Early Exploitability Prediction of Just-Disclosed Software Vulnerabilities

Anonymous Author(s)

***Abstract*—Software vulnerabilities are security threats in source code that may enable malicious users to cause harm or loss. With the rate of newly discovered and publicly disclosed vulnerabilities increasing over time, the need for rapid and reliable techniques to estimate the risk of possible exploitation is increasing. Researchers have been experimenting with machine learning to predict the risk of exploitation of a disclosed vulnerability. Despite the good performance obtained, the currently available solutions leverage information that is not yet available when a new CVE is published in the first place. These predictions can be made only after in-depth analyses, i.e., when the CVSS, CWE, or CPE data become accessible. In this paper, we aim at experimenting with *early* exploitability prediction models that exclusively rely on the data immediately available in just-disclosed vulnerabilities, i.e., the brief description and the online discussions referenced in the CVE published before the disclosure date. We assess the effectiveness of 81 machine learning models, trained using different learning and data balancing algorithms, plus the effect of three information retrieval weighting schemas to build the dataset. The results show both the strengths and limitations of early exploitability prediction, calling for a more comprehensive data collection process and for novel systems that can assist developers in managing the vulnerabilities affecting their applications by leveraging early predictions.**

***Index Terms*—Empirical; Exploitability Prediction; Software Vulnerabilities; Machine Learning.**

I. Introduction

Software vulnerabilities represent serious threats to software dependability that allow malicious users to attack its confidentiality, integrity, or availability [36], [57]. Unlike traditional bugs, vulnerabilities require specific methods and techniques to be detected [7], [53] and removed [19] promptly to avoid undesired consequences [31]. However, just like bugs [21], not all vulnerabilities are the same, as their exploitation may require different expertise and may have variable consequences. Since the number of newly discovered vulnerabilities increases rapidly [35], both vendors (i.e., owners of vulnerable products) and clients are interested in obtaining reliable feedback on the risk that a vulnerability may be exploited. This is useful for various tasks, including prioritizing security verification [34], [67] and identifying suitable preventive actions to reduce the risk of an attack [30], such as the replacement of vulnerable constructs or APIs [43], [50].

Researchers have been envisioning novel methods to estimate the severity of newly discovered vulnerabilities. The most basic mechanisms are driven by the CVSS (Common Vulnerability Scoring System) *"Base"* score[1] assigned: a vulnerability receiving a high CVSS *"Base"* score is deemed more severe than others and, to a certain extent, with higher chances to be exploited earlier. Unfortunately, these assumptions do not always hold, as a higher severity score does not necessarily imply a more likely exploitability, and neither do lower scores denote less risk of being exploited [1]. This is, for instance, the case of the infamous HEARTBLEED bug (CVE-2014-0160), whose CVSS 2.0 *"Base"* score was 5.0 (medium severity), way lower than other vulnerabilities never exploited in the wild [31]. For this reason, the use of the CVSS *"Base"* is not recommended to estimate the probability of being attacked.

An alternative approach is concerned with adopting machine [15], [18], [70] and deep [40] learning to predict whether a new vulnerability will be exploited, either labeling it as *likely exploitable* or estimating a probability [46] using a large number of predictors from different data sources. In this respect, researchers have been using many sources of information connected to a vulnerability, ranging from its brief description contained in the CVE (Common Vulnerabilities and Exposures) record—i.e., a data structure containing all the information linked to a disclosed vulnerability—to online mentions from social media or the dark web [70]. The vast majority of the proposed models rely on the content of CVEs that were available when the datasets were built, hence also including the data that became available days or weeks after the official disclosure of a new vulnerability—for instance, CVE-2020-0583 received the CVSS *"Base"* score only 10 days after the disclosure. Indeed, as soon as a new CVE is added to the official database, it is only made of a short description and at least one public reference, as required by CVE [33]. Thus, all the CVSS scores, the appropriate CWE (Common Weakness Enumeration)[2] and CPE (Common Platform Enumeration)[3] cannot be used *before* the in-depth analysis made by security experts. During this period, the existing prediction models cannot use these data, leaving developers without any estimate of the risk associated with a new vulnerability. We recognize the need for an *early* exploitability analysis of software vulnerabilities to provide developers and security experts with preliminary information that could be used to assess the exploitability of a newly-disclosed vulnerability and take appropriate preventive actions to limit the possible harm of a vulnerability. However, to the best of our knowledge, no work has been done in this respect.

In this paper, we empirically evaluate the effectiveness of *early* exploitability prediction modeling that exclusively

[1]CVSS website: https://www.first.org/cvss

[2]COMMON WEAKNESS ENUMERATION: https://cwe.mitre.org

[3]COMMON PLATFORM ENUMERATION: https://nvd.nist.gov/products/cpe

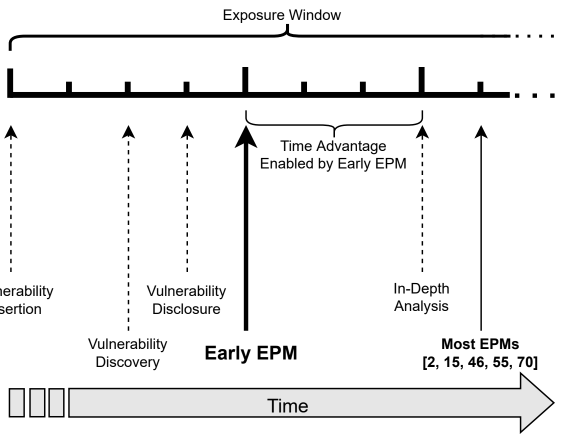

Fig. 1: The applicability moment of our exploitability prediction model on the general vulnerability life cycle, also compared with the applicability of other solutions in literature.

uses the data available in the CVE record of a *just-disclosed* software vulnerability, i.e., the short description and the referenced *online discussions*—e.g., mailing lists and security advisories—published *before* the disclosure date. We compare 81 different configurations of models trained using different machine learning and data balancing algorithms and based on features extracted from natural language text. We also assess the effect of applying different information retrieval (IR) weighting schemas for assigning the values to each textual feature. In this work, we only target publicly-disclosed vulnerabilities to predict whether they will be exploited or not (classification problem). Thus, we do not provide solutions for zero-day exploits, as their prediction is more challenging due to a lack of data before the exploitation [14]. Our contributions can be summarized as follows:

1) An empirical investigation of how different machine learning settings behave when predicting the exploitability of just-disclosed vulnerabilities using three different information retrieval (IR) weighting schemas to build the dataset;
2) A detailed data collection procedure on how to aggregate online discussion data pointed by the external references in the CVE records;
3) An online appendix [4] containing all the data and scripts we employed and the steps to replicate the data collection and the machine learning pipeline.

