The Journal of Systems & Software 201 (2023) 111679

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

The anatomy of a vulnerability database: A systematic mapping study☆

Xiaozhou Li [a,c,*], Sergio Moreschini [a], Zheying Zhang [a], Fabio Palomba [b], Davide Taibi [a,c]

[a] Tampere University, Tampere, Finland
[b] SeSa Lab - University of Salerno, Fisciano, Italy
[c] University of Oulu, Oulu, Finland

ARTICLE INFO

Article history:
Received 18 August 2022
Received in revised form 2 February 2023
Accepted 13 March 2023
Available online 22 March 2023

Keywords:
Software security
Vulnerability databases
Systematic mapping studies
Software evolution

ABSTRACT

Software vulnerabilities play a major role, as there are multiple risks associated, including loss and manipulation of private data. The software engineering research community has been contributing to the body of knowledge by proposing several empirical studies on vulnerabilities and automated techniques to detect and remove them from source code. The reliability and generalizability of the findings heavily depend on the quality of the information mineable from publicly available datasets of vulnerabilities as well as on the availability and suitability of those databases. In this paper, we seek to understand the anatomy of the currently available vulnerability databases through a systematic mapping study where we analyze (1) what are the popular vulnerability databases adopted; (2) what are the goals for adoption; (3) what are the other sources of information adopted; (4) what are the methods and techniques; (5) which tools are proposed. An improved understanding of these aspects might not only allow researchers to take informed decisions on the databases to consider when doing research but also practitioners to establish reliable sources of information to inform their security policies and standards.

1. Introduction

Software security has been always considered a crucial non-functional requirement to meet when developing software (Dowd et al., 2006). With the rise of novel technologies and devices, e.g., Internet-of-Things (IoT) devices empowered by artificial intelligence approaches, the need for secure software is becoming even more important to avoid malicious accesses to data and information treated by software systems (Duc et al., 2017). When it comes to software engineering, security refers to the design and implementation of programs that are resilient to external attacks (McGraw, 2004), other than to the verification and validation mechanisms that might be put in place to manage it (Edmundson et al., 2013; Felderer et al., 2016; Howard, 2006).

Software vulnerabilities are among the major threats to security (Iannone et al., 2022). These are weaknesses introduced by programmers that may be exploited by externals to cause loss or harm during software maintenance and evolution (Decan et al., 2018; Plate et al., 2015).

The software engineering research community has been targeting the problem of vulnerabilities from multiple perspectives, by understanding their life cycle (Iannone et al., 2022; Shahzad et al., 2012), their impact on code quality and reliability (Finifter et al., 2013; Kim and Lee, 2018; Gonzalez et al., 2019), and defining several automated approaches and tools to support their detection (Hydara et al., 2015; McKinnel et al., 2019; Svacina et al., 2020; Lomio et al., 2022).

A significant amount of research done in the area, both in terms of empirical studies and approaches defined, relied on the elaboration of data coming from publicly available vulnerability databases. The mining of vulnerability repositories indeed represents a widely-adopted research approach that is useful to feed machine learning, deep learning, static and dynamic analysis, and other techniques used for detecting vulnerabilities (Ghaffarian and Shahriari, 2017; Murtaza et al., 2016). As such, the quality of the recommendations provided by empirical studies in literature and the detection results provided by automated approaches heavily depend on the quality of the information available in those repositories.

Our work stems from this consideration and aims at providing a comprehensive view of the sources of information typically used to study vulnerabilities and build automated approaches for software vulnerability detection. We address our goal with a *systematic mapping analysis* of the literature Petersen et al. (2008). Through this process, we identify and classify the existing literature on vulnerability databases in an effort of providing insights into their *anatomy*. We specifically investigate five aspects: (1)

☆ Editor: Heiko Koziolek.

* Corresponding author at: University of Oulu, Oulu, Finland.
E-mail addresses: xiaozhou.li@oulu.fi (X. Li), sergio.moreschini@tuni.fi (S. Moreschini), zheying.zhang@tuni.fi (Z. Zhang), fpalomba@unisa.it (F. Palomba), davide.taibi@oulu.fi (D. Taibi).

https://doi.org/10.1016/j.jss.2023.111679

What are the most common security-specific public databases of security vulnerabilities employed by the research community; (2) What are the goals to employ vulnerability databases by the research community; (3) What are the other sources of information adopted to facilitate such goals; (4) what are the methods and techniques adopted; and (5) Which tools are proposed by adopting or for investigating vulnerability databases. Important as software security is, understanding the research domain of vulnerability databases via investigating these research questions shall certainly contribute to this critical area. The results of our work can indeed inform researchers about the existing vulnerability databases and their characteristics so that they can take informed decisions on the databases to consider when designing future approaches for vulnerability discovery. At the same time, an improved understanding of how vulnerability reports are created, stored, and managed may be useful for practitioners interested in enhancing their security policies and standards.

Structure of the paper. Section 2 introduces the background information about vulnerability databases. Section 3 reports on the research method employed to conduct the systematic mapping study. In Section 4 we analyze the results addressing the five goals of the study. Section 5 presents the main discussion points and the implications coming from our analysis. The possible threats to validity of the study are discussed in Section 6. Section 7 discusses the related work. Finally, Section 8 concludes the paper and outlines our future research agenda.

2. Background information

A vulnerability database collects, maintains, and disseminates information about discovered security vulnerabilities. The National Vulnerability Database (NVD) (National Vulnerability Database (NVD), 2022) is one of the influential vulnerability databases. It was created based on the list of Common Vulnerability and Exposures (CVE) (CVE, 2022) entries. Using CVEs and CVE identifiers ensures that unique vulnerabilities are discussed, and that information about the same vulnerability is shared by different parties. Many studies employed the NVD reports and the CVE entries to construct datasets for data-driven vulnerability detection and prediction. For example, Gkortzis et al. (2018) searched the NVD reports to create a dataset VulinOSS that reports vulnerabilities of 8694 open-source components to analyze the diverse software characteristics and their relation to security vulnerabilities. Nikitopoulos et al. (2021) analyzed the GitHub commits referenced by NVD and CVE entries to curate a labeled dataset, CrossVul, containing 27476 vulnerable source code files and the respective patched security counterparts retrieved from Git commits. The dataset can be used to train models to detect security patch commits. Similarly, to investigate an automated approach to identifying fixes for new vulnerability disclosures in SAP development units, Ponta et al. (2019) manually collected data from both the NVD and the project-specific web resources to curate a dataset of vulnerabilities and mapped them to the fixes.

Additionally, there are vulnerabilities reported by other security advisory sources such as Security Focus, IBM's X-Force, etc. The key aspects of vulnerabilities in these different security databases are described differently and are complementary (Massacci and Nguyen , 2010; Guo et al., 2021). To meet the different needs in software security management, there have been studies to create a hybrid vulnerability database (Yun-hua and Pei , 2010) by analyzing the CVE, NVD, X-Force databases or propose an ontology (Fedorchenko et al., 2017) to construct a hybrid security repository that incorporates the information security concepts from databases and their relations.

While many attempts have been made to curate datasets for investigating the security aspect of software components, a systematic study of research publications on the use of software vulnerabilities from different data sources remains under-explored. Specifically, there is a lack of comprehensive understanding of the motivation for using vulnerability datasets, the sources of information on security vulnerabilities, the methods and tools to adopt the databases, etc. To better understand these aspects, we conduct a systematic mapping study of the research on vulnerability databases.

3. Research method

The goal of the systematic mapping study is to summarize the state of the art on the use of public vulnerability databases, with the purpose of deriving limitations and open challenges that researchers might need to address to further support practitioners and researchers in devising methodologies and tools to deal with software vulnerabilities. In the context of our research we elaborated a number of research questions that aim at targeting the problem under different perspectives. The metrics for answering each of the questions are the sorted list of categorized items summarized from the systematically selected articles.

These are listed in the following:

- **RQ**$_1$. *What are the most common security-specific public databases of security vulnerabilities employed by the research community?*
- **RQ**$_2$. *What are the goals to employ vulnerability databases by research communities?*
- **RQ**$_3$. *What are the other sources of information adopted to facilitate such goals?*
- **RQ**$_4$. *What are the methods and techniques adopted?*
- **RQ**$_5$. *Which tools are proposed for adopting or for investigating vulnerability databases?*

Our systematic mapping study adheres to the commonly adopted guidelines provided by Petersen et al. (2008). In addition, we also followed the guidelines by Wohlin (2014), which are connected to the adoption of the so-called "snowballing", i.e., the analysis of the references of primary studies aiming at extracting additional relevant sources to use when summarizing the current knowledge on a given subject. When reporting the research method adopted, we followed the recently defined ACM/SIGSOFT Empirical Standards[1] Given the nature of our study and the currently available standards, we followed the *"General Standard"* and *"Systematic Reviews"* definitions and guidelines.

3.1. Defining the search process

The main challenge of any systematic mapping study concerns the definition of a search string that can lead to the retrieval of a complete set of studies to analyze (Kitchenham and Charters, 2007). Our search strategy comprised a number of steps, namely the search terms identification, the specification of the resources to be searched, the definition of the search process, and finally the definition of article selection criteria (see Fig. 1).

Search String. We first used the research questions to identify the major terms that we aimed at considering. As such, we started with the terms *"software vulnerabilit*"* and *"software vulnerabilit* database"*. Secondly, for these terms, we found commonly used alternative spellings and/or synonyms. This step led to the inclusion of terms like *"security*

[1] Available at: https://github.com/acmsigsoft/EmpiricalStandards.

Table 1
Inclusion and exclusion criteria.

Inc./Exc.	Criteria
Inclusion	Papers on how to use vulnerability databases
	Papers on how to enhance vulnerability databases
	Papers on how to create vulnerability databases
	Papers proposing methods to analyze the vulnerability datasets
	Papers using vulnerability databases
Exclusion	Not in English
	Duplicated (post summarizing other websites)
	Out of topic (using the terms for other purposes)
	Non peer-reviewed papers
	Research Plans, roadmaps, vision papers
	Not employing any vulnerability databases

vulnerabilit"*, *"security weakness*"* for the original term *"software vulnerabilit*"*, but also *"dataset*"* and *"repositor*"* as synonyms of *"database"*. To check for consistency and completeness, we verified the presence of the keywords in any relevant paper that was initially scanned: the third step consisted of verifying the presence of any additional term that we did not originally include. In our case, this step did not return any terms. For this reason, we then proceeded with the usage of boolean operators to relate the various terms identified so far: we used the *OR* operator to concatenate alternative spellings and synonyms, while the *AND* operator was used to concatenate the major terms. The final outcome was represented by the following search string:

"(security OR vulnerabilit OR weakness*)"*
AND
"(database OR dataset* OR repositor*)"*

Resources to be searched. After establishing a search string, we defined the set of literature databases to search for. We first considered Scopus,[2] which is the most extensive literature database up to date. For double-checking the results achieved from Scopus, we also considered IEEEXplore[3] the ACM Digital Library,[4] the ScienceDirect,[5] and the citation database Web of Science that index articles published by a number of publishers. The selection of these databases was mainly driven by the popularity and potential level of completeness that they ensure. As a matter of fact, the selected databases are widely recognized as the most representative of the research in software engineering (Kitchenham and Charters, 2007), other than being used by many systematic literature and mapping studies (Azeem et al., 2019; Hall et al., 2011; Petersen et al., 2008; Sharafi et al., 2015). It is worth pointing out that we consciously excluded Google Scholar[6] from the set of databases: it does not include sources that have been already published, but also unpublished research (e.g., preprints currently available on open-access archives like for ArXiv and others). To avoid the analysis of articles that are still not peer-reviewed, we then decided not to rely on Google Scholar.

Inclusion and Exclusion Criteria. As for the inclusion criteria, these were mainly connected to the usefulness of an article to address our research objectives. As described in Table 1, we included papers based on five criteria that map our research questions.

To be useful for addressing our research questions, the articles retrieved were scanned for consistency and adequacy. The full list of inclusion and exclusion criteria is available in Table 1. As shown, we first filtered out papers that were not written in English, that were duplicated, and not discussing topics connected to our research questions. In addition, we also excluded papers that are not peer-reviewed[7] and short papers that only present preliminary ideas. It is also worth remarking that, in cases where we recognized that an article represented an extension of a previously published paper, e.g., journal papers extending conference publications, we only kept the extension, hence filtering out the previous, preliminary version. In addition, we also screen out studies that do not employ any vulnerability databases in their main contribution; therefore, studies regarding vulnerability detection using static analysis are not included herein.

Table 2
Initial literature search by library.

Library	Count
Scopus	1023
IEEE	277
ACM	256
Science direct	113
Web of science	132
Non-duplicates	1205

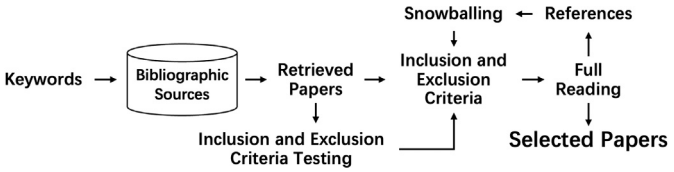

Fig. 1. The search and selection process.

3.2. Applying the search process

After defining the key elements of our mapping study, we proceeded with the application of the search string on the search databases. We did not put any time restriction on the search process in an effort of collecting as many articles as possible and, therefore, be as complete as possible in our reporting. It is inevitable a certain number of potential primary studies are not included at first; however, the snowballing step shall compensate for the selection.

