Computers & Security 131 (2023) 103286

Contents lists available at ScienceDirect

Computers & Security

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

Common vulnerability scoring system prediction based on open source intelligence information sources

Philipp Kühn*, David N. Relke, Christian Reuter

Science and Technology for Peace and Security (PEASEC), Technical University of Darmstadt, Germany

ARTICLE INFO

Article history:
Received 10 October 2022
Revised 29 March 2023
Accepted 2 May 2023
Available online 9 May 2023

Keywords:
IT Security
Common vulnerability scoring system
Classification
National vulnerability database
Security management
Deep learning

ABSTRACT

The number of newly published vulnerabilities is constantly increasing. Until now, the information available when a new vulnerability is published is manually assessed by experts using a Common Vulnerability Scoring System (CVSS) vector and score. This assessment is time consuming and requires expertise. Various works already try to predict CVSS vectors or scores using machine learning based on the textual descriptions of the vulnerability to enable faster assessment. However, for this purpose, previous works only use the texts available in databases such as National Vulnerability Database. With this work, the publicly available web pages referenced in the National Vulnerability Database are analyzed and made available as sources of texts through web scraping. A Deep Learning based method for predicting the CVSS vector is implemented and evaluated. The present work provides a classification of the National Vulnerability Database's reference texts based on the suitability and crawlability of their texts. While we identified the overall influence of the additional texts is negligible, we outperformed the state-of-the-art with our Deep Learning prediction models.

1. Introduction

IT systems are now ubiquitous and fundamental to society, businesses, and individuals. Failures and disruptions can have catastrophic consequences for those affected. In 2017, for example, two waves of ransomware attacks occurred, each resulting in major outages to businesses and infrastructure (Elizabeth Dwoskin and Karla Adam, 2017; McQuade, 2018). The vulnerability that enabled these attacks had been known and fixed a month before the first attack. In other attacks, such as the one on Microsoft Exchange Server in early 2021, only a few days passed between the discovery of the vulnerability and the start of attacks (Brian Krebs, 2021).

It is therefore important for researchers or system administrators to learn about vulnerabilities as early as possible, analyze them and initiate countermeasures. Various publicly accessible databases, such as the National Vulnerability Database (NVD)[1] and the Common Vulnerabilities and Exposures (CVE)[2] collect, structure and prepare the published vulnerabilities for this purpose. However, relevant information can also be found on many other platforms, such as social media (especially Twitter), blogs, news portals, and company websites.

The Common Vulnerability Scoring System (CVSS) is used to categorize different aspects of vulnerabilities. The result of this categorization is a vector whose elements are a machine-readable representation of the vulnerability's properties.[3] Based on the components of the Common Vulnerability Scoring System (CVSS) vector a numerical vulnerability score (CVSS severity score) is calculated. The vulnerability assessment is usually performed by IT security experts based on the available Open Source Intelligence (OSINT) information. OSINT refers to the structured collection and analysis of information that is freely available to the public.

There is a certain period of time when the information about a new vulnerability is published, but the assessment made by experts is not yet available (Elbaz et al., 2020; Ruohonen, 2019). Due to the large mass of published vulnerabilities, it is difficult for researchers or, *e.g.*, responsible persons in companies to assess each new vulnerability themselves. They are therefore dependent on the assessments of experts. Accordingly, the longer it takes for the assessment to become available, the longer it takes for countermeasures to be taken to mitigate the vulnerability. During this period, the vulnerable systems are vulnerable to attack without the

* Corresponding author.
E-mail addresses: kuehn@peasec.tu-darmstadt.de (P. Kühn), david.relke@stud.tu-darmstadt.de (D.N. Relke), reuter@peasec.tu-darmstadt.de (C. Reuter).

[1] nvd.nist.gov.
[2] cve.mitre.org.
[3] https://www.first.org/cvss/specification-document.

https://doi.org/10.1016/j.cose.2023.103286
0167-4048/

responsible parties knowing about it. It is therefore important that the assessment is available as soon as possible.

Various works (Elbaz et al., 2020; Han et al., 2017; Shahid and Debar, 2021) try to perform this assessment automatically based on the textual information available about a vulnerability using Machine Learning (ML). This would allow for a much faster assessment. The vulnerability could already be assessed in an automated way when it is published and the time window in which no at least preliminary assessment is available is kept small. It would also allow experts to prioritize and make recommendations for the assessment.

Previous work largely uses only the short descriptions of vulnerabilities from NVD and CVE with some exceptions (Almukaynizi et al., 2017; Chen et al., 2019b). Han et al. (2017), for instance, present a system for classifying vulnerabilities into different severity levels based on CVSS. From Khazaei et al. (2016) comes a work on predicting the numerical CVSS severity score. In addition, there are methods that automatically predict the entire CVSS vector (Elbaz et al., 2020). Another work by Kuehn et al. (2021) describes a system that uses Deep Learning to predict the CVSS vector. However, the system requires labels created by experts to train, which significantly increases the required effort for larger datasets. Further, Deep Learning (DL) profits from large training datasets to which the reference texts could contribute, which is currently not leverage by related work. While crawling and analyzing OSINT information might pose threats to individuals, *e.g.*, privacy intrusion, it is mandatory to make current prediction systems more robust (Riebe et al., 2023).

Goal This work aims to use as much textual data as possible to predict the CVSS vector of a vulnerability. This is to achieve the most accurate estimation of the CVSS vector possible. It should be possible to use not only the short description of the vulnerability, but also other types of texts, such as Twitter posts and news articles for prediction in case of a new vulnerability. Possible sources of textual information about vulnerabilities should be found and categorized. We aim to answer the following research questions: *Where can relevant textual information on vulnerabilities be found outside vulnerability databases **(RQ1)**?* and *To which degree are public data sources beyond vulnerability databases suitable for predicting the CVSS vector **(RQ2)***? This will clarify whether there are typical sources that regularly report on current vulnerabilities and whether these are suitable as a basis for building a dataset for training a ML system.

Here, a first impression shall be gained by a rough manual search and then the sources referenced in the databases shall be analyzed automatically with regard to the type and scope of the references (*e.g.*, blog posts, patchnotes, GitHub issues). With the help of the texts, a ML model for predicting the CVSS vector is to be trained. The data must be filtered and cleaned for this purpose. The ML model shall use Deep Learning and use state-of-the-art models as a basis. The model is evaluated and compared to previous work.

Contributions The contribution to current research is an analysis of the references contained in the databases. This will categorize the references in terms of certain characteristics and suitable for ML models and can serve as a starting point for further work on the use of the references **(C1)**. A method that collects and processes the text contained on the referenced web pages will be presented. In addition, a system is implemented and evaluated that, unlike previous work, such as Elbaz et al. (2020) and Kuehn et al. (2021), uses more extensive text from the references in addition to descriptions of vulnerabilities from the databases **(C2)**. This method for predicting CVSS vectors surpasses the current state-of-the-art. Further, do we present an extensive explainability analysis of our trained models as part of our evaluation **(C3)**.

Outline The state of the art in research is considered in Section 2, followed by a preliminary analysis of the references included in NVD (*cf.* Section 3). Requirements for references and the texts contained in them are defined and consequently the individual references are evaluated, resulting in a selection of references. Section 4 explains the procedure for collecting the texts from the references and a system for retrieving, processing, and storing the texts is presented. Section 5 evaluates the ML system, while Section 6 discusses and compares the results with other work. Finally, a conclusion is drawn in Section 7.

2. Related work

This section gives an overview over the state of the art in research. We focus literature dealing with the prediction of CVSS vectors, scores, or levels. In addition, work that uses sources other than NVD in this context is considered. Automated assessment should provide a time advantage over the assessment by human experts. In this regard, different papers come to different conclusions regarding the duration of the assessment, and the exact methodology is not always clear. Elbaz et al. (2020) state for the observed period from 2007 to 2019 that 90% of vulnerabilities were assessed within just under 30 days, with a median of only one day, while Chen et al. (2019b) indicate an average of 132 days between publication and assessment for an observed period of 23 months in 2018 and 2019.

