JOURNAL OF
CYBERSECURITY

Research Paper

Improving vulnerability remediation through better exploit prediction

Jay Jacobs[1], Sasha Romanosky[2,*], Idris Adjerid[3] and Wade Baker[3]

[1]Cyentia; [2]RAND Corporation and [3]Virginia Tech

*Correspondence address: 1200 South Hayes St, Arlington, VA, USA. E-mail: sromanos@rand.org

Abstract

Despite significant innovations in IT security products and research over the past 20 years, the information security field is still immature and struggling. Practitioners lack the ability to properly assess cyber risk, and decision-makers continue to be paralyzed by vulnerability scanners that overload their staff with mountains of scan results. In order to cope, firms prioritize vulnerability remediation using crude heuristics and limited data, though they are still too often breached by known vulnerabilities for which patches have existed for months or years. And so, the key challenge firms face is trying to identify a remediation strategy that best balances two competing forces. On one hand, it could attempt to patch all vulnerabilities on its network. While this would provide the greatest 'coverage' of vulnerabilities patched, it would inefficiently consume resources by fixing low-risk vulnerabilities. On the other hand, patching a few high-risk vulnerabilities would be highly 'efficient', but may leave the firm exposed to many other high-risk vulnerabilities. Using a large collection of multiple datasets together with machine learning techniques, we construct a series of vulnerability remediation strategies and compare how each perform in regard to trading off coverage and efficiency. We expand and improve upon the small body of literature that uses predictions of 'published exploits', by instead using 'exploits in the wild' as our outcome variable. We implement the machine learning models by classifying vulnerabilities according to high- and low-risk, where we consider high-risk vulnerabilities to be those that have been exploited in actual firm networks.

Key words: vulnerability management; exploited vulnerability; CVSS; security risk management; machine learning; precision; recall

Introduction

A critical challenge for many modern organization is understanding how to minimize the cost of managing and protecting its information assets and business systems. A core component of this challenge is adopting a vulnerability management process that can detect, and remediate known vulnerabilities.[1] Unfortunately, despite decades of research and technical innovations, there have been few advances in remediation practices. Given that firms will always have more exposed vulnerabilities than resources to fix them, firms struggle trying to develop and apply a remediation strategy that will optimally patch those vulnerabilities that pose the greatest risk, while also deprioritizing those vulnerabilities that pose the lowest risk.

In theory, a firm seeks to balance two competing forces. On one hand, it could attempt to patch all vulnerabilities identified on its network, which would provide the greatest 'coverage' of vulnerabilities patched, but would inefficiently consume resources to fixing vulnerabilities that pose a lower risk. On the other hand, the firm could patch a small set of high-risk vulnerabilities. While this strategy might be highly 'efficient', many other potentially high-risk vulnerabilities may remain exposed. In practice, most firms are not sophisticated about managing this trade-off and use heuristic strategies to prioritize their remediation efforts; for example, a common approach is to remediate all vulnerabilities above a certain severity score. However, many of the common heuristics used by firms have

---

1 We are not concerned with zero-day vulnerabilities in this article.

been found to be sub-optimal [1, 2] and in some cases, no better than randomly choosing vulnerabilities to remediate [3].

One of the key reasons the current approaches are ineffective is that firms cannot effectively assess whether a given vulnerability poses a meaningful threat. For instance, prior work suggests that of all the publicly known vulnerabilities, only 10–15% actually ever have a known exploit written for them, and even fewer still are ever weaponized as part of hacking toolkits [4]. An even smaller proportion of vulnerabilities are ever targeted against an organization in the wild. Sabottke, Suciu, and Dumitras [5] find that as few as 1.4% of published vulnerabilities have exploits which have been observed in the wild. Given that so few vulnerabilities are actually a focus for attackers in the real world, a promising approach towards remediation is to identify vulnerabilities which are likely to be actually exploited, and therefore prioritize firm efforts towards remediating those vulnerabilities first.

Because an exploit observed in the wild is the most relevant proxy for the probability that an exposed vulnerability can be used to compromise a firm's network, our focus in this article is on building predictive models that can identify an exploited vulnerability.

We start by generating ground truth data of whether a vulnerability has an exploit observed in the wild. We aggregate data from a variety of sources, including a private dataset generated by a partner security firm that monitors more than 100 000 corporate networks (almost 200 billion observations of network traffic). To our knowledge, ours is the most comprehensive ground truth data used in this kind of prediction effort, and the only one to rely on data from a diverse set of real-world intrusion detection systems. Notably, we observe exploits in the wild for 5.5% of vulnerabilities in our dataset compared to 1.4% in prior works. We also collect a novel set of features for prediction; for example, we text mine detailed descriptions of each vulnerability to extract 191 tags (e.g. 'buffer overflow', 'denial of service') that are included as features in our prediction model. Combining this unique dataset with gradient boosted trees we build a predictive model with exploits in the wild as the target variable.

The combination of a more comprehensive measure of exploits in the wild and novel features for prediction results in a model that can effectively identify vulnerabilities with a high risk of exploitation in the wild. Our model achieves an accuracy of 94.5 (with the ROC AUC of 0.91) and produces a significant lift in prediction coverage and efficiency compared to prior work (as shown in Fig. 1a and b).[2]

More importantly, the lift in prediction accuracy that our model provides significantly improves the efficiency of remediation strategies. Informed by our prediction model, a firm seeking broad coverage of vulnerabilities that are exploited in the wild (e.g. 70%) can achieve this by remediating only ∼7900 of all vulnerabilities. On the other hand, this same level of coverage would require the remediation of more than three times the vulnerabilities using both comparable heuristic approaches (∼34 000) and ML models built on the features utilized by prior works (∼30 000). This reduction in the number of vulnerabilities that a firm needs to remediate is due to a dramatic decrease in the number of false positives remediated under an approach informed by our model.

Our work contributes to the literature on the economics of information systems, and computer science literature on vulnerability remediation. In addition, we believe this work has significant implications for decision makers when assessing cyber security risk, to include firms, federal agencies and national security policy makers.

In further work, we will examine more deeply the mechanics behind our model, identify which features are most important to predicting exploits in the wild, and generate predictive models for important subsets of vulnerabilities (e.g. based on the most popular vendors).

The next section discusses related literature, followed by a description of the datasets used in this research. We then present the results from three separate vulnerability remediation strategies using simple rule-based approaches, followed by a full machine learning prediction model. We conclude with a discussion on limitations and conclusion.

Related literature

This article draws on a body of work focused on the economics of information security. Our work is particularly relevant to a stream of research evaluating the efficiency of vulnerability remediation; a key function related to how organizations proactively guard against security incidents. This stream of research includes evaluations of the appropriateness of strategies for identifying new vulnerabilities and disseminating that information to relevant stakeholders [6–9], the design of incentives and liability with respect to vulnerability remediation [10–12], and the efficacy of strategies that firm take when attempting to prioritize the vulnerabilities they address [1–3]. A consistent finding in this stream of research is that the status quo for how organizations address vulnerability remediation is often suboptimal and has significant room for improvement. For example, [3] evaluate the effect on exploit risk of prioritizing vulnerabilities with high severity scores (a common practice in organizations) is equivalent to randomly selecting vulnerabilities to address.