The search results are reported in Table 2, which shows how many papers have been identified when querying each of the considered databases after the exclusion step. The initial candidate set was composed of 1,736 papers, which was reduced to 1,140 after removing the duplicates.

Afterward, the first two authors of this paper assumed the role of *inspectors*. They first conducted a pilot investigation, where they verified the validity of the exclusion and inclusion criteria: in this respect, they first independently analyzed an initial set composed of 50 articles, randomly selected from the candidate

[2] The Scopus database: https://www.scopus.com.
[3] The IEEEXplore database: https://ieeexplore.ieee.org/.
[4] The ACM Digital Library: https://dl.acm.org.
[5] The ScienceDirect: https://www.sciencedirect.com/.
[6] Google Scholar: https://scholar.google.com.

[7] Even if we did not consider Google Scholar as database, we could still obtain non-peer-reviewed articles.

Table 3
Initial literature search by library.

Step	# Papers
Retrieve from sources (unique)	1205
Read by Title & Abstract	1096 rejected
Full read	67 rejected
Snowballing	27
Selected Papers	69

set. After the pilot, the inspectors met and discussed the results: this procedure did not eventually lead to modifications in the exclusion and inclusion criteria, possibly indicating their completeness and suitability for our study.

Once the inspectors had completed the pilot, they proceeded with the application of the exclusion criteria to the set of retrieved articles. This was still done by the inspectors independently. The analysis was first done based on the title and abstract. In cases where the inspectors were doubtful, they proceeded with the full read of papers. After the independent analysis, the two inspectors compared their results in order to reach a full consensus on the articles that should have been removed from the analysis. In case of disagreement, the inspectors first proceeded and read the entire article and then opened a discussion. If this was not enough, the other authors of the paper were involved in the decision making process.

This scanning process led to the exclusion of 1096 articles. The remaining 109 were further considered for inclusion. The inspectors proceeded with the full-text reading, still independently analyzing whether an article should be included or not. In so doing, they applied the inclusion criteria. After the independent analysis and joint discussion, 42 papers were accepted for the mapping study.

Each of these articles was then subject to another round of review, which was performed with the aim of applying a backward and forward snowballing process. With the forward snowballing, the inspectors looked at the articles that cited each of the papers. To accomplish this task, the inspectors relied on Google Scholar, which allows them to easily search for this information. As for the backward snowballing, the inspectors looked at the references of each paper in order to verify whether some relevant piece of research was missing. The backward snowballing process was repeated until no new papers were identified, i.e., the inspectors did not limit the search to the references of the articles accepted, but also went through the references of the cited articles, performing additional snowballing steps. The number of iterations was 2. Due to the initial exclusion step, the snowballing results in a considerable number of additional papers. Overall, the snowballing led to the identification of 27 extra-papers. Hence, the total number of papers led to 69 (see Table 3).

3.3. Data extraction

From the 69 primary studies (PS_s) selected previously based on the inclusion and exclusion criteria, we extract the according data therein and map the different data to the answering of each RQ. The extraction process was driven by the *open coding* research method, namely through an analytic process by which we could assign concepts (codes) to the observed data. In our specific case, we assigned a category to each paper based on the objective of our research questions. For instance, we assigned a code reporting the main goal of a paper with respect to the use of vulnerability databases in RQ2, while we tagged a paper based on the methods/techniques employed in the context of RQ4. The process was iterative and lead by the first two authors of the paper, who acted as the inspectors.

Specifically, the following steps are performed:

- In the initial stage, the inspectors independently analyzed a subset of 10 articles and assigned codes with the aim of addressing RQ_2, RQ_3, and RQ_4. The inspectors were free to assign as many codes as they found relevant. Afterward, they scheduled an in-person meeting to discuss the codes assigned so far, in an effort of finding an agreement. The meeting lasted 1.5 h. In the first place, the inspectors analyzed each of the ten papers and explained how they came up with the codes — this process was performed to increase the awareness of an inspector with respect to the choices made by the other. Secondly, they discussed their choices and found an agreement on the codes. Finally, they computed the inter-rater agreement through the Cohen's k coefficient, which measured 0.38, showing a low level of agreement.
- On the basis of the discussion had during the first meeting, the inspectors reworked the codes assigned. Then, they took the other 20 papers into account and proceeded with a new classification exercise. In this stage, the inspectors mainly attempted to use the codes that previously emerged, yet they were allowed to define new codes whenever needed. At the completion of the task, the two inspectors scheduled another in-person meeting to open a new discussion on their work. The second meeting lasted 1 h. The Cohen's k coefficient scored 0.49 (moderate), hence showing a substantial improvement.
- The inspectors reworked the codes of the previously coded papers. Afterward, they started the analysis of the remaining papers. Also, in this case, the inspectors were allowed to define new codes, if needed. Once the task was completed, the inspectors planned a final in-person meeting to assess their work — this lasted around 2 h. Two key insights emerged from such a meeting. First, no new codes were identified during the last coding exercise. As such, we reached the so-called *theoretical saturation*, namely the phase where the analysis does not propose newer insights and all concepts are developed. Second, the agreement between the inspectors scored 0.64, which may be interpreted as good. This further corroborates the completion of the data analysis. As a result, therefore, all papers were classified according to the concepts of interest.
- As a final step, we proceeded with additional validation of the codes assigned by the inspectors. In particular, the last three authors of the paper went through papers and codes in an effort of identifying possible inconsistencies and/or erroneous interpretations made during the first steps. This validation did not lead to further modifications. Hence, we could consider the data analysis completed.

3.4. Replicability

In order to allow replication and extension of our work by other researchers, we prepared a replication package[8] for this study with the complete results obtained.

4. Analysis of the results

Via the previously described process, we selected 69 papers. Amongst these selected papers (SPs), 56 of them are conference papers with 12 journal articles and one book chapter (shown in Fig. 2). Meanwhile, when dividing these selected papers by the publication year, we can observe the trend of the research on vulnerability databases. Shown in Fig. 3, the number of publications per year is stable from 2009 to 2019 when it increases in 2020 and 2021, which is likely due to the increased application of data-driven approaches. For the results in 2022, we only collected the ones published before 10.2022.

[8] https://figshare.com/s/266e02712de1f16a854a

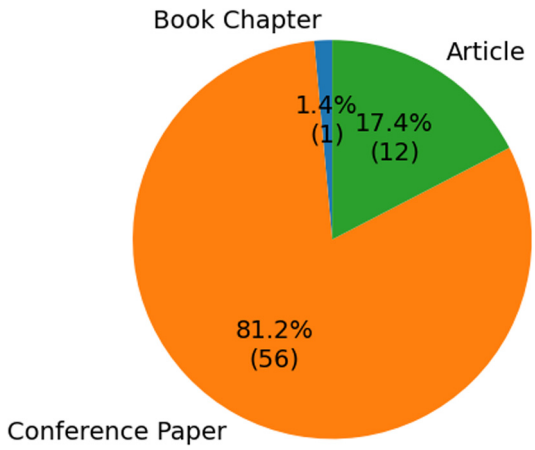

Fig. 2. Paper types.

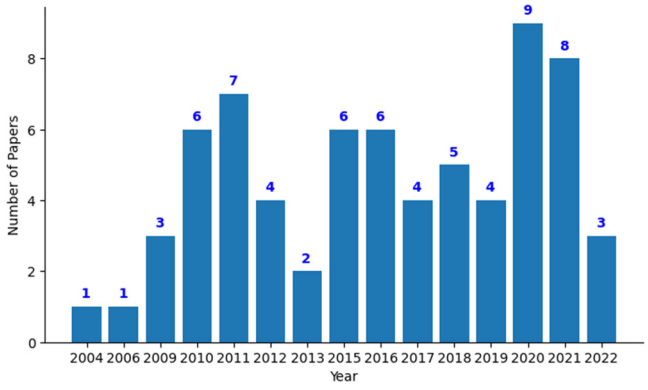

Fig. 3. Paper by year.

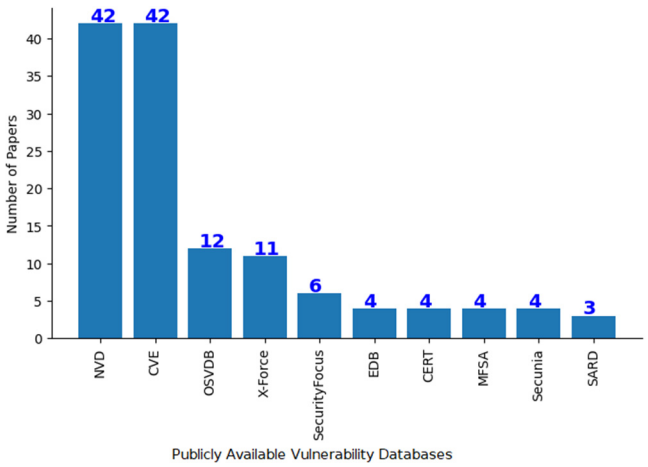

Fig. 4. Database distribution.

4.1. RQ1. What are the most common security-specific public databases of security vulnerabilities employed by the research community?

To answer RQ1, we identify the vulnerability databases employed in the selected papers and investigate which are the commonly adopted ones. In this work, we consider public platforms providing a record of existing vulnerability information as vulnerability databases, regardless of the format used to store such information. For example, CVE (a list of publicly disclosed cybersecurity vulnerabilities), NVD (A vulnerability database synchronized with CVE), and VMware Security Advisories (a list of security vulnerabilities in VMware products) are all considered vulnerability databases. Within the 69 selected papers, 25 public vulnerability databases were utilized. The information regarding these vulnerability databases is shown in Table 4.

65 of the 69 selected papers adopted either NVD or CVE out of the 26 vulnerability databases reported in Table 4 with 19 of them adopting both. The NVD includes various relevant databases that facilitate the automation of vulnerability management, security measurement, and compliance. It encompasses various information, such as security checklist references (i.e., CVE dataset), security related software weakness (i.e., CWE), impact metrics (i.e., CVSS), and so on. It is common for studies to extract information from all the datasets mentioned above when adopting NVD. For example, Gallon (2011) studies the distribution of CVSS metrics in NVD. Shahzad et al. (2012) studies the life cycle of a large number of vulnerabilities from NVD when, especially, the authors investigate the evolution of CVSS-vector metrics and the general trend of CVSS score for short-listed vendors. Therefore, similarly, for this case, we consider it employs both NVD and CVSS.

As shown in Table 4, we can draw the conclusion that NVD and CVE are the most commonly adopted vulnerability databases. Besides, both OSVDB and X-Force have also been adopted in 12 studies, when the majority of those studies also adopt either NVD or CVE. Vache (2009) is the only study among them that adopts only OSVDB without using NVD or CVE. The authors investigate the vulnerability life cycle events by comparing the vulnerability disclosure date and the exploit creation date. For such a purpose, they emphasize OSVDB can provide the patch date and exploit data information.

Besides the existing public vulnerability databases displayed in Fig. 4, some studies propose custom databases. For example, Reis and Abreu (2017) creates a custom vulnerability database, Secbench, by mining GitHub projects for different vulnerability patterns. Yap and Zhong (2004) proposes the design of a new proof-of-concept vulnerability database allowing effective data integration from multiple sources. Allodi and Massacci (2012) constructed another custom database, EKITS, from the vulnerabilities used in exploit kits from the black market. Fan et al. (2020) created a custom database, Big-Vul, containing only C/C++ code vulnerabilities from open-source GitHub projects. These custom databases are not used in the other selected papers.

To summarize and compare the popular vulnerability databases, Tan et al. listed a set of 28 security vulnerability databases which can be divided by their publishers (government or enterprise) (Tan et al., 2019). By comparing the results, we find only 10 of the vulnerability databases identified herein are mentioned by Tan et al. (2019). Meanwhile, many vulnerability databases mentioned in Tan et al. (2019), especially the ones described in the Chinese language, are not often adopted in academia.

4.2. RQ_2. What are the goals to employ vulnerability datasets by research communities?

Toward answering RQ2, we investigate the goals of the selected papers when employing their selected vulnerability databases. We summarize the goals of selected papers into the following categories. Among the 69 selected papers, we identify the following 8 main goals:

- *Analysis* - The contribution of the paper is to provide analytical results showing latent insights about one or multiple existing vulnerability databases.

Table 4
Vulnerability databases.

