Learning Word Representation for the Cyber Security Vulnerability Domain

Sara Mumtaz*, Carlos Rodriguez†, Boualem Benatallah*, Mortada Al-Banna* and Shayan Zamanirad*
*School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
† Universidad Católica Nuestra Señora de la Asunción

{s.mumtaz, b.benatallah, m.al-banna, shayan.zamanirad}@unsw.edu.au, carlos.rodriguez@uc.edu.py

***Abstract*—There have been ever-increasing amounts of security vulnerabilities discovered and reported in recent years. Much of the information related to these vulnerabilities is currently available to the public, in the form of rich, textual data (e.g. vulnerability reports). Many of the state-of-the-art techniques used today to process such textual data rely on so-called word embeddings. As of today, several pre-trained embeddings have been created, many of which rely on general-purpose training datasets such as Google News and Wikipedia. More recently, other domain-specific word embeddings have been created (e.g. in the context of software development) to cope with terminology and ambiguity limitations of existing general-purpose embeddings. The availability of word embeddings for specialised domains is critical for the effectiveness of domain-specific tasks that rely on this technique. In this paper, we propose a word embedding for the cyber security vulnerability domain. We train our embedding model on multiple, rich and heterogeneous security vulnerability information sources publicly available on the web. The benefits of such specialised word embedding are demonstrated through a qualitative comparison of word similarity and the exemplary task of matching security professionals to vulnerability discovery tasks posted to bug bounty programs. We also introduce a new dataset of words pairs similarity with a human judgement that can be used as a benchmark. Our experimental results show that, in the context of cyber security, our domain-specific word embedding outperforms existing pre-trained embeddings built on general-purpose and software engineering datasets.**

***Index Terms*—Cyber security vulnerability, word embedding, representation learning, crowdsourcing, vulnerability discovery**

I. Introduction

With the growing use of Internet-based technologies, supporting an increasingly inter-connected world, the chances of cyber-attacks and threats have also increased [40]. These may vary from minor incidents of personal data theft to major attacks contributing to taking control over entire systems. For example, in 2016, botnet *Mirai* launched a large-scale Distributed Denial of Service (DDoS) attack through infected Internet of Things (IoT) devices. This attack disabled internet access to millions of users in the US West Coast[1]. Similar events are raising the concerns of organisations regarding the protection of their assets, which, in turn, is resulting in a massive amount of security-related information being generated in the form of reports, security bulletins, advisories, standards, among other types of security vulnerability information resources.

The unstructured nature of these resources, such as the textual descriptions of vulnerabilities, opens up big opportunities as well as challenges from an information and knowledge management perspective, as well as from a Natural Language Processing (NLP) viewpoint. Here, researchers and practitioners can analyse such information and develop useful and practical applications to help improve the security of the systems employed by organisations and businesses in their day-to-day operations. Examples include detection of DDoS attacks in cyber security systems [31], intelligent malware detection [37] and classification [21], and cyberattacks event detection [19]. Apart from everyday operations, they can also benefit from the creation of automatic alerts for critical patches recently released by vendors. This affects the software infrastructure employed by an organisation. Another example includes a system that allows software developers and DevOps to find vulnerability related information (e.g. security advisories) through semantically rich Natural Language Interfaces (NLIs) [29].

In recent years, much attention has been devoted to the analysis and representation of unstructured data through neural network inspired language models (known as *embedding* models) [5], [23], [27]. The most common example is that of word embedding (WE) [23], which aims at representing words in a Vector Space Model (VSM). WEs illustrate the latent structure present in the text such that semantically similar words appear close to each other in the vector space. In other words, these word embedding models learn to generate dense, continuous, low dimensional vector representations of words from raw corpora in an unsupervised way. Several WE models have been released over the past years, many of which have been trained on large corpora such as Wikipedia[2] and GoogleNews[3]. Arguably, these embeddings have proven useful for general-purpose NLP tasks [20]. However, they have inherent problems when it comes to domain-specific tasks. Firstly, they may fall short in correctly assigning semantics to the terminology that is specific to the domain [12]. An example demonstrating this would be: The semantics of the term "*dirty cow*" in

Sara Mumtaz is the corresponding author.
[1]https://www.us-cert.gov/ncas/alerts/TA16-288A
[2]https://github.com/idio/wiki2vec
[3]https://code.google.com/archive/p/word2vec

cyber security (i.e. *"a copy-on-write security vulnerability"*) and the more widely known and colloquial meaning of the same words (*"an unclean and large ungulate mammal"*). Secondly, acronyms such as "XSS" and "CWE-79" (both used to refer to *"cross-site scripting vulnerability"*) may not always be available in general-purpose WEs. The limited ability of general-purpose WEs in terms of coping with such ambiguities ("*dirty cow*") and lack of specialised terminology ("XSS") can become a significant barrier for the effectiveness of security-related information systems that rely on such embeddings.

