Risk Prioritization by Leveraging Latent Vulnerability Features in a Contested Environment

Kenneth Alperin
kenneth.alperin@ll.mit.edu
MIT Lincoln Laboratory
Lexington, Massachusetts

Allan Wollaber
allan.wollaber@ll.mit.edu
MIT Lincoln Laboratory
Lexington, Massachusetts

Dennis Ross
dennis.ross@ll.mit.edu
MIT Lincoln Laboratory
Lexington, Massachusetts

Pierre Trepagnier
ptrepagnier@ll.mit.edu
MIT Lincoln Laboratory
Lexington, Massachusetts

Leslie Leonard
leslie.c.leonard@erdc.dren.mil
U.S. Army Engineer Research and Development Center
Vicksburg, Mississippi

ABSTRACT

Cyber network defenders face an overwhelming volume of software vulnerabilities. Resource limitations preclude them mitigating all but a small number of vulnerabilities on an enterprise network, so proper prioritization of defensive actions are of paramount importance. Current methods of risk prioritization are predominantly expert-based, and many include leveraging Common Vulnerability Scoring System (CVSS) risk scores. These scores are assigned by subject matter experts according to conventional methods of qualifying risk. Vulnerability mitigation strategies are then often applied in CVSS score order. Our vulnerability assessment system, in contrast, takes a predominantly data-driven approach. In general, we associate a risk metric of vulnerabilities with existence of corresponding exploits. Our assumption is that if an entity has invested time and money to exploit a particular vulnerability, this is a critical gauge of that vulnerability's importance, and hence risk.

Prior work presented a model that allows for the creation of prioritized vulnerabilities based on their association-likelihood with exploits, outperforming then-current methods. Because the initial approach only leveraged one vulnerability feature, we extended the vulnerability feature space by incorporating additional features derived from natural language processing. The importance metric is still given by a vulnerability-exploit relationship, but by processing text descriptions and other available information, our system became significantly more accurate and predictive. We next propose a mechanism that customizes vulnerability risks according to their exploitation likelihood in a contested environment given site-specific threat intelligence information, namely, attacks by an Advanced Persistent Threat (APT) group. Utilizing held-back data, we then demonstrate that latently similar vulnerabilities, which could be targeted by the same adversary, see higher risk ratings.

AISec'19, November 15, 2019, London, United Kingdom

ACM ISBN 978-1-4503-6833-9/19/11…$15.00
https://doi.org/10.1145/3338501.3357365

CCS CONCEPTS

• **Security and privacy → Vulnerability management**; • **Computing methodologies → Information extraction**; *Neural networks.*

KEYWORDS

Natural Language Processing, Vulnerability, Exploit, Risk Model, Machine Learning

ACM Reference Format:
Kenneth Alperin, Allan Wollaber, Dennis Ross, Pierre Trepagnier, and Leslie Leonard. 2019. Risk Prioritization by Leveraging Latent Vulnerability Features in a Contested Environment. In *12th ACM Workshop on Artificial Intelligence and Security (AISec'19), November 15, 2019, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3338501.3357365

1 INTRODUCTION

To detect existing software vulnerabilities, cyber network defenders often run software vulnerability scans on the networks. These vulnerabilities are enumerated with a Common Vulnerabilities and Exposures (CVE) identification number [15], and are stored in the National Vulnerability Database (NVD), which is maintained by the National Institute of Standards and Technology (NIST). Each of these CVEs is assigned a Common Vulnerability Scoring System (CVSS) base score [16]. This risk score indicates the perceived severity of a vulnerability. The values range from 0 to 10, with 10 being the highest severity. Defender priorities for vulnerability mitigation and/or patch deployment are often given in order of the highest CVSS base score. However, the CVSS risk score is fundamentally based on a static formula that aggregates the opinions of subject matter experts (SMEs) via their CVSS vector assignments and does not explicitly consider adversarial motivations or ability. While the experts have identified risk factors in vulnerabilities, these factors are rolled up into an overall risk score with a formula given by SME-determined coefficients. While it is arguably acceptable to use a CVSS score-driven prioritization in an adversary-agnostic setting, in a contested environment against a known or suspected adversary, it is not appropriate to continue that strategy.

An overreliance on adversary-agnostic SME observations leads to two immediate shortcomings of this method. First, a software vulnerability scan might discover thousands or more vulnerabilities on a network. Thus, it may be infeasible to patch even CVEs with

the highest CVSS base scores due to the time and resources required for remediation actions, especially noting that 13.5% of the NVD vulnerabilities are scored 9-10 (over 15K of the 112K indexed at the time of this writing). The second, and more fundamental limitation, is that CVEs with high CVSS base scores do not correlate well to the emergence of corresponding exploits [4]. Starting in 2016 with [27], our research on vulnerability mitigation has taken a predominantly data-driven approach to address the observed shortcomings of CVSS based remediation strategies. We associate vulnerability risk with existence of corresponding exploits, hypothesizing that if and when an entity has invested time and money to exploit a given vulnerability, this action is a critical gauge of that vulnerability's importance. We then consider that importance to be a measure of the vulnerability's risk. Identifying the existence of exploits as a risk metric allows our approach of vulnerability assessment to be data-driven, and because many catalogs of exploits exist, including ones grouped by threat, we are able to tune our assessment to particular adversary profiles.