Our work also relates to an emerging body of research coupling machine learning methods with diverse feature sets drawn from information security contexts and explores the value of prediction for information security management [13–15]. This literature holds significant potential value for a few reasons. First, information security contexts are marked by considerable technical and managerial uncertainty. The management of this uncertainty is exacerbated by the limited resources that most organizations dedicate to security. Second, there are long-standing challenges with properly leveraging massive collections of data (network traffic data, user logs, etc.) that hold within them relevant insights for efficiently managing current and future security risks. For instance, current tools are often inadequate at prioritizing and identifying key risks and often inundate security professionals with too many alerts and false alarms. As a result, critical warnings are either never generated or lost in a sea of other, often less important, warnings. Third, vulnerabilities may be unknown to organizations (e.g. zero days) and thus difficult to identify with standard approaches. In these cases, machine learning methods (e.g. those focused on anomalous pattern detection) can identify new attacks by recognizing changes in network or user behaviour [13, 15].

A nascent stream in this body of work most directly relates to our work and combines machine learning approaches with a variety of datasets in order to reduce the uncertainty surrounding exploit risk of disclosed vulnerabilities [4, 5, 16, 17]. While these works start to bridge the gap between vulnerabilities and the real-world risk they present to systems, the majority of these works (with the exception of [5]) focus on vulnerabilities with a published exploits as opposed to an exploit observed in the wild. This is because predicting actual exploits in the wild remains a difficult prediction problem. In particular, and as we noted previously, exploits in the wild are rare. Of all the vulnerabilities that are disclosed, only a small subset have written exploits. Even fewer still have exploits

---

2 False Positive Rate (FPR), and True Positive Rate (TPR).
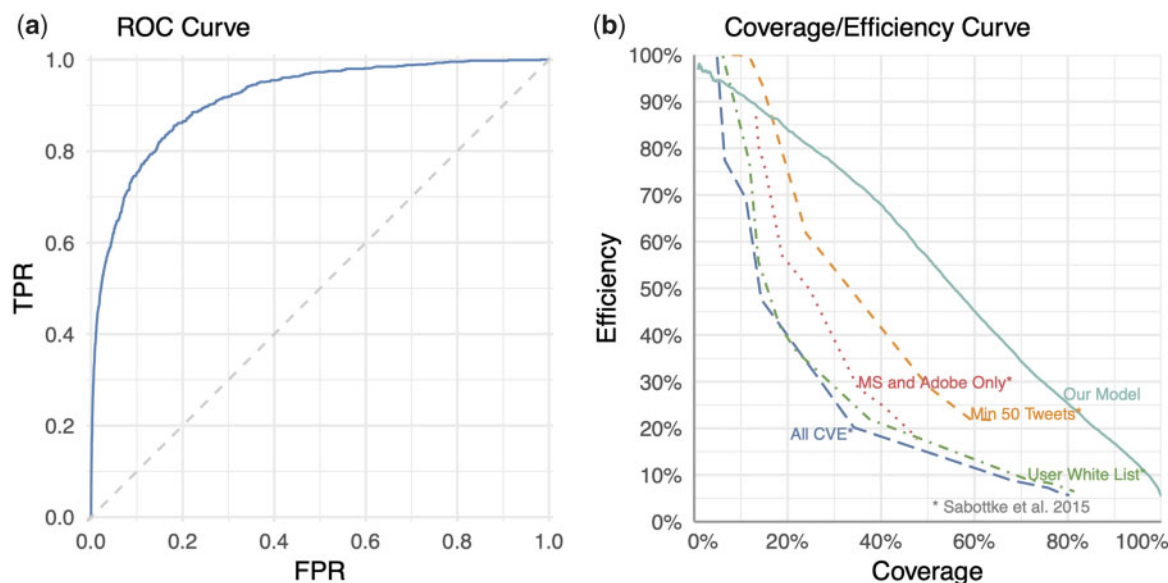
Figure 1: (a) ROC curve of our model output and (b) comparison of exploit prediction models [5].

which are weaponized and become available as part of hacking tool-sets. Smaller still is the proportion of exploits that are observed in use in the real world (e.g. on corporate networks). For example, prior works find that less than 1.4% of disclosed vulnerabilities are observed in the wild.

This raises two notable problems for these prediction efforts. First, classification methods have trouble disentangling rare events and require rich feature sets to differentiate vulnerabilities with high risk of exploit from more benign vulnerabilities. Second, there is limited ground truth data on which exploits are actually being levied against organizations. For example, [5] available signature data-bases from Symantec to identify vulnerabilities likely to be in the field, but these data have notable limitations as they are not based on actual network traffic from corporate networks and only capture vulnerabilities in Microsoft products. As we discuss in the following section, a key contribution of our work is to address these data limitations to build a more accurate and usable prediction model for vulnerability remediation. We do so by leveraging a unique set of features for prediction and amalgamating data sources to more comprehensively capture exploits that exist in the wild.

Data

This article leverages multiple datasets collected in partnership with Kenna Security, a large, US-based vulnerability and threat management company.[3] As a foundation, we use a dataset of vulnerabilities published by MITRE's Common Vulnerability Enumeration (CVE) effort between 2009 and 2018. The CVE list is widely known, and is the authoritative source of publicly known vulnerabilities. MITRE's enumeration includes a unique identifier (CVE number), a short free-text description, and a list of references to additional details of the vulnerability (in the form of URLs). After cleaning, and including only valid and fully documented vulnerabilities, this dataset contains over 75k observations. While CVE data exist for years prior to 2009 and past 2018, we limit the sample for the availability of data across all categories of features.

We also collect additional information for each CVE. We used NIST's National Vulnerability Database (NVD) to gather the vulnerability score and vulnerability characteristics, as defined by the Common Vulnerability Scoring System (CVSS). We also collected the Common Platform Enumeration (CPE) information, which provides a standard machine-readable format for encoding names of IT products, platforms and vendors.

Next, we retrieved the descriptive text from the list of references present in each CVE. We then extracted common multiword expressions from the raw text using Rapid Automatic Keyword Extraction (Rose *et al.*, 2010) and manually culled and normalized a list of 191 tags encoded as binary features for each vulnerability.

For each vulnerability, we also gathered real-world information about whether exploit code was published for it, and separately whether the vulnerability was exploited in the wild. These are related, but not identical concepts. The former relates to vulnerabilities for which exploit code has been written (but not necessarily used), and the latter relates to actual exploits observed in (at least attempted) malicious attacks. Since there is no authoritative or complete source of exploit information, we use many sources.

Information about written exploit code comes from Exploit DB, Metasploit, D2 Security's Elliot Kit and Canvas Exploitation Framework, Contagio, Reversing Labs and Secureworks CTU. These sources produced 9726 observations of written exploit code.

Data relating to exploits found in the wild were collected from FortiGuard Labs[4] which draws from Fortinet's vast array of devices/sensors collecting billions of threat events and incidents observed in live production environments around the world. In addition, we collected exploit data from SANS Internet Storm Center, Secureworks CTU, Alienvault's OSSIM metadata and Reversing Labs metadata. These sources produced 4183 observations of exploits used in the wild.

Finally, through Kenna Security, we obtained a count of the prevalence of each vulnerability derived from scans of hundreds of corporate (customer) networks derived from vulnerability scanner information. This rich feature-set of real-world data is used to enhance the performance of the models.

