. After a vulnerability in a product is found, the discoverer reports it to the affected software vendor(s) and coordinates with the software vendor(s) for mitigation plans. Meanwhile, the discoverer requests a CVE ID from a CNA that controls the vulnerability disclosure process. Before the disclosure, the CNA assigns and validates the vulnerability and coordinates a proper disclosure time with the discoverer and affected software vendor(s). Users typically become aware of vulnerabilities from the CVE record and get the fixes from the affected software vendors.

A CNA can also be the discoverer of a vulnerability, or could be the software vendor of a vulnerable product. In all these cases, the CNA could be the one issuing its own CVE IDs, or the CVE ID could be assigned by another CNA. Since being a CNA indicates that the CNA has the expertise of vulnerabilities in a specific area, self-assigned vulnerabilities by a CNA are not unheard of. On the one hand, self-assignment reflects an earlier awareness of a vulnerability than attackers, while on the other hand, self-assignment could be subjective and underestimate the impact of the vulnerability. By contrast, delegated vulnerabilities indicate that there is a potential delay of becoming aware of a vulnerability from the discoverer.

There is no uniform policy amongst organizations on how to disclose a vulnerability in multi-party coordination. Several corporations, organizations, and governments have been working on providing guidelines and practices for multi-party vulnerability coordination. For example, MITRE has published the practices for a Coordinated Vulnerability Disclosure (CVD) process.[4] The US' Cybersecurity and Infrastructure Security Agency (CISA) proposed important steps and factors for CVD to ensure that users and administrators receive clear and actionable information in a timely manner.[5] The European Cyber Resilience Act was published in September 2022.[6] However, only CISA suggested more concrete guidelines, such as an explicit number of days for disclosure, as stated "*CISA may disclose vulnerabilities as early as 45 days after the initial attempt to contact the vendor is made regardless of the availability of a patch or update.*" Therefore, our study aims to figure out the current practices from the 13 leading companies (CNAs) by measuring the time taken between the disclosure of vulnerabilities and the release of their fixes in the context of multi-party coordination.

3 Related Work

We discuss prior empirical studies of disclosure and patching (Section 3.1) and large-scale vulnerability analysis (Section 3.2) and highlight the difference between this study and prior studies (Section 3.3).

3.1 Disclosure and Patching

Vulnerability disclosure reflects exposing sufficient information of vulnerabilities to the public, where stakeholders can pick up the information and perform mitigation plans. Zhao et al. [47] observed from two bounty platforms, i.e., Wooyun and HackerOne, that researchers have been making significant contributions, in terms of vulnerability discovery, to secure tens of thousands of organizations on the Internet. Arora et al. [9] conducted a study to estimate the impact of vulnerability disclosure and availability of patches from both the attackers' and software vendors' perspectives. The study suggested that

[4] `https://cve.mitre.org/cve/researcher_reservation_guidelines`
[5] `https://www.cisa.gov/coordinated-vulnerability-disclosure-process`
[6] `https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act`

although software vendors quickly reacted to disclosed vulnerabilities, the frequency of attacks also increased. The study further revealed that open source vendors patched faster than closed source vendors.

Patching applications and operating systems is a common strategy to avoid vulnerability exploits. Some prior works focused on patch development. Nappa et al. [33] evaluated the impact of code reuse on patching vulnerabilities on 1,593 vulnerabilities from 10 popular client applications. The authors observed that the delay of patch releases between a reused code snippet in two applications is a median of 11 days up to 118 days. Li and Paxson [28] investigated the patch release process and characterized security patches from a diverse set of 682 open-source software projects, including more than 4k bug fixes for over 3k vulnerabilities. The study revealed that the fixes for 78.8% of all CVEs were committed on the disclosure date, while 50% of them were patched more than a week before the disclosure date. In our work, we study the delay of patch releases w.r.t the disclosure date, which is the most risky time in a vulnerability lifecycle, and characterize the influential factors related to the delay.

Another line of research is related to patch identification and propagation in various sets of open-source projects, i.e., program languages (e.g., Python [37], JavaScript [14]), devices (e.g., IoT [32], Android [17, 45]), or diverse sets of projects [14, 31, 44]. For example, Wang et al. [44] developed a machine learning model to identify hidden security patches in open-source projects, i.e., patches without explicit description in change logs or release notes. These patches could be adopted by similar types of open-source projects that have the same vulnerability and are unpatched. Our study focuses on vulnerabilities in products of top companies (CNAs) in security and investigates their practices.

3.2 Large-scale Vulnerability Analysis

Due to the increase in availability of big data technology in recent years, many prior works have conducted analyses on a large-scale set of vulnerabilities. Liu et al. [30] studied the characteristics (e.g., developer, blamed code) of vulnerabilities from the patches of 3,806 vulnerabilities in 5 popular open source projects, written in C/C++. For example, vulnerable code are usually introduced by less than 10% of developers in a project. Shahzad et al. [40] characterized vulnerabilities from a wide range of aspects, including when they were introduced, how they evolved over time and which types of vulnerabilities lead to a higher number of exploits. Feutrill et al. [18] showed that the time series of public vulnerability disclosures have a long range dependence, with burstiness, high variation, and slow convergence. Many prior work relied on their own set of vulnerabilities to develop relevant tools, e.g., detection [16, 21, 29] prediction [13, 41], notification [43], etc.

3.3 Multi-party coordination of CNAs for vulnerability fix releases and disclosure

Differentiating from the aforementioned prior work, we study the efficiency of patch releases from top companies that control the vulnerability disclosure process, as a CNA. As discussed in Section 2.4, these CNAs have expertise in security, compared to ordinary software projects/vendors, and are involved in multi-party coordination on fixing vulnerabilities.

Although the number of vulnerability exploits surges quickly after disclosure and drops quickly after patching [9, 10], in such a complex coordination model, the involved parties face many challenges (e.g., patching fixes to multiple affected products, the information leakages on social media), ending up with delays [38]. First of all, each party has its own strategy for dealing with vulnerabilities, development policy and release cycles. Second, these policies and release cycles prevent simultaneous rollout of fixes from all affected vendors at once. The more time passes since the release of the first fixed product, the greater the pressure for the affected vendors that have not yet developed or integrated a fix into their product. In this paper, we call the time between the rollout of the first and the last product impacted by a vulnerability, the vulnerability's "*patch coherence*". Moreover, when an affected vendor is not able to fix a vulnerability, the vendor seeks help from another vendor, leading to an increase in involved parties and further complicating the fixing process. Furthermore, a vulnerable project might have its own derived projects that are blocked until the vulnerable project releases the fix, again increasing the difficulty of finding a coordinated schedule and delaying the release of fixes.

Given the special role of CNAs in multi-party coordination on fixing vulnerabilities, this paper focuses on empirically understanding the coordination practices employed by the top CNAs.

4 Study Design

This section presents the approach for our empirical study to understand the coordination efficiency of CNAs as software vendors. RQ1 studies the amount of self-assigned and delegated vulnerabilities and their threat levels. RQ2 measures whether the delay of patch releases for self-assigned and delegated vulnerabilities is influenced by the type of the affected product. Finally, in order to understand the efficiency of collaboration across CNAs, RQ3 studies the delay of patch releases between CNAs that are affected by the shared vulnerabilities.

4.1 Subject CNA Selection

As discussed in Section 2.4, CNAs are responsible for assigning, validating and disclosing vulnerabilities, while software vendors are responsible for developing and releasing vulnerability fixes. To gain insights from current practices,

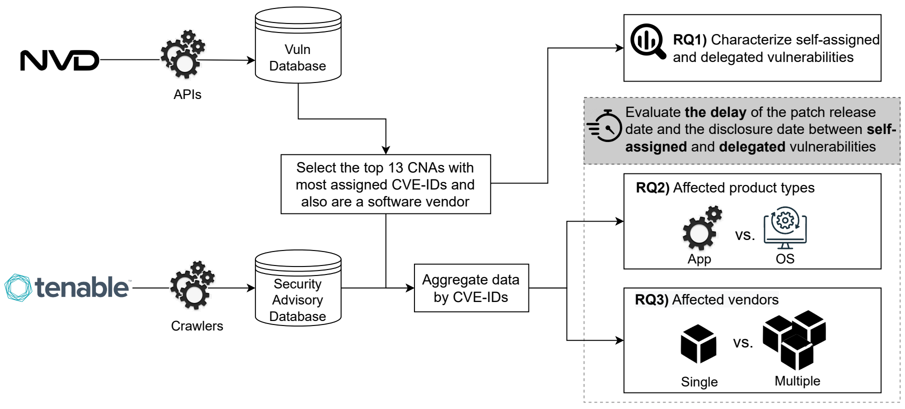
NVD
APIs
Vuln Database
Select the top 13 CNAs with most assigned CVE-IDs and also are a software vendor
tenable
Crawlers
Security Advisory Database
Aggregate data by CVE-IDs
RQ1) Characterize self-assigned and delegated vulnerabilities
Evaluate the delay of the patch release date and the disclosure date between self-assigned and delegated vulnerabilities
RQ2) Affected product types
App
vs.
OS
RQ3) Affected vendors
Single
vs.
Multiple

Fig. 2: Overview of our study approach.

we selected 13 CNAs who are also software vendors and amongst the top 20 CNAs in terms of the number of CVE IDs they assigned. Table 1 presents the top 20 CNAs along with the number of assigned CVEs, the affected product types, the month of their first assigned CVE ID, and their organization type. Table 1 shows that 14 out of the top 20 CNAs are software vendors. 4 out of 20 are organizations regarding security (i.e., MITRE and CERT), 1 is a bug bounty platform and 1 represents a group of corporations related to the Android platform. We exclude the #10 ranked CNA related to the Android platform since it involves several corporations. We unify vulnerabilities assigned by Talos (#16) with those assigned by Cisco (#6) since Talos is an intelligence group of Cisco. Finally, we obtain a total of 13 CNAs that are also software vendors for our study.

4.2 Data Collection

Figure 2 presents an overview of our study approach. Our dataset consists of vulnerability reports from the NVD and their security advisories from `tenable.com`.[7] A vulnerability report includes the information (e.g., CVE ID, description) related to a vulnerability (cf. Section 2), while a security advisory indicates how to fix the vulnerable product of a particular affected vendor. We only consider vulnerability reports with a valid CVE ID.[8] First, we extract 106,828 vulnerabilities, i.e., unique CVE IDs, that were assigned by the top 20 CNAs and whose CVE ID was issued between Jan. 2010 and Dec. 2020, before we remove 9,836 invalid vulnerabilities that were rejected, disputed or duplicated, i.e., 96,992. The numbers in the 3rd column of Table 1 present the

[7] `https://www.tenable.com/`

[8] `https://cve.mitre.org/cve/list_rules_and_guidance/correcting_counting_issues.html`

Table 1: The top 20 CNAs with the most valid CVE IDs assigned. Note that the three RQs study the bold CNAs, which are both CNAs and software vendors. Note that a vulnerability may affect several CNAs' products, i.e., both self-assigned and delegated vulnerabilities, so that the sum of the 4th and 5th columns is greater than the unique number of the studied vulnerabilities that affected the 13 CNAs, i.e., 39,409.