Our Contributions. To address the issues mentioned above, in this paper, we propose a domain-specific WE for the security vulnerability domain by utilising multiple, rich and heterogeneous sources of security vulnerability information from the web. To build this embedding, we leverage a word2vec skip-gram model [23] to capture the semantic relationship and association between words in this context. We demonstrate the suitability of our domain-specific WE in three ways. Foremost by comparing its performance against existing, pre-trained WEs through a qualitative comparison of word similarity. Secondly, we generate a word similarity dataset through crowd/human judgement for this domain to be used for the evaluation of cyber security tasks. Finally, through the task of matching security professionals (SecPros) (i.e. white hat hackers) to bug bounty programs publicly available on Hackerone[4]. The results show that our domain-specific, security vulnerability WE and word similarity dataset achieve a better performance w.r.t. their counterparts.

The rest of the paper is organized as follows. Section II describes the step-by-step procedure to training the word embedding. The experiments, evaluations and a case study collectively are presented in Section III. Related work is summarized in Section IV, and, in the end, Section V concludes the paper with future work.

II. Learning Representation for Cyber Security Vulnerabilities

Word embedding, also known as a distributed representation of words [24], is a popular way of representing words in a low-dimensional vector space. It has the property that words that share similar context (and semantics) have a close embedding in the corresponding vector space. This powerful property has been extensively employed in various domains such as software engineering [12], experts finding [26] and event type recognition [38] to develop semantic similarity techniques. In this section, we provide step-by-step details on how we leverage state-of-the-art technologies to build such WE for the cyber security vulnerability domain (shown in Fig. 1). To the best of our knowledge, this is the first work that studies the training of WEs for this domain by employing multiple, rich and heterogeneous *security vulnerability* datasets, making also the resulting WE available[5] to the public.

[4]https://www.hackerone.com

[5]https://github.com/unsw-cse-soc/Vul_Word2Vec

A. Collecting the Data

Where does this data/information come from? Information related to cyber security vulnerabilities can be collected from multiple, heterogeneous sources [29]. Such information typically come in the form of crowd-sourced vulnerability reports (e.g. Hackerone), vulnerability databases (e.g. National Vulnerability Database[6]), expert's blogs[7], security advisories[8], standards (e.g. Common Weakness Enumeration[9]), vendor's patch reports (e.g. Microsoft Security Response Center[10]), security related Q&As (e.g. Stack Exchange $Q\&As$[11]), among other sources. Given that larger amounts of training data typically produce more accurate WEs [24], we propose to collect and leverage all these vulnerability information sources to train our WE. More specifically, in this work, we use Vulners[12] as our main source of security vulnerability information. Vulners is a repository that collects vulnerability-related bulletins from more than 120 different sources (examples of such sources were listed previously). It contains more than 1.1 million vulnerability-related bulletins that represent approximately 4 GB of data. Another rich source of information is the English Wikipedia. We selected the main pages from the security category[13] and their related pages from ten levels of subcategories. A total of approximately 6,140 pages were obtained after removing duplicates. In addition, we complement this dataset with dumps of the Information Security Stack Exchange $Q\&As$, Common Weakness Enumeration (CWE) and Stack Overflow (SO) (although the latter is not strictly focused on security vulnerabilities, it contains useful security- and vulnerability-related discussions in the form of $Q\&As$).

B. Data Pre-Processing and Cleaning

The data pre-processing and cleaning step involves removing HTML tags and processing JSON documents to extract textual data (e.g. Vulners' dataset consists mainly of JSON documents). We also apply standard text pre-processing techniques such as converting the whole corpus into lowercase and tokenisation. Special characters and punctuation are also removed to retain only words. Furthermore, we filter out stop words (e.g. articles and prepositions), which typically consists of extremely common words that do not contribute much to the semantics of the key concepts and terminologies in this domain.

C. Phrases Extraction

Following data cleaning, the next step is to extract meaningful phrases [24] from the data. It is worth noting that in the domain of cyber security vulnerabilities, most of the words are in the form of phrases and have a meaning that is not simply

