DarkEmbed: Exploit Prediction with Neural Language Models

Nazgol Tavabi
USC Information Sciences Institute
nazgolta@isi.edu

Palash Goyal
USC Information Sciences Institute
palashgo@usc.edu

Mohammed Almukaynizi
Arizona State University
malmukay@asu.edu

Paulo Shakarian
Arizona State University
shak@asu.edu

Kristina Lerman
USC Information Sciences Institute
lerman@isi.edu

Software vulnerabilities can expose computer systems to attacks by malicious actors. With the number of vulnerabilities discovered in the recent years surging, creating timely patches for every vulnerability is not always feasible. At the same time, not every vulnerability will be exploited by attackers; hence, prioritizing vulnerabilities by assessing the likelihood they will be exploited has become an important research problem. Recent works used machine learning techniques to predict exploited vulnerabilities by analyzing discussions about vulnerabilities on social media. These methods relied on traditional text processing techniques, which represent statistical features of words, but fail to capture their context. To address this challenge, we propose DarkEmbed, a neural language modeling approach that learns low dimensional distributed representations, i.e., embeddings, of darkweb/deepweb discussions to predict whether vulnerabilities will be exploited. By capturing linguistic regularities of human language, such as syntactic, semantic similarity and logic analogy, the learned embeddings are better able to classify discussions about exploited vulnerabilities than traditional text analysis methods. Evaluations demonstrate the efficacy of learned embeddings on both structured text (such as security blog posts) and unstructured text (darkweb/deepweb posts). DarkEmbed outperforms state-of-the-art approaches on the exploit prediction task with an F_1-score of 0.74.

Introduction

Vulnerabilities in software expose computer systems to attacks by cybercriminals. The consequences of an attack can be severe, as demonstrated on May 12, 2017, when Wannacry ransomware (Martin, Kinross, and Hankin 2017), exploiting a vulnerability in Microsoft Windows operating system, crippled hundreds of thousands of computer systems worldwide, including critical systems used by hospitals and others health services (Kostov, Neumann, and Woo 2017). To avoid attacks on their software, vendors need to create patches for discovered vulnerabilities. However, not every vulnerability is equally critical to patch. While a growing number of vulnerabilities are discovered each year—in the first four months of 2017 alone more than 5,000 vulnerabilities were disclosed by National Vulnerability Database (NVD)[1]—fewer than 3% of these have exploits that exist in the wild (Sabottke, Suciu, and Dumitras 2015; Allodi and Massacci 2014). Given that so few vulnerabilities are exploited, how should vendors prioritize which ones to patch? To address this problem, researchers have recently turned to machine learning techniques to analyze different data sources about vulnerabilities for clues to exploitability (Bozorgi et al. 2010; Edkrantz and Said 2015). Along this approach (Sabottke, Suciu, and Dumitras 2015) used terms appearing in Twitter posts associated with vulnerabilities, as features to train a classifier to predict which ones will be exploited. However, traditional text mining approaches fail to capture the context of the discussions, and thereby have a hard time distinguishing between potentially threatening posts and non-malicious discussions of vulnerabilities. The two posts below illustrate these differences.

- "...first advertise of this kit after several months of shutdown. rates for wm are 20/30%prices:100$/day600$/week2000$/ month...exploits:cve-2015-5122cve-2015-5119cve-2015-3043cve-2015-2419cve-2015-2445cve-2015-0311cve-2014-6332 ..."
- "...this is a really dangerous security flaw. poc of cve-2014-0476 is available lookup google linux kernel vulnerable to privilege escalation and dos attack"

The first post advertises an exploit kit for sale on a darkweb marketplace with a considerable price, which is a leading indicator of an attack. In contrast, the second post simply talks about a vulnerability. Given the words in the two posts, the second post seems more likely to be connected to a threat, but this is actually not the case. Traditional text mining methods that do not capture the context of words will fail to detect the differences. Another disadvantage is that they use sparse, high-dimensional features, which may lead to suboptimal performance in a classification task.