NVD, CVSS, Information Sources Johnson et al., 2018 perform a statistical analysis of CVSS vectors in different databases containing vulnerabilities. In doing so, they show that despite different sources, the CVSS vector is always comparable and, consequently, seem to be robust. They state the NVD is the most robust information source for CVSS information. On the other hand, Dong et al. (2019) show that information in the NVD itself is sometimes inconsistent and propose a system that relies on external sources to find, for example, missing versions of the software in question in the NVD. Accordingly, Kuehn et al. (2021) present an information quality metric for vulnerability databases and improve several drawbacks in the NVD. In addition to vulnerability databases, other sources of information are used in vulnerability management. Sabottke et al. (2015) use Twitter to predict whether a vulnerability will actually be exploited. Almukaynizi et al. (2017) go a step further and use other data sources, such as ExploitDB[4] and Zero Day Initiative[5]. However, no text is used, but the simple existence of an article about a vulnerability is used as a feature for the ML model.

CVSS Prediction A large number of works deal with the prediction of CVSS vector, scores, or levels starting from text. As one of the first works, Yamamoto et al. (2015) use sLDA (Mcauliffe and Blei, 2007) to predict the CVSS vector based on the descriptions. For predicting the score, Khazaei et al., 2016 use Support Vector Machines (SVMs), random forests (Breiman, 2001), and fuzzy logic. Spanos et al. (2017) predict the CVSS vector using random forests and boosting (Freund and Schapire, 1999). DL is first used in this context by Han et al. (2017). By using an Convolutional Neural Network (CNN), no feature engineering is required. However, in doing so, the model only determines the CVSS severity level from the options *Critical, High, Medium*, and *Low*. Gawron et al. (2018) use DL in addition to Naive Bayes, but here the result is a CVSS vector. Twitter serves as the data source for Chen et al. (2019a). The ML model is based on Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and predicts CVSS score. Sahin and Tosun (2019) also

[4] https://www.exploit-db.com/.
[5] https://www.zerodayinitiative.com/.

improve on the Han et al. (2017) approach by using a LSTM. Gong et al. (2019) show a multi-task learning method that sets up multiple classifiers on a single Neural Network (NN), making it more efficient. Liu et al. (2019) use the Chinese equivalent, the China National Vulnerability Database of Information Security (CNNVD), as the data source rather than the NVD. Jiang and Atif (2021) take scores not only from the NVD but also from other sources as a basis for their prediction of the score. The work of Elbaz et al. (2020) focuses on a particularly tractable classification of the CVSS vector. Therefore, they do not use dimension reduction techniques. Kuehn et al. (2021) use DL to predict the CVSS vector, based on the NVD's descriptions, with the goal to aid security experts in their final decision. The most recent approach proposed Shahid and Debar (2021), which uses a separate classifier based on a Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2018) to determine the CVSS vector for each component of the vector. Several proposals rely solely on the textual data from the NVD. Some use text from Twitter or simple binary features, such as the existence of an article about a particular vulnerability. Other vulnerability context tasks also use few different data sources. Yitagesu et al. (2021) also use Twitter as a source for a model for Part-of-Speech (POS) tagging. Liao et al. (2016) propose a system which draws on several sources to filter Indicators of Compromise (IoC) from natural text.

Research Gap OSINT is widely used in IT security (Chen et al., 2019; Liao et al., 2016; Pastor-Galindo et al., 2020; Sabottke et al., 2015). Various works exist on the prediction of CVSS vectors based on descriptions. However, as research shows, few OSINT vulnerability sources are used (Le et al., 2021), especially in the context of CVSS score, level, or vector prediction, and if they are, very simple features from other sources are used (Almukaynizi et al., 2017). Furthermore, there is no systematic analysis of the suitablility of NVD references for CVSS vector prediction approaches.

3. Preliminary analysis

The authors performed an exploratory analysis of the available data, i.e. vulnerability descriptions and outgoing references from the NVD, to identify data suitability criteria and requirements for web scraping. Suitable in the sense of the present work are texts that describe a vulnerability and can be directly assigned to a vulnerability via the CVE identification number. Some special factors have to be considered:

- Each text shall be uniquely assignable to one and only one vulnerability via the CVE identification number. Without this criterion a text could be used as a training example for two different permutations of one of the components of the CVSS vector. This makes it difficult for the ML algorithm to identify the relevant properties of the vulnerability. Also, since the vulnerabilities covered in a text may be very different, it does not make sense to use the same text for multiple vulnerabilities. It is even possible that only one vulnerability is really described, although several with different target vectors are mentioned.
- The texts should not contain the target variable, *i.e.*, the CVSS vector. Otherwise, the ML model could predict the target parameter based on the variable present in the input, without any actual meaningful learning effect.
- There should be as little noise as possible. This ensures a high quality of the prediction. As stated in Section 2, the data otherwise contain patterns that could negatively affect the ML model.

Our secondary goal with this exploratory analysis is to identify *where to find usable data, assess the data quality* and *how it can be used*. Those questions correlate with our research questions (*cf.* Section 1).

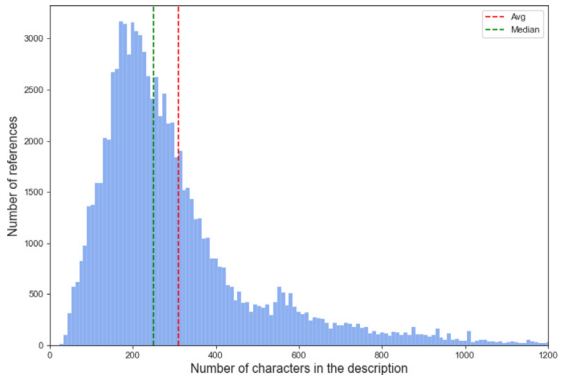

Fig. 1. Distribution of National Vulnerability Database description lengths.

3.1. Descriptions in the NVD

The first and most important starting point for finding texts about vulnerabilities is the NVD. We consider NVD entries from 2016 to 2021, based on the introduction of the current CVSS standard version 3. Entries without CVSS version 3 information are excluded. This is the case for vulnerabilities in 2016, when CVSSv3 was still in the process of wide adoption, and in 2021, where the CVSSv3 vector was not yet available at the time the entries were retrieved. In total, we collected 88 979 entries.

Individual entries in the NVD contain a short, expert curated[6] description of the vulnerability. The length of the descriptions for our collected entries ranges between 23 and 3835 characters, with an average of 310 and a median of 249. Fig. 1 shows the distribution of the length of the descriptions. Descriptions longer than 1000 characters are very rare, with the 95th percentile already at 746 characters. The information content of texts correlates with the pure length of the texts, apart from some exceptions.[7] Likewise, a single, short sentence cannot describe all aspects of the vulnerability. As Fig. 1 illustrates, there are a large number of vulnerabilities in NVD with very short descriptions.

Literature shows that the quality of vulnerability descriptions in the NVD differs (Kuehn et al., 2021) and the quality can only be assessed to a limited extent without a deeper analysis. A random sample shows that many descriptions contain less information about the actual vulnerability, but list, *e.g.*, affected products and version numbers. Such information is unrelated to the characteristics of the vulnerability and is therefore of little usefulness to predict the vulnerability severity. Nevertheless, Shahid and Debar (2021) show that good results in the prediction of the CVSS vector are possible based only on NVD descriptions. Their method of CVSS score prediction achieves a Mean Squared Error (MSE) of 1.79 and a correctly predicted score in 53% of all cases.

3.2. Reference analysis

Each NVD entry references websites. To identify, which websites are suitable to be crawled we first analyze what kind of references are involved and, based on these insights, build categories for reference domains. Second, we rate these groups based on their crawlability and potential text quality.

In the given subset of all entries of the NVD there are a total of 2 51 485 references. The median number of references per vulnerability is 2. Many vulnerabilities have only a single reference,

[6] https://www.cve.org/ResourcesSupport/FAQs#pc_cve_recordscve_record_descriptions_created.

[7] Some descriptions list other, non-identical, vulnerabilities, which artificially increases the length of the description without giving further content.

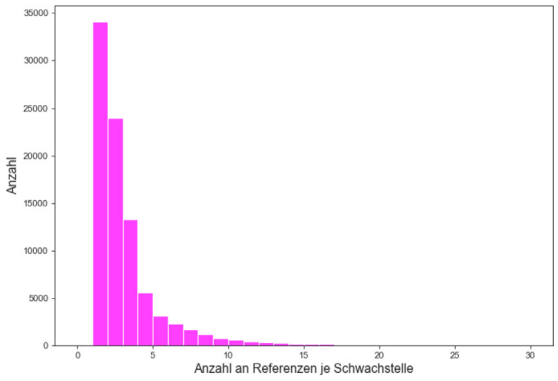

Fig. 2. Distribution of number of references per vulnerability in National Vulnerability Database.