[6]https://nvd.nist.gov

[7]https://www.schneier.com

[8]https://www.mozilla.org/en-US/security/advisories

[9]https://cwe.mitre.org

[10]https:// portal.msrc.microsoft.com/en-us

[11]https://security.stackexchange.com

[12]https://vulners.com

[13]https://en.wikipedia.org/wiki/Category:Computer_security

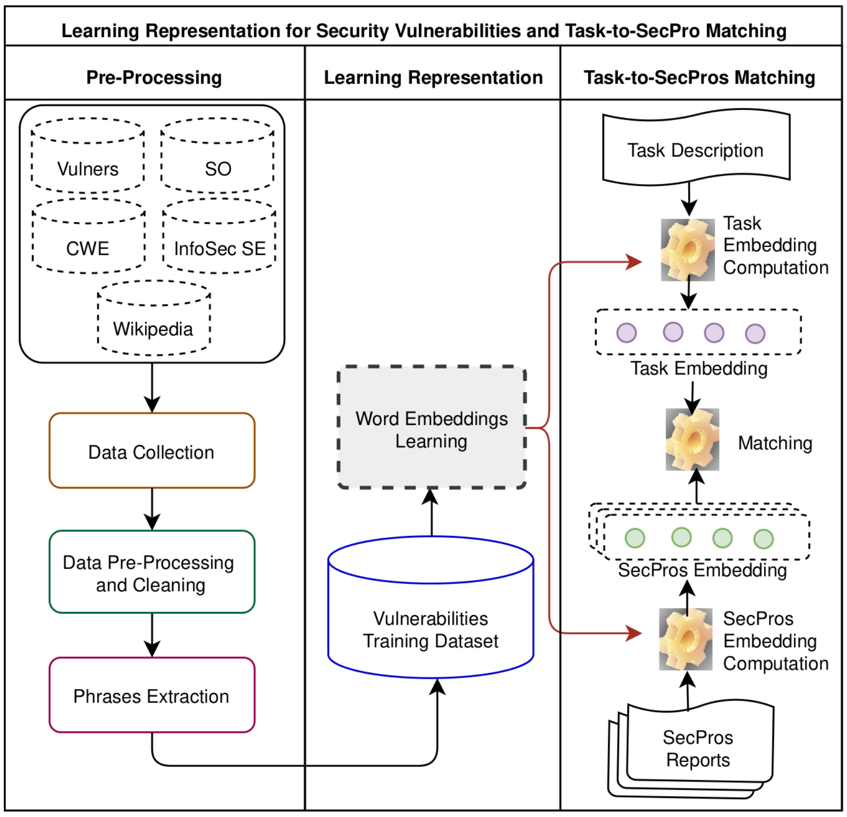

Fig. 1. Overall Process for training security vulnerability word embeddings and using them for matching vulnerability discovery tasks to SecPros.

the sum of the meanings of its constituent words. For instance, *SQL Injection* is a meaningful phrase that as a whole represent a single concept (it is a type of vulnerability). Thus, having an embedding for the whole phrase is more useful in this domain than having separate embeddings for each individual, constituent word. Such phrases can be identified by using techniques such as data-driven [24] and scoring functions [7]. In our work, we identify these phrases using a scoring function approach called Normalized Pointwise Mutual Information (NPMI) [7], in which phrases are formed based on the strength of association between words in a phrase. The association between two cyber security words i and j can be defined as follow:

$$NPMI(i,j) = \frac{log\frac{(P(i,j))}{P(i)P(j)}}{-log(P(i,j))}$$

where $P(i,j)$ refers to the joint probability of word i occurring with word j in a dataset within a fixed context windows size. Similarly, $P(i)$ and $P(j)$ indicates the probability of independent occurrences of a word i and j, respectively in the dataset. Based on this formulation, if two words co-occur more frequently together than they do independently, the score will be positive and vice versa. Hence, *SQL Injection* can be represented as a single token, i.e. *SQL_Injection*, and will be treated as an individual token during training. By extension, this process can be applied to any n-gram (e.g. 3-grams in this work) that has been observed a significant number of times (e.g. 100 times in our case). The resulting dataset consisting of approximately 1 billion unique words and phrases[14] is then represented in an embedding space as explained next.

D. Security Vulnerability Word Embeddings Learning

Our work leverages on state-of-the-art algorithms widely used in NLP communities. Several such algorithms (e.g. word2vec [23] and Glove [27]) come with efficient implementations that are readily available as libraries for popular programming languages. In this work, we adopt word2vec [23], which comes in two variants, namely, Continuous Bag-of-Words (CBOW) and Skip-Gram. More specifically, we leverage the Skip-gram model, which can be trained on rich, heterogeneous data to predict the context word w_c given a target word w_i in a certain sliding window and map words onto a vector space. More formally, given a word sequence $D = (w_1, w_2 .....w_n)$, it focuses on maximising the probability of a context word given a target word, which can be represented as follows:

$$P(w_{i-c}, w_{i-c+1}, ..., w_{i-1}, w_{i+1}, ..., w_{i+c-1}, w_{i-c}|w_i)$$

We experimented with different configurations settings to determine the hyperparameters of the model for the training

[14]Note that a phrase is considered as a word token during the training process. We therefore refer to both word and phrase as a 'word'.

process. After experimentation, we opt for a vector size of 150. This determines the dimensionality of the embedding space and produces more accurate results in terms of word vectors accuracy and similarity lookup. For medium-sized datasets (like ours), a higher dimension size can produce the same result at the cost of more resources usage and training time [23], [24]. In addition, we use a context sliding window set to 5, with the minimum frequency of words (in the dataset) of 20. Furthermore, a negative sampling of 10 helps in determining better prediction result [36], [39]. Upon several iterations of training, the neural embedding produces a set of word vectors that can support a variety of mapping functions (such as cosine similarity [22] and euclidean distance [9]). We make the word embedding (*SecVuln*) available to practitioners and researchers as a Web service (REST API), which can be accessed from `http://secvul.ap.ngrok.io/api/`.