VDB	Description	Selected papers
1337Day	The Underground, is one of the world's most popular and comprehensive computer security web sites.	Fedorchenko et al. (2015)
Android Security Bulletins	The available Android Security Bulletins, which provide fixes for possible issues affecting devices running Android.	Linares-Vásquez et al. (2017)
CERT	The CERT/CC Vulnerability Notes Database is run by the CERT Division of Carnegie Mellon University.	Yap and Zhong (2004), Frei et al. (2006), Roschke et al. (2009), Massacci and Nguyen (2010)
CoopVDB	The Cooperative Vulnerability Database by The Center for Education and Research in Information Assurance and Security (CERIAS) of Purdue University	Roschke et al. (2009)
CVE	CVE is a list of records, each containing an identification number, a description, and at least one public reference, for publicly known cybersecurity vulnerabilities.	Yap and Zhong (2004), Frei et al. (2006), Wang and Guo (2009), Yun-hua and Pei (2010), Bozorgi et al. (2010), Neuhaus and Zimmermann (2010), Chen et al. (2010), Massacci and Nguyen (2010), Chang et al. (2011), Massacci et al. (2011), Edwards and Chen (2012), Kuo et al. (2013), Glanz et al. (2015), Fedorchenko et al. (2015), Camilo et al. (2015), Perl et al. (2015), ur Rahman et al. (2016), Takahashi and Inoue (2016), Xianghui et al. (2015), Jimenez et al. (2016), Reis and Abreu (2017), Fedorchenko et al. (2017), Han et al. (2017), Linares-Vásquez et al. (2017), Chen et al. (2018), Williams et al. (2018), Fan et al. (2020), Wu et al. (2020), Jiang et al. (2019), Vanamala et al. (2020), Chen et al. (2020), Antal et al. (2020), Sönmez (2021), Nikitopoulos et al. (2021), Wang et al. (2021), Jiang et al. (2021), Yuan et al. (2021), Guo et al. (2021), Yosifova et al. (2021), Challande et al. (2022), Hong et al. (2022), Forain et al. (2022)
DoE-CIRC	DOE-CIRC provides the U.S. Department of Energy with incident response, reporting, and tracking, along with other computer security support.	Roschke et al. (2009)
Dragonsoft	DragonSoft Vulnerability DataBase by DragonSoft Security Associates, Inc.	Roschke et al. (2009)
EDB	The Exploit Database (EDB) is an ultimate archive of exploits and vulnerable software.	Allodi and Massacci (2012), Fedorchenko et al. (2015), ur Rahman et al. (2016), Fedorchenko et al. (2017)
CERT-FR(Previous FrSIRT)	French Security Incident Response Team	Frei et al. (2006)
JVN	Japan's national vulnerability database. It is maintained by the Japan Computer Emergency Response Team Coordination Center and the Japanese government's Information-Technology Promotion Agency.	Takahashi and Inoue (2016)
MFSA	Mozilla Foundation Security Advisories	Massacci and Nguyen (2010), Massacci et al. (2011), Nguyen and Massacci (2012), Alves et al. (2016)
MSRC	The Microsoft Security Response Center is part of the defender community and on the front line of security response evolution. For over twenty years we have been working to improve security for customers. Our mission is to protect customers and Microsoft from current and emerging threats related to security and privacy.	Kuo et al. (2013), Jiang et al. (2021)
NVDB	OpenBSD Vulnerability Database (NVDB) is a vulnerability database of OpenBSD constructed for studying the vulnerability discovery process.	Massacci and Nguyen (2010)
NVD	The NVD is the U.S. government repository of standards based vulnerability management data, it includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics.	Frei et al. (2006), Wang and Guo (2009), Yun-hua and Pei (2010), Huang et al. (2010), Massacci and Nguyen (2010), Zheng et al. (2011), Tripathi and Singh (2011), Gallon (2011), Chang et al. (2011), Liu and Zhang (2011), Massacci et al. (2011), Allodi and Massacci (2012), Nguyen and Massacci (2012), Shahzad et al. (2012), Kim et al. (2013), Glanz et al. (2015), Fedorchenko et al. (2015), Zhang et al. (2015), Wen et al. (2015), Camilo et al. (2015), Takahashi and Inoue (2016), Alqahtani et al. (2016), Jimenez et al. (2016), Fedorchenko et al. (2017), Gkortzis et al. (2018), Chen et al. (2018), Ponta et al. (2019), Williams et al. (2018), Li et al. (2019), Jimenez et al. (2019), Jiang et al. (2019), Huang et al. (2020), Nerwich et al. (2020), Zou et al. (2019), Chen et al. (2020), Nikitopoulos et al. (2021), Wang et al. (2021), Kuehn et al. (2021), Jiang et al. (2021), Wu and Wang (2011), Aksu et al. (2018)
OSVDB	The Open Sourced Vulnerability Database (OSVDB) was an independent and open-sourced vulnerability database. The goal of the project was to provide accurate, detailed, current, and unbiased technical information on security vulnerabilities.	Frei et al. (2006), Roschke et al. (2009), Vache (2009), Bozorgi et al. (2010), Massacci and Nguyen (2010), Zheng et al. (2011), Tripathi and Singh (2011), Shahzad et al. (2012), Kim et al. (2013), Fedorchenko et al. (2015), Wen et al. (2015), Fedorchenko et al. (2017)

(*continued on next page*)

Table 4 (*continued*).

VDB	Description	Selected papers
SARD	Software Assurance Reference Dataset (SARD) is to provide users, researchers, and software security assurance tool developers with a set of known security flaws.	Xiaomeng et al. (2018), Li et al. (2019), Zou et al. (2019)
Secunia	Secunia Research criticality rating and Common Vulnerability Scoring System (CVSS) metrics are issued following distinct analysis including product context and related security best practices to allow for a greatly improved means of prioritizing by criticality.	Frei et al. (2006), Roschke et al. (2009), Chen et al. (2010), Tripathi and Singh (2011),
SecurityFocus	The SecurityFocus Vulnerability Database provides security professionals with the most up-to-date information on vulnerabilities for all platforms and services.	Frei et al. (2006), Roschke et al. (2009), Tripathi and Singh (2011), ur Rahman et al. (2016)
Snyk.io	Snyk is a developer security platform. Integrating directly into development tools, workflows, and automation pipelines, Snyk makes it easy for teams to find, prioritize, and fix security vulnerabilities in code, dependencies, containers, and infrastructure as code.	Decan et al. (2018)
VMware Security Advisories	former VMware Tanzu Reports and Pivotal VulnerabilityReport. Now is a document remediation for security vulnerabilities that are reported in VMware products.	Jiang et al. (2021)
VulDB	Number one vulnerability database hosting and explaining vulnerabilities since 1970. - VulDB.	Alves et al. (2016)
Vupen	VUPEN Security offers defensive and offensive cybersecurity intelligence and advanced in-house vulnerability research	Zheng et al. (2011), Liu and Zhang (2011)
X-Force	IBM X-Force Exchange is a threat intelligence sharing platform enabling research on security threats, aggregation of intelligence, and collaboration with peers.	Frei et al. (2006), Roschke et al. (2009), Yun-hua and Pei (2010), Chen et al. (2010), Massacci and Nguyen (2010), Zheng et al. (2011), Tripathi and Singh (2011), Liu and Zhang (2011), Fedorchenko et al. (2015, 2017), Guo et al. (2021)
XSA	Security Advisories for Xen Project	Alves et al. (2016)
Securiteam	A commercial vulnerability database for the network vulnerability scanner product	Roschke et al. (2009)
CNVD/CNNVD	the Chinese National Vulnerability Database	Forain et al. (2022)

- *Merging* - The contribution of the paper is to merge multiple existing vulnerability databases.
- *Creation* - The contribution of the paper is to provide the creation of new vulnerability databases by collecting security vulnerability information from other sources.
- *Application* - The contribution of the paper is to provide solutions to existing research gaps or industrial issues by adopting one or multiple vulnerability databases.
- *Classification* - The contribution of the paper is to provide vulnerability categorization or categorization approaches using one or multiple vulnerability databases.
- *Enhancement* - The contribution of the paper is to improve the quality of the existing vulnerability databases by adding information obtained from other sources.
- *Comparison* - The contribution of the paper is to provide a comparison between two or more existing vulnerability databases.
- *Detection* - The contribution of the paper is to provide approaches to detect vulnerabilities in software applications by adopting existing vulnerability databases.

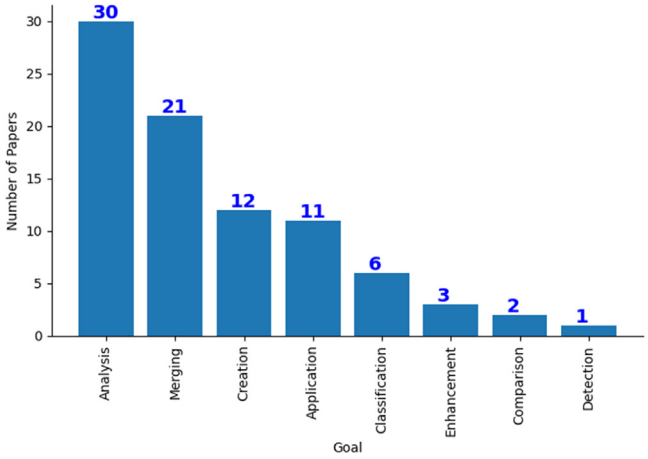

Fig. 5. Paper goals.

Fig. 5 summarizes the distribution of the different goals identified within the selected papers.

From Fig. 5 we can state that the main goal when using vulnerability databases (more than 46% of the works) is to provide analytical insights. The second notable goal (detected in ∼ 30% of the works) is to merge multiple existing databases. All the other goals can be classified as marginal as none of them exceeds 16% in our usage distribution.

Table 5 presents the categorized contributions of the selected papers mapped to the summarized goals.

As we can observe from both Fig. 5 and Table 5, 30 of the 69 selected papers contribute to analyzing the vulnerability databases as well as their related information. Therein, eight studies focus on investigating the connection between vulnerabilities and other relevant information. For example, Edwards and Chen (2012) studies the correlation between the changes in issue density and those in vulnerability discovery rate in new releases. Other studies, like Camilo et al. (2015), Alves et al. (2016), Alqahtani et al. (2016), Jimenez et al. (2016), Sönmez (2021), investigate the correlation between vulnerabilities and software repository information, e.g., pre-release bugs, issues, commit messages and metrics. Meanwhile, seven studies investigate the life cycle of vulnerabilities while five studies the trends in vulnerability as well as their metrics. Massacci et al. (2011) looks into the case of Firefox and the evolution of its source code, investigating the phenomena of "after-life vulnerabilities". Vache (2009) analyzes quantitatively the vulnerability life cycle and the patch disclosure behaviors related. Decan et al. (2018) studies the impact of vulnerabilities on the npm packages and their dependencies,

Table 5
Vulnerability databases research goals.

Goal	Contribution	Selected papers
Analysis	Trend of vulnerabilities and metrics	Frei et al. (2006), Neuhaus and Zimmermann (2010), Chang et al. (2011), Williams et al. (2018), Jiang et al. (2019)
	Impact of vulnerabilities and metrics	Frei et al. (2006), Shahzad et al. (2012), Linares-Vásquez et al. (2017), Decan et al. (2018), Jiang et al. (2021)
	Distribution of vulnerabilities	Gallon (2011), Liu and Zhang (2011)
	Quality of vulnerability DBs	Massacci and Nguyen (2010), Kuehn et al. (2021)
	Life cycle of vulnerabilities	Vache (2009), Massacci et al. (2011), Shahzad et al. (2012), Linares-Vásquez et al. (2017), Decan et al. (2018), Williams et al. (2018)
	Vulnerability discovery validation	Nguyen and Massacci (2012), Xianghui et al. (2015), Xiaomeng et al. (2018), Li et al. (2019), Jimenez et al. (2019), Huang et al. (2020), Antal et al. (2020)
	Connection between vulnerabilities and other info	Shahzad et al. (2012), Edwards and Chen (2012), Camilo et al. (2015), Alves et al. (2016), Alqahtani et al. (2016), Jimenez et al. (2016), Jiang et al. (2019), Sönmez (2021)
	Vulnerability types	Neuhaus and Zimmermann (2010), Linares-Vásquez et al. (2017)
	Others	Vache (2009), Liu and Zhang (2011), Chen et al. (2018), Nerwich et al. (2020)
Application	Prediction on vulnerability attributes	Zhang et al. (2015), Han et al. (2017), Chen et al. (2020), Yuan et al. (2021), Yosifova et al. (2021)
	Security information extraction	Perl et al. (2015), Alqahtani et al. (2016), Wu et al. (2020), Guo et al. (2021), Wu and Wang (2011)
Classification	Classification based on vulnerability information	Huang et al. (2010), Chen et al. (2010), Wen et al. (2015), Vanamala et al. (2020)
	Classification toward prediction/detection	Bozorgi et al. (2010), Perl et al. (2015)
Comparison	Comparing information contents in prominent VDBs	Tripathi and Singh (2011), Forain et al. (2022)
Creation	Creating by merging information	Yap and Zhong (2004), Zheng et al. (2011), Allodi and Massacci (2012), Fedorchenko et al. (2015), Ponta et al. (2019), Fan et al. (2020), Nikitopoulos et al. (2021), Wang et al. (2021)
	Creating by extracting information	Reis and Abreu (2017), Gkortzis et al. (2018)
Detection	Multiclass vulnerability detection	Zou et al. (2019)
Enhancement	Enhancing vulnerability info quality	Glanz et al. (2015), Takahashi and Inoue (2016), Kuehn et al. (2021)
Merging	Merging as is	Roschke et al. (2009), Wang and Guo (2009), Yun-hua and Pei (2010), Kuo et al. (2013), Kim et al. (2013), ur Rahman et al. (2016), Fedorchenko et al. (2017)
	Merging for analysis	Shahzad et al. (2012), Jiang et al. (2019), Nerwich et al. (2020)
	Merging for classification	Vanamala et al. (2020)
	Merging for creation	Yap and Zhong (2004), Zheng et al. (2011), Allodi and Massacci (2012), Fedorchenko et al. (2015), Ponta et al. (2019), Fan et al. (2020), Nikitopoulos et al. (2021), Wang et al. (2021), Challande et al. (2022), Hong et al. (2022)

regarding, the effectiveness of vulnerability discovery and fixing, as well as the related effect.

