NL2Vul: Natural Language to Standard Vulnerability Score for Cloud Security Posture Management

Muhammed Fatih Bulut, Jinho Hwang[§]
IBM T.J. Watson Research Center
{mfbulut, jinho}@us.ibm.com

***Abstract*—Cloud Security Posture Management (CSPM) tools have been gaining popularity to automate, monitor and visualize the security posture of multi-cloud environments. The foundation to assess the risk lies on being able to analyze each vulnerability and quantify its risk. However, the number of vulnerabilities in National Vulnerability Database (NVD) has skyrocketed in recent years and surpassed 144K as of late 2020. The current standard vulnerability tracking system relies mostly on human-driven efforts. Besides, open-source libraries do not necessarily follow the standards of vulnerability reporting set by CVE and NIST, but rather use Github issues for reporting. In this paper, we propose a framework, *NL2Vul*, to measure score of vulnerabilities with minimal human efforts. *NL2Vul* makes use of deep neural networks to train on descriptions of software vulnerabilities from NVD and predicts vulnerability scores. To flexibly expand the trained NVD model for different data sources that are being used to evaluate the risk posture in CSPM, *NL2Vul* uses transfer learning for quick re-training. We have evaluated *NL2Vul* with vanilla NVD, public Github issues of open source projects, and compliance technology specification documents.**

***Index Terms*—Cloud and software security, Vulnerability, AI**

I. Introduction

With ever increasing security spending [5], enterprises strive to adopt cloud while maintaining a secure and safe multi-cloud environment. To take advantage of the opportunity, many leading security companies have started to provide tools to better understand and manage the risk in a multi-cloud environment [1, 10]. The process itself is called Cloud Security Posture Management (CSPM).

CSPM covers end-to-end security of application from development to deployment in Cloud. CSPM tools usually concern multiple security processes, such as Vulnerability Management and Configuration Management, and allow customers to be able to monitor, visualize and take action on their cloud environments based on the actual risk. The input to quantify the risk for Vulnerability Management is the vulnerabilities themselves and the input for configuration management is the non-compliant controls, governed by the Compliance Technical Specification Documents (simply techspecs) such as CIS Benchmarks [3].

As practitioners prioritize mitigation in the CSPM tools based on the risk calculated using the existing vulnerabilities, the fundamental challenge converges on the problem to predict the scores of vulnerabilities given their descriptions in the first place. Progress has already been made with machine learning algorithms and natural language processing (NLP). The machine learning algorithms such as random forest with multi-target or multi-labeled classification techniques have achieved 68.6% accuracy for target years [20, 26]. Also, the deep neural networks (DNNs) such as convolutional neural network (CNN) with word embedding have made use of NVD data to improve prediction performance [17, 21]. However, since these approaches use only the NVD dataset as training data, applying the models to different data sources (such as techspec controls) significantly degrades the prediction performance.

In this paper, we devise a framework, *NL2Vul*, using DNNs and transfer learning to initially train on the NVD dataset and extend the model to apply for other security datasets, including techspecs which are used in enterprises for configuration management and Github issues where most open source projects use to report and discuss vulnerabilities. Our contributions include:

- *NL2Vul*, a framework to train a model with NVD datasets and extend the model using transfer learning with other smaller-sized datasets for other security problem domains,
- Investigation of well-known DNNs such as CNN, long short term memory (LSTM), transformers (such as BERT) and different state-of-the-art word embeddings and language modeling techniques for predicting the score and severity of a given vulnerability description,

The rest of the paper is organized as follows. In §II, we discuss the related work. In §III, we explain the dataset that we use for training in our experiments. In §IV, we discuss *NL2Vul* framework and the models. In §V, we demonstrate the evaluation results and finally conclude with §VI.

II. Related Work

Using machine learning techniques for vulnerability analysis has been a hot topic in literature. In [13], authors use machine learning to classify whether a given vulnerability will be exploited in future or not. In [30], authors study the use of NVD data to predict the likelihood of a piece of software to have vulnerability. In [23], authors examine the trends and patterns of software vulnerabilities in NVD. Besides, there are also various works that use machine learning to help detect the vulnerable code and components [24, 29].