95% have 8 or fewer references. There are a few outliers with over 100 references. The distribution of the number of references can be seen in Fig. 2.

Over time, the diversity of references increased slightly. From 2016 until 2021 there are 6013 different referenced domains, of which about 75% are accounted for by the 50 most frequent ones. In 2016, 990 different domains are referenced with 86% of all references coming from the 50 most referenced websites. In 2021, this trend increases to a total of 1711 domains referenced and the top 50 account for 74% of all references, showing an increase of diversity. We build the 100 most frequently referenced NVD reference domains based on our dataset (*cf.* Table 1). These domains account for 83% of all references in NVD.

Table 1 shows the 30 most frequently referenced domains, with additional entries to properly represent each group.

We analyze which domains contain suitable descriptions of vulnerabilities and to what extent they are usable. Based on their characteristics, we derive groups of references. In the following, we present and describe the six identified groups in conjunction with some sample domains.

(1) **Version control and bug tracker services**
Examples: GitHub, crbug.com, bugzilla.mozilla.com
These sites mostly contain program code, output and log files, technical descriptions, and bug discussions. A more abstract description of the vulnerabilities is rarely found. The domains are operated by the producers of the software, but contributions by users are also possible. Hence, there is not always an information verification by experts. On some sites the structure of the references is always identical, on others the structure is inconsistent.

(2) **Mailing Lists**
Examples: lists.fedoraproject.org, lists.apache.org, lists.debian
Contributions origin from different individual users and are mostly unstructured and inconsistent texts and code fragments. As a result, some references to a domain may allow a unique mapping from CVE-ID to text, while this is not possible for other references to the same domain. Descriptions of vulnerabilities may be present, however, these are predominantly technical details. On some domains, vulnerabilities fixed with an update are also only mentioned without further text.

(3) **Patchnotes**
Examples: support.apple.com, oracle.com, helpx.adobe.com
These are often maintained large commercial vendors. A single reference to one of the domains in this group typically contains information about many different vulnerabilities that have been closed with an update. On some domains, descriptions of the vulnerabilities are published, on others, the CVE-ID is only mentioned. References to one and the same domain have mostly identical structures over the whole observed period. The articles are written by employees of the respective companies.

(4) **Security Advisories**
Examples: tools.cisco.com, security.gentoo.org, ibm.com
Vendors describe vulnerabilities in their own products in more detail on domains in this group. Often, only one vulnerability is covered in a reference. The structures of the references on a domain are the same. The descriptions of vulnerabilities are relatively detailed. The authors are employees of the respective companies.

(5) **Third party articles about vulnerabilities**
Companies or users publish articles on domains of this group about weak points in the products of other manufacturers. In some cases, this is part of a commercial business model based on services. Unlike the vulnerability-focused mailing lists, the structure of these posts is consistent. The contributions on some sites origin from professional employees, while on other sites unverified users are the authors of the texts.

(6) **Blog posts and social media**
Examples: medium.com, twitter.com, groups.google.com
References to domains from this group show high diversity. The structure of the contributions is inconsistent. Authors may be professional contributors as well as unverified users. A clear assignment of CVE-ID to text depends on the authors of the specific contributions, not on the website itself.

Table 1
The 30 most referenced domains in NVD, with additional entries to properly represent each identified group. # gives the position in the Top-100, *Num.* refers to how many times the given domain is referenced, *Gr.* gives the assigned group, and *Avail.* depicts, whether unavailable domain is unreachable, redirect to domains unrelated to the vulnerability, or are only reachable after login.

#	URL	Num.	Gr.	Avail.
1	github.com	25,064	1	√
2	securityfocus.com	20,645	2	√
3	www.securitytracker.com	10,842	2	√
4	access.redhat.com	8627	3/4	√
5	support.apple.com	8069	3	√
6	lists.opensuse.org	7930	2	√
7	lists.fedoraproject.org	7212	2	√
8	www.oracle.com	7006	3	√
9	lists.apache.org	6294	2	√
10	www.debian.org	5614	2/3	√
11	security.gentoo.org	5289	4	√
12	usn.ubuntu.com	5225	3	√
13	lists.debian.org	4921	2	√
14	portal.msrc.microsoft.com	4391	3	√
15	www.openwall.com	4136	2	√
16	packetstormsecurity.com	4068	4	√
17	source.android.com	3672	3	√
18	seclists.org	3462	2	√
19	www.exploit-db.com	3412	5	√
20	tools.cisco.com	3019	4/5	√
21	security.netapp.com	2890	5	√
22	ibm.com	2807	4	√
23	exchange.xforce.ibmcloud.com	2673	4	√
24	helpx.adobe.com	2643	3	√
25	zerodayinitiative.com	2547	5	√
26	bugzilla.redhat.com	2482	1	√
27	rhn.redhat.com	2019	4	√
28	www.mozilla.org	1785	4	√
29	crbug.com	1458	1	√
30	www.ubuntu.com	1397	3	√
...				
33	bugzilla.mozilla.org	1075	1	√
...				
47	wpscan.com	791	5	√
...				
66	medium.com	443	6	√

Table 2
Overview of the 5-point scale evaluation for the usability of the different groups in combination with the usual origin of the content.

Group	Origin	Unique	Uniform	Abs. text
VCS/Bug Tracker	*User*	√√√	√√√	√√
Mailing Lists	*User*	√√	√	√√
Patchnotes	*Vendor*	√√√	√√√√√	√√
Advisories	*Vendor*	√√√√√	√√√√√	√√√√
Third Party	3rd-P.	√√√√	√√√√√	√√√√
Blogs/Social Media	*User*	√√	√	√√√

Our criteria for the usability of texts (*cf.* Section 3) cannot be met by general purpose crawling approaches, like trafilatura (Barbaresi, 2021), which ignore the characteristics of the target-domain. Instead, solutions must be tailored to the target domain. This is the only way to extract texts from the references that meet our requirements. Since numerous domains are referenced, a pre-selection must be made.

The presented groups differ in terms of the usability of the references. Within the groups the domains are differently suitable. Ideally, text references enable a unique mapping from a CVE-ID to text. The text must be an abstract description, since technical details such as code descriptions out-of-scope in the present work. Since web scraper use the Hypertext Markup Language (HTML) source code's structure of the domain to extract the correct text, individual references to a domain should therefore always have the same structure.

A unified structure is used on domains where contributions are published or at least reviewed by a single entity. For the first and second group, there is only a higher-level structure, but not a uniform structure of the actual contribution. For example, the basic structure of a reference to an issue in GitHub is always the same, however, the structure of the actual issue description might differ in each case. Table 2 shows an simplified overview of the different groups, whether they meet the uniqueness-, uniformity-, and abstract-text-requirements based on a 5-point scale.

Domain Selection For the domain selection, it must be considered whether it is worth the effort to adapt a web scraper for a domain. Pages with the same structure and content require less effort and promise a better yield, as the texts will be more likely to meet the established criteria.

Starting from the frequency ranking of domains (*cf.* Table 1), a domain selection is made based on the domains group and the group ranking of Table 2.

- ibm.com
  Group 4 - 3447 References
  IBM publishes collected information about vulnerabilities in its own products. The individual texts are rather short. The assignment of text to CVE ID is easy thanks to the uniform structure of the articles.
- tools.cisco.com
  Group 4 - 3019 references.
  Cisco publishes detailed descriptions for vulnerabilities in its own products or in third-party products that Cisco uses or integrates into its own products, such as frameworks. In addition, technical details and code are sometimes included. The structure of the articles is very similar.
- zerodayinitiative.com
  Group 5 - 2899 references. Trend Micro[8] acts as a middleman between the discoverers of zero-day vulnerabilities and the manufacturers of the affected products. The advisories are then published. The structure and type of description are always the same.
- talosintelligence.com
  Group 5 - 1335 References
  Talos is a commercial company belonging to Cisco offering services and products related to IT security. The website publishes articles about vulnerabilities discovered by Talos. The articles are very detailed. The text on the website includes code, version numbers, CVSS vector and other information in addition to the description. However, the text itself is structured by headings that are consistent for all posts.
- qualcomm.com
  Group 3/4 - 1048 References
  Contains information collected monthly on vulnerabilities in Qualcomm products. Descriptions are brief. The structure is consistent, and the articles are sorted into tables. Partially the URLs deposited in the NVD are incorrect, because Qualcomm has changed the Uniform Resource Locator (URL) scheme over time. However, the monthly posts are still accessible under a modified URL.
- support.f5.com
  Group 5 - 932 references
  F5 provides commercial IT security services and products. The referenced papers describe individual vulnerabilities in products developed by F5. The structure is consistent.
- wpscan.com
  Group 5 - 803 references.
  A provider that rehashes vulnerabilities from the WordPress ecosystem and offers services related to the security of WordPress installations. For each CVE Identification number exists a short description, the structure of the page is the same throughout.
- intel.com
  Group 4 - 771 references
  Intel publishes here lists of vulnerabilities that have been fixed with an update. The structure of the pages is always identical and an assignment is possible without any problems.
- snyk.io
  Group 5 - 671 references
  Snyk offers several commercial vulnerability management products. The company maintains a public database of vulnerabilities in Open-Source-Software (OSS), respectively in open source ecosystems like Node Package Manager (npm) or Maven. The descriptions are sometimes very detailed and the structure of the contributions is always identical.