	CNAs	**# of CVEs**	**# of delegated CVEs**[9]	**# of self-assigned CVEs**[8]	**Affected product types**[6]	**Type**[7]
1	MITRE	44,383	3,939	0	-	NP
2	**Red Hat**	7,331	333	1,466	A.O	C
3	**Oracle**	6,076	218	5,529	A.H.O	C
4	**Microsoft**	5,786	13	5,438	A.H.O	C
5	**IBM**	4,241	209	4,113	A.H.O	C
6	**Cisco systems**	4,098	32	4,069	A.H.O	C
7	**Apple**	3,906	481	3,827	A.H.O	C
8	**Adobe**	3,583	2,716	3,224	A	C
9	CERT/CC[1]	2,635	105	0	-	N
10	Android[2]	2,607	2,331	0	-	C
11	**Qualcomm**	1,722	597	1,124	A.H.O	C
12	**Google**	1,719	576	1,361	A.H.O	C
13	ICS-CERT[3]	1,714	10	0	-	N
14	JPCERT/CC[4]	1,655	93	0	-	N
15	**Mozilla**	1,278	458	1,201	A.H.O	C
16	Talos[5]	962	0	49	A.H.O	C
17	HackerOne	920	30	0	-	B
18	**Dell**	848	31	263	A.H.O	C
19	**Intel**	770	31	669	A.H.O	C
20	**Huawei**	758	1	732	A.H.O	C

[1] A computer emergency response team (CERT) is an expert group that handles computer security incidents. CERT/CC represents the CERT Coordination Center.
[2] Android indicates a group of people associated with Google Inc. or the Open Handset Alliance.
[3] ICS-CERT represents the CERT for industrial control systems (ICS).
[4] JPCERT/CC represents the CERT in Japan.
[5] Talos is an intelligence group in Cisco.
[6] 'A' represents application product type, 'H' represents hardware and 'O' represents operating system.
[7] Organizational type: 'C' represents commercial, 'N' represents not-profit, 'B' represents bug bounty platform and 'NP' represents not-for-profit.
[8] The CNAs that are not a software vendor do not have self-assigned CVEs.
[9] Note that the delegated CVEs only affected the products of the other studied CNAs.

numbers of valid CVE IDs assigned by the top 20 CNAs. As discovering vulnerabilities takes time and developers usually take more than two years to fix them after the discovery [34], we decided to keep a safety margin of 1 year for developing vulnerability fixes between the end of our data set (Dec. 2020) and the time at which we started this work (Jan. 2022). To better understand the efficiency of vulnerability fix releases for both self-assigned and delegated vulnerabilities, we then identify delegated vulnerabilities in each CNA amongst the vulnerabilities assigned by all the 20 CNAs, while discarding the vulnerabilities that did not affect any product of the 13 studied CNAs. Finally, our dataset contains a total of 39,409 vulnerabilities.

For each of these 39,409 vulnerabilities, we extract the corresponding vulnerability report from the NVD official data feeds.[9] Such a vulnerability report includes its CVE ID, assignor (CNA), CVSS scores, published date (i.e., disclose to the public), and affected software configurations according to the CPE specification (discussed in Section 2.3).

Furthermore, we also extract patch dates from security advisories in `tenable.com`.[10] `Tenable.com` is a vulnerability exposure company that offers tools for over 30k companies around the world to protect their systems from security risks. `Tenable.com` organizes security advisories that are made by a software vendor who released a fix (e.g. security advisory, patch) for a given vulnerability. A security advisory in `Tenable.com` includes its CVE ID(s), affected software configurations (CPEs), and its patch publication date. Note that a security advisory might contain several vulnerabilities when a software vendor releases the fixes of these vulnerabilities in one security update.

We aggregate the vulnerability reports and security advisories by CVE IDs and affected software configurations (CPEs).

4.3 Efficiency measurement

To study the efficiency of CNAs in multi-party coordination on vulnerability fix releases and disclosure, we measure the efficiency of a CNA (as a software vendor) from three aspects: 1) the awareness of vulnerabilities of its own products in RQ1, 2) the delay of patch releases w.r.t. the disclosure date in RQ2 and RQ3, i.e., Diff_{pd}, and 3) the delay between the first and last patch releases across the CNAs when they are affected by a shared vulnerability in RQ3, i.e., its patch coherence Diff_{pp} (see Section 3.3).

First, as discussed in Section 2.4, since a CNA represents an expert in vulnerabilities, identifying self-assigned vulnerabilities reflects the extent of efficiency in finding vulnerabilities in its own products. In addition, self-assigned vs. delegated vulnerabilities are treated in terms of the capability of a given CNA to validate whether a vulnerability affects its own product vs. other CNAs' products.

Second, for a given CNA, the delay (Diff_{pd}) of patch releases of a vulnerability represents the time between the disclosure date of a vulnerability and

[9] `https://nvd.nist.gov/vuln/data-feeds`
[10] Note that we use tenable.com, since the open-source vulnerability database (OSVDB), which was widely used to study the patch available date for security vulnerabilities [19, 40], was shutdown permanently in 2016, `https://www.securityweek.com/osvdb-shut-down-permanently`.

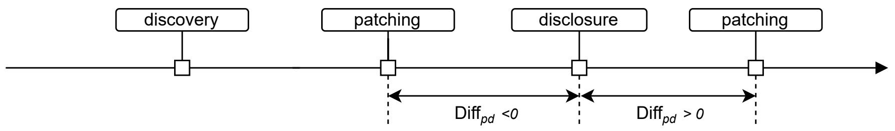

Fig. 3: Illustration of the measurement of Diff$_{pd}$. The Diff$_{pd}$ of a vulnerability is calculated by subtracting the disclosure date from the patch (available) date. A negative value (Diff$_{pd}$ <0) indicates the patch is available before the disclosure date.

the availability of a patch for it, as shown in Figure 3. The most risky vulnerabilities are those that are disclosed to the public without an available fix [40], in which case the Diff$_{pd}$ is positive. In fact, in many cases, an exploitation method is provided when a vulnerability is disclosed [39], enabling anyone to exploit the vulnerability. In other words, the delay Diff$_{pd}$ indicates the size of the window during which attackers could actively exploit the vulnerability at a large scale. A negative value of Diff$_{pd}$ indicates that a fix (patch) was released before the disclosure date of the vulnerability, which is the optimal case for users to avoid vulnerability exploits. The larger the value of Diff$_{pd}$, the more risky a victim system is.

We compare the delay (Diff$_{pd}$) from several aspects within one CNA and across the studied CNAs: self-assigned vs. delegated, affected product types, and the number of affected vendors (CNAs). We detail our approaches in the section of each RQ.

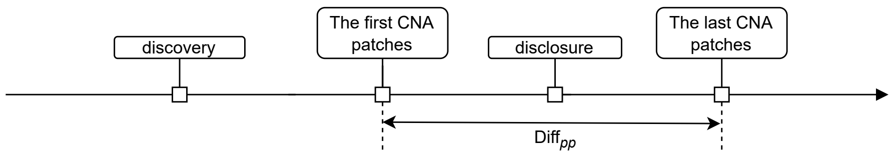

Fig. 4: Illustration of the measurement of Diff$_{pp}$ (patch coherence).

Lastly, as a vulnerability could affect several CNAs' products, these CNAs in principle should coordinate vulnerability fix releases and disclosure. We measure such coordination across the affected CNAs by the metric of "*patch coherence*", which is the delay (Diff$_{pp}$) between the first and last patch releases across the CNAs, as shown in Figure 4. A small value of patch coherence indicates a more consistent speed of patch releases from the affected vendors, suggesting that users of all affected products become protected in the same period. We compare the delay (Diff$_{pp}$) from several aspects: self-assigned vs. delegated, vulnerability weaknesses, and pairs of affected CNAs. We detail our approaches in RQ3.

5 Results

5.1 RQ1: What are the characteristics of self-assigned vs. delegated vulnerabilities?

Motivation: Given the rise of security issues within software supply chains, fixing vulnerabilities requires multi-party coordination (as discussed in Section 2.4). Delegated vulnerabilities indicate not only a potential delay of becoming aware of the vulnerabilities, but also indicate that users might wait for a longer time to get the fixes. This increases the chance of vulnerability exploits and results in disruption in software supply chains and promoting conflicts between their users. Understanding the characteristics of self-assigned and delegated vulnerabilities gives insights to software vendors when developing fixes and to users on securing their systems.

Approach: We measure the prevalence of self-assigned and delegated vulnerabilities by calculating the number and percentage of vulnerabilities in each group for each of the 13 CNAs. To analyze the characteristics of self-assigned and delegated vulnerabilities, we investigate the affected product types (i.e., application, hardware and operating system) from the CPE configurations, the severity level using the CVSS 2.0, and the assignor-assignee pairs of the CNAs.

Results: 12 out of the 13 studied CNAs (i.e., all except for Google) have 3 up to 49 times as many self-assigned vulnerabilities as delegated ones. Google instead has a majority (77%) of delegated vulnerabilities. In general, the CNAs seem to dedicate substantial effort on their own products. More specifically, Figure 5 indicates that Apple, Microsoft and Red Hat are the CNAs with a similar percentage of self-assigned and delegated vulnerabilities, i.e., Apple: 53% vs. 47%, Microsoft: 53% vs. 47%, Red Hat: 50% vs. 50%.

Mitre (not one of the 13 CNAs) has assigned vulnerabilities to all the 13 CNAs. Figure 6 shows the heat map of the assignor and assignee pairs of the 13 CNAs (and Mitre) for delegated vulnerabilities. In terms of being an assignor, Mitre is the #1 CNA followed by Red Hat and Dell, who assigned vulnerabilities to 11 and 8 CNAs, respectively. Mitre has focused on software security for more than two decades and won several awards and notable achievements.[11] Red Hat[12] and Dell[13] have a sterling reputation in software security. Adobe assigned vulnerabilities to 6 CNAs and has assigned the largest number of vulnerabilities to Microsoft (2,700) and Apple (2,516) among all the assignors.