Closer to our work, there are previous works in literature which try to predict the Common Vulnerability Scoring System (CVSS) metrics' values. In [26], authors use a traditional

[§] Author is now with Facebook (jinhohwang@fb.com). The work has been done while the author was with IBM.

2021 IEEE 14th International Conference on Cloud Computing (CLOUD) | 978-1-6654-0060-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/CLOUD53861.2021.00073

machine learning approach to predict the vulnerability score and CVSS metrics of a given vulnerability description. Similarly, in [20], authors propose to use a traditional machine learning approach that combines both word and character features to achieve a better performance in prediction than [26]. Both of these works try to capture the CVSS metrics in their prediction as in ours. However, our work differs in three ways. First, we use DNN based model architecture as opposed to traditional machine learning techniques given the sheer amount of available data. Second, we evaluate the different state-of-the-art embedding techniques and thoroughly investigate pros and cons of alternative approaches. Third, we extend the use of NVD data for prediction of vulnerability risk score to other domains, such as techspec documents and Github issues.

Aside from predicting the CVSS metrics, there is a body of work to predict the severity rating of vulnerability description. In [27] and [28], traditional machine learning techniques are used to predict the severity of a vulnerability. Different than these two works, DNNs are also used to predict the vulnerability severity rating. In [21], authors use DNNs to predict the severity of cross site scripting (XSS) vulnerability text data. Similarly, in [17], authors propose to use CNN along with word embedding, word2vec, to predict the severity rating. Our work differs in the following ways. First, our model architecture captures the CVSS metrics, not the severity directly, which enables NL2Vul to be able to provide finer granularity details about the vulnerability than just the severity. Second, we make use of data better as the way how the datasets are used in these papers is limited. While [21] uses a non-standard data, XSS vulnerability, the generality to the vulnerability assessment is lost. In [17], authors undersample the data in order to alleviate the lack of labeled data for some categories, hence this results in using less data for training and testing. This does not make use of full capacity of DNNs, which are known to be data hungry. Third, we extend the use of NVD data for prediction of vulnerability risk score to other domains to prove the validity and importance of the NVD data to other closely related security domains.

III. Data

National Vulnerability Database (NVD) The primary data source is from the NVD [8]. NVD data is publicly available and NIST provides REST APIs [9] to allow easy data access programmatically. Each Common Vulnerabilities and Exposures (CVE) entry in NVD contains information about the affected vendors, products, problem type based on common weakness enumeration (CWE), description and impact. Impact contains the detailed score of the vulnerability based on CVSS [4].

As of writing this paper there are more than 144*K* reported vulnerabilities in NVD, each of which has a score assigned by NVD analysts. From the severity distribution of the dataset, 9% of the vulnerabilities are categorized as *Low*, 57% as *Medium* and 34% as *High*. When we look at the statistics of description text included in NVD, one observation on CVE description is that texts are short and concise with 40 words as mean, 35 as median, and 24 as standard deviation.

Compliance Technical Specification Documents Compliance technical specification documents (simply *techspecs* or *benchmarks*) provide best practices on how to configure the systems and applications so that they are secure and in compliant with the regulatory requirements. Center for information security (CIS) [3] and security technical implementation guides (STIGs) [11] documents are examples of techspecs. IBM has a long history of providing IT services to the other companies and as a result, has a long and rich set of such documents available. There is a benchmark available for each operating system platform or applications, In each benchmark, there are multiple controls, and each control has a title and description that explain what the control is, and rationale and impact statement to explain why it is needed. Additionally, for each control we have "Security Value" column which defines the risk factor of the control.

In the experiments, we use 34 techspecs for different platforms and applications ranging from operating systems such as Linux, Windows to applications like Symantec, DB2 and Oracle DB. We have in total 2244 controls in all of these techspecs. The combined text for a control has 29 words as mean, 24 as median, and 19 as standard deviation. As it appears, techspec control descriptions are short and concise. 40% of the controls are categorized as *High*, 36% as *Medium*, and 24% as *Low* severity.