To address these challenges, we describe a neural language model that analyzes discussions about vulnerabilities to predict whether they will be exploited in the wild. Specifically, we use paragraph vector (Le and Mikolov 2014), an unsupervised algorithm that embeds variable-length texts in a low-dimensional vector space, to learn distributed representations of discussions on the darkweb/deepweb (*D2Web*).

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

We then train a classifier to recognize posts discussing vulnerabilities that will be exploited in the wild.

The paragraph vector is effective, because it captures the meaning of discussions and their other characteristics, such as language, and indicator words. Evaluations show that it outperforms classifiers which use word frequencies by 10% in predicting exploited vulnerabilities. The method also decreases the dimension of the feature space by 0.001 of the original values. Moreover, we show that adding other features, such as CVSS score of the vulnerability and whether it appeared in ExploitDB, improves prediction performance by 12%.

Overall, our paper makes the following contributions:

- We propose DarkEmbed, an efficient algorithm which utilizes neural language models to learn features of conversations on the D2Web.
- We use DarkEmbed to predict whether a vulnerability discussed on D2Web will be exploited.
- We extend DarkEmbed to use other features of vulnerabilities, such as CVSS score, and evaluated how much these features improve predictions.
- We use the same approach of DarkEmbed on security blog posts to detect new exploited vulnerabilities at the earliest opportunity.
- Using DarkEmbed, we identify keywords in D2Web indicative of exploit probability, i.e., words which are associated with high and low rates of exploitation.

The rest of the paper is organized as follows. We first review existing research on estimating the likelihood that a vulnerability will be exploited. We then review neural language model which learns distributed representations of text, as well as the data set we use. We evaluate the learned representations of D2Web posts on the task of exploit prediction and conclude with the discussion of the implications of the results.

Related work

A great deal of the current research on cybersecurity defense has focused on detecting emerging cyber threats. Although limited, the work on predicting cybersecurity incidents is gaining larger attention in recent years (Liu et al. 2015; Soska and Christin 2014; Hao et al. 2016; Sabottke, Suciu, and Dumitras 2015) - along the same line comes our work. Several approaches to evaluating the severity of software vulnerabilities and predicting whether they will be exploited have been pursued. The National Institute of Standards and Technology, NIST, uses Common Vulnerability Scoring System (CVSS) to assess the severity of the vulnerability (Quinn et al. 2010). This metric assigns a score to vulnerabilities, which is formulated using different characteristics, such as ease of exploit and scale of damage it may cause if exploited (Scarfone and Mell 2009). Unfortunately, this metric was proven not to be very effective, since it marks many vulnerabilities as exploitable though majority of them will never be attacked (Allodi and Massacci 2014).

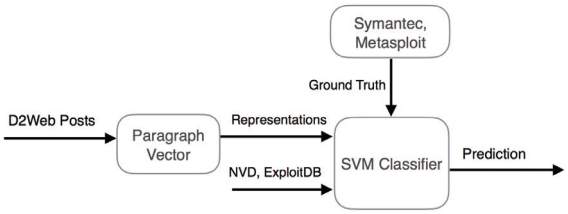

Figure 1: The framework of DarkEmbed.

This is also the shortcoming of other standard scoring systems, such as Microsoft's exploitability index [2] and Adobe Priority Rating [3].

With the ever growing number of vulnerabilities discovered and the threats they pose, different data sources have been generated and are publicly available to help enhance cybersecurity. Some previous works have combined these different data sources to get more accurate predictions (Bozorgi et al. 2010; Edkrantz and Said 2015), specially since machine learning algorithms can easily combine these different sources to achieve best results. NVD and ExploitDB are among those data sources used in different methods. *NVD*: NIST provides this database which has a list of vulnerabilities disclosed, it also contains descriptions, CVSS score and other metrics for each vulnerability. *ExploitDB* [4]: It is a repository for exploits, reported by security researchers. It provides proof of concept exploits which shows the vulnerability is exploitable but not necessary exploited. Another data source is blog posts written by cyber security experts, security analysts as well as white hat hackers, which has not been used in previous works and provides news and updated information about cyber security topics.