11 selected papers focus on the application of vulnerability databases for different purposes. Therein, five of the papers propose methods to use vulnerability data to predict the attributes of vulnerabilities. For example, Zhang et al. (2015) uses machine learning methods on NVD vulnerability data to predict the time to the next vulnerability. Han et al. (2017), Chen et al. (2020), Yuan et al. (2021), and Yosifova et al. (2021) also propose approaches using machine learning or deep learning to predict vulnerability severity, vulnerability relatedness, security entity relationship, and vulnerability types. The other four papers conduct research on extracting security information from vulnerability databases. Alqahtani et al. (2016) proposes a semantic web approach to extract the ontological representation of vulnerability databases as well as the traceability links toward software repositories. Guo et al. (2021) proposes a deep learning method to extract key aspects of vulnerability information from unstructured vulnerability description.

Specifically, six papers propose approaches toward vulnerability classification. Therein, two propose methods to use classification models to predict potential vulnerability attributes. Bozorgi et al. (2010) proposes to use trained linear support vector machines (SVM) classifiers to predict whether and how soon a vulnerability is likely to be exploited. Perl et al. (2015) also uses SVM classification method to detect suspicious commits. Meanwhile, four papers propose a classification based on different vulnerability information. Huang et al. (2010) proposes a text clustering method on the vulnerability descriptions from NVD where 45 main clusters are detected as the main taxonomies. Chen et al. (2010) proposes a classification framework using SVM on the diverse taxonomic features of multiple vulnerability databases, which also the phenomena that the majority of the security risks are harbored by a small set of services. Wen et al. (2015) proposes an automatic categorization framework of vulnerabilities using text mining. Vanamala et al. (2020) uses topic modeling to classify existing vulnerability topics toward OWASP top 10 vulnerabilities.

21 selected papers provide approaches to merge multiple vulnerability databases for various purposes. Seven papers propose merging approaches as is. For example, Roschke et al. (2009) proposes an approach toward unifying vulnerability information for attack graph construction; Wang and Guo (2009) proposes an ontological approach for vulnerability management to connect vulnerabilities in NVD with additional information, e.g., inference rules, knowledge representation, etc. Meanwhile, eight papers propose methods to merge existing vulnerability databases and potentially other sources of information toward creating new ones. For example, Yap and Zhong (2004) proposes a new vulnerability database, Movtraq, by integrating general vulnerability

information with additional environmental requirements information and vulnerability impact from CERT, Bugtraq, etc. Zheng et al. (2011) proposes an alliance model, named IVDA, which aims to integrate security databases managed by security organizations from different countries. This study also proposes the international vulnerability description (IVD) to identify vulnerabilities and avoid redundancy. Furthermore, two papers propose approaches toward new vulnerability database creation by extracting information. Reis and Abreu (2017) proposes a new database of "real" security vulnerabilities, SECBENCH, by mining the millions of commits for 16 different vulnerability patterns from GitHub repositories. Herein, the authors refer to "real" vulnerabilities as the ones in contrast to the artificial hand-seeded faults used in empirical studies due to the challenges of vulnerability extraction or reproduction. Gkortzis et al. (2018) proposes a new dataset of security vulnerabilities in open-source systems, VulinOSS, which maps the OSS version references and various vulnerability reports from NVD.

Five papers contribute to other purposes, including comparing the information contents of different vulnerability databases (Tripathi and Singh, 2011, Forain et al., 2022), multi-class vulnerability detection (Zou et al., 2019), and vulnerability information quality enhancement (Glanz et al., 2015; Takahashi and Inoue, 2016; Kuehn et al., 2021).

4.3. RQ_3 . What are the other sources of information adopted to facilitate such goals?

For RQ3, we investigate what other resources of information are adopted by the selected papers that facilitate their studies toward the above mentioned goals. These information sources are categorized as follows.

- *Vulnerability databases* - The public platforms providing a record of existing vulnerability information (e.g., NVD, CVE, SecurityFocus, etc.)
- *Project data* - The set of information regarding any software projects and products (e.g., GitHub projects, Derby, Chromium, etc.)
- *Identifier* - The pre-defined indicator sets that facilitate fast and accurate correlation of configuration data across multiple information sources and tools (e.g., CCE, CPE, etc.)
- *Doc and Articles* - The collections of documents and articles that contain security and vulnerability related information (e.g., Microsoft Knowledge Database, Cybersecurity news, etc.)
- *Bug report databases* - The collections of bug reports from bug tracking systems or testing tools for specific software projects or from software collaboration platforms (e.g., Bugzilla, LaunchPad, etc.)
- *Others* - The other sources that provide additional information.

Within the 69 selected papers, 38 of them employed other information sources besides vulnerability databases listed in Table 4. Therein, four main types of information have been identified with the number of selected papers adopting each type of source shown in Table 6.

Therein, 15 selected papers adopted specific software project data or software projects databases as the additional information sources. Amongst them, 11 papers utilized software repository information from GitHub. For example, Perl et al. (2015) uses the commit data, specifically the vulnerability-contributing commits, from GitHub projects, together with the CVE database, conducting the large-scaled mapping between them. Jimenez et al. (2016) uses the source code of Android operating system from GitHub to investigate the comprehensive list of issues leading to Android vulnerabilities. Reis and Abreu (2017) also uses the source code data from 248 GitHub projects, as well as the commit messages to investigate the different patterns of security vulnerabilities and attacks. Besides GitHub data, other software project data sources are adopted, including Maven project data (Alqahtani et al., 2016), Software Heritage Graph Dataset (Antal et al., 2020), NASA Metrics Data Program (MDP), and OpenBSD project data (Massacci and Nguyen , 2010), Chromium project data (Camilo et al., 2015), and PROMISE repository data (Massacci and Nguyen , 2010).

Five papers use the data from bug report systems, e.g., Bugzilla and Bugtraq, supporting their study with vulnerability databases. For example, both Bugzilla and Bugtraq data are used in Massacci and Nguyen (2010) as part of the database comparison. Nguyen and Massacci (2012) uses bug report data from Bugzilla together with NVD data to investigate the impact of different vulnerability definitions on the "goodness-of-fitness" of vulnerability discovery models. Alves et al. (2016) uses Bugzilla data together with the vulnerabilities information of five projects to investigate the relation between software metrics and existing vulnerabilities. Chen et al. (2020) uses Bugzilla data as part of the training data, together with data from different issue trackers (e.g., Jira tickets, GitHub issues) toward the prediction of vulnerability relatedness. Hong et al. (2022) proposes a high-coverage approach collecting known security patches by tracking multiple data sources including issue trackers like Bugzilla, GitHub projects, and information from Stack Overflow.

Meanwhile, the Common Platform Enumeration (CPE) Dictionary as a configuration identifier for vulnerabilities is also commonly adopted. For example, Wang and Guo (2009) uses CPE as one of the critical information sources for the proposed ontology for vulnerability management. Fedorchenko et al. (2015) and Zhang et al. (2015) also use CPE for the integration of vulnerability-related information for the purposes of database merging and vulnerability prediction respectively.

Security related documents and articles are also used to support the studies on vulnerabilities. For example, Reis and Abreu (2017) aims to design a database for security testing with vulnerabilities mined from GitHub where the OWASP Top 10 2017 is used for the identification of a considerable amount of trending security patterns. Vanamala et al. (2020) also adopts the OWASP Top 10 risks as the vulnerability types for the proposed topic modeling and classification of the CVE database. ThreatPost and Microsoft Security Bulletin are also used as additional information sources supporting vulnerability database integration (ur Rahman et al., 2016) and database comparison (Massacci and Nguyen , 2010).

Furthermore, there are other types of vulnerability-related information sources commonly adopted by a number of the selected papers. Therein, CVSS is the most adopted information utilized by 15 selected papers where it quantifies the evaluation of vulnerability impact. For example, five studies, Frei et al. (2006), Gallon (2011), Chang et al. (2011), Liu and Zhang (2011), and Jiang et al. (2019), investigate the trends and distribution of vulnerabilities in databases in terms of CVSS; Seven studies, Roschke et al. (2009), Wang and Guo (2009), Shahzad et al. (2012), Kuo et al. (2013), Kim et al. (2013), ur Rahman et al. (2016), and Jiang et al. (2019), propose approaches to merge vulnerability databases also taken into account CVSS as one of their key information resources; CVSS is also used in two studies toward vulnerability-related predictions, i.e., Zhang et al. (2015), Han et al. (2017). In addition, OVAL, Jira issues, CAPEC, CRE, ERI, SCAP, Code gadgets, emails from the OSS project mailing list, and user contributed attacks and vulnerabilities are also used as information sources in 10 studies reported in Table 6.

Table 6
Other information sources adopted.

Type	Source	Description	Selected papers
Project DB	GitHub data	information retrieved from GitHub	Perl et al. (2015), Jimenez et al. (2016), Reis and Abreu (2017), Gkortzis et al. (2018), Ponta et al. (2019), Jimenez et al. (2019), Fan et al. (2020), Chen et al. (2020), Nikitopoulos et al. (2021), Wang et al. (2021), Hong et al. (2022)
	Maven Project	Apache Maven is a software project management and comprehension tool.	Alqahtani et al. (2016)
	Software Heritage Graph Dataset	The dataset links together file content identifiers, source code directories, Version Control System (VCS) commits tracking evolution over time.	Antal et al. (2020)
	OpenBSD Project	A FREE, multi-platform 4.4BSD-based UNIX-like operating system	Massacci and Nguyen (2010)
	NASA MDP	NASA IV&V Facility Metrics Data Program	Massacci and Nguyen (2010)
	Chromium Project	Include Chromium and Chromium OS, the open-source projects.	Camilo et al. (2015)
	PROMISE	a public repository that hosts many sanitized data sets used in many prediction models.	Massacci and Nguyen (2010)
Bug Report	Bugzilla	a web-based general-purpose bug tracking system and testing tool	Massacci and Nguyen (2010), Nguyen and Massacci (2012), Alves et al. (2016), Chen et al. (2020), Hong et al. (2022)
	Bugtraq	An electronic mailing list dedicated to issues about computer security.	Massacci and Nguyen (2010)
	Jira	a proprietary issue tracking product	Chen et al. (2020)
Identifier	CCE	unique identifiers to system configuration issues	Fedorchenko et al. (2017)
	CPE	a structured naming scheme for information technology systems, software, and packages	Wang and Guo (2009), Fedorchenko et al. (2015), Zhang et al. (2015), Takahashi and Inoue (2016), Fedorchenko et al. (2017), Jiang et al. (2019, 2021)
Doc&Articles	OWASP	OWASP Top 10 is a standard awareness document for developers and web application security.	Reis and Abreu (2017), Vanamala et al. (2020)
	ThreatPost	an independent news site which is a leading source of information about IT and business security.	ur Rahman et al. (2016)
	MS Security Bulletin	Security bulletin for The Microsoft Security Response Center	Massacci and Nguyen (2010)
	StackOverflow	Q&A Forum	Hong et al. (2022)
Others	CVSS	an open framework for communicating the characteristics and severity of software vulnerabilities.	Frei et al. (2006), Roschke et al. (2009), Wang and Guo (2009), Bozorgi et al. (2010), Gallon (2011), Chang et al. (2011), Liu and Zhang (2011), Shahzad et al. (2012), Kuo et al. (2013), Kim et al. (2013), Zhang et al. (2015), ur Rahman et al. (2016), Han et al. (2017), Jiang et al. (2019), Kuehn et al. (2021)
	OVAL	A community-developed language for determining vulnerability and configuration issues	Roschke et al. (2009), Zhang et al. (2015), Aksu et al. (2018)
	CAPEC	a comprehensive dictionary of known patterns of attack employed by adversaries	Wang and Guo (2009), Xianghui et al. (2015), Fedorchenko et al. (2017), Jiang et al. (2019), Yuan et al. (2021)
	CRE	Common Remediation Enumeration (CRE)	Fedorchenko et al. (2017)
	ERI	Extended Remediation Information (ERI)	Fedorchenko et al. (2017)
	User contribution	user reported attacks, and vulnerabilities.	Nerwich et al. (2020)
	SCAP	Security Content Automation Protocol	Wang and Guo (2009)
	Code gadgets	(code) statements that are semantically related to each other.	Zou et al. (2019)
	Emails	emails from OSS projects mailing list	Chen et al. (2020)

4.4. RQ_4 . What are the methods and techniques adopted?

For RQ4, we investigate and summarize the methods and techniques applied by the selected papers in terms of how they utilize the vulnerability databases and other information sources toward the goals mentioned above. The methods and techniques are categorized as follows.

- *Info Integration* - The approach is a combination of traditional methods to match relevant information manually (e.g., reading vulnerability reports and matching them to source code), using pre-defined identifiers (e.g., using CVE Id match vulnerabilities from different vendors) or with source code (e.g., using basic string manipulation and compare methods to match users' report to NVD vulnerabilities).
- *Statistics* - Statistical methods are applied to the vulnerability-related information sources to gain further insights.
- *Machine Learning* - Using machine learning algorithms (e.g., Naive Bayes, Logistic Regression, Decision Tree, etc.) to support the analysis of vulnerability database information.
- *Deep Learning* - Using deep learning algorithms (e.g., CNN, RNN, etc.) to support the analysis of vulnerability database information.
- *Data Collecting* - Collecting vulnerability data from public databases or other sources via web scraping or other crawling techniques to support the analysis of vulnerability database information.
- *Text Analysis* - Analyzing the collected textual data manually or using NLP techniques to extract insights from vulnerability-related information.