On the other hand, Red Hat, Google and Apple are the assignee of 12 CNAs, followed by Microsoft who is the assignee of 11 CNAs. Among

[11] https://www.mitre.org/about/awards-and-recognition
[12] https://www.redhat.com/ja/about/press-releases/press-bestlinuxsolutions
[13] https://www.dell.com/en-us/blog/dell-software-security-solutions-of-awards-and-honors/

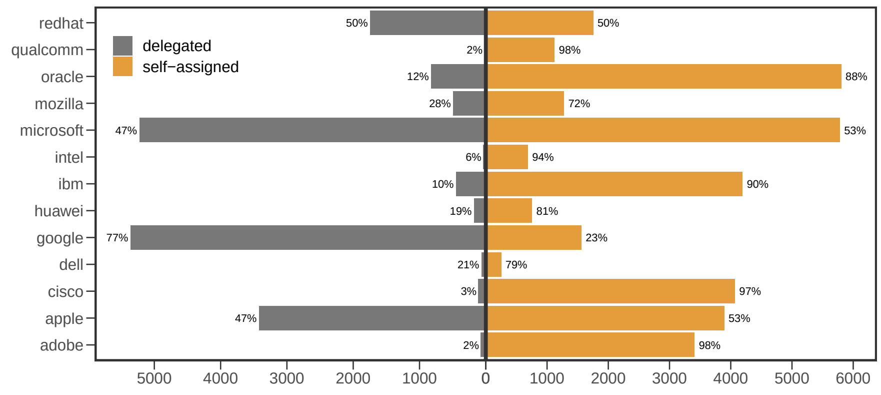

Fig. 5: The number and percentage of self-assigned vs. delegated vulnerabilities in the 13 CNAs.

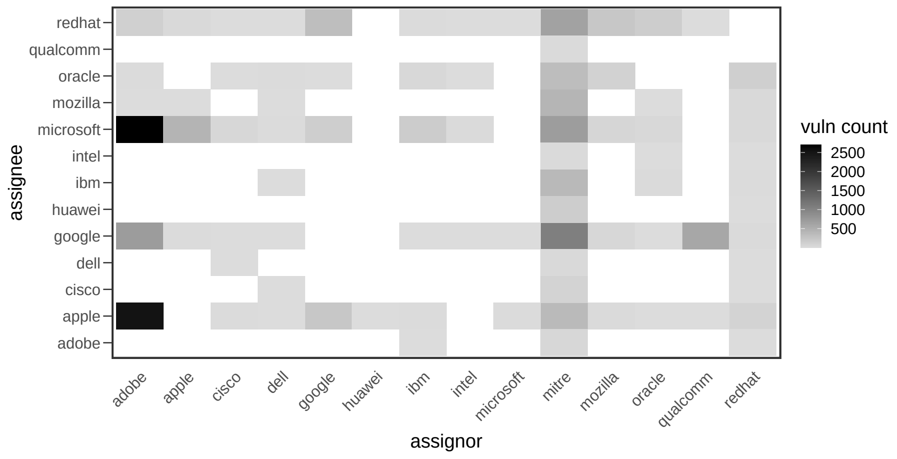

Fig. 6: Heat map of the assignor-assignee pairs based on the number of delegated vulnerabilities. Note that Mitre is the #1 assignor but is not included in the studied CNAs.

these four CNAs, Red Hat has the smallest number of delegated vulnerabilities (1,555), compared to the other three CNAs (Google: 2,481, Apple: 3,296, Microsoft: 4,418). Red Hat and Google have the largest number of vulnerabilities delegated from Mitre, while Apple and Microsoft get them from Adobe. In addition, Qualcomm only has one assignor, i.e., Mitre.

For the application type of product, the 12 CNAs have 2 up to 49 times as many self-assigned vulnerabilities as delegated vulnerabilities. For the operating-system type of product, Google, Microsoft, Red Hat, and Oracle have a majority of delegated vulnerabilities (98%, 58%, 67%, and 55%, respectively), while Apple has a similar

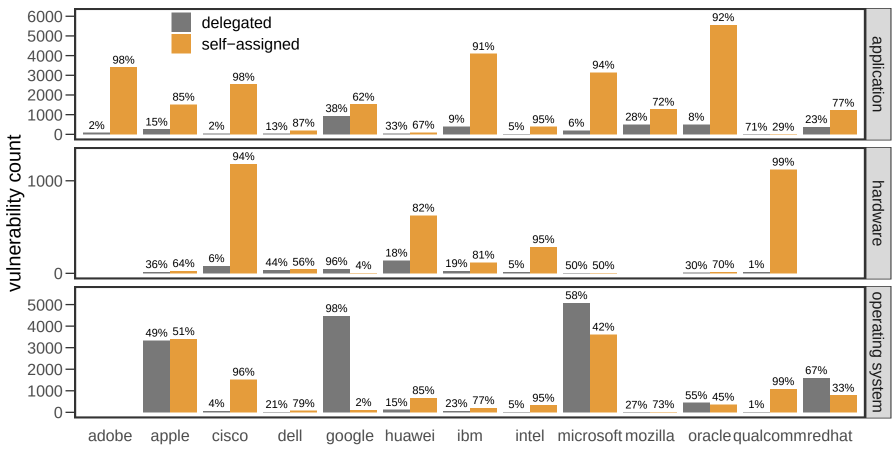

Fig. 7: The number and percentage of self-assigned vs. delegated vulnerabilities in the 13 CNAs, based on the three product types, i.e., application, hardware and operating-system.

percentage of delegated (49%) and self-assigned (51%) vulnerabilities. In general, the application type of product is responsible for the largest percentage (47%) of vulnerabilities across the CNAs, while the hardware type of product accounts for the smallest percentage (18%) of vulnerabilities. Both application and hardware types of products have a median of 3 vulnerabilities across the CNAs. The unique number of affected hardware products (i.e., 9,364) of the studied affected vendors is approximately three times as high as that of affected application products (i.e., 3,346). Figure 7 shows the percentages of self-assigned and delegated vulnerabilities in the three product types for each of the studied CNAs. In 12 out of the 13 CNAs, the majority (at least 62%) of vulnerabilities affecting the application type of product are self-assigned, compared to 8 out the 13 CNAs for the operating-system type of product.

Delegated vulnerabilities are more severe than self-assigned vulnerabilities. For the application type of product, delegated vulnerabilities have a median CVSS score of 7.5, i.e., high-severity, while self-assigned vulnerabilities have a median CVSS score of 6.8 (medium-severity) in Figure 8. The difference between delegated and self-assigned vulnerabilities is statistically significant by a Wilcoxon test, with a p-value < 0.01 ($\alpha = 0.01$). The effect size is small (0.26). For the operating-system type of product, although delegated and self-assigned vulnerabilities both have a median CVSS score of 7.1, delegated vulnerabilities have higher first and third quartiles than self-assigned vulnerabilities, i.e., both the top and bottom half of the distribution of delegated vulnerabilities are more severe than that of self-assigned vulnerabilities.

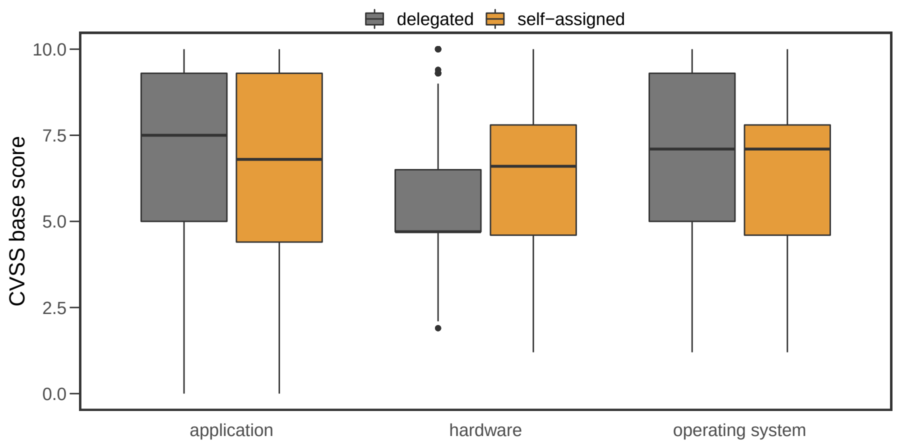

Fig. 8: For the application type of products, delegated vulnerabilities have a higher CVSS 2.0 score than self-assigned vulnerabilities.

Summary of RQ1

12 out of the 13 studied CNAs have a higher percentage of self-assigned vulnerabilities than delegated vulnerabilities, though 3 of them have a similar percentage of both types of vulnerabilities. The studied CNAs have a majority of self-assigned vulnerabilities for the application type of product, compared to the hardware and operating-system types of products. However, delegated vulnerabilities are more severe than self-assigned vulnerabilities for the application and operating-system types of products across the CNAs.

5.2 RQ2: How fast do the CNAs develop fixes for self-assigned vs. delegated vulnerabilities?

Motivation: The results in RQ1 reveal that the studied CNAs have a majority of self-assigned vulnerabilities. However, delegated vulnerabilities have a higher threat level, suggesting a higher impact when such a vulnerability is disclosed without an available fix. We aim to understand the speed of releasing vulnerability fixes in these CNAs, the duration of the period users are at risk, and whether the threat level of a vulnerability influences the speed of patch releases.

Approach: As discussed in Section 4.3, we compute the delay (Diff$_{pd}$) of patch releases for each vulnerability. To measure the speed of patch releases, we classify the delay into 5 duration categories (w.r.t. the disclosure date): $<$ -30, $[-30, 0)$, $[0, 30)$, $[30, 90)$, and $\geqslant$ 90 days and calculate the number of self-assigned and delegated vulnerabilities in each category for the studied CNAs.

These five categories represent earlier-than-one-month before the disclosure date, within-one-month before the disclosure date, within-one-month after the disclosure date, one-quarter after the disclosure date, and longer-than-one-quarter after the disclosure date.

We compare the Diff_{pd} delay between the **application** and **operating-system** types of products, since the hardware type of product only accounts for the minority (13%) of vulnerabilities and `tenable.com` does not have security advisories for vulnerabilities that affected this type of product. Note that Qualcomm has insufficient data in `tenable.com` because it has the majority of its vulnerabilities in the hardware type of product (as shown in Figure 7). Therefore, we only study 12 out of the 13 CNAs in this RQ. In addition, we compute the CVSS scores of the vulnerabilities for the two product types.

Results: For the *application* type of product, the CNAs release fixes for delegated vulnerabilities at a slower pace than self-assigned vulnerabilities by a median of 1 day, yet typically still before disclosure. In particular, self-assigned vulnerabilities are fixed in a median of 2 days before the disclosure date across the CNAs, while delegated vulnerabilities are fixed in a median of 1 day before the disclosure date. The difference of the delay is significant by a Wilcoxon test, with a p-value < 0.01 and a medium effect size of 0.53.