In other previous works it has been proposed that using discussions surrounding a vulnerability in social media like Twitter (Mittal et al. 2016) or marketplaces on the darkweb (Marin, Diab, and Shakarian 2016; Samtani et al. 2016) can help predict exploitation. Specifically, (Sabottke, Suciu, and Dumitras 2015) was able to predict exploited vulnerabilities more accurately than existing methods. However, it looks at the words surrounding that vulnerability which fails to capture semantics of the words and leads to data sparsity and high dimensionality. The approach discussed in the current paper addresses some of the problems of existing methods by using neural embeddings of discussions about vulnerabilities.

Methodology

Our framework for exploit prediction consists of two components: (1) learning embeddings of D2Web posts, and (2) exploit classifier. Figure 1 illustrates the juxtaposition of these components. We learn the distributed representations of the D2Web posts and use them as features, potentially with other features, such as CVSS score and exploitDB, in a

[2] https://technet.microsoft.com/en-us/security/cc998259.aspx
[3] https://helpx.adobe.com/security/severity-ratings.html
[4] https://www.exploit-db.com

classifier which predicts whether vulnerabilities mentioned in the posts will be exploited.

D2Web Crawling Infrastructure

To collect data, we use the infrastructure for crawling the darkweb and deepweb originally introduced in (Robertson et al. 2017; Nunes et al. 2016). In this context, *darkweb* refers to sites accessed through anonymization protocols such as Tor and I2P, while *deepweb* refers to non-indexed sites on the open Internet (Shakarian, Gunn, and Shakarian 2016). The crawling infrastructure handles sites of both types. The framework consists of an infrastructure that enables lightweight crawlers and parsers that are focused on specific sites. At the time of this writing, we have created crawlers and parsers for a manually-compiled list of over 200 sites relating to malicious hacking and/or online financial fraud, including fishing, spear-fishing, ransomware, credit card frauds, etc. This framework also helps ensure that the obtained data remains relevant to cyber-security: indeed, many darkweb and deepweb sites also create forums for other illicit activities, such as drug markets and the sale of stolen goods.

Learning Embeddings

Recent works in natural language processing popularized distributed representation learning and introduced a family of neural language models (Bengio et al. 2003; Mnih and Hinton 2007) to model sequences of words in sentences and documents. These models embed words in a fixed-dimension vector space, such that words in similar contexts tend to produce similar representations in vector space. These distributed representations of words capture many linguistic regularities of human language, such as syntactic, semantic similarity and logical analogy. We learn a context-based representation of D2Web posts in two steps. First, we learn distributed representations of words using *word embedding*. To go from distributed representations of words to distributed representations of variable-length D2Web posts, we could simply aggregate vectors of all the words contained in a post and compute their average. However, this methods does not work as well as using *paragraph embedding* to learn the global context of words in the entire post. We describe these methods below. An embedding projects words in a lower-dimensional vector space with d dimensions, so that each word w_i is represented by a d-dimensional vector v_i. Words that are used in similar contexts will be closer to one another in this vector space. While context usually implies semantic or meaning of the word, here it simply captures how the word is used within a sequence of words. For example, given two sentences—"The cat sat on the mat." and "The dog sat on the floor."—"dog" and "cat" are used in similar contexts, and thus, may be similar.

Of the many proposed models for learning distributed representations (Mikolov et al. 2013a; 2013b; Bengio et al. 2003; Collobert and Weston 2008), we use Skip-Gram with Negative Sampling (SGNS) (Mikolov et al. 2013b). The model takes as input a tokenized text corpus $C = \{w_1, w_2, \ldots, w_n\}$ and creates a context for each word w_i as $\{w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k}\}$ where k is the context length. Given the embedding of word w_i, v_i, the model aims to reconstruct the embedding of the context, $\{v_{i-k}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+k}\}$. It randomly samples "negative" examples i.e. words which do not co-occur together and maximizes (minimizes) the probability of observing positive (negative) examples from the data.

