A Survey on Data-driven Software Vulnerability Assessment and Prioritization

TRIET H. M. LE and HUAMING CHEN, CREST - The Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia

M. ALI BABAR, CREST - The Centre for Research on Engineering Software Technologies, The University of Adelaide, Australia and Cyber Security Cooperative Research Centre, Australia

Software Vulnerabilities (SVs) are increasing in complexity and scale, posing great security risks to many software systems. Given the limited resources in practice, SV assessment and prioritization help practitioners devise optimal SV mitigation plans based on various SV characteristics. The surge in SV data sources and data-driven techniques such as Machine Learning and Deep Learning have taken SV assessment and prioritization to the next level. Our survey provides a taxonomy of the past research efforts and highlights the best practices for data-driven SV assessment and prioritization. We also discuss the current limitations and propose potential solutions to address such issues.

CCS Concepts: • General and reference → Surveys and overviews; • Security and privacy → Software security engineering; • Computing methodologies → Machine learning; Neural networks;

Additional Key Words and Phrases: Software vulnerability, Vulnerability assessment and prioritization

1 INTRODUCTION

Software Vulnerabilities (SVs) are security bugs that can negatively affect the confidentiality, integrity and availability of software systems.\(^1\) The exploitation of these SVs such as the Heartbleed attack\(^2\) can damage the operations and reputation of millions of software systems and organizations globally, resulting in huge financial losses as well. Therefore, it is important to promptly remediate critical SVs as promptly as possible.

Vulnerability assessment is required to prioritize the remediation of critical SVs among a large and increasing number of SVs each year [168] (e.g., more than 18,000 SVs were reported on National Vulnerability Database (NVD) [134] in 2020\(^3\)). SV assessment includes tasks that determine various characteristics such as the types, exploitability, impact and severity levels of SVs [87]. Such characteristics help understand and select high-priority SVs to resolve early given the limited effort and resources. For example, an identified cross-site scripting (XSS) or SQL injection vulnerability in a web application will likely require an urgent remediation plan. These two types of SVs are well-known and can be easily exploited by attackers to gain unauthorized access and compromise sensitive data/information. On the other hand, an SV that requires admin access or happens only in a local network will probably have a lower priority since only a few people can initiate an attack.

There has been an active research trend to assess and prioritize SVs using increasingly large data from multiple sources. Many studies have proposed different Natural Language Processing (NLP), Machine Learning (ML) and Deep Learning (DL) techniques to leverage such data to automate various tasks such as predicting the Common Vulnerability Scoring System [54] (CVSS) metrics (e.g., [72, 107, 169]) or public exploits (e.g., [19, 22, 157]). These prediction models can learn the patterns automatically from vast SV data, which would be otherwise impossible to do manually. Such patterns are utilized to speed up the assessment and prioritization processes of ever-increasing

\(^1\)https://nvd.nist.gov/vuln
\(^2\)https://heartbleed.com
\(^3\)http://nvd.nist.gov/vuln/search/statistics?form_type=Basic&results_type=statistics&search_type=all

Authors’ addresses: Triet H. M. Le, triet.h.le@adelaide.edu.au; Huaming Chen, huaming.chen@adelaide.edu.au; M. Ali Babar, ali.babar@adelaide.edu.au, The University of Adelaide, Adelaide, Australia.
and more complex SVs, significantly reducing practitioners’ effort. Despite the rising research interest in data-driven SV assessment and prioritization, to the best of our knowledge, there has been no comprehensive survey on the state-of-the-art methods and existing challenges in this area. Our Contributions. 1 We are the first to review in-depth the research studies that automate data-driven SV assessment and prioritization tasks leveraging SV data and NLP/ML/DL techniques. 2 We categorize and describe the key tasks performed in relevant primary studies. 3 We synthesize and discuss the pros and cons of data, features, models, evaluation methods and metrics commonly used in the reviewed studies. 4 We highlight the challenges with the current practices and propose potential solutions moving forward. Our findings can provide useful guidelines for researchers and practitioners to effectively utilize data to perform SV assessment and prioritization.

Related Work. There have been several existing surveys/reviews on SV analysis and prediction, but they are fundamentally different from ours (see Table 1). Ghaffarian et al. [63] conducted a seminal survey on ML-based SV analysis and discovery. Subsequently, several studies [109, 162, 167, 203] reviewed DL techniques for detecting vulnerable code. However, these prior reviews did not describe how ML/DL techniques can be used to assess and prioritize the detected SVs. There have been other relevant reviews on using Open Source Intelligence (OSINT) (e.g., phishing or malicious emails/URLs/IPs) to make informed security decisions [51, 144, 176]. However, these OSINT reviews did not explicitly discuss the use of SV data and how such data can be leveraged to automate the assessment and prioritization processes. Moreover, most of the reviews on SV assessment and prioritization have focused on either static analysis tools [97] or mostly rule-based approaches (e.g., expert systems or ontologies) [91]. These methods rely on pre-defined patterns and struggle to work with new types and different data sources of SVs compared to contemporary ML or DL approaches presented in our survey [11, 66, 69]. Recently, Dissanayake et al. [42] reviewed the socio-technical challenges and solutions for security patch management that involves SV assessment and prioritization after SV patches are identified. Unlike [42], we focus on the challenges, solutions and practices of automating various SV assessment and prioritization tasks with data-driven techniques. We also consider all types of SV assessment/prioritization regardless of the patch availability.

Paper Outline. The remainder of the paper is organized as follows. Section 2 presents the scope and the taxonomy of papers covered in this survey. Sections 3, 4, 5, 6, 7 review the studies in each theme of the taxonomy. Section 8 identifies and discusses the common practices and respective implications for data-driven SV assessment and prioritization. Section 9 discusses the open challenges and proposes some future directions of this research area. Finally, section 10 concludes the survey.
2 SURVEY OVERVIEW

2.1 Background and Scope of the Survey

Our survey’s focus is on data-driven SV assessment and prioritization. The assessment and prioritization phases are between the SV discovery/detection and SV remediation/mitigation/fixing/patching phases in an SV management lifecycle [58]. The assessment phase unveils the characteristics of the SVs found in the discovery phase to locate “hot spots” that contain many highly critical/severe SVs and require higher attention in a system. In the prioritization phase, practitioners use the assessment outputs to devise an optimal remediation plan, i.e., the order/priority of fixing each SV, based on available human and technological resources. SVs would then be mitigated/fixed accordingly to the prioritized plan in the remediation phase. Unlike the existing surveys on rule-based or experience-based SV assessment and prioritization [42, 91, 97] that hardly utilize the potential of SV data in the wild, this survey aims to review research papers that have leveraged such data to automate tasks in this area using data-driven models. To keep our focus, we do not consider papers that only perform manual analyses or descriptive statistics (e.g., taking mean/median/variation of data) without using any data-driven models as these techniques cannot automatically assess or prioritize new SVs. We also do not directly compare the absolute performance of all the related studies as they did not use exactly the same experimental setup (e.g., data sources and model configurations). Note that we still identify and discuss in-depth the common practices and challenges of the studies as well as suggest potential directions to advance the field in sections 8 and 9.

2.2 Taxonomy of Data-driven Software Vulnerability Assessment and Prioritization

Based on the scope given in section 2.1, we identified five main themes of the relevant studies in the area of data-driven SV assessment and prioritization, as shown in Figure 1. Specifically, we extracted the themes by grouping related SV assessment or prioritization tasks that the surveyed
studies aim to automate/predict using data-driven models. Note that a paper is categorized into more than one theme if that paper develops models for multiple cross-theme tasks.

We acknowledge that there can be other ways to categorize the studies. However, we assert the reliability of our taxonomy as all of our themes (except theme 5) align with the security standards used in practice. For example, Common Vulnerability Scoring System (CVSS) [54] provides a framework to characterize exploitability, impact and severity of SVs (themes 1-3), while Common Weakness Enumeration (CWE) [127] includes many vulnerability types (theme 4). Hence, we believe our taxonomy can help identify and bridge the knowledge gap between the academic literature and industrial practices, making it relevant and potentially beneficial for both researchers and practitioners. Details of each theme in our taxonomy are covered in subsequent sections.

3 EXPLOITATION PREDICTION

This section covers the Exploitation theme that automates the detection and understanding of both Proof-of-Concept (PoC) and real-world exploits targeting identified SVs. The outputs of this theme can give insights into the origin of SVs as well as how and when attackers would take advantage of such SVs to compromise a system of interest, assisting practitioners to quickly react to the more easily exploitable or already exploited SVs. The papers in this theme can be categorized into three groups/sub-themes: (i) Exploit likelihood, (ii) Exploit time, (iii) Exploit characteristics, as summarized in Tables 2, 3 and 4, respectively.

3.1 Exploit Likelihood

The first sub-theme is exploit likelihood that predicts whether SVs would be exploited in the wild or PoC exploits would be released publicly (see Table 2). In 2010, Bozorgi et al. [19] were the first to use SV descriptions on Common Vulnerabilities and Exposures (CVE) [126] and Open Source Vulnerability Database (OSVDB) to predict exploit existence based on the labels on OSVDB. In 2015, Sabottke et al. [157] conducted a seminal study that used Linear SVM and SV information on Twitter to predict PoC exploits on ExploitDB [160] as well as real-world exploits on OSVDB, Symantec’s attack signatures [20] and private Microsoft’s security advisories [122]. These authors urged to explicitly consider real-world exploits as not all PoC exploits would result in exploitation in practice. They also showed SV-related information on Twitter can enable earlier detection of exploits than using expert-verified SV sources (e.g., NVD).

Built upon these two foundational studies [19, 157], the literature has mainly aimed to improve the performance and applicability of exploit prediction models by leveraging more exploit sources and/or better data-driven techniques/practices. Many researchers [4, 5, 47, 48, 83, 177] increased the amount of ground-truth exploits using extensive sources other than ExploitDB and Symantec in [19, 157]. The sources were security advisories such as Zero Day Initiative [121], Metasploit [148], SecurityFocus [81], Recorded Future [60], Kenna Security [90], Avast, ESET [50], Trend Micro [120], malicious activities in hosts based on traffic of spam/malicious IP addresses [196] and Darkweb sites/forums/markets [137]. In addition to enriching exploit sources, better data-driven models and practices for exploit prediction were also studied. Ensemble models (e.g., Random forest, eXtreme Gradient Boosting (XGBoost) [33], Light Gradient Boosting Machine (LGBM) [89]) were shown to outperform single-model baselines (e.g., Naïve Bayes, SVM, Logistic regression and Decision tree) for exploit prediction [40, 52, 79, 83]. Additionally, Bullough et al. [22] identified and addressed

---

4 An exploit is a piece of code used to compromise vulnerable software [157]. Real-world exploits are harmful & used in real host/network-based attacks. PoC exploits are unharmful & used to show the potential threats of SVs in penetration tests.

5 http://osvdb.org. Note that this database has been discontinued since 2016.

6https://twitter.com

7https://avast.com/exploit-protection.php. This link was provided by de Sousa et al. [40], but it is no longer available.
Table 2. List of the surveyed papers in the Exploit Likelihood sub-theme of the Exploitation theme. Note: The nature of task of this sub-theme is binary classification of existence/possibility of proof-of-concept and/or real-world exploits.

| Study | Data source | Data-driven technique |
|-------|-------------|-----------------------|
| Bozorgi et al. 2010 [19] | CVE, Open Source Vulnerability Database (OSVDB) | Linear Support Vector Machine (SVM) |
| Sabotik et al. 2015 [157] | NVD, Twitter, OSVDB, ExploitDB, Symantec security advisories, private Microsoft security advisories | Linear SVM |
| Edkrautz et al. 2015 [47, 48] | NVD, Recorded Future security advisories, ExploitDB | Naive Bayes, Linear SVM, Random forest |
| Bullough et al. 2017 [22] | NVD, Twitter, ExploitDB | Linear SVM |
| Almukaynizi et al. [4, 5] | NVD, ExploitDB, Zero Day Initiative security advisories & Darkweb forums/markets | SVM, Random forest, Naive Bayes, Bayesian network, Decision tree, Logistic regression |
| Xiao et al. 2018 [196] | NVD, SecurityFocus security advisories, Symantec Spam/malicious activities based on daily blacklists from abuseat.org, spamhaus.org, spamcop.net, uceprotect.net, wphlinfo & list of unpatched SVs in hosts | Identification of malicious activity groups with community detection algorithms + Random forest for exploit prediction |
| Tavabi et al. 2018 [177] | NVD, 200 sites on Darkweb, ExploitDB, Symantec, Metasploit | Paragraph embedding + Radial basis function kernel SVM |
| de Sousa et al. 2020 [40] | NVD, Twitter, ExploitDB, Symantec | Linear SVM, Logistic regression, XGBoost, Light Gradient Boosting Machine (LGBM) |
| Fang et al. 2020 [52] | NVD, ExploitDB, SecurityFocus, Symantec | fastText + LGBM |
| Huang et al. 2020 [79] | NVD, CVE Details, Twitter, ExploitDB, Symantec security advisories | Random forest |
| Jacobs et al. 2020 [83] | NVD, Kenna Security \[Exploit DB, Metasploit, FortiGuard Labs, SANS Internet Storm Center, Securewords CTU, AlienVault OSSIM, Canvas/D2 Security’s Elliot Exploitation Frameworks, Contagio, Reversing Labs\] | XGBoost |
| Yin et al. 2020 [200] | NVD, ExploitDB, General text: Book Corpus & Wikipedia for pretraining BERT models | Fine-tuning BERT models pretrained on general text |
| Bhatt et al. 2021 [14] | NVD, ExploitDB | Features augmented by SV types + Decision tree, Random forest, Naive Bayes, Logistic regression, SVM |
| Suciu et al. 2021 [174] | NVD, Vulners database, Twitter, Symantec, SecurityFocus, IBM X-Force Threat Intelligence \[Exploit sources: ExploitDB, Metasploit, Canvas, D2 Security’s Elliot, Tenable, Skybox, AlienVault, Contagio\] | Multi-layer perceptron |
| Younis et al. 2014 [202] | Vulnerable functions from NVD (Apache HTTP Server project), ExploitDB, OSVDB | SVM |
| Yan et al. 2017 [199] | Executables (binary code) of 100 Linux applications | Combining ML (Decision tree) output & fuzzing with a Bayesian network |
| Tripathi et al. 2017 [179] | Program crashes from VDiscovery \[26, 67\] & LAVA [43] datasets | Static/Dynamic analysis features + Linear/Radial basis function kernel SVM |
| Zhang et al. 2018 [204] | Program crashes from VDiscovery \[26, 67\] dataset | n-grams of system calls from execution traces + Online passive-aggressive classifier |