Github Issues Github issues are another type of text where vulnerability is defined. Usually when a vulnerability is first discovered, an issue is created and contributors start commenting on it. If a CVE is newly created or there is an existing CVE, contributors often mention the CVE ID in their comments. From public Github repositories, 100*K* popular repositories based on number of stars and clones have been selected and any issues or comments that mention CVE have been collected for training. 3249 issues in total have been collected. For an issue, the text has 1347 words as mean, 385 as median, and 2879 as standard deviation. Later, we postprocess the data to match the CVE IDs with their corresponding scores in NVD database to establish the ground truths. In the severity distribution of this dataset, 38% are categorized as *High*, 60% as *Medium*, and 2% as *Low*. Hence the distribution of the data is not uniform and present a challenge for machine learning to capture the nuances of each severity category.

IV. Natural Language to Vulnerability (*NL2Vul*) Architecture

Figure 1 illustrates the architecture of *NL2Vul* framework. The framework consists of four main components: input layer (Embedding), pre-trained layer (transfer learning), hidden layers and output layer. The input layer uses word embedding that captures context of a word in a vulnerability description, semantic and syntactic similarity, and relation with other words, and language modeling techniques that capture context to distinguish between words and phrases. We elaborate the

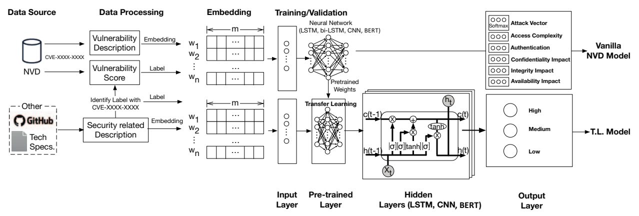

Fig. 1: *NL2Vul* Framework

input layer in §IV-A. Next, in §IV-B, the pre-trained layer follows to apply pre-trained model (i.e. weights in a separate embedding model with different dataset) to transfer learned knowledge from the base datasets. In this paper, we train the base model with NVD and this base model is used to train other datasets. The hidden layers can have different neural network architectures. We explain the details of hidden layers in §IV-C. Lastly, the output layer renders the final classification (§IV-D). The output model, either vanilla NVD model or transfer learning model, is used to predict software vulnerabilities.

A. Input Layer (Embedding)

Machine learning models only process numbers, thus one needs to convert texts into some numerical representation. Word2vec [22] is a well known embedding technique. The word embedding can represent a word with a small vector size (e.g. 300), while capturing the relationship or pattern between words that have similar meaning. Word2vec achieves this objective by training a shallow neural networks which aim to predict the other words for a given word within the same text.

NLP techniques have been improved significantly in the last few years. In particular, language modeling techniques have helped increase the accuracy of output results. Basically, the language model allows us to predict the missing word of a given sentence. The general language models such as BERT [14] are widely used to apply domain-specific problems with fine-tuning. Different from Word2vec, BERT capture the context.

In the input layer of *NL2Vul*, we experiment three different embeddings, namely *word2vec*, *word2vec-p* and *BERT*. Word2vec is trained using the training data, whereas for *word2vec-p* we use pre-trained version from Google's word2vec model [6], and this is trained on Google News, which contains 300-dimensional vector for 3 million words and phrases. For BERT, we use the pre-trained version in Tensorflow hub [2]. This particular version of BERT is trained with Wikipedia and BookCorpus [31] and has a vector size of 768. In addition to the pre-trained version of BERT, we also fine-tuned the BERT model described in the original paper [14] with the training data as explained in IV-C.

B. Pre-Trained Layer

Transfer learning is known to be an effective methodology for extending knowledge from one task to another. Specifically, it is quite effective when the tasks are in closely related domain, and there is a small dataset in one task than the other one. As shown in Figure 1, *NL2Vul* uses the learned parameters from vanilla NVD model and re-trains for a different task. *NL2Vul* removes the last layer of vanilla NVD model and adds a new output layer, and make all layers to be trainable, then starts training with the new data.