II. Background and Related Work

A. Software Vulnerabilities Life Cycle

MITRE defines a software vulnerability as a flaw in a software component caused by a weakness in the source code that malicious users can exploit to cause damage to the confidentiality, integrity, or availability of the impacted components.[4] According to this definition, a vulnerability is an identifiable instance of a defect affecting the source code and is characterized by its life cycle inside the flawed system. This life cycle starts with the introduction of the vulnerability in the system, meaning that there is a certain point in the lifetime of a software product in which the vulnerability is—either involuntarily or voluntarily—introduced in the code [44]. The vulnerability is potentially exploitable as soon as the flawed code is released, starting its *exposure window* [36]. After its insertion, the vulnerability can be identified in several ways: internally through manual code inspection or running static analysis tools [7], [77], or from external bug reports [22], [71] or online discussions on public and independent channels (e.g., Security Focus).[5] Once the vulnerability is discovered, the recommended action should be to warn a larger audience about the security issue affecting the software. To this aim, the vendor may request the allocation of a CVE record to have a unique identifier and a standardized descriptor of that specific flaw, which third-party security experts must approve. This certification step ensures that the reported vulnerability is valid and comes from a trustworthy source, which is why an external reference describing the issue is mandatory. The time ranging from the discovery to the public disclosure varies from vendor to vendor, depending on the policy adopted [6], [22]. After the official disclosure by CVE, another external party may provide deeper analyses on the nature of the vulnerability, for example, by identifying the appropriate weakness type (CWE), determining the affected vulnerable configurations (CPE), and estimating its exploitability and potential impact. The famed framework CVSS—defined and managed by the Forum of Incident Response and Security Teams (FIRST)—is commonly adopted to measure this latter aspect. The CVSS standard defines three metrics groups (*"Base"*, *"Temporal"* and *"Environmental"*) that capture different aspects of a vulnerability, each containing a set of ordinal-scale metrics. The most relevant ones belong to the *"Base"* group, which is meant to stay stable after the initial assessment—in contrast with the *"Temporal"* and *"Environmental"* ones that are subject to continuous updates. In particular, the *"Base"* metrics values can also be aggregated together to form a continuous value ranging from 0 to 10, known as the *"Base" score*, that summarizes the overall severity of the vulnerability. This score is computed according to a conversion table by remapping each value to a pre-defined number. Despite its popularity, CVSS measurement has not been exempt from criticism. One of its main weak points is its subjectiveness caused by a lack of clearly defined criteria for assigning the correct value to each metric. Indeed, the scoring procedure is driven by a set of self-asked questions about the characteristics of a vulnerability. In this respect, different analysts might interpret these criteria differently based on their knowledge and experience, causing the final base score to vary among raters. Such a scenario might pose a serious threat to vulnerability management since most strategies rely on the CVSS severity scores to establish the order according to which the security issues must be

[4]CVE Glossary: https://www.cve.org/ResourcesSupport/Glossary

[5]Security Focus website: http://online.securityfocus.com/bid

handled [38]. Moreover, in a non-negligible number of cases, the CVSS measurement is not available until some days after disclosing a CVE, which can be too late in certain unfortunate cases [33]. These motivations laid the foundations of the works that employed machine learning to predict the CVSS vector—or part of it—using the initial information available at disclosure time [33], [40], [49]. Despite this, it has been widely recognized that the CVSS base score is not a proxy for the exploitability risk of a vulnerability; therefore, it cannot be used as a recommendation mechanism to guide how vulnerabilities must be managed [1], [42].

B. Exploitability Prediction Modeling

Several works have considered alternative ways to assess the risk of newly-disclosed vulnerabilities in the last decade, leveraging many data types extracted from different sources. Bozorgi et al. [15] have been among the first to use machine learning to predict the exploitability of known vulnerabilities based on a large variety of metadata extracted from the Open Sourced Vulnerability Database (OSVDB) and the CVE LIST. They trained an SVM-based binary classifier to predict whether a given vulnerability is likely to be exploited in the (near) future, using a dataset of over 10 thousand CVE records disclosed between 1991 and 2007. Their model used a wide spectrum of predictors, ranging from numerical features—e.g., the CVSS metric values, the disclosure date—to the binary occurrence (presence/absence) of the tokens extracted from all the text fields using the bag-of-words (BoW) method—e.g., the CVE description, the product name, the list of external references. They labeled as exploitable (positive class) those vulnerabilities that indicated their exploitation status in their metadata, leaving the rest as unexploitable (negative class). They observed high accuracy ($\approx 90\%$) in the offline setting (splitting the entire dataset into training and test sets randomly), also considering the problem imbalance. In addition, they compared the SVM score (i.e., the fitted model without the decision function) with the CVSS *exploitability* score of each vulnerability in the dataset, observing that the latter is not correlated to the classifier scores and highlighting the poor significance of this metric value. This work inspired many other empirical studies that investigated the performance of different learning algorithms—e.g., LOGISTIC REGRESSION and RANDOM FOREST [13]—and data sources to build the ground truth—such as EXPLOIT DATABASE or SYMANTEC ATTACK SIGNATURES [32]—both leveraging the metadata found in the CVE records. Lyu et al. [55] experimented with more sophisticated natural language processing (NLP) techniques and a character-level Convolutional Neural Network (CNN) to predict the exploitability using only the textual description associated with each CVSS value. Although most works treated the problem as a classification task, Jacobs et al. [46] presented the Exploitability Prediction Scoring System (EPSS) that estimates—using a LOGISTIC REGRESSION model—the probability that a given disclosed vulnerability will be exploited in the following 12 months. Instead of relying on traditional feature extraction techniques (e.g., Bag-of-Words), they collected the number of occurrences of a curated list of tags extracted using the RAKE tool [69]. Moreover, they navigated the external references reported in the CVE record and scraped the text from the HTML pages to further expand the amount of data available in their dataset. The use of textual data was further investigated in other studies, experimenting with alternative ways to extract features and assign them values. Most of these studies relied on traditional information retrieval (IR) weighting schemas, such as *word counting* or *term frequency – inverse document frequency* (TF-IDF) [2], [32], [66], [70], while others exploited NLP techniques, such as WORD2VEC models, to provide a compact representation of the words in a document corpus [40], [55]. Sabottke et al. [70] mined text explicitly mentioning CVEs from TWITTER to expand the pool of predictors. Besides the traditional word counting, they also computed additional metrics from the tweet statistics, such as the number of retweets and replies. Similarly, Almukaynizi et al. [2] used the mentions in the dark web assigning to each extracted token their TF-IDF weight. Apparently, only Jacobs et al. [45] contemplated the text contained in the URLs in CVE records to widen the feature set but did not provide any detail on how they mined the HTML pages. Only Sabottke et al. [70] and Almukaynizi et al. [2] were careful to select only those data published before the date on which a CVE was exploited in the wild—following the recommendation described by Reinthal et al. [66], who claimed that any realistic exploitability prediction model should not leverage the data arriving after the exploitation.

This is just one of the limitations that Bullough et al. [18] identified in many previous works on the matter; in this respect, the authors presented challenges for a realistic machine learning-based exploitability prediction model. Previous studies have legitimately considered the effect of the imbalance ratio, but the re-sampling algorithms were inappropriately applied to the entire dataset, affecting the test sets as well [61]. Similar to the limitations found by Reinthal et al. [66], many works had used all the data found in the CVE records when they were mined, including all the subsequent updates they received even after the exploitation. This approach is not representative of what would happen in reality. Indeed, a realistic prediction should be executable as soon as a new CVE is published, without waiting for the availability of additional metadata like the CWE, the CVSS score, or the CPE, that are known to be added only some days or weeks after the disclosure, as observed by Elbaz et al. [33].

Our work. We analyze six (plus three baselines) learning algorithms leveraging only the description and the online discussion data that were published *before* the disclosure of a new vulnerability. These textual data undergo the traditional information retrieval pre-processing steps to extract a large set of features weighted using three different schemas, i.e., *word counting*, *term frequency*, and *term frequency – inverse document frequency*.