One possible reason why patches for delegated vulnerabilities still appear before the disclosure date is that, even though there is a potential delay of becoming aware of vulnerabilities, the vulnerability has a low severity and can be fixed easily. Another possible reason is that the CNA (as a coordinator) and the affected software vendors agreed on a schedule to only disclose the vulnerability once the fixes are released. It is also possible that a vulnerability prompts the affected vendor to re-design certain functionalities in the vulnerable product, then only to publicly disclose upon release of the first re-design product version (even if this causes substantial delays). For example, Adobe was aware of CVE-2016-4271 approximately six months before the disclosure date.[14] This CVE was a flaw in Adobe Flash allowing to bypass the local sandbox and exfiltrate important data on Windows (e.g., user credentials). Adobe redesigned Flash Player 23 to no longer use a sandbox.

For the application type of product, all the studied CNAs release the fixes for most of the **self-assigned** vulnerabilities within 30 days before the disclosure date, while 2 of the CNAs release fixes for most of the **delegated** vulnerabilities at least 90 days after the disclosure date ($\geqslant$ 90 days). Table 2 shows all details of the 5 duration categories of the Diff_{pd} delay for self-assigned and delegated vulnerabilities. In general, the CNAs are anticipated to deal better with self-assigned vulnerabilities than delegated vulnerabilities in terms of the delay of patch releases.

Although there is a potential delay of becoming aware of delegated vulnerabilities, 6 out of the 12 CNAs release the fixes for most of their delegated

[14] https://blog.bjornweb.nl/2017/02/flash-bypassing-local-sandbox-data-exfiltration-credentials-leak/

Table 2: The number of self-assigned and delegated vulnerabilities that affect the **application** type of product for the 5 duration (days) categories of the Diff_{pd} delay for the studied CNAs. For each row, the bold number indicates the category with the largest number of respective vulnerabilities among the 5 duration categories.

CNA	**Assignor**	$<$ **-30**	**-30∼0**	**0∼30**	**30∼90**	$\geqslant$ **90**
Adobe	Self-assigned	447	**2,609**	366	97	12
	Delegated	5	**33**	8	3	3
Apple	Self-assigned	177	**686**	358	160	114
	Delegated	3	33	26	41	**88**
Cisco	Self-assigned	55	**380**	138	5	6
	Delegated	**1**	-	-	-	-
Dell	Self-assigned	1	**3**	-	-	-
	Delegated	**2**	1	1	-	-
Google	Self-assigned	586	**856**	136	3	-
	Delegated	48	**576**	67	2	-
Huawei	Self-assigned	-	-	-	-	-
	Delegated	-	-	1	1	1
IBM	Self-assigned	141	**451**	101	84	101
	Delegated	42	**66**	22	22	25
Intel	Self-assigned	-	-	-	-	-
	Delegated	-	**2**	-	-	-
Microsoft	Self-assigned	16	**1,104**	401	11	7
	Delegated	1	**10**	7	8	5
Mozilla	Self-assigned	584	**731**	97	25	1
	Delegated	85	**343**	47	6	5
Oracle	Self-assigned	226	**2,775**	730	94	208
	Delegated	31	26	44	66	**175**
Red Hat	Self-assigned	144	**234**	94	60	99
	Delegated	10	34	46	**79**	67

vulnerabilities within 30 days before disclosure. On the other hand, 1 CNA (i.e., Red Hat) requires 30∼90 days after disclosure (median of 42 days), and 2 CNAs (i.e., Apple and Oracle) even more than 90 days after disclosure (median of 117 and 120 days, respectively). Due to insufficient vulnerabilities for Dell, Intel and Huawei, we encourage future studies to collect more data to analyze the generalizability of our results.

For the *operating-system* type of product, the CNAs again release fixes for delegated vulnerabilities at a slower pace than self-assigned vulnerabilities, by a median of 5 days. More specifically, the fixes for delegated vulnerabilities are released in a median of 3 days after the disclosure date, while the fixes for self-assigned vulnerabilities are released in a median of -2 days before the disclosure date. Table 3 indicates that 8 out of the 12 CNAs release fixes for most of the **self-assigned** vulnerabilities within 30 days before the disclosure date (-30∼0 days), 1 CNA (Mozilla) more than 30 days before the disclosure date ($<$ -30 days), and 1 CNA (Red Hat) more

Table 3: The number of self-assigned and delegated vulnerabilities that affect the **operating-system** type of product for the 5 duration (days) categories of the Diff$_{pd}$ delay for the studied CNAs. For each row, the bold number indicates the category with the largest number of respective vulnerabilities among the 5 duration categories.

CNA	**Assignor**	$<$ **-30**	**-30∼0**	**0∼30**	**30∼90**	$\geqslant$ **90**
Adobe	Self-assigned	-	-	-	-	-
	Delegated	-	-	-	-	-
Apple	Self-assigned	1,086	**1,861**	223	231	70
	Delegated	32	33	27	94	**112**
Cisco	Self-assigned	27	**338**	281	4	39
	Delegated	**2**	1	-	-	-
Dell	Self-assigned	-	-	-	-	-
	Delegated	**2**	-	-	-	-
Google	Self-assigned	3	**10**	-	-	-
	Delegated	11	**23**	-	-	-
Huawei	Self-assigned	4	**6**	-	-	-
	Delegated	**2**	-	-	-	-
IBM	Self-assigned	11	**22**	2	-	1
	Delegated	3	**6**	2	-	1
Intel	Self-assigned	-	**1**	-	-	-
	Delegated	-	-	-	-	-
Microsoft	Self-assigned	7	**2,031**	616	25	5
	Delegated	5	**206**	89	13	3
Mozilla	Self-assigned	**3**	1	-	-	-
	Delegated	**3**	-	-	-	-
Oracle	Self-assigned	1	**90**	56	13	15
	Delegated	1	4	8	18	**30**
Red Hat	Self-assigned	4	6	6	10	**42**
	Delegated	-	5	25	31	**89**

than 90 days after the disclosure date ($\geqslant$ 90 days). In contrast, for **delegated** vulnerabilities, 3 out of the 12 CNAs release fixes for the largest number of vulnerabilities within 30 days before the disclosure date (-30∼0 days), 3 CNAs (i.e., Apple, Oracle, and Red Hat) are more than 90 days after the disclosure date ($\geqslant$ 90 days), and 4 CNAs are more than 30 days before the disclosure date ($<$ -30 days), though the numbers for these 4 CNAs are relatively small.

The delay (Diff$_{pd}$) of patch releases for delegated vulnerabilities that affect the *operating-system* type of product is larger than for those that affect the *application* type of product, by a median of 4 days. In particular, only for high-severity **delegated** vulnerabilities, the delay (Diff$_{pd}$) of patch releases for the operating-system type of product is similar to the application type of product (Wilcoxon test: p-value = 0.23) in Figure 9. For medium-severity **delegated** vulnerabilities, the delay (Diff$_{pd}$) of patch releases for the operating-system type of product is longer than the application type of product by a median of 18 days (Wilcoxon test: p-value $<$ 0.01), with

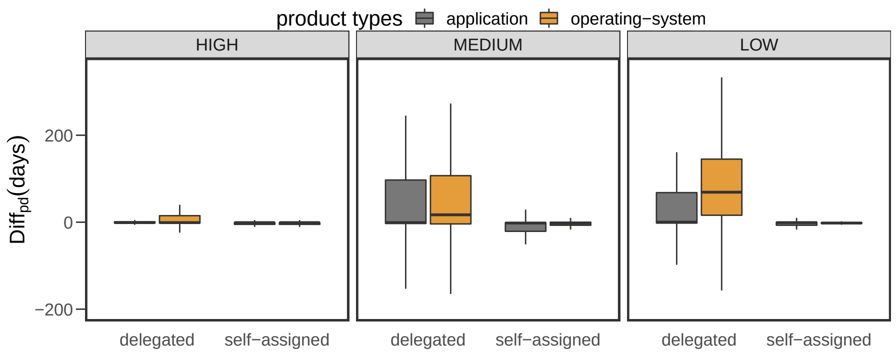

Fig. 9: The comparison of the delay (Diff_{pd}) of releasing patches between the severity levels (i.e., high, medium and low), self-assigned vs delegated, and the product types across the CNAs.

a negligible effect size of 0.04. For low-severity **delegated** vulnerabilities, the gap of the delay (Diff_{pd}) between the two product types is a median of 69 days (Wilcoxon test: p-value < 0.01), with a small effect size of 0.3. Hence, the Diff_{pd} delay of delegated vulnerabilities in operating-systems-related products interact with the severity level of the vulnerabilities. The higher the severity of a delegated vulnerability, the smaller the Diff_{pd} delay, i.e., a quicker fix.

On the other hand, for **self-assigned** vulnerabilities, the delay of vulnerabilities that affect the application type of product is similar to those that affect the operating-system type of product across all severity levels (i.e., high, medium, and low). According to a Chi-square test, the severity levels and assignor (i.e., self-assigned or delegated) of a vulnerability are two independent variables (p-value < 0.05). When a vulnerability in a product is found by the assignor (i.e., the product owner), the Diff_{pd} delay is similar across all severity levels for both product types, typically close to the disclosure date.

Summary of RQ2

Across the two product types, the studied CNAs release fixes for self-assigned vulnerabilities faster than for delegated vulnerabilities. The delay (Diff_{pd}) of patch releases for self-assigned vulnerabilities does not correlate with the vulnerabilities' severity level, while for delegated vulnerabilities, the lower the severity level, the slower the patch is released. Furthermore, the gap of the delay between the severity levels in vulnerable operating-systems-type products is larger than in application-type products.

5.3 RQ3: How effectively do the CNAs coordinate to release fixes for shared vulnerabilities?

Motivation: The results in RQ1 and RQ2 indicate that delegated vulnerabilities are more severe than self-assigned vulnerabilities and the delay of patch releases for delegated vulnerabilities is longer than self-assigned vulnerabilities. Furthermore, delegated vulnerabilities indicate that there exists a potential delay before the affected CNAs are aware of these vulnerabilities, unless CNA coordination is able to mitigate it. In order to measure the coordination efficiency on fixing vulnerabilities, we focus on vulnerabilities that affect more than one CNA's products, which we refer to as *multi-affected vulnerabilities*.

Approach: To know the extent of efficiency of coordination across the CNAs, we first compare the number and percentage of vulnerabilities that affected more than one CNA's products to those that affect only one CNA (i.e., *single-affected vulnerabilities*) for the 5 duration categories of RQ2. We compare the three CVSS score metrics (i.e., base, impact, and exploitability) between multi-affected and single-affected vulnerabilities. Similar to RQ2, we compute the Diff_{pd} delay and compare the delay between multi-affected and single-affected vulnerabilities from two aspects: (1) self-assigned vs. delegated and (2) the affected product types (i.e., application vs. operating system vs. both). Vulnerabilities affecting products of both types indicate more complex issues for which the affected vendors need a longer time to (jointly) develop a fix, potentially having a higher impact on victim systems. Similar to RQ2, due to insufficient data for hardware-type products in `tenable.com`, we only study 12 out of the 13 CNAs (exclude Qualcomm).