As shown in Fig. 6, within the 69 selected papers, nearly half (33 papers) adopt the conventional information integration methods to analyze, merge or utilize vulnerability databases. Therein, for 19 papers, such a method serves the purpose of database merging. Four of the papers propose the construction of common security-based databases using identifiers. For example,

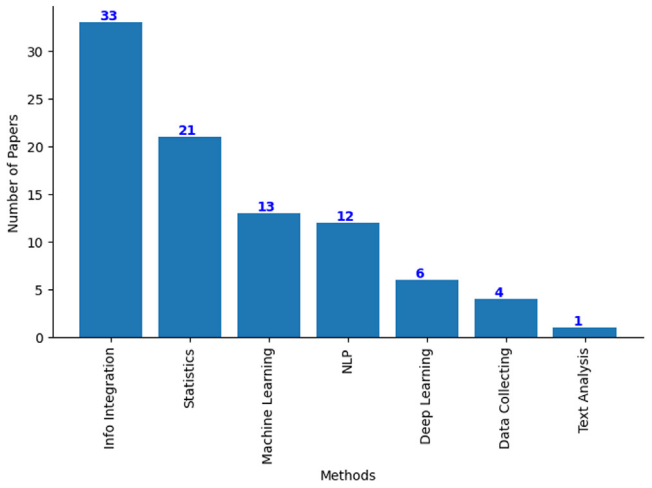

Fig. 6. Method distribution.

Yun-hua and Pei (2010), Shahzad et al. (2012), and Kuo et al. (2013) propose the integration of information from multiple existing vulnerability databases using CVEID as an identifier. Ponta et al. (2019) propose similar integration of databases but also mention, besides CVEID, project-specific identifiers can also be used. Two of the papers propose a common schema of relations between security info to integrate vulnerability databases and other sources. Both Fedorchenko et al. (2015) and Fedorchenko et al. (2017) propose ontological approaches to integrate security information, including both dynamical and static content, via a common schema of the relations. Meanwhile, four of the papers (Yap and Zhong, 2004; Roschke et al., 2009; Wang and Guo, 2009; Nerwich et al., 2020) extract information from texts, such as the description of the vulnerabilities, to support the integration. Commit histories are also used as the connector between vulnerabilities and project commits by Fan et al. (2020), Nikitopoulos et al. (2021). Zheng et al. (2011) proposes to use systematic policies and language to archive international vulnerability databases using Vulnerability Citation Index (VCI) as a unified identifier to avoid redundancy. There are also four papers that do not specify by what identifiers to merge the databases and/or other information.

Furthermore, 26 of the papers combine information from vulnerability databases with possibly other sources for database integration or analyzing and exploring in-depth insight therein. For example, Huang et al. (2020) builds a prototype system that accesses and integrates information, such as, exploit script, the configuration of software, proof of vulnerability, and vulnerability description . The system can be used to store and automatically process vulnerability information for cloud-native application vulnerability database. Antal et al. (2020) integrates the information of vulnerabilities and that of vulnerability resolution commits to analyze the typical security issue types and the security issues reaction times. Allodi and Massacci (2012) construct a new database by integrating the standard vulnerability form NVD with the ones currently used in exploit kits from the black market. ur Rahman et al. (2016), Kim et al. (2013) propose high-abstraction level system design which indicates the integration of multiple vulnerability databases and other sources of information.

In the results, nearly one third (21) of the selected papers adopt statistical methods. For example, Frei et al. (2006) uses statistical distribution, e.g., Exponential, Pareto, and Weibull, to analyze the exploit availability of vulnerabilities. Similarly, Vache (2009) uses the probability distribution to characterize the vulnerability life cycle and exploit creation events when Massacci and Nguyen (2010) analyzes the distribution of vulnerability severity ranking levels on NVD and the trend of these severity ranking levels by years. Camilo et al. (2015) adopts the Cohen's D static (Cohen, 2013) to examine the amount of overlap between neutral and vulnerable files with respect to a number of bugs. Wu et al. (2020) uses the Wilcoxon rank-sum test to analyze the statistical significance amongst classifiers. Furthermore, correlation analysis is also applied, for example, Edwards and Chen (2012) analyzes the correlation between issue density and annual vulnerability.

Meanwhile, AI-based approaches, including machine learning (ML), deep learning (DL), and natural language processing (NLP), are also commonly applied to the large volume of vulnerability data. Therein, several studies propose ML-based methods for vulnerability classification. For example, Yosifova et al. (2021) adopts Linear Support Vector, Naive Bayes and Random Forest Classifier to classify and predict vulnerability types. Perl et al. (2015) also uses Linear Support Vector Machines (SVM) to classify commits data toward detecting the ones that contribute to vulnerabilities. Wen et al. (2015) uses also SVM algorithm with Radial kernel function to train and classify the vulnerability data in the target database. Xiaomeng et al. (2018) proposes using deep-learning algorithm, e.g., CNN or RNN, on the vectorized representation of regularized code data toward vulnerability analysis. Yuan et al. (2021) also uses CNN-based graph attention network toward the prediction of security entity relationships. Furthermore, NLP-based methods are also commonly applied. For example, Williams et al. (2018) uses Supervised Topical Evolution Model (STEM) on a large volume of vulnerability-describing reports from NVD to analyze the evolving trends of the vulnerabilities. Vanamala et al. (2020) also utilizes topic modeling methods to classify the entries of CVE database.

To be noted, some of the papers do not apply AI-based approaches but conduct crawling of the vulnerability data. For such situations, the method is marked as "data collecting". For example, Reis and Abreu (2017) extracts the indications of a vulnerability fix or patch committed by the developers from GitHub projects. Chen et al. (2018) propose a text mining approach to predict invalid vulnerability reports. To identify invalid CVEs, they extract text features using term frequency. However, no NLP tools or techniques are mentioned when conducting such tasks. To clarify the differences, we categorize the method as "text analysis" instead of "NLP".

4.5. RQ_5 . Which tools are proposed for adopting or investigating vulnerability databases?

For RQ5, we summarize what tools are proposed to support the research on vulnerability databases. Toward answering this research question, we investigate the tools proposed by the selected studies that utilize vulnerability databases, and possibly the other information sources identified in Table 6. Within the 69 selected papers, three of them propose tools that adopt vulnerability databases.

The proposed tools include:

- μVulDeePecker (Zou et al., 2019) is a deep learning-based multi-class vulnerability detection tool upgraded from the original VulDeePecker (Li et al., 2018). The original VulDeePecker uses Bidirectional Long-Short Time Memory (BLSTM) neural network to detect software vulnerabilities. As a binary classifier, it can only report the target source code

being vulnerable or not without detecting the vulnerability types. The μVulDeePecker tool adopts the novel concept of code attention and uses a novel feature fusion-oriented three BLSTM networks toward multi-class vulnerability detection with vulnerability types identified. For this tool, 181641 code gadgets, labeled either vulnerable or non-vulnerable, are obtained from SARD and NVD covering 40 vulnerability types, which are used as training and testing datasets.

- VCCFinder (Perl et al., 2015) is a code analysis tool that flags suspicious commits by using an SVM-based detection model. A large-scale database mapping CVE database to collected vulnerability-contributing commits (VCCs) is built for the evaluation.
- EVMAT (Aksu et al., 2018) is a dashboard solution for monitoring enterprise vulnerability levels for proper enterprise risk management. It can also automatically gather system characteristics based on OVAL and further evaluates software vulnerabilities installed in a computer resource based on the data retrieved from NVD. The tool can also provide a quantified evaluation of the vulnerability score of an enterprise.

The results show that compared to the number of selected papers, the number of proposed tools is limited. Therein, though public vulnerability databases are adopted, many are not integrated into tools directly. For example, in Zou et al. (2019), SARD and NVD are used to create training and testing datasets. These datasets are not integrated as tool components.

5. Discussion and implications

The results of the research questions elaborated within the scope of our systematic mapping study allow us to provide a number of implications for researchers and practitioners, which we discuss in the following.

On vulnerability databases and their adoption. By summarizing the previous studies on vulnerability databases, we could observe that NVD and CVE are, by far, the most commonly used vulnerability databases by researchers. At the same time, we discovered that several countries have their own national security vulnerability databases, e.g., FrSIRT from France, JVN from Japan, or CNNVD from China. So far, the main purpose of those databases is to serve as a collection and standardized reference to known vulnerabilities. Despite the availability of these alternative databases, we pointed out that researchers have rarely adopted them for research purposes. Indeed, only the work by Zheng et al. (2011) proposed a framework aiming at exploiting vulnerability databases from different countries in an effort of providing a common framework.

Perhaps more importantly, the research effort conducted to merge the pieces of information available in these vulnerability databases is still limited. According to our mapping study (see Table 6), most of the studies that attempted to merge data from different sources only focused on Github data, CPE, and CVSS, therefore neglecting the potential contributions brought by the alternative databases.

When considering this perspective, multiple challenges and open questions arise. In the first place, we call for more research aiming at assessing the ecological validity of the findings achieved by researchers so far. Indeed, the *generalizability* of the conclusions drawn in literature might be potentially be threatened by the specific characteristics of NVD and CVE and, therefore, additional insights might be identified when considering alternative databases. In the second place, the research efforts conducted to merge vulnerability data might be extended to a more comprehensive set of vulnerability databases, further contributing to the generation of *more robust benchmarks*, *more generalizable*, and *deeper* analysis/understanding of the connections between software vulnerabilities in the wild. Last but not least, a few attempts have been made to enhance existing vulnerability databases, despite the availability of other sources of information that can be used to complement them. In this sense, the findings of our mapping study may serve as a basis for novel datasets, benchmarks, and empirical investigations into vulnerabilities.

☞ *Implication #1. Our findings represent a call for researchers working in the area of software security, software vulnerability analytics, and empirical software engineering, who are called to revisit previous findings on more comprehensive sets of vulnerability databases and provide more information about the ecological validity of the findings reported in the literature.*

☞ *Implication #2. The variety of vulnerability databases identified in our work represents an opportunity for researchers to build novel, unified data sources, and benchmarks, which might be exploited to better understand the nature and the relations between software vulnerabilities. The additional sources of information available on software vulnerabilities might serve as a further instrument to enhance existing databases. Moreover, we foresee the opportunity for companies to develop new integrated data sources and provide direct API access to developers that need to continuously check the existence of possible vulnerabilities on the software component they are using.*

On the limitations of vulnerability databases. As part of our systematic mapping, we could discover that only a limited set of studies employed vulnerability databases in the context of vulnerability detection (e.g., Zou et al., 2019), database comparison (e.g., Tripathi and Singh, 2011), and database quality improvement (e.g., Glanz et al., 2015). Instead, these studies preferred uncovering vulnerability data by mining other data sources (Lin et al., 2020). This clearly indicates that the current structure and information provided by the existing vulnerability databases are limited, preventing their wider adoption in research. Very likely, most of the limitations are due to the limited amount of metadata provided to researchers. For instance, let us consider the case of vulnerability detection approaches. Most of these approaches aim at detecting vulnerabilities at line-, function, or file-level. However, most vulnerability databases do not provide fine-grained pieces of information on the location of vulnerabilities that can be actually exploited by vulnerability detection approaches. As such, the practical usefulness of vulnerability databases is threatened and our study promotes further considerations on the way vulnerability databases can be helpful for software security, both from the researcher's and practitioner's perspective. For instance, we may envision a stronger, collaborative effort involving researchers, practitioners, and government agencies with the aim of revisiting the way the vulnerabilities stored in public databases are collected and made available: in this respect, novel data collection and reporting guidelines might be devised to let the contributors of those databases be aware of the need of curating vulnerabilities with additional meta-data. At the same time, the limited amount of literature targeting data

quality contributes to having a few insights into the information needed by researchers. We, therefore, call for further research efforts on the matter, as works aiming at integrating information within those databases might be a valuable contribution to the field and enable additional research on software vulnerabilities—as also pointed out in the previous discussion and implication point.

☞ *Implication #3. Current vulnerability databases lack fine-grained information and metadata, hence threatening their suitability for research targeting vulnerability detection. More research on data quality and information needs should therefore be considered in an effort of establishing new ways through which vulnerability databases can support the research and practice on software security. We recommend practitioners and in particular the maintainers of the vulnerability databases, to complement the information, introducing fine-grained meta-data and considering data quality aspects.*

On the actionability of vulnerability databases. One of the most surprising outcomes of our systematic mapping study was concerned with how vulnerability databases have been used by researchers. Most studies employed databases to conduct empirical analyses, while only two tools were proposed within the 64 selected papers. Two other tool-proposing papers were identified during the "full-reading" step but were rejected as a consequence of the inclusion and exclusion criteria. Grieco et al. (2016) proposed VDiscover, a tool using state-of-the-art machine learning techniques to detect and predict vulnerabilities in test cases. The proposed tool utilizes a customized dataset built from the test case data from the Debian bug tracker; however, the study does not utilize any public vulnerability databases. Liu et al. (2018) proposed CBTracer to monitor software runtime executions by catching its real-time I/O traffic, which continuously builds a security database including vulnerability discovery and exploit generation. The data sources used for CBTracer include exploit challenges, e.g., Capture The Flag (CTF) challenges and Cyber Grand Challenge (CGC), with no vulnerability databases adopted. Our findings, along with the further evidence provided by the papers by Grieco et al. (2016) and by Liu et al. (2018), further highlight the limited suitability of existing vulnerability databases for research purposes other than empirical analysis. This statement is also supported by the fact that these tools are neither industrialized nor properly maintained, meaning that the tools devised on the basis of existing vulnerability databases seem to build on insufficient or incomplete pieces of information. In this sense, our findings suggest that more research should focus on how to make vulnerability databases actionable for industrially-relevant research or suitable for practitioners interested in exploiting vulnerability data for analytical instruments. For the large open-source community, especially, for the relevant research on open-source software quality and sustainability (Li et al., 2022b,a), vulnerability databases should also be seen as critical sources.

