Proactive Identification of Cybersecurity Threats Using Online Sources

by

Mohammed Almukaynizi

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2019 by the
Graduate Supervisory Committee:

Paulo Shakarian, Chair
Dijiang Huang
Ross Maciejewski
Gerardo I. Simari

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

Many existing applications of machine learning (ML) to cybersecurity are focused on detecting malicious activity already present in an enterprise. However, recent high-profile cyberattacks proved that certain threats could have been avoided. The speed of contemporary attacks along with the high costs of remediation incentivizes avoidance over response. Yet, avoidance implies the ability to predict—a notoriously difficult task due to high rates of false positives, difficulty in finding data that is indicative of future events, and the unexplainable results from machine learning algorithms.

In this dissertation, these challenges are addressed by presenting three artificial intelligence (AI) approaches to support prioritizing defense measures. The first two approaches leverage ML on cyberthreat intelligence data to predict *if* exploits are going to be used in the wild. The first work focuses on *what* data feeds are generated after vulnerability disclosures. The developed ML models outperform the current industry-standard method with F1 score more than doubled. Then, an approach to derive features about *who* generated the said data feeds is developed. The addition of these features increase recall by over 19% while maintaining precision. Finally, frequent itemset mining is combined with a variant of a probabilistic temporal logic framework to predict *when* attacks are likely to occur. In this approach, rules correlating malicious activity in the hacking community platforms with real-world cyberattacks are mined. They are then used in a deductive reasoning approach to generate predictions. The developed approach predicted unseen real-world attacks with an average increase in the value of F1 score by over 45%, compared to a baseline approach.

To my son, Fahd

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor Dr. Paulo Shakarian for his experienced guidance, tremendous assistance, and continuous encouragement throughout this journey; without which, this dissertation would not have been possible. I am extremely grateful to my committee members, Dr. Huang, Dr. Maciejewski, and Dr. Simari for their insightful comments and constructive feedback, which helped better improve my dissertation. Special thanks go to Dr. Simari for the fruitful scientific collaboration during the last year.

I have also had the pleasure of collaborating with multiple researchers during my Ph.D. I would like to thank Jana Shakarian, Dr. Kristina Lerman, Nazgol Tavabi, Palash Goyal, Malay Shah, Malav Shah, Krishna Dharaiya, Manoj Senguttuvan, Alexander Grimm, Dipsy Kapoor, and Timothy Siedlecki. Thank you to my colleagues in the Cyber-Socio Intelligent Systems (CySIS) Lab at ASU—Dr. Eric Nunes, Dr. Elham Shabani, Vivin Paliath, Soumajyoti Sarkar, Ericsson Marin, Hamidreza Alvari, Abhinav Bhatnagar, Ashkan Aleali, and Ruocheng Guo—for productive discussions and for enabling such enjoyable environment.

Most of all, my deepest thanks go to every member of my lovely family for being the enduring source of inspiration for me to pursue my Ph.D. Words cannot express my gratitude to my parents, Basil and Suaad, for showering me with love, and putting me through the best education possible. I would not have gotten through this doctorate if it was not for them. My lovely wife, Sumayyah, has made incredible sacrifices over the years of this journey. Without her immense support and understanding, I would not have made it this far.

Finally, this dissertation would not have been possible without funding from King Saud University (Riyadh, Saudi Arabia), and the Office of Naval Research, grant N00014-15-1-2742 and NEPTUNE program. I would also like to thank Cyber Re-

connaissance, Inc. (CYR3CON) for supplying the cyberthreat intelligence data. Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the funding agencies.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Proactive identification of cyberattacks is critical for prioritizing the process of deploying security measures and allocating resources. Being proactive requires the ability to predict if and when cyberattacks will likely occur. To do so, two environments need to be taken into consideration: the defender's environment and the attacker's environment. In practice, the dominant viewpoint of cyber-defense solely focuses on the defender's environment. For example, consider the current tools and systems used for quantifying, monitoring, and managing cyber-risk. On the one hand, most tools and systems for quantifying cyber-risk only consider technical characteristics of the defender's networked assets and their vulnerabilities, such as the number of public-facing hosts, the number of vulnerable software products they run, and the ease of exploiting these software vulnerabilities. These tools use a wide range of data sources, such as security advisory databases, software vendor websites, penetration testing tools, and vulnerability databases. On the other hand, most tools and systems for cyber-defense focus on detecting abnormalities already present in the defender's systems and networks, such as intrusion detection and spam filtering. This viewpoint to cybersecurity has a main shortcoming: the lack of a holistic consideration of the attacker's activity, a key aspect to proactively identify *if* and *when* cyber-attacks will occur. Fortunately in the recent few years, significant interest has grown towards tools, systems, and analyses for understanding and monitoring the hacker activity using cyberthreat intelligence gathered from the hacking community online platforms. This trend arose as a response to the growing broad realization of the importance of the attacker's role.