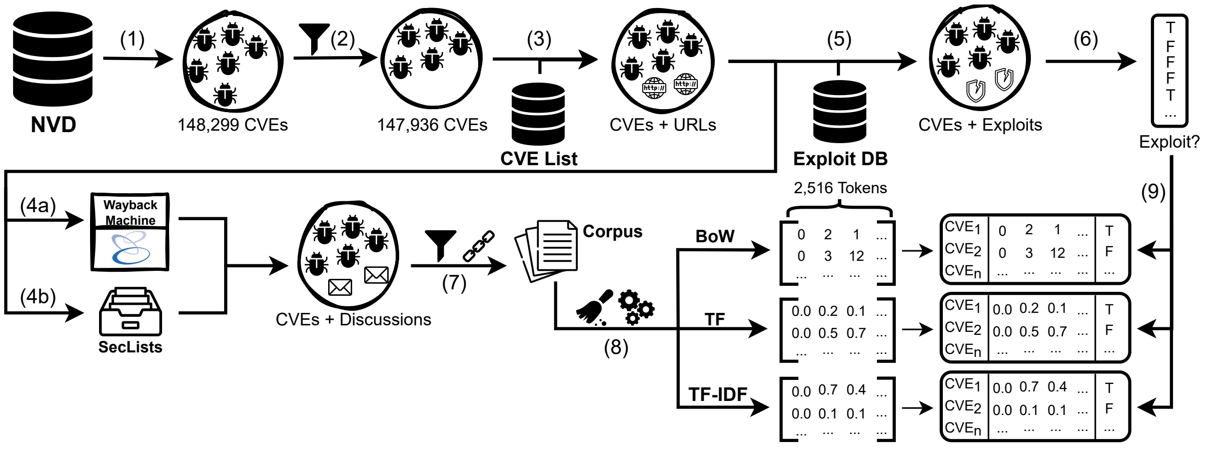

Fig. 2: Overview of the data collection process adopted.

III. Empirical Study Design

In the first part of this section (III-A), we formulate the goal of our study according to the Goal-Question-Metric (GQM) template [9]. In section III-B, we describe the steps that we followed to get the data used in our empirical investigation. Section III-C explains how we set up, ran, and evaluated the machine learning classifiers to find the best possible exploitability prediction model built using different configurations and answer our research questions.

A. Study Goal and Research Questions

The *goal* of this empirical study was to investigate the performance of machine learning-based classifiers to predict the exploitability of a just-disclosed vulnerability, with the *purpose* of providing early feedback on the exploitability of new vulnerabilities in a realistic scenario. The *perspective* is of both practitioners and researchers. The former are interested in obtaining as much information as possible to (i) have an initial assessment supporting the CVSS measurement and (ii) understand when and how the vulnerabilities afflicting their applications must be handled. The latter are interested in comprehending the predictive capabilities of textual features—obtained from unstructured natural language text—for exploitability classifiers and assessing the effectiveness of learning and data balancing algorithms for this task.

To the best of our knowledge, the current research in exploitability prediction has always included all the data available at the time of building the dataset—for training and validating the prediction models—from the CVE databases. This scenario caused the models to rely on security analysts' subjective measures—e.g., the CVSS scores—at a non-negligible distance from the CVE disclosure date [33]. In other words, we hypothesize that any *realistic* and *useful* prediction model should only leverage those pieces of information available at the *disclosure time*, i.e., immediately after the disclosure of a vulnerability. The first part of our empirical investigation focused on the feasibility of employing machine learning to classify potentially-exploitable known vulnerabilities using exclusively the data that are already available just after the official disclosure on CVE databases. Thus, we asked:

RQ$_1$. *How well does an exploitability prediction model perform when exclusively using the data available at the disclosure time of a new vulnerability?*

Moreover, we wanted to assess the differences in classification performance between several learning algorithms and dummy baseline models, such as the random, optimistic (i.e., always says "non-exploitable"), and pessimistic (i.e., always says "exploitable") classifiers [39], [62]. In addition, we also took into account the effect of different data balancing algorithms to improve the quality of the learning by modifying the instances in the training set [61]. Therefore, we asked:

RQ$_2$. *Which is the best combination of learning and data balancing algorithms to train an exploitability prediction model that exclusively uses the data available at the disclosure time of a new vulnerability?*

The ordinary mechanism to raise awareness about recently-discovered security issues is to open a free commentary about them on public mailing lists or any other discussion channel [6], [22]. Such data sources contain valuable information that may provide additional insight into the seriousness of the vulnerability, for example, by reporting a crash stack trace [72], reproduction steps [23], or even code snippets showing Proof of Concepts (PoCs) [74]. Nevertheless, extracting these pieces of information is not a trivial task, owing to (i) the lack of form of structured text and (ii) the limited number of automated mining tools, as those available are meant to work on traditional bug reports and for specific programming languages [11]. The only way to process such unstructured text is to rely on black-box IR techniques to extract features from a corpus made of documents written in natural language [12]. The second part of our research dealt with the impact of different weighting schemas, i.e., simple *word counting* (known as Bag-of-Words, BoW), *term frequency*

(TF), and *term frequency – inverse document frequency* (TF-IDF), when used to assign the values to the textual feature vector extracted from free text [8].

RQ$_3$. *Which is the best IR weighting schema for assigning values to the textual feature vector extracted from unstructured text written in natural language?*

B. Data Collection

The *context* of this empirical study was made of publicly-disclosed vulnerabilities accompanied by external references to online discussions, such as public mailing lists and security advisories. To this aim, we adopted a systematic data collection procedure to link all the existing known vulnerabilities to public websites where they had been likely discussed for the first time before the official disclosure date. Then, we mined a large set of public scripts and PoCs available online and linked them to the vulnerabilities they exploit to carry out (pseudo-)realistic attacks. Figure 2 describes all the steps (from 1 to 9) we took to collect the data needed to answer our research questions. The scripts to replicate the entire data collection phase are available in the online appendix [4].

Mining Known Vulnerability Data. We relied on the National Vulnerability Database (NVD),[6] a comprehensive catalog of disclosed vulnerabilities reported in the form of CVE records. In particular, it expands the details provided by the upstream CVE LIST managed by MITRE,[7] analyzing their severity through CVSS vectors, labeling the known affect software versions by using CPE, etc. NVD has been the basis of many empirical studies on software vulnerabilities [44], [47], [54], [59], so it is considered a reliable source of high-quality information. Our study did not target any specific platform or programming language, so we collected all existing vulnerabilities available in NVD at the time of the study. Thus, we downloaded the full dump of NVD curated by the CVE SEARCH project[8] on November 03, 2021, made of 148,299 entries (Step 1 in Figure 2). We pre-processed this raw dataset to ensure that the quality of the data contained in the dataset was suitable for our study. In particular, we did not consider those entries that were (1) malformed (e.g., the identifier did not point to a really-existing CVE record), (2) rejected (i.e., the CVE identifier was allocated but never approved at the final stage), or (3) lacking any critical information (i.e., the description or the publication date). We could recognize those cases by inspecting the content of the dump: malformed CVE identifiers did not adhere to the pattern `CVE-XXXX-YYYY`; rejected CVEs had a clear statement of their rejection in the description; missing information could be spotted by looking for `null` values in the data fields. In the end, the three filters led to the removal of just 180, 97, and 86 entries, respectively (Step 2 in Figure 2), resulting in 147,936 valid CVEs.