---

3   See http://www.kennasecurity.com

4   See https://fortiguard.com/

Table 1: Data summary

Data type	Source(s)	Observations (n)	Features (p)[a]
CVE Number[b]	Mitre CVE database, 2009–18	75 976	
CVSS score[c]	NIST's NVD	75 423	20
Products affected[d]	NIST's CPE	75 582	43
Vendor affected[e]	NIST's CPE	75 582	26
Reference Lists	MITRE's CVE list	75 976	31
Vulnerability Tags[f]	Text Scraped from referenced URLs listed in MITRE's CVE list	68 491	83
Exploitation observed in the wild	FortiGuard Labs, SANS Internet Storm Center, Securewords CTU, Alienvaults OSSIM meta-data, and Reversing Labs metadata.	9726	1
Exploit code published	Exploit DB, exploitation frameworks (Metasploit, D2 Security's Elliot Kit and Canvas Exploitation Framework), Contagio, Reversing Labs and Secureworks CTU	4183	4
Vulnerability prevalence data	Kenna Security	35 405	1

[a]This refers to the number of variables collected per data source.

[b]The CVE number is just an identifier to tie disparate data sets and was not used in any model.

[c]The version 2 base score, temporal score, and the 6 base metrics, each with 3 values per metric.

[d]Initially we constructed a dataset of 28 647 product names, but reduced this to 43 features that exceeded threshold for determining near zero-variance.

[e]Initially we constructed a dataset of 11 554 attributes, but this was again reduced to the 26 most common without near-zero variance.

[f]Initially we constructed a dataset of 191 commonly use expressions, but this was again reduced to the 83 most common without near-zero variance.
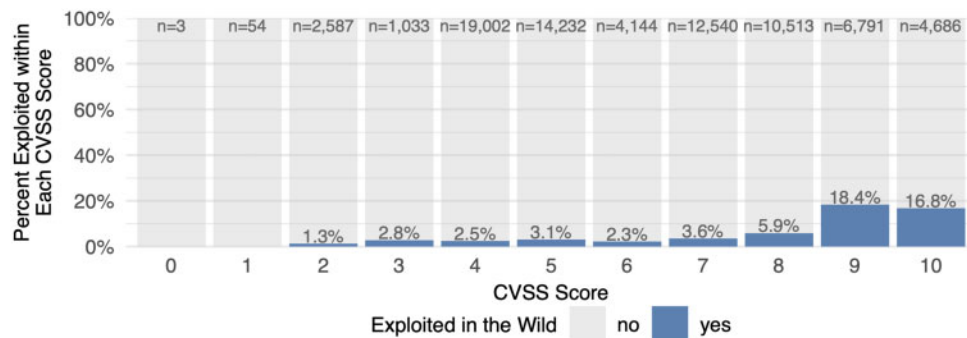
Figure 2: Exploited vulnerabilities, by CVSS severity.

Note that we removed those features that were sparsely populated with zero or near-zero variance. For binary variables, near zero-variance was set at a threshold 0.5%, and so features with less than 0.5% unique values were removed. Summary information for these data are shown in Table 1.

Next, we highlight three of the key data sources used in our prediction models, CVSS scores, published exploits and reference tagging.

CVSS

The Common Vulnerability Scoring System (CVSS v2) was first developed in 2003 and has become an international[5] and de facto standard for measuring the severity of a vulnerability. CVSS produces a numeric score between 0 (lowest severity) and 10 (highest severity) and is fundamentally an ordinal scale, based on 6 immutable characteristics of a vulnerability, and is independent of any user

environmental configurations, security controls or known exploits. Figure 2 shows descriptive information regarding the distribution of vulnerabilities by CVSS (v2) score. In addition, for each integer range, we show the proportion of vulnerabilities that have been exploited. For example, of all vulnerabilities, 4686 were given a score of 10, of which almost 17% of which were observed to be exploited.

Visual inspection shows that vulnerabilities with a CVSS score of 9 or 10 have a higher proportion of exploited vulnerabilities, relative to the base rate of 5.5%. Also, just under half of all exploited vulnerabilities have a CVSS score of 9 or higher.

Published exploits

As mentioned, we consider 'published exploits' to be software code posted publicly and designed to compromise a weakness in another

---

5 ITU-T X.1521.

software application, while 'exploits in the wild' are instances of this code observed to be targeted at actual corporate networks. There is an important relationship to note between published exploits, and exploits found in the wild, as shown in Fig. 3.

First, notice that overall our dataset contains 9.7k published exploits, and 4.2k observed exploits in the wild. That is, about 12.8% (9.7k/76k) of all vulnerabilities between 2009 and 2018 had published exploit code, while only about 5% (4.2k/76k) of all vulnerabilities were exploited in the wild. Further, 'only about half of all exploited vulnerabilities (2.1k) have associated published code'. This is, in itself is, an important finding because it suggests the need for an improved approach to vulnerability remediation. Specifically, previous research has used published exploits as the outcome target, while exploits in the wild represents, what we believe, is the preferred outcome measure. .

Reference tagging

Reference tags as used in our data are collected by a multistep process. We first begin by using the text description from MITRE's CVE list, then scrape the text from the URLs listed in the references. We then extract 'multi-word expressions' from the text and then manually curate a list of 'multi-word expressions'. Note that other research uses a simple 'bag of words' (single word frequencies independent of context) approach. The result is a list of expressions like 'code execution', 'directory traversal', 'sql injection' and 'denial of service'. We then manually reviewed the expressions and combined

Figure 3: Published exploits vs exploits in the wild.

multiple expressions into a single tag. For example, 'buffer overflow' may be expressed in the text as 'buffer overrun', 'stack overflow' or 'heap overflow' and the manual review helped reduce different variations of the same concept into a single tag. This resulted in vulnerabilities being assigned a normalized tag derived by matching expressions.

Our justification for identifying the multi-word expressions is that there is a significant difference in exploiting a 'memory corruption' vulnerability and a 'double free memory' vulnerability. Using multi-word expressions enables us to isolate those two concepts into two distinct features.

Additionally, we found that the brief text descriptions in MITRE's CVE list did not adequately capture all the complexities behind each vulnerability. By scraping the referenced websites in each CVE, we expect to achieve a more complete text descriptions of attributes that may be sought out and exploited by attackers (e.g. authenticated, local/remote, labels and phrases that describe the vulnerability itself, and the impact). The use of the Rapid Automatic Keyword Extraction [18] technique along with manual curation was used to tag concepts within each vulnerability and is expected to outperform previous. Figure 4 shows a random selection of 20 tags from the CVE references, along with the overall frequency (text at the top of each bar) and the proportion of vulnerabilities with that tag that were observed to be exploited in the wild. After removing tags with near-zero variance, our final reference tag feature set has 83 individual tags.

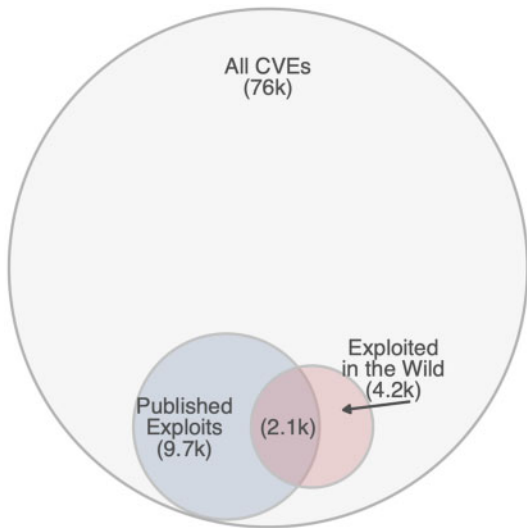Next, we describe the model evaluation criteria and machine learning configuration.