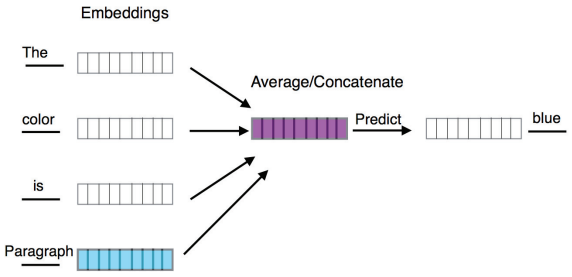

Figure 2: Framework for learning paragraph vectors.

To learn the distributed representation for the entire post, we follow the intuition of learning word embeddings. Here, instead of predicting a context for a particular word, the model samples multiple contexts from the paragraph and predicts the next word given the context (Figure 2). The context is obtained using a sliding window of length k over the paragraph. The representation is learned using stochastic gradient descent (Rumelhart, Hinton, and Williams 1988) and gradients are calculated using back propagation.

We use all the posts to learn distributed representations, since having a larger corpus helps to learn better embeddings. One of the advantages of using the paragraph vector is that it simplifies the task of handling multiple languages. Posts in different languages are embedded in the same vector space, making their comparison easier. In addition, since they may naturally fall into different clusters within this space, it is easy to identify the language of the post, which may help learn the language bias in D2Web vulnerability posts leading to more accurate exploit prediction.

Classification

We formulate exploit prediction as a classification task. Given a set of posts discussing vulnerabilities and *ground truth* containing positive examples (vulnerabilities for which exploits exist in the wild), we train a classifier to recognize posts that discuss exploited vulnerabilities. As features for the classifier, we use vectors representing post embeddings and number of times a vulnerability was mentioned in D2Web. Then, given a new post mentioning a vulnerability, the classifier decides whether that vulnerability is exploited.

For this problem Support Vector Machines (SVM) (Cortes and Vapnik 1995) with Radial basis function (RBF) kernel performs better than other examined classifiers. SVM is a supervised learning model which finds a set of hyperplanes that best separate different classes by having the largest margin. We also explored using Random Forest classifier, a combination of decision trees (Quinlan 1986), in which ran-

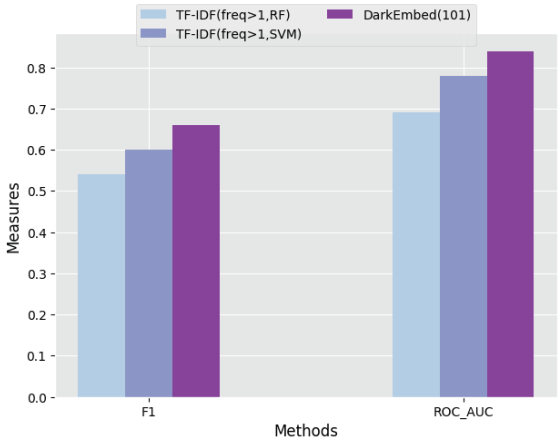

Figure 3: Comparison of the performance of classifiers for vulnerability exploit prediction.

dom selection of features are given and the final output is decided by taking a vote from individual tree predictors.

Results

We used a dataset containing almost 2,500,000 messages posted on a variety of darkweb and deepweb sites over the period from 2010 through August 2017. These posts were in 17 different languages, with English, Arabic and Russian being the most common languages. We identified vulnerabilities mentioned in D2Web posts using regular expression patterns to match CVEs, the unique identifiers of vulnerabilities. Since our goal is to predict vulnerabilities that are likely to be exploited, the posts referencing vulnerabilities after the exploitation date were removed from the data. This filtering step left 4898 posts mentioning 1886 distinct CVEs, some vulnerabilities were mentioned in more than one post. For the posts mentioning more than one vulnerability, we only considered the less frequently mentioned CVE. The ground truth was obtained from two sources: (1) *Symantec*'s anti-virus [5] and Intrusion Detection Systems [6] attack signatures and (2) a database of the exploits deployed for *Metasploit*.