We have so far discussed risk as a metric without formalization. This is partly due to the somewhat misleading, or at least idiosyncratic, use of the term "risk" in vulnerability literature. Risk is normally defined as (probability) x (consequence) [18]. Within this model, both probability and consequence are given as functions of time. Thus, the risk measurement can be projected as a curve in some three-dimensional space with axes of probability, consequence, and time. Discussion of the consequence of exploitation of a cyber asset is usually calculated as a separate exercise and is relegated to fields such as mission assurance, or key cyber terrain assessment; in the vulnerability assessment literature, "risk" and "probability of compromise" are essentially interchangeable. We will follow this convention, while noting that a more realistic risk metric should be obtained by associating the probability of compromise with the consequence of that compromise. We instead place the emphasis on driving down the probability of compromise as fast as possible by producing a prioritization of mitigation actions.

Given these assumptions, we show that using natural language processing features from vulnerability descriptions provides several benefits to an assessment system. The improved system significantly improves the accuracy of associating vulnerabilities to likelihood of exploitation when compared to the prior clustering of CVSS vector method. It further demonstrates an ability to tune a network's risk assessment to a specific adversary or threat actor. This tuned risk is based on the vulnerabilities and software services the actor utilizes in exploiting networks as well as similar vulnerabilities. This change is significant, as attacker models for different adversaries can be used to customize risk prioritization in a contested environment in near-real time, allowing defenders to better visualize their network from an attacker's point of view. Such fluidity allows operators to gain situational awareness on their network's vulnerabilities based on specific and/or changing threats.

2 RELATED WORK

Early work relating vulnerability features to exploitation likelihood represented an attempt to better determine the relationship between CVSS version 2 vulnerability scores to their exploitation likelihood using data from [4]. This investigation suggested that exploitation likelihood increased superlinearly [27]. However, as Allodi and Massaci observed, the fraction of vulnerabilities marked as likely to be exploited remained so high as to be impractical for cyber defenders. Worse still, only a slim minority of those vulnerabilities ever developed exploits. [21] explored latent vulnerability features in the CVSS version 2 vectors and determined that a spectral clustering approach provides superior rankings to CVSS scoring with lower false positive rates. It was further shown that this approach's accuracy would degrade versus time [20] because it still relied on a single vulnerability feature (CVSS base vector). CVSS version 3, now in more common usage, includes a temporal vector and score that makes use of an "exploit code maturity" field spanning severity of "unproven", "proof-of-concept", "functional", and "high". However, these SME-entered fields are not always current in vulnerability scans and are adversary-agnostic.

There is prior work that leverages machine learning to associate vulnerabilities to likelihood of exploitation. [14] used Common Weakness Enumerations (CWEs) from the NVD to develop association rules of vulnerabilities and discovered some co-occurring properties of vulnerabilities, but mentions future work should look into natural language processing to create association rules that would be able to infer the laws of vulnerabilities. Edkrantz et al. employed a bag of words model from the CVE descriptions, mentions of CVEs in social media and blogs, and integrated that data into a Support Vector Machine (SVM) classifier to achieve 83% accuracy on predicting links to the Exploit-DB dataset [7]. DarkEmbed is a model that learns embeddings of blog posts that discuss CVEs to predict whether vulnerabilities will be exploited, with an area under of the curve of 0.87 [24]. VULCON is a vulnerability management strategy that uses a mixed-integer multiobjective optimization algorithm to prioritize vulnerabilities for patching subject to resource constraints [8].

There is related work with respect to natural language processing of CVE descriptions. Topic analysis on CVE descriptions has been shown to help identify emerging trends [17]. Key words and phrases filtered from CVE descriptions can add the cause and effect to CVE descriptions to populate an attack graph [25]. There have been multiple efforts for using features extracted from natural language processing on CVE descriptions to predict aspects of the CVSS scoring. Supervised Latent Dirichlet Allocation (SLDA) has been implemented to estimate CVSS base metrics from natural language processing of CVE descriptions [28], and Yamamoto et al. also found that the year that a CVE is published enhances accuracy, since threats evolve over time. Spanos et al. developed a term-document matrix with machine learning classifiers on CVE descriptions to predict the CVSS severity rating (low, medium, high) of a CVE [22], and a distributional sentence model with a neural network was also shown to predict the CVSS severity rating of a CVE [10]. Extracting the Term Frequency-Inverse Document Frequency (TF-IDF) matrix from the CVE descriptions has also been shown to predict the CVSS scores themselves with 88% accuracy [12]. Altogether these efforts demonstrate that the CVE text descriptions do represent a potential signal that can be leveraged by ML classifiers.

There has been some work into exploring the behavior of different attackers, but no work to our knowledge has considered using attacker modeling in concert with machine learning for the

purposes of vulnerability or software service management. Different machine learning methods have been analyzed on predicting vulnerability likelihood of exploitation from Twitter and NVD data, and exploring the attacker and defender strategy co-evolution has been mentioned as future work in this space [6]. Allodi and Etalle develop an attacker threat model and provide evidence that strongly suggests that attackers do not arbitrarily choose vulnerabilities to exploit amongst all available options, and they propose a model that categorizes attackers according to their resources [3]. Allodi et al. also introduced a work-averse attacker model and provided evidence that adversaries will likely continue to exploit vulnerabilities against software services for which they have previously developed capabilities [5]. The STIX-Analyzer framework defines an ontology for CVEs and STIX and maps threats to the ontology with a rules-based system, but does not prioritize risk for vulnerabilities [19]. Multi-level machine learning models trained on network data have been shown to predict attacker behavior, but they do not focus on vulnerabilities [26]. Jacobs et al. also develop machine learning models for predicting coverage and efficiency in patching strategy and associating vulnerabilities with exploits in general, but without an emphasis on particular association with attackers [11].