The selected web pages are referenced a total of 14 925 times. However, it is to be expected that not all references are available anymore.

Since we use novel information sources for our proposal, we also want to verify, whether information sources, which are currently regarded as high-quality information sources, *e.g.*, exploit-DB and Common Weakness Enumeration (CWE), are able to improve current models. Hence, we select their texts as well to train a *ground-truth* model, to identify, whether additional data have an actual influence in the training process.

Special Features of Twitter Twitter is an important medium in IT security and has been the subject of several works (Chen et al., 2019c; Sabottke et al., 2015). Twitter is also frequently referenced in NVD and is found among the 100 most referenced websites. However, a preliminary analysis shows that the references are unusable. In some cases, only user profiles are referenced, such as for CVE-2021-25179.[9] The reference twitter.com/gm4tr1x is the profile

[8] website: trendmicro.com/de_en/business.html.

[9] https://www.nvd.nist.gov/vuln/detail/CVE-2021-25179.

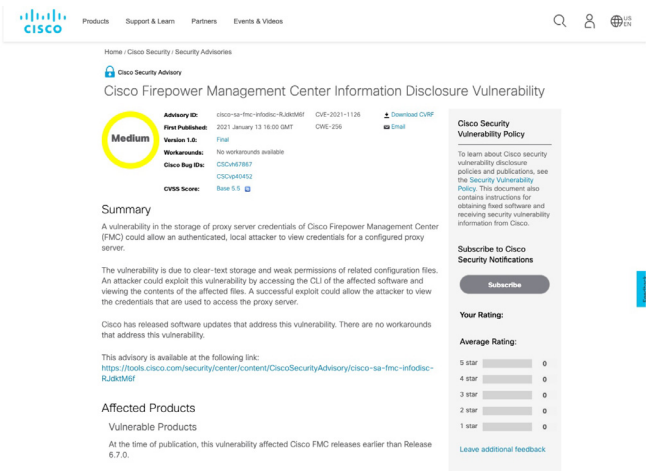

Fig. 3. Example of Cisco website, referenced in https://www.nvd.nist.gov/vuln/detail/CVE-2021-1148.

of the vulnerability's discoverer.[10] User profiles provide no meaningful information for the present work. Generally, such references are not in line with the CVE's reference requirements.[11] In our dataset 17% of references on Twitter are links to profiles. Other references are retweets, such as seen in CVE-2021-27549 [12], yielding the same problem. The original tweet is also referenced in the NVD.

Some Twitter references actually contain a description of the vulnerability. Twitter is thus very important as a medium to exchange information between experts in a short amount of time, but cannot serve well as a source for texts in this work.

4. Implementation

While the previous section (*cf.* Section 3) examined the space of available references and accompanying requirements, this section explains the process of web scraping and model training.

4.1. Web scraping

The selected domains (*cf.* Section 3.2) are publicly available, but no Application Programming Interface (API) exists to retrieve their content. So the texts have to be extracted from the pages via web scraping.

Through the robots.txt[13], the operator of a website can select which bots should access which URLs. However, this employs only a soft restriction, since it cannot be technically enforced. With the Python library `urllib`, the robots.txt of the selected domain is checked whether access to the NVD referenced in the URLs is allowed. In some cases, a delay between requests is desired due to the non-standard directive *crawl-delay*. The developed web scrapers respect this accordingly.

While trafilatura (Barbaresi, 2021) seem promising, our insights from Section 3 show, that it should be avoided in the present work. Figure 3 shows an example of the relevant part of Cisco's website. It contains the requested description as well as other texts that is present on this page. The static texts, such as headings and various legal information, are the same for each reference and represent noise. While trafilatura removes parts such as the page header, bigger chucks like the legal information are still present during text extraction.

The results are similar for *ibm.com, zerodayinitiative.com, wpscan.com, talosintelligence.com*, and *snyk.io*. Some unwanted content could still be removed by filtering the output by trafilatura, but this would require post-processing, which negates the idea of trafilatura. On some pages of *qualcomm.com* and *intel.com* multiple vulnerabilities are treated together, which introduces noise in the training process. During implementation we identified, that, *e.g.*, *qualcomm.com* changed its URL structure, so that some of the referenced URLs are unavailable. However, a manual search shows that the pages themselves are still present under other URLs.

Since Trafilatura cannot execute JavaScript, the pages of *support.f5.com* cannot be retrieved at all. This is because the server responds to initial Hyper Text Transfer Protocol (HTTP) GET requests for the referenced URLs with a JavaScript file embedded in HTML. In a browser, the script is then executed and thus the actual page content is loaded. While Trafilatura might work in other contexts, it is, in many ways, not suitable for the present work, partly due to the special requirements (*cf.* Section 3).

Several other technologies offer better controllability and in-depth filtering capabilities. Selenium[14] is a framework for automated testing of web applications and enables automatic control of full-featured web browsers in the background, *e.g.*, Google Chrome and Mozilla Firefox. Through APIs for various programming languages, including Python, the web browser can be controlled. The APIs allow access to the Document Object Model (DOM) representation of the HTML content of the accessed web page. For testing, user interaction can be simulated, such as clicks or input. Selenium thus provides everything necessary to JavaScript enriched web pages. However, it is not a lightweight and particularly fast solution.

Beautiful Soup [15] is an OSS web scraping library for Python. It allows parsing of HTML files. The user can navigate through the API structure to get selected parts of the web page. Beautiful Soup is lightweight and faster than Selenium, but is limited to HTML content. If parts of the page are reloaded using JavaScript, Beautiful Soup cannot access them accordingly.

Since the amount of references to be retrieved with the Web Scraper is limited to 14 925 and the retrieval is done only once, time plays only a minor role. Rendering the web pages with Selenium takes most of the time. The speed can be increased linearly by parallelization.

The program is structured according to the producer-consumer design pattern. First, all URLs are collected, then multiple threads are started to process the URLs in parallel. The correct web scraper is selected based on the URL.

The web scrapers for *talosintelligence.com* and *intel.com* are implemented using Beautiful Soup, and Selenium is used for the rest of the pages. The Beautiful Soup based web scrapers take about a second to retrieve and parse a web page, while Selenium based web scraper usually takes about five seconds. The web scraper first waits until the requested page is fully loaded and no more JavaScript is executed. Sometimes this leads to a blockade, because JavaScript is executed permanently. Therefore, the execution times out after 20 s. The page with the actual text is usually fully loaded by that time and can be parsed. Since such timeouts seldom occur, resulting idle times are negligible. In total, a complete run over all references in the selection took about 12 h at a measured Internet speed of about 50 MBit/s and five parallel web scrapers.

[10] The reference to the SolarWinds vendor page lists the name *Gabriele Gristina* as the discoverer. His LinkedIn and GitHub account are also referenced, in addition to the Twitter profile.

[11] https://www.cve.org/ResourcesSupport/AllResources/CNARules#section_8-3_cve_record_reference_requirements.

[12] Referencing https://www.twitter.com/0xabc0/status/1363855602477387783.

[13] https://www.robotstxt.org/.

[14] https://www.selenium.dev/.

[15] https://www.crummy.com/software/BeautifulSoup/.

Table 3
Number and proportion of each reference successfully retrieved by the web scraper from the preselection.