Symantec attack signatures report exploits detected in the wild and their corresponding vulnerabilities, along with the time the exploit was discovered. *Metasploit* is a popular open source penetration testing framework which allows usage of install-and-test exploits developed by the cybersecurity community and a company called Rapid7[7]. Each Metasploit's exploit is reported with the date it was deployed. The vulnerabilities mentioned on D2Web were labeled positive, if they have a corresponding attack signature in Symantec's list or exploits available on Rapid7's site, and negative otherwise. Of the CVE mentioned on D2Web, only 149 are classified as *exploited* - these represent only 8% of the vulnerabilities in our dataset.

[5] https://www.symantec.com/security_response/landing/azlisting.jsp
[6] https://www.symantec.com/security_response/attacksignatures/
[7] https://www.rapid7.com/db/modules/

Exploit Prediction

We train a classifier to recognize vulnerabilities discussed in posts that will be subsequently exploited. We use F_1 score and AUC (area under the ROC-curve) to evaluate classification performance. To optimize performance, we tune parameters to the data. Most of the parameters are for learning the embeddings, including dimension of the representations, window size, the degree of negative sampling, and frequency threshold for words. Having a high dimension space gives the model the ability to better represent the posts; however, it takes more space and might lead to sparse representations. Window is the context referred to in previous sections, used for predicting the next word. Higher window sizes takes longer to train but it might be able to better capture the context. Negative sampling means randomly sampling words which do not co-occur together, and minimizing the probability of observing those words together.

Comparison to Baseline

As an alternative to word embeddings, we use TF-IDF-based representation of D2Web posts as the baseline for comparing performance. This approach is similar in spirit to existing work that predicts exploits based on online discussions of vulnerabilities (Sabottke, Suciu, and Dumitras 2015). TF-IDF approach represents posts as vectors with the same length as the vocabulary of the entire text corpus, i.e., posts. Each entry in the vector corresponds to a unique word, and its weight gives the frequency of that word in the post (TF) divided by its document frequency (IDF), i.e., the number of posts in which the word appears. Since the TF-IDF vectors can be quite large, classification methods using them would experience slow processing time and large memory usage. To reduce the size of document vectors, instead of the entire vocabulary, often a subset of the most frequent words is used to represent the documents. These document vectors are then used in the classification task. Also since TF-IDF results in high dimensional representations, random forest can usually perform better in these problems, hence we used both classifiers (SVM and Random Forest) on TF-IDF features. Figure 3 reports the performance of (1) the random forest classifier and (2) the SVM classifier on TF-IDF vectors as features. The TF-IDF vectors were constructed for words appearing more than once in the dataset (61,995 words). Finally, the figure also reports the performance of DarkEmbed using a 101 dimensional embedding space and SVM classifier. DarkEmbed outperforms baseline.

Adding Features

Post embeddings can be combined with other features of vulnerabilities to improve performance of exploit prediction. For example a binary feature indicating whether the vulnerability appears in ExploitDB, or its CVSS scores from NVD, can be used by the classifiers to improve performance. To illustrate, we combined CVSS score for each vulnerability and a binary feature for ExploitDB with D2Web post's embeddings. The added features improved classification performance from F_1 measure of 0.66 to 0.74. Figure 4 shows that incrementally adding each feature improves classifier performance.

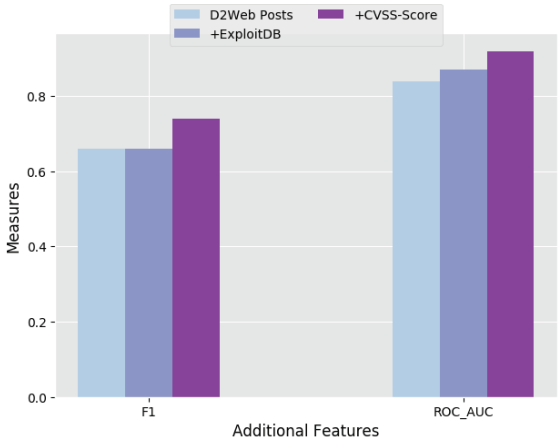

Figure 4: Performance of classifiers using additional features on the vulnerability exploit prediction task. Each new feature is added incrementally.