Model development

Exploited in the wild as the outcome measure

Before we describe the modeling approach, we first describe what we mean by 'risk' and therefore how we classify a high-risk vulnerability. Many information security best practices, including the US-based NIST recognize that risk is a combination of 'threat' (capability and intent of the threat actor), 'vulnerability' (weakness or exposure) of the targeted system, and the 'impact' (consequence or disruption) to the organization conditional on a successful attack (See Fig. 3, p12 , [19]). In this research, we seek to develop vulnerability remediation strategies that are agnostic to individual firms, and so we restrict our definition of risk by abstracting away firm-specific criteria such as a firm's security controls and impact to the organization. As a result, for the purpose of this article, our use of risk reflects threat only—'the exposure posed by an exploitable vulnerability'.

As mentioned, while previous work uses published exploits, we believe this is a premature measure, and that the most appropriate
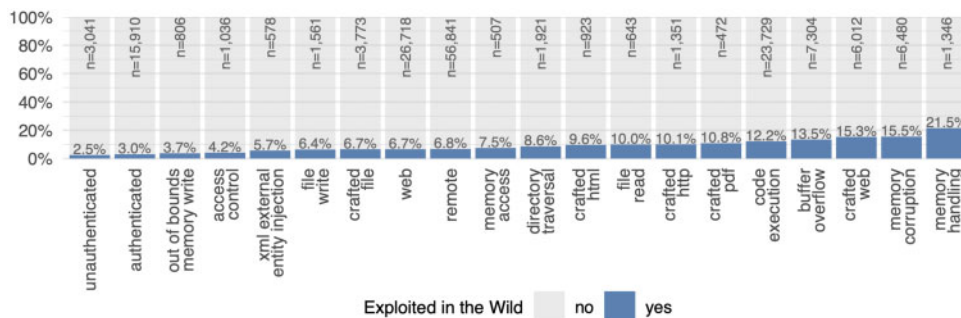

Figure 4: Prevalence and exploitation of multiword expressions (20 randomly sampled shown).

measure of threat relates to whether a vulnerability has actually been exploited in the wild, and if it has, we then consider the vulnerability to pose a high risk to the organization.[6]

Evaluation criteria

Throughout our analysis, we will evaluate the performance of the models along four dimensions: coverage, efficiency, accuracy and level of effort.

'Coverage' (recall) and 'efficiency' (precision) are two common evaluation criteria used in information theory and machine learning (classification) models. In our context, coverage measures the completeness of remediation. For example, of all vulnerabilities that should be remediated, what percentage was patched?: if 100 vulnerabilities are being exploited, and yet only 15 are remediated, the coverage of this prioritization strategy is 15%. Coverage is represented mathematically as the true positives divided by the sum of the true positives and false negatives, or $TP/(TP + FN)$, and is maximized as the number of false negatives tends to zero.

On the other hand, 'efficiency' measures the optimality of remediation efforts. Of all vulnerabilities remediated, what percentage 'should' have been addressed? For example, if we remediate 100 vulnerabilities, but only 15 are ever exploited, the efficiency of this strategy would be 15%. The other 85% represents resources that would have been better spent elsewhere. Efficiency is represented algebraically as true positives divided by the sum of the true positives and false positives, or $TP/(TP + FP)$, and is maximized as the number of false positives tends to zero.

The ideal strategy, of course, achieves 100% coverage and 100% efficiency (i.e. only patch those that you need to, and no more), but a trade-off exists between the two. A strategy that prioritizes only high-risk vulnerabilities may have a good efficiency, but it comes at the price of low coverage, because many vulnerabilities with existing exploits have a low severity score. Conversely, we could improve coverage by remediating more vulnerabilities with high severity scores, but suffer in efficiency due to fixing vulnerabilities that were never exploited and therefore pose lower risk.

'Accuracy' measures the overall correctness of predictions. Accuracy is the sum of the true positives and negatives divided by the sum of samples, or $(TP + TN)/(TP + TN + FP + FN)$.[7]

A given set of coverage and efficiency parameters uniquely defines the performance of a remediation strategy. What this does not explicitly communicate, however, is the number of vulnerabilities that the organization would need to patch in order to satisfy that strategy. Therefore, we define 'level of effort', as one measure of the 'cost' to a firm from that remediation strategy.[8]

Basic and machine learning models

In the next section, we explore three distinct vulnerability remediation strategies that firms may employ. Each strategy uses a different set of features (i.e. distinct independent variables), and for each

strategy, we first compute the outcome parameters (coverage, efficiency, accuracy and the level of effort) from a simple rule-based approach, and then compare those results to a machine learning model using the same feature sets (but which computes additional variable interactions).[9]

All machine learning models are gradient boosted trees, generated with Extreme Gradient Boosting (XGBoost) [20]. Gradient boosted decision trees improve on the standard decision trees by building successive decision tree models (boosting) which seek to classify and correct errors in previous models [21]. These types of ensemble methods have seen significant success in a number of classification tasks. We also explored using the Random Forest and linear SVM algorithms. However with the area under the precision-recall curve being 0.51 for random forest and 0.494 for the linear SVM, we selected XGBoost as our final model with the AUC of 0.55.[10] Because of the extreme class imbalance, we down-sampled [22] the majority class during training. However when evaluating the models we use the entire (unbalanced) test sample. Because of the sparseness of some features, we performed 5-fold stratified cross-validation repeated five times to limit possible over-fitting in the training. Since each model produces a probability that a particular sample will or will not be exploited-in-the-wild, a cut-off must be selected to classify the data. This cut-off is tunable in a way that can balance efficiency vs coverage. We select a cut-off value which maximizes the F1 score [23], which is the weighted harmonic mean of coverage and efficiency.

Feature-based remediation strategies

CVSS score

A primary aspect of vulnerabilities often considered during remediation decisions is the severity or impact that a vulnerability would have if it were successfully exploited. Assigning remediation efforts based on CVSS score has become a familiar and popular means of prioritizing patching, especially given how CVSS considers scores from 7 to 8.9 to be 'high severity', while scores above 9.0 are critical. For example, The US Department of Homeland Security (DHS) issued a binding operational directive to federal agencies, requiring them to patch critical vulnerabilities within 15 days and high severity ones within 30 days.[11] In addition, the Payment Card Industry Data Security Standard (PCI-DSS) requires that all merchants remediate vulnerabilities with a CVSS score above 4.0 [24].

The prediction results for a firm adopting a remediation strategy driven by CVSS score, including the machine learning model which accounts for interactions, are shown in Table 2. For brevity, we only show tabular results for four rules-based strategies, while we display full results in Fig. 5.

The machine learning (ML) model for this strategy uses 20 features of CVSS for training, and is shown in Fig. 5. The CVSS feature set includes each of the base metrics as defined in CVSS (version 2): attack

---

6   We recognize that different organizations may have different decision strategies and needs for asset protection (which may suggest a strategy for remediating 'all' vulnerabilities), we believe it is reasonable to first consider an efficient approach of first patching those that will be exploited.

7   This particular metric can create misleading results when the data is imbalanced such as we have in the exploitation data. Since only 5.5% of our vulnerabilities have been observed exploited in the wild, a simple model of remediating nothing yields 94.5% accuracy.

8   Certainly, this is an imperfect measure of the cost of patching to firm because of the many efficiencies of scale involved in enterprise patch management.