C. Hidden Layers

For text classification there are three widely adopted architectures, namely convolutional neural networks (CNN), recurrent neural networks (RNN) and transformers (e.g. BERT).

Convolutional Neural Networks (CNN): CNN has been widely used in computer vision related tasks such as object detection, facial recognition and medical image analysis. Similar efforts have been proposed for using the same in NLP tasks such as sentence classification. We have adopted the approach in [12, 19] which have shown to be powerful for text classification tasks.

Long Short Term Memory (LSTM): RNN is a popular method when dealing with NLP tasks. LSTM is one of RNN architectures [18]. LSTM enables to learn relationship in a long text without losing the information by using "forget", "input" and "output" gates. LSTM has proven to perform very well on various NLP tasks, and this motivates us to utilize LSTM. In addition, Bidirectional LSTM (LSTM-bi) connects two hidden layers of opposite directions to the same output, and this simultaneously combines information from past (backwards) and future (forward) states, and hence we also try LSTM-bi in our experiments.

BERT: Similar to RNNs, the Transformer models are also designed to handle sequential data such as text. However, in comparison to RNNs, transformers do not require text to be processed in order and this provides both speed and accuracy improvements. One such transformer model, BERT, proved to be very effective and beat the performance of other approaches in multiple domains while allowing quick fine tuning for other tasks. *NL2Vul* uses BERT architecture as is from the original paper [14], and customize the output layer as explained in IV-D to reflect the CVSS or some custom metrics.

D. Output Layer

The last layer of *NL2Vul* outputs resulting classifications, which capture different metrics in CVSS [4]. Since software vulnerability descriptions often indicate different aspects of an attack, it makes sense to capture these different aspects in the output layer. As shown in Figure 1, the output layer renders six different metrics (follows CVSS v2), and each has three labels. These metrics are used to calculate vulnerability score. For the output layer of the transfer learning model, we use a different layer (as shown in Figure 1) than the vanilla NVD model to demonstrate the feasibility of capturing different output types than CVSS v2.

V. Evaluation

A. Methodology

Base Score Metric (BM) of CVSS consists of 6 dimensions. Each of these metrics has mutually exclusive three values

TABLE I: Model performance for NVD data

Embedding	Model	Accuracy Metrics				
		MAE	**MdAE**	**F** (micro)	**F** (macro)	**F** (weig.)
Word2Vec	CNN	**0.699**	**0.0**	**0.810**	**0.764**	**0.809**
	LSTM	0.703	0.0	0.806	0.760	0.805
	LSTM-bi	0.717	0.0	0.801	0.755	0.801
Word2Vec-pre	CNN	0.728	0.0	0.799	0.750	0.798
	LSTM	0.752	0.0	0.790	0.735	0.788
	LSTM-bi	0.759	0.0	0.793	0.742	0.792
BERT	CNN	0.723	0.0	0.803	0.750	0.801
	LSTM	0.767	0.0	0.789	0.743	0.789
	LSTM-bi	0.740	0.0	0.792	0.741	0.791
	BERT	0.704	0.0	0.806	0.755	0.805

to choose from. For example, the attack vector has three choices: *local*, *adjacent network* and *network*. Based on the selection for each of these metrics, CVSS defines a formula to calculate a score from 0 to 10 and assigns a category of High, Medium or Low as severity for v2. Hence the input to our DNN model is a text (description) and the output is both numerical (CVSS score) and categorical values (High, Medium, Low). To measure the accuracy of the model in terms of the numerical value, we use mean absolute error (MAE) and median absolute error (MdAE). To evaluate the accuracy of the categorical values, we use F-score which is calculated as a combination of precision and recall [16]. Since in terms of the categories, we have a multi-class classification problem at hand, three different widely used averaging techniques are used to evaluate the F-score: *micro*, *macro* and *weighted*.