	References	
Webpage	Crawled	Ratio
ibm.com	2868	0.83
tools.cisco.com	3004	0.99
zerodayinitiative.com	2899	**1.0**
talosintelligence.com	1201	0.89
qualcomm.com	697	0.66
support.f5.com	740	0.79
wpscan.com	35	*0.04*
intel.com	731	0.94
snyk.io	627	0.93
Total	12,802	0.85

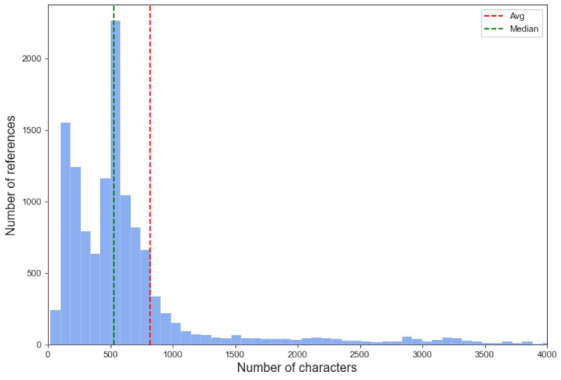

Fig. 4. Distribution of text lengths retrieved with the web scraper.

As mentioned before, some URLs for *qualcomm.com* are unavailable. Hence, the web scraper is set up to first check the NVD's reference and if it fails, start another attempt corrected URL, corresponding to the current URL scheme. This reliably fixes the URL problems for *qualcomm.com*.

Further, we use known high quality information sources, which relate their texts to the NVD's vulnerability identifier to evaluate, whether the quality of the information sources might be problematic. Here we use the CWE[16] and exploit-DB[17] as information sources. As for the CWE we add the texts of the referenced CVE id to the training set for a *quality assurance model* (*cf.* Table 6), while for exploit-DB we collected the texts for 2402 vulnerabilities, based on a listing by Mitre[18].

In total, 12 802 references (85%) of the 14 925 original ones are successfully retrieved (*cf.* Table 3). During the crawling process, we identified problems with *wpscan.com*. The domain permits all bot access in its robots.txt, but blocks all requests after five initial ones in quick succession. This means that it is not possible to retrieve many references in a meaningful way. Of the 803 references originally available, only 35 were retrieved (indicated with italic font).

Figure 4 shows the distribution of lengths of successfully retrieved texts. The average is 817.11, the median is 532 characters. The texts are between 32 and 32 206 characters long. Thus, the obtained texts are significantly longer than the descriptions from NVD (*cf.* Fig. 1).

4.2. Deep learning classifier

The goal of the work is to predict the entire CVSS basis vector, the problem is split into several sub-problems in the form of classifying the individual components of the vector. The components of the CVSS vector are Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality Impact (C), Integrity Impact (I), and Availability Impact (A)[19]. For each component there is an independent classifier. As a result, eight models must be trained separately.

4.2.1. Model selection

Shahid and Debar (2021) use a model based on BERT (Devlin et al., 2018) for their work. A classifier in the form of a fully-connected feed-forward NN is placed on top of the BERT base model in each case. Shahid and Debar (2021) use BERT-small (Turc et al., 2019), rather than the original version of BERT (Devlin et al., 2018). This model achieves a similar result in various benchmarks with significantly fewer parameters than BERT, but is faster to train. Since eight models must be trained, we adapt this idea to use one of the smaller BERT models. DistilBERT (Sanh et al., 2020) gives an even slightly better results than BERT-small (Turc et al., 2019) while also having fewer parameters than the original BERT.

For our implementation, the OSS library transformers[20] from Huggingface (Wolf et al., 2020) is used. This provides an abstraction of the actual PyTorch[21] models and provides easy access to many different pre-trained models. DistilBERT (Sanh et al., 2020), BERT-small and BERT-medium[22] (Turc et al., 2019), among others, are available via the transfomers API.

4.2.2. Training

The entire dataset is composed of descriptions from the NVD and texts retrieved from the selected domains (*cf.* Section 4.1). We crawled 88 979 NVD descriptions and 12 755 texts, for a total of 10 1734 datapoints. In the following, NVD descriptions and retrieved texts are treated identically, *i.e.*, the origin of texts is ignored. The dataset is split into a training set with 75% and a test set with 25% of the texts. This ensures that texts referring to the CVE ID are always also in the same set.

For our *quality model*, which uses exploit-DB and CWE as additional data sources, we aim to train a model on a dataset similar to the one described, the number of data points stays the same as before, but the text from exploit-DB and CWE are appended to the training set texts to enhance their quality. During manual analysis, we saw, that this might push the boundary of the 512 tokens, which can be used as input for our language models, but the latter parts of the exploid-DB texts usually contain source code, which should be ignored either way. This differs from the mentioned method of creating a data point for each single text, ignoring its origin. Our motivation behind doing this is (i) when creating a single data point for each single text, including the CWE, we might get different texts all mapping to the same CVSS value, which should worsen the model quality and (ii) this would result in different test sets, making a comparison of model quality difficult.

The training of the individual DistilBERT, BERT-small, and BERT-medium models is performed independently on the LIchtenberg high-performance computer. It provides Graphics Processing Units (GPUs) of type Nvidia Ampere 100 and Volta 100. The batch size is set based on the available GPU. Table 4 shows the possible batch size and time needed for six epochs of training including evaluation after each epoch. As one be seen, the training time does not decrease quite linearly with batch size. The speed of GPUs also plays an important role. In experiments, the training could also be performed on Nvidia T40 and K80 with 16Gb memory.

[16] https://www.cwe.mitre.org/.
[17] https://www.exploit-db.com/.
[18] https://www.cve.mitre.org/data/refs/refmap/source-EXPLOIT-DB.html.
[19] https://www.first.org/cvss/specification-document.
[20] https://www.huggingface.co/docs/transformers/index.
[21] https://www.pytorch.org/.
[22] https://www.huggingface.co/prajjwal1/bert-medium.

Table 4
Batch size and training duration of the different models (Dist. = DistilBERT, Sm. = BERT-small, Med. = BERT-medium) depending on the used GPUs.

GPU info		Batch size			Time [min]		
Model	Mem.	Dist.	Sm.	Med.	Dist.	Sm.	Med.
A100	40Gb	48	128	56	60	25	35
V100	32Gb	40	96	48	132	50	95
T40/K80	16Gb	24	64	28	-	-	-

Shahid and Debar (2021) freeze the layers of the BERT model for the first three epochs of training and only let the classifier adapt.

5. Evaluation

The previously trained models are evaluated in this section. For this purpose, different metrics for the individual classifiers are considered and compared, including white-box indicators to reconstruct the decision process of our models. Finally, we determine whether the additional texts have an impact on the overall score.

5.1. Classifier

Table 5 shows various metrics (Accuracy, Recall, Precision, F1, Cohen κ) of our classifiers. The F1 scores are arithmetic means (macro weighted), so the different distribution of target variables is not taken into account.

All models, except the Availability Impact (A) model, achieve F1-scores above 0.8 for all components. The quality of our classifiers is thus comparable to the classifiers of Shahid and Debar (2021). However, a clear improvement cannot be seen from the metrics.

For Attack Vector (AV), all models achieve very good predictions for the overrepresented values *N* and *L*. Although the values *P* and *A* occur very rarely, the classifiers still manage to correctly detect over 70%.

The classifiers for Attack Complexity (AC), Privileges Required (PR), and A work for frequent values, but are much worse for less frequent ones. While the F1 score for AC and PR is unremarkable in each case, this imbalance is evident in the lower Cohen's κ. In particular, for AC *H*, the classifiers are not reliable in this way. For A, only around 40% are correctly detected for *L*, which is the lowest rate of all classifiers.

For User Interaction (UI), Scope (S), Confidentiality Impact (C), and Integrity Impact (I), the classifiers are good to very good, with the best results for UI.

Overall, a highly uneven distribution of values in the dataset tends to lead to worse results in predicting the underrepresented values, which is a common problem with DL.

5.2. CVSS Score

To obtain the total CVSS score, the results of the classifiers of a model are combined. From the individual components, the score is calculated according to the CVSS standard.[23] The obtained scores are compared with the expert generated scores in the NVD.