3 HYPOTHESIS

We first hypothesize that including additional features beyond CVSS vector alone will more accurately associate vulnerabilities with risk of exploitation; this has already proven successful in related works [6, 11, 28]. These extra features incorporate natural language processing on the CVE descriptions. This approach was determined to be reasonable because it potentially converts a descriptive human assessment of a particular vulnerability to numeric (vector) values that distinguish it in the context of other vulnerability descriptions. Specifically, we use Latent Semantic Analysis (LSA) to discover latently similar "topics" within vulnerability descriptions in an unsupervised learning setting. The resulting vectors can then be processed via supervised machine learning, resulting in a more accurate association of vulnerabilities to exploits than CVSS base scores.

We propose the additional hypothesis that recalculating risk scores for vulnerabilities based on different sets of threat profiles should lead to a more accurate prioritization of vulnerabilities in the circumstance that historical information or threat intelligence indicates that particular adversaries or threat-groups may target a defended network of interest. The results of this approach would allow a cyber defender to tune a risk prioritization to better prepare for or defend against adversaries on a contested network.

We incorporate both of these hypotheses into a prototype threat assessment system. By expanding the vulnerability feature space with the latent semantic analysis vector, threat tuning and attacker modeling are enabled by analyzing connections between threat actors and CVE descriptions. These associations and improved accuracy demonstrate a novel capability for cyber defenders.

4 METHODOLOGY

We begin by considering the vulnerabilities enumerated in the National Vulnerability Database (NVD, https://nvd.nist.gov). We include all CVE entries available from 1999 through February 4, 2019. Each of the CVEs in the NVD include a text description of their corresponding vulnerability. These entries are regular and can be parsed into usable feature data. A small percentage of CVEs are disputed or rejected after first being published; these problematic CVEs are easily filtered out. The resulting dataset contains 112, 503 CVEs.

Several standard natural language processing problems were present when considering the CVE descriptions. We addressed each of them through our choices of algorithms and techniques when examining the CVE dataset. Each CVE has a short human-language description with high variance. The average descriptions length is 40 ± 22 words totaling 279 ± 168 characters. Histograms of the distributions of the word and character lengths in the descriptions are shown in Figures 1 and 2.

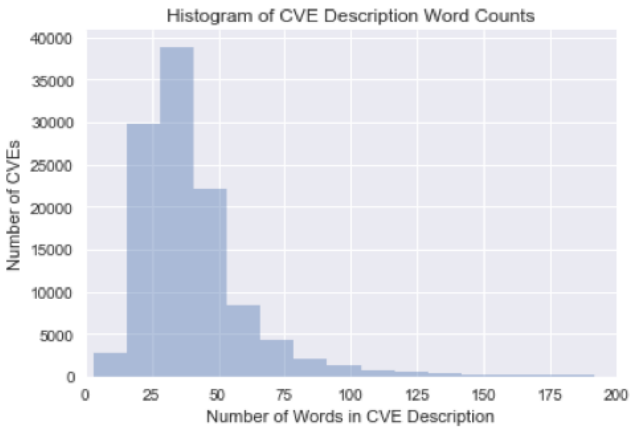

Figure 1: Distribution of CVE Word Count

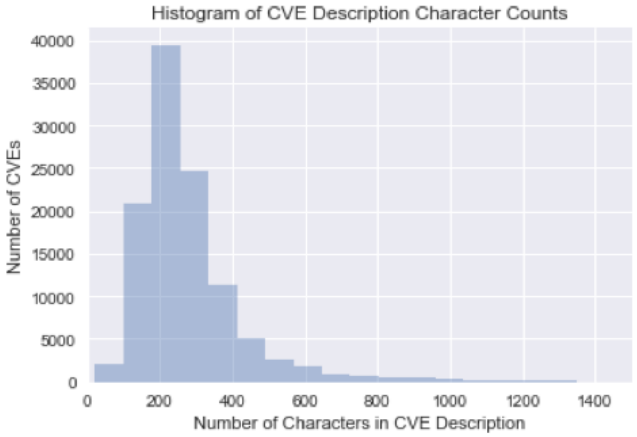

Figure 2: Distribution of CVE Character Count

There are not many words and characters in these CVEs, as the average number of characters in a CVE is just about the same as the maximum number of characters now allowed in a Twitter post. The CVE descriptions often have cyber-specific terminology,

vendor/product names and versions, and specific filenames, so those all need to be accounted for and parsed properly. There are also sometimes errors and spacing issues (e.g., no spaces to start a new sentence). An illustrative description is shown below for CVE-2017-6517:

> CVE-2017-6517 (3/23/17): Microsoft Skype 7.16.0.102 contains a vulnerability that could allow an unauthenticated, remote attacker to execute arbitrary code on the targeted system. This vulnerability exists due to the way .dll files are loaded by Skype. It allows an attacker to load a .dll of the attacker's choosing that could execute arbitrary code without the user's knowledge.The specific flaw exists within the handling of DLL (api-ms-win-core-winrt-string-l1-1-0.dll) loading by the Skype.exe process.