9   Note that we apply no pre-processing steps to the data in order to exclude vulnerabilities that may be highly unlikely to be exploited because this would require subjective assessment of a very low-risk vulnerability.

10   Note the area under the curve metrics here are for the precision-recall curve not the receiver operating characteristic (ROC) curve. The AUC of the ROC curve for XGBoost is 0.91.

11   See https://cyber.dhs.gov/bod/19-02/.

Table 2: CVSS prediction results

Strategy	Accuracy (%)	Efficiency (%)	Coverage (%)	Level of effort (# vulns)	Efficiency by chance (%)	Coverage by chance (%)
CVSS 10+	90.40	16.80	18.80	4686	5.53	6.20
CVSS 9+	84.70	17.80	48.70	11 477	5.53	15.20
CVSS 8+	72.40	12.10	63.60	21 990	5.53	29.10
CVSS 7+	57.00	9.00	74.30	34 530	5.53	45.70
CVSS 6+	51.80	8.30	76.60	38 674	5.53	51.20
CVSS 5+	34.10	6.90	87.10	52 906	5.53	70.00
CVSS 4+	10.20	5.70	98.50	71 908	5.53	95.10
ML Model	85.3	18.80	50.20	11 153	5.53	14.80

Note: Efficiency by chance and coverage by chance are the overall results from a strategy of randomly selecting vulnerabilities to remediate.
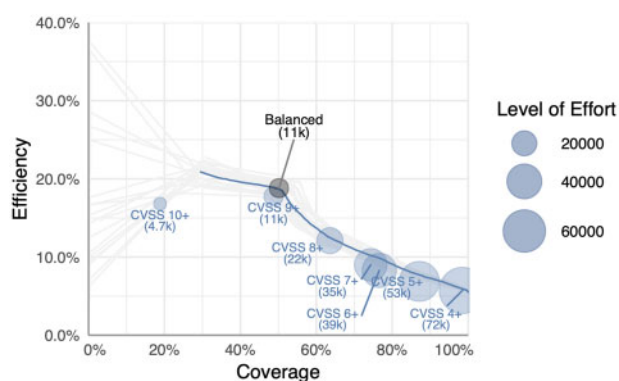
Figure 5: CVSS prediction results.

Note: Level of effort refers to the numbers of vulnerabilities to be patched by the given strategy.

vector, access complexity, authentication, and three impact metrics each for confidentiality, integrity and availability.[12]

These results suggest that a rule-based strategy of remediating all vulnerabilities with CVSS 7 or higher would achieve coverage of slightly over 74% with an efficiency of 9%, and accuracy of only 57%. This appears to be the best balance among CVSS-based strategies, even though it would still result in unnecessarily patching 31k (76k total–35k patched) unexploited vulnerabilities. Further, note that the PCI-DSS mandate requires that vulnerabilities scoring 4+ be patched, which is equivalent to an efficiency of random patching, provides only slightly better coverage, and at the cost of fixing 72k vulnerabilities.

While some research shows that a strategy of patching by CVSS score is no better than random [3], our results show that a strategy of patching CVSS score 9+ performs significantly better than random.

While the ML model performs better than a strategy of patching by any single score, the differences are not dramatic. Indeed, for most all CVSS strategies, the machine learning model performs only marginally better.

Next, we examine a strategy of patching according to whether an exploit involving the vulnerability has been published publicly.

Published exploits

It is a relatively recent capability that organizations are able to record information about vulnerabilities that have been exploited in the wild. Prior to that, and a resource that is still quite valuable, practitioners had relied on datasets either describing how to exploit a vulnerability, or providing working or proof-of-concept code to exploit known vulnerabilities. One may well consider that these 'published' exploit repositories are reasonable proxies (predictors) for exploits found in the wild, and absent ground truth data (i.e. complete information about all exploits), firms might employ a remediation strategy based off of published exploits. Indeed, as described above, a number of research efforts have used published exploits as their outcome target variable.

In Table 3, we provide prediction results for three exploit code repositories (Elliot, Exploit DB and Metasploit), and our machine learning model.

We first observe that of the rules-based approaches, the *Exploit DB* strategy has the best coverage, but suffers from considerably poor efficiency, while Metasploit performs exceptionally well in efficiency (highest out of all the rules-based approaches), with considerable reduction in coverage, and drastically smaller level of effort required to satisfy the strategy. The *Elliot DB* strategy, while not efficient as the *Metasploit* strategy, incurs a relatively tiny level of effort of just 285 vulnerabilities that would need to be patched. These same results are also presented graphically in Fig. 6.

Visually, it is even more clear how well the *Metasploit* strategy performs relative to the other rules-based models, suggesting that vulnerabilities (and exploits) that appear in this repository are very likely to be seen in the wild. On the other hand, exploits that appear in Exploit DB require a great deal more coverage, for little efficiency gain.

There are important distinctions between these exploit repositories which may help explain these variations. Exploit DB is an online repository of scripts and source code for exploits. Execution of the exploits listed in exploit DB often require a minimal level of technical

---

12 Attack vector refers to the network proximity required by an attacker in order to exploit a vulnerability (e.g. differentiating whether an attacker can launch a single packet from across the internet, or whether she requires physical access to the vulnerable device). Authentication refers to the level of additional authentication privileges the attacker requires in order to exploit the vulnerability (e.g. must the attacker authenticate

to the vulnerable system before attack, or can she execute the attack anonymously). Access complexity considers whether the attacker must rely on actions beyond her control (such as winning a computing race condition) in order to carry out the attack. The three impact metrics measure the degree of loss by the system. Further information can be found at https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator.

Table 3: Published exploits prediction results

Strategy	Accuracy (%)	Efficiency (%)	Coverage (%)	Level of effort (#vulns)	Efficiency by chance (%)	Coverage by chance (%)
Elliot	94.35	34.40	2.30	285	5.53	0.40
Exploit DB	86.78	17.90	38.80	9055	5.53	12.00
Metasploit	94.89	63.20	18.20	1208	5.53	1.60
ML Model	95.36	70.50	27.90	1657	5.53	2.20
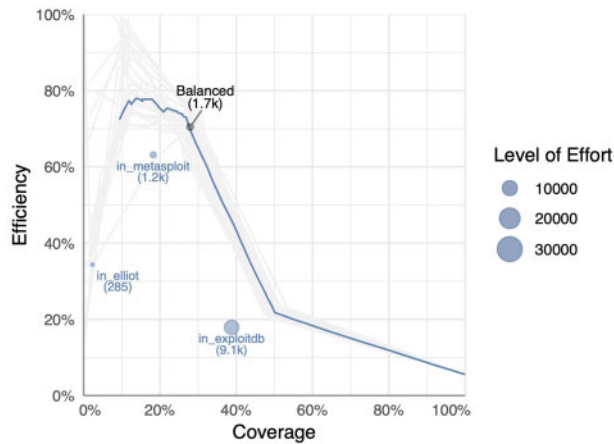
Figure 6: Published exploits prediction results.

Note: Level of effort refers to the numbers of vulnerabilities to be patched by the given strategy.

ability such as knowledge of how to edit and execute a scripting language or knowledge to compile source code to get a working exploit. The D2 Elliot Web Exploitation Framework (Elliot) and the Metasploit framework are both applications that can be installed and each contains a set of working exploit modules. Unlike Exploit DB, the technical ability required to exploit a target system is lower since the exploit code is easily executable through a user interface. Those differences undoubtedly affect their placement on the efficiency/coverage metrics. Given the data at hand, it is difficult to parse out if the Metasploit community is talented at identifying vulnerabilities that will be exploited in the wild, or if vulnerabilities are exploited in the wild because the modules exist in an exploitation framework with a user interface. But it is clear that over 60% of the modules associated with CVEs are observed to be exploited in the wild.