Several issues with exploit prediction models, e.g., time sensitivity of SV data, already-exploited SVs before disclosure and training data imbalance, helping to improve the practical application of such models. Recently, Yin et al. [200] demonstrated that transfer learning is an alternative solution for improving the performance of exploit prediction with scarcely labeled exploits. Specifically, these authors pre-trained a DL model, BERT [41], on massive non-SV sources (e.g., text on Book Corpus [209] and Wikipedia [59]) and then fine-tuned this pre-trained model on SV data using additional pooling and dense layers. Bhatt et al. [14] also suggested that incorporating the types of SVs (e.g., SQL injection) into ML models can further enhance the predictive effectiveness. Suciu...
| Study              | Nature of task                                      | Data source                                                                 | Data-driven technique                      |
|-------------------|----------------------------------------------------|----------------------------------------------------------------------------|---------------------------------------------|
| Bozorgi et al.    | Binary classification: Likelihood that SVs would be exploited within 2 to 30 days after disclosure | CVE, OSVDB                                                                  | Linear SVM                                  |
| Edkrantz 2015     | Binary classification: Likelihood of SV exploits within 12 months after disclosure | NVD, ExploitDB, Recorded Future security advisories                        | SVM, K-Nearest Neighbors (KNN), Naïve Bayes, Random forest |
| Jacobs et al. 2019| NVD, Kenna Security Exploit sources: Exploit DB, Metasploit, D2 Security’s Elliot & Canvas Exploitation Frameworks, Fortinet, Proofpoint, AlienVault & GreyNoise | Logistic regression                                                          |                                             |
| Chen et al. 2019  | Binary classification: Likelihood that SVs would be exploited within 1/3/6/9/12 months after disclosure Regression: number of days until SV exploits after disclosure | CVE, Twitter, ExploitDB, Symantec security advisories                      | Graph neural network embedding + Linear regression, Bayes, Random forest, XGBoost, Lasso/Ridge regression |

et al. [174] empirically showed that unifying SV-related sources used in prior work (e.g., SV databases [19], social media [157], SV-related discussions [177] and PoC code in ExploitDB [83]) supports more effective and timely prediction of functional exploits [57].

Besides using SV descriptions as input for exploit prediction, several studies in this sub-theme have also predicted exploits on the code level. Younis et al. [202] predicted the exploitability of vulnerable functions in the Apache HTTP Server project. Specifically, these authors used an SVM model with features extracted from the dangerous system calls [13] in entry points/functions [115] and the reachability from any of these entry points to vulnerable functions [77]. Moving from high-level to binary code, Yan et al. [199] first used a Decision tree to obtain prior beliefs about SV types in 100 Linux applications using static features (e.g., `hexdump`) extracted from executables. Subsequently, they applied various fuzzing tools (i.e., Basic Fuzzing Framework [24] and OFuzz [25]) to detect SVs with the ML-predicted types. They finally updated the posterior beliefs about the exploitability based on the outputs of the ML model and fuzzers using a Bayesian network. The proposed method outperformed `exploitable`\(^8\) a static crash analyzer provided by Microsoft. Tripathi et al. [179] also predicted SV exploitability from crashes (i.e., VDiscovery [26, 67] and LAVA [43] datasets) using an SVM model and static features from core dumps and dynamic features generated by the Last Branch Record hardware debugging utility. Zhang et al. [204] proposed two improvements to Tripathi et al. [179]’s approach. These authors first replaced the hardware utility in [179] that may not be available for resource-constrained devices (e.g., IoT) with sequence/\(n\)-grams of system calls extracted from execution traces. They also used an online passive-aggressive classifier [37] to enable online/incremental learning of exploitability for new crash batches on-the-fly.

### 3.2 Exploit Time

After predicting the likelihood of SV exploits in the previous sub-theme, this sub-theme provides more fine-grained information about the exploit time (see Table 3). Besides performing binary classification of exploits, Bozorgi et al. [19] and Edkrantz [47] also predicted the time frame (2-30 days in [19] and 12 months in [47]) within which exploits would happen after the disclosure of SVs. Jacobs et al. [84] then leveraged multiple sources containing both PoC and real-world exploits, as given in Table 3, to improve the number of labeled exploits, enhancing the prediction of exploit appearance within 12 months. Chen et al. [30] predicted whether SVs would be exploited within

\(^8\)https://microsoft.com/security/blog/2013/06/13/exploitable-crash-analyzer-version-1-6
1-12 months as well as the exploit time (number of days) after SV disclosure using Twitter data. The authors proposed a novel regression model whose feature embedding was a multi-layer graph neural network [95] capturing the content and relationships among tweets, respective tweets’ authors and SVs. The proposed model outperformed many baselines and was later integrated into an SV assessment system called VEST [29] to provide timely information for practitioners. To the best of our knowledge, at the time of writing, Chen et al. [29, 30] have been the only researchers pinpointing the exact exploit time of SVs rather than large/uncertain time-frames (e.g., months) in other studies, helping practitioners to devise much more fine-grained remediation plans.

3.3 Exploit Characteristics

Exploit characteristics is the final sub-theme that reveals various requirements/means of exploits (see Table 4), informing the potential scale of SVs; e.g., remote exploits likely affect more systems than local ones. The commonly used outputs are the Exploitability metrics provided by versions 2 [55] and 3 [56, 57] of Common Vulnerability Scoring System (CVSS).

Many studies have focused on predicting and analyzing version 2 of CVSS exploitability metrics (i.e., Access Vector, Access Complexity and Authentication). Yamamoto et al. [198] were the first one to leverage descriptions of SVs on NVD together with a supervised Latent Dirichlet Allocation topic model [16] to predict these CVSS metrics. Subsequently, Wen et al. [191] used Radial Basis Function (RBF)-kernel SVM and various SV databases/advisories other than NVD (e.g., SecurityFocus, OSVDB and IBM X-Force [163]) to predict the metrics. Le et al. [107] later showed that the prediction of CVSS metrics suffered from the concept drift issue; i.e., descriptions of new SVs may contain Out-of-Vocabulary terms for prediction models. They proposed to combine sub-word features with traditional Bag-of-Word (BoW) features to infer the semantics of novel terms/words from existing ones, helping assessment models be more robust against concept drift. Besides prediction, Toloudis et al. [178] used principal component analysis [194] and Spearman’s ρ correlation coefficient to reveal the predictive contribution of each word in SV descriptions to each CVSS metric. However, this technique does not directly produce the value of each metric.

Recently, several studies have started to predict CVSS version 3 exploitability metrics including the new Privileges and User Interactions. Ognawala et al. [138] fed the features generated by a static analysis tool, Macke [139], to a Random forest model to predict these CVSS version 3 metrics for vulnerable software/components. Later, Chen et al. [29] found that many SVs were disclosed on Twitter before on NVD. Therefore, these authors developed a system built on top of a Graph Convolutional Network [94] capturing the content and relationships of related Twitter posts about SVs to enable more timely prediction of the CVSS version 3 metrics. Elbaz et al. [49] developed a linear regression model to predict the numerical output of each metric and then obtained the respective categorical value with the numerical value closest to the predicted value. For example, a predicted value of 0.8 for Attack Vector CVSS v3 is mapped to Network (0.85) [56]. To prepare a clean dataset to predict these CVSS metrics, Jiang et al. [85] replaced inconsistent CVSS values in various SV sources (i.e., NVD, ICS CERT and vendor websites) with the most frequent value. Instead of building a separate model for each CVSS metric, there has been another family of approaches predicting these metrics using a single model to increase efficiency. Gawron et al. [61] and Spanos et al. [169] predicted multiple CVSS metrics as a unique string instead of individual values. The output of each metric is then extracted from the concatenated string. Later, Gong et al. [64] adopted the idea of a unified model from the DL perspective by using the multi-task learning paradigm [208] to predict CVSS metrics simultaneously. The model has a feature extraction module (based on a Bi-LSTM model with attention mechanism [8]) shared among all the CVSS metrics/tasks, yet specific prediction head/layer for each metric/task. This model outperformed single-task counterparts while requiring much less time to (re-)train.
### Table 4. List of the surveyed papers in the Exploit Characteristics sub-theme of the Exploitation theme.

| Study                        | Nature of task                                      | Data source                                                                 | Data-driven technique                                                                 |
|------------------------------|-----------------------------------------------------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Yamamoto et al. 2015 [198]   | Multi-class classification: CVSS v2 (Access Vector & Access Complexity metrics) | NVD                                                                         | Supervised Latent Dirichlet Allocation (LDA)                                           |
| Wen et al. 2015 [191]        | Binary classification: CVSS v2 (Authentication metric) | NVD, OSVDB, SecurityFocus, IBM X-Force                                      | Radial basis function kernel SVM                                                       |
| Le et al. 2019 [107]         |                                                      | NVD                                                                         | Concept-drift-aware models with Naïve Bayes, KNN, Linear SVM, Random forest, XGBoost, LGBM |
| Toloudis et al. 2016 [178]   |                                                      | NVD                                                                         | Principal component analysis & Spearman correlation coefficient                        |
| Ognawala et al. 2018 [138]   | Multi-class classification: CVSS v3 (Attack Vector, Attack Complexity & Privileges Required metrics) | NVD (buffer overflow SVs) & Source code of vulnerable software/components     | Combining static analysis tool (Macke [139]) & ML classifiers (Naïve Bayes & Random forest) |
| Chen et al. 2019 [29]        | Binary classification: CVSS v3 (User Interaction metric) | NVD                                                                         | Graph convolutional network                                                             |
| Elbaz et al. 2020 [49]       | Multi-class/Binary classification: CVSS v2/v3       | NVD                                                                         | Mapping outputs of Linear regression to CVSS metrics with closest values               |
| Jiang et al. 2020 [85]       |                                                      | NVD, ICS Cert, Vendor websites (Resolve inconsistencies with a majority vote) | Logistic regression                                                                    |
| Gawron et al. 2017 [61]      | Multi-target classification: CVSS v2               | NVD                                                                         | Naïve Bayes, Multi-layer Perceptron (MLP)                                             |
| Spanos et al. 2018 [169]     |                                                      | NVD                                                                         | Random forest, boosting model, Decision tree                                          |
| Gong et al. 2019 [64]        | Multi-task classification: CVSS v2                 | NVD                                                                         | Bi-LSTM with attention mechanism                                                       |
| Chen et al. 2010 [35]        | Multi-class classification: Platform-specific vulnerability locations (Local, Remote, Local area network) & vulnerability causes (e.g., Access/Input/Origin validation error) | NVD, Secunia vulnerability database, SecurityFocus, IBM X-Force                  | Linear SVM                                                                             |
| Ruohonen et al. 2017 [152]   | Binary classification: Web-related exploits or not  | ExploitDB                                                                   | LDA + Random forest                                                                    |
| Aksu et al. 2018 [2]         | Multi-class classification: author-defined pre-/post-condition privileges (None, OS (Admin/User), App (Admin/User)) | NVD                                                                         | RBF network, Linear SVM, NEAT [173], MLP                                               |
| Liu et al. 2019 [111]        |                                                      | NVD                                                                         | Information gain + Convolution neural network                                          |
| Kanakogi et al. 2021 [88]    | Multi-class classification: Common Attack Pattern Enumeration and Classification (CAPEC) | NVD, CAPEC                                                             | Doc2vec/tf-idf with cosine similarity                                                   |

Although CVSS exploitability metrics were most commonly used, several studies used other schemes for characterizing exploitation. Chen et al. [35] used Linear SVM and SV descriptions to predict multiple SV characteristics, including three SV locations (i.e., Local, LAN and Remote) on SecurityFocus [81] and Secunia [82] databases as well as 11 SV causes on SecurityFocus. Regarding the exploit types, Rouhonen et al. [152] used LDA [17] and Random forest to classify whether an exploit would affect a web application. This study can help find relevant exploits in components/subsystems of a large system. For privileges, Aksu et al. [2] extended the Privileges Required metric of CVSS by incorporating the context (i.e., Operating system or Application) to which privileges are

---

Access/Input/Origin validation error, Atomicity/Configuration/Design/Environment/Serialization error, Boundary condition error, Failure on exceptions, Race condition error
applied (see Table 4). They found MLP [73] to be the best-performing model for obtaining these privileges from SV descriptions. They also utilized the predicted privileges to generate attack graphs (sequence of attacks from source to sink nodes). Liu et al. [111] advanced this task by combining information gain for feature selection and Convolutional Neural Network (CNN) [93] for feature extraction. Regarding attack patterns, Kanakogi et al. [88] found Doc2vec [103] to be more effective than term-frequency inverse document frequency (tf-idf) when combined with cosine similarity to find the most relevant Common Attack Pattern Enumeration and Classification (CAPEC) [124] for a given SV on NVD. Such attack patterns can manifest how identified SVs can be exploited by adversaries, assisting the selection of suitable countermeasures.

Summary of the Exploitation theme: The primary tasks are binary classification of whether Proof-of-Concept (PoC)/real-world exploits of SVs would appear and multi-classification of exploit characteristics based on CVSS versions 2 and 3. The predictions of real-world exploits and CVSS v3 require more data to reach the same performance as PoC exploits and CVSS v2, respectively. More fine-grained prediction of exploits such as when exploits would be created or methods used to craft such exploits is also still limited.

4 IMPACT PREDICTION
This section describes the Impact theme that determines the (negative) effects that SVs have on a system of interest if such SVs are exploited. There are five key tasks that the papers in this theme have automated/predicted: (i) Confidentiality impact, (ii) Integrity impact, (iii) Availability impact, (iv) Scope and (v) Custom vulnerability consequences (see Table 5).

4.1 Confidentiality, Integrity, Availability, and Scope
A majority of the papers have focused on the impact metrics provided by CVSS, including versions 2 [55] and 3 [56, 57]. Versions 2 and 3 share three impact metrics Confidentiality, Integrity and Availability. Version 3 also has a new metric, Scope, that specifies whether an exploited SV would affect only the system that contains the SV. For example, Scope changes when an SV occurring in a virtual machine affects the whole host machine, in turn increasing individual impacts.