III. Experiments

This section provides the performance evaluation of our learned representation of cyber security word and phrases through a set of experiments. We first evaluate the quality of our word embedding against state-of-the-art pre-trained embeddings using the standard words similarity task. We then describe the steps for the construction of new word similarity dataset along with its evaluation with human judgement. Finally, we present an exemplary task of security professional selection for bug bounty programs.

A. Evaluation

Words Similarity. For evaluating our word embedding as compared to existing pre-trained embeddings, we provided examples of similar words as emerging from the corresponding vector space. We randomly collected words from the cyber security domain[15] and using both our WE and the pre-trained ones, we map the collected words to vectors in the corresponding vector spaces and then obtained the nearest words based on cosine similarity [24]. Table I shows the list of top-similar words retrieved for the collected words as obtained from our WE and publicly available state-of-the-art WEs based on GoogleNews[16] and Stack Overflow [12]. The similar words returned by our WE reflect a good coherence in contrast with the other two WEs. As an example, for the input keyword *virus*, the similar words returned by our WE include *trojan* and *worm* (which are types of malware), while SO WE returns words such as *mcafee* and *avast* (which are rather products used as anti-viruses). GoogleNews WE, instead, returns more general words and outside the domain and interest of cyber security vulnerabilities (such as *flu_virus* and *bird_flu_virus*). In some cases, similar words produced by our WE conform to SO WE. This is expected given that SO WE is trained on a dataset coming from a close domain (software development) where security issues are also discussed (although to a much lesser extent w.r.t. our domain-specific datasets). Finally, the ranking and similarity scores (relevance) produced by our WE are higher than those produced by SO WE.

Dataset Construction: Word Similarity dataset in the context of Cyber Security Domain. One of the most common approaches for assessing the quality of word embeddings is to evaluate how well the similarity scores of the word vectors correlate with human judgement [33]. Several similarity datasets exist, including WordSim-353 [15] and Stanford's Contextual Word Similarity (SCWS) [17], which serve as benchmarks for analysing the performance of WEs. However, they are general-purpose datasets and may carry different word senses (e.g. Python, sniffing) in other domains [34]. Additionally, there is no such words similarity dataset available in the domain of cyber security that can help in improving the performance of NLP and information retrieval tasks in this domain [35]. This encourages us to build domain-specific, word similarity dataset based on multiple and rich sources from cyber security as a benchmark for this community. One of the main contributions of our work is thus the construction of new words similarity dataset built on the top of our embedding (*SecVuln*). The construction method will be described in detail next.

Construction Steps. What kinds of words should be included in the word similarity dataset? We use two different approaches for words selection: *Popularity* and *Importance*. The overall goal is to select a diversified list of words and phrases. (i) Popularity refers to a selection of words based on their frequencies and occurrences in a dataset. By following the settings in [17] for popular words selection, we divide them into three groups: Top 2000, between 2000 and 5000, and between 5000 and 10000 words based on their frequency in a dataset. However, unlike their approach, we do not consider *parts of speech* (e.g. verbs and nouns) for selection as we consider them irrelevant in our domain. Instead, we focus on the importance of words in the dataset: For example, in the domain of cyber security vulnerability, some of the words might not be frequent but still important (e.g. fuzzing). We use the well-known TF-IDF score [8] for computing the importance of words to create a group of important words.

We select a sample of 50 words from each of the popular three groups mentioned above alongwith 50 words from the important words group. This results in a combined list L of approximately 200 words. If a word appears more than once in L, we keep one occurrence and select the next one.

Next, we focus on the *normalization* of the selected words: We apply words lemmatization using WordNet [25] to convert the selected words into its base form (such as *exploited* to *exploit* and *viruses* to *virus*). It is worth noting that some of the technical words are not available in general-purpose WordNet [25] database, as expected, so we convert them to their original form (e.g. *pen testing* is converted to *penetration testing*) with the help of open Wikipedia search and $NIST$ standard terminologies[17]. This step is done to improve the precision and accuracy of our dataset.

[15]https://niccs.us-cert.gov/about-niccs/glossary
[16]https://news.google.com

[17]https://csrc.nist.gov/glossary?index=P

TABLE I
LIST OF TOP-5 MOST SIMILAR WORDS RETURNED BY DIFFERENT WORD EMBEDDINGS

Keyword	SecVuln WE	Stack Overflow WE	Google News WE
virus	malware, infected, antivirus, trojan, worm	malware, antivirus, spyware, mcafee, avast	flu_virus, bird_flu_virus, influenza_ virus, swine_flue_virus, infection
bot	botnet, googlebots, spam, spammer, crawler	chatbot, slackbot, telegram, wechat, botfather	botnet, robot, worm, koobface, trojan
sniffing	sniff, eavesdropping, sniffed, capturing, sniffer	sniff, spoofing, sniffers, man-in-the-middle, snooping	sniff, smelling, snuffling, chewing, sniffers
exploit	vulnerability, vulnerable, targets, bug, exploitation	exploiting, exploitable, thwart, subvert, abuse	exploiting, capitalize, exploitable, expose, utilize
metasploit	msf, metasploit_framework, msfconsole, copypasta, db_nmap	w3af, xposed, configure, module-starter, pkgsrc	*Not in vocabulary*