Table 6 shows the Mean Absolute Error (MAE) and MSE and the fraction of vulnerabilities where the predicted score is higher, lower, or equal to the NVD's Ground Truth (GT). We also included the results of our *high-quality references* model, which was trained on NVD, CWE, and exploit-DB texts (*cf.* Section 4.1 and Section 4.2.2), as well as a comparison of our approach against the proposals by Shahid and Debar (2021); Spanos et al. (2017). In Fig. 5 the distribution of differences from true to predicted score of

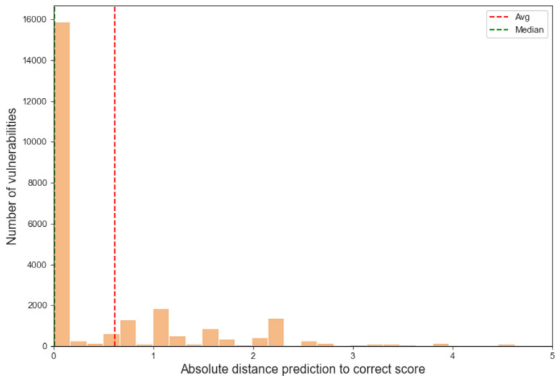

Fig. 5. Distribution of difference between the predicted score with the DistilBERT classifiers and Ground Truth (GT) Common Vulnerability Scoring System (CVSS) score.

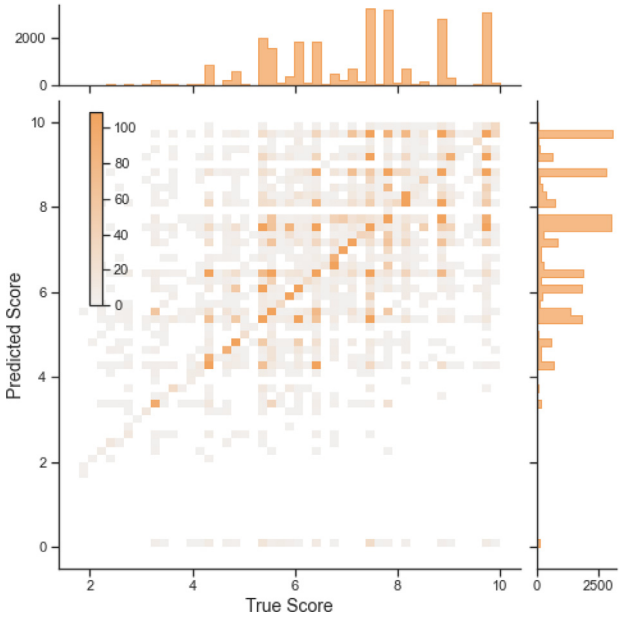

Fig. 6. Distribution of DistilBERT classifier predicted scores and the correct Common Vulnerability Scoring System (CVSS) scores.

DistilBERT classifiers is shown. The average difference is 0.6, while 75% of all predictions are within the range of 1 around the actual score.

Figure 6 shows the ratio of true to predicted scores for the DistilBERT classifiers. All models predict scores rather higher than lower compared to the original score, but the difference is very small. In general, it is better to score a vulnerability too high than too low, but depending on the use case, this can be a problem (*e.g.*, if an information overload is already present). Table 6 shows, that there is no difference, in terms of MSE and MAE, compared to the reference OSINT texts. This is either the result of a negligible influence of the concatinated text parts during the training and classification process (*cf.* Section 4.2.2), or the already high quality of other OSINT references. Either case should be inspected in future work. Another striking phenomenon is the series of predictions with a score of 0.0. These are 111 (DistilBERT), 151 (Bert-small), and 121 (Bert-medium) predictions. If *N* is predicted for each of the C, I, and A components, this makes the Impact Sub Score (ISS)[24] equal to 0. The Impact Sub Score (ISS) is multiplied by the other CVSS components, resulting in 0.0 for the total score.

[23] https://www.first.org/cvss/specification-document.

[24] https://www.first.org/cvss/specification-document.

Table 5
Accuracy (Acc), Recall (Rec), Precision (Prec), F1 Score, and Cohen's κ (Cohen) for the eight classifiers (Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality Impact (C), Integrity Impact (I)) of each model. Calculated on the test set.

	DistilBERT					BERT-small					BERT-medium				
	Acc	Rec	Prec	F1	Cohen	Acc	Rec	Prec	F1	Cohen	Acc	Rec	Prec	F1	Cohen
AV	0.93	0.82	0.85	0.84	0.84	0.91	0.82	0.82	0.82	0.81	0.93	0.83	0.85	0.84	0.82
AC	0.96	0.78	0.85	0.82	0.64	0.96	0.8	0.84	0.83	0.61	0.94	0.79	0.84	0.8	0.61
PR	0.87	0.8	0.81	0.8	0.74	0.86	0.8	0.81	0.8	0.73	0.87	0.8	0.82	0.8	0.74
UI	0.95	0.94	0.93	0.93	0.87	0.93	0.94	0.94	0.94	0.88	0.94	0.94	0.94	0.94	0.88
S	0.95	0.92	0.93	0.92	0.85	0.95	0.91	0.93	0.92	0.85	0.95	0.92	0.93	0.93	0.86
C	0.89	0.85	0.86	0.87	0.8	0.89	0.85	0.86	0.87	0.8	0.88	0.85	0.87	0.86	0.8
I	0.9	0.87	0.9	0.89	0.83	0.88	0.87	0.89	0.87	0.81	0.88	0.87	0.89	0.88	0.83
A	0.9	0.75	0.8	0.77	0.81	0.9	0.72	0.8	0.75	0.8	0.9	0.76	0.77	0.76	0.81

Table 6
Mean Squared Error (MSE), Mean Absolute Error (MAE), and proportion of correct ($Pred_c$), too high ($Pred_h$), and too low ($Pred_l$) predictions.

	MSE	MAE	$Pred_c$	$Pred_h$	$Pred_l$
DistilBERT	**1.44**	**0.61**	62.1%	20.5%	17.3%
$DistilBERT_{quality}$	1.44	0.61	62.2%	17.5%	20.2%
BERT-small	1.52	0.624	61.6%	20%	18.2%
BERT-medium	1.47	0.617	61.6%	20%	18.1%
Shahid and Debar (2021)	1.79	0.73	55.3%	-	-
Spanos et al. (2017)	-	1.74	-	-	-

There are no vulnerabilities that actually have this combination. Since there are few such predictions, the problem goes unnoticed in the metrics. However, critical vulnerabilities (with scores above 9.0) would be directly discarded because of this problem.

5.3. Explainability and interpretability

For IT security applications, it is important that ML procedures are explainable. DL models usually lack this property. It is difficult to understand what the model has learned in its entirety. However, individual examples provide some insight into the model.

The PyTorch library *Captum*[25] implements various algorithms that help explain DL models. For the following examples, we use Layer Integrated Gradients from Captum on the trained DistilBERT classifiers (Sundararajan et al., 2017). Table 7 shows example description results for the CVSS prediction of CVE-2016-0775.[26] Words that argue for the classification in each class are marked in green, words that argue against are marked in red. For clarity, only the words with the greatest influence are marked. The classifier's prediction is *L*, but *N* would have been correct.

For human experts, the phrase "[..] allows remote attackers to [..]" is a clear indication that the Attack Vector (AV) is *Network*. As Table 7 shows, "remote attackers" also argues for *N*. However, the word "file" at the end of the description is strongly scored against *N* and for *L*.

This way, a rough understanding of the representation learned by the models is gained. Many assessments are reasonable, but the DL models still remain a black box for users.

5.4. Influence of additional texts

We evaluate whether the texts retrieved via web scrapers have a positive effect compared to using the NVD's descriptions only. For this purpose, new DistilBERT models are trained on the descriptive texts from NVD only. DistilBERT was chosen because the models performed best overall in our previous evaluation (*cf.* Table 5). The models trained exclusively on NVD descriptions are called $DistilBERT_{desc}$ in the following.

Table 8 shows the results of the evaluation on different datasets. The *Combined* dataset is the previously used dataset of descriptions (2016–2021) and retrieved texts. *Desc* contains only the descriptions, but no additional reference texts. A new test dataset $Desc_{2022}$ is used, consisting of 5641 descriptions published between January and May 2022. These descriptions were not previously used for training and evaluation.

On the combined dataset, the $DistilBERT_{desc}$ classifiers achieve significantly better scores. Over 80% of the predictions were correct. We saw significant improvements for AV, AC, and A over the combined trained DistilBERT. On the NVD descriptions, the combined trained DistilBERT is significantly better than $DistilBERT_{desc}$. The additional reference texts do have a positive effect. For $Desc_{2022}$ the models are on par. $DistilBERT_{desc}$ is only slightly better here.