The studies that predicted the CVSS impact metrics are mostly the same as the ones predicting the CVSS exploitability metrics in section 3. Given the overlap, we hereby only describe the main directions and techniques of the Impact-related tasks rather than iterating the details of each study. Overall, a majority of the work has focused on classifying CVSS impact metrics (versions 2 and 3) using three main learning paradigms: single-task [29, 49, 85, 107, 138, 191, 198], multi-target [61, 169] and multi-task [64] learning. Instead of developing a separate prediction model for each metric like the single-task approach, multi-target and multi-task approaches only need a single model for all tasks. Multi-target learning predicts concatenated output; whereas, multi-task learning uses shared feature extraction for all tasks and task-specific softmax layers to determine the output of each task. These three learning paradigms were powered by applying and/or customizing a wide range of data-driven methods. The first method was to use single ML classifiers like supervised Latent Dirichlet Allocation [198], Principal component analysis [178], Naïve Bayes [61, 107, 138], Logistic regression [85], Kernel-based SVM [191], Linear SVM [107], KNN [107] and Decision tree [169]. Other studies employed ensemble models combining the strength of multiple single models such as Random forest [107, 138], boosting model [169] and XGBoost/LGBM [107]. Recently, more studies moved towards more sophisticated DL architectures such as MLP [61], attention-based (Bi-)LSTM [64] and graph neural network [29]. Ensemble and DL models usually beat the single ones, but there is a lack of direct comparisons between these two emerging model types.
Table 5. List of the surveyed papers in the Impact theme.

| Study                          | Nature of task                        | Data source                                      | Data-driven technique                                          |
|-------------------------------|---------------------------------------|--------------------------------------------------|----------------------------------------------------------------|
| Yamamoto et al. 2015 [198]    | Multi-class classification: CVSS v2   | NVD                                              | Supervised Latent Dirichlet Allocation                         |
| Wen et al. 2015 [191]         | NVD, OSVDB, SecurityFocus, IBM X-Force| Radial basis function kernel SVM                  |
| Le et al. 2019 [107]          | NVD                                   | Concept-drift-aware models with Naïve Bayes, KNN, Linear SVM, Random forest, XGBoost, LGBM |
| Toloudis et al. 2016 [178]    | Correlation analysis: CVSS v2         | NVD                                              | Principal component analysis & Spearman correlation coefficient |
| Ognawala et al. 2018 [138]    | Multi-class classification: CVSS v3   | NVD (buffer overflow SVs) & Source code of vulnerable software/components | Combining static analysis tool (Macke [139]) & ML classifiers (Naïve Bayes & Random forest) |
| Chen et al. 2019 [29]         | Binary classification: Scope in CVSS v3| CVE, NVD, Twitter                               | Graph convolutional network                                   |
| Elbaz et al. 2020 [49]        | Multi-class classification: CVSS v2/v3| NVD                                              | Mapping outputs of Linear regression outputs to CVSS metrics with closest values |
| Jiang et al. 2020 [85]        | Binary classification: Scope in CVSS v3| NVD, ICS Cert, Vendor websites (Resolve inconsistencies with a majority vote) | Logistic regression                                            |
| Gawron et al. 2017 [61]       | Multi-target classification: CVSS v2  | NVD                                              | Naïve Bayes, MLP                                              |
| Spanos et al. 2018 [169]      |                                       | NVD                                              | Random forest, boosting model, Decision tree                   |
| Gong et al. 2019 [64]         | Multi-task classification: CVSS v2    | NVD                                              | Bi-LSTM with attention mechanism                              |

Sub-theme: 5. Custom Vulnerability Consequences

| Study                          | Nature of task                        | Data source                                      | Data-driven technique                                          |
|-------------------------------|---------------------------------------|--------------------------------------------------|----------------------------------------------------------------|
| Chen et al. 2010 [35]         | Multi-label classification: Platform-specific impacts (e.g., Gain system access) | NVD, Secunia vulnerability database, SecurityFocus, IBM X-Force | Linear SVM                                                    |

4.2 Custom Vulnerability Consequences

Although CVSS metrics provide different levels of impact, in practice, practitioners may want more interpretable information about custom vulnerability consequences for a system of interest to devise more effective remediation strategies. Chen et al. [35] curated a list of 11 vulnerability consequences\(^\text{10}\) from X-Force [163] and Secunia [82] vulnerability databases. They then used a Linear SVM model to perform multi-label classification of these consequences for SVs, meaning that an SV can lead to more than one consequence. To the best of our knowledge, this is the only study that has pursued this research direction so far. We speculate that the main challenge with this direction is that there is no standardized taxonomy of vulnerability consequences like CVSS.

\(^{10}\)Gain system access, Bypass security, Configuration manipulation, Data/file manipulation, Denial of Service, Privilege escalation, Information leakage, Session hijacking, Cross-site scripting (XSS), Source spoofing, Brute-force proneness.

**Summary of the Impact theme:** The common task is to predict the impact base metrics provided by CVSS versions 2 and 3. These impact metrics are usually predicted together with the exploitability metrics given their similar nature (multi-class classification) using either task-wise models or a unified (multi-target or multi-task) model. Fine-grained SV consequences have also been explored, but they still require a more systematic taxonomy.
Table 6. List of the surveyed papers in the Severe vs. Non-Severe sub-theme of the Severity theme. Note: The nature of task here is binary classification of severe SVs with High/Critical CVSS v2/v3 severity levels.

| Study             | Data source (software project)                                                                 | Data-driven technique                                                                 |
|-------------------|-----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Kudjo et al. 2019 | NVD (Mozilla Firefox, Google Chrome, Internet Explorer, Microsoft Edge, Sea Monkey, Linux Kernel, Windows 7, Windows 10, Mac OS, Chrome OS) | Term frequency & inverse gravity moment weighting + KNN, Decision tree, Random forest |
| Chen et al. 2020  | NVD (Adobe Flash Player, Enterprise Linux, Linux Kernel, Foxit Reader, Safari, Windows 10, Microsoft Office, Oracle Business Suites, Chrome, QuickTime) | Term frequency & inverse gravity moment weighting + KNN, Decision tree, Naive Bayes, SVM, Random forest |
| Kudjo et al. 2020 | NVD (Google Chrome, Mozilla Firefox, Internet Explorer and Linux Kernel)                       | Find the best smallest training dataset using KNN, Logistic regression, MLP, Random forest |
| Malhotra et al. 2021 | NVD (Apache Tomcat)                                                                               | Chi-square/Information gain + bagging technique, Random forest, Naive Bayes, SVM         |

5 SEVERITY PREDICTION

This section discusses the work in the Severity theme. Severity is often a function/combination of Exploitation (section 3) and Impact (section 4). SVs with higher severity usually require more urgent remediation. There are three main prediction tasks in this theme: (i) Severe vs. Non-severe, (ii) Severity levels and (iii) Severity score, shown in Tables 6, 7, respectively.

Similar to the Exploitation and Impact themes, many studies in the Severity theme have used CVSS versions 2 and 3. According to both CVSS versions, the severity score shares the same range from 0 to 10, with an increment of 0.1. Based on the score, the studies have either defined a threshold to decide whether an SV is severe (requiring high attention), or predicted levels/groups of severity score that require a similar amount of attention or determined the raw score value.

5.1 Severe vs. Non-Severe

The first group of studies have aimed to classify whether an SV is severe or non-severe, making it a binary classification problem (see Table 6). These studies have typically selected severe SVs as the ones with at least High severity level (i.e., CVSS severity score is equal or larger than 7.0). Kudjo et al. [99] showed that using term frequency (BoW) with inverse gravity moment weighting [32] to extract features from SV descriptions can enhance the performance of ML models (i.e., KNN, Decision tree and Random forest) in predicting the severity of SVs. Later, Chen et al. [31] confirmed that this feature extraction method was also effective for more projects and classifiers (e.g., Naive Bayes and SVM). Besides investigating feature extraction, Kudjo et al. [98] also highlighted the possibility of finding Bellwether, i.e., the smallest set of data that can be used to train an optimal prediction model, for classifying severity. Recently, Malhotra et al. [114] revisited this task by showing that Chi-square and information gain can be effective dimensionality reduction techniques for multiple classifiers (i.e., bagging technique, Random forest, Naive Bayes and SVM).

5.2 Severity Levels

Rather than just performing binary classification of whether an SV is severe, several studies have identified one among multiple severity levels that an SV belongs to (see Table 7). This setting can be considered as multi-class classification. Spanos et al. [170] were to first one to show the applicability of ML to classify SVs into one of the three severity levels using SV descriptions. These three levels are provided by NVD and based on the severity score of CVSS version 2 [55] and WIVSS [171], i.e., Low (0.0 – 3.9), Medium (4.0 – 6.9), High (7.0 – 10.0). Note that WIVSS assigns different weights for the Confidentiality, Integrity and Availability impact metrics of CVSS, enhancing the ability to capture varied contributions of these impacts to the final severity score. Later, Wang et al. [186] showed that XGBoost [33] performed the best among the investigated ML classifiers for predicting
### Table 7. List of the surveyed papers in the Severity Levels sub-theme of the Severity theme.

| Study            | Nature of task                                                                 | Data source                                                                 | Data-driven technique                                                                 |
|------------------|--------------------------------------------------------------------------------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| Spanos et al. 2017 [170] | **Multi-class classification**: NVD severity levels based on CVSS v2 & WIVSS (High, Medium, Low) | NVD                                                                          | Decision tree, SVM, MLP                                                               |
| Wang et al. 2019 [186] | **Multi-class classification**: NVD severity levels based on CVSS v2 (High, Medium, Low) | NVD (XSS attacks)                                                            | XGBoost, Logistic regression, SVM, Random forest                                      |
| Le et al. 2019 [107]  | **Multi-class classification**: NVD severity levels based on CVSS v2 (High, Medium, Low) | NVD                                                                          | Concept-drift-aware models with Naïve Bayes, KNN, Linear SVM, Random forest, XGBoost, LGBM |
| Liu et al. 2019 [113]  | **Multi-class classification**: NVD, China National Vulnerability Database (XSS attacks) | NVD, China National Vulnerability Database (XSS attacks)                      | Recurrent Convolutional Neural Network (RCNN), Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM) |
| Sharma et al. 2020 [164] | **Multi-class classification**: Atlassian categories of CVSS severity score (Critical, High, Medium, Low) | CVE Details                                                                  | CNN                                                                                   |
| Han et al. 2017 [72]  | **Multi-class classification**: Atlassian categories of CVSS severity score (Critical, High, Medium, Low) | CVE Details                                                                  | 1-layer CNN, 2-layer CNN, CNN-LSTM, Linear SVM                                         |
| Sahin et al. 2019 [158] | **Multi-class classification**: Atlassian categories of CVSS severity score (Critical, High, Medium, Low) | NVD                                                                          | 1-layer CNN, LSTM, XGBoost, Linear SVM                                                 |
| Nakagawa et al. 2019 [132] | **Multi-class classification**: Atlassian categories of CVSS severity score (Critical, High, Medium, Low) | CVE Details                                                                  | Character-level CNN vs. Word-based CNN + Linear SVM                                   |
| Gong et al. 2019 [64]  | **Multi-task classification**: Atlassian categories of CVSS severity score (Critical, High, Medium, Low) | CVE Details                                                                  | Bi-LSTM with attention mechanism                                                      |
| Chen et al. 2010 [35]  | **Multi-class classification**: severity levels of Secunia (Extremely/highly/moderately/less/non-critical) | CVE, Secunia vulnerability database, SecurityFocus, IBM X-Force                | Linear SVM                                                                           |
| Zhang et al. 2020 [205] | **Multi-class classification**: Platform-specific levels (High/Medium/Low) | China National Vulnerability Database                                          | Logistic regression, Linear discriminant analysis, KNN, CART, SVM, bagging/boosting models |
| Khazaei et al. 2016 [92] | **Multi-class classification**: 10 severity score bins (one unit/bin) | CVE & OSVDB                                                                  | Linear SVM, Random forest, Fuzzy system                                                |

these three NVD-based severity levels. Le et al. [107] also confirmed that ensemble methods (e.g., XGBoost [33], LGBM [89] and Random forest) outperformed single models (e.g., Naïve Bayes, KNN and SVM) for this task. Predicting severity levels has also been tackled with DL techniques [113, 164] such as Recurrent Convolutional Neural Network (RCNN) [102], Convolutional Neural Network (CNN) [93], Long-Short Term Memory (LSTM) [74]. These studies showed potential performance gain of DL models compared to traditional ML counterparts. Han et al. [72] showed that DL techniques (i.e., 1-layer CNN) also achieved promising results for predicting a different severity categorization, namely Atlassian’s levels.\(^\text{11}\) Such findings were successfully replicated by Sahin et al. [158]. Nakagawa et al. [132] further enhanced the DL model performance for the same task by incorporating the character-level features into a CNN model [206]. Complementary to performance enhancement, Gong et al. [64] proposed to predict these severity levels concurrently with other CVSS metrics in a single model using multi-task learning [208] powered by an attention-based Bi-LSTM shared feature extraction model. The unified model was demonstrated to increase both the prediction effectiveness and efficiency. Besides Atlassian’s categories, several studies applied ML models to predict severity levels on other platforms such as Secunia [35] and China National Vulnerability Database\(^\text{12}\) [205]. Instead of using textual categories, Khazaei et al. [92] divided the CVSS severity score into 10 bins with 10 increments each (e.g., values of 0 – 0.9 are in one bin) and obtained decent results (86-88% Accuracy) using Linear SVM, Random forest and Fuzzy system.