The next challenge consists in obtaining the second word (for each of the words previously obtained) for building word pairs for the dataset. In order to achieve this, we exploit our word embedding (*SecVuln*) and compute top k similar words for each word in the list L via cosine similarity [18]. These top k similar words serve as the second word in word pairs. For example, the top similar words for 'steal' are: (*gain_access, theft, hijack, grab, hack, impersonate, compromise*). Here, it should be noted that the same normalisation technique used before is applied the second word of the word pair. Additionally, word pairs consisting of same words with different order (e.g. $< virus, worm >$ and $< worm, virus >$) are also filtered out. After filtration, this step generated a list L' of approximately 2000-word pairs.

Validation through a Crowd. We now turn to the problem of validating the similarity dataset constructed in the previous step. Here, we focus on evaluating how well the word pairs above agree with human judgements. We validate our words pairs similarity dataset through a cyber-security-aware crowd [17]. In order to do so, we have created a crowdsourcing task and conducted a user study (in-line with UNSW Sydney's ethics approval and guideline). We used a web-based survey tool 'Qualtrics'[18] to collect participants' opinion. In this study, we ask participants (either enrolled in cyber security courses (i.e. students) or a member of cyber security groups (i.e. professionals)) to annotate our word pairs based on their similarity. For each word pair, they have to select the similarity relationship between words in a pair using a Likert scale from 0 to 3 as follows: 0 $\rightarrow$ not related at all, 1 $\rightarrow$ weakly related, 2 $\rightarrow$ related, and 3 $\rightarrow$ strongly related.

As discussed above, to ensure the quality of these annotations, we engaged only participants with cyber security knowledge and background. Moreover, considering prior work [32], which has shown that for any labelling task, given good quality contributors (as in our case: knowledgeable students), three participants opinion suffice to get reliable results. Therefore, we also target three independent participants annotations for each task. To be more precise, we divide the total number of word pairs into several tasks, where each task consists of fifty-word pairs. Thus, the 2000-word pairs result in forty tasks (i.e. 2000/50=40), which must be completed by approximately 120 participants (i.e. 40*3=120).

Upon the completion of this study, we aggregate the feedback from the participants by majority votes. For weakly related words, if two-third of participants' ratings agreed with each other, we keep that word pair; otherwise, it is discarded from the list. On average, the participants rated 35.5% as strongly related, 30.5% related, 25.8% weakly related and 8.1% are not related at all. Table II shows a sample of words pairs accepted or rejected by the participants for a given word. The resulting words similarity dataset is available to the public[19].

TABLE II
A SAMPLE OF PARTICIPANTS' RATINGS

Keyword	Accepted Words as Word Pair	Rejected Word(s)
vulnerability	bug, flaw, exploit, risk, threat	possibility, fact
snooping	eavesdropping, sniffing, mitm, spoofing	communication
backporting	patches, upstream, improvements, releases	stable, rolled

Results. This step evaluates how well the word pairs in word similarity data agree with human judgements. For our preliminary investigation, we take a sample of 200-word pairs annotated by users on a Likert Scale, as mentioned in the previous step. Following SCWS dataset evaluation [17], we also compute the Spearman's rank correlation between our word embedding similarity score and users annotations. The higher the number, the stronger the correlations. The score obtained ($r_s = 0.62$) indicates that our WE similarity score and the human ratings/score strongly correlated and well agreed with human judgement [17].

Case Study. To show the quality of our learned embeddings, we also conducted a case study for task-to-SecPros matching in crowdsourced vulnerability discovery programs (bug bounty programs). Organizations have relied on crowdsourcing vulnerability discovery for finding vulnerabilities in their software/products. One of the challenges associated with this approach is the identification of experts within the crowd in

[18]https://research.unsw.edu.au/qualtrics

[19]https://github.com/unsw-cse-soc/Vul_Word2Vec

order to invite them to participate in the crowdsourced task of vulnerability discovery [1]. In this context, given a vulnerability discovery task T, the goal is to find a set of SecPros U with appropriate required skills and knowledge to fulfill task T. Here, the aim is to derive a suitable representation of SecPros expertise and tasks in a vector space (similar to how words are represented in a WE) in order to facilitate the matching of SecPros to tasks. To do so, we leverage on textual contents (i.e. vulnerability reports submitted by SecPros) and consider it as an indicator of SecPros expertise and represent them in a vector space. The detailed description of this process is as follows (see the right-most column in Fig. 1)

Report Collection. We collect publicly available reports of each SecPro participating in *Hackerone*[20], a popular and specialized bug bounty platform. Approximately 5000 publicly available reports are collected. We group such reports by SecPros to use their contents as the basis to represent SecPros' expertise. It should be noted that SecPros with at least one accepted report are selected for experiments to filter out low quality SecPros. Thus, the resulting dataset consists of approximately 1500 SecPros.