Figure 3 depicts a pipeline of natural language processing techniques to take the raw CVE descriptions from the NVD and generate different features from them for use in a binary classification for associating CVEs and exploits. This pipeline helped us to evaluate feature efficacy and exploration from different pipeline stages and after different algorithms. The pipeline starts from the raw descriptions, which is then converted into a corpus via the removal of questioned CVEs as described above. The next step in this pipeline is to perform data wrangling to clean and regularize it for natural language processing algorithms. Before cleaning, there are 227, 915 terms that show up at least once in one of the CVE descriptions. All stopwords are removed from the corpus; stopwords are very common terms such as "the" and "is" that can be filtered out of text before using natural language processing techniques. Regular expressions were used to replace any filename with the string "filename". Punctuation and numbers were removed as well. All of the remaining terms in each description were tokenized to identify terms without whitespace, and terms were also stemmed using a Porter Stemmer (stemming is the process of combining terms that share the same root or are the same verb with different tense, such as "load" and "loaded"). After this cleaning, there are 93, 866 unique terms in the corpus, representing 41.2% of the original terms in the CVE descriptions.

4.1 Natural Language Processing of CVE Descriptions

These techniques significantly reduced the number of terms in the CVE description corpus. The cleaned corpus of CVE descriptions was then transformed using TF-IDF to associate each term with a weight between 0 and 1 to indicate the importance of that term to the description as a whole. TF-IDF places greater weight on words that appear in a document multiple times, but down-weights by how frequently all documents include that word. For example, words that show up often in a document but also show up in most documents are weighted low, whereas words that frequently show up in a few documents are weighted high. Returning to the description of CVE-2017-6517, the highest weighted terms by TF-IDF are "load" and "skype". TF-IDF outputs a large, sparse matrix with its size given by number of unique, cleaned terms by the number of documents (CVE text descriptions).

The TF-IDF matrix is large and sparse, as each description has nonzero weights for a handful of terms out of all of the possible terms that appear in the corpus, so we perform dimensionality reduction to reduce the feature space. Latent Semantic Analysis (LSA) is a dimensionality reduction algorithm that uses a truncated Singular Value Decomposition (SVD) to infer similarity patterns between terms and documents and combine them into a vector of features with a value between -1 and 1. The results in this manuscript used the scikit-learn recommended parameter of 100 for the number of features to generate from the LSA, although we explored the accuracy of tuning that parameter and found it to be relatively insensitive so long as at least 25 features are preserved. The vector of LSA values per CVE description is one of the NLP features we extracted for evaluation in association of vulnerabilities with likelihood of exploitation, and it is the one that performed the best. We also explored the use of K-Means clustering to assign cluster labels to the CVEs based on the vector of LSA values, but that feature did not perform as well as just using the vector of LSA values.

Besides the CVSS score, CVSS vector, and CVE description, we also explored the Common Platform Enumerations (CPEs). CPEs provide a standardized way to break down information for software products and platforms in a machine-readable format. We used CPE version 2.3, which is formatted as: cpe:2.3:application/host/operating system:vendor:product:version:*:*:*:*:*:*:*:*. The single CPE associated with CVE-2017-6517 (CVEs may match with multiple CPEs) is cpe:2.3:a:microsoft:skype:7.16.0.102:*:*:*:*:*:*:*, which means that CVE-2017-6517 discusses an application with vendor Microsoft and product Skype that is version 7.16.0.102. CPEs were processed to generate categorical features based on application, host, or operating system, as well as a flag signifying whether the CPE was associated with the 25 most common vendors. However, none of these features were nearly as predictive of exploits for vulnerabilities as the natural language processing LSA values (64% and 70% as accurate as our best model, respectively), so they will not be discussed further. However, the CPEs proved useful in developing a mechanism for labelling attacker models for threat tuning, which will be discussed later.

4.2 Supervised Machine Learning for Binary Classification

We next evaluate a few machine learning algorithms that perform binary classification for accuracy and runtime speed. For our initial study, each CVE is assigned a label of 1 if there is an exploit for that vulnerability in Exploit-DB (https://www.exploit-db.com) and a 0 otherwise. We consider logistic regression, random forests, and neural network classifiers. Logistic regression is a machine learning technique that performs regression analysis for a linear combination of independent features. It has a fast runtime and outputs calibrated probabilities, which is useful in graduated risk scoring. However, if any nonlinear effects are desired, they are difficult to incorporate in logistic regression.

Random forests fit decision trees on subsets of training data and take the average results of all of the decision trees to improve the accuracy of prediction and reduce overfitting. Decision trees are able to account for dependent features that can produce nonlinear

Figure 3: Natural Language Processing Pipeline Used to Extract Features for Binary Classification

effects. As the number of decision trees is increased, the accuracy of the algorithm improves, but the runtime becomes longer.

Neural networks use a group of weighted artificial neurons or nodes that are interconnected between layers in order to learn a function that associates an output quantity to an input vector. They can be very accurate for binary classification methods, particularly with more nodes and layers in the network. A dense layer of a neural network connects every node from the previous layer with every node in the dense layer. A dropout layer randomly drops a percentage of nodes and connections from a layer, which greatly reduces overfitting of a model to the training data [23]. Figure 4 summarizes the neural network architecture that we considered for our application.