\(^{11}\)https://www.atlassian.com/trust/security/security-severity-levels

\(^{12}\)https://www.cnvd.org.cn
Table 8. List of the surveyed papers in the Severity Score sub-theme of the Severity theme. Notes: † denotes that the severity score is computed from ML-predicted base metrics using the formula provided by an assessment framework (CVSS and/or WIVSS).

| Study | Nature of task | Data source | Data-driven technique |
|-------|----------------|-------------|-----------------------|
| Sahin et al. 2019 [158] | Regression: CVSS v2 (0-10) | NVD | 1-layer CNN, LSTM, XGBoost regressor, Linear regression |
| Wen et al. 2015 [191] | OSVDB, SecurityFocus, IBM X-Force | Radial basis function kernel SVM† |
| Ognawala et al. 2018 [138] | Regression: CVSS v3 (0-10) | NVD (buffer overflow SVs) | Combining a static analysis tool (Macke [139]) & ML classifiers (Naïve Bayes & Random forest)† |
| Chen et al. 2019 [28, 29] | NVD, CVE, NVD, Twitter | Graph convolutional network |
| Anwar et al. 2020 [6] | NVD | Linear regression, Support vector regression, CNN, MLP |
| Elbaz et al. 2020 [49] | Regression: CVSS v2/v3 (0-10) | NVD | Mapping outputs of Linear regression to CVSS metrics with closest values† |
| Jiang et al. 2020 [85] | NVD, ICS Cert, Vendor websites (Resolve inconsistencies with a majority vote) | Logistic regression† |
| Spanos et al. 2018 [169] | Regression: CVSS v2 & WIVSS (0-10) | NVD | Random forest, boosting model, Decision tree† |
| Toloudis et al. 2016 [178] | Correlation analysis: CVSS v2 & WIVSS (0-10) | NVD | Principal component analysis & Spearman correlation coefficient |

5.3 Severity Score

To provide even more fine-grained severity value than the categories, the last sub-theme has predicted the severity score (see Table 8). Using SV descriptions on NVD, Sahin et al. [158] compared the performance of ML-based regressors (e.g., XGBoost [33] and Linear regression) and DL-based ones (e.g., CNN [93] and LSTM [74]) for predicting the severity score of CVSS version 2 [55]. These authors showed that DL-based approaches generally outperformed the ML-based counterparts. For CVSS version 3 [56, 57], Chen et al. [28, 29] and Anwar et al. [6] also reported the strong performance of DL-based models (e.g., CNN and graph convolutional neural network [94]). Some other studies did not directly predict severity score from SV descriptions, instead they aggregated the predicted values of the CVSS Exploitability (see section 3) and Impact metrics (see section 4) using the formulas of CVSS version 2 [49, 85, 169, 191], version 3 [49, 85, 138] and WIVSS [169]. We noticed the papers predicting both versions (e.g., CVSS versions 2 vs. 3 or CVSS version 2 vs. WIVSS) usually obtained better performance for version 3 and WIVSS than version 2 [49, 85]. These findings may suggest that the improvements made by experts in version 3 and WIVSS compared to version 2 help make the patterns in severity score clearer and easier for ML models to capture. In addition to predicting severity score, Toloudis et al. [178] examined the correlation between words in descriptions of SVs and the severity values of such SVs, aiming to shed light on words that increase or decrease the severity score of SVs.

**Summary of the Severity theme:** Predicting the severity levels/score is much more prevalent than binary classification of the severity. A majority of the severity outputs are derived from the base score of CVSS versions 2 and 3. In practice, score gives more fine-grained information (fewer SVs per value) for practitioners to rank/prioritize SVs than categorical/binary levels. However, predicting continuous score values is challenging and requires more robust models as it involves higher uncertainty than classifying fixed/discrete levels.
### Table 9. List of the surveyed papers in the Type theme.

| Study                        | Nature of task                           | Data source                              | Data-driven technique                                      |
|------------------------------|-----------------------------------------|------------------------------------------|------------------------------------------------------------|
| **Sub-theme: 1. Common Weakness Enumeration (CWE)**                             |                                         |                                                         |
| Wang et al. 2010 [185]       | Multi-class classification: CWE classes | NVD, CVSS                                | Naïve Bayes                                                |
| Shiui et al. 2015 [166]      |                                         | NVD                                      | SVM                                                       |
| Na et al. 2016 [130]         |                                         | NVD                                      | Naïve Bayes                                                |
| Ruohonen et al. 2018 [153]   |                                         | NVD, CWE, Snyk                           | tf-idf with 1/2/3-grams and cosine similarity              |
| Huang et al. 2019 [78]       |                                         | NVD                                      | MLP, Linear SVM, Naïve Bayes, KNN                         |
| Aota et al. 2020 [7]         |                                         | NVD                                      | Random forest, Linear SVM, Logistic regression, Decision tree, Extremely randomized trees, LGBM |
| Aghaei et al. 2020 [1]       |                                         | NVD                                      | Adaptive fully-connected neural network with one hidden layer |
| Das et al. 2021 [38]         |                                         | NVD, CWE                                 | BERT, Deep Siamese network                                 |
| Zou et al. 2019 [211]        |                                         | NVD & Software Assurance Reference Dataset (SARD) | Three Bi-LSTM models for extracting and combining global and local features from code functions |
| Murtaza et al. 2016 [129]    | Unsupervised learning: sequence mining of SV types (over time) | NVD (CWE & CPE)                           | 2/3/4/5-grams of CWEs                                     |
| Lin et al. 2017 [110]        | Unsupervised learning: association rule mining of CWE-related aspects (prog. language, time of introduction & consequence scope) | CWE                                      | FP-growth association rule mining algorithm               |
| Han et al. 2018 [71]         | Binary/Multi-class classification: CWE relationships (CWE links, link types & CWE consequences) | CWE                                      | Deep knowledge graph embedding of CWE entities            |
| **Sub-theme: 2. Custom Vulnerability Types**                                    |                                         |                                                         |
| Venter et al. 2008 [182]     | Unsupervised learning: clustering       | CVE                                      | Self-organizing map                                        |
| Neuhaus et al. 2010 [133]    | Unsupervised learning: topic modeling   | CVE                                      | Latent Dirichlet Allocation (LDA)                         |
| Mounika et al. 128, 181      |                                         | CVL, Open Web Application Security Project (OWASP) | LDA                                                     |
| Aljedaani et al. 2020 [3]    |                                         | SV reports (Chromium project)            | LDA                                                       |
| Williams et al. 192, 193      | Multi-class classification: manually coded SV types | NVD                                      | Supervised Topical Evolution Model & Diffusion-based storytelling technique |
| Russo et al. 2019 [155]      |                                         | NVD                                      | Bayesian network, J48 tree, Logistic regression, Naïve Bayes, Random forest |
| Yan et al. 2017 [199]        |                                         | Executables of 100 Linux applications     | Decision tree                                             |
| Zhang et al. 2020 [205]      | Multi-class classification: platform-specific vulnerability types | China National Vulnerability Database    | Logistic regression, Linear discriminant analysis, KNN, CART, SVM, bagging/-boosting models |

### 6 Type Prediction

This section reports the work done in the Type theme. Type groups SVs with similar characteristics, e.g., causes, attack patterns and impacts, and thus facilitating the reuse of known prioritization and remediation strategies employed for prior SVs of the same types. Two key prediction outputs are: (i) Common Weakness Enumeration (CWE) and (ii) Custom vulnerability types (see Table 9).
6.1 Common Weakness Enumeration (CWE)

The first sub-theme determines and analyzes the patterns of the SV types provided by Common Weakness and Enumeration (CWE) [127]. CWE is currently the standard for SV types with more than 900 entries. The first group of studies has focused on multi-class classification of these CWEs. Wang et al. [185] were the first to tackle this problem with a Naïve Bayes model using the CVSS metrics (version 2) [55] and product names. Later, Shuai et al. [166] used LDA [17] with a location-aware weighting to extract important features from SV descriptions for building an effective SVM-based CWE classifier. Na et al. [130] also showed that features extracted from SV descriptions can improve the Naïve Bayes model reputed in [185]. Ruohonen et al. [153] investigated an information retrieval method, i.e., term-frequency inverse document frequency (tf-idf) and cosine similarity, to detect the CWE-ID with a description most similar to that of a given SV collected from either NVD or Snyk.13 This method performed well for CWEs without clear patterns/keywords in SV descriptions. Aota et al. [7] utilized the Boruta feature selection algorithm [101] and Random forest to improve the performance of base CWE classification. Base CWEs give more fine-grained information for SV remediation than the categorical CWEs used in [130].

There has been a recent rise in using neural network/DL based models for CWE classification. Huang et al. [78] implemented a deep neural network with tf-idf and information gain for the task and obtained better performance than SVM, Naïve Bayes and KNN. Aghaei et al. [1] improved upon [7] for both categorical (coarse-grained) and base (fine-grained) CWE classification with an adaptive hierarchical neural network to determine sequences of less to more fine-grained CWEs. To capture the hierarchical structure and rare classes of CWEs, Das et al. [38] matched SV and CWE descriptions instead of predicting CWEs directly. They presented a deep Siamese network with a BERT-based [41] shared feature extractor that outperformed many baselines even for rare/unseen CWE classes. Recently, Zou et al. [211] pioneered the multi-class classification of CWE in vulnerable functions curated from Software Assurance Reference Dataset (SARD) [135] and NVD. They achieved high performance (~95% F1-score) with DL (Bi-LSTM) models. The strength of their model came from combining global (semantically related statements) and local (variables/statements affecting function calls) features. Note that this model currently only works for functions in C/C++ and 40 selected classes of CWE.

Another group of studies has considered unsupervised learning methods to extract CWE sequences, patterns and relationships. Sequences of SV types over time were identified by Murtaza et al. [129] using an n-gram model. This model sheds light on both co-occurring and upcoming CWEs (grams), raising awareness of potential cascading attacks. Lin et al. [110] applied an association rule mining algorithm, FP-growth [70], to extract the rules/patterns of various CWEs aspects including types, programming language, time of introduction and consequence scope. For example, buffer overflow (CWE type) usually appears during the implementation phase (time of introduction) in C/C++ (programming language) and affects the availability (consequence scope). Lately, Han et al. [71] developed a deep knowledge graph embedding technique to mine the relationships among CWE types, assisting in finding relevant SV types with similar properties.

6.2 Custom Vulnerability Types

The second sub-theme is about custom vulnerability types other than CWE. Venter et al. [182] used Self-organizing map [96], an unsupervised clustering algorithm, to group SVs with similar descriptions on CVE. To the best of our knowledge, this is one of the earliest studies that automated SV type classification. Topic modeling is another popular unsupervised learning model [3, 128, 133, 181] to categorize SVs without an existing taxonomy. Neuhaus et al. [133] applied LDA [17]

13https://snyk.io/vuln
on SV descriptions to identify 28 prevalent SV types and then analyzed the trends of such types over time. The identified SV topics/types had considerable overlaps (up to 98% precision and 95% recall) with CWEs. Mounika et al. [128, 181] extended [133] to map the LDA topics with the top-10 OWASP [143]. However, the LDA topics/keywords did not agree well (< 40%) with the OWASP descriptions, probably because 10 topics did not cover all the underlying patterns of SV descriptions. Aljedaani et al. [3] again used LDA to identify 10 types of SVs reported in the bug tracking system of Chromium$^{14}$ and found memory-related issues were the most prevalent topics.

Another group of studies has classified manually defined/selected SV types rather than CWE as some SV types are encountered more often in practice and require more attention. Williams et al. [192, 193] applied a supervised topical evolution model [131] to identify the features that best described the 10 pre-defined SV types$^{15}$ prevalent in the wild. These authors then used a diffusion-based storytelling technique [9] to show the evolution of a particular topic of SVs over time; e.g., increasing API-related SVs requires hardening the APIs used in a product. To support user-friendly SV assessment using ever-increasing unstructured SV data, Russo et al. [155] used Bayesian network to predict 10 pre-defined SV types.$^{16}$ Besides predicting manually defined SV types using SV natural language descriptions, Yan et al. [199] used a decision tree to predict 22 SV types prevalent in the executables of Linux applications. The predicted type was then combined with fuzzers’ outputs to predict SV exploitability (see section 3.1). Besides author-defined types, custom SV types also come from specific SV platforms. Zhang et al. [205] designed an ML-based framework to predict the SV types collected from China National Vulnerability Database. Ensemble models (bagging and boosting models) achieved, on average, the highest performance for this task.

**Summary of the Type theme:** Detecting and characterizing coarse-grained and fine-grained CWE-based SV types are the frequent tasks. The large number and hierarchical structure of classes are the main challenges with CWE classification/analysis. To circumvent such issues of CWE, author-selected or platform-specific SV types can be considered.

7 MISCELLANEOUS TASKS

The last theme is Miscellaneous Tasks covering the studies that are representative yet do not fit into the four previous themes. This theme has three main sub-themes/tasks: (i) Vulnerability information retrieval, (ii) Cross-source vulnerability patterns and (iii) Vulnerability fixing effort (see Table 10).

7.1 Vulnerability Information Retrieval

The first and major sub-theme is vulnerability information retrieval that studies data-driven methods to extract different SV-related entities (e.g., affected products/versions) and their relationships from SV data. Note that the current sub-theme extracts assessment information appearing explicitly in SV data (e.g., SV descriptions on NVD) rather than predicting implicit properties as done in prior sub-themes. For instance, CWE-119, i.e., “Improper Restriction of Write Operations within the Bounds of a Memory Buffer”, can be retrieved directly from CVE-2020-28022,$^{17}$ but not from CVE-2021-2122.$^{18}$ The latter case would require techniques in the CWE sub-theme (see section 6.1).