In order to evaluate the performance, three different sets are used: training, validation and test. We randomly spare the 70% of the data as training, 15% as validation, and the remaining 15% as testing. A separate validation data enables us to measure the improvements in each epoch of training. The performance results presented in this paper, are the results of the test sets, neither validation nor the training sets.

We run each experiment for a maximum of 250 epochs with a batch size of 64. *NL2Vul* monitors the improvements in validation sets and if there is no improvement that is more than 0.001 MAE for 10 consecutive epochs, *NL2Vul* assumes that the model has converged and stops training, and restores the best performing epoch weights.

B. Vanilla NVD Model Performance

Table I shows the performance of the vanilla NVD models. CNN achieves the highest score in our experiments. This is mainly because the short text sizes in NVD makes it difficult for LSTM, LSTM-bi and BERT to outperform the CNN model architecture. Among different embeddings/language modelings, we obtain the best results with word2vec, which we train using the training data of NVD. Pre-trained embeddings are trained with a corpus such as Wikipedia that is not directly related to the software vulnerability descriptions and therefore perform slightly worse. For vanilla NVD model, we achieve a great performance of MAE of 0.699, MdAE of 0 and F-score of (micro): 0.81, (macro): 0.764 and (weighted): 0.809. Both high micro, macro and weighted values indicate that the model predicts the correct class in overall, within for each different classes and weighted by each class distribution altogether. Although the results are close to each other, we also observe that BERT model (both as embedding and as a

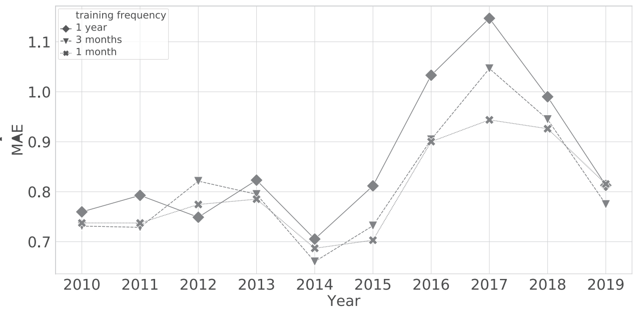

Fig. 2: MAE (0-10): 2010-19 with different training frequency

model) outperforms word2vec-pre slightly in most cases. This finding aligns with the recent state-of-the-art achievements of the BERT as a general purpose language modeling technique.

Optimal Frequency for Training As softwares change, vulnerabilities and their characteristics also change as well. In this experiment, we evaluate the yearly performance of the vanilla NVD model with multiple frequency of training; monthly, 3-monthly and yearly as data become available.

As a methodology, we look at the 10 years: from 2010 to 2019. Depending on the training frequency, we take the last period of the data as testing and the previous periods as training and validation. For example, for yearly training frequency, for each year, we use the previous years data as training (and validation), and the year we are looking as the test data. For training frequencies other than yearly, we take the mean of the performance as the representative performance for the year. In all our experiments, we use the best performing vanilla NVD model (CNN with word2vec).

Figure 2 shows the performance of vanilla NVD model in terms of MAE for different training frequencies–F-scores results have indicated a similar pattern. As shown in the figure, there is an apparent performance drop in years 2016, 2017 and 2018 for all training frequencies. However, as the training frequency periodicity is reduced to monthly, the impact of the performance drop also reduces as well. In contrasts to 3-monthly training frequency, monthly training frequency does not seem to improve the performance significantly except for the year 2017. In the light of these results, monthly training for *NL2Vul* looks reasonable. At the same time, based on the performance drop, and continuous monitoring, periodicity can be reduced to more frequent periods, to dynamically mitigate the effects as shown in Figure 2 in year 2017.

Vanilla NVD - Comparison Table II shows the comparison of *NL2Vul* with the work Le et al. [20] and Spanos et al. [26], which are the two closest articles to our work. As reported in Le et al., comparison table was already produced with Spanos et al. and we take the same reported results for comparison from the two papers. Note that the same dataset are used for the sake of comparison.