In addition, an interesting artefact emerges from the machine learning curve of Fig. 6. The discontinuity is likely an artefact of a model built on a small number of binary variables. There are very few variations in the way the binary variables interact and the left side from the discontinuity is likely revealing a handful of vulnerabilities with unique mixtures of the variables. This would explain why the individual cross-validation folds fan out as precision increases (gray lines in Fig. 6)—a small number of unique combinations may appear unevenly in either the training or testing data set, creating the variation. Nevertheless, notice how this ML model performs better than any of the individual strategies, achieving a better balance of coverage and efficiency.

The final remediation strategy presented below considers patching according to select keywords found in the vulnerability description.

Reference tagging

Finally, we consider a remediation strategy consisting of commonly used keywords as provided in the CVE reference links. Keywords can provide a shorthand insight into the characteristics and potential impact from particular sets of vulnerabilities, beyond what is captured by CVSS. And so these become useful heuristics upon which to based a remediation strategy.

Rather than show the full curated 83 unique reference tags in Table 4, we selected 4 tags, but all 83 tags are represented in Fig. 7.

Perhaps unsurprisingly, all the rule-based approaches are relatively inefficient. Tags like 'code execution' and 'remote' (i.e. 'remotely exploitable') achieved relatively similar efficiency but at a much greater cost of coverage. For example, a firm would need to remediate 24k vulnerabilities under the code execution strategy, while only 7.3k vulnerabilities under the buffer overflow strategy—for the same level of efficiency. The results are shown graphically in Fig. 7.

Overall, like the other rules-based strategies, focusing on individual features (whether CVSS, published exploit or reference tags) as a decision point yields inefficient remediation strategies. This holds true for all of the individual multi-word expressions. And as with all other strategies, the machine learning model using the full set of multi-word expressions out-performs the simple strategies with a better balance between efficiency and coverage, and with less effort.

The 'memory corruption' and 'buffer overflow' strategies likely enjoy higher (or approximately equal) efficiency for lower coverage possibly because it is these vulnerabilities that offer the most attractive targets for malicious actors. Interestingly, if it were true that buffer overflow vulnerabilities were also the most severe (in terms of impact 'and' exploitability), then one might expect there to be a strong correlation between these two strategies and the *CVSS 9 or 10* strategies. Indeed, as shown in Fig. 7, there is some regional proximity of these strategies, suggesting that a rules-based approach based on any one of these approaches is approximately equivalent. Although a *CVSS 10* strategy will perform better than a *CVSS 9* or either 'memory corruption' and 'buffer overflow strategies'.

We provide more explanation and discussion of these strategies and our full model next.

Full machine learning strategy

We have seen in the rule-based models that simple heuristics do not perform well and while machine learning models based on individual classes of features outperform the rule-based approach, they can be improved with a machine learning approach using all data available. Therefore, we construct a machine learning model using the full data set outlined in Table 1, using 75 585 unique CVEs and 209 features from each vulnerability.

Table 4: Reference tagging prediction results

Strategy	Accuracy (%)	Efficiency (%)	Coverage (%)	Level of effort (#vulns)	Efficiency by chance (%)	Coverage by chance (%)
memory corruption	88.50	15.48	24.00	6480	5.53	8.60
code execution	70.80	12.25	69.50	23 729	5.53	31.40
remote	29.40	6.77	92.00	56 841	5.53	75.20
buffer overflow	87.40	13.47	23.50	7304	5.53	9.70
ML Model	88.80	23.5	45.9	14 841	5.53	19.60
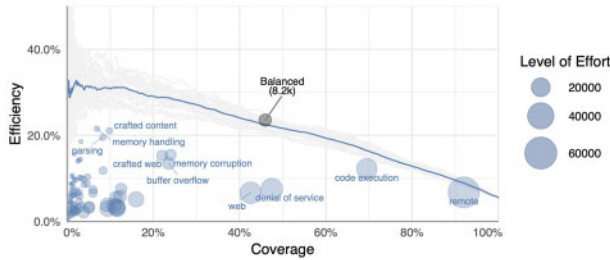
Figure 7: Reference tagging prediction results.

Note: Level of effort refers to the numbers of vulnerabilities to be patched by the given strategy.
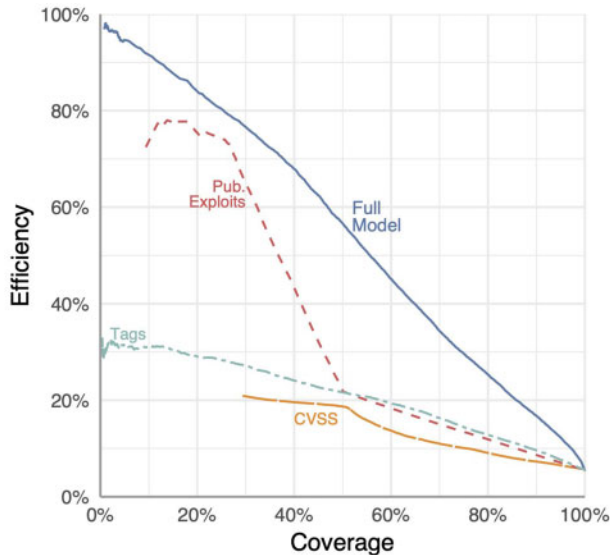
Figure 8: Prediction performance of all models.

Unsurprisingly, the coverage/efficiency curve for the full model outperforms the reduced-feature models and all of the simple heuristic models, as shown in Fig. 8.

Observe how each of the three feature-based models bear approximately the same slope and position above about 50% coverage, with the *CVSS* model performing slightly less efficiently. Below 50% coverage, on the other hand, while the *Reference Tags* and *CVSS* models continue along similar trends, the *Published Exploits* model becomes extremely efficient, even approaching the *Full Model*.

Comparing the coverage and efficiency output from each strategy has been useful, but does not provide a complete picture. The decision to remediate a vulnerability will be associated with a very specific level of effort by the firm (e.g. the effort in patching). Unfortunately, there is no single metric by which to evaluate a solution and firms are not homogenous among capabilities, resources, and risk tolerance. And so one approach is to optimize along both coverage and efficiency (which is done by maximizing the F1 measure). But what if the firm has few resources and needs to be more efficient? What if the firm wants to improve coverage at the expense of efficiency?

We consider these what-if prospects by creating a 'highly efficient' scenario (which doubles the weight placed on efficiency in the F-measure) and a 'broad coverage' scenario (which doubles the weight of coverage in the F-measure). Any emphasis on one comes at a cost of the other, and the level of effort follows. Table 5 shows the metrics for each scenario.

We now compare all heuristic strategies outlined in this document with the model output as shown in Fig. 9.

Note that almost all of the simple strategies appear in the lower fifth of the diagram (performing below 20% efficiency), suggesting that any vulnerability risk management programme relying on simple (single) characteristics (like vulnerability severity, or key phrases) will have acute efficiency problems from unnecessarily patching many vulnerabilities. The consistently low efficiency implies a slow and steady increase in the level of effort in order to reach broad coverage regardless of the simple decision point.