In addition to organizations, cybersecurity has lately become a common concern to public individuals. Part of that is due to the large-scale, highly-destructive cybersecurity incidents that took place in the past few years. For example, the NotPetya ransomware, first identified in June 2017, exploits a vulnerability in various Microsoft operating systems (CVE-2017-0144),[1] and it is believed to have resulted in an over $10 billion total loss in damage [2]. Also, the Equifax data breach, due to an exploit to an Apache Struts vulnerability (CVE-2017-5638),[2] in August 2017, affected over 140 million customers with a direct cost of at least $575 million [3]. The commonality across these incidents is the use of exploits to known software vulnerabilities that had patches released by the software vendors a long time before attacks are observed. These attacks would have been avoided if the victims had installed the patches early enough. Yet, deploying all patches as they become available is impossible due to the increasing number of patches that are released every day and the high cost of deploying patches (e.g., some hosts need to be taken offline to install patches). Therefore, prombtly patching vulnerabilities that are at high risk of being exploited could not be easy without the ability to predict if and when exploits will be circulated in the wild.

Within the research community, most data-driven studies leveraging machine learning (ML), and artificial intelligence (AI) in general, for cybersecurity applications are primarily focused on developing mechanisms to improve the detection of malicious behavior (e.g., network intrusion detection [4, 5, 6], malware detection [7, 8, 9], phishing attack detection [10, 11], botnet detection [12, 13]). Although these studies significantly improved the performance of traditional signature-based tools of cyberthreat detection, they only detect abnormalities that already exist at the systems being de-

[1] `https://nvd.nist.gov/vuln/detail/CVE-2017-0144`

[2] `https://nvd.nist.gov/vuln/detail/CVE-2017-5638`

fended. Other studies developed mechanisms analyzing human behavior, including hacker engagement, to support a better understanding of the cybersecurity landscape (e.g., cyber forensics, deception, and attribution [14, 15], analysis of cybercrime on darkweb[3] and deepweb[4] sites (D2web) [16, 17, 18, 19], and understanding the impact of adversarial ML [20, 21]). However, these studies do not predict cyberthreats.

The literature that is most related to this dissertation is the work on predicting cyberthreats (e.g., predictive domain and IP blacklisting [22, 23, 24, 25], predicting enterprise-targeted outsider attacks [26, 27, 28], and predicting exploits in the wild [29, 1, 30, 31, 32]). Predicting cyberthreat is a notoriously difficult task due to high rates of false positives, difficulty in finding data that is indicative of future events, and the unexplainable results from machine learning algorithms. For example, Soska and Christin [22] developed a ML-based approach that predicts whether a given website will turn malicious in the future using features such as webpage structure, content, and traffic statistics. The approach achieved a true positive rate of 66% and a false positive rate of 17%, making its utility questionable. Moreover, Bullough et al. [30] discussed the current methods that use social media (e.g., Twitter[5]) to predict exploits in the wild. Due to the poor performance, the authors noted that such methods would not be useful in practice for patch prioritization.

The central objective of this dissertation is to proactively identify cyberthreats that are likely to be leveraged by malicious hackers—in particular, we want to identify if and when attacks are likely to occur in the future. The developed approaches are designed to support prioritizing the daily cyber-defense tasks. In all parts of this

[3] "Darkweb" refers to the anonymous communications offered by encrypted networks, such as "Tor".

[4] "Deepweb" refers to Internet websites that are not indexed and have restricted access, such as "invite-only" Web forums.

[5] `https://twitter.com`

dissertation, we empirically show the utility of the developed approaches by using real-world hacker and security incident data. This dissertation is organized as follows:

- **Chapter 2: Proactive identification of exploits in the wild through vulnerability mentions online.** This chapter focuses on (1) discovering the current sources of threat intelligence that are indicative of future availability of exploits in the wild (a key principle to reducing false positives), (2) demonstrating the viability of the proposed models under deployment settings, and (3) providing empirical analysis about the likelihood of exploitation for vulnerabilities when they are mentioned on each data source, considering the availability of such information relative to the time an exploit is found in the wild. Our results outperform the best performance achieved by the widely-used standard severity scoring system (a.k.a., CVSS[6]), with F1 score more than doubled, demonstrating the viability of our approach for practical use.