The main motivation in these two papers are the claim that new vulnerabilities demonstrate different patterns and concepts than the old vulnerabilities and therefore for evaluation the authors set the latest years from 2016 to 2018 as testing and

TABLE II: Model Performance Comparison with Le et al. [20] and Spanos et al. [26]

CVSS	CNN-Word2Vec		Max of [20] / [26]	
	Accuracy	F(macro)	Accuracy	F(macro)
Access Vector	**0.931** (+2%)	**0.591** (+9%)	0.914	0.540
Access Complexity	**0.777** (+8%)	**0.648** (+36%)	0.718	0.476
Authentication	**0.926** (+6%)	**0.546** (+24%)	0.875	0.442
Confidentiality	**0.805** (+11%)	**0.779** (+7%)	0.727	0.717
Integrity	**0.823** (+8%)	**0.797** (+6%)	0.763	0.749
Availability	**0.794** (+12%)	**0.769** (+8%)	0.712	0.711
Severity	**0.754** (+10%)	**0.700** (+22%)	0.686	0.575

use the previous years for training. As shown previously, 2016-2018 are particularly the worst performing years in the NVD dataset. Given the ability to be able to retrain the machine learning models frequently, setting particular years aside may not reflect the ideal results in a real-world scenario, where more frequent training is possible and sometimes necessary. Hence, in our experiments, different than the previous work, we continuously train the model every month as data becomes available, and test with the upcoming month while covering the same testing periods. We take the average of all monthly results as the representative performance of *NL2Vul*. We use the best performing model from vanilla NVD, which is CNN with word2vec.

As shown in Table II, *NL2Vul* outperforms Le et al. and Spanos et al.'s best results in all cases, including CVSS metrics and overall severity category in terms of both accuracy and F-score (macro)– similar results are observed for both micro and weighted F-scores. In most cases, *NL2Vul* improves the results of Le et al. and Spanos et al.'s best results, up to 36%.

C. Transfer Learning Case Studies

In this section, we use two small size datasets to demonstrate how our trained DNNs could be effective for other type of texts other than the NVD data, i.e. texts from compliance techspec documents and Github issues.

Although the common theme of both techspec documents and Github issues are security, still each poses its own characteristics. As a result directly applying the model trained with NVD data, to these domains may not work very well. Here, instead, we explore the transfer learning approach [15]. The idea in transfer learning is to use an existing model which was trained for similar tasks but has a lot more data than the task at hand. Transfer learning fits very well to our problem here, where we do not have many labeled data for both techspec documents and Github issues, however we do have some limited data. On the other hand, we have more than 144K NVD data, where we can train our vanilla NVD model.

Experiments methodology The best performing vanilla NVD model, CNN with word2vec, is selected for the experiments, then transfer learning is applied to obtain a new model (NVD-t) for the use case at hand. Similar to our vanilla NVD model experiments, we use 70% of the new data as training, 15% for validation and 15% for testing. The results of vanilla NVD model and NVD-t are compared with traditional machine learning approaches, namely Logistic Regression (adopted for multi label classification) with maximum 1000 number of iterations and Random Forest (an ensemble of decision trees) with number of trees 200.

Compliance Tech Spec Documents Table III shows the result of the experiment for compliance techspec documents. As shown in the table, transfer learning performs better than the other approaches. Particularly the vanilla NVD model performs worse and indicates the different type of the data that techspecs has from NVD descriptions.