The Full Model, on the other hand, strictly dominates all other strategies by a considerable amount.

Discussion

In this article, we sought to test and evaluate three practical vulnerability remediation strategies using: CVSS scores, published exploits and attributes of the vulnerability (discovered through multi-word expression tagging). For each strategy, we presented the coverage and efficiency results from a simple rules-based approach, as well as a machine learning model. Finally, we integrated our rich dataset into a full machine learning model in order to construct an optimal model.

We find that applying machine learning approaches to existing feature sets strictly dominate heuristic approaches (although sometimes only by a slight margin). This is because most heuristic approaches make decisions based on one or a limited amount of data points. Heuristic approaches cannot possibly factor in all the information or interaction effects of a more advanced machine learning model.

While we constructed our tables with a discrete decision, recall that the output of each model is a probability that a particular vulnerability will (or will not) be exploited in the wild. In reality, of

Table 5: Full Model, with 3 scenarios

Scenario	Accuracy (%)	Efficiency (%)	Coverage (%)	Level of effort (#vulns)
Highly Efficient	94.5	71.4	37.3	2186
Balanced	94.5	54.6	53.0	4058
Broad Coverage	94.5	36.8	69.1	7851
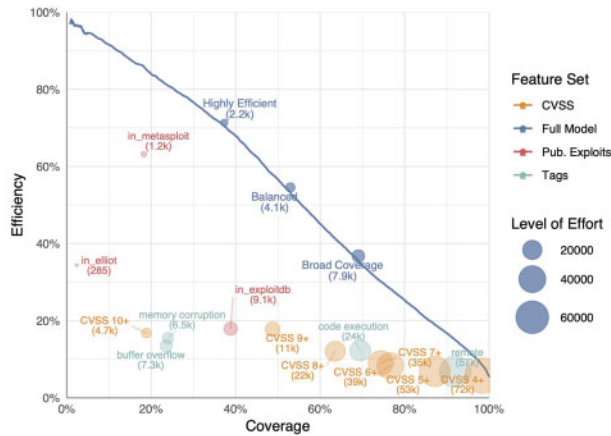
Figure 9: All models 2.

Note: the figure includes results for four separate models (feature sets), listed by colour. Also, the size of the circles represents the number of vulnerabilities scan that would need to be remediated (i.e. the level of effort) in order to satisfy that remediation strategy.

course, firms face prioritization decisions after each new vulnerability report and enterprise vulnerability scan. Therefore, in practice, we consider how firm preferences and risk tolerances many vary, which we reflect in our overall strategies of 'highly efficient', 'balanced' and 'broad coverage'. We would expect firms with a lower capacity for remediating vulnerabilities to adopt a 'highly efficient' strategy fixing some proportion of vulnerabilities based on the probability output from the model, and maximizing the benefit from their limited resources. As a firm gains experience and capacity, they can push towards a balanced approach. While it may require a greater level of effort after each scan, this maturity elevates a firm to a higher rate of coverage. Finally, the more mature organizations that have the capacity for a higher level of effort may achieve a higher rate of coverage much more quickly than the other two strategies.

While we have demonstrated how our full model substantially outperforms other strategies, we also recognize that there is much opportunity for improvement in both coverage and efficiency. Indeed, despite the marked improvement over CVSS based strategies, the model presented here still requires sacrificing efficiency for high coverage values. This is somewhat unsurprising given the highly imbalanced nature of the dataset. However, future work could focus on refining data sources and models to improve coverage. Nevertheless, regardless of the capacity and current strategy of an individual firm, a machine learning approach will reduce costs and maintain their existing level of coverage. For example, if a firm addresses vulnerabilities that have a proof-of-concept code published in Exploit DB, our model will achieve a comparable level of

coverage, 'but at one-quarter the level of effort'. If the firm is currently addressing vulnerabilities with a CVSS score of 7 and above, the 'broad coverage' strategy will achieve comparable coverage, 'but also, with one-quarter of the effort'.

So far, our focus has been on the predictive power of a machine learning approach, but we recognize the importance of interpreting each contribution individually. An initial analysis shows the relative importance of: published exploit, followed by the prevalence of the vulnerability across the hundreds of vulnerability scans collected by the security service provider. Other important features include the tag 'code execution' (such as a remote code execution flaw), followed by whether a Microsoft reference is listed in the published CVE, the CVSS Base score and finally the number of references listed in the CVE description.

Limitations

We recognize that our results and inferences are limited to the data collected by our data partners, and therefore may not be representative of all firms across the globe. Specifically, the vulnerability scan data are limited to the commercial and open-source vulnerability scanners used, as well as the kinds of software operating in these networks, and therefore, may be susceptible to missed observations of exploit data.[13]

Throughout this article, we refer to known exploits as 'exploits in the wild', but we recognize that this inference is limited to exploits that have been observed by signature-based intrusion detection/prevention (IDS/IPS) sensors and are limited in their scope. First, we do not know how many of the non-detected vulnerabilities did not have signatures generated, therefore we could never observe the exploit in the wild. Secondly, the IDS/IPS may not have had the most up to date signatures and therefore missed some known exploits. The visibility of these sensors is limited to the enterprise networks within which they were placed, and may therefore have missed exploits targeting other networks.

Also, we do not consider zero-day vulnerabilities in this research, or exploits against zero-day vulnerabilities given that these vulnerabilities are, by definition, not observable by signature-based vulnerability scanners.

We acknowledge that our data reflect a point in time, and so our knowledge, and inferences, are limited to that point. Consequently, we would miss-characterize vulnerabilities that, at the time of data collection were not exploited, but later were exploited. But given the massive class imbalance and the collection of exploited vulnerabilities (5.5% of all vulnerabilities), the influence of mislabeled target data should be minimal. Moreover, our analysis makes the assumption that the current extracted features we use to predict exploitation in the wild will continue to perform well in the future. This may not be the case. A fuller analysis of the strength and influence of each input on past and future exploits and how they change over time will make fruitful ground for future research.

---

13 This phenomenon is similar to the notion of the limitation of blacklists, as described in [25].

We attempted to make clear that our measure of risk relates to threat only (i.e. the likelihood that a vulnerability will be exploited), and does not consider all other factors that determine risk, such as the strength security controls employed by target organizations, or the value of the asset or data on that assets or the actual impact that a successful exploit may cause. Further, while there are many factors that determine the probability of exploit (such as the motivation and capability of the attacker, economic, diplomatic or political considerations), we are not able to separately identify these factors and therefore our outcome measure is simply whether an exploit was observed (or not).

In addition to efficiency and coverage, we considered a third measure of evaluation called 'level of effort' which represented the number of vulnerabilities that the organization would have to patch in order to satisfy that strategy. We realize that this level of effort is notional, and does reflect the actual number of vulnerabilities to be patched for every firm, but instead reflects the number of vulnerabilities that a hypothetical firm would have to patch, if it contained as a baseline the total number of vulnerabilities included in our dataset (76k). While this is a useful number to track in order to gauge the relative effort between strategies, it is not reflective of the total effort for a firm for a number of reasons. First, the cost of applying patches is likely not constant across firms and vulnerabilities. Certainly, firms will have varying capabilities of acquiring, testing and deploying patches across their internal and external enterprise networks. And firms vary in their level of effort required to apply a given patch, in terms of testing and applying throughout their network. Some firms may have 10 instances of a given vulnerability, while another firm may have 10 000 instances. Moreover, vendors seldom release a patch for a single vulnerability, but often bundle many fixes up into a single 'patch'. And so, our results may underestimate the economies of scale that are achieved when firms apply bundled patches.