Mining Online Discussion Data. Any CVE record is equipped with a continuously updated list of external reference URLs pointing to web pages concerning that specific vulnerability, e.g., online discussion, official patches, bug reports. In this respect, our goal was to define an exploitability prediction model that leverages only those available references at the disclosure time, as we believe they could be the reason behind the CVE number request. However, the references in the NVD dump do not report the date they were added to the CVE records (e.g., since the creation or at a later stage). We observed that the CVE LIST website maintains the references differently: each is labeled with a special keyword indicating its type (e.g., vendor advisory, mailing list) and origin (i.e., the website it points to).[9] We collected 713,311 references among all the 147,936 CVE records (Step 3 in Figure 2). The most recurring keyword was *BID* (SECURITY FOCUS Bugtraq ID), which refers to security advisories published on the SECURITY FOCUS website (43% of CVEs had at least one reference to this category). Such a website has long been considered a reliable source to report security bugs—each uniquely identified with a *BID*—and tracks existing solutions and working exploits [35]. Moreover, its plain HTML structure allowed to easily recover all the data using a simple HTML parser with the application of filters to exclude those references published after the vulnerability disclosure date. For this reason, we opted to ignore the references listed in the NVD dump, traversed the *BID*-labeled URLs in CVE LIST, and parsed their content using the BEAUTIFULSOUP library for PYTHON.[10] The only available option to recover the data was parsing the HTML page, as SECURITY FOCUS exposes no public API. Note that since February 2020, SECURITY FOCUS has stopped publishing further *BID* advisories; hence the URLs currently stored in the CVE records are inaccessible. We circumvented this limitation by exploiting the WAYBACK MACHINE service provided by the INTERNET ARCHIVE library [3], which offers free access to many digital resources that were once available on the web. In our case, we queried the API exposed by the service that returns an active URL—stored in its archives—having the same HTML contents as the input URL (Step 4a in Figure 2).[11]

We also observed that *BID* reports are commonly related to a discussion on a public mailing list known as BUGTRAQ, one of the most popular discussion boards where participants have been conducting discussions on newly-discovered vulnerabilities since 1993. All the discussions are held in natural language (commonly English) without following any specific text structure. The only consistency among the discussions is the header, which contains the original publication date. Consequently, we considered *BUGTRAQ* references in addition to those labeled with *BID*. BUGTRAQ has encountered a similar fate to SECURITY FOCUS, as it was shut down in 2020; yet, we still considered it alongside SECURITY FOCUS as it was referenced

[6]NVD website: https://nvd.nist.gov
[7]CVE LIST website: https://cve.mitre.org
[8]CVE SEARCH dump: https://www.cve-search.org/dataset
[9]CVE References: https://cve.mitre.org/data/refs/index.html
[10]BEAUTIFULSOUP website: https://beautiful-soup-4.readthedocs.io
[11]WAYBACK MACHINE API: https://archive.org/help/wayback_api.php

by a non-negligible part of the CVEs we selected (14% CVEs had at least one *BUGTRAQ* reference). All the discussions held in the past are now archived by third-party websites, such as SECLISTS,[12] from which we downloaded all the discussions pointed by the CVEs records in our context. To recover the missing mailing list discussion, we exploited the format of the *BUGTRAQ* identifiers, made of eight digits representing the publication day according to the format `YYYYMMDD`, plus a short text summarizing the content of the discussion. Both the year and the month allowed us to reach a page on SECLISTS containing the list of *BUGTRAQ*s of that period, from which we retrieved the entry that had the highest similarity—using the Gestalt Pattern Matching [65]—with the short text in the *BUGTRAQ* identifier. Then, we mined the content of the matched discussions leveraging BEAUTIFULSOUP (Step 4b in Figure 2). To summarize, 70,516 out of 147,936 CVEs had at least one *BID* or *BUGTRAQ* type reference, linked to 65,982 *BID* references and 26,390 *BUGTRAQ* references. We considered these numbers sufficiently high for our goals.

Mining Exploit Data. The CVE references labeled as *EXPLOIT-DB* point to EXPLOIT DATABASE (EDB in short),[13] which is the most comprehensive collection of public exploits and PoCs that explicitly target known vulnerabilities. Navigating these URLs could establish the links between the vulnerabilities and their exploits. However, we observed that only a minimal set of CVE records had an explicit link to EDB (i.e., 7.9%), likely owing to an improper curation of the CVE records—and not necessarily to a real lack of an exploit. Hence, similarly to what Bhatt et al. [13] did, we rebuilt the links crossing the opposite way, connecting all the exploits in EDB to the affected CVEs using the metadata contained in the exploit entry. To this aim, we downloaded the complete list of exploits at the date of November 03, 2021, from the official GITHUB repository[14] to obtain the list of valid exploit identifiers, queried the EDB website, and parsed the HTML pages—still using BEAUTIFULSOUP—of each exploit to collect the target CVEs, if made explicit. Note that a single exploit or PoC may target more than one, often related, vulnerability; similarly, more than one exploit may affect a single vulnerability. In the end, a total of 23,998 exploits were collected, linked to 22,853 different CVEs, corresponding to 15.45% of the total (Step 5 in Figure 2).

C. Experimental Setup

Datasets Preparation. The datasets for training and validating the machine learning models are all made of 147,936 instances, representing the vulnerabilities. We set the values of the *dependent variable* based on the existence of at least one public exploit or PoC in EXPLOIT DATABASE. Doing so generated 15.45% of vulnerabilities labeled with `true`—i.e., "exploitable"— while the remaining 84.55% were labeled with `false`, allowing the binary classification (Step 6 in Figure 2).

[12]SECLISTS website: https://seclists.org
[13]EDB website: https://www.exploit-db.com
[14]EDB repository: https://github.com/offensive-security/exploitdb

Starting from the online discussion data collected in the previous steps, we treated the text data associated with each vulnerability as a *document* (for a total of 147,936 documents) so that the entire list of CVEs would form a *corpus* of documents. Specifically, for each vulnerability, we concatenated (1) the brief description reported in its CVE record, (2) the associated *BID* advisories mined from SECURITY FOCUS, and (3) the *BUGTRAQ* discussions concerning it (Step 7 in Figure 2). We discarded the text of those *BID* and *BUGTRAQ* references published after the CVE disclosure date or whose format did not allow the recovery of the date. This happened for 5,102 out of 65,982 *BID*s and for 4,463 out of 26,390 *BUGTRAQ*s. While these two criteria caused some instances not to have any *BID* or *BUGTRAQ* reference, we still did not discard them entirely as the sole CVE short description might contain enough information to predict their exploitability, as observed in similar works [40], [55].

Afterward, each document underwent a set of preprocessing steps to remove irrelevant pieces of information and facilitate the extraction of textual features. In particular, we first employed a set of regular expressions to detect and remove data that could negatively affect the feature extraction, such as websites, URLs, e-mail addresses, PGP signatures and messages, hex numbers, and words containing repetitions of the same letters in a row for at least three times [12], [40], [52]. Secondly, we applied the lower case reduction, removed non-alphabetic characters (punctuation and Unicode symbols), and split the remaining content into tokens using the whitespace as a separator. Lastly, we removed any English stop word [27], [73], applied the suffix stripping using Porter's stemmer [63], and dropped those tokens appearing in only a tiny amount of documents, i.e., below 0.1% [78]. In the end, we obtained a total of 2,516 terms from the entire corpus, which constituted the set of *independent variables*—i.e., features—of our models. To summarize, each vulnerability was represented as a compact vector made of 2,516 components, making them suitable for a machine learning task.

The set of all the vector representations extracted from the corpus forms a *document-term* matrix, where the rows represent the documents while the columns the terms. The exact values assigned to the matrix elements depend on the weighting schema applied. To answer **RQ**$_3$, we adopted three traditional IR weighting schemas, i.e., (i) the simple *word counting* (known as *Bag-of-Words*, BoW) that assigns to the ij-th element the number of times the j-th term appears in the i-th document [41], [80]; (ii) the *term frequency* (TF) that assigns to the ij-th element the number of times the j-th term appears in the i-th document divided by the total number of terms in the i-th document [75]; (iii) the *term frequency – inverse document frequency* (TF-IDF) representation that multiplies all the TFs by the logarithm of the IDF value, i.e., giving more importance to terms appearing many times in a document (high TF) and rarely in other documents, implying a high discriminating power (high *idf*) [75]. We computed these values using the implementations provided by the SCIKIT-

LEARN library.[15] The values coming from the three schemas were used to build three different document-term matrices (Step 8 in Figure 2), used, in turn, to build three different datasets, sharing the same dependent variable and subject to the same machine learning pipeline (Step 9 in Figure 2).