- **Chapter 3: Combining social network analysis with ML techniques to predict exploits in the wild.** This part focuses on (1) introducing an approach that uses the users' post/reply activities to generate a directed social graph, (2) demonstrating the value of the approach in supporting predictive features to ML models, and (3) providing observations from the connectivity measures of the individuals who authored postings preceding exploits in the wild. We specifically demonstrate that social network data improves the value of F1 score by about 6% compared to the highest F1 score achieved by a model that does not use these features.

- **Chapter 4: A rule learning-based approach to predict organization-targeted external threats.** In this chapter, we develop an approach that

[6] `https://www.first.org/cvss`

combines ML with knowledge representation and reasoning to learn rules with indicators of certain future cyberthreats, such as spam email campaigns. The rules are then used in a detective approach to generate actionable warnings. This approach was integrated into an intelligent system that submits warnings to security operations centers [28]. Our key goal is to use reasoning to generate *improved*, *timely*, *transparent*, and *actionable* predictions that are understandable by a human analyst. Two datasets are used in this work: (a) a dataset of historical records of attack attempts that were recorded and noted from the network traffic logs of a defense-industrial-base organization, and (b) an online access to a commercially-available streaming API supplying data about D2web activity. Our approach produced predictions with improved performance compared to statistical baseline predictive models.

- **Chapter 5: Conclusion and future work.** This chapter provides a summary of the main ideas and results presented in this dissertation. It concludes with some future directions to extend the work developed in this dissertation.

Chapter 2

PROACTIVE IDENTIFICATION OF EXPLOITS IN THE WILD THROUGH VULNERABILITY MENTIONS ONLINE

2.1 Introduction

Software vulnerabilities are weaknesses in software products that can be exploited by attackers to compromise the confidentiality, integrity, or availability of the system hosting these products and cause harm [33]. An exploit is a piece of code or a chunk of data that modifies the functionality of a system using an existing vulnerability [34]. Today, vulnerability exploitation is perhaps the most common method used by hackers to compromise these three objectives of cybersecurity. Defending against vulnerability exploitation is a difficult task that is widely-recognized by cybersecurity researchers and practitioners [35, 31, 1]. Part of this comes from the difficulty in keeping pace with the ever-increasing number of vulnerabilities that are publicly disclosed. Moreover, malicious actors continue to smartly target a smaller fraction of published vulnerabilities. Therefore, identifying if a vulnerability will likely be exploited by hackers is key for a holistic security defense.

The National Institute of Standards and Technology (NIST)[1] maintains a comprehensive list of publicly disclosed vulnerabilities in the National Vulnerability Database (NVD).[2] The NVD also provides information regarding targeted software products (CPE),[3] Common Vulnerability Scoring System (CVSS)[4] that evaluates the vulnera-

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

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

[3]https://nvd.nist.gov/cpe.cfm

[4]https://nvd.nist.gov/vuln-metrics/cvss

bilities in terms of exploitability and impact, and the date a vulnerability was published. In 2018 alone, more than 16,500 vulnerabilities have been disclosed in the NVD, an increase of over 13% from 2017, and of over 150% from 2016.

Once vulnerabilities are publicly disclosed, the likelihood of their exploitation raises drastically [36]. With limited resources, organizations often look to prioritize which vulnerabilities to patch. They do so by assessing the impact on their assists and reputation if vulnerabilities are exploited. In this chapter, the exploits that have been used to target systems in real-world attacks are referred to as *real-world exploits*. On the other hand, a *proof-of-concept exploit (PoC)* is typically developed to verify a reported software flaw in order to reserve a CVE number or illustrate how a vulnerability can be exploited. PoCs generally require additional functionalities to be weaponized and be useful by malicious hackers. While the presence of a PoC is a leverage for hackers, it does not necessarily imply exploitation in real-world attacks.

To be on the safe side, standard risk assessment systems such as the CVSS score, Microsoft Exploitability Index,[5] and Adobe Priority Rating[6] report many vulnerabilities to be severe. The foregoing systems are broadly viewed as guidelines to supply vulnerability management teams with tools that help in patch prioritization. One commonality across those systems is that they rank vulnerabilities based on historical attack patterns that are relevant to the technical details of vulnerabilities that are evaluated, rather than what malicious hackers discuss and circulate in the underground forums and marketplaces. This does little to alleviate the problem since the great majority of the highly-rated vulnerabilities will never be attacked [35].