In regard to modelling techniques, given the sparseness of some variables and relatively small base rate of exploits (upon which model identification is achieved), proper prediction and estimation of all features may be limited. We therefore tried to address this by dropping those features which were excessively sparse, and remarking on this where appropriate.

Conclusion

The ability to effectively assess the risk of a vulnerability, to understand how prevalent it is, and to make inferences about the risk it poses to either a single firm, or to sectors of critical infrastructure, is an extraordinarily important matter. Indeed, it is one of the foundations of information security risk management, domestic protection of critical infrastructure, and even national security. And it is still unsolved.

Evidence-based vulnerability risk assessment, such as the research conducted in this article, has the power to drive better patching decisions by software vendors who are making their own economic and political decisions about where to prioritize resources for developing and issuing patches. The better information vendors have about the actual risk to their customers, the sooner they can protect us.

Our results also provide an opportunity to improve the existing CVSS standard, and specifically the temporal metric group, in order to produce a better risk score. While the CVSS base score reflects severity only, the CVSS standard also specifies temporal metrics which reflect properties that change over time, such as the threat and exploitability of a vulnerability. As such, the exploit prediction results in this research could be used to augment the overall CVSS framework.

Our results also are relevant to federal policy makers such as the DHS which is charged with protecting US federal networks, and support protection of US critical infrastructure. Our results provide an opportunity for DHS to integrate evidence-based learning into their threat reports (from CISA/US-CERT) in order to provide improved measures of risk (and impact) to their constituents. Better information about vulnerability exploits can also be leveraged by them to disseminate throughout their Automated Information Sharing (AIS) network in order to better protect subscribers.

In addition, we believe this research can inform the White House National Security Council (NSC), as well as other country's governing bodies during deliberations on the use or restriction of zero-day vulnerabilities as part of a Vulnerability Equities Process (VEP) [26]. Central to the discussions of 'defensive equities' are questions about the prevalence of vulnerabilities in, and therefore risks to, the country's citizens and businesses.

Finally, we believe this research will help crystalize the value of information sharing between organizations. Too often information sharing practices and agencies have struggled to demonstrate real value and prove actual effectiveness to their users. Individual organizations, themselves, are stymied by legal, bureaucratic and technical obstacles. Our results, therefore, provide justification for sharing very tangible and practical kinds of data (namely, evidence of exploits used in the wild), understanding how they can be analysed, and, most importantly, understanding what the results mean and what to do about them. The results further help characterize the value of the hundreds of indicators of compromise (IoCs) that follow from incident reports, relative to outcome variables like exploit in the wild.

In sum, despite the small but growing body of literature on exploit prediction, firms and government agencies have been unable to practically integrate evidence-based wisdom into their decision-making processes. We sincerely hope that this, and other similar research, can better help provide practical insights into this critical problem of assessing risk.

Acknowledgements

References

[1]. Dey D, Lahiri A, Zhang G. Optimal policies for security patch management. *INFORMS J Comput* 2015;**27**:462–77.

[2]. Beattie S, Arnold S, Cowan C, *et al*. Timing the application of security patches for optimal uptime. *In LISA* 2002;**2**:233–42.

[3]. Allodi L, Massacci F. Comparing vulnerability severity and exploits using case-control studies. *ACM Trans Informat Syst Security* 2014;**17**: 1–20.

[4]. Mehran B, Saul LK, Savage S, *et al. Beyond Heuristics: Learning to Classify Vulnerabilities and Predict Exploits*. 2010. http://cseweb.ucsd. edu/~saul/papers/kdd10_exploit.pdf (20 January 2019, date last accessed).

[5]. Sabottke C, Suciu O, Dumitras T. Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits. In: *USENIX Security Symposium*, pp. 1041–56. 2015.

[6]. Kannan K, Telang R. Market for software vulnerabilities? Think again. *Manag Sci* 2005;**51**:726–40.

[7]. Ransbotham S, Mitra S, Ramsey J. Are markets for vulnerabilities effective? *Mis Quart* 2012;**36**:43–64.

[8]. Cavusoglu H, Cavusoglu H, Raghunathan S. Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. *IEEE Trans Software Eng* 2007;**33**:171–85.

[9]. Arora A, Telang R, Xu H. Optimal policy for software vulnerability disclosure. *Manag Sci* 2008;**54**:642–56.

[10]. August T, Dao D, Kim K. Market segmentation and software security: pricing patching rights. *Manag Sci* 2019.

[11]. August T, Tunca TI. Who should be responsible for software security? A comparative analysis of liability policies in network environments. *Manag Sci* 2011;**57**:934–59.

[12]. Cavusoglu H, Cavusoglu H, Zhang J. Security patch management: share the burden or share the damage?. *Manag Sci* 2008;**54**:657–70.

[13]. Münz G, Li S, Carle G. Traffic anomaly detection using k-means clustering. In: *GI/ITG Workshop MMBnet*, pp. 13–14. 2007.

[14]. Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Computer Commun Rev* 2004;**34**:219–30.

[15]. Garcia-Teodoro P, Diaz-Verdejo J, Maciá-Fernández G, *et al.* Anomaly-based network intrusion detection: techniques, systems and challenges. *Computers & Security* 2009;**28**:18–28.

[16]. MICHEL Edkrantz, Said A. Predicting exploit likelihood for cyber vulnerabilities with machine learning. Unpublished Master's Thesis, Chalmers University of Technology Department of Computer Science and Engineering, Gothenburg, Sweden, pp. 1–6. 2015.

[17]. Bullough BL, Yanchenko AK, Smith CL, *et al.* Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics*, pp. 45–53. ACM. 2017.

[18]. Rose S, Engel D, Cramer N, *et al.* Automatic keyword extraction from individual documents. In: *Text Mining: Applications and Theory*, pp. 1–20. 2010. 10.1002/9780470689646.ch1.

[19]. NIST. *Guide for Conducting Risk Assessments SP 800-30, National Institute of Standards and Technology.* US Department of Commerce. 2012.

[20]. Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System, KDD '16, San Francisco, CA, 2016. https://arxiv.org/abs/1603.02754. (16 February 2019, date last accessed).

[21]. Friedman JH. Greedy function approximation: a gradient boosting machine. *Annals Stat* 2001;1189–232.

[22]. Kubat M, Matwin S. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In: *Fourteenth International Conference on Machine Learning.* 2000.

[23]. Chinchor N. MUC-4 evaluation metrics. In: *Proceedings of the Fourth Message Understanding Conference*, pp. 22–29. 1992. 10.3115/1072064.1072067.

[24]. PCI. Approved Scanning Vendors Program Guide, v3.1, PCI Security Standards Council. 2018. https://www.pcisecuritystandards.org/documents/ASV_Program_Guide_v3.1.pdf.

[25]. Metcalf L, Spring J. Everything You Wanted to Know About Blacklists but Were Afraid to Ask. 2019. Resources.sei.cmu.edu. Retrieved 13 October 2019, from https://resources.sei.cmu.edu/library/asset-view.cfm? assetid=83438.

[26]. Romanosky S. Developing an Objective, Repeatable Scoring System for a Vulnerability Equities Process, *Lawfare Blog*, 2019. https://www.lawfareblog.com/developing-objective-repeatable-scoring-system-vulnerability-equities-process (19 February 2019, date last accessed).