These results are rather unexpected, *i.e.*, the model, which is trained purely on NVD descriptions performs significantly better on the prediction of all texts (including references) and the other way around, and both models perform similar on a new dataset. The performance of the DistilBERT model trained on the combined dataset can be explained with the higher robustness, but the former cannot. The only clue might be the high quality of the NVD descriptions, but this is counterintuitive to the results of Kuehn et al. (2021).

However, since all models classify texts, which is comparable to predicting a discrete value, small differences might lead to a large difference in the score. If, for example, the models are tasked to predict the CVSS score for CVE-2022-23442[27] and it would falsely predict the C impact as *None*, the impact score would be 0 and with it the whole CVSS score would result in 0. There may be a small but crucial difference between the DistilBERT-combined and DistilBERT-descriptions classifiers for a single CVSS component, but the cause of the surprising results from Table 8 could not be determined.

6. Discussion

This section discusses the results (*cf.* Section 5) and points out future work.

Analysis of References and Web Scraping

The preliminary analysis identified sources of textual vulnerability information besides the NVD's **(RQ1)**. Hereby, we grouped sources and rated their vulnerability uniqueness, uniformity of texts, and the presence of an abstract vulnerability description (*cf.* Table 2). Due to the strict selection, only references from

[25] https://www.captum.ai/.
[26] https://www.nvd.nist.gov/vuln/detail/CVE-2016-0775.
[27] https://www.nvd.nist.gov/vuln/detail/CVE-2022-23442, with an expert rated CVSS vector of *AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N*.

Table 7
Influence of words on prediction of Attack Vector (AV) (with the parameters N - Network, L - Local, A - Adjacent, P - Physical) based on description of CVE-2016-0775. Green corresponds to a positive influence, red to a negative influence. Predicted AV: *L*. Correct AV: *N*.

AV	Text
N	Buffer over flow in the ImagingFliDecode function in libImaging/Fl iDe code.c in Pillow before 3.1.1 allows remote attackers to cause a denial of service (crash) via a crafted FLI file.
L	Buffer overflow in the Imaging FliDecode function in libImaging/ Fl iDe code .c in Pillow before 3.1.1 allows remote attackers to cause a denial of service (crash) via a crafted FL I file.
A	Buffer overflow in the ImagingFliDecode function in libImaging/Fl iDe code.c in Pillow before 3.1.1 allows remote attackers to cause a denial of service (crash) via a crafted FLI file .
P	Buffer overflow in the ImagingFliDecode function in libImaging/Fl iDe code.c in Pillow before 3.1.1 allows remote attackers to cause a denial of service (crash) via a crafted FLI file .

Table 8
Mean Squared Error (MSE) and Mean Absolute Error (MAE) on different datasets.

	Combined		Desc		$Desc_{2022}$	
	MSE	MAE	MSE	MAE	MSE	MAE
DistilBERT	1.44	0.61	0.455	0.203	1.941	0.811
$DistilBERT_{desc}$	0.544	0.248	1.393	0.604	1.928	0.788

groups 3, 4 and 5 (*cf.* Section 3.2) are eligible. However, the retrieved texts for this purpose contain almost exclusively the abstract description. Whether noise would play a major role in the texts is unclear, which could be explored for further work. While relaxing our criteria would make significantly more web pages usable, the current used language model might not be suited for such task.

Since the used DL models are optimized for natural language, log files and source code could not be used. Some of the references mix code and natural language. The currently available Natural Language Processing (NLP) models are not able to use source code in addition to natural language. Separate models for source code could be used for this in the future. With such an improvement, future work can build on our reference analysis (*cf.* Section 3.2) and try gather groups with mixed information types.

But, with the current state of research, adaptation to each website is necessary, which increases the effort linearly with the number of websites to create large datasets. Technically, there is otherwise little potential for optimizations to the implementation of web scraping. Web scrapers can be parallelized as is and used productively. Other solutions not based on manually customized web scrapers are not currently available. The need for manually adapted web scraping would also be eliminated by a uniform standard, *e.g.*, Common Security Advisory Framework (CSAF).[28]

Deep Learning Classifier Several different current DL models were successfully trained and evaluated as classifiers for the components of CVSS vectors.

The retrieved reference texts could be used as a dataset together with the descriptions. The obtained classifiers achieve good scores in several metrics (Elbaz et al., 2020; Shahid and Debar, 2021). In particular, the DistilBERT model provides good results. Therefore, the question of whether public data sources beyond databases are suitable for predicting the CVSS vector **(RQ2)** can be answered this way: Texts from OSINT sources are usable for CVSS prediction, but do not have a clear positive impact on the result in this form. It is possible to use OSINT as a textual source as a basis for CVSS prediction. Since the models require little time to train, it would also be possible to train regularly to incorporate new information into the classifier's decision. However, the expected positive effect on the quality of the models did not occur due to the additional texts (*cf.* Section 5). Even just including known high quality OSINT sources besides the NVD did not positively influence the trained models. Either, the models are not influenced by these texts anymore due to the high bias of the previous text passages, *i.e.*, the models are already saturated by the texts, or the texts used in the other training processes are of very high quality already. We omitted using additional high quality sources like ATT&CK, due to the given results (not any movement of results despite the additional texts).

Usability in Real-World Applications The results in Table 8 show, that our DistilBERT model, trained on the combined dataset, is able to predict the correct vulnerability score based on the description with a MSE of 0.46 and MAE of 0.2. This very accurate compared to other currently available methods Table 6. Whether our results are suitable to support or even replace human practitioners in cyber security depends on the use case. If the goal is to overcome current information overflow and get a first hint of the vulnerability landscape, it should be considered using a tool guided by the proposed CVSS models. The accuracy of predictions is very good and enables to get a hint of the final CVSS score. If the use case should be completely outsourcing the NVD expert knowledge to such models, we would not recommend this step. ML models can only be as good as their training data and might be biased or be prone to domain shift, *i.e.*, perform worse on new data due to drifting terminology. While the latter problem might be solved by regular retraining steps of the models, the former cannot simply be solved. One might, however, argue, that the bias can also be given by human personnel, but quality assurance steps in the pipeline of vulnerability assessment should always consider such problems, *e.g.*, by verifying scores by different personnel. The models could, however, support NVD personnel in the assessment process of new vulnerabilities and the training of new personnel. Further, should the insights of human personnel in the IT infrastructure always outweight the prediction of singular models.

Limitations & Future Work In the area of web scraping, the paper is limited by the structure of the referenced web pages (*cf.* Section 3). Future work may simplify web text collection. This could lower the effort required to adapt web scrapers to different web pages. Optimally, a solution would be as easy to use as Trafilatura, while still being able to find only the text related to a specific CVE ID. Also, a Graphical User Interface (GUI) based program could be developed that allows the selection of elements on a web page. Based on this selection, the program could then

[28] https://www.oasis-open.github.io/csaf-documentation/.

generate the necessary code for the web scraper in a selected web scraping framework.

For the CVSS classifier it needs to be investigated whether more texts lead to better results. The influence of noise should be clarified as well and, based on this, the criteria for usable text established in this work should be evaluated again.

The existing method may have potential for optimization at various points. Depending on the specific use case, all texts from references could be used for training. The results in Tables 6 and 8 suggest that there might be potential for a currently unknown combination of high quality texts and different volumes of texts, that could improve the quality of CVSS prediction models. Unrealistic scores with a score of 0 can be prevented by minor additions to the logic of the classifiers. Instead of BERT models trained on general language, models trained specifically on texts from IT security could also serve as a basis.

Data augmentation can be used to improve or compensate for the uneven distribution of different variables in the dataset (Bayer et al., 2023). Section 5.3 has shown that the decisions of the classifiers are only partially understandable. Further work can improve the explainability and interpretability of the models.

Additionally, the present work lacks a comprehensive comparison against previous work. This results either from missing published models of previous work to reconstruct the results for a fixed test-set or from the disjoint metrics, that are published. Hence, a comparison was not possible.