TABLE III: F-scores for Transfer Learning

Model	Tech Spec			Github		
	micro	macro	weighted	micro	macro	weighted
NVD	0.355	0.272	0.305	0.605	0.337	0.553
NVD-t	**0.837**	**0.838**	**0.838**	**0.934**	**0.912**	**0.934**
Logistic Regression	0.781	0.778	0.781	0.902	0.887	0.903
Random Forest	0.817	0.819	0.816	0.861	0.853	0.865

Data in Wild (Github Issues) As a second use case, we apply the transfer learning methodology to a dataset derived from Github issues. The language used in Github issues is informal, meaning fairly unstructured, and shows different style of writing since non-security experts are involved in writing. How the data is collected is explained in §III in details. Since the data is very noisy, we cleaned the data by performing the followings: 1) merge the title and comments of an issue as one description of text; 2) remove any URL and HTML that appeared in text; 3) search a pattern of "CVE-*" in the text to used as a ground truth label; 4) if more than one CVE id are detected, take the highest score as the ground truth.

As shown in Table III, transfer learning outperforms the other approaches. Likewise, vanilla NVD model performs the worst for Github issues. Due to the nature of the text in Github, which is verbose and closer to vulnerability description, the transfer learning performs significantly better.

D. How NL2Vul *is Used in Practice?*

NL2Vul has been in production and used in two use cases in IBM. First to assess the vulnerability score for a given description of a vulnerability (vanilla nvd) and second to evaluate the risk (vulnerability score) for each techspec document controls (nvd-t) in order to assess the operational risk of customer's cloud environment (CSPM).

Vulnerability score estimation is used by Subject Matter Experts (SMEs) in IBM to report the vendor rating of vulnerabilities that are found in IBM and supported products. Using *NL2Vul*, SME inputs the description of the vulnerability and the CVSS score is generated automatically. Table IV shows an example vulnerability description and its associated output. As can be seen from the table, in addition to the CVSS vector with selected metrics, score, severity and explanation for each metrics are provided by *NL2Vul* for SME to be able to easily confirm the results. Explanation indicates which words are the most contributing factor for each metric. For example, *remote*, *denial* and *lead* are the top 3 words (in order) that contributes most to the decision of *Attack Vector: Network*. *NL2Vul* uses Lime [25] which regards the AI model as a black box to generate the explanation.

In addition to the vulnerability score prediction, *NL2Vul* is also employed in IBM to evaluate the risk in customer's cloud environment as a part of IBM Multicloud Management

TABLE IV: Example Vulnerability Description (*CVE-2020-28283*) and *NL2Vul* Output

Description	CVSS v2 Vector	Score	Severity	Explainability	
Prototype pollution vulnerability in 'libnested' versions 0.0.0 through 1.5.0 allows an attacker to cause a denial of service and may lead to remote code execution	V:N/AC:L/Au:N/C:P/I:P/A:P	7.5	High	AV:N	remote, denial, lead
				AC:L	execution, code, remote
				Au:N	remote, denial, lead
				C:P	code, execution, remote
				I:P	execution, remote, code
				A:P	denial, cause, vulnerability

offering [7]. This use case is identical to the use case explained in §V for compliance tech spec documents. *NL2Vul* is used as an AI-assisted tool by SMEs to map multiple CIS Benchmark controls to risk scores. Then the risk scores for uncompliant controls are aggregated to assess the risk of the cloud environment for resources (e.g. Virtual Machines), resource-groups and the overall account.

VI. Conclusion

We have described *NL2Vul* framework that uses DNNs and transfer learning to predict vulnerability score for a given textual description of the vulnerability. The framework includes input layer, pre-trained layer, hidden layer, and output layer. The base model learned from the vanilla NVD dataset is used as a pre-trained model for the transfer learning tasks. Our evaluation showed the vanilla NVD model can achieve 81% accuracy in predicting the severity of a given vulnerability description. We have also demonstrated that the transfer learning in *NL2Vul* is effective with small-sized datasets as compared to traditional machine learning approaches in two use cases: compliance techspec documents and Github issues. Our approach paved the way for analysts (such as NVD) to use *NL2Vul* to be able to quickly assess the vulnerability score, as well as CSPM tools to be able to score the risk of non-compliant techspec controls and reduce the time to value to better assess the risk in customers' cloud environments.

References

[1] Aqua cspm. https://www.aquasec.com/products/cspm/.