In practice, current methods of patch prioritization appear to fall short [37, 35]. Verizon reported that over 99% of breaches are caused by exploits to known vul-

[5]https://technet.microsoft.com/en-us/security/cc998259.aspx

[6]https://helpx.adobe.com/security/severity-ratings.html

nerabilities [37]. Cisco also reported that "the gap between the availability and the actual implementation of such patches is giving attackers an opportunity to launch exploits," [38]. For some vulnerabilities, the time window to patch the system is very small. For instance, exploits targeting the Heartbleed[7] bug in the OpenSSL[8] cryptographic software library were detected in the wild 21 hours after the vulnerability was publicly disclosed [39]. Hence, organizations need to efficiently assess the likelihood that a vulnerability is going to be exploited in the wild, while keeping the false alarm rate low.

Only a small fraction (less than 3%) of vulnerabilities disclosed in the NVD are exploited in the wild [32, 40, 35, 41, 1, 42]—a result confirmed in this chapter. In addition, previous studies have found that the CVSS score provided by NIST is not an effective predictor of exploitation [35, 1, 31]. It has previously been proposed that other methods such as the use of social media [43, 1], darkweb markets [16, 44, 45], and certain white hat[9] websites like Contagio,[10] would be suitable alternatives. However, these approaches have their limitations. For instance, methodological concerns on the use of social media for exploit prediction were recently raised in [30]; and data feeds for exploit and malware were limited to single sites and were only used for analysis to provide insights on economic factors of those sites [44, 45]. While other studies demonstrate the viability of data collection, they do not quantify the results of prediction [16, 43].

After reviewing the literature, including studies on data gathered from darkweb and deepweb (D2web) [46, 47, 35, 16, 48], conducting analyses on data feeds collected

[7] `http://heartbleed.com`

[8] `https://www.openssl.org`

[9] White hat, or white-hat, is an Internet term that is often used to refer to ethical hackers and penetration testers.

[10] `http://contagiodump.blogspot.com`

from various online sources (e.g., SecurityFocus[11] and Talos[12]), and after over one hundred interviews with professionals working for managed security service providers (MSSPs),[13] firms specializing in cyber-risk assessment, and security specialists working for managed (IT) service providers (MSPs), three data sources have been identified that can represent the current threat intelligence used for vulnerability prioritization: (1) ExploitDB (EDB),[14] contains information on PoC exploits for vulnerabilities provided by security researchers from various blogs and security reports; (2) Zero Day Initiative (ZDI),[15] is curated by a commercial firm called TippingPoint and uses a variety of reported sources focusing on disclosures by various software vendors and their security researchers; and (3) a collection of information scraped from over 120 sites on the D2web from a system introduced in [18, 49] and currently maintained by the cybersecurity firm CYR3CON.[16] The intuition behind each of these feeds was not only to utilize information that was aggregated over numerous related sources, but also to represent feeds commonly used by cybersecurity professionals.

This chapter focuses on vulnerabilities that have been publicly disclosed in 2015 or 2016. The presented models employ supervised machine learning techniques using Symantec[17] attack signatures as ground truth. Specifically, in this chapter,

[11] `http://www.securityfocus.com`

[12] `https://www.talosintelligence.com/vulnerability_reports`

[13] An MSSP is a service provider that provides its clients with tools that continuously monitor and manage wide range of cybersecurity-related activities and operations, which may include threat inelegance, virus and spam blocking, and vulnerability and risk assessment.

[14] `https://www.exploit-db.com`

[15] `http://www.zerodayinitiative.com`

[16] `https://www.cyr3con.com`

[17] `https://www.symantec.com`

- The utility of the proposed machine learning models in predicting exploits in the wild is demonstrated with true positive rate (TPR)[18] of 90% while maintaining a false positive rate (FPR)[19] of less than 15%. In addition, the proposed model is compared to a recent benchmark model that utilized online mentions for exploit prediction [1]. The proposed model achieves a significantly higher precision while maintaining recall. The robustness of the presented model against various adversarial data manipulation strategies is also discussed.

- Using vulnerability mentions on EDB, ZDI, and D2web, the increase in the vulnerability exploitation likelihood over vulnerabilities only disclosed on the NVD is studied. We provide results demonstrating that the likelihood of exploitation given vulnerability mention on EDB (9%), ZDI (12%) and D2web (14%) as compared to the NVD (2.4%). The availability of such information relative to the time an exploit is found in the wild is also studied.