---

$^{14}$https://bugs.chromium.org/p/chromium/issues/list
$^{15}$1. Buffer errors, 2. Cross-site scripting, 3. Path traversal, 4. Permissions and Privileges, 5. Input validation, 6. SQL injection, 7. Information disclosure, 8. Resources Error, 9. Cryptographic issues, 10. Code injection.
$^{16}$1. Authentication bypass or Improper Authorization, 2. Cross-site scripting or HTML injection, 3. Denial of service, 4. Directory Traversal, 5. Local/Remote file include and Arbitrary file upload, 6. Information disclosure and/or Arbitrary file read, 7. Buffer/stack/heap/integer overflow, 8. Remote code execution, 9. SQL injection, 10. Unspecified vulnerability
$^{17}$https://nvd.nist.gov/vuln/detail/CVE-2020-28022
$^{18}$https://nvd.nist.gov/vuln/detail/CVE-2021-21220
### Table 10. List of the surveyed papers in the Miscellaneous Tasks theme.

| Study                      | Nature of task                                                                 | Data source                                                                 | Data-driven technique                                                                 |
|----------------------------|---------------------------------------------------------------------------------|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
|                            | **Sub-theme: 1. Vulnerability Information Retrieval**                            |                                                                             |                                                                                        |
| Weerawardhana et al. 2014  | *Multi-class classification*: Extraction of entities (software name/version, impact, attacker/user actions) from SV descriptions | NVD (210 randomly selected and manually labeled SVs)                          | Stanford Named Entity Recognizer implementing a CRF classifier                           |
| Dong et al. 2019 [44]      | *Multi-class classification*: Vulnerable software names/versions                | CVE Details, NVD, ExploitDB, SecurityFocus, SecurityFocus Forum, SecurityTracker, Openwall | Word-level and character-level Bi-LSTM with attention mechanism                        |
| Gonzalez et al. 2019 [65]  | *Multi-class classification*: Extraction of 19 Vulnerability Description Ontology classes from SV descriptions | NVD                                                                         | Naive Bayes, Decision tree, SVM, Random forest, Majority voting model                   |
| Binyamini et al. 2020 [15] | *Multi-class classification*: Extraction of entities (attack vector/means/technique, privilege, impact, vulnerable platform/version/OS, network protocol/port) from SV descriptions to generate MulVal interaction rules | NVD                                                                         | Bi-LSTM with various feature extractors: word2vec, ELMo, BERT (pre-trained or trained from scratch) |
| Guo et al. 2020 [68]       | *Multi-class classification*: Extraction of entities (SV type, root cause, attack type, attack vector) from SV descriptions | NVD, SecurityFocus                                                          | CNN, Bi-LSTM (with or without attention mechanism)                                      |
| Waareus et al. 2020 [188]  | *Multi-class classification*: Common Product Enumeration (CPE)                   | NVD                                                                         | Word-level and character-level Bi-LSTM                                                  |
| Yitagesu et al. 2021 [201] | *Multi-class classification*: Part-of-speech tagging of SV descriptions         | NVD, CVE, CWE, CAPEC, CPE, Twitter, PTB corpus [116]                          | Bi-LSTM                                                                                 |
| Sun et al. 2021 [175]      | *Multi-class classification*: Extraction of entities (vulnerable product/version/component, type, attack type, root cause, attack vector, impact) from ExploitDB to generate SV descriptions | NVD, ExploitDB                                                               | BERT models                                                                            |
|                            | **Sub-theme: 2. Cross-source Vulnerability Patterns**                            |                                                                             |                                                                                        |
| Horawalavithana et al. 2019 | Regression: Number of software development activities on GitHub after disclosure of SVs | Twitter, Reddit, GitHub                                                     | MLP, LSTM                                                                              |
| Xiao et al. 2019 [197]     | Knowledge-graph reasoning: modeling the relationships among SVs, its types and attack patterns | CVE, CWE, CAPEC (Linux project)                                             | Translation-based knowledge-graph embedding                                              |
|                            | **Sub-theme: 3. Vulnerability Fixing Effort**                                    |                                                                             |                                                                                        |
| Othmane et al. 2017 [141]  | Regression: time (days) to fix SVs                                               | Proprietary SV data collected at the SAP company                             | Linear/Tree-based/Neural network regression                                            |

Most of the retrieval methods in this sub-theme have been formulated under the multi-class classification setting. One of the earliest works was conducted by Weerawardhana et al. [190]. This study extracted software names/versions, impacts and attacker’s/user’s action from SV descriptions on NVD using Stanford Named Entity Recognition (NER) technique, a.k.a. CRF classifier [53]. Later, Dong et al. [44] proposed to use a word/character-level Bi-LSTM to improve the performance of extracting vulnerable software names and versions from SV descriptions available on NVD and other SV databases/advisories (e.g., CVE Details [213], ExploitDB [160], SecurityFocus [81], SecurityTracker [161] and Openwall [147]). Based on the extracted entities, these authors also highlighted the inconsistencies in vulnerable software names and versions across different SV sources. Besides version products/names of SVs, Gonzalez et al. [65] used a majority vote of different ML models (e.g., SVM and Random forest) to extract the 19 entities of Vulnerability Description
Ontology (VDO) [136] from SV descriptions to check the consistency of these descriptions based on the guidelines of VDO. Since 2020, there has been a trend in using DL models (e.g., Bi-LSTM, CNNs or BERT [41]/ELMo [146]) to extract different information from SV descriptions including required elements for generating MulVal [142] attack rules [15] or SV types/root cause, attack type/vector [68], Common Product Enumeration (CPE) [125] for standardizing names of vulnerable vendors/products/versions [188], part-of-speech [201] and relevant entities (e.g., vulnerable products, attack type, root cause) from ExploitDB to generate SV descriptions [175]. BERT models [41], pre-trained on general text (e.g., Wikipedia pages [59] or PTB corpus [116]) and fine-tuned on SV text, have also been increasingly used to address the data scarcity/imbalance for the retrieval tasks.

7.1.1 Cross-source Vulnerability Patterns. The second sub-theme, cross-source vulnerability patterns, finds commonality and/or discovers latent relationships among SV sources to enrich information for SV assessment and prioritization. Horawalavithana et al. [76] found a positive correlation between development activities (e.g., push/pull requests and issues) on GitHub and SV mentions on Reddit19 and Twitter. These authors then used DL models (MLP [73] and LSTM [74]) to predict the appearance and sequence of development activities when SVs were mentioned on the two social media platforms. Xiao et al. [197] applied a translation-based graph embedding method to encode and predict the relationships among different SVs and the respective attack patterns and types. This work [197] was based on DeepWeak of Han et al. [71], but it still belongs to this sub-theme as they provided a multi-dimensional view of SVs using three different sources (NVD [134], CWE [127] and CAPEC [124]). Xiao et al. [197] envisioned that their knowledge graph can be extended to incorporate the source code introducing/fixing SVs.

7.1.2 Vulnerability Fixing Effort. The last sub-theme is vulnerability fixing effort that focuses on estimating SV fixing effort through proxies such as the SV fixing time, usually in days. Othmane and the co-authors were among the first to approach this problem. These authors first conducted a large-scale qualitative study at the SAP company and identified 65 important code-based, process-based and developer-based factors contributing to the SV fixing effort [12]. Later, the same group of authors [141] leveraged the identified factors in their prior qualitative study to develop various regression models such as linear regression, tree-based regression and neural network regression models, to predict time-to-fix SVs using the data collected at SAP. These authors found that code components containing detected SVs are more important for the prediction than SV types.

Summary of the Miscellaneous theme: The key task is to retrieve SV-related entities and characteristics from SV descriptions. Besides information retrieval, other tasks such as linking multi-sources, extracting cross-source patterns or estimating fixing effort are useful to obtain richer SV information for assessment and prioritization, yet still in early stages.

8 ANALYSIS OF DATA-DRIVEN APPROACHES FOR SOFTWARE VULNERABILITY ASSESSMENT AND PRIORITIZATION

We extract and analyze the five key elements for data-driven SV assessment and prioritization: (i) Data sources, (ii) Model features, (iii) Prediction models, (iv) Evaluation techniques and (v) Evaluation metrics. These elements correspond to the four main steps in building data-driven models: data collection (data sources), feature engineering (model features), model training (prediction models) and model evaluation (evaluation techniques/metrics) [69, 156]. These elements are applicable to different tasks as many tasks share the same inputs (e.g., SV descriptions) and/or nature of

19https://reddit.com
Table 11. Five most frequent data sources, features, models, evaluation techniques and evaluation metrics used for the five identified SV assessment and prioritization themes. **Notes:** The top-1 to top-5 values of each criterion are organized from top to bottom. We only included the values that have been used in at least two studies, and thus some cells were left with a dash (–). k-CV stands for k-fold cross-validation.

| Exploitation | Impact | Severity | Type | Misc. Tasks |
|--------------|--------|----------|------|-------------|
| **Criterion: Data Sources** |
| NVD/CVE      | NVD/CVE| NVD/CVE  | NVD/CVE| NVD/CVE     |
| ExploitDB    | Twitter| CVE Details| CWE  | CWE         |
| Symantec     | SecurityFocus| Twitter| –    | ExploitDB, CAPEC |
| Twitter      | Symantec| SecurityFocus| –    | SecurityFocus, CPE |
| SecurityFocus| –      | OSVDB, X-Force| –    | –           |

| **Criterion: Model Feature** |
| BoW          | BoW    | tf-idf   | BoW  | NVD/CVE metadata |
| tf-idf       | tf-idf | BoW      | tf-idf| BERT           |
| NVD/CVE metadata| –      | DL model specific features| n-grams| word2vec |
| Static features| –      | word2vec | BERT | –           |
| Existence of exploits| –      | n-grams, tf-idf, Twitter metadata| –    | –           |

| **Criterion: Prediction Model** |
| Linear SVM classifier| Naïve Bayes| Linear SVM classifier| Latent Dirichlet Allocation| Bi-LSTM with/without attention |
| Random forest        | Random forest| Random forest| Naïve Bayes, Linear SVM, Logistic regression, Bayesian network| Linear regression, MLP regressor, RBF network regressor, Gaussian processes |
| Naïve Bayes          | Linear SVM classifier, Logistic regression| ML for base metrics + Severity formulas| Random forest, Decision tree, MLP, KNN, Supervised Topical Evolution Model| – |
| Logistic regression  | XGBoost    | Word-level CNN | – | – |
| XGBoost              | –         | KNN, Linear SVM regressor| – | – |

| **Criterion: Evaluation Technique** |
| Single k-CV w/o test| Single k-CV w/o test| Single k-CV w/o test| Single random train/test| Multiple random train/val/test |
| Single time-based train/test| Single random train/val/test| Single train/val/test| No mention| Single random train/test |
| Multiple time-based train/test| Single k-CV with test| Single k-CV with test| Multiple random training/test| Single k-CV w/o test |
| Single random train/test| –| Multiple time-based train/set| No qualitative/only qualitative validation| Single random train/val/test |
| Single random train/val/test, Single k-CV with test| –| Single random train/test, Multiple random train/test| Single k-CV w/o test, Multiple time-based train/test| Single time-based train/test |

| **Criterion: Evaluation Metric** |
| F1-score          | F1-score, Accuracy| Accuracy| Precision| Precision |
| Precision, Recall | Precision, Recall| F1-score| Recall| Recall |
| Accuracy          | –                  | Precision, Recall| F1-score| F1-score |
| ROC-AUC           | –                  | Mean absolute error| Accuracy| Root mean squared error |
| Precision-Recall-AUC, Confusion matrix| Accuracy| Mean absolute percentage error| –| Correlation coefficient, Coefficient of determination |
tasks (e.g., classification/regression). To highlight the most common practices that can be reused or compared with for future work, we focus on the top-5 most frequent values per theme in each of the five aforementioned elements (see Table 11). Some themes (e.g., Impact and Miscellaneous) did not have all top-5 values as these themes had fewer papers and the papers used the same values.

8.1 Data sources
Identifying and collecting rich and reliable SV-related data are the first tasks to build data-driven models for automating SV assessment and prioritization tasks. As shown in Table 11, a wide variety of data sources have been considered to accomplish the five identified themes.

Across the five themes, NVD [134] and CVE [126] have been the most prevalently used data sources. CVE provides a unique ID (i.e., CVE-ID) for each reported SV and NVD complements CVE with other information such as SV descriptions, CVSS metrics/score, CWE and external resources. The popularity of NVD/CVE is mainly because they publish expert-verified SV information that can be used to develop prediction models. Firstly, many studies have considered SV descriptions on NVD/CVE as model inputs. Secondly, the SV characteristics on NVD have been heavily used as assessment outputs in all the themes, e.g., CVSS Exploitability metrics for Exploitation, CVSS Impact/Scope metrics for Impact, CVSS severity score/levels for Severity, CWE for Type, CWE/CPE for Miscellaneous tasks. Thirdly, external sources on NVD/CVE have enabled many studies to obtain richer SV information (e.g., exploitation availability/time [29] or vulnerable code/crashes [179, 199]) and extract relationships among multiple SV sources to develop a knowledge graph of SVs (e.g., [71, 197]). Despite all the aforementioned advantages, NVD/CVE still suffer from a poor coverage of relevant external sources (e.g., SV fixing code) [75].

To enrich the SV information on NVD/CVE, many other security advisories and SV databases have been commonly leveraged by the reviewed studies, notably ExploitDB [160], Symantec [20, 21], SecurityFocus [81], CVE Details [213] and OSVDB. Most of these sources disclose PoC (ExploitDB and OSVDB) and/or real-world (Symantec and Security Focus) exploits. However, real-world exploits are much rarer and different compared to PoC ones [83, 157]. Future work should explore more data sources (other than the ones in Table 2) and better methods to retrieve real-world exploits (see section 9.1). Additionally, CVE Details and OSVDB are SV databases like NVD yet with a few key differences. CVE Details explicitly monitors Exploit-DB entries that may be missed on NVD and provides a more user-friendly interface to view/search SVs. OSVDB also reports SVs that do not appear on NVD (without CVE-ID), but this site was discontinued in 2016.

Besides official/expert-verified data sources, we have seen an increasing interest in mining SV information from informal sources that also contain non-expert generated content such as social media (e.g., Twitter) and darkweb. Especially, Twitter has been widely used for predicting exploits as this platform has been shown to contain many SV disclosures even before official databases like NVD [28, 157]. Recently, darkweb forums/sites/markets have also gained traction as SV mentions on these sites have a strong correlation with their exploits in the wild [4, 5]. However, SV-related data on these informal sources are much noisier because they neither follow any pre-defined structure nor have any verification and they are even prone to fake news [157]. Thus, the data integrity of these sources should be checked, potentially by checking the reputation of posters, to avoid inputting unreliable data to prediction models and potentially producing misleading findings.

8.2 Model features
Collected raw data need to be represented by suitable features for training prediction models. There are three key types of feature representation methods in this area: term frequency (e.g., BoW, tf-idf and n-grams), DL learned features (e.g., BERT and word2vec) and source/expert-defined metadata (e.g., CVSS metrics and CPE on NVD or tweet properties on Twitter), as summarized in Table 11.
Regarding the term-frequency based methods, BoW has been the most popular one. Its popularity is probably because it is one of the simplest ways to extract features from natural language descriptions of SVs and directly compatible with popular ML models (e.g., Linear SVM, Logistic regression and Random forest) in section 8.3. Besides using plain term count/frequency, other studies have also considered different weighting mechanisms such as inverse document frequency weighting (tf-idf) or tf-igm [32] inverse gravity moment weighting (tf-igm). Tf-igm has been shown to work better than BoW and tf-idf at classifying severity [31, 99]. However, the applicability and generalizability of tf-igm for other SV assessment and prioritization tasks are still unknown.