[2] Bert pre-trained model. https://tfhub.dev/google/bert_uncased_L-12_H-768_A-12/1.

[3] Cis benchmarks. https://www.cisecurity.org/cis-benchmarks.

[4] Cvss implementation guidance. https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7946.pdf. [ONLINE].

[5] Gartner. https://www.gartner.com/en/newsroom/press-releases/2020-06-17-gartner-forecasts-worldwide-security-and-risk-managem. [ONLINE].

[6] Google's word2vec model. https://code.google.com/archive/p/word2vec.

[7] Multi cloud management offering - risk. https://www.ibm.com/support/knowledgecenter/en/SSFC4F_2.2.0/mcm/compliance/risk_about.html. [ONLINE].

[8] National vulnerability database. https://nvd.nist.gov. [ONLINE].

[9] National vulnerability database feed. https://nvd.nist.gov/vuln/data-feeds. [ONLINE].

[10] Palo alto networks cspm. https://www.paloaltonetworks.com/prisma/cloud/cloud-security-posture-management.

[11] Security technical implementation guides. https://iase.disa.mil/stigs.

[12] Mark J. Berger. Large scale multi-label text classification with semantic word vectors. 2015.

[13] Mehran Bozorgi, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 105–114, New York, NY, USA, 2010. ACM.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[15] Chuong B Do and Andrew Y Ng. Transfer learning for text classification. In *Advances in Neural Information Processing Systems*, pages 299–306, 2006.

[16] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Proceedings of the 27th European Conference on Advances in Information Retrieval Research*, ECIR05, page 345359, Berlin, Heidelberg, 2005. Springer-Verlag.

[17] Zhuobing Han, Xiaohong Li, Zhenchang Xing, Hongtao Liu, and Zhiyong Feng. Learning to predict severity of software vulnerability using only vulnerability description. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 125–136. IEEE, 2017.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[19] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[20] Triet Huynh Minh Le, Bushra Sabir, and M Ali Babar. Automated software vulnerability assessment with concept drift. In *Proceedings of the 16th International Conference on Mining Software Repositories*, pages 371–382. IEEE Press, 2019.

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

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[23] Syed Shariyar Murtaza, Wael Khreich, Abdelwahab Hamou-Lhadj, and Ayse Basar Bener. Mining trends and patterns of software vulnerabilities. *J. Syst. Softw.*, 117(C):218–228, July 2016.

[24] Stephan Neuhaus, Thomas Zimmermann, Christian Holler, and Andreas Zeller. Predicting vulnerable software components. In *ACM Conference on computer and communications security*, pages 529–540. Citeseer, 2007.

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[26] Georgios Spanos and Lefteris Angelis. A multi-target approach to estimate software vulnerability characteristics and severity scores. *Journal of Systems and Software*, 146:152–166, 2018.

[27] Georgios Spanos, Lefteris Angelis, and Dimitrios Toloudis. Assessment of vulnerability severity using text mining. In *Proceedings of the 21st Pan-Hellenic Conference on Informatics*, page 49. ACM, 2017.

[28] Peichao Wang, Yun Zhou, Baodan Sun, and Weiming Zhang. Intelligent prediction of vulnerability severity level based on text mining and xgbboost. In *2019 Eleventh International Conference on Advanced Computational Intelligence (ICACI)*, pages 72–77. IEEE, 2019.

[29] Fabian Yamaguchi, Felix Lindner, and Konrad Rieck. Vulnerability extrapolation: Assisted discovery of vulnerabilities using machine learning. In *Proceedings of the 5th USENIX Conference on Offensive Technologies*, WOOT'11, pages 13–13, Berkeley, CA, USA, 2011. USENIX Association.

[30] Su Zhang, Doina Caragea, and Xinming Ou. An empirical study on using the national vulnerability database to predict software vulnerabilities. In *Proceedings of the 22Nd International Conference on Database and Expert Systems Applications - Volume Part I*, DEXA'11, pages 217–231, Berlin, Heidelberg, 2011. Springer-Verlag.

[31] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724, 2015.