Setting the Machine Learning Pipeline. Once collecting the datasets, we setup the machine learners to predict the exploitability under the three different weighting schemas. We experimented with six widely-adopted learning algorithms in binary classification, i.e., (i) LOGISTIC REGRESSION (LR) [58], (ii) NAÏVE BAYES (NB) [68], K-NEAREST NEIGHBORS (KNN) [81], (iv) SUPPORT VECTOR MACHINE (SVM) [25], (v) DECISION TREE (DT) [17], and (vi) RANDOM FOREST (RF) [16]. We also included three additional classifiers used as baselines, i.e., OPTIMISTIC CLASSIFIER (OPT), PESSIMISTIC CLASSIFIER (PES), and RANDOM CLASSIFIER (RND). All the models were trained and validated using a 10-fold cross-validation strategy [76], in which the dataset is divided into 10 random subsets of approximately equal size (folds). Within each of the 10 iterations, nine folds were used to train the models, while the last one was used to validate the fitted models, guaranteed to change in each iteration. Before the actual training, we applied some actions to increase the models' learning capabilities. We considered the imbalance ratio in the training dataset (as the number of instances labeled as `false` was much higher than the `true` instances) experimenting with SMOTE oversampling [24], NEARMISS undersampling (version 3) [79], or no re-sampling technique at all. Then, we run a hyper-parameters optimization using RANDOM SEARCH algorithm [10] to find the best combination of hyper-parameters maximizing the F-measure [64], which will be used when the model is tested. Overall, we made 81 configurations (three re-balancing settings, nine learning algorithms, and three different datasets), each trained and tested for 10 times, for a total of 810 runs.

Models Evaluation. When the models were tested against the test fold at each iteration, we collected their results in the form of *confusion matrices*, reporting the predictions in terms of True/False Positives and True/False Negatives. Afterward, we computed the traditional performance metrics for the binary classification task, i.e., precision, recall, and F-measure [64]. Moreover, we also included the Matthews's Correlation Coefficient (MCC) [56], which takes into account all the four quadrants of the confusion matrices, and the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) [26], which gives an overview of the classification performance at different decision thresholds. We computed these metrics during all cross-validation iterations and for all the 81 combinations, obtaining a set of distributions per performance metric. The complete results are available in the online appendix [4].

During the model evaluation and analysis phase, we ran statistical tests to assess the significance of the results. Namely, we used the Friedman test [37] followed by the Nemenyi post-hoc test [60] to determine which of the 81 configurations showed statistically significant differences for a given evaluation metric, setting the significance level α to 0.05. Such an approach is meant for multiple comparisons and does not assume any particular data distribution—as our samples were not always normally distributed [29]. The online appendix [4] includes the raw output of the statistical analyses, accompanied by graphical visualizations of the resulting *p-values*, displayed in the form of heat maps.

[15]Textual features extraction with SCIKIT-LEARN: https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction

IV. ANALYSIS AND DISCUSSION OF THE RESULTS

In this section, we present the results obtained in the experiments, to assess whether online discussion data are sufficient to build good-enough *early* exploitability prediction models. For the sake of conciseness, in this paper, we show the performances only in terms of F-measure and AUC-ROC metrics, but we provide the complete results in the online appendix [4]. We choose these two metrics as the main performance indicators as they capture two different points of view on the models' capabilities. The F-measure represents an aggregation of precision and recall, which are the crucial metrics for evaluating classifiers [8]. The trade-off between these two values is particularly tricky in the context of exploitability prediction of vulnerabilities, as practitioners may wish for high precision to correctly identify the dangerous instances, but also for high recall to avoid false negatives—i.e., vulnerabilities which are labeled as "safe" by the algorithm, but are actually exploitable. From the point of view whereby it would be better to have false positives rather than false negatives, another trade-off has to be considered, namely using the False Positive Rate. Therefore, it is necessary to focus deep analyses on the AUC-ROC score, which considers the recall and False Positive Rate and encapsulates the compromise between the two values at different decision thresholds. We show the distributions of the considered performance metrics via the box plots depicted in Figures 3 and 4, which summarize the experiments run with the top-5 classifiers using the three IR weighting schemas and applying the three data balancing strategies.

From a general point of view, the observed performances are quite acceptable in most of the experiments run. In particular, the AUC-ROC score ranges from 70% to over 90%, and the F-measure is around 60% in most configurations. This encourages a positive response to **RQ**$_1$: machine learning models for early exploitability prediction of vulnerabilities seem to perform quite well using only the data available at disclosure time. Although the overall performances are not extremely high, they can be considered reasonable to meet the prior need for an *early* prediction model, namely the possibility to obtain preliminary indications on the dangerousness of the vulnerabilities affecting the system, to take countermeasures as soon as possible. It goes without saying that the utility of an *early* prediction model does not exhaust over time, as such a model is not meant to be disposable. Indeed, it can be continuously enhanced as soon as further data become available in the life cycle of vulnerabilities. We thoroughly discuss this uplifting scenario in Section V.

To dive deep into the results of the analyses, we report that most classifiers performed quite well with respect to the OPTIMISTIC, PESSIMISTIC, and RANDOM classifiers that we considered as a baseline for our study. This finding is supported by statistical tests results and corroborates the overall positive evaluation of the models. The only exception to this promising trend is given by the NAÏVE BAYES (NB) classifier—not included in the top-5—and reveals poor performance with respect to the other algorithms, particularly in terms of F-measure. As a matter of fact, this metric value never hits 50%, indicating a heavy tendency to misclassify. This could be probably due to the fact that NB requires totally independent features to perform well, which is not the case with textual features since some words may often co-occur. The scores do not improve when data balancing techniques are applied—NAÏVE BAYES performs even worse than the baseline RANDOM and PESSIMISTIC classifiers when using data weighted by simple counting (BoW) and augmented via SMOTE. The data balancing techniques heavily affect the performance of the KNN algorithm as well: it does not show significant differences across different weighting schemas, but its performance drops significantly when a data balancing approach is applied. More specifically, using NEARMISS undersampling leads to a decrease of 5% in the F-measure, and augmenting the dataset via SMOTE oversampling lowers the value by 10%. The reason behind this finding could be given by the fact that KNN is not good at handling a large number of features—which is the case for our datasets.

Answering our **RQ**$_3$, we assess that in almost all the experimental settings, the BoW weighting schema yields the worst performance among the three IR approaches. In particular, the LOGISTIC REGRESSION (LR) and SVM classifiers reveal significant evidence of this performance gap in terms of both F-measure and AUC-ROC score. This could be caused by the essence of the BoW approach itself, which only considers the number of occurrences of terms in the text without performing any readjustment operations based on their context. Weighting techniques, such as TF and TF-IDF, operate with respect to the document in which the terms appear, exhibiting better results. These considerations raise the idea that applying context-aware text pre-processing techniques might allow the acquisition of more relevant information embedded within the words. However, it is worth pointing out that the DECISION TREE (DT) classifier is the only one that achieves better performances when leveraging the BoW dataset, especially regarding AUC-ROC score. Moreover, it is particularly interesting to note that DT is the worst-performing algorithm among the top-5, while its "ensemble version", i.e., RANDOM FOREST (RF), showed to be the best one, especially on an oversampled dataset—this answers our **RQ**$_2$. We take this finding as a hint encouraging the exploration of *ensemble* learning strategies, which might reveal more satisfactory performances than traditional supervised approaches. As a matter of fact, the datasets used for our experiments were massively large, and previous work has demonstrated that as the dimensionality of data grows, RF can still deal with it, giving sufficiently high performances [20].

☞ **Answers to our RQs.** Exploitability prediction models show good-enough performances when exclusively leveraging on data available at the disclosure time of vulnerabilities. In particular, the RANDOM FOREST algorithm is the best-performing approach and shows the highest scores on datasets balanced via SMOTE oversampling. TF and TF-IDF provide higher-quality datasets when assigning the weights to the textual features.