Recently, Neural Network (NN) or DL based features such as word2vec [123] and BERT [41] have been increasingly used to improve the performance of predicting CVSS exploitation/impact/severity metrics [64, 72], CWE types [38] and SV information retrieval [68, 188]. Compared to BoW and its variants, NN and DL can extract more efficient and context-aware features from vast SV data [104]. NN/DL techniques rely on distributed representation to encode SV-related words using fixed-length vectors much smaller than a vocabulary size. Moreover, these techniques capture the sequential order and context (nearby words) to enable better SV-related text comprehension (e.g., SV vs. general exploit). Importantly, these NN/DL learned features can be first trained in a non-SV domain with abundant data (e.g., Wikipedia pages [59]) and then transferred/fine-tuned in the SV domain to address limited/imbalanced SV data [200]. The main concern with these sophisticated NN/DL features is their limited interpretability, which is an exciting research area (see section 9.2.2).

The metadata about SVs on data sources can also enrich the information in descriptions or code for SV assessment and prioritization. For example, prediction of exploits and their characteristics have been enhanced using CVSS metrics [5], CPE [2] and SV types [14] on NVD. Additionally, Twitter-related statistics (e.g., number of followers, likes and retweets) have been shown to increase the performance of predicting SV exploitation, impact and severity [29, 157]. Recently, alongside features extracted from vulnerable code, the information about a software development process and involved developers have also been extracted to predict SV fixing effort [141]. Currently, metadata-based and text-based features have been mainly integrated by concatenating their respective feature vectors (e.g., [4, 5, 28, 30]). An alternative yet unexplored way is to build a separate model for each feature type and then combine these models using meta-learning (e.g., model stacking [46]).

8.3 Prediction models

The extracted features enter a wide variety of ML/DL-based prediction models shown in Table 11 to automate various SV assessment and prioritization tasks. Classification techniques have the largest proportion, while regression and unsupervised techniques are less common. The use of these techniques varies among the themes; i.e., Exploitation, Impact and Severity share many similar data-driven approaches, while Type and Miscellaneous themes use diverse methods.

Linear SVM [36] has been the most frequently used classifier, especially in the Exploitation, Impact and Severity themes. This popularity is reasonable as Linear SVM works well with the commonly used features, i.e., BoW and tf-idf, extracted from natural language descriptions of SVs [10], as mentioned in section 8.2. Besides Linear SVM, Random forest, Naïve Bayes and Logistic regression have also been common classification models. In recent years, advanced boosting models (e.g., XGBoost [33] and LGBM [89]), and more lately, DL techniques (e.g., CNN [93], Bi-LSTM with attention [8], graph neural network [94], deep Siamese network [150] and deep multi-task learning [208]) have been increasingly utilized and shown better results than simple ML models like Linear SVM or Logistic regression. Simple baselines should still be considered alongside sophisticated ones as simple methods can perform on par with advanced ones in certain tasks [117].

Besides classification, various prediction models have also been investigated for regression (e.g., predicting exploit time, severity score and fixing time). Linear SVM has again been the most
commonly used regressor as SV descriptions have usually been the regression input. It is worth noting that many studies in the Severity theme did not build regression models to directly obtain the severity score (e.g., [49, 85, 138, 169, 191]). Instead, they used the formulas defined by different frameworks (e.g., CVSS versions 2/3 [55, 56] or WIVSS [171]) to compute the severity score from the base metrics predicted by respective classification models. We argue that more effort should be invested in determining the severity score directly from SV input data as these severity formulas can be subjective [172]. We also observe that there is still limited use of DL models for regression compared to classification tasks.

In addition to supervised (classification/regression) techniques, unsupervised learning has also been considered for extracting underlying patterns of SV data, especially in the Type theme. Notably, Latent Dirichlet Allocation (LDA) [17] is the most commonly used topic model to identify latent topics/types of SVs without relying on a labeled taxonomy. The identified topics have been mapped to the existing SV taxonomies such as CWE [133] and OWASP [128, 181]. It is worth noting that the topics generated by topic models like LDA can also be used as additional features for classification/regression models or building topic-wise models to capture local SV patterns [119].

### 8.4 Evaluation techniques

It is important to evaluate a trained model to ensure the model meets certain requirements (e.g., advancing the state-of-the-art). The evaluation generally needs to be conducted on a different set of data other than the training set to avoid overfitting and objectively estimate model generalizability [73]. The commonly used evaluation techniques are summarized in Table 11.

The reviewed studies have mostly used one or multiple validation and/or test sets\(^{20}\) to evaluate their models, in which each validation/test set has been either randomly or time-based selected. Specifically, k-fold cross-validation is one of the most commonly used techniques in all five themes. The number of folds has usually been 5 or 10, but less standard values like 4 [199] have also been used. However, k-fold cross-validation uses all parts of data at least once for training; thus, there is no hidden test set to evaluate the optimal model with the highest (cross-)validation performance.

To address the lack of hidden test set(s), a common practice in the studied papers has been to split a dataset into single training and test sets, sometimes with an additional validation set for tuning hyperparameters to obtain an optimal model. Recently, data has been increasingly split based on the published time of SVs to better reflect the changing nature of ever-increasing SVs [22, 107]. There have been various ratios for random (e.g., 80:20, 75:25 or 67:33) and time-based (e.g., week/month/year-wise) splits. However, the results reported using single validation/test sets may be unstable (i.e., unreproducible results using different set(s)) [149].

To ensure both the time order and reduce the result randomness, we recommend using multiple splits of training and test sets in combination with time-based validation in each training set. Statistical analyses (e.g., hypothesis testing and effect size) should also be conducted to confirm the reliability of findings with respect to the randomization of models/data in multiple runs [39].

### 8.5 Evaluation metrics

Evaluating different aspects of a model requires respective proper metrics. The popular metrics for evaluating the tasks in each theme are given in Table 11.

Across the five themes, Accuracy, Precision, Recall and F1-score [86] have been the most commonly used metrics. The prevalence of these four metrics aligns with a large number of classification tasks performed in the five themes. However, Accuracy is not a suitable measure for SV assessment

\(^{20}\)Validation set(s) helps optimize/tune a model (finding the best task/data-specific hyperparameters), and test set(s) evaluates the optimized/tuned model. Using only validation set(s) means evaluating a model with default/pre-defined hyperparameters.
A Survey on Data-driven Software Vulnerability Assessment and Prioritization

9 OPEN RESEARCH CHALLENGES AND FUTURE DIRECTIONS OF DATA-DRIVEN SOFTWARE VULNERABILITY ASSESSMENT AND PRIORITIZATION

We discuss three main open challenges with the reviewed studies of data-driven SV assessment and prioritization and then present nine potential directions to address such challenges (see Figure 2).

9.1 SV Data Availability

This section focuses on three key issues with the currently used data sources and potential solutions. First, the current data hardly contain specific developer’s concerns and practices when addressing real-world SVs (section 9.1.1). Second, the data sources still miss many SV-related bugs reported in issue tracking systems, limiting the amount of data for training prediction models (section 9.1.2).

21https://stackoverflow.com/questions/34698161

Fig. 2. List of challenges and future directions for data-driven SV assessment and prioritization.
Third, some tasks/outputs (e.g., real-world exploit prediction) suffer from limited and/or imbalanced labeled data, potentially leading to unreliable performance of fully-supervised models (section 9.1.3).

9.1.1 Utilization of developer Q&A platforms and version control systems. Developer Question & Answer (Q&A) platforms like Stack Overflow and Security StackExchange\(^\text{22}\) contain tens of thousands of posts about challenges and solutions shared by millions of developers when tackling known SVs in real-world scenarios [105]. One of the key insights of Le et al. [105]’s study is that the top SV types that developers usually struggle with are not always the same as those reported on SV databases (CWE [127] or OWASP [143]). Thus, future work should also consider real-world development-related issues discussed on developer Q&A platforms for automatically assessing and prioritizing SVs. For example, the fixing effort of SVs may depend on the technical difficulty of implementing the respective mitigation strategies in a language or system of interest.

Version control systems like GitHub\(^\text{23}\) provide details about how developers addressed past SVs in real-world projects. Shrestha et al. [165] found developers sometimes discuss/disclose SV-related information on GitHub discussions even before the studied social media such as Twitter or Reddit. These findings show the potential of using GitHub discussions to complement the current sources for earlier SV assessment and prioritization. Moreover, Walden [183] demonstrated the impact of a major SV (i.e., Heartbleed) on the characteristics (e.g., code complexity/style, contributors and development practices) of a single project (i.e., OpenSSL). Based on Walden’s findings, future work can study whether the impact of an SV would be similar or different in multiple affected projects. Such investigation would give insights into the possibility of leveraging data from large projects to perform SV assessment and prioritization in smaller projects with the same or similar SVs.

9.1.2 Integration of SV data on issue tracking systems. Issue/bug tracking systems like JIRA, Bugzilla or GitHub issues\(^\text{24}\) have been reporting numerous security-related bugs, many of which are SVs, but they have been underexplored for data-driven SV assessment and prioritization. Besides providing SV descriptions like CVE/NVD, these bug reports also contain other artifacts such as steps to reproduce, stack traces and test cases that give extra information about SVs [210]. However, it is not trivial to obtain and integrate these SV-related bug reports with the ones on SV databases.

One way to retrieve SVs on issue tracking systems is to use security bug reports [3]. Much research work has been put into developing effective models to automatically retrieve security bug reports (e.g., [62, 145, 195]). Among these studies, Wu et al. [195] manually verified and cleaned the security bug reports to provide a clean dataset for automated security bug report identification. However, more of such manual effort is still required to obtain up-to-date data because the original security bug reports in [195] were actually a part of the dataset collected back in 2014 [140].

It is worth noting that not all security bug reports are related to SVs such as issues/improvements in implementing security features.\(^\text{25}\) Thus, future studies need to filter out these cases before using security bug reports for SV assessment and prioritization. We also emphasize that some SV-related bug reports are overlapping with the ones on NVD (e.g., the SV report AMBARI-14780\(^\text{26}\) on JIRA refers to CVE-2016-0731 on CVE/NVD). Such overlaps would require data cleaning during the integration of reports on issue tracking systems and SV databases to avoid data duplication (e.g., similar SV descriptions) when developing SV assessment and prioritization models.

\(^{22}\)Stack Overflow: https://stackoverflow.com & Security StackExchange: https://security.stackexchange.com
\(^{23}\)https://github.com
\(^{24}\)JIRA: https://www.atlassian.com/software/jira, Bugzilla: https://www.bugzilla.org/, GitHub issues: https://docs.github.com/en/issues
\(^{25}\)The security bug report AMBARI-1373 on JIRA (https://issues.apache.org/jira/browse/AMBARI-1373) was about improving the front-end of AMBARI Web by displaying the current logged in user.
\(^{26}\)https://issues.apache.org/jira/browse/AMBARI-14780
9.1.3 Relaxation of fully-supervised learning. Supervised learning models of many tasks in the five themes (see section 8.3) require fully labeled data, but the data of some tasks are quite limited. To address the data-hungriness of these fully-supervised learning models, future studies can approach the SV assessment and prioritization tasks with low-shot learning and/or semi-supervised learning.

Low-shot learning a.k.a. few-shot learning is designed to perform supervised learning using only a few examples per class, significantly reducing the labeling effort [187]. So far, only one study utilized low-shot learning with a deep Siamese network [38] (i.e., a shared feature model with similarity learning) to effectively predict SV types (CWE) and even generalize to unseen classes (i.e., zero-shot learning). There are still many opportunities for investigating different few-shot learning techniques for other SV assessment and prioritization tasks. Note that the shared features in few-shot learning can also be enhanced with pretrained models (e.g., BERT [41]) on another domain/task/project with more labeled data than the current task/project in the SV domain.

Semi-supervised learning enables training models with limited labeled data yet a large amount of unlabeled data [180], potentially leveraging hidden/unlabeled SVs in the wild. Recently, we have seen an increasing interest in using different techniques of this learning paradigm in the SV domain such as collecting SV patches using multi-view co-training [159] or retrieving SV discussions on developer Q&A sites using positive-unlabeled learning [106]. However, it is still little known about the effectiveness of semi-supervised learning for SV assessment and prioritization.

9.2 Real-world Application and Evaluation

The experimental performance of some SV assessment and prioritization models is promising, but the real-world applicability of such models is still questionable. First, these models may not be useful in practice due to delayed inputs and coarse-grained outputs (section 9.2.1). Second, many models are black-box, limiting the understanding of the model predictions (section 9.2.2). Third, some models are evaluated in over-optimistic conditions far from real-world scenarios (section 9.2.3).

9.2.1 More timely and fine-grained prediction. Although SV descriptions have been commonly used as model inputs (see section 8.1), these descriptions are usually published long after SVs introduced/discovered in codebases [118]. One potential solution to this issue is to perform assessment and prioritization of SVs in code commits. Code commits contain changes made by developers to fix a bug/SV, implement a new feature or refactor code, and new SVs may be introduced in such changes [18]. Commit-level prediction would allow just-in-time SV assessment and prioritization as soon as SVs are introduced, reducing the waiting time for SV information to be verified and published on security advisories/databases. It should be noted that report-level prediction is still important for assessing and prioritizing third-party libraries/software, especially the ones without available code (commits), and/or SVs missed by commit-level prediction.

CVSS [54] has been most frequently used for assessing the exploitability, impact and severity levels/score of SVs (see sections 3, 4, 5), but there are increasing concerns that CVSS outputs are still generic. Specifically, Spring et al. [172] argued that CVSS tends to provide one-size-fits-all assessment metrics regardless of the context of SVs; i.e., the same SVs in different domains/environments are assigned the same metric values. For instance, banking systems may consider the confidentiality and integrity of databases more important than the availability of web/app interfaces. In the future, alongside CVSS, prediction models should also incorporate the domain/business knowledge to customize the assessment of SVs to a system of interest (e.g., the impact of SVs on critical component(s) and/or the readiness of developers/solutions for mitigating such SVs in the current system). Future case studies with practitioners are also desired to correlate the quantitative performance of models and their usability/usefulness in real-world systems (e.g., reducing more critical SVs yet using fewer resources).
9.2.2 Enhancing model interpretability. Model interpretability is important to increase the transparency of the predictions made by a model, allowing practitioners to adjust the model/data to meet certain requirements [207]. Unfortunately, very few reviewed papers (e.g., [72, 178]) explicitly discussed important features and/or explained why/when their models worked/failed for a task.