☞ *Implication #4. Current vulnerability databases seem not to provide enough support for building tools and analytical instruments. More research should target the actionability of the current databases and possibly inform researchers on how to complement the available pieces of information with industrially-relevant insights.*

6. Threats to validity

In this section, we follow the three categories of threats to validity in software engineering secondary studies proposed by Ampatzoglou et al. (2019). Compared to the four-category guideline by Wohlin et al. (2012), this categorization is more suitable for secondary studies in the software engineering domain. Herein, we discuss the *Study Selection Validity*, *Data Validity*, and *Research Validity* of our study and the potential mitigation to their impact.

Study Selection Validity: In this study, the search strategy, review protocol, and the data extraction process were entirely based on established systematic mapping guidelines (Petersen et al., 2008, 2015; Kitchenham et al., 2022). By doing so, we reduced the threats to the initial search and study filtering processes of the secondary study planning phase. Especially, the search string was formulated to include keywords identified from research questions and diversified using synonyms. Although the automated search covers most publications, we admit that potential issues and limitations may arise during the search process, such as (a) limitations on the search string and (b) searching only on article titles, which may miss some relevant studies. To mitigate the search limitations and extend the coverage of studies, we used snowballing as the complementary. We reviewed all the references listed in the selected studies and evaluated all the papers that reference the selected ones, which resulted in 26 additional relevant publications. The inclusion and exclusion criteria were defined and piloted to assist the study selection. The criteria are in line with the goal and research questions of the paper and follow the guidelines recommended by Petersen et al. (2008). Two authors conducted the study selection independently. The other authors were involved in the discussion to resolve the disagreement. Furthermore, the study selection was conducted in December 2021. At the time of preparing the submission of this study report in 2022, we executed the queries to search for relevant studies published since our last queries, and the new queries resulted in 3 additional studies which were also included in the analysis. This reduces the potential threat of an incomplete report.

Data Validity: Regarding the data extraction process, a similar procedure is conducted where the first two authors carried out an iterative and analytic process driven by the open coding method to identify the classification schema. The last three authors further reviewed and validated the codes assigned to all the selected studies. For example, the three sets of categories for RQ_2, RQ_3, and RQ_4 extracted from the open coding method largely reduced the bias in classification schema and the mapping of data. For the data analysis process, thanks to the pre-defined categories, the extracted results can be easily summarized and displayed in the forms of bar-charts and tables. On the other hand, publication bias is also a potential threat to data validity where methods, techniques and usage goals from companies are not included sufficiently due to confidential policies, which is hard to be mitigated. Such a perspective can be further investigated via industrial surveys in future studies.

Research Validity: The study can be replicated by following the replication documentation and the steps meticulously. The search strings and details on the systematic mapping study process are all described in detail in Appendix, by which the scholar can easily replicate the study. Before the start of this study, multiple discussion sessions were organized by all the authors to determine the research method. As the decision on adopting a systematic mapping study was agreed upon by all authors, it shall mitigate the threat of the research method bias. After the selection of research method, all the authors also determined the research question together via several iterations.

Table 7
Comparison to related systematic studies.

	Our Study	Alqahtani (Alqahtani, 2022)
Research Questions	In common: RQ1. VDBs commonly used	
	RQ2 Goals to employ RQ3 Other info sources RQ4 Methods and techs RQ5 Tools proposed	RQ2 Security topics RQ3 Topics changed overtime
	In common: Frei et al. (2006), Vache (2009), Neuhaus and Zimmermann (2010), Camilo et al. (2015), Shahzad et al. (2012)	
#SP	64	94
Years	2004–2022	2006–2017
Contributions	Identify the common VDBs; Focus on the goals and methods used with/for the common public VDBs in academia; Identify other info sources adopted supporting the use of VDBs; Identify tools to support such application;	Identify the common VDBs used in SE community; Focus on the common security topics and their changes overtime;
VDBs Commonly Used	In common: NVD, CVE, OSVDB, SecurityFocus	
	1337Day, Android Security Bulletins, CERT, CoopVDB, DoE-CIRC, Dragonsoft, EDB, CERT-FR, JVN, MFSA, MSRC, NVDB, SARD, Secunia, Snyk.io, VMware Security Advisories, VulDB, Vupen, X-Force, XSA, Securiteam, CNVD/CNNVD	OWASP, CWE

7. Related work

Software security vulnerabilities are a constant threat to the software industry. The exploitation of vulnerabilities can lead to unauthorized breaches and cause significant financial losses and reputational damage to both software companies and customers.

As an early form of quality assurance in software development, software vulnerability prediction is a data-driven process to leverage historical software vulnerability knowledge for classifying vulnerable code modules. McKinnel et al. (2019) performed a systematic literature review to investigate the use of artificial intelligence and machine learning techniques for software vulnerability assessment and their performance. The authors selected 31 relevant studies and identified the scalability and the need for real-time identification of exploitable vulnerabilities as the research challenges and opportunities. The authors' findings indicate the increasing attempts to leverage AI in vulnerability assessment. Similar findings are further reported in another systematic literature review by Eberendu et al. (2022) to investigate approaches to software vulnerability detection. The authors selected 55 studies published between 2015 and 2021. The results showed that besides the static and dynamic analysis, machine learning and deep learning approaches were mostly used to detect software vulnerability. Although there are studies on tools and applications for software vulnerability detection and prediction, an investigation of the use of data sources for such tools and approaches is lacking.

A recent study by Croft et al. (2022) reports challenges and solutions to data preparation for vulnerability prediction. Based on the 61 selected studies, the authors identified 16 data challenges, most of which were related to data generalizability, accessibility, label collection effort, etc. The results show that the complexity of real-world vulnerabilities and the difficulty in preparing vulnerability datasets form the major barrier to the adoption of vulnerability prediction in the industry. Similar findings have been reported in other studies on vulnerability assessment (Turhan et al., 2009; Morrison et al., 2015; McKinnel et al., 2019; Chakraborty et al., 2021). The studies show a clear need of creating high-quality datasets that provide data provenance and comprehensive information for better sharing and governance of vulnerability data (Croft et al., 2022).

There are also other studies investigating public vulnerability databases via systematic mapping study or systematic literature review. Alqahtani conducted a survey on 99 relevant software engineering research articles toward investigating the use of vulnerabilities databases in the software engineering domain Alqahtani (2022). The study focuses on the security topics covered in software engineering studies as well as in different software engineering activities. While Alqahtani's research reports on which are the commonly adopted vulnerability databases but does not investigate the methods, the related information sources, or the tools. Lin et al. also conducted a survey reviewing the literature on building high-quality vulnerability datasets (Lin et al., 2022). The study aims to investigate how data mining and data processing techniques are adopted to generate vulnerability datasets to facilitate vulnerability discovery. However, Lin et al.'s study is neither a systematic mapping study nor a systematic literature review.

We compare our study to the previously mentioned systematic mapping study on vulnerability databases, i.e., Alqahtani (2022), regarding the overlap of research questions, covered periods, number of selected studies, etc. Details of the comparison are present in Table 7.

Alqahtani's study focuses on the vulnerability databases used in software engineering domain when our study does not apply exclusions on other domains. Only five selected papers are identical (i.e., Frei et al. (2006), Vache (2009), Neuhaus and Zimmermann (2010), Camilo et al. (2015), Shahzad et al. (2012)) between the two studies. Such a difference is caused by the study focus. Alqahtani's study include papers that use vulnerability databases on the level of individual vulnerabilities. For example, Ming et al.'s study (Ming et al., 2016) on a hybrid taint analysis tool that completely decouples the program execution and taint analysis is included by Alqahtani. The study selects 10 individual vulnerabilities from CVE database as tests to evaluate

Table 8
SEGRESS checklist for the secondary study methods and our mapping study.

SEGRESS checklist items	Our mapping study
Eligibility criteria	The systematic mapping study seeks to understand the anatomy of the current availability of vulnerability databases, with the purpose of deriving limitations and open challenges that researchers might need to address to further support practitioners and researchers in devising methodologies and tools to deal with software vulnerabilities. The inclusion criteria were mainly connected to the usefulness of a study to address our research objectives, and they are: • Papers on how to use vulnerability databases • Papers on how to enhance vulnerability databases • Papers on how to create vulnerability databases • Papers proposing methods to analyze the vulnerability datasets • Papers using vulnerability databases The exclusion criteria are: • Not in English • Duplicated (post summarizing other websites) • Out of topic (using the terms for other purposes) • Non peer-reviewed papers • Research Plans, roadmaps, vision papers • Not employing any vulnerability databases Our selected studies excluded studies reported systematic review or mapping studies related to vulnerability assessment and prediction (Croft et al., 2022; McKinnel et al., 2019; Eberendu et al., 2022) and the use of vulnerability databases in software engineering process (Alqahtani, 2022), but are not primarily about methods and tools for adopting vulnerability databases, although they were referenced in the discussion of related work. All selected studies are published by Oct. 2022.
Information sources	Five most representative databases for research in software engineering are used in the study, and they are Scopus, IEEEXplore, ACM Digital Library, ScienceDirect, and Web of Science. In addition, Google Scholar was used in the forward snowballing process, since it allows to easily search for the citations.
Search Strategy	The search string was identified in the following steps: (1) identify the major terms using research questions; (2) commonly used alternative spellings and/or synonyms; (3) verify the presence of any additional terms that were not included in the first two steps. When no new terms were identified we concatenated the major terms using AND and the alternatives and synonyms using OR. The Search string is "(security OR vulnerabilit* OR weakness*)"AND "(database* OR dataset* OR repositor*)" The search string was used to search for relevant studies in the chosen databases. An initial search applying to article title and abstract resulted in over 45k results from Scopus alone. Considering the feasibility and our aim to include articles with their core contribution targeting the use of vulnerability databases, we narrowed down the scope by applying the search string to the article title. We received 1205 non-duplicated studies, of which 42 papers were selected for the mapping study using the predefined inclusion and exclusion criteria. Each of the 42 studies then underwent another round of searches in a backward and forward snowballing process. We relied on Google Scholar to search the articles citing each of the identified studies in the forward snowballing, and browsed the reference list of each identified study in the backward snowballing. The processes were repeated for newly identified studies until no new studies were identified. The snowballing process resulted in 27 articles. Hence, the total number of identified studies was 69.
Selection Process	The selection process was conducted through independent analysis and joint discussion. The first two authors of this paper acted as inspectors. They first conducted a pilot investigation to verify the validity of the exclusion and inclusion criteria: in this respect, they first independently analyzed an initial set composed of 50 articles, randomly selected from the candidate set. After the pilot, the inspectors met and discussed the results: this procedure did not eventually lead to modifications of the exclusion and inclusion criteria, which may indicate their completeness and suitability for our study. After the inspectors completed the pilot, they independently applied the exclusion criteria to the set of retrieved articles. The analysis was first done based on the title and abstract. In cases where the inspectors were doubtful, they read the paper in full. After the independent analysis, the two inspectors compared their results in order to reach full consensus on articles that should beremoved from the analysis. In case of disagreement, the inspectors first proceeded and read the entire article and then opened a discussion. If this was not enough, the other authors of the paper were involved in the decision making process.
Data Collection Process	The data collection process was driven by the open coding research method, namely through an analytic process by which we could assign concepts (codes) to the observed data. Specifically, the following steps were performed: 1. In the initial stage, the inspectors (the first two authors) independently analyzed a subset of 10 articles and assigned codes with the aim of addressing RQ2, RQ3, and RQ4. The inspectors were free to assign as many codes as they found relevant. Afterward, they scheduled an inperson meeting to discuss about the codes assigned so far, in an effort of finding an agreement. The meeting lasted 1.5 h. In the first place, the inspectors analyzed each of the ten papers and explained how they came up with the codes — this process was performed to increase the awareness of an inspector with respect to the choices made by the other. Secondly, they discussed about their choices and found an agreement on the codes. Finally, they computed the inter-rater agreement through the Cohen's k coefficient, which measured 0.38, showing a low level of agreement.

the accuracy of our offline symbolic taint analysis in the task of software attack detection when it is neither using nor applying methods or techniques upon CVE database as a whole (Ming et al., 2016). Therefore, similar studies are excluded in our study. Furthermore, our study covers more recent studies from the year 2017 to 2022 compared to Alqahtani's paper. Alqahtani's paper contributes specifically on the identification of security topics and their changes overtime. Comparatively, our paper focuses on investigating the use of vulnerability database as a whole, as well as the methods, techniques, external information sources and tools adopted or created for such a purpose.

Especially, regarding the results coverage, our study and Alqahtani's paper have four commonly used vulnerability databases in common, i.e., NVD, CVE, OSVDB, SecurityFocus. OWASP and CWE are also included as common vulnerability databases in Alqahtani's paper. Our paper considers OWASP as external information source because *OWASP Top 10 is a standard*

Table 9
SEGRESS checklist for the secondary study methods and our mapping study (continues).