Each algorithm was trained with stratified 5-fold cross-validation, meaning that the CVE dataset was split into 5 equal portions with the same proportion of CVEs with and without known exploits. Then the algorithm was run 5 times, each time using a different fold as the test set with the other four as the training set. Then, the area under the curve (AUC) for the Receiver Operating Characteristic (ROC) for each run of the algorithm was averaged. An example ROC curve for logistic regression on the LSA values for natural language processing (NLP) is shown in Figure 5. This indicates that the logistic regression can be significantly more predictive of vulnerability exploitation in Exploit-DB than CVSS score.

When classifying vulnerabilities to exploits using CVSS base scores with logistic regression, the area under the ROC curve is 0.63. But by using the natural language processing feature of the vector of LSA values with logistic regression, the area under the ROC curve increases by 36% to 0.86, which is a significant increase in accuracy and a large decrease in the false positive rate. With CVSS2 scores in particular, up to a false positive rate of 0.18 (about 1 out of 5), the classifier is actually slightly worse than randomly classifying a vulnerability. For the NLP feature, a false positive rate of 0.18 corresponds to a true positive rate of about 0.72, which is a much better ratio of true positives to false positives. This shows that a network defender interested in prioritizing vulnerability mitigations with respect to exploit availability in Exploit-DB would be much more efficient using the logistic regression classifier than using CVSS base scores. We tested for potential label leakage issues by investigating whether the word "exploit" appearing in a CVE description correlated to a higher risk score for that CVE. In total, there were 7,502 CVEs with the word "exploit" in their CVE description. Those CVEs had an average risk score of 0.052, whereas the average risk score of all of the CVEs was 0.102, suggesting that the term "exploit" in the CVE description was not a critical feature driving up the risk score.

To investigate risk scoring accuracy over a time period, we also tried training on all CVEs published through September 2017 and testing on the CVEs published thereafter. There were 91, 035 CVEs in the training set and 21, 469 CVEs in the testing set. The area under the ROC curve came out to 0.75, which is still 19% higher than the area under the curve of 0.63 for the CVSS base scores, suggesting that the exploitability of past threats is more predictive than the severity of the threats themselves.

Once the vector of LSA values turned out to be a much-improved feature over CVSS2 base scores, we looked into different machine learning classification methods to evaluate the accuracy and runtime of those classifiers. Out of the classifiers discussed earlier, logistic regression was the fastest; see Table 1. Random forests and neural networks were slightly more accurate when there were more trees in the random forests classifier and more iterations of training for neural networks. However, time and resource constraints can be a limiting concern in relevant deployment scenarios, and marginal gains in model accuracy may not always be worth the expense. Logistic regression is almost as accurate as much more time-intensive methods, and allows for customization against particular threats and adversaries on-the-fly, since the training time is minimal. Therefore we selected logistic regression for our prototype. Nonetheless, for static threats and/or large compute environments, a periodic batch job with a neural network would definitely be more accurate.

Up until this point in this manuscript, the existence of an exploit in Exploit-DB has been used as the critical label for training the supervised risk models. However, Exploit-DB is almost certainly not the proper risk model that should be used on a daily basis to defend a specific network [4, 11]. For example, Grimes [9] argues that the most relevant threat information can be determined from current and historical experience that is specific to the defended network, noting also that it is "impossible to prepare for all threats...", and, "a lessor ranked vulnerability should be given higher criticality if it has been used in the past or is currently actively being used against the organization". Cyber defenders presently lack a tool to help quantitatively re-prioritize vulnerability risks according to how they are or have recently been attacked by a known adversary. In

Figure 4: Neural Network Architecture Used for Binary Classification of CVEs to Likelihood of Exploitation.

Table 1: Comparison of Classification Methods Accuracy and Training Times

Classification Method	Parameters	AUC	Runtime
Logistic Regression	Solver='lbfgs'	.856	0.2 min
Random Forests	10 Trees	.834	1.2 min
Random Forests	100 Trees	.881	9.6 min
Random Forests	500 Trees	.889	49.4 min
Neural Network	Epochs = 3, Batch Size = 20	.879	2.2 min
Neural Network	Epochs = 20, Batch Size = 5	.886	46.3 min

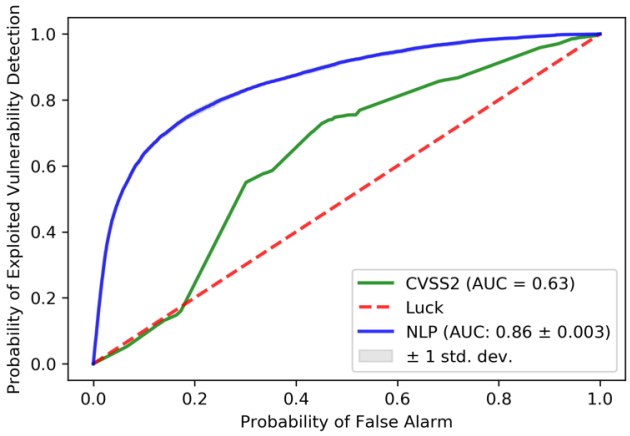

Figure 5: Receiver Operator Characteristic (ROC) Curve of Classification for Predicting Exploits in CVEs using Logistic Regression with Natural Language Processing LSA Values (blue) and via Decreasing CVSSv2 Base Score (green).

the next section, we propose a first-of-a-kind method that uses site-specific threat intelligence to reprioritize vulnerability risk, leveraging the same latent vulnerability features developed above.