Table 1: Classification results on different methods

Method	F_1	AUC
Baseline methods		
TF-ID(RF)	0.54	0.69
TF-IDF(SVM)	0.60	0.78
DarkEmbed		
101 dimensions	0.66	0.84
Adding features to DarkEmbed(101 dim)		
+ExploitDB	0.66	0.87
+CVSS-Score	0.74	0.92
Using blogs to detect exploited vulnerabilities		
Blogs	0.80	0.87

Using Security Blogs

As mentioned earlier, the ground truth for this task was obtained from Symantec and Metasploit penetration tools. Although most cyber attacks are caused by a handful of vulnerabilities, which are already included in our ground truth, there are other exploited vulnerabilities that are not included in these sources. To address this gap in the ground truth, we used blogs written by cyber security experts to identify new exploited vulnerabilities. We collected blog posts from 218 cyber security experts, covering period from 2001 to 2017.

To identify exploited vulnerabilities mentioned in blogs, we applied the DarkEmbed approach to blogs by using embeddings of blogs, along with other features, to classify vulnerabilities. Here, we did not filter out posts published after exploit date as we aim to detect exploited vulnerabilities instead of predicting them. Also, we only considered posts mentioning a single vulnerability. We used embedding of size 150 (blog posts are lengthier that darkweb posts), CVSS score and number of times a vulnerability was mentioned in this dataset as features. Note that the optimal embedding size was obtained through cross validation. With 1613 blog posts in our dataset, we were able to achieve $F_1 = 0.80$ and $AUC = 0.87$.

Table 2: Software related discriminative words identified by DarkEmbed

Category	Words	# of vul.	# of exploits	% exploits
Positive	Flash	19	14	73.7%
	Adobe	21	14	66.7%
	XP	16	10	62.5%
	Microsoft	68	25	36.8%
	Windows	42	13	31.0%
Negative	iOS	4	0	0%
	Samba	7	0	0%
	Kernel	16	0	0%
	Android	30	0	0%
	Linux	38	6	15.8%

Distinctive Words

In order to better interpret DarkEmbed results, we identified key words in D2Web indicative of exploitability. Using classifications of our final classifier, D2Web posts were separated into two classes: posts mentioning exploited vulnerabilities (positive) and other posts (negative). Frequencies of words in a specific class relative to the size of the class were calculated. The words with highest difference in relative frequencies between the two classes were marked as distinctive words of that class. Since D2Web posts are in different languages many of these words were not in English.

The distinctive words identified fall into two categories: general purpose words and software related words. Some general words indicative of exploitation identified by DarkEmbed are “exploit”, “vulnerable” and “push” while those associated with low exploitation probability are “long”, “char” and “local”. Table 2 shows words related to software identified by our model to positively and negatively impact exploitability. We observe that the software detected correlate with the exploits in the wild. For example, more than 50% of the vulnerabilities of Flash, Adobe and Microsoft were exploited whereas none of vulnerabilities associated with iOS, Samba and Android were exploited.

Discussion and Future Work

Learning distributed representations of discussions on D2Web, by embedding them in a lower-dimensional space, leverages the ability of neural language models to capture fine-grained statistical relationships between words. Instead of treating each word independently, as traditional statistical approaches to text analysis do, embeddings capture deeper relationships between words that represent their meaning in context. Another advantage of this approach is its compression: for example, we used it to represent a corpus with over a million posts and tens of thousands of unique words using vectors of length 100. In contrast, traditional text approaches require large vectors to represent frequencies of all words.

We showed that post embeddings allow us to learn features of discussions about vulnerabilities on the D2Web that are useful for predicting whether the vulnerabilities will be exploited in the wild. We did this by training a classifier to use post embeddings to discriminate between discussions about exploited vulnerabilities.