SV assessment and prioritization can draw inspiration from the related SV detection area where the interpretability of (DL-based) prediction models has been actively explored mainly by using (i) specific model architectures/parameters or (ii) external interpretation models/techniques [207]. In the first approach, prior studies successfully used the feature activation maps in a CNN model [154] or leveraged attention-based neural network [45] to highlight and visualize the important code tokens that contribute to SVs. The second approach uses separate interpretation models on top of trained SV detectors. The interpretation models are either domain/model-agnostic [189], domain-agnostic yet specific to a model type (graph neural network [108]) or SV-specific [212]. The aforementioned approaches produce local/sample-wise interpretation, which can be aggregated to obtain global/task-wise interpretation. The global interpretation is similar to the feature importance of traditional ML models [27] such as the weights of linear models (e.g., Logistic regression) or the (im)purity of nodes split by each feature in tree-based models (e.g., Random forest). However, it is still unclear about the applicability/effectiveness of these approaches for interpreting ML/DL-based SV assessment and prioritization models, requiring further investigations.

9.2.3 Realistic evaluation settings. Most of the reviewed studies have evaluated their prediction models without capturing many factors encountered during the deployment of such models to production. Specifically, the models used in practice would require to handle new data and be robust against adversarial data from informal sources such as social media or darkweb.

There are concerns with both predicting and integrating new SV data. Regarding the prediction, Out-of-Vocabulary words in new data need to be properly accommodated to avoid performance degradation of prediction models [107]. Regarding the new data integration, online/incremental training on new data can be considered instead of batch training on the whole dataset to reduce computational cost [23]. The time-based splits should be used rather than random splits for evaluating online training to avoid leaking unseen (future) patterns to the model training (see section 8.4).

Regarding the model robustness, only three reviewed studies considered adversarial attacks as part of their evaluation [5, 157, 196]. However, a recent survey shows the prevalence of adversarial attacks targeted models in cybersecurity [151]. Thus, there is certainly a need for more evaluation of adversarial robustness for SV assessment and prioritization models, especially DL-based ones.

9.3 Data-driven SV Assessment and Prioritization of Data-driven Systems

Compared to other systems, reporting/analyzing SVs of data-driven/Artificial Intelligence (AI)-based systems is still in its infancy [100]. Data-driven systems (e.g., smart recommender systems, chatbots, robots, and autonomous cars) are an emerging breed of systems whose cores are powered by AI technologies, e.g., ML and DL models built on data, rather than human-defined instructions as in traditional systems. We discuss three key challenges of SV assessment and prioritization of data-driven systems compared to traditional systems and suggest potential solutions. Firstly, the current SV assessment frameworks need customizations to better reflect the nature of SVs in data-driven systems (section 9.3.1). Secondly, there is a lack of SVs collected from real-world data-driven systems, limiting the potential of data-driven SV assessment and prioritization (section 9.3.2). Thirdly, the current models require redesign, especially in the SV representation, to capture unique characteristics and artifacts of data-driven systems (section 9.3.3).

9.3.1 Designing a compatible SV assessment framework. CVSS [54] is currently the most popular SV assessment framework for traditional systems, but its compatibility with data-driven systems
still requires more investigation. The current CVSS documentation lacks instructions on how to assign metrics/score for SVs in data-driven systems. For example, it is unclear how to assign static CVSS metrics to systems with automatically updated data-driven models [34] because adversarial examples for exploitation would likely change after the models are updated. Such ambiguities should be clarified/resolved in future CVSS versions as data-driven systems become more prevalent.

The types of SVs in ML/DL models in data-driven systems are also mostly different from the ones provided by CWE [127]. The difference is mainly because these new SVs do not only emerge from configurations/code as in traditional systems, but also from training data and/or trained models [151]. Thus, we recommend that a new category of these SVs should be studied and potentially incorporated into CWE, similar to the newly added category for architectural SVs.27

9.3.2 Collection of SVs in data-driven systems. To the best of our knowledge, there has been no existing large-scale dataset of SVs in ML/DL models deployed in real-world data-driven systems. Very few of such SVs have been reported in the wild, one of which is CVE-2019-20634.28 More of these SVs are required to help develop sufficiently effective SV assessment and prioritization models. One potential way to build such a dataset is to first match the (pre-trained) ML/DL models proposed in the literature or released on model repositories (e.g., Tensorflow Hub29) with the ones used in real-world systems either on version control systems or in mobile apps [80]. The matched models would then be tested against various adversarial attacks to identify corresponding SVs. Notably, significant effort is still required to define/label assessment outputs of these SVs (see section 9.3.1).

9.3.3 Representation of SVs in data-driven systems. Existing SV assessment and prioritization models for traditional systems have not considered unique data/model-related characteristics/features of data-driven systems [184]. Specifically, data-driven systems also encompass information about data (e.g., format, type, size and distribution) and ML/DL model(s) (e.g., configurations, parameters and performance). It is worth noting that SVs of ML/DL models in data-driven systems can also come from the frameworks used to develop such models (e.g., Tensorflow or Keras)30. However, developers of data-driven systems may not be aware of the (security) issues in the used ML/DL frameworks [112]. Thus, besides currently used features, future work should also consider the information about underlying data/models and used ML/DL development frameworks to improve the SV representation for building models to assess and prioritize SVs in data-driven systems.

10 CONCLUSIONS

Assessment and prioritization are crucial phases to optimize resource utilization in addressing SVs at scale. The two phases have witnessed radical transformations following the increasing availability of SV data from multiple sources and advances in data-driven techniques. We presented a taxonomy to summarize the five main directions of the research work so far in this area. We identified and analyzed the key practices to develop data-driven models in the reviewed studies. We also highlighted the open challenges and suggested respective solutions to advance the field.

We envision the field will largely continue to improve the effectiveness of the presented tasks by leveraging more enriched data sources and sophisticated data-driven models, especially DL-based ones. Besides the improved performance, we also see many open opportunities/concerns in under-explored aspects of developing such advanced models. Overall, a deeper understanding of practitioners’ concerns and real-world usage scenarios is the key to bridging the current gap between model development in academia and model deployment in production.

27https://cwe.mitre.org/data/definitions/1008.html
28https://nvd.nist.gov/vuln/detail/CVE-2019-20634
29https://www.tensorflow.org/hub
30Tensorflow: https://github.com/tensorflow/tensorflow & Keras: https://github.com/keras-team/keras
ACKNOWLEDGMENT

The work was supported by the Cyber Security Research Centre Limited whose activities are partially funded by the Australian Government’s Cooperative Research Centre Programme.

REFERENCES

[1] Ehsan Aghaei, Waseem Shadid, and Ehab Al-Shaer. 2020. ThreatZoom: CVE2CWE using hierarchical neural network. arXiv preprint arXiv:2009.11501 (2020).

[2] M Ugur Aksu, Kemal Bicakci, M Hadi Dilek, A Murat Ozbayoglu, and E Islam Tatli. 2018. Automated generation of attack graphs using nvd. In Proceedings of the 8th Conference on Data and Application Security and Privacy. 135–142.

[3] Waji Aljedaani, Yasir Javed, and Mamduh Allan. 2020. LDA categorization of security bug reports in Chromium projects. In Proceedings of the 2020 European Symposium on Software Engineering. 154–161.

[4] Mohammed Almukaynizi, Eric Nune, Krishna Dhariaiya, Manoj Senguttuvan, Jana Shakkarian, and Paulo Shkarian. 2017. Proactive identification of exploits in the wild through vulnerability mentions online. In 2017 International Conference on Cyber Conflict (CyCon US). IEEE, 82–88.

[5] Mohammed Almukaynizi, Eric Nune, Krishna Dhariaiya, Manoj Senguttuvan, Jana Shakkarian, and Paulo Shkarian. 2019. Patch before exploited: An approach to identify targeted software vulnerabilities. In AI in Cybersecurity. Springer, 81–113.

[6] Afsah Anwar, Ahmed Abusnaina, Songqing Chen, Frank Li, and David Mohaisen. 2020. Cleaning the NVD: Comprehensive quality assessment, improvements, and analyses. arXiv preprint arXiv:2006.15074 (2020).

[7] Masaki Aota, Hideaki Kanehara, Masaki Kubo, Noboru Murata, Bo Sun, and Takeshi Takahashi. 2020. Automation of Vulnerability Classification from its Description using Machine Learning. In 2020 IEEE Symposium on Computers and Communications (ISC). IEEE, 1–7.

[8] Dzmitry Bahdanau, Kynghyun Cho, and Yoshua Bengio. 2014. Neutral machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014).

[9] Roberto Camacho Barranco, Arnold P Boedihardjo, and M Shahriar Hosain. 2019. Analyzing evolving stories in news articles. International Journal of Data Science and Analytics 8, 3 (2019), 241–256.

[10] Atrey Basu, Christine Walters, and M Shepherd. 2003. Support vector machines for text categorization. In 36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the. IEEE, 7–pp.

[11] Michael Z Bell. 1985. Why expert systems fail. Journal of the Operational Research Society 36, 7 (1985), 613–619.

[12] Lotfi ben Othmane, Golriz Chehrazi, Eric Bodden, Petar Tsalovski, Achim D Brucker, and Philip Miseldine. 2015. Factors impacting the effort required to fix security vulnerabilities. In International Conference on Information Security. Springer, 102–119.

[13] Massimo Bernaschi, Emanuele Gabrielli, and Luigi V Mancini. 2002. REMUS: A security-enhanced operating system. ACM Transactions on Information and System Security (TISSEC) 5, 1 (2002), 36–61.

[14] Navneet Bhatt, Adarsh Anand, and VSS Yadavalli. 2021. Exploitability prediction of software vulnerabilities. Quality and Reliability Engineering International 37, 2 (2021), 648–663.

[15] Hodaya Binyamini, Ron Bitton, Masaki Inokuchi, Tomohiko Yagyu, Yuval Elovici, and Asaf Shabtai. 2020. An automated, end-to-end framework for modeling attacks from vulnerability descriptions. arXiv preprint arXiv:2008.04377 (2020).

[16] David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In Proceedings of the 20th International Conference on Neural Information Processing Systems. 121–128.

[17] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. Journal of machine learning research 3, Jan (2003), 993–1022.

[18] Amiangshu Bosu, Jeffrey C Carver, Munawar Hafiz, Patrick Hilley, and Derek Janni. 2014. Identifying the characteristics of vulnerable code changes: An empirical study. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. 257–268.

[19] Mehran Bozorgi, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. 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. 105–114.

[20] Broadcom. [n. d.]. Symantec attack signatures. Retrieved May, 2021 from https://bit.ly/symantec_att_sign

[21] Broadcom. [n. d.]. Symantec threat explorer. Retrieved May, 2021 from https://bit.ly/symantec_threats

[22] Benjamin L Bullough, Anna K Yanchenko, Christopher L Smith, and Joseph R Zipkin. 2017. Predicting exploitation of disclosed software vulnerabilities using open-source data. In Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics. 45–53.

[23] George G Cabral, Leandro L Minku, Emad Shihab, and Suhaiq Mujahid. 2019. Class imbalance evolution and verification latency in just-in-time software defect prediction. In 2019 IEEE/ACM 41st International Conference on
A Survey on Data-driven Software Vulnerability Assessment and Prioritization

29

CERT. [n. d.]. Basic fuzzing framework. Retrieved May, 2021 from https://vuls.cert.org/confluence/display/tools/CERT+BFF+-+Basic+Fuzzing+Framework

Sang Kil Cha. [n. d.]. Ofuzz. Retrieved May, 2021 from https://github.com/sangkic/ofuzz

Sang Kil Cha, Thanassis Avgerinos, Alexandre Rebert, and David Brumley. 2012. Unleashing mayhem on binary code. In 2012 IEEE Symposium on Security and Privacy. IEEE, 380–394.

Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. Computers & Electrical Engineering 40, 1 (2014), 16–28.

Haipeng Chen, Jing Liu, Rui Liu, Noseong Park, and VS Subrahmanian. 2019. VASE: A Twitter-based vulnerability analysis and score engine. In 2019 IEEE International Conference on Data Mining (ICDM). IEEE, 976–981.

Haipeng Chen, Jing Liu, Rui Liu, Noseong Park, and VS Subrahmanian. 2019. VEST: A system for vulnerability exploit scoring & timing. In IJCAI. 6503–6505.

Haipeng Chen, Rui Liu, Noseong Park, and VS Subrahmanian. 2019. Using twitter to predict when vulnerabilities will be exploited. In Proceedings of the 25th International Conference on Knowledge Discovery & Data Mining. 3143–3152.

Jinfu Chen, Patrick Kwaku Kudjo, Solomon Mensah, Selasie Aformaley Brown, and George Akorfu. 2020. An automatic software vulnerability classification framework using term frequency-inverse gravity moment and feature selection. Journal of Systems and Software 167 (2020), 110616.

Kewen Chen, Zuping Zhang, Jun Long, and Hao Zhang. 2016. Turning from TF-IDF to TF-IGM for term weighting in text classification. Expert Systems with Applications 66 (2016), 245–260.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd International Conference on Knowledge Discovery and Data Mining. 785–794.

Yang Chen, Andrew E Santos, Ang Ming Yi, Abhishek Sharma, Asankhaya Sharma, and David Lo. 2020. A machine learning approach for vulnerability curation. In Proceedings of the 17th International Conference on Mining Software Repositories. 32–42.

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

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. Machine learning 20, 3 (1995), 273–297.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive aggressive algorithms. (2006).

Siddhartha Shankar Das, Edoardo Serra, Mahantesh Halappanavar, Alex Pothen, and Ehab Al-Shaer. 2021. V2W-BERT: A framework for effective hierarchical multiclass classification of software vulnerabilities. arXiv preprint arXiv:2102.11498 (2021).

Francisco Gomes de Oliveira Neto, Richard Torkar, Robert Feldt, Lucas Gren, Carlo A Furia, and Ziwei Huang. 2019. Evolution of statistical analysis in empirical software engineering research: Current state and steps forward. Journal of Systems and Software 156 (2019), 246–267.