4.3 Threat Tuning and Attacker Model

In this section, we consider how to reprioritize vulnerability risk scores according to their likelihood of exploitation by Advanced Persistent Threat (APT) group 28, show how the resulting prioritizations are different from an Exploit-DB-driven or CVSS score-driven risk ranking, and then demonstrate that held-out vulnerabilities associated with these APTs rank much more highly than they otherwise would by using either CVSS scores or Exploit-DB. The approach is overly simplistic in that it inherently assumes that the latent vulnerability features correspond well with adversarial exploit investment, i.e., that adversaries will likely continue to exploit vulnerabilities against software services that they have previously developed capabilities to reverse engineer [5]. Nevertheless, it is an effective demonstration of an approach that allows defenders to re-envision their software vulnerabilities from the perspective of a known adversary.

To develop the attacker model, we first picked a particular APT group, APT 28, and read through two threat reports [1][13] to get a list of CVEs that were known to be exploited by the threat group, as well as a list of software services the threat group targeted. In reading those threat reports, there were 12 CVEs listed that APT 28 exploited as well as five software services (Adobe Flash, Java, Windows, Microsoft Office, and Microsoft Word). Even though Microsoft Word is a subset of Microsoft Office, it is still considered separate since there are some CVEs that discuss Microsoft Word but not Microsoft Office, and vice versa.

For developing the custom threat profile, we hypothesize that APT 28, having sunk costs in exploiting the aforementioned software services, would be likely to continue to target those services. Therefore, we relabeled all CVEs in the database that included at least one of the software services mentioned as a target of APT 28 with a "1", with the exception of the 12 CVEs explicitly mentioned in the report. The rest of the CVEs were labelled as "0". In total, there were 7,049 CVEs that mentioned one of the targeted services, which is 6.99% of them. To train and evaluate the attacker model, logistic regression was used with stratified 5-fold cross-validation as described previously. However, in all cases we held out the 12 CVEs that APT 28 had exploited to see how well the model would predict

the risk scores for those CVEs and to make sure that the model did not overfit to those particular CVEs. We note that additional threat intelligence could also be used to customize a threat profile, e.g., targeted hardware, CVSS vector features, or search strings, but do not explore that here.

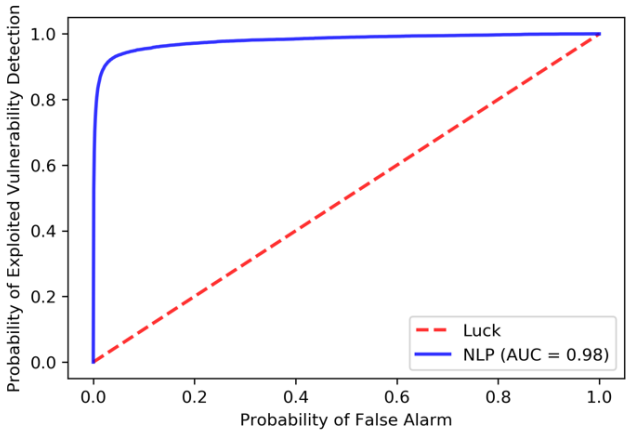

Figure 6: ROC Curve for APT 28 Classifier

The AUC for the APT 28 attacker model is 0.98, which indicates that the model was able to develop features that classified very well to the generated labelling based on the software services from the threat reports; the ROC curve is provided in Figure 6. However, we took a deeper dive into the 12 CVEs that APT 28 has exploited in the past, and compared them by CVSS score, Exploit-DB risk score, and APT 28 risk score, which was the output of the logistic regression before going into the threshold to classify as a 0 or 1. We also ranked each score across the 112,503 CVEs that we used in this work according to its inclusion in the top riskiest percentiles; those rankings are included in Table 2. The top percentiles are shown in bold, and, for 9 out of 12 APT 28 targeted vulnerabilities, the custom scoring is closest to the top of the list.

The resulting risk scores from using Exploit-DB and the attacker model for APT 28 also show that the threat profiles correctly prioritize different vulnerabilities based on their associations with exploits. For APT 28, two-thirds of the CVEs are between 9-10, but there are four that are between 6.5-7.2 that are also exploited by the threat group; this indicates that the attacker model approach can recommend vulnerabilities that would otherwise have a reduced priority. In addition, with the attacker model, all of the CVEs exploited by APT 28 are ranked in the top 8.3% of all CVEs for that threat profile, whereas with CVSS base scores they are all ranked in the top 47.0%, and for Exploit-DB they are all ranked in the top 98.7%. This shows the attacker model prioritization correctly places the true positives near the top of the list. As Table 3 shows, to capture 80% of the 12 APT 28 CVEs, the attacker model results in having to remediate the top 6.5% of vulnerabilities, whereas the Exploit-DB model results in the top 39.4%, and CVSS scoring results in the top 34.3%. To capture 90% and 100% in the Top-N percentile, the attacker model significantly outperforms Exploit-DB and CVSS score. Using the attacker model, all of the APT 28 CVEs would be prioritized in the top 8.3%, which reduces the work required by the Exploit-DB model by a factor of 11.8 and CVSS by a factor of 5.7.