SEGRESS checklist items	Our mapping study
Data Collection Process	2. On the basis of the discussion had during the first meeting, the inspectors reworked on the codes assigned. Then, they took other 20 papers into account and proceeded with a new classification exercise. In this stage, the inspectors mainly attempted to use the codes previously emerged, yet they were allowed to define new codes whenever needed. At the completion of the task, the two inspectors scheduled another in-person meeting to open a new discussion on their work. The second meeting lasted 1 h. The Cohen's k coefficient scored 0.49 (moderate), hence showing a substantial improvement. 3. The inspectors reworked on the codes. Afterward, they started the analysis of the remaining papers. Also in this case, the inspectors were allowed to define new codes, if needed. Once the task was completed, the inspectors planned a final in-person meeting to assess their work—this lasted around 2 h. Two key insights emerged from such a meeting. First, no new codes were identified during the last coding exercise. As such, we reached the so-called theoretical saturation, namely the phase where the analysis does not propose newer insights and all concepts are developed. Second, the agreement between the inspectors scored 0.64, which may be interpreted as good. This further corroborates the completion of the data analysis. As a result, therefore, all papers were classified according to the concepts of interest. 4. As a final step, we proceeded with an additional validation of the codes assigned by the inspectors. In particular, the last three authors of the paper went through papers and codes in an effort of identifying possible inconsistencies and/or erroneous interpretations made during the first steps. This validation did not eventually lead to further modifications. Hence, we could consider the data analysis completed.
Data items	To identify the main characteristics of each selected study, the context related data items were collected. The list of data items is below. • Goal to employing vulnerability databases in each selected study were collected. The goals of study are summarized to address RQ2. • Methods and techniques applied to achieve the goals were identified to address RQ4. • Vulnerability Databases used in each study were collected and summarized to address RQ1. • Other databases and information adopted in each study were collected to summarize the range of information sources used for achieving the goals, which addresses RQ3 and complements RQ1. • Creating new vulnerability databases or merging from existing databases for custom vulnerability databases were identified and collected for each study, addressing RQ2 and complementing RQ1. • Proposed tools for investigating vulnerability databases and other information sources were collected from each selected study to address RQ5. The data was collected using the open coding research method, that is to say, through an analytic process by which we could assign concepts (codes) to the observed data, addressing RQ2, RQ3, and RQ4. It is described in the Data Collection Process.
Study Risk Of Bias Assessment	The processes of study selection and data collection were conducted through independent analysis and joint discussion. The details have been explained in the Selection Process and Data Collection Process. In the data collection process, the Cohen's k coefficient score was calculated at each step to measure the inter-rater agreement on the codes assignment by inspectors. The scores indicated substantial improvement of the reliability for codes in the data collection process.
Effect Measures	Since the research questions do not involve identifying the definition of outcome metrics used in empirical studies, the effect measures were not applied in this systematic mapping study.
Analysis and Summarizing methods	The characteristics of selected studies were sought following the data items and recorded in a shared excel file which can always be revisited. We summarized the data by identifying themes emanating from the identified codes. These identified themes gave us the categories reported in the Results section. The charts were created with selected data using Excel's chart creator, and tables were created based on the recorded data.
Reporting Bias Assessmen	Not relevant for mapping studies
Certainty Assessment	Not relevant for mapping studies

awareness document for developers and web application security.[9] To be noted, besides the Top 10 list, OWASP foundation provides tools, e.g., OWASP Dependency-Track,[10] which is a vulnerability dependency tracking tool that integrates with NVD and other vulnerability databases. Its analysis covers a number of different vulnerability databases. Meanwhile, CWE, as *a community-developed list of software and hardware weakness types,*[11] is only used together with NVD or CVE. Compared to Alqahtani's paper, we also find there are 22 other vulnerability databases used in our selected papers, which are listed in Table 7.

8. Conclusion

Vulnerability databases have been playing a crucial role in collecting, maintaining, and disseminating information about discovered software vulnerabilities, which contributes to software security. Along with the advance in computing methods and the growth of the data, it is highly required to understand how vulnerability databases are used. We conducted a systematic mapping study on the academic literature published before October 2022 in order to examine the existing body of knowledge in the adopted vulnerability databases. Based on the 69 selected papers, we investigate what are vulnerability databases commonly utilized, what other sources of information are also used, what are the goals of using them, what methods are adopted, and what tools are proposed. The summarized results show that NVD and CVE are the most commonly adopted in vulnerability database related studies with various other sources of information also adopted, e.g., software project data, bug reports, security documents, and articles, etc. Meanwhile, besides the general methods of information integration, data-driven methods are commonly adopted when studying vulnerability data. The goals of the studies are mainly focusing on the analysis, classification, comparison, enhancement, merging, and general analysis of the vulnerability database themselves while vulnerability detection is seldom studied with vulnerability databases utilized.

9 https://owasp.org/www-project-top-ten/
10 https://owasp.org/www-project-dependency-track/
11 https://cwe.mitre.org/

CRediT authorship contribution statement

Xiaozhou Li: Conceptualization, Methodology, Writing – original draft. **Sergio Moreschini:** Conceptualization, Methodology, Writing – original draft. **Zheying Zhang:** Conceptualization, Methodology, Writing – original draft. **Fabio Palomba:** Conceptualization, Supervision, Reviewing and editing. **Davide Taibi:** Conceptualization, Supervision, Reviewing and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Fabio Palomba gratefully acknowledges the support of the Swiss National Science Foundation through the SNF Projects No. PZ00P2_186090. This work has been partially supported by the EMELIOT national research project, funded by the MUR under the PRIN 2020 program (Contract 2020W3A5FY).

Appendix. Comparing our mapping study with the SEGRESS checklist for secondary study methods

See Tables 8 and 9.

References

Alqahtani, S.S., 2022. A study on the use of vulnerabilities databases in software engineering domain. Comput. Secur. 116, 102661.
Ampatzoglou, A., Bibi, S., Avgeriou, P., Verbeek, M., Chatzigeorgiou, A., 2019. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. Inf. Softw. Technol. 106, 201–230.
Azeem, M.I., Palomba, F., Shi, L., Wang, Q., 2019. Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. Inf. Softw. Technol. 108, 115–138.
Chakraborty, S., Krishna, R., Ding, Y., Ray, B., 2021. Deep learning based vulnerability detection: Are we there yet. IEEE Trans. Softw. Eng..
Cohen, J., 2013. Statistical Power Analysis for the Behavioral Sciences. Routledge.
Croft, R., Xie, Y., Babar, M.A., 2022. Data preparation for software vulnerability prediction: A systematic literature review. IEEE Trans. Softw. Eng. 1.
CVE, 2022. https://cve.mitre.org/.
Dowd, M., McDonald, J., Schuh, J., 2006. The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities. Pearson Education.
Duc, A.N., Jabangwe, R., Paul, P., Abrahamsson, P., 2017. Security challenges in IoT development: a software engineering perspective. In: Proceedings of the XP2017 Scientific Workshops. pp. 1–5.
Eberendu, A.C., Udegbe, V.I., Ezennorom, E.O., Ibegbulam, A.C., Chinebu, T.I., et al., 2022. A systematic literature review of software vulnerability detection. Eur. J. Comput. Sci. Inform. Technol. 10 (1), 23–37.
Edmundson, A., Holtkamp, B., Rivera, E., Finifter, M., Mettler, A., Wagner, D., 2013. An empirical study on the effectiveness of security code review. In: International Symposium on Engineering Secure Software and Systems. Springer, pp. 197–212.
Felderer, M., Büchler, M., Johns, M., Brucker, A.D., Breu, R., Pretschner, A., 2016. Security testing: A survey. In: Advances in Computers. Vol. 101. Elsevier, pp. 1–51.
Finifter, M., Akhawe, D., Wagner, D., 2013. An empirical study of vulnerability rewards programs. In: 22nd {USENIX} Security Symposium. {USENIX} Security 13, pp. 273–288.
Ghaffarian, S.M., Shahriari, H.R., 2017. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. ACM Comput. Surv. 50 (4), 1–36.
Gonzalez, D., Alhenaki, F., Mirakhorli, M., 2019. Architectural security weaknesses in industrial control systems (ICS) an empirical study based on disclosed software vulnerabilities. In: International Conference on Software Architecture. ICSA, pp. 31–40.
Grieco, G., Grinblat, G.L., Uzal, L., Rawat, S., Feist, J., Mounier, L., 2016. 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.
Hall, T., Beecham, S., Bowes, D., Gray, D., Counsell, S., 2011. A systematic literature review on fault prediction performance in software engineering. IEEE Trans. Softw. Eng. 38 (6), 1276–1304.
Howard, M.A., 2006. A process for performing security code reviews. IEEE Secur. Priv. 4 (4), 74–79.
Hydara, I., Sultan, A.B.M., Zulzalil, H., Admodisastro, N., 2015. Current state of research on cross-site scripting (XSS)–A systematic literature review. Inf. Softw. Technol. 58, 170–186.
Iannone, E., Guadagni, R., Ferrucci, F., De Lucia, A., Palomba, F., 2022. The secret life of software vulnerabilities: A large-scale empirical study. IEEE Trans. Softw. Eng..
Kim, S., Lee, H., 2018. Software systems at risk: An empirical study of cloned vulnerabilities in practice. Comput. Secur. 77, 720–736.
Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering.
Kitchenham, B.A., Madeyski, L., Budgen, D., 2022. SEGRESS: Software engineering guidelines for reporting secondary studies. IEEE Trans. Softw. Eng..
Li, X., Moreschini, S., Pecorelli, F., Taibi, D., 2022a. OSSARA: abandonment risk assessment for embedded open source components. IEEE Softw. 39 (04), 48–53.
Li, X., Moreschini, S., Zhang, Z., Taibi, D., 2022b. Exploring factors and metrics to select open source software components for integration: An empirical study. J. Syst. Softw. 188, 111255.
Li, Z., Zou, D., Xu, S., Ou, X., Jin, H., Wang, S., Deng, Z., Zhong, Y., 2018. Vuldeepecker: A deep learning-based system for vulnerability detection. In: Proceedings of the 25th Annual Network and Distributed System Security Symposium. NDSS.
Lin, Y., Li, Y., Gu, M., Sun, H., Yue, Q., Hu, J., Cao, C., Zhang, Y., 2022. Vulnerability dataset construction methods applied to vulnerability detection: A survey. In: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops. DSN-W, IEEE, pp. 141–146.
Lin, G., Wen, S., Han, Q.-L., Zhang, J., Xiang, Y., 2020. Software vulnerability detection using deep neural networks: a survey. Proc. IEEE 108 (10), 1825–1848.
Liu, Y., Jianwei, Z., Zhang, C., 2018. CBTracer: Continuously building datasets for binary vulnerability and exploit research. In: Proceedings of the First Workshop on Radical and Experiential Security. pp. 1–7.
Lomio, F., Iannone, E., De Lucia, A., Palomba, F., Lenarduzzi, V., 2022. Just-in-time software vulnerability detection: Are we there yet? J. Syst. Softw. 111283.
McGraw, G., 2004. Software security. IEEE Secur. Priv. 2 (2), 80–83.
McKinnel, D.R., Dargahi, T., Dehghantanha, A., Choo, K.-K.R., 2019. A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. Comput. Electr. Eng. 75, 175–188.
Ming, J., Wu, D., Wang, J., Xiao, G., Liu, P., 2016. Straighttaint: Decoupled offline symbolic taint analysis. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. pp. 308–319.
Morrison, P., Herzig, K., Murphy, B., Williams, L., 2015. Challenges with Applying Vulnerability Prediction Models. HotSoS '15, Association for Computing Machinery, New York, NY, USA.
Murtaza, S.S., Khreich, W., Hamou-Lhadj, A., Bener, A.B., 2016. Mining trends and patterns of software vulnerabilities. J. Syst. Softw. 117, 218–228.
National Vulnerability Database (NVD), 2022. https://nvd.nist.gov/general. (Accessed 30 April 2022).
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. In: 12th International Conference on Evaluation and Assessment in Software Engineering. Vol. 12. EASE, pp. 1–10.
Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. 64, 1–18.
Plate, H., Ponta, S.E., Sabetta, A., 2015. Impact assessment for vulnerabilities in open-source software libraries. In: International Conference on Software Maintenance and Evolution. ICSME, pp. 411–420.
Sharafi, Z., Soh, Z., Guéhéneuc, Y.-G., 2015. A systematic literature review on the usage of eye-tracking in software engineering. Inf. Softw. Technol. 67, 79–107.
Svacina, J., Raffety, J., Woodahl, C., Stone, B., Cerny, T., Bures, M., Shin, D., Frajtak, K., Tisnovsky, P., 2020. On vulnerability and security log analysis: A systematic literature review on recent trends. In: International Conference on Research in Adaptive and Convergent Systems. pp. 175–180.
Tan, T., Wang, B., Tang, Y., Zhou, X., Han, J., 2019. A method for vulnerability database quantitative evaluation. Comput. Mater. Continua 61 (3), 1129–1144.
Turhan, B., Menzies, T., Bener, A.B., Di Stefano, J., 2009. On the relative value of cross-company and within-company data for defect prediction. Empirical Softw. Engg. 14 (5), 540–578.
Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: EASE 2014.
Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., 2012. Experimentation in Software Engineering. Springer.

The Selected Papers (SP_s)

Aksu, M.U., Bicakci, K., Dilek, M.H., Ozbayoglu, A.M., Tatli, E.I., 2018. Automated generation of attack graphs using NVD. In: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy. pp. 135–142.

Allodi, L., Massacci, F., 2012. A preliminary analysis of vulnerability scores for attacks in wild: The ekits and sym datasets. In: Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security. pp. 17–24.

Alqahtani, S.S., Eghan, E.E., Rilling, J., 2016. Tracing known security vulnerabilities in software repositories-A Semantic Web enabled modeling approach. Sci. Comput. Programm. 121, 153–175.

Alves, H., Fonseca, B., Antunes, N., 2016. Software metrics and security vulnerabilities: dataset and exploratory study. In: 2016 12th European Dependable Computing Conference. EDCC, IEEE, pp. 37–44.