In addition, as discussed in Section 4.3, we evaluate coordination efficiency across the CNAs by the Diff_{pp} delay between patch releases for shared vulnerabilities, i.e., *patch coherence*. In particular, we compute the Diff_{pp} delay for each of the multi-affected vulnerabilities across the studied CNAs. A large Diff_{pp} indicates that the CNAs are less effective in coordinating vulnerability fixes across their products, since the fixes are spread out across time, with some affected products being prone to exploitation of a vulnerability for a much longer time.

We analyze the delay (Diff_{pp}) between the first and last patch dates from the following three aspects:

1) **The number of affected variants**: A variant represents a flavour of a product designed for particular conditions. For example, Cisco Catalyst's campus LAN switches have several variants, such as the Catalyst 6500 series and Catalyst 7600 series that support different network layers. A higher number of affected variants indicates that the affected vendors need more time to adopt the fixes to all the affected variants of a particular product.
2) **Vulnerability weaknesses**: Since every vulnerability corresponds to a variety of weaknesses on a victim system, we study whether the delay of a particular type of weakness is longer than the others. We identify the

top 10 weaknesses in the multi-affected vulnerabilities (see Table 5), based on the CWE (Common Weakness Enumeration) framework [3] that has been widely used in prior work [12, 46], and compare the delay among the top 10 CWEs. The CWE framework lists common security weaknesses with their information, including description, code examples, detection methods, and potential mitigation. The CWE framework also classifies associated weaknesses in groups that developers and practitioners could then use as a common language for vulnerability discussions.

3) **Pairs of studied CNAs**: We are interested in whether certain pairs of CNAs have a closer coordination of vulnerabilities than others. As the results in RQ2 indicate that the Diff$_{pd}$ delay varies across the CNAs (see Tables 2 and 3), we classify pairs of the studied CNAs that exhibit a similar pattern of releasing fixes. For example, a pair of CNAs that usually release fixes before the disclosure is more likely to coordinate on a fix, achieving better (lower) patch coherence. We consider any pairs from Adobe, Google, Microsoft and Mozilla as a group, which we refer to as "AGMM", since these CNAs release fixes of most of both types (i.e., self-assigned and delegated) of vulnerabilities across the two product types before disclosure (Tables 2 and Table 3). By contrast, Apple, Oracle and Red Hat have an inconsistent speed of releasing fixes, which we group together as "AOR". We assign the remaining pairs as "others". In addition, the AGMM and AOR groups represent two extreme groups of CNAs: one of them is quicker with releasing fixes, while the other is slower, yielding a lower and upper bound of patch coherence across the studied CNAs.

Results: Adobe (80%), Apple (53%) and Red Hat (75%) are impacted by the majority of multi-affected vulnerabilities. Apple and two other CNAs (Microsoft and Mozilla) have a similar percentage (≈50%) of single-affected and multi-affected vulnerabilities, i.e., Apple: 53% vs. 47%, Microsoft: 46% vs. 54% and Mozilla: 49% vs. 51%. Figure 10 indicates that the other 9 out of 12 studied CNAs have a majority (at least 69%) of single-affected vulnerabilities.

The 5,339 multi-affected vulnerabilities that affect both product types are more severe than (1) those that affect one product type and (2) single-affected vulnerabilities (Kruskal-Wallis test: p-value < 0.05). More specifically, Figure 11 shows that multi-affected vulnerabilities that affect the two product types have a median base score of 9.3, a median exploitability score of 8.6 and a median impact score of 10, which suggests an extremely high-severity level. By contrast, multi-affected vulnerabilities that only affect one type of product (i.e., application or operating-system) are less severe than single-affected vulnerabilities, with median base scores of 5.8 and 5, respectively. Such vulnerabilities have a similar median score across the three metrics of CVSS scores in Figure 11 (grey bars).

The delay (Diff$_{pd}$) of patch releases for multi-affected vulnerabilities is longer than for single-affected vulnerabilities, especially for delegated ones that are also multi-affected. For **delegated** vulnerabili-

Table 4: The comparison of the delay Diff_{pd} of patch releases between single-affected vs. multi-affected and delegated vs. self-assigned vulnerabilities for the studied CNAs. For each row, the bold number indicates the category with the largest number of vulnerabilities among the 5 duration categories.

CNA	**Assignor**	**Type**	**< -30**	**-30∼0**	**0∼30**	**30∼90**	**⩾ 90**
Adobe	Self-assigned	Single	26	**515**	35	3	5
		Multi	421	**2,094**	331	94	7
	Delegated	Single	3	**18**	1	2	2
		Multi	2	**15**	7	1	1
Apple	Self-assigned	Single	1,087	**2,150**	458	346	149
		Multi	176	**397**	123	45	35
	Delegated	Single	11	**36**	19	15	15
		Multi	24	30	34	120	**185**
Cisco	Self-assigned	Single	80	**723**	414	9	44
		Multi	4	**11**	10	-	1
	Delegated	Single	-	1	**4**	-	-
		Multi	1	1	-	-	-
Dell	Self-assigned	Single	-	1	**3**	-	-
		Multi	-	-	-	-	-
	Delegated	Single	4	1	-	4	-
		Multi	-	-	1	-	-
Google	Self-assigned	Single	260	**431**	83	1	-
		Multi	329	**435**	53	2	-
	Delegated	Single	27	**293**	42	1	-
		Multi	32	**306**	25	1	-
Huawei	Self-assigned	Single	4	**6**	-	-	-
		Multi	-	-	-	-	-
	Delegated	Single	**2**	-	-	-	-
		Multi	-	-	1	1	1
IBM	Self-assigned	Single	137	**422**	95	72	86
		Multi	15	**51**	8	12	16
	Delegated	Single	42	**65**	22	22	17
		Multi	3	7	2	-	**9**
Intel	Self-assigned	Single	-	**1**	-	-	-
		Multi	-	-	-	-	-
	Delegated	Single	-	**1**	-	-	-
		Multi	-	**1**	-	-	-
Microsoft	Self-assigned	Single	21	**3,126**	1,013	36	11
		Multi	2	**9**	4	-	1
	Delegated	Single	1	**17**	5	13	6
		Multi	5	**199**	91	8	2
Mozilla	Self-assigned	Single	231	**367**	40	21	-
		Multi	356	**365**	57	4	1
	Delegated	Single	69	**251**	15	3	2
		Multi	19	**92**	32	3	3
Oracle	Self-assigned	Single	132	**2,251**	436	59	118
		Multi	95	**614**	359	48	105
	Delegated	Single	6	**8**	2	3	6
		Multi	26	22	50	81	**199**
Red Hat	Self-assigned	Single	106	**174**	34	24	25
		Multi	42	66	66	46	**116**
	Delegated	Single	2	**4**	1	3	2
		Multi	8	35	70	107	**154**

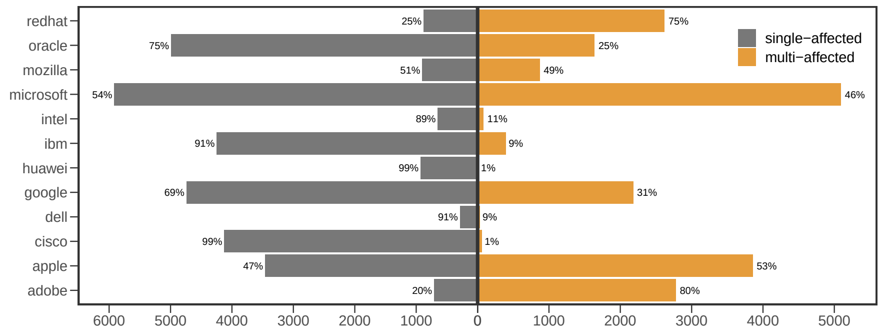

Fig. 10: The number and percentage of single-affected and multi-affected vulnerabilities in each of the studied CNAs.

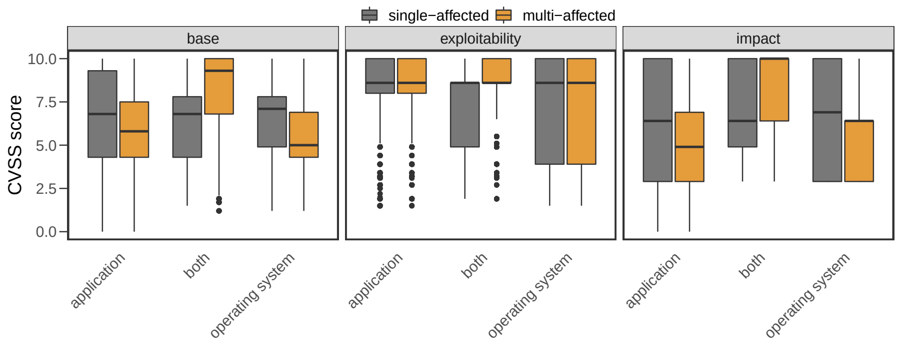

Fig. 11: The three metrics of CVSS scores for single-affected and multi-affected vulnerabilities that affect one (i.e., either application or operating-system) or two types (i.e., both) of products.

ties, the CNAs release fixes for **multi-affected** vulnerabilities in a median of 4 days (after the disclosure date), while they release fixes for **single-affected** vulnerabilities in a median of -2 days (before the disclosure date). For **self-assigned** vulnerabilities, the CNAs release fixes for both **single- and multi-affected** vulnerabilities in a median of -2 days (before the disclosure date). For an individual CNA, similar to the results for the application and operating-system types of products in RQ2 (Table 2 and 3), IBM, Apple, Oracle, and Red Hat release fixes for the largest number of **multi-affected and delegate** vulnerabilities as late as at least 90 days after the disclosure date ($\geqslant$ 90 days) in Table 4, which is slower than the other studied CNAs.

The Diff$_{pp}$ delay (i.e., patch coherence) for multi-affected vulnerabilities that affected less than 5 variants of both product types is the shortest (median of 2 days), compared to vulnerabilities that affected one type of product (median of 186 days for the application type, and 334 for the operating system type), as shown in Figure 12.

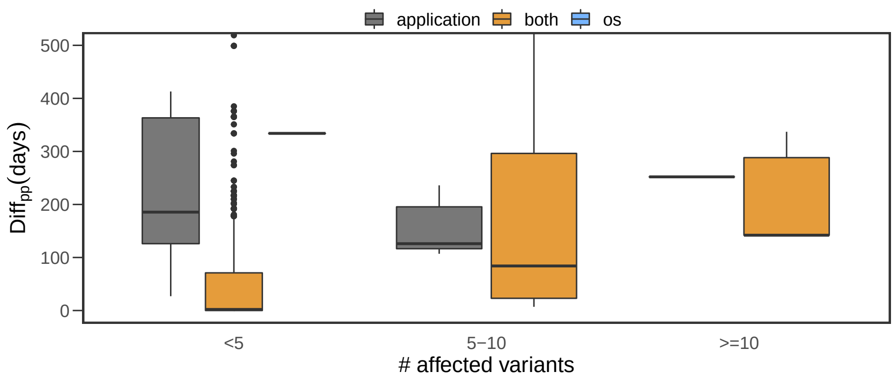

Fig. 12: Comparison of the delay (Diff$_{pp}$) between when the first and last patches are available across the studied CNAs, based on affected product types and the number of affected variants. A variant represents a flavour of a product designed for particular conditions (e.g., x84 or x64).