We conducted a complementary investigation at a finer granularity to broaden the scope of the discussion and deeply assess the causes and meaning of the results. In particular, we run additional analyses to determine the overlap of the predictions output by the different classifiers and evaluate their *agreement*. The complete results of these analyses are available in the online appendix [4]. We found out that 2.8% of the instances were misclassified by all the non-baseline learning algorithms. Such instances are characterized by the absence of data derived from online discussions; therefore, they are only described by the brief definition contained in their CVE records. Although the simple CVE descriptions proved sufficient for other tasks concerning known vulnerabilities—e.g., severity prediction [40] or weakness type classification [5]—we conjecture that predicting exploitability requires additional information from different data sources that provide a more comprehensive overview of the security issue. As for the instances on which the classifiers disagreed, we noticed that 1.6% of them got properly classified by the second and third best classifiers, namely LOGISTIC REGRESSION and SVM, but got misclassified by RANDOM FOREST. This further encourages research on ensemble learning approaches and classification techniques leveraging majority voting, which could combine the forces of the individual methods and mitigate their mistakes.

V. FURTHER DISCUSSION AND TAKE-AWAY MESSAGES

On Potential Usages of Early Prediction Models. This work investigates the feasibility of adopting an *early* exploitability prediction model that provides initial feedback about just-disclosed vulnerabilities. We observed good-enough results that allow us to distill a set of practical implications from which practitioners may benefit. The first advantage allows security analysts, i.e., those responsible to carry out the CVSS measurement, to have extra support when they judge the vulnerability's nature. The second advantage is hastening the *vulnerability remediation* without necessarily waiting for the CVSS analysis before taking countermeasures. For instance, instead of scheduling the updates according to the CVSS base score, developers can use the responses from an early classifier to assign a higher priority to the vulnerabilities labeled as truly-exploitable. This strategy is also applicable for temporary workarounds while waiting for the vendor's patch.

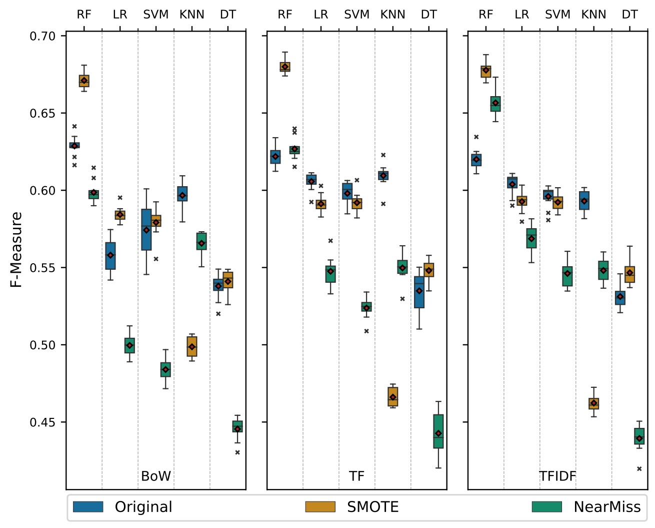

Fig. 3: The F-measure scores obtained by the top-5 learning algorithms using the datasets built according to the weighting schemas and adjusted with the data balancing approaches.

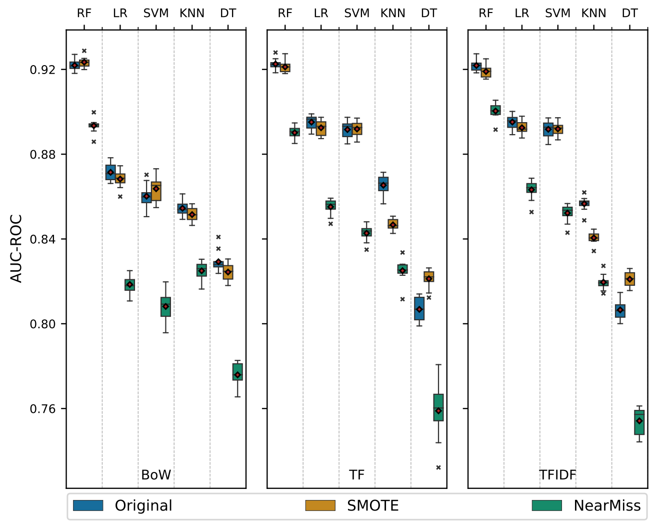

Fig. 4: The AUC-ROC scores obtained by the top-5 learning algorithms using the datasets built according to the weighting schemas and adjusted with the data balancing approaches.

Engineering The Feature Space. The results have shown that the most efficient models were obtained with the TF-IDF schema. This finding should not be surprising since TF-IDF is meant to give lesser relevance to those terms appearing at least once in many different documents—indicating their low discriminating power. All three experimented schemas rely upon the document-term matrix representation built using the same pre-processing steps that transformed documents into a bag of words. The main advantage of this technique over document embedding representations—such as DOC2VEC [51]—is that each feature represents elements in the documents (i.e., the terms) instead of some latent variables. Such a scenario enables in-depth analyses of the models to understand why they were able to make specific predictions, such as using *Explainable AI* techniques. Unfortunately, the Bag-of-Words method suffers from the so-called "curse of dimensionality" since the number of terms in the document corpus directly influences the number of features. In this work, we adopted some data cleaning and filtering actions to achieve a reasonable dimensionality but still risk dropping some relevant terms. In this respect, alternative IR approaches would be worth experimenting with, such as the *Latent Semantic Indexing* (LSI) [28], which clusters documents containing similar concepts and deals with cases of synonymy and polysemy. This technique transforms the document-term matrix into a *document-concept* matrix, where each concept would be mapped to a feature, reducing the feature space and, hopefully, improving the interpretability of the predictions.

Early Predictions and Beyond. We have commented on the new scenarios opened by adopting early exploitability prediction models. Nevertheless, our work did not aim at providing the best possible exploitability prediction model since we acknowledge the existence of better performance achieved by existing models in the literature. Thus, combining our contributions with all the efforts made by other researchers on this topic, we envision the employment of *incremental* exploitability prediction, i.e., an integrated system that provides predictions on the exploitability using as much information as possible at a given time. For instance, at the zero-day, it would rely only on the short description and the initial online discussions, just like we have presented in this paper; afterward, on the day the in-depth analyses are made, it would consider all the features obtainable from the CVSS vector to improve its predictive power hopefully. This solution may express its full usefulness in the case of borderline classifications, i.e., when the predictions fall too close to the decision threshold, making the model somehow unsure about the appropriate class. In such scenarios, the system might suggest waiting for additional data, such as the CVSS metrics, so that the incremented model can provide better responses.

VI. Threats to Validity

Threats to Construct Validity. We mined the full content of the National Vulnerability Database combined with CVE LIST to collect all the known vulnerabilities disclosed until November 03, 2021, being careful to avoid the inclusion of malformed and rejected CVEs. We did not perform an extensive manual validation of the retrieved dataset to detect possible curation errors, such as a CVE incorrectly pointing to an external reference related to a different vulnerability. Still, we are confident that the considered data sources are reliable since both databases are known to maintain high-quality data.

Still, regarding external references, we deliberately focused only on the URLs labeled as *BID* or *BUGTRAQ* for three reasons: (1) URLs of these kinds point to well-known sources where developers used to discuss vulnerabilities way before their official disclosure; (2) the pointed websites were easy to mine, with a common HTML parser, for collecting the required data; (3) developing a generic script able to mine all the thousands of websites reachable from the CVEs would have been impractical. We handled the shutdown of both SECURITY FOCUS and BUGTRAQ using the WAYBACK MACHINE service and the SECLISTS archive to recover the missing links with the CVE records. Nevertheless, we cannot guarantee the freedom from missing or incorrect links caused by WAYBACK MACHINE or the imprecision of the pattern matching heuristic employed to reach the right page on SECLISTS. In any case, the approach of early prediction models is not strictly bound to *BID* and *BUGTRAQ* references, and it can be adopted to any other source of online discussions with just minor tweaks.