Antal, G., Keleti, M., Hegedűs, P., 2020. Exploring the security awareness of the python and javascript open source communities. In: Proceedings of the 17th International Conference on Mining Software Repositories. pp. 16–20.

Bozorgi, M., Saul, L., Savage, S., Voelker, G.M., 2010. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 105–114.

Camilo, F., Meneely, A., Nagappan, M., 2015. Do bugs foreshadow vulnerabilities? A study of the chromium project. In: 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE, pp. 269–279.

Challande, A., David, R., Renault, G., 2022. Building a commit-level dataset of real-world vulnerabilities. In: Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy. pp. 101–106.

Chang, Y.Y., Zavarsky, P., Ruhl, R., Lindskog, D., 2011. Trend analysis of the CVE for software vulnerability management. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing. IEEE, pp. 1290–1293.

Chen, Q., Bao, L., Li, L., Xia, X., Cai, L., 2018. Categorizing and predicting invalid vulnerabilities on common vulnerabilities and exposures. In: 2018 25th Asia-Pacific Software Engineering Conference. APSEC, IEEE, pp. 345–354.

Chen, Y., Santosa, A.E., Yi, A.M., Sharma, A., Sharma, A., Lo, D., 2020. A machine learning approach for vulnerability curation. In: Proceedings of the 17th International Conference on Mining Software Repositories. pp. 32–42.

Chen, Z., Zhang, Y., Chen, Z., 2010. A categorization framework for common computer vulnerabilities and exposures. Comput. J. 53 (5), 551–580.

Decan, A., Mens, T., Constantinou, E., 2018. On the impact of security vulnerabilities in the npm package dependency network. In: Proceedings of the 15th International Conference on Mining Software Repositories. pp. 181–191.

Edwards, N., Chen, L., 2012. An historical examination of open source releases and their vulnerabilities. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. pp. 183–194.

Fan, J., Li, Y., Wang, S., Nguyen, T., 2020. AC/C++ code vulnerability dataset with code changes and CVE summaries. In: Proceedings of the 17th International Conference on Mining Software Repositories. pp. 508–512.

Fedorchenko, A., Kotenko, I.V., Chechulin, A., 2015. Integrated repository of security information for network security evaluation. J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. 6 (2), 41–57.

Fedorchenko, A.V., Kotenko, I.V., Doynikova, E.V., Chechulin, A.A., 2017. The ontological approach application for construction of the hybrid security repository. In: 2017 IEEE International Conference on Soft Computing and Measurements. IEEE, pp. 525–528.

Forain, I., de Oliveira Albuquerque, R., de Sousa Júnior, R.T., 2022. Towards system security: What a comparison of National Vulnerability databases reveals. In: 2022 17th Iberian Conference on Information Systems and Technologies. CISTI, IEEE, pp. 1–6.

Frei, S., May, M., Fiedler, U., Plattner, B., 2006. Large-scale vulnerability analysis. In: Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense. pp. 131–138.

Gallon, L., 2011. Vulnerability discrimination using cvss framework. In: 2011 4th IFIP International Conference on New Technologies, Mobility and Security. IEEE, pp. 1–6.

Gkortzis, A., Mitropoulos, D., Spinellis, D., 2018. Vulinoss: a dataset of security vulnerabilities in open-source systems. In: Proceedings of the 15th International Conference on Mining Software Repositories. pp. 18–21.

Glanz, L., Schmidt, S., Wollny, S., Hermann, B., 2015. A vulnerability's lifetime: enhancing version information in CVE databases. In: Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business. pp. 1–4.

Guo, H., Xing, Z., Chen, S., Li, X., Bai, Y., Zhang, H., 2021. Key aspects augmentation of vulnerability description based on multiple security databases. In: 2021 IEEE 45th Annual Computers, Software, and Applications Conference. pp. 1020–1025.

Han, Z., Li, X., Xing, Z., Liu, H., Feng, Z., 2017. Learning to predict severity of software vulnerability using only vulnerability description. In: 2017 IEEE International Conference on Software Maintenance and Evolution. ICSME, IEEE, pp. 125–136.

Hong, H., Woo, S., Choi, E., Choi, J., Lee, H., 2022. xVDB: A high-coverage approach for constructing a vulnerability database. IEEE Access 10, 85050–85063.

Huang, M., Fan, W., Huang, W., Cheng, Y., Xiao, H., 2020. Research on building exploitable vulnerability database for cloud-native app. In: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference. Vol. 1. ITNEC, IEEE, pp. 758–762.

Huang, S., Tang, H., Zhang, M., Tian, J., 2010. Text clustering on national vulnerability database. In: 2010 Second International Conference on Computer Engineering and Applications. Vol. 2. IEEE, pp. 295–299.

Jiang, Y., Atif, Y., Ding, J., 2019. Cyber-physical systems security based on a cross-linked and correlated vulnerability database. In: International Conference on Critical Information Infrastructures Security. Springer, Cham, pp. 71–82.

Jiang, Y., Jeusfeld, M., Ding, J., 2021. Evaluating the data inconsistency of open-source vulnerability repositories. In: The 16th International Conference on Availability, Reliability and Security. pp. 1–10.

Jimenez, M., Papadakis, M., Bissyandé, T.F., Klein, J., 2016. Profiling android vulnerabilities. In: 2016 IEEE International Conference on Software Quality, Reliability and Security. QRS, IEEE, pp. 222–229.

Jimenez, M., Rwemalika, R., Papadakis, M., Sarro, F., Le Traon, Y., Harman, M., 2019. The importance of accounting for real-world labelling when predicting software vulnerabilities. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 695–705.

Kim, G., Oh, J., Seo, D., Kim, J., 2013. The design of vulnerability management system. Int. J. Comput. Sci. Netw. Secur. (IJCSNS) 13 (4), 19.

Kuehn, P., Bayer, M., Wendelborn, M., Reuter, C., 2021. OVANA: An approach to analyze and improve the information quality of vulnerability databases. In: The 16th International Conference on Availability, Reliability and Security. pp. 1–11.

Kuo, C.T., Ruan, H.M., Chen, S.J., Lei, C.L., 2013. Design and implementation of a self-growth security baseline database for automatic security auditing. In: Advances in Intelligent Systems and Applications. Vol. 2. Springer, Berlin, Heidelberg, pp. 177–184.

Li, Z., Zou, D., Tang, J., Zhang, Z., Sun, M., Jin, H., 2019. A comparative study of deep learning-based vulnerability detection system. IEEE Access 7, 103184-103197.

Linares-Vásquez, M., Bavota, G., Escobar-Velásquez, C., 2017. An empirical study on android-related vulnerabilities. In: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories. MSR, IEEE, pp. 2–13.

Liu, Q., Zhang, Y., 2011. VRSS: A new system for rating and scoring vulnerabilities. Comput. Commun. 34 (3), 264–273.

Massacci, F., Neuhaus, S., Nguyen, V.H., 2011. After-life vulnerabilities: a study on firefox evolution, its vulnerabilities, and fixes. In: International Symposium on Engineering Secure Software and Systems. Springer, Berlin, Heidelberg, pp. 195–208.

Massacci, F., Nguyen, V.H., 2010. Which is the right source for vulnerability studies? An empirical analysis on mozilla firefox. In: Proceedings of the 6th International Workshop on Security Measurements and Metrics. pp. 1–8.

Nerwich, M., Gauravaram, P., Paik, H.Y., Nepal, S., 2020. Vulnerability database as a service for IoT. In: International Conference on Applications and Techniques in Information Security. Springer, Singapore, pp. 95–107.

Neuhaus, S., Zimmermann, T., 2010. Security trend analysis with CVE topic models. In: 2010 IEEE 21st International Symposium on Software Reliability Engineering. IEEE, pp. 111–120.

Nguyen, V.H., Massacci, F., 2012. An independent validation of vulnerability discovery models. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. pp. 6–7, May.

Nikitopoulos, G., Dritsa, K., Louridas, P., Mitropoulos, D., 2021. CrossVul: a cross-language vulnerability dataset with commit data. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 1565–1569.

Perl, H., Dechand, S., Smith, M., Arp, D., Yamaguchi, F., Rieck, K., Fahl, S., Acar, Y., 2015. VCCFinder: Finding potential vulnerabilities in open-source projects to assist code audits. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 426–437.

Ponta, S.E., Plate, H., Sabetta, A., Bezzi, M., Dangremont, C., 2019. A manually-curated dataset of fixes to vulnerabilities of open-source software. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories. pp. 383–387.

Reis, S., Abreu, R., 2017. SECBENCH: A database of real security vulnerabilities. In: SecSE@ ESORICS. pp. 69–85.

Roschke, S., Cheng, F., Schuppenies, R., Meinel, C., 2009. Towards unifying vulnerability information for attack graph construction. In: International Conference on Information Security. Springer, Berlin, Heidelberg, pp. 218–233.

Shahzad, M., Shafiq, M.Z., Liu, A.X., 2012. A large scale exploratory analysis of software vulnerability life cycles. In: 2012 34th International Conference on Software Engineering. pp. 771–781.

Sönmez, F., 2021. Classifying common vulnerabilities and exposures database using text mining and graph theoretical analysis. In: Machine Intelligence and Big Data Analytics for Cybersecurity Applications. Springer, Cham, pp. 313–338.

Takahashi, T., Inoue, D., 2016. Generating software identifier dictionaries from Vulnerability Database. In: 2016 14th Annual Conference on Privacy, Security and Trust. PST, IEEE, pp. 417–420.

Tripathi, A., Singh, U.K., 2011. Taxonomic analysis of classification schemes in vulnerability databases. In: 2011 6th International Conference on Computer Sciences and Convergence Information Technology. ICCIT, IEEE, pp. 686–691.

ur Rahman, M., Deep, V., Multhalli, S., 2016. Centralized vulnerability database for organization specific automated vulnerabilities discovery and supervision. In: 2016 International Conference on Research Advances in Integrated Navigation Systems. RAINS, IEEE, pp. 1–5.

Vache, G., 2009. Vulnerability analysis for a quantitative security evaluation. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE, pp. 526–534.

Vanamala, M., Yuan, X., Roy, K., 2020. Topic modeling and classification of Common Vulnerabilities And Exposures database. In: 2020 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems. icABCD, IEEE, pp. 1–5.

Wang, J.A., Guo, M., 2009. OVM: An ontology for vulnerability management. In: Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies. pp. 1–4.

Wang, X., Wang, S., Feng, P., Sun, K., Jajodia, S., 2021. PatchDB: A large-scale security patch dataset. In: 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DSN, IEEE, pp. 149–160.

Wen, T., Zhang, Y., Wu, Q., Yang, G., 2015. ASVC: An automatic security vulnerability categorization framework based on novel features of vulnerability data. J. Commun. 10 (2), 107–116.

Williams, M.A., Dey, S., Barranco, R.C., Naim, S.M., Hossain, M.S., Akbar, M., 2018. Analyzing evolving trends of vulnerabilities in national vulnerability database. In: 2018 IEEE International Conference on Big Data. Big Data, IEEE, pp. 3011–3020.

Wu, B., Wang, A.J.A., 2011. EVMAT: An OVAL and NVD based enterprise vulnerability modeling and assessment tool. In: Proceedings of the 49th Annual Southeast Regional Conference. pp. 115–120.

Wu, X., Zheng, W., Chen, X., Wang, F., Mu, D., 2020. CVE-assisted large-scale security bug report dataset construction method. J. Syst. Softw. 160, 110456.

Xianghui, Z., Yong, P., Zan, Z., Yi, J., Yuangang, Y., 2015. Research on parallel vulnerabilities discovery based on open source database and text mining. In: 2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing. IIH-MSP, IEEE, pp. 327–332.

Xiaomeng, W., Tao, Z., Runpu, W., Wei, X., Changyu, H., 2018. CPGVA: Code property graph based vulnerability analysis by deep learning. In: 2018 10th International Conference on Advanced Infocomm Technology. ICAIT, IEEE, pp. 184–188.

Yap, R.H., Zhong, L., 2004. A machine-oriented integrated vulnerability database for automated vulnerability detection and processing. In: Large Installation System Administration. LISA.

Yosifova, V., Tasheva, A., Trifonov, R., 2021. Predicting vulnerability type in common vulnerabilities and exposures (CVE) database with machine learning classifiers. In: 2021 12th National Conference with International Participation. ELECTRONICA, IEEE, pp. 1–6.

Yuan, L., Bai, Y., Xing, Z., Chen, S., Li, X., Deng, Z., 2021. Predicting entity relations across different security databases by using graph attention network. In: 2021 IEEE 45th Annual Computers, Software, and Applications Conference. COMPSAC, IEEE, pp. 834–843.

Yun-hua, G., Pei, L., 2010. Design and research on vulnerability database. In: 2010 Third International Conference on Information and Computing. IEEE, pp. 209–212.

Zhang, S., Ou, X., Caragea, D., 2015. Predicting cyber risks through national vulnerability database. Inform. Secur. J. A Glob. Perspect. 24 (4–6), 194–206.

Zheng, C., Zhang, Y., Sun, Y., Liu, Q., 2011. IVDA: International vulnerability database alliance. In: 2011 Second Worldwide Cybersecurity Summit. WCS, IEEE, pp. 1–6.

Zou, D., Wang, S., Xu, S., Li, Z., Jin, H., 2019. μ VulDeePecker: A deep learning-based system for multiclass vulnerability detection. IEEE Trans. Dependable Secure Comput. 18 (5), 2224–2236.