Extraction. Next, we decompose the titles of reports into meaningful keywords $(k_1, k_2...k_n)$ with the help of a part-of-speech tagger[21].

SecPros Embedding. Following keywords extraction, the next step is to represent these keywords in an embeddings space to be used later for matching. More precisely, for the computation of the vector representation of SecPros expertise $\overrightarrow{U}$, we leverage upon our trained WE and obtain word vector v_w for each word w in a report title. Since titles are short and vary in length, we apply a simple yet powerful technique of vector averaging [11] [26] to obtain a vector representation for each report of SecPros. That is,

$$\overrightarrow{R_m} = \frac{1}{n}\sum_{i=1}^{n} e(k_i)$$

where $\overrightarrow{R_m}$ is the vector representation of SecPro u_j's report m, and n is the number of words in his/her report title. This computation is repeated for each report and a vector representation for SecPro $\overrightarrow{U}$ is obtained by averaging the reports vectors $\overrightarrow{R_m}$.

Task Embedding. For a given task T, we repeat the same representation process used for SecPros embedding. We extract keywords to represent them based on our WE, and use vector averaging to generate a vector $\overrightarrow{T}$ for a task. The keywords, in this case, are the in-scope vulnerabilities from the task description.

Similarity Computation. This vector representation allows us to use vector-based distance measure (e.g. cosine similarity) to compute the similarity between a task $\overrightarrow{T}$ and SecPros $\overrightarrow{U}$. The closer the value of similarity to 1, the higher the relevancy of a task with the SecPros' expertise. This results in a ranked list of SecPros in a decreasing order of their relevance.

[20]https://hackerone.com/hacktivity

[21]https://nlp.stanford.edu/software/tagger.shtml

Evaluation. For evaluation purposes, we use SecPros listed in HackerOne's leader-board[22] as a gold standard. The quality of the resulting list of SecPros is evaluated via precision@N [41], which measures the number of relevant SecPros that are returned by our method found with the top N results. Thus, if a SecPro belongs to our gold standard list, the result is 1. Otherwise, the result comes as 0. The results in term of finding relevant SecPros are given in Table III. The precision when using our WE is consistently better at P@5 and P@10 w.r.t. SO WE, with an improvement of 8% in terms of Mean Averaged Precision (MAP). It should be noted that we are not making any judgement on the matching algorithm itself. We rather focus on a performance comparison based on the WE employed. These findings show that our domain-specific WE has the capability to capture better the terminology and semantics used in the cyber security vulnerability domain.

TABLE III
CASE STUDY: TASK-TO-SECPROS MATCHING

Embeddings	**P@5**	**P@10**	**MAP**
SO WE	0.33	0.45	0.50
SecVuln WE	0.52	0.55	0.58

B. Model Applications

In this paper, we developed a domain specific WE targeting the cyber security vulnerability domain. Three exemplary applications of our WE include *query expansion*, *natural language interaction* and *job recommendation*.

Query Expansion. The idea of query expansion is to include additional words/terms to existing queries for improving search results. Our experiments show that similar words produced by our WE are more closely related to each other compared to other WEs. We can use these words to effectively and semantically expand queries to support the exploration of vulnerability information. This can prove very useful, especially for learning purposes. As an example, for a learner trying to find information about "malware" the query can be expanded to include "viruses", "worms", "trojans", etc. This can help learners explore further related terms to have a wholesome understanding of the security topics.

Natural Language Interaction. Interacting with database by using natural language queries through virtual assistants (a.k.a *chatbots*) is another usage of SecVul embedding model [2]. By leveraging SecVul WE, stored information (in a database) are encoded into vectors where each vector is a representation of a database element (such as a table, column, cell). Equipping with such vectorial representations, the chatbot can process users' NL queries even if they do not follow the underlying database schema [6], [14], [16]. For example, to query a database column named "patch", a user can also use alternative words such as "fix" or "update" in his NL query.

[22]https://hackerone.com/leaderboard/all-time

Job Recommendation. The demand for SecPros is rising as the incidences of security breaches increase [28]. However, the cyber security domain is currently facing a problem of "skill gap", i.e. a mismatch between the skills required by employers and the ones SecPros actually possess [10]. Effective representation of skills (e.g. based on factual data such as reports submitted to bug bounty programs) and jobs in an embedding space can help mitigate this problem. Our WE can be extended to include job descriptions and resumes of SecPros. The learned vectors and co-occurrences of skills can then be used for a job recommendation.