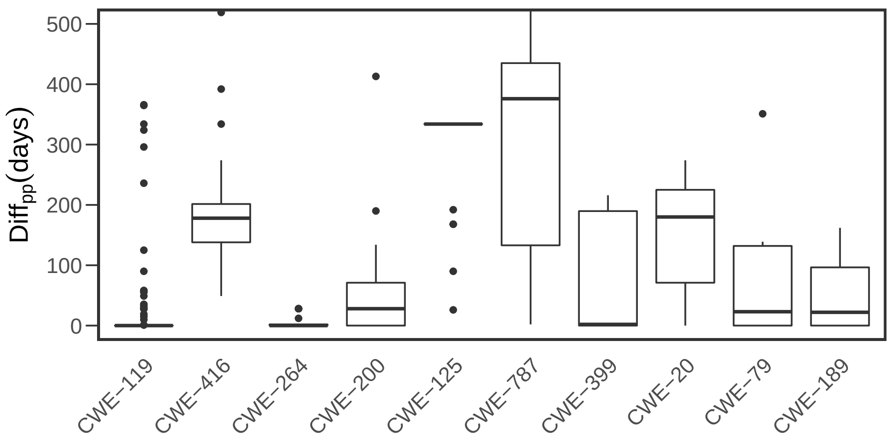

Fig. 13: The comparison of the delay (Diff$_{pp}$) between when the first and last patch dates, base on the top 10 vulnerability weaknesses (CWEs) in the multi-affected vulnerabilities. Table 5 reflects the details of the top 10 CWEs.

One possible reason for this is that multi-affected vulnerabilities that affected both types of products are more severe (see Figure 11). In general, the Diff$_{pp}$ delay of multi-affected vulnerabilities is a median of 35 days across the CNAs. Figure 12 also indicates that the more products affected by a multi-affected vulnerability, the longer the patch coherence.

The Diff$_{pp}$ delay (i.e., patch coherence) for vulnerabilities with weaknesses related to buffer overflow (CWE-119) and permissions, privileges, and access controls (CWE-264) is the shortest (median of

Table 5: The top 10 common weaknesses (CWEs) in the multi-affected vulnerabilities.

#	CWE	Name	Count
1	CWE-119	Improper restriction of operations within the bounds of a memory buffer	142
2	CWE-416	Use after free	60
3	CWE-264	Permissions, privileges, and access controls	27
4	CWE-200	exposure of sensitive information to an unauthorized actor	25
5	CWE-125	Out-of-bounds read	25
6	CWE-787	Out-of-bounds write	21
7	CWE-399	Resource management errors	16
8	CWE-20	Improper input validation	14
9	CWE-79	Cross-site scripting	11
10	CWE-189	Numeric errors	11

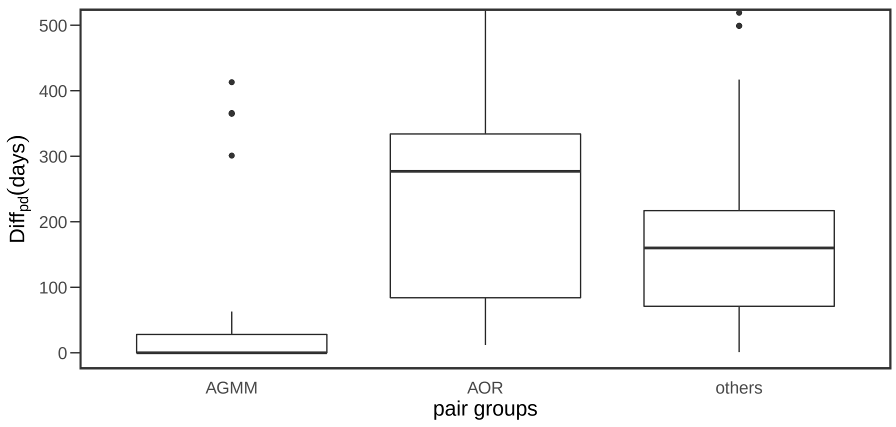

Fig. 14: Comparison of the delay (Diff$_{pp}$) between the first and last patches. “AGMM” represents any pair of vendors from Adobe, Google, Microsoft, and Mozilla. “AOR” represents any pair of vendors from Apple, Oracle, and Red Hat.

0 days), as shown in Figure 13. In general, the Diff$_{pp}$ delay of multi-affected vulnerabilities is a median of 35 days across the CNAs. Table 5 shows the top 10 weaknesses in the multi-affected vulnerabilities, along with their name and count. Four (CWE-119, CWE-416, CWE-125, CWE-787) out of the top 10 CWEs are memory errors, four (CWE-264, CWE-200, CWE-20, CWE-189) are related to design flaws, two (CWE-399, CWE-79) are related to resource management errors.

Although some CWEs are related to a similar issue, the patch coherence of such related vulnerability types can still vary substantially. For example, vulnerabilities with CWE-119 (#1) “buffer overflow” and CWE-416 (#2) “use after free" are memory errors that both could be detected by static analysis tools and even be automatically fixed [27]. However, the patch coherence for CWE-119 is a median of 0 days, while for CWE-416 is a median of 178 days (Wilcoxon test: p-value < 0.01, a large effect size of 1.32).

When any pair of CNAs from Adobe, Google, Microsoft, and Mozilla (i.e., the AGMM group) are affected by a common vulnerability, the Diff$_{pp}$ delay (i.e., patch coherence) is a median of 0 days, while for any pairs of CNAs from Apple, Oracle, and Red Hat (i.e., the AOR group), the Diff$_{pp}$ delay is a median of 277 days. The difference between the two groups of CNAs is significant (Wilcoxon test: p-value < 0.01) and the effect size is large (1.94). Figure 14 shows a large variance in the AOR group, compared to the variance in the AGMM group. This could be due to the security update policy of each CNA, which may vary by product. For example, in the AOR group, Oracle releases security updates quarterly,[15] while Apple[16] and Red Hat[17] have a more flexible schedule. By contrast, in the AGMM group, Google,[18] Microsoft[19] and Mozilla[20] release security updates either monthly or in (under) 30 days. As a result, the variance of the patch coherence in the AGMM group is relatively small. Another possible reason could be the development process and management in a CNA, e.g., low-priority for products in an old version.

Summary of RQ3

Although the majority (i.e., 9) of the studied CNAs have a minority (<50%) of multi-affected vulnerabilities, these vulnerabilities are severe according to the CVSS scores, especially when they affect both types of products. The delay (Diff$_{pd}$) of patch releases for multi-affected vulnerabilities is longer than that of single-affected vulnerabilities. When the CNAs are affected by the common vulnerabilities, the patch coherence of vulnerabilities with weaknesses related to buffer overflow (CWE-119) and access controls (CWE-264) is a median of 0 days. Furthermore, the patch coherence in the AGMM group (i.e., Adobe, Google, Microsoft, and Mozilla) has a median of 0 days.

[15] `https://www.oracle.com/corporate/security-practices/assurance/vulnerability/`

[16] `https://support.apple.com/en-us/HT201220`

[17] `https://access.redhat.com/solutions/3711551`

[18] `https://chromium.googlesource.com/chromium/src/+/master/docs/security/severity-guidelines.md`

[19] `https://www.microsoft.com/en-us/msrc/faqs-security-update-guide`

[20] `https://support.mozilla.org/en-US/kb/managing-firefox-updates`

6 Discussion

In this section, we discuss the implications of our findings for practitioners (Section 6.1), researchers (Section 6.2) and CNAs (Section 6.3).

6.1 Practitioners

Practitioners should be aware of the potential issue of patch incoherence, i.e., a large value of $\mathbf{Diff}_{pp}$, after the fix of a given vulnerability is available. Although the studied CNAs are leading security companies, the delay Diff_{pp} of patch releases between them for a given vulnerability could vary from 0 days up to more than one year. The patch incoherence indicates the period that some of the vulnerable products affected by a vulnerability with an available fix remain unfixed. The delay Diff_{pp} is impacted by product types, vulnerability weaknesses, and the affected vendors. For example, CVE-2011-3071[21] allowed remote attackers to cause a denial of service in Chrome (Google) and Safari (Apple). Google made an announcement of security updates for CVE-2011-3071 along with 11 other vulnerabilities in a point release,[22] while Apple made an announcement for CVE-2011-3071 along with 120 other vulnerabilities in a major release.[23]

We compare patch coherence with two prior studies. Nappa et al. [33] observed that the patch coherence of vulnerabilities in a shared code snippet between two applications is a median of 11 days from 10 popular client-side applications. The patch coherence in our work is longer (i.e., a median of 35 days) than the prior work since we study vulnerabilities that affected two CNAs' products with different code bases, which needs developers of each of the CNAs to develop their own fix. For dependent packages in the npm ecosystem, Decan et al. [15] showed that 50% of the dependent packages affected by vulnerable upstream projects took nearly 14 months to release fixes, while their upstream usually had a fix available within one month. The patch coherence of the dependent packages is longer than of the projects in our work. One possible reason of the poor patch coherence for dependent packages in npm could be due to the development process and management of these dependent projects, e.g., additional migration effort (e.g., code changes) to fix the vulnerable upstream projects [26].

Practitioners should develop a better coordination model on vulnerability fix releases to reduce the size of the window of vulnerability exploits (i.e., obtain a negative value of $\mathbf{Diff}_{pd}$) for delegated vulnerabilities. Each individual CNA releases fixes for vulnerabilities with a delay Diff_{pd} w.r.t. the disclosure date (Tables 2 to 4). The Diff_{pd} delay for delegated vulnerabilities is longer than for self-assigned vulnerabilities,

[21] `https://nvd.nist.gov/vuln/detail/CVE-2011-3071`

[22] `https://chromereleases.googleblog.com/2012/04/stable-and-beta-channel-updates.html`

[23] `https://support.apple.com/en-us/HT202561`

Table 6: Comparison of the Diff_{pd} delay with prior work.