The text of the online discussions contained many irrelevant data, such as e-mail addresses, PGP signatures, and hex numbers. We applied regular expressions to capture these patterns and remove them to improve the quality of the document corpus. Moreover, we adopted the recommended pre-processing steps when working with natural language text to generate a compact and representative vocabulary. We are aware that our data cleaning procedure may not be complete and could have left other forms of noise, such as partial code snippets; yet, to the best of our knowledge, there are no tools able to capture partial code elements for any programming language; hence we opted not to implement an ad-hoc solution as it would have required dedicated effort and extensive validation.

Threats to Internal Validity. Each CVE record was deemed an independent document made of its short description plus the content of all the *BID* and *BUGTRAQ* references. Before extracting the words vocabulary, we concatenated all these data sources together to handle those cases when a CVE did not have any *BID* and *BUGTRAQ* reference to avoid discarding those instances because of missing data. Our goal was to provide a flexible approach that applies to any CVE, either with or without external references, and easily adaptable to work with any other reference type.

As indicated by Bullough et al. [18], several exploitability prediction models in literature had some issue in their machine learning setup. We were careful to avoid applying the data balancing and hyper-parameter optimization to the entire dataset, but only on the training set made up at each iteration of the cross-validation. Unfortunately, the high number of tokens plus the very high sample size (over 140,000 instances) rendered any robust feature selection technique inapplicable due to extremely high memory consumption. Still, we reduced the feature space size by filtering out those tokens appearing in less than 0.1% of documents. We manually inspected the generated vocabularies at different thresholds—i.e., none, 0.01%, 0.05%, 0.1%, 0.5%, 1%—and selected the one showing the best trade-off between not losing highly-informative words [27] and having a reasonable number of features to tune and train the models [48], that is 0.1%.

Threats to External Validity. We used EXPLOIT DATABASE as the main source to build our ground truth (i.e., to label the CVEs with `true` and `false`) because of its reliability and completeness. Nevertheless, it only stores PoC and exploits publicly released by their authors without tracing any exploit in the wild (i.e., via attack signatures). Therefore, our models can only predict likely exploitations that are going to be publicly released without generalizing to other forms of exploits.

Threats to Conclusion Validity. From all the 81 configurations, we collected several metrics that capture the classifiers' performance from different points of view, reducing the risk of making erroneous conclusions. We also employed Friedman [37] and Nemenyi tests [60] to discern the significant differences between the performances of the configurations. Such tests are suitable when the distributions are not normal, as the ones we obtained, and to handle multiple comparisons [29]. We validated each model with a 10-fold cross validation to mitigate the possible effects that may occur when splitting the dataset. Our conclusions are only applicable to the *offline* scenario, i.e., when the disclosure date of the CVEs is not taken into account when making up the training and test sets.

VII. CONCLUSION

This paper presented a large-scale empirical evaluation of the effectiveness of *early* exploitability prediction models relying on the data available in a just-disclosed CVE, comparing many classifiers and data balancing algorithms using three different weighting schemas to create the datasets. The results highlighted that the choice of the weighting schema determines non-trivial implications on the models' capabilities. In particular, the simple word counting approach led to the poorest performances. Encouraging results have been provided by the RANDOM FOREST algorithm, which is shown to be the best-performing, especially on a dataset balanced via SMOTE oversampling. From what we observed, the ensemble methods could further improve the quality of the predictions, which we plan to explore in our future work.

Additional research directions include an extensive comparison of early and traditional exploitability prediction models, i.e., assessing whether the use of data arriving days after the disclosure actually benefits or damages the performance. Then, our envisioned *incremental* prediction system could be deployed in a realistic context to see how it supports vulnerability remediation and how developers react to its recommendations. From a different perspective, we hypothesize that exploitability prediction modeling can be further improved by using fine-grained text analysis tools that extract relevant elements from an unstructured text, such as code snippets or stack traces, to have a more-relevant feature space.

REFERENCES

[1] L. Allodi and F. Massacci. Comparing vulnerability severity and exploits using case-control studies. *ACM Trans. Inf. Syst. Secur.*, 17(1), aug 2014.

[2] M. Almukaynizi, E. Nunes, K. Dharaiya, M. Senguttuvan, J. Shakarian, and P. Shakarian. Proactive identification of exploits in the wild through vulnerability mentions online. In *2017 International Conference on Cyber Conflict (CyCon U.S.)*, pages 82–88, 2017.

[3] Y. AlNoamany, A. AlSum, M. C. Weigle, and M. L. Nelson. Who and what links to the internet archive. *International Journal on Digital Libraries*, 14(3):101–115, Aug 2014.

[4] Anonymous. Early exploitability prediction of just-disclosed software vulnerabilities – online appendix. https://figshare.com/s/165b39c7094c7831365e, 2022.

[5] M. Aota, H. Kanehara, M. Kubo, N. Murata, B. Sun, and T. Takahashi. Automation of vulnerability classification from its description using machine learning. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2020.

[6] A. Arora, A. Nandkumar, and R. Telang. Does information security attack frequency increase with vulnerability disclosure? an empirical analysis. *Information Systems Frontiers*, 8:350–362, 01 2006.

[7] A. Arusoaie, S. Ciobâca, V. Craciun, D. Gavrilut, and D. Lucanu. A comparison of open-source static analysis tools for vulnerability detection in c/c++ code. In *2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pages 161–168, 2017.

[8] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books. ACM Press, 1999.

[9] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. *Encyclopedia of Software Engineering*, 1994.

[10] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[11] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim. Extracting structural information from bug reports. In *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, MSR '08, page 27–30, New York, NY, USA, 2008. Association for Computing Machinery.

[12] N. Bettenburg, E. Shihab, and A. E. Hassan. An empirical study on the risks of using off-the-shelf techniques for processing mailing list data. In *2009 IEEE International Conference on Software Maintenance*, pages 539–542, 2009.

[13] N. Bhatt, A. Anand, and V. S. S. Yadavalli. Exploitability prediction of software vulnerabilities. *Quality and Reliability Engineering International*, 37(2):648–663, 2021.

[14] L. Bilge and T. Dumitraş. Before we knew it: An empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 833–844, New York, NY, USA, 2012. Association for Computing Machinery.

[15] M. Bozorgi, L. K. Saul, S. Savage, and G. M. Voelker. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, page 105–114, New York, NY, USA, 2010. Association for Computing Machinery.

[16] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[17] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen. *Classification and regression trees Regression trees*. Chapman and Hall, 1984.

[18] B. L. Bullough, A. K. Yanchenko, C. L. Smith, and J. R. Zipkin. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, IWSPA '17, page 45–53, New York, NY, USA, 2017. Association for Computing Machinery.

[19] G. Canfora, A. Di Sorbo, S. Forootani, A. Pirozzi, and C. A. Visaggio. Investigating the vulnerability fixing process in oss projects: Peculiarities and challenges. *Computers Security*, 99:102067, 2020.

[20] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 96–103, New York, NY, USA, 2008. Association for Computing Machinery.

[21] G. Catolino, F. Palomba, A. Zaidman, and F. Ferrucci. Not all bugs are the same: Understanding, characterizing, and classifying bug types. *Journal of Systems and Software*, 152:165–181, 2019.

[22] H. Cavusoglu, H. Cavusoglu, and S. Raghunathan. Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. *IEEE Transactions on Software Engineering*, 33(3):171–185, 2007.

[23] O. Chaparro, C. Bernal-Cárdenas, J. Lu, K. Moran, A. Marcus, M. Di Penta, D. Poshyvanyk, and V. Ng. Assessing the quality of the steps to reproduce in bug reports. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, page 86–96, New York, NY, USA, 2019. Association for Computing Machinery.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, jun 2002.