One potential issue with this method was that the model might simply be finding software service names in the descriptions of the CVEs to use as a feature itself which could lead to label leakage. To test for this, we also ran an experiment where the names of the software services APT 28 was known to exploit were removed from the CVE descriptions. The AUC for this model came out to 0.87, meaning that the model is still pretty accurate even without the software service names in the descriptions. We also believe that the high accuracy arises because humans tend to write descriptions of vulnerabilities among the same software services similarly (this data is nearly always available at the time of prediction; it is not a classic label leakage problem in which the label becomes unavailable at the time of inference).

Although this is a very simple way of associating specific threat intelligence with vulnerability information (relabeling via adversary-associated software services), it is a successful demonstration of a novel mechanism of threat-tuning in a contested environment that is otherwise impossible with current tools. We anticipate that other features available from threat intelligence would be more predictive than software services alone.

5 CONCLUSIONS AND FUTURE WORK

Building upon the now established idea that exploitation likelihood ought to drive vulnerability risk rankings, we have presented a supervised machine learning approach that associates latent vulnerability features with their likelihood of exploitation according to site-specific threat intelligence. Using Exploit-DB as an evaluator of the pipeline, the strongest vulnerability feature found is the LSA vector information derived from natural language processing of the vulnerability text descriptions. The most accurate learning model found was a 500-tree random forest, although a neural network may also be used for similar accuracy (0.89 using Area Under the Curve (AUC) of the receiver operating characteristic as an accuracy measure), and logistic regression's accuracy of 0.86 is close enough that its runtime tradeoff makes it an attractive approach for in-house deployment. Increasing the number and size of the LSA vectors also enhances accuracy with corresponding runtime and memory trade-offs. All of our supervised-learning based approaches are more accurate than using CVSS score.

Having established the machine learning pipeline, we then demonstrated a simple way that it could be used to re-prioritize vulnerability risks within a contested environment using site-specific threat intelligence. We found that by retraining the model according to software services known to be targets of APT 28 (assuming they are attacking a site of interest), held-out vulnerabilities were found to go from the top 98.7% using Exploit-DB to the top 47.0% using CVSS score to 8.3% using the customized APT risk model. Using specific threat intelligence is quantitatively superior to any other industry-wide approach, despite the relative simplicity of our risk model.

As future work, we note that we did not consider obvious enhancements to the risk model such as incorporating vulnerability reachability in a network topology, potential mission impact of the host, or actual economic value of the defended asset or exploit

Table 2: Detailed Risk Scoring for APT 28 Associated CVEs Using CVSS, LSA-based Exploit-DB, and Custom APT 28 Attacker Models

CVE ID	CVSS Score	Exploit-DB Score	APT 28 Score	CVSS %	Exploit-DB %	APT 28 %
CVE-2015-2590	10	0.001	1.000	3.2	98.7	1.1
CVE-2016-7255	7.2	0.250	1.000	34.3	11.1	1.1
CVE-2017-0263	7.2	0.236	1.000	34.3	11.7	1.1
CVE-2016-7855	10	0.069	1.000	3.2	36.0	2.4
CVE-2015-7645	9.3	0.245	0.999	9.6	11.3	2.6
CVE-2015-1701	7.2	0.090	0.996	34.3	28.3	3.4
CVE-2015-5119	10	0.087	0.977	3.2	29.2	4.2
CVE-2015-3043	10	0.138	0.967	3.2	18.9	4.4
CVE-2017-0262	9.3	0.066	0.917	9.6	37.4	5.2
CVE-2015-2424	9.3	0.075	0.792	9.6	33.3	6.5
CVE-2016-4117	10	0.060	0.565	3.2	40.8	8.0
CVE-2017-11292	6.5	0.062	0.534	47.0	39.4	8.3

Table 3: Summary of Percentages of Vulnerabilities That Would be Recommended by Each of 3 Models (Custom Attacker, CVSS, and Exploit-DB) to Mitigate Increasing Percentages of the CVEs Identified as APT 28 Targets.

APT 28 CVEs % Mitigated	Attacker	CVSS2	Exploit-DB
80	6.5	34.3	39.4
90	8.0	34.3	40.8
100	8.3	47.0	98.7

development costs. Future risk models should incorporate a more detailed induction of the adversarial costs of developing exploits against latently similar vulnerabilities instead of assuming that the feature space "takes care" of these associations; this should be possible as new economic exploitation models emerge [2]. Additionally, we did not consider the risk induced from supporting a particular software service whose likelihood of new vulnerability discovery and exploitation could be inferred from prior data or the risk reduction that could be obtained via defender actions. Lastly, future work could be done in exploring other methods for creating features, such as word or paragraph embeddings, and compare that method against the LSA method for the CVE descriptions.

6 ACRONYMS

APT: Advanced Persistent Threat
AUC: Area Under the Curve
CPE: Common Platform Enumeration
CVE: Common Vulnerabilities and Exposures
CWE: Common Weakness Enumeration
CVSS: Common Vulnerability Scoring System
LSA: Latent Semantic Analysis
NIST: National Institute of Standards and Technology
NLP: Natural Language Processing
NVD: National Vulnerability Database
ROC: Receiver Operating Characteristic
SME: Subject Matter Expert
SVD: Singular Value Decomposition
SVM: Support Vector Machine
TF-IDF: Term Frequency-Inverse Document Frequency

ACKNOWLEDGMENTS

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Department of the Army under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Army. This work was supported in part by high performance computer time and resources from the DoD High Performance Computing Modernization Program.