7. Conclusion

Vulnerabilities in IT systems pose a major threat to society, businesses, and individuals. A fast and reliable assessment of newly published vulnerabilities is therefore necessary. The increasing amount of new vulnerabilities makes timely assessment by human experts difficult. Therefore, various works (Elbaz et al., 2020; Kuehn et al., 2021; Shahid and Debar, 2021) deal with automated prediction of CVSS vector, score or level by ML. These works, however, focused on the NVD data alone, rather than using additional OSINT texts for vulnerabilities. In this work, the possibility of using OSINT as a vulnerability texts source was investigated. First, a preliminary analysis of the referenced domain in NVD vulnerability entries was performed. In this, the domains are classified into groups based on criteria according to their usability. This resulted in a pre-selection of domains which later are scraped. The reference texts and NVD descriptions, as well as selected high quality sources, were used as training set for different DL-based classifiers. Finally, the classifiers were evaluated and their quality was assessed. The classifiers achieve good results in predicting the individual components of the CVSS vectors. The CVSS scores computed from them have low error rates and real world usage of the proposed models should be considered for certain use cases.

Declaration of Competing Interest

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

CRediT authorship contribution statement

Philipp Kühn: Conceptualization, Methodology, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing. **David N. Relke:** Methodology, Data curation, Investigation, Software, Validation, Visualization, Writing – original draft. **Christian Reuter:** Funding acquisition, Project administration, Supervision, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgments

We thank all anonymous reviewers of this work. This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the German Federal Ministry for Education and Research (BMBF) in the project CYWARN (13N15407).

References

Almukaynizi, M., Nunes, E., Dharaiya, K., Senguttuvan, M., Shakarian, J., Shakarian, P., 2017. Proactive identification of exploits in the wild through vulnerability mentions online. CyCon U.S. '17.

Barbaresi, A., 2021. Trafilatura: a web scraping library and command-line tool for text discovery and extraction. ACL/IJCNLP'21.

Bayer, M., Kaufhold, M.-A., Reuter, C., 2023. Survey on Data Augmentation for Text Classification ACM Computing Surveys. (CSUR) 55 (7), 1–39. doi:10.1145/3544558.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Brian, K., 2021. A basic timeline of the exchange mass-hack – krebs on security.

Chen, H., Liu, J., Liu, R., Park, N., Subrahmanian, V., 2019. VASE: a twitter-based vulnerability analysis and score engine. ICDM '19.

Chen, H., Liu, J., Liu, R., Park, N., Subrahmanian, V.S., 2019b. VEST: a system for vulnerability exploit scoring & timing.

Chen, H., Liu, R., Park, N., Subrahmanian, V.S., 2019. Using twitter to predict when vulnerabilities will be exploited. SIGKDD '19.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. NORD'19.

Dong, Y., Guo, W., Chen, Y., Xing, X., Zhang, Y., Wang, G., 2019. Towards the detection of inconsistencies in public security vulnerability reports. USENIX Security '19.

Dwoskin, E., Adam, K., 2017. Nations Race to Contain Widespread Hacking - The Washington Post. Washington Post.

Elbaz, C., Rilling, L., Morin, C., 2020. Fighting N-day vulnerabilities with automated CVSS vector prediction at disclosure. In: Proceedings of the 15th International Conference on Availability, Reliability and Security.

Freund, Y., Schapire, R.E., 1999. A short introduction to boosting. J. Jpn. Soc. Artif.Intell. 14 (5), 771–780.

Gawron, M., Cheng, F., Meinel, C., 2018. Automatic vulnerability classification using machine learning. In: Cuppens, N., Cuppens, F., Lanet, J.-L., Legay, A., Garcia-Alfaro, J. (Eds.), Risks and Security of Internet and Systems.

Gong, X., Xing, Z., Li, X., Feng, Z., Han, Z., 2019. Joint prediction of multiple vulnerability characteristics through multi-task learning. In: 2019 24th International Conference on Engineering of Complex Computer Systems (ICECCS).

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

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput 9 (8), 1735–1780.

Jiang, Y., Atif, Y., 2021. An approach to discover and assess vulnerability severity automatically in cyber-physical systems. In: 13th International Conference on Security of Information and Networks.

Johnson, P., Lagerstrom, R., Ekstedt, M., Franke, U., 2018. Can the common vulnerability scoring system be trusted? A Bayesian analysis. IEEE Trans. Dependable Secure Comput. 15 (6), 1002–1015.

Le, T.H.M., Chen, H., Babar, M.A., 2021. A survey on data-driven software vulnerability assessment and prioritization. arXiv preprint arXiv: 2107.08364[cs].

Khazaei, A., Ghasemzadeh, M., Derhami, V., 2016. An automatic method for CVSS score prediction using vulnerabilities description. J. Intell. Fuzzy Syst. 30 (1), 89–96.

Kuehn, P., Bayer, M., Wendelborn, M., Reuter, C., 2021. OVANA: An Approach to Analyze and Improve the Information Quality of Vulnerability Databases. Proceedings of the 16th International Conference on Availability. Reliability and Security, 11. doi: 10.1145/3465481.3465744.

Liao, X., Yuan, K., Wang, X., Li, Z., Xing, L., Beyah, R., 2016. Acing the IOC game: toward automatic discovery and analysis of open-source cyber threat intelligence. In: Proceedings of the ACM Conference on Computer and Communications Security.

Liu, K., Zhou, Y., Wang, Q., Zhu, X., 2019. Vulnerability severity prediction with deep neural network. In: 2019 5th International Conference on Big Data and Information Analytics (BigDIA).

Mcauliffe, J., Blei, D., 2007. Supervised topic models. Advances in Neural Information Processing Systems.

McQuade, M., 2018. The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Wired.

Pastor-Galindo, J., Nespoli, P., Gómez Má rmol, F., Martínez Pérez, G., 2020. The not yet exploited goldmine of OSINT: opportunities, open challenges and future trends. IEEE Access 8, 10282–10304.
Riebe, T., Bäumler, J., Kaufhold, MA., et al., 2023. Values and Value Conflicts in the Context of OSINT Technologies for Cybersecurity Incident Response: A Value Sensitive Design Perspective. Comput Supported Coop Work doi:10.1007/s10606-022-09453-4.
Ruohonen, J., 2019. A look at the time delays in CVSS vulnerability scoring. Appl. Comput. Inf. 15 (2), 129–135.
Sabottke, C., Suciu, O., Dumitras, T., 2015. Vulnerability disclosure in the age of social media: exploiting twitter for predicting real-world exploits. USENIX Security '15.
Sahin, S.E., Tosun, A., 2019. A conceptual replication on predicting the severity of software vulnerabilities. In: Proceedings of the Evaluation and Assessment on Software Engineering.
Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.
Shahid, M., Debar, H., 2021. CVSS-BERT: explainable natural language processing to determine the severity of a computer security vulnerability from its description. arXiv preprint arXiv: 2111.08510.
Spanos, G., Angelis, L., Toloudis, D., 2017. Assessment of vulnerability severity using text mining. In: ACM International Conference Proceeding Series.
Sundararajan, M., Taly, A., Yan, Q., 2017. Axiomatic attribution for deep networks. ML '17.
Turc, I., Chang, M.-W., Lee, K., Toutanova, K., 2019. Well-read students learn better: on the importance of pre-training compact models.
Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A., 2020. Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.
Yamamoto, Y., Miyamoto, D., Nakayama, M., 2015. Text-mining approach for estimating vulnerability score. BADGERS '15.
Yitagesu, S., Zhang, X., Feng, Z., Li, X., Xing, Z., 2021. Automatic part-of-speech tagging for security vulnerability descriptions. MSR '21.

Philipp Kühn is a research associate and doctoral student at the chair of Science and Technology for Peace and Security (PEASEC) in the department of computer science of the Technical University of Darmstadt. He primarily researches the topics of extracting information from public data sources, with a focus on IT security, its preparation and further processing. For this purpose, he uses methods from the field of Natural Language Processing as well as Deep Learning. Furthermore, he also conducts research on topics of intergovernmental cooperation in the field of IT security.

David N. Relke was research assistant at the chair of Science and Technology for Peace and Security (PEASEC) in the department of computer science of the Technical University of Darmstadt. He studied Computer Science with a focus on IT Security and Machine Learning.

Christian Reuter is Full Professor at Technical University of Darmstadt. His chair Science and Technology for Peace and Security (PEASEC) in the Department of Computer Science combines computer science with peace and security research. He holds a Ph.D. in Information Systems (University of Siegen). On the intersection of the disciplines (A) Cyber Security and Privacy, (B) Peace and Conflict Studies as well as (C) Human-Computer Interaction, he and his team specifically address (1) Peace Informatics and technical Peace Research, (2) Crisis Informatics and Information Warfare as well as (3) Usable Safety, Security and Privacy.