Further research is needed to understand the interpretation

of features learned using distributed representations. Another direction is applying this approach to other discussion forums where malicious actors may discuss software vulnerabilities. For example, social media posts are an interesting application domain that poses challenges to text analysis due to the brevity of posts.

Acknowledgements

This work was supported by the Office of the Director of National Intelligence (ODNI) and the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL) contract number FA8750-16-C-0112. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, or the U.S. Government.

References

Allodi, L., and Massacci, F. 2014. Comparing vulnerability severity and exploits using case-control studies. *ACM Trans. Inf. Syst. Secur.* 17(1):1:1–1:20.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3:1137–1155.

Bozorgi, M.; Saul, L. K.; Savage, S.; and Voelker, G. M. 2010. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In *KDD2010*, 105–114.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 160–167.

Cortes, C., and Vapnik, V. 1995. Support-vector networks. *Mach. Learn.* 20(3):273–297.

Edkrantz, M., and Said, A. 2015. Predicting cyber vulnerability exploits with machine learning. In *SCAI*.

Hao, S.; Kantchelian, A.; Miller, B.; Paxson, V.; and Feamster, N. 2016. Predator: Proactive recognition and elimination of domain abuse at time-of-registration. In *CCS2016*, 1568–1579.

Kostov, N.; Neumann, J.; and Woo, S. 2017. Cyberattack victims begin to assess financial damage. Wall Street Journal.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In Xing, E. P., and Jebara, T., eds., *ICML*, volume 32, 1188–1196.

Liu, Y.; Sarabi, A.; Zhang, J.; Naghizadeh, P.; Karir, M.; Bailey, M.; and Liu, M. 2015. Cloudy with a chance of breach: Forecasting cyber security incidents. In *Usenix Security*.

Marin, E.; Diab, A.; and Shakarian, P. 2016. Product offerings in malicious hacker markets. In *ISI*, 187–189.

Martin, G.; Kinross, J.; and Hankin, C. 2017. Effective cybersecurity is fundamental to patient safety.

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

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mittal, S.; Das, P. K.; Mulwad, V.; Joshi, A.; and Finin, T. 2016. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *ASONAM*, 860–867.

Mnih, A., and Hinton, G. 2007. Three new graphical models for statistical language modelling. In *ICML*, 641–648.

Nunes, E.; Diab, A.; Gunn, A.; Marin, E.; Mishra, V.; Paliath, V.; Robertson, J.; Shakarian, J.; Thart, A.; and Shakarian, P. 2016. Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *ISI*, 7–12. IEEE.

Quinlan, J. R. 1986. Induction of decision trees. *Mach. Learn.* 1(1):81–106.

Quinn, S. D.; Scarfone, K. A.; Barrett, M.; and Johnson, C. S. 2010. Sp 800-117. guide to adopting and using the security content automation protocol (scap) version 1.0. Technical report, Gaithersburg, MD, United States.

Robertson, J.; Diab, A.; Marin, E.; Nunes, E.; Paliath, V.; Shakarian, J.; and Shakarian, P. 2017. *Darkweb Cyber Threat Intelligence Mining*. Cambridge University Press.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1988. Neurocomputing: Foundations of research. Cambridge, MA, USA: MIT Press. chapter Learning Representations by Back-propagating Errors, 696–699.

Sabottke, C.; Suciu, O.; and Dumitras, T. 2015. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *USENIX*, 1041–1056.

Samtani, S.; Chinn, K.; Larson, C.; and Chen, H. 2016. Azsecure hacker assets portal: Cyber threat intelligence and malware analysis. In *ISI*, 19–24.

Scarfone, K., and Mell, P. 2009. An analysis of cvss version 2 vulnerability scoring. In *SESM*, 516–525.

Shakarian, J.; Gunn, A. T.; and Shakarian, P. 2016. Exploring malicious hacker forums. In *Cyber Deception*. Springer. 261–284.

Soska, K., and Christin, N. 2014. Automatically detecting vulnerable websites before they turn malicious. In *Usenix Security*, 625–640.