REFERENCES

[1] 2017. (2017). https://www2.fireeye.com/rs/848-DID-242/images/APT28-Center-of-Storm-2017.pdf
[2] Luca Allodi. 2017. Economic factors of vulnerability trade and exploitation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1483–1499.
[3] Luca Allodi and Sandro Etalle. 2017. Towards realistic threat modeling: attack commodification, irrelevant vulnerabilities, and unrealistic assumptions. In *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense*. ACM, 23–26.
[4] Luca Allodi and Fabio Massacci. 2014. Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security (TISSEC)* 17, 1 (2014), 1.
[5] Luca Allodi, Fabio Massacci, and Julian M Williams. 2017. The work-averse cyber attacker model: Theory and evidence from two million attack signatures. *Available at SSRN 2862299* (2017).
[6] Benjamin L Bullough, Anna K Yanchenko, Christopher L Smith, and Joseph R Zipkin. 2017. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*. ACM, 45–53.
[7] Michel Edkrantz, Staffan Truvé, and Alan Said. 2015. Predicting vulnerability exploits in the wild. In *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. IEEE, 513–514.
[8] Katheryn A Farris, Ankit Shah, George Cybenko, Rajesh Ganesan, and Sushil Jajodia. 2018. Vulcon: A system for vulnerability prioritization, mitigation, and management. *ACM Transactions on Privacy and Security (TOPS)* 21, 4 (2018), 16.
[9] Roger A Grimes. 2016. Fixing the #1 Problem in Computer Security: A Data-Driven Defense. (Jan 2016). https://gallery.technet.microsoft.com/Fixing-the-1-Problem-in-2e58ac4a

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

[11] Jay Jacobs, Sasha Romanosky, Idris Adjerid, and Wade Baker. 2019. Improving Vulnerability Remediation Through Better Exploit Prediction. *2019 Workshop on the Economics of Information Security* (2019).

[12] Atefeh Khazaei, Mohammad Ghasemzadeh, and Vali Derhami. 2016. An automatic method for CVSS score prediction using vulnerabilities description. *Journal of Intelligent & Fuzzy Systems* 30, 1 (2016), 89–96.

[13] Fireeye Labs. 2015. Operation RussianDoll: Adobe & Windows Zero-Day Exploits Likely Leveraged by Russia's APT28 in Highly-Targeted Attack Âń Operation RussianDoll: Adobe & Windows Zero-Day Exploits Likely Leveraged by Russia's APT28 in Highly-Targeted Attack. (Apr 2015). https://www.fireeye.com/blog/threat-research/2015/04/probable_apt28_useo.html

[14] Zhechao Lin, Xiang Li, and Xiaohui Kuang. 2017. Machine learning in vulnerability databases. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, Vol. 1. IEEE, 108–113.

[15] Robert A Martin. 2001. Managing vulnerabilities in networked systems. *Computer* 34, 11 (2001), 32–38.

[16] Peter Mell, Karen Scarfone, and Sasha Romanosky. 2006. Common vulnerability scoring system. *IEEE Security & Privacy* 4, 6 (2006), 85–89.

[17] Stephan Neuhaus and Thomas Zimmermann. 2010. Security trend analysis with cve topic models. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*. IEEE, 111–120.

[18] Office of the Deputy Assistant Secretary of Defense for Systems Engineering. 2017. Department of Defense Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs. *Department of Defense* (2017).

[19] Sara Qamar, Zahid Anwar, Mohammad Ashiqur Rahman, Ehab Al-Shaer, and Bei-Tseng Chu. 2017. Data-driven analytics for cyber-threat intelligence and information sharing. *Computers & Security* 67 (2017), 35–58.

[20] Dennis M Ross, Allan B Wollaber, and Pierre C Trepagnier. [n. d.]. Vulnerability Ranking by Exploit Association in a Latent Feature Space. *Journal of Defnse Modeling and Simulation* (Revision Submitted) ([n. d.]).

[21] Dennis M Ross, Allan B Wollaber, and Pierre C Trepagnier. 2017. Latent feature vulnerability ranking of CVSS vectors. In *Proceedings of the Summer Simulation Multi-Conference*. Society for Computer Simulation International, 19.

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

[23] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[24] Nazgol Tavabi, Palash Goyal, Mohammed Almukaynizi, Paulo Shakarian, and Kristina Lerman. 2018. Darkembed: Exploit prediction with neural language models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[25] Malgorzata Urbanska, Indrajit Ray, Adele E Howe, and Mark Roberts. 2012. Structuring a vulnerability description for comprehensive single system security analysis. *Rocky Mountain Celebration of Women in Computing, Fort Collins, CO, USA* (2012).

[26] Sridhar Venkatesan, Shridatt Sugrim, Rauf Izmailov, Cho-Yu J Chiang, Ritu Chadha, Bharat Doshi, Blaine Hoffman, E Allison Newcomb, and Norbou Buchler. 2018. On Detecting Manifestation of Adversary Characteristics. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*. IEEE, 431–437.

[27] Allan B Wollaber and Pierre C Trepagnier. 2016. Exploit Probabilities Conditioned on CVSS Scores. *MIT Lincoln Laboratory Technical Report* (Nov 2016).

[28] Yasuhiro Yamamoto, Daisuke Miyamoto, and Masaya Nakayama. 2015. Text-mining approach for estimating vulnerability score. In *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*. IEEE, 67–73.