Beyond the positive results obtained in this work and possible application scenarios of our WE, we believe that there is still enough space for improvement. For example, there is an opportunity to improve the current WE by collecting and adding more data, including security vulnerability research papers, Github[23] repositories, Reddit[24] forums, getting started guides[25], among other sources. Another alternative is to leverage on recent embedding technologies to represent higher level concepts such as entity and relationship that can better capture the relatedness of concepts in the cyber security domain [14]. Work in this direction can potentially facilitate tasks such as data integration and knowledge graph construction within the cyber security domain.

IV. Related Work

Learning Representation for Cyber Security. The concept of learning representation from data using neural networks has been employed extensively in several machine learning tasks such as natural language processing [3]. Recently, these models have been widely adopted in cyber security applications [35] [4]. Different deep learning techniques and advanced representation models are quickly becoming popular in the cyber security domain. For example, authors in [31] proposed an artificial neural network-based technique to separate the traffic of DDoS from the real traffic in the network. Similarly, DeepAM, a deep learning framework, is presented in [37] to detect malware in network traffic intelligently. These efforts are made to help organizations and businesses in managing the security and protection of their systems. However, unlike these efforts, our work is mainly focusing on providing building blocks (i.e. word embeddings) to the above mentioned techniques. In this context, such techniques typically take pre-trained vectors of a fixed length as input. Their performance therefore rely on the underlying representation of words or textual corpus. While general pre-trained word vectors trained on Wikipedia or GoogleNews datasets are publicly available, for domain-specific tasks such as intrusion or malware detection, the accuracy of the approaches relying on them may decrease with the usage of general-purpose embeddings. A closely related work [30] from the literature trained an embedding model on sparse vulnerability-related text coming from two datasets. Unlike their work, our SecVul WE is built on multiple, heterogeneous data sources and gives a diverse and rich representation of words in the cyber security vulnerability domain.

[23]https://github.com

[24]https://www.reddit.com

[25]https://forum.bugcrowd.com/t/researcher-resources-how-to-become-a-bug-bounty-hunter/1102

Words Similarity Dataset. Measuring the similarity of reports and artifacts (e.g. detection of duplicate vulnerability discovery reports) is one of the key tasks in the cyber security vulnerability domain. There are numerous available similarity datasets annotated by humans that help in improving the performance of these similarity-based tasks. One of such datasets is the general-purpose lexical database called WordNet [25]. Word Sim353 [15] and SCWS [17] are also human labelled datasets widely used to serve the same purpose. However, when it comes to domain-specific terminologies and word representations, they may not suffice in representing the given context of the domain and may lead to ambiguity of words. It may carry different meanings in the context of cyber security and general-purpose dataset. The software engineering community has addressed this problem by introducing word similarity datasets [34] based on Stack Overflow, one of the largest software engineering question answering community. While the software engineering community dataset can work better compared to the general-purpose dataset as shown in Table I, it may not properly fit for domain-specific tasks such as the ones discussed in this paper. Our work introduces such dataset for the cyber security domain to help both researchers and practitioners overcome this limitation.

V. Conclusion

In this work, we propose a domain-specific word embedding for the cyber security vulnerability domain by utilizing textual data from multiple, heterogenous sources and leveraging on state-of-the-art Natural Language Processing embeddings technology. The experimental results indicate that the proposed word embedding outperforms the state-of-the-art, pre-trained ones in security vulnerability related tasks. Here, not only our word embedding is able to obtain a better coverage for domain-specific terminology but it also helps with mitigating the ambiguities that emerge from general purpose word embeddings. Our work can thus contribute as a key building block for more advanced techniques that can benefit from distributed representation of words in the context of cyber security vulnerabilities, including supervised learning, natural language interaction, query expansion, job recommendation, among other tasks. We make our word embedding publicly available for community reuse.

This research is part of a larger and ambitious goal of creating robust embeddings for cyber security and cyber threat intelligence artifacts to support security experts and organizations in keeping their software systems safe. Directions for future work within this line include the enrichment of existing data with Knowledge Graphs [13], the creation of a taxonomy of skills to support skills representations within this domain and the exploration of recent advances in cognitive databases [6], [16] and their possible applications to the cyber security domain.

References

[1] M. Al-Banna, B. Benatallah, D. Schlagwein, E. Bertino, and M. C. Barukh, "Friendly hackers to the rescue: How organizations perceive crowdsourced vulnerability discovery." in *PACIS*, 2018, p. 230.

[2] R. G. Athreya, A.-C. Ngonga Ngomo, and R. Usbeck, "Enhancing community interactions with data-driven chatbots–the dbpedia chatbot," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 143–146.

[3] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[4] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, 2019.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[6] R. Bordawekar, B. Bandyopadhyay, and O. Shmueli, "Cognitive database: A step towards endowing relational databases with artificial intelligence capabilities," *arXiv preprint arXiv:1712.07199*, 2017.

[7] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction," *Proceedings of GSCL*, pp. 31–40, 2009.

[8] D. M. Christopher, R. Prabhakar, and S. Hinrich, "Introduction to information retrieval," *An Introduction To Information Retrieval*, vol. 151, no. 177, p. 5, 2008.