	Data[1]	**Subjects**	**Types**	$\textbf{Diff}^{2}_{pd}$
Our work	2010 - 2020	13 CNAs	companies	median of -2 day
Shahzad et al. [40]	1998 - 2011	8 vendors	both[3]	$<$ 0 days[4]
Li and Paxson [28]	Till 2016	682 projects	open-source	$<$ -7 days
Piantadosi et al. [35]	337 vulns	Apache Http and Tomcat	open-source	median -12 and -54 days
Decan et al. [15]	2012 - 2017	npm	open-source	$<$ 0 days[5]
Alfadel et al. [7]	550 vulns	Python	open-source	$>$ 0 days[6]

[1] Note that we present either the data period of studied vulnerabilities or the number of studied vulnerabilities, when applicable.
[2] Note that we present an approximate value with the percentage in the footnote when the prior work did not state a precise value.
[3] 2 open-source vendors and 6 companies.
[3] 73% before disclosure.
[4] 84% before disclosure.
[5] 51% after disclosure.

even though the risk level for delegated vulnerabilities is higher than for self-assigned ones. Since delegated vulnerabilities indicate that at least one CNA develops a fix and another CNA controls the disclosure process, i.e., at least two CNAs are involved, a larger value of Diff_{pd} (>0) could be an indicator of low-quality coordination between these involved CNAs. Since this of course depends on other factors, like the difficulty of fixing a given vulnerability, future work should study the quality of coordination in more detail.

In order to put our findings in perspective, we compare the Diff_{pd} delay with prior studies and summarize the result in Table 6. We observe that the trend of the Diff_{pd} for vulnerabilities in our work is similar to vulnerabilities from 1998 to 2011 studied by Shahzad et al. [40]. While several other works [7, 15, 28, 35] studied vulnerabilities in open-source projects, our study focuses on corporations (CNAs). Yet, the gap of the Diff_{pd} delay between open-source projects and corporations is not significantly large, as these studies found that the value of the (Diff_{pd}) delay in open-source projects usually is negative (before disclosure), except in the Python ecosystem [7].

6.2 Researchers

Researchers could explore more dimensions (e.g., developer team size) on understanding the delay of patch releases w.r.t the disclosure date in order to reduce such delay, especially for delegated vulnerabilities. We investigate several dimensions (e.g., product types, assignor) and observe the similarity and difference across the CNAs. For example, the CNAs often release fixes for self-assigned vulnerabilities within 30 days before the disclosure date, while they have an inconsistent pace for delegated vulnerabilities, from -30 days up to $\geqslant$ 90 days (RQ2). Given the increasing number of vulnerabilities and software supply chain attacks, and the fact that a vulnerability disclosed without any available fix indicates the highest risk, a shorter delay of patch releases w.r.t. the disclosure date reduces the chance of vulnerability exploits at a large scale.

In addition, **more research is required for vulnerabilities in the operating-system and hardware types of product.** The delay Diff_{pd} of releasing fixes for delegated vulnerabilities in the operating-system type of product is longer than that of the application type of product, by a median of 4 days (RQ2). However, for the application type of product, the median is minus 1 day (fix available before disclosure), compared to plus 3 days (after disclosure) for the operating-system type of product. Furthermore, the lower the severity level of a vulnerability in the operating-system type of product, the longer the delay (Diff_{pd}) of releasing fixes is. While a large number of prior studies has been conducted on application vulnerabilities [20, 22, 25, 36], vulnerabilities in operating systems (e.g., Windows, Debian) are explored less commonly, even though operating-system vulnerabilities impact an order of magnitude more users than application-type ones. Furthermore, software providers often do not provide updates for all affected operating system versions, due to end-of-life (EOL) [17].

Finally, a large number (5,339) of multi-affected vulnerabilities affect both operating-system and application types of products. These vulnerabilities are extremely severe, with a median base score of 9.3, median exploitability score of 8.6 and a median impact score of 10. As such, the delay of releasing fixes in a median of 4 days after the disclosure date leads to a large impact on a victim system. More research should dive into detection tools for such a kind of vulnerability or techniques to secure operating systems.

6.3 CNAs

The existing CNAs should consider designing a different disclosure process for delegated vulnerabilities, compared to self-assigned vulnerabilities. First, affected software vendors have a potential delay of becoming aware of delegated vulnerabilities, compared to self-assigned vulnerabilities. Second, although delegated vulnerabilities account for a minority ($<$50%) of vulnerabilities in each of the studied CNAs except Google (Figure 5), they are more severe than self-assigned vulnerabilities. Delegated vulnerabilities in the application and operating-system types of products have a median CVSS score of 7.5 and 7.1, respectively, i.e., both are high-severity. Moreover, the results in RQ2 indicate that the delay Diff_{pd} of releasing fixes for delegated vulnerabilities in the operating-system type of product is a median of 3 days up to more than 90 days after the disclosure date. This gives attackers a large window to exploit delegated vulnerabilities because the fixes are not yet available, even though the public is aware of these vulnerabilities. Furthermore, given the increasing number of software supply chains and the number of software

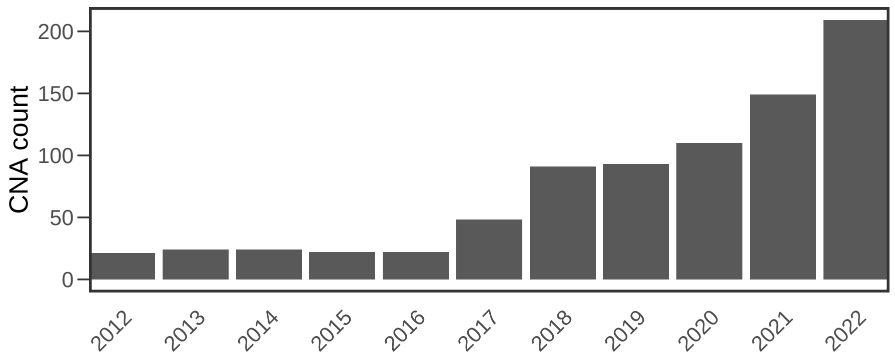

Fig. 15: The number of CNAs in the CVE program.

vendors in a software supply chain, such a large window implies a potential large scale of losses.

Due to the increasing number of vulnerabilities in recent years and a large number ($>$15k) of vulnerabilities in recent years,[24] many new CNAs have joined the CVE program to share efforts of assigning and disclosing vulnerabilities. Figure 15 indicates that the number of CNAs doubled twice, from 2017 to 2018 and from 2020 to 2022. Since the CVE program does not specify general guidelines of vulnerability disclosure for CNAs, each CNA discloses vulnerabilities based on its own background knowledge and policy. As such, given the high-risk of delegated vulnerabilities, a new CNA may disclose a vulnerability earlier/later than it should be. In addition, even the delay of patch releases for delegated vulnerabilities is a median of 1 day before disclosure, there exists a high chance of vulnerability exploitation due to the inconsistent pace of patch releases across the affected vendors, i.e., patch incoherence.

The concept of coordinated vulnerability disclosure (CVD) should be widely broadcast to ensure the security of affected products. The software security-related communities, including researchers, companies and governments, have been devoted to deriving best practices for multi-party coordination of fixing vulnerabilities, given today's era of widespread open-source reuse. However, it takes time to develop practices across software projects, companies, organizations and countries, and to start applying them. For example, the European Union Agency for Cybersecurity (ENISA) reported that the notion of CVD remains fragmented in the 2023 report[25], even though the European Cyber Resilience Act was published in September, 2022.[26] Our results identify traces of such fragmentation as there exist Diff_{pd} inconsistency and a large delay of patch releases across CNAs.

[24] `https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cvss-severity-distribution-over-time`

[25] `https://www.enisa.europa.eu/news/coordinated-vulnerability-disclosure-towards-a-common-eu-approach`

[26] `https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act`

Our results could be taken into account when developing CVD best practices. For example, Table 4 indicates that most of the studied CNAs release the fixes for the largest (bold) number of multi-affected vulnerabilities (both self-assigned and delegated) 30 days before the disclosure (-30∼0 days). Therefore, for a given vulnerability, an affected vendor should be able to release the fix within 30 days after the disclosure of the vulnerability (0∼30 days). There might be some exceptional policies, such as, for a given vulnerability, a vendor having more than 5 variants of the affected products should release the fixes for all affected variants within 90 days after the disclosure (see Figure 12).

7 Threats to Validity

7.1 Internal Validity

We calculate the delays Diff_{pd} and Diff_{pp} based on the collected security advisories from `tenable.com` because the open-source vulnerability database (OSVDB) was shutdown permanently in 2016.[27] Although `tenable.com` might miss some security advisories, our data set includes 82% of the 39,409 collected vulnerabilities (cf. Section 4.2) that affected application and operating-system types of products (i.e., software-related products) to provide insightful information of current practices that are used by top companies. We encourage future studies to explore other data sources to generalize our results.

We only study the application and operating-system types of products according to the CPE configurations in RQ2 and RQ3, due to the lack of security advisories related to hardware type of vulnerabilities in `tenable.com`. However, vulnerabilities in the hardware type of product are also severe in the context of software supply chains. For example, attackers gained complete control via the Starbleed (CVE-2019-14626)[28] vulnerability in FPGA (Field Programmable Gate Arrays) chips, which can be found in many safety-critical applications. In addition, as the CPE configurations only include the product type (e.g., "application" type) of an affected product, we cannot classify the affected product into finer-grained categories using the provided data. One way we tried to obtain finer-grained category data is by mining an existing database of open-source projects, i.e., OpenHub. While OpenHub did contain a statistically representative sample of 130 (4%) out of the studied 3,349 application-type products by exact name matching, this sample was biased in several ways. For example, 8 out of the 130 products have recorded data about their organizations and these 8 products were published by 3 vendors. The 8 products account for 4.2% of the studied vulnerabilities that affected the application-type products in our dataset. Moreover, OpenHub only lists open-source products, while our study includes application-type products from both open-source and commercial vendors. Due to the aforementioned reasons, we

[27] `https://www.securityweek.com/osvdb-shut-down-permanently`
[28] `https://nvd.nist.gov/vuln/detail/CVE-2019-14626`

encourage future work to explore more data sources to have a comprehensive analysis of application-type products.

Another internal threat to validity is that our study focuses on vulnerabilities with a valid CVE ID and excluded those (9,836) with an invalid CVE ID.[29] These CVE IDs with an invalid status may still correspond to vulnerabilities to a certain extent, but were invalid due to a variety of reasons. For example, one CVE should be split into several CVEs to reflect the risks of these CVEs, or the definition of a CVE is controversial from different CNAs, ending up with a dispute status.

7.2 External Validity

These practices shed light into the deeper understanding of how fast the CNAs react to vulnerabilities, but we cannot claim that our results generalize to all other CNAs that are not studied in this work. Therefore, we acknowledge that future studies are required to add more data sources and reach more general conclusions.

8 Conclusion

In this work, we investigate the efficiency of vulnerability coordination and fix releases w.r.t the disclosure date from a set of 39,409 vulnerabilities and the top 13 CNAs that have played several roles in multi-party coordination on fixing vulnerabilities. Using the dataset, we analyze influential factors of vulnerability fix releases and disclosure, including the assignor of a vulnerability, affected product types and the number of affected vendors. In addition, we extensively measure the efficiency of patch releases and patch coherence in the multi-party coordination.