[25] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[26] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[27] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella. Applying a smoothing filter to improve ir-based traceability recovery processes: An empirical investigation. *Information and Software Technology*, 55(4):741–754, 2013.

[28] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[29] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, dec 2006.

[30] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar. Software security patch management - a systematic literature review of challenges, approaches, tools and practices. *Information and Software Technology*, 144:106771, 2022.

[31] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, page 475–488, New York, NY, USA, 2014. Association for Computing Machinery.

[32] M. Edkrantz and A. Said. Predicting cyber vulnerability exploits with machine learning. In *SCAI*, pages 48–57, 2015.

[33] C. Elbaz, L. Rilling, and C. Morin. Fighting n-day vulnerabilities with automated cvss vector prediction at disclosure. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ARES '20, New York, NY, USA, 2020. Association for Computing Machinery.

[34] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner. Security testing: A survey. In *Advances in Computers*, volume 101, pages 1–51. Elsevier, 2016.

[35] S. Frei, M. May, U. Fiedler, and B. Plattner. Large-scale vulnerability analysis. In *Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense*, LSAD '06, page 131–138, New York, NY, USA, 2006. Association for Computing Machinery.

[36] S. Frei, D. Schatzmann, B. Plattner, and B. Trammell. *Modeling the Security Ecosystem - The Dynamics of (In)Security*, pages 79–106. 07 2010.

[37] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[38] C. Fruhwirth and T. Mannisto. Improving cvss-based vulnerability prioritization and response with context information. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 535–544, 2009.

[39] S. Haiduc, G. Bavota, R. Oliveto, A. De Lucia, and A. Marcus. Automatic query performance assessment during the retrieval of software artifacts. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE 2012, page 90–99, New York, NY, USA, 2012. Association for Computing Machinery.

[40] Z. Han, X. Li, Z. Xing, H. Liu, and Z. Feng. Learning to predict severity of software vulnerability using only vulnerability description. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 125–136, 2017.

[41] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[42] H. Holm, M. Ekstedt, and D. Andersson. Empirical analysis of system-level vulnerability metrics through actual attacks. *IEEE Transactions on Dependable and Secure Computing*, 9(6):825–837, 2012.

[43] K. Huang, B. Chen, L. Pan, S. Wu, and X. Peng. Repfinder: Finding replacements for missing apis in library update. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 266–278. IEEE, 2021.

[44] E. Iannone, R. Guadagni, F. Ferrucci, A. De Lucia, and F. Palomba. The secret life of software vulnerabilities: A large-scale empirical study. *IEEE Transactions on Software Engineering*, pages 1–1, 2022.

[45] J. Jacobs, S. Romanosky, I. Adjerid, and W. Baker. Improving vulnerability remediation through better exploit prediction. *Journal of Cybersecurity*, 6(1), 09 2020. tyaa015.

[46] J. Jacobs, S. Romanosky, B. Edwards, I. Adjerid, and M. Roytman. Exploit prediction scoring system (epss). *Digital Threats*, 2(3), jul 2021.

[47] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y. Le Traon, and M. Harman. The importance of accounting for real-world labelling when predicting software vulnerabilities. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, page 695–705, New York, NY, USA, 2019. Association for Computing Machinery.

[48] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Machine Learning: ECML-98*, pages 137–142, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[49] A. Khazaei, M. Ghasemzadeh, and V. Derhami. An automatic method for cvss score prediction using vulnerabilities description. *Journal of Intelligent & Fuzzy Systems*, 30:89–96, 2016. 1.

[50] R. G. Kula, D. M. German, A. Ouni, T. Ishio, and K. Inoue. Do developers update their library dependencies? *Empirical Software Engineering*, 23(1):384–417, Feb 2018.

[51] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[52] X. Li, P. Liang, and Z. Li. Automatic identification of decisions from the hibernate developer mailing list. In *Proceedings of the Evaluation and Assessment in Software Engineering*, EASE '20, page 51–60, New York, NY, USA, 2020. Association for Computing Machinery.

[53] H. Liang, X. Pei, X. Jia, W. Shen, and J. Zhang. Fuzzing: State of the art. *IEEE Transactions on Reliability*, 67(3):1199–1218, 2018.

[54] F. Lomio, E. Iannone, A. De Lucia, F. Palomba, and V. Lenarduzzi. Just-in-time software vulnerability detection: Are we there yet? *Journal of Systems and Software*, page 111283, 2022.

[55] J. Lyu, Y. Bai, Z. Xing, X. Li, and W. Ge. A character-level convolutional neural network for predicting exploitability of vulnerability. In *2021 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 119–126, 2021.

[56] B. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442–451, 1975.

[57] G. McGraw. *Software Security: Building Security in*. Addison-Wesley professional computing series. Addison-Wesley, 2006.

[58] S. Menard. *Applied logistic regression analysis*, volume 106. Sage, 2002.

[59] P. Morrison, K. Herzig, B. Murphy, and L. Williams. Challenges with applying vulnerability prediction models. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, HotSoS '15, New York, NY, USA, 2015. Association for Computing Machinery.

[60] P. Nemenyi. Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1962.

[61] F. Pecorelli, D. Di Nucci, C. De Roover, and A. De Lucia. A large empirical assessment of the role of data balancing in machine-learning-based code smell detection. *Journal of Systems and Software*, 169:110693, 2020.

[62] F. Pecorelli, F. Palomba, D. Di Nucci, and A. De Lucia. Comparing heuristic and machine learning approaches for metric-based code smell detection. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pages 93–104, 2019.

[63] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, Jan 1980.

[64] D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[65] J. Ratcliff and D. Metzener. Pattern matching: the gestalt approach. https://www.drdobbs.com/database/pattern-matching-the-gestalt-approach/184407970?pgno=5, 1988. Accessed: 2022-03-125.

[66] A. Reinthal, E. L. Filippakis, and M. Almgren. Data modelling for predicting exploits. In N. Gruschka, editor, *Secure IT Systems*, pages 336–351, Cham, 2018. Springer International Publishing.

[67] S. Renatus, C. Bartelheimer, and J. Eichler. Improving prioritization of software weaknesses using security models with avus. In *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 259–264. IEEE, 2015.

[68] I. Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[69] S. Rose, D. Engel, N. Cramer, and W. Cowley. *Automatic Keyword Extraction from Individual Documents*, chapter 1, pages 1–20. John Wiley Sons, Ltd, 2010.

[70] C. Sabottke, O. Suciu, and T. Dumitraş. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, page 1041–1056, USA, 2015. USENIX Association.

[71] A. D. Sawadogo, Q. Guimard, T. F. Bissyandé, A. K. Kaboré, J. Klein, and N. Moha. Early detection of security-relevant bug reports using machine learning: How far are we? *CoRR*, abs/2112.10123, 2021.

[72] A. Schroter, A. Schröter, N. Bettenburg, and R. Premraj. Do stack traces help developers fix bugs? In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 118–121, 2010.

[73] C. Silva and B. Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1661–1666 vol.3, 2003.

[74] M. Soltani, F. Hermans, and T. Bäck. The significance of bug report elements. *Empirical Software Engineering*, 25(6):5255–5294, Nov 2020.

[75] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, Jan 1972.

[76] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133, 1974.

[77] J. Viega, J. Bloch, Y. Kohno, and G. McGraw. Its4: a static vulnerability scanner for c and c++ code. In *Proceedings 16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 257–267, 2000.

[78] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[79] J. Zhang and I. Mani. KNN Approach to Unbalanced Data Distributions: A Case Study Involving Information Extraction. In *Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Datasets*, 2003.

[80] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[81] Z. Zhang. Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, 4(11), 2016.