[9] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and image processing*, vol. 14, no. 3, pp. 227–248, 1980.

[10] V. S. Dave, B. Zhang, M. Al Hasan, K. AlJadda, and M. Korayem, "A combined representation learning approach for better job and skill recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 1997–2005.

[11] C. De Boom, S. Van Canneyt, T. Demeester, and B. Dhoedt, "Representation learning for very short texts using weighted word embedding aggregation," *Pattern Recognition Letters*, vol. 80, pp. 150–156, 2016.

[12] V. Efstathiou, C. Chatzilenas, and D. Spinellis, "Word embeddings for the software engineering domain," in *Proceedings of the 15th International Conference on Mining Software Repositories*. ACM, 2018, pp. 38–41.

[13] M. Färber, B. Ell, C. Menne, and A. Rettinger, "A comparative survey of dbpedia, freebase, opencyc, wikidata, and yago," *Semantic Web Journal*, vol. 1, no. 1, pp. 1–5, 2015.

[14] R. C. Fernandez and S. Madden, "Termite: a system for tunneling through heterogeneous data," in *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, 2019, pp. 1–8.

[15] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th International Conference on World Wide Web*. New York, NY, USA: ACM, 2001, pp. 406–414.

[16] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," *arXiv preprint arXiv:2004.02349*, 2020.

[17] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 873–882.

[18] D. Jurafsky and J. H. Martin, *Speech and language processing*. Pearson London, 2014, vol. 3.

[19] R. P. Khandpur, T. Ji, S. Jan, G. Wang, C.-T. Lu, and N. Ramakrishnan, "Crowdsourcing cybersecurity: Cyber attack detection using social media," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1049–1057.

[20] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014.

[21] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2016, pp. 137–149.

[22] R. Mihalcea, C. Corley, C. Strapparava *et al.*, "Corpus-based and knowledge-based measures of text semantic similarity," in *Aaai*, vol. 6, no. 2006, 2006, pp. 775–780.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[25] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[26] S. Mumtaz, C. Rodriguez, and B. Benatallah, "Expert2vec: Experts representation in community question answering for question routing," in *International Conference on Advanced Information Systems Engineering*. Springer, 2019, pp. 213–229.

[27] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[28] L. E. Potter and G. Vickers, "What skills do you need to work in cyber security?: A look at the australian market," in *Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research*. ACM, 2015, pp. 67–72.

[29] C. Rodriguez, S. Zamanirad, R. Nouri, K. Darabal, B. Benatallah, and M. Al-Banna, "Security vulnerability information service with natural language query support," in *Proceedings of the 31st International Conference on Advanced Information Systems Engineering(CAiSE'19)*, 2019.

[30] A. Roy, Y. Park, and S. Pan, "Learning domain-specific word embeddings from sparse cybersecurity texts," *arXiv preprint arXiv:1709.07470*, 2017.

[31] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown ddos attacks using artificial neural networks," *Neurocomputing*, vol. 172, pp. 385–393, 2016.

[32] V. S. Sheng, F. Provost, and P. G. Ipeirotis, "Get another label? improving data quality and data mining using multiple, noisy labelers," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 614–622.

[33] B. Shi, W. Lam, S. Jameel, S. Schockaert, and K. P. Lai, "Jointly learning word embeddings and latent topics," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 375–384.

[34] Y. Tian, D. Lo, and J. Lawall, "Sewordsim: Software-specific word similarity database," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 568–571.

[35] M. Usman, M. A. Jan, X. He, and J. Chen, "A survey on representation learning efforts in cybersecurity domain," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–28, 2019.

[36] S. Wijeratne, L. Balasuriya, A. Sheth, and D. Doran, "Emojinet: An open service and api for emoji sense discovery," in *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[37] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "Deepam: a heterogeneous deep learning framework for intelligent malware detection," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 265–285, 2018.

[38] S. Zamanirad, B. Benatallah, M. C. Barukh, C. Rodriguez, and R. Nouri, "Dynamic event type recognition and tagging for data-driven insights in law-enforcement," *Computing*, pp. 1–25, 2020.

[39] S. Zamanirad, B. Benatallah, M. Chai Barukh, F. Casati, and C. Rodriguez, "Programming bots by synthesizing natural language expressions into api invocations," in *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2017. Piscataway, NJ, USA: IEEE Press, 2017, pp. 832–837. [Online]. Available: http://dl.acm.org/citation.cfm?id=3155562.3155665

[40] M. Zhao, J. Grossklags, and P. Liu, "An empirical study of web vulnerability discovery ecosystems," in *ACM SIGSAC 2015, Denver, CO, USA, October 12-16, 2015*, 2015, pp. 1105–1117.

[41] Z. Zhao, Q. Yang, D. Cai, X. He, and Y. Zhuang, "Expert finding for community-based question answering via ranking metric network learning," in *IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 3000–3006.