Our findings have revealed that the size of the window of vulnerability exploitation in the CNAs is usually small (with fixes released before the disclosure), but large for delegated vulnerabilities in operating-system products and multi-affected vulnerabilities affecting both operating-system and application products (with fixes released after disclosure). Additionally, the patch coherence for vulnerabilities varies from a median of 35 days up to more than one year, due to weaknesses and pairs of the affected CNAs.

These insights indicate that the window size of potential vulnerability exploitation for a given vulnerability in a vulnerable product is not significantly large. However, given the poor patch coherence and the context of software supply chains, both software vendors and users are still at high-risk for a long period. CNAs could design better disclosure policies for various kinds of vulnerabilities and a better fix release process. Researchers could investigate additional practices to improve such policies and processes, such as ways

[29] `https://cve.mitre.org/cve/list_rules_and_guidance/correcting_counting_issues.html`

to patch vulnerable products that are affected by vulnerabilities that already have an available fix.

Data Availability Statement

The data that support the findings of this study are available on the CVE [2] and tenable.com[30] websites.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. CNA (online). URL `https://www.cve.org/ProgramOrganization/CNAs`. Last accessed: 2022-05-02
2. CVE (online). URL `https://cve.mitre.org/`. Last accessed: 2022-05-02
3. CWE (online). URL `https://cwe.mitre.org/`. Last accessed: 2022-05-02
4. Guidelines and practices for multi-party vulnerability coordination and disclosure (online). URL `https://www.first.org/global/sigs/vulnerability-coordination/multiparty/guidelines-v1.0`. Last accessed: 2022-05-02
5. National vulnerability database (online). URL `https://nvd.nist.gov/`. Last accessed: 2022-05-02
6. U.S. national institute of standards and technology. CVSS information (online). URL `https://nvd.nist.gov/vuln-metrics/cvss`. Last accessed: 2022-05-02
7. Alfadel, M., Costa, D.E., Shihab, E.: Empirical analysis of security vulnerabilities in python packages. In: 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER'21), pp. 446–457. IEEE (2021)
8. Allodi, L.: Economic factors of vulnerability trade and exploitation. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp. 1483–1499 (2017)
9. Arora, A., Krishnan, R., Nandkumar, A., Telang, R., Yang, Y.: Impact of vulnerability disclosure and patch availability-an empirical analysis. In: Third Workshop on the Economics of Information Security, vol. 24, pp. 1268–1287 (2004)
10. Arora, A., Krishnan, R., Telang, R., Yang, Y.: An empirical analysis of software vendors' patch release behavior: impact of vulnerability disclosure. Information Systems Research **21**(1), 115–132 (2010)

[30] `https://www.tenable.com/`

11. BHARGAVA, G., CHANDINI, M.: Then and now: On the maturity of the cybercrime markets (2018)
12. Blocken, B.: 50 years of computational wind engineering: past, present and future. Journal of Wind Engineering and Industrial Aerodynamics **129**, 69–102 (2014)
13. Chen, X., Zhao, Y., Cui, Z., Meng, G., Liu, Y., Wang, Z.: Large-scale empirical studies on effort-aware security vulnerability prediction methods. IEEE Transactions on Reliability **69**(1), 70–87 (2019)
14. Chinthanet, B., Kula, R.G., McIntosh, S., Ishio, T., Ihara, A., Matsumoto, K.: Lags in the release, adoption, and propagation of npm vulnerability fixes. Empirical Software Engineering (EMSE'21) **26**(3), 1–28 (2021)
15. Decan, A., Mens, T., Constantinou, E.: On the impact of security vulnerabilities in the npm package dependency network. In: Proceedings of the 15th international conference on mining software repositories (MSR'18), pp. 181–191 (2018)
16. Dolan-Gavitt, B., Hulin, P., Kirda, E., Leek, T., Mambretti, A., Robertson, W., Ulrich, F., Whelan, R.: Lava: Large-scale automated vulnerability addition. In: 2016 IEEE Symposium on Security and Privacy (SP'16), pp. 110–121. IEEE (2016)
17. Farhang, S., Kirdan, M.B., Laszka, A., Grossklags, J.: Hey google, what exactly do your security patches tell us? a large-scale empirical study on android patched vulnerabilities. arXiv preprint arXiv:1905.09352 (2019)
18. Feutrill, A., Roughan, M., Ross, J., Yarom, Y.: A queueing solution to reduce delay in processing of disclosed vulnerabilities. In: 2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, pp. 1–11. IEEE (2020)
19. Frei, S., May, M., Fiedler, U., Plattner, B.: Large-scale vulnerability analysis. In: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense, pp. 131–138 (2006)
20. Goyal, P., Parmar, V., Rishi, R., et al.: Manet: vulnerabilities, challenges, attacks, application. IJCEM International Journal of Computational Engineering & Management **11**(2011), 32–37 (2011)
21. Grieco, G., Grinblat, G.L., Uzal, L., Rawat, S., Feist, J., Mounier, L.: Toward large-scale vulnerability discovery using machine learning. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 85–96 (2016)
22. Gupta, S., Gupta, B.B.: Detection, avoidance, and attack pattern mechanisms in modern web application vulnerabilities: present and future challenges. International Journal of Cloud Applications and Computing (IJCAC17) **7**(3), 1–43 (2017)
23. Huang, Z., DAngelo, M., Miyani, D., Lie, D.: Talos: Neutralizing vulnerabilities with security workarounds for rapid response. In: 2016 IEEE Symposium on Security and Privacy (SP'16), pp. 618–635. IEEE (2016)
24. Joh, H., Malaiya, Y.K.: Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics. In: Proceedings of the 2011 International Conference on Security and Management

(SAM'11), vol. 1, pp. 10–16 (2011)
25. Jovanovic, N., Kruegel, C., Kirda, E.: Pixy: A static analysis tool for detecting web application vulnerabilities. In: 2006 IEEE Symposium on Security and Privacy (SP'06), pp. 6–pp. IEEE (2006)
26. Kula, R.G., German, D.M., Ouni, A., Ishio, T., Inoue, K.: Do developers update their library dependencies? Empirical Software Engineering (EMSE'18) **23**(1), 384–417 (2018)
27. Lee, J., Hong, S., Oh, H.: Memfix: static analysis-based repair of memory deallocation errors for c. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 95–106 (2018)
28. Li, F., Paxson, V.: A large-scale empirical study of security patches. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 2201–2215 (2017)
29. Li, Z., Zou, D., Xu, S., Jin, H., Zhu, Y., Chen, Z.: Sysevr: A framework for using deep learning to detect software vulnerabilities. IEEE Transactions on Dependable and Secure Computing (2021)
30. Liu, B., Meng, G., Zou, W., Gong, Q., Li, F., Lin, M., Sun, D., Huo, W., Zhang, C.: A large-scale empirical study on vulnerability distribution within projects and the lessons learned. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE'20), pp. 1547–1559. IEEE (2020)
31. Machiry, A., Redini, N., Camellini, E., Kruegel, C., Vigna, G.: Spider: Enabling fast patch propagation in related software repositories. In: 2020 IEEE Symposium on Security and Privacy (SP'20), pp. 1562–1579. IEEE (2020)
32. Nakajima, A., Watanabe, T., Shioji, E., Akiyama, M., Woo, M.: A pilot study on consumer iot device vulnerability disclosure and patch release in japan and the united states. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), pp. 485–492 (2019)
33. Nappa, A., Johnson, R., Bilge, L., Caballero, J., Dumitras, T.: The attack of the clones: A study of the impact of shared code on vulnerability patching. In: 2015 IEEE symposium on security and privacy (SP'15), pp. 692–708. IEEE (2015)
34. Ozment, A., Schechter, S.E.: Milk or wine: does software security improve with age? In: USENIX Security Symposium, vol. 6, pp. 10–5555 (2006)
35. Piantadosi, V., Scalabrino, S., Oliveto, R.: Fixing of security vulnerabilities in open source projects: A case study of apache http server and apache tomcat. In: 2019 12th IEEE Conference on software testing, validation and verification (ICST'19), pp. 68–78. IEEE (2019)
36. Rafique, S., Humayun, M., Hamid, B., Abbas, A., Akhtar, M., Iqbal, K.: Web application security vulnerabilities detection approaches: A systematic mapping study. In: 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'15), pp. 1–6. IEEE (2015)

37. Ruohonen, J.: An empirical analysis of vulnerabilities in python packages for web applications. In: 2018 9th International Workshop on Empirical Software Engineering in Practice (IWESEP'18), pp. 25–30. IEEE (2018)
38. Ruohonen, J., Rauti, S., Hyrynsalmi, S., Leppänen, V.: A case study on software vulnerability coordination. Information and Software Technology **103**, 239–257 (2018)
39. Sabottke, C., Suciu, O., Dumitraș, T.: Vulnerability disclosure in the age of social media: Exploiting twitter for predicting {Real-World} exploits. In: 24th USENIX Security Symposium (USENIX Security 15), pp. 1041–1056 (2015)
40. Shahzad, M., Shafiq, M.Z., Liu, A.X.: A large scale exploratory analysis of software vulnerability life cycles. In: 2012 34th International Conference on Software Engineering (ICSE'12), pp. 771–781. IEEE (2012)
41. Shin, Y., Meneely, A., Williams, L., Osborne, J.A.: Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE transactions on software engineering (TSE'10) **37**(6), 772–787 (2010)
42. Sood, A.K., Bansal, R., Enbody, R.J.: Cybercrime: Dissecting the state of underground enterprise. IEEE internet computing **17**(1), 60–68 (2012)
43. Stock, B., Pellegrino, G., Rossow, C., Johns, M., Backes, M.: Hey, you have a problem: On the feasibility of {Large-Scale} web vulnerability notification. In: 25th USENIX Security Symposium, pp. 1015–1032 (2016)
44. Wang, X., Sun, K., Batcheller, A., Jajodia, S.: Detecting "0-day" vulnerability: An empirical study of secret security patch in OSS. In: 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'19), pp. 485–492. IEEE (2019)
45. Wu, D., Gao, D., Cheng, E.K., Cao, Y., Jiang, J., Deng, R.H.: Towards understanding android system vulnerabilities: techniques and insights. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), pp. 295–306 (2019)
46. Zhang, H., Wang, S., Li, H., Chen, T.H.P., Hassan, A.E.: A study of C/C++ code weaknesses on stack overflow. IEEE Transactions on Software Engineering (TSE'21) (2021)
47. Zhao, M., Grossklags, J., Liu, P.: An empirical study of web vulnerability discovery ecosystems. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1105–1117 (2015)