Daniel Alves de Sousa, Elaine Ribeiro de Faria, and Rodrigo Sanches Miani. 2020. Evaluating the performance of Twitter-based exploit detectors. arXiv preprint arXiv:2011.03113 (2020).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

Nesara Dissanayake, Asangi Jayatilaka, Mansooroh Zahedi, and M Ali Babar. 2020. Software security patch management–A systematic literature review of challenges, approaches, tools and practices. arXiv preprint arXiv:2012.00544 (2020).

Brendan Dolan-Gavitt, Patrick Hulin, Engin Kirda, Tim Leek, Andrea Mambretti, Wil Robertson, Frederick Ulrich, and Ryan Whelan. 2016. Lava: Large-scale automated vulnerability addition. In 2016 IEEE Symposium on Security and Privacy (SP). IEEE, 110–121.

Ying Dong, Wenbo Guo, Yueqi Chen, Xinyu Xing, Yuqing Zhang, and Gang Wang. 2019. Towards the detection of inconsistencies in public security vulnerability reports. In 28th {USENIX} Security Symposium. 869–885.

Xu Duan, Jingzheng Wu, Shouling Ji, Zhiqing Rui, Tianyue Luo, Mutian Yang, and Yanjun Wu. 2019. VulSniper: Focus your attention to shoot fine-grained vulnerabilities. In IJCAI. 4665–4671.

Saso Dzeroski and Bernard Zenko. 2002. Is combining classifiers better than selecting the best one?. In ICML, Vol. 2002. Citeseer, 123e30.

MICHEL Edkrantz. 2015. Predicting exploit likelihood for cyber vulnerabilities with machine learning. Master’s thesis.

Michel Edkrantz, Staffan Truvé, and Alan Said. 2015. Predicting vulnerability exploits in the wild. In 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing. IEEE, 513–514.

Clément Elbaz, Louis Rilling, and Christine Morin. 2020. Fighting N-day vulnerabilities with automated CVSS vector prediction at disclosure. In the 15th International Conference on Availability, Reliability and Security. 1–10.

ESET. [n. d.]. ESET security advisories. Retrieved May, 2021 from https://bit.ly/eset_virus
[107] Triet HM Le, Bushra Sabir, and Muhammad Ali Babar. 2019. Automated software vulnerability assessment with concept drift. In Proceedings of the 16th International Conference on Mining Software Repositories (MSR). IEEE, 371–382.

[108] Yi Li, Shaohua Wang, and Tien N Nguyen. 2021. Vulnerability detection with fine-grained interpretations. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.

[109] Guanjun Lin, Sheng Wen, Qing-Long Han, Jun Zhang, and Yang Xiang. 2020. Software vulnerability detection using deep neural networks: a survey. Proc. IEEE 108, 10 (2020), 1825–1848.

[110] Zhechao Lin, Xiang Li, and Xiaohui Kuang. 2017. Machine learning in vulnerability databases. In 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Vol. 1. IEEE, 108–113.

[111] Hailong Liu and Bo Li. 2019. Automated classification of attacker privileges based on deep neural network. In International Conference on Smart Computing and Communication. Springer, 180–189.

[112] Jaikun Liu, Qiao Huang, Xin Xia, Emad Shihab, David Lo, and Shanping Li. 2020. Is using deep learning frameworks free? characterizing technical debt in deep learning frameworks. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society. 1–10.

[113] Kai Liu, Yun Zhou, Qingyong Wang, and Xianqiang Zhu. 2019. Vulnerability severity prediction with deep neural network. In 2019 5th International Conference on Big Data and Information Analytics (BigDIA). IEEE, 114–119.

[114] Ruchika Malhotra et al. 2021. Severity Prediction of Software Vulnerabilities Using Textual Data. In Proceedings of International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications. Springer, 453–464.

[115] Pratyusa K Manadhata and Jeannette M Wing. 2010. An attack surface metric. IEEE Transactions on Software Engineering 37, 3 (2010), 371–386.

[116] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. (1993).

[117] A. Mazzuora-Rozo, A. Mojica-Hanke, M. Linares-Vasquez, and G. Bavota. 2021. Shallow or deep? An empirical study on detecting vulnerabilities using deep learning. In Proceedings of the 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC). 276–287.

[118] Andrew Meneely, Harshvardhan Srinivasan, Ayemi Musa, Alberto Rodriguez Tejeda, Matthew Mokary, and Brian Spates. 2013. When a patch goes bad: Exploring the properties of vulnerability-contributing commits. In 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE, 65–74.

[119] Tim Menzies, Suvodeep Majumder, Nikhila Balaji, Katie Brey, and Wei Fu. 2018. 500+ times faster than deep learning(a case study exploring faster methods for text mining stackoverflow). In 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR). IEEE, 554–563.

[120] Trend Micro. [n. d.]. Trend Micro security advisories. Retrieved May, 2021 from https://www.trendmicro.com/vinfo/us/threat-encyclopedia

[121] Trend Micro. [n. d.]. ZeroDay Initiative security advisories. Retrieved May, 2021 from http://www.zerodayinitiative.com

[122] Microsoft. [n. d.]. Microsoft security advisories. Retrieved May, 2021 from https://bit.ly/ms_sec_advisories

[123] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546 (2013).

[124] Mitre. [n. d.]. Common Attack Pattern Enumeration and Classification. Retrieved May, 2021 from https://capec.mitre.org

[125] Mitre. [n. d.]. Common Platform Enumeration. Retrieved May, 2021 from https://cpe.mitre.org

[126] MITRE. [n. d.]. Common Weakness Enumeration. Retrieved May, 2021 from https://cwe.mitre.org

[127] MITRE. [n. d.]. Common Vulnerabilities and Exposures. Retrieved May, 2021 from https://cve.mitre.org/

[128] MITRE. [n. d.]. Common Attack Pattern Enumeration and Classification. Retrieved May, 2021 from https://capec.mitre.org

[129] MITRE. [n. d.]. Common Platform Enumeration. Retrieved May, 2021 from https://cpe.mitre.org

[130] Shunta Nakagawa, Tatsuya Nagai, Hideaki Kanehara, Keisuke Furumoto, Makoto Takita, Yoshiaki Shiraiishi, Takeshi Takahashi, Masami Mohri, Yasuhiro Takano, and Masakatu Morii. 2019. Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description. IEICE Transactions on Information and Systems 102, 9 (2019), 1679–1682.

[131] Stephan Neuhaus and Thomas Zimmermann. 2010. Security trend analysis with cve topic models. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.

[132] Shunta Nakagawa, Tatsuya Nagai, Hideaki Kanehara, Keisuke Furumoto, Makoto Takita, Yoshiaki Shiraiishi, Takeshi Takahashi, Masami Mohri, Yasuhiro Takano, and Masakatu Morii. 2019. Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description. IEICE Transactions on Information and Systems 102, 9 (2019), 1679–1682.

[133] Shunta Nakagawa, Tatsuya Nagai, Hideaki Kanehara, Keisuke Furumoto, Makoto Takita, Yoshiaki Shiraiishi, Takeshi Takahashi, Masami Mohri, Yasuhiro Takano, and Masakatu Morii. 2019. Character-level convolutional neural network for predicting severity of software vulnerability from vulnerability description. IEICE Transactions on Information and Systems 102, 9 (2019), 1679–1682.
Zhiyuan Wan, Xin Xia, David Lo, and Gail C Murphy. 2019. How does machine learning change software development?

Dimitrios Toloudis, Georgios Spanos, and Lefteris Angelis. 2016. Associating the Severity of Vulnerabilities with SEMANTICs: A brief survey. Soft Computing: Theories and Applications

Jonathan Spring, Eric Hatleback, Allen Householder, Art Manion, and Deana Shick. 2021. Time to change the CVSS? Network Security

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security

Prasha Shrestha, Arun Sathanur, Suraj Maharjan, Emily Saldanha, Dustin Arendt, and Svitlana Volkova. 2020. Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit. Plos one 15, 3 (2020), e0230250.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security

Prasha Shrestha, Arun Sathanur, Suraj Maharjan, Emily Saldanha, Dustin Arendt, and Svitlana Volkova. 2020. Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit. Plos one 15, 3 (2020), e0230250.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security

Prasha Shrestha, Arun Sathanur, Suraj Maharjan, Emily Saldanha, Dustin Arendt, and Svitlana Volkova. 2020. Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit. Plos one 15, 3 (2020), e0230250.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security

Prasha Shrestha, Arun Sathanur, Suraj Maharjan, Emily Saldanha, Dustin Arendt, and Svitlana Volkova. 2020. Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit. Plos one 15, 3 (2020), e0230250.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security

Prasha Shrestha, Arun Sathanur, Suraj Maharjan, Emily Saldanha, Dustin Arendt, and Svitlana Volkova. 2020. Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit. Plos one 15, 3 (2020), e0230250.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security

Prasha Shrestha, Arun Sathanur, Suraj Maharjan, Emily Saldanha, Dustin Arendt, and Svitlana Volkova. 2020. Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of GitHub, Twitter and Reddit. Plos one 15, 3 (2020), e0230250.

Bo Shuai, Haifeng Li, Mengjun Li, Quan Zhang, and Chaojing Tang. 2013. Automatic classification for vulnerability based on machine learning. In 2013 IEEE International Conference on Information and Automation (ICIA). IEEE, 312–318.

Shashank Kumar Singh and Amrita Chaturvedi. 2020. Applying deep learning for discovery and analysis of software vulnerabilities: A brief survey. Soft Computing: Theories and Applications (2020), 649–658.

Vincent Smyth. 2017. Software vulnerability management: how intelligence helps reduce the risk. Network Security
A Survey on Data-driven Software Vulnerability Assessment and Prioritization

[190] Sachini Weerawardhana, Subhojeet Mukherjee, Indrajit Ray, and Adele Howe. 2014. Automated extraction of vulnerability information for home computer security. In International Symposium on Foundations and Practice of Security. Springer, 356–366.

[191] Tao Wen, Yuqing Zhang, Ying Dong, and Gang Yang. 2015. A novel automatic severity vulnerability assessment framework. Journal of Communications 10, 5 (2015), 320–329.

[192] Mark A Williams, Roberto Camacho Barranco, Sheikh Motahar Naim, Sumi Dey, M Shahriar Hossain, and Monika Akbar. 2020. A vulnerability analysis and prediction framework. Computers & Security 92 (2020), 101751.

[193] Mark A Williams, Sumi Dey, Roberto Camacho Barranco, Sheikh Motahar Naim, M Shahriar Hossain, and Monika Akbar. 2018. Analyzing evolving trends of vulnerabilities in national vulnerability database. In 2018 IEEE International Conference on Big Data (Big Data). IEEE, 3011–3020.

[194] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. Chemometrics and intelligent laboratory systems 2, 1-3 (1987), 37–52.

[195] Xiaoxue Wu, Wei Zheng, Xin Xia, and David Lo. 2021. Data quality matters: A case study on data label correctness for security bug report prediction. IEEE Transactions on Software Engineering (2021).

[196] Chaowei Xiao, Armin Sarabi, Yang Liu, Bo Li, Mingyan Liu, and Tudor Dumitras. 2018. From patching delays to infection symptoms: Using risk profiles for an early discovery of vulnerabilities exploited in the wild. In 27th {USENIX} Security Symposium ({USENIX} Security 18), 903–918.

[197] Hongbo Xiao, Zhenchang Xing, Xiaohong Li, and Hao Guo. 2019. Embedding and predicting software security entity relationships: A knowledge graph based approach. In International Conference on Neural Information Processing. Springer, 50–63.

[198] Yasuhiro Yamamoto, Daisuke Miyamoto, and Masaya Nakayama. 2015. Text-mining approach for estimating vulnerability score. In 2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS). IEEE, 67–73.

[199] Guanhua Yan, Junchen Lu, Zhan Shu, and Yunus Kucuk. 2017. Exploitmeter: Combining fuzzing with machine learning for automated evaluation of software exploitability. In 2017 IEEE Symposium on Privacy-Aware Computing (PAC). IEEE, 164–175.

[200] Jiao Yin, MingJian Tang, Jinli Cao, and Hua Wang. 2020. Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. Knowledge-Based Systems 210 (2020), 106529.

[201] Sofonias Yitagesu, Xiaowang Zhang, Zhiyong Feng, Xiaohong Li, and Zhenchang Xing. 2021. Automatic part-of-speech tagging for security vulnerability descriptions. In 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). IEEE, 29–40.

[202] Awd A Younis and Yashwant K Malaiya. 2014. Using software structure to predict vulnerability exploitation potential. In 2014 IEEE Eighth International Conference on Software Security and Reliability-Companion. IEEE, 13–18.

[203] Peng Zeng, Guanjun Lin, Lei Pan, Yonghang Tai, and Jun Zhang. 2020. Software vulnerability analysis and discovery using deep learning techniques: A survey. IEEE Access (2020).

[204] Li Zhang and Vrizlynn LL Thing. 2018. Assisting vulnerability detection by prioritizing crashes with incremental learning. In TENCON 2018-2018 IEEE Region 10 Conference. IEEE, 2080–2085.

[205] Xiong Zhang, Haoran Xie, Hao Yang, Hongkai Shao, and Minghao Zhu. 2020. A general framework to understand vulnerabilities in information systems. IEEE Access 8 (2020), 121858–121873.

[206] Yi Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. arXiv preprint arXiv:1509.01626 (2015).

[207] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. 2020. A survey on neural network interpretability. arXiv preprint arXiv:2012.14261 (2020).

[208] Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering (2021).

[209] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision. 19–27.

[210] Thomas Zimmermann, Rahul Premraj, Nicolas Bettenburg, Sascha Just, Adrian Schroter, and Cathrin Weiss. 2010. What makes a good bug report? IEEE Transactions on Software Engineering 36, 5 (2010), 618–643.

[211] Deqing Zou, Sujuan Wang, Shouhuai Xu, Zhen Li, and Hai Jin. 2019. µVulDeePecker: A deep learning-based system for multiclass vulnerability detection. IEEE Transactions on Dependable and Secure Computing (2019).

[212] Deqing Zou, Yawei Zhu, Shouhuai Xu, Zhen Li, Hai Jin, and Hengkai Ye. 2021. Interpreting deep learning-based vulnerability detector predictions based on heuristic searching. ACM Transactions on Software Engineering and Methodology (TOSEM) 30, 2 (2021), 1–31.

[213] Serkan Özkan. [n. d.]. CVE Details. Retrieved May, 2021 from https://www.cvedetails.com