- Exploited vulnerabilities are analyzed based on various other features derived from these data sources, such as the language used. Apparently, Russian language sites on the D2web discussing vulnerabilities are 19 times more likely to be exploited than random vulnerabilities, more likely than vulnerabilities described on websites in any of the other languages. Additionally, the probability of exploitation is investigated in terms of both data source and software vendor.

The rest of the chapter is organized as follows. Related work is discussed in Section 2.2. Section 2.3 outlines some challenges related to the problem addressed in this chapter. Section 2.4 provides an overview of the presented exploit prediction

[18] A metric that measures the proportion of exploited vulnerabilities that are correctly predicted from all exploited vulnerabilities.

[19] A metric that measures the proportion of non-exploited vulnerabilities that are incorrectly predicted as being exploited from the total number of all non-exploited vulnerabilities.

model and describes the data sources used. Vulnerability analysis is discussed in Section 2.5. In Section 2.6, experimental results are provided for predicting the likelihood of vulnerability exploitation. The robustness of the presented machine learning model against adversarial data manipulation is demonstrated in Section 2.7. A discussion on the viability of the proposed model and the cost of misclassification is provided in Section 2.8.

2.2 Related Work

Predicting cybersecurity events is one of those domains that have recently received a growing attention [50, 22, 51, 52, 40]. While important for prioritizing defense measures, only little work in this line of research has been proposed so far compared to work proposed on detecting cyberthreats that are already present in an organization's network. Previous studies have attempted to address this problem using both standard scoring systems (in particular, the CVSS base score) and machine learning techniques. An approach to predict the likelihood that a software contains a yet-to-be-discovered vulnerability was proposed in [53]. In this study, data feeds from the NVD were leveraged to predict the time a next vulnerability will be discovered in a given software product. The results showed a poor predictive capability of the NVD data. The CVSS version 2.0 is a poor indicator of predicting whether a vulnerability will be exploited, as demonstrated by [35]. The authors' analysis showed that deciding to patch a vulnerability because of a high CVSS score is equivalent to randomly guessing which vulnerability to patch. Yet, integrating information about whether a PoC exploit is available should significantly improve the accuracy of such a decision [35].

Our approach closely resembles previous work on using publicly disclosed vulnerability information as features to train machine learning models to predict whether a

given vulnerability will be exploited. Bozorgi et. al. [52] proposed a model that engineered features from two online sources, namely Open Source Vulnerability Database (OSVDB)[20] and the NVD to predict whether PoCs will be developed for a particular vulnerability. In their data, 73% of the vulnerabilities are labeled as exploited, a percentage that are orders of magnitude higher than what is reported in recent literature (less than 3%) [40, 35, 41, 1]—a result confirmed in this chapter. The reason behind this is that the authors considered the existence of PoC as weaponized exploitation, which is incorrect for most cases. Using this assumption, and using a support vector machine classifier, their approach predicts whether vulnerabilities will have PoCs available, a problem that is different from the one studied in this chapter. A similar technique was employed by [40] using the NVD as the data source. They used Exploit-DB as ground truth, with 27% vulnerabilities labeled as exploited (having PoC exploits). High accuracy was achieved on a balanced dataset. Our analysis aims to predict vulnerabilities that will be weaponized for real-world attacks and not just have PoC exploits available.

Building on the work on using publicly disclosed vulnerabilities, Sabottle et al. [1] looked into predicting exploitation of vulnerabilities discussed on Twitter.[21] In that study, Twitter short posts, called tweets,[22] that had references to CVE numbers were collected. A linear SVM classifier is trained for prediction. As the source of ground truth data, Symantec attack signatures are used to label positive samples. Comparing to previous predictive studies, even though the authors of [1] maintained the class ratio of 1.3% vulnerabilities exploited, they used a resampled and balanced dataset to report the results. Additionally, the temporal aspect (training data should precede

[20] `https://blog.osvdb.org`

[21] `https://twitter.com`

[22] In Twitter platform, a post is called a *tweet*, and each tweet is limited to 280 characters.

testing) of the tweets is not maintained while performing the experiments. This temporal intermixing causes future events to influence the prediction of past events, a practice that is shown to lead to unrealistic predictions [30]. In this chapter, the class imbalance and the temporal aspect are respected while reporting the results.

Particularly for the cybersecurity incident prediction problem, the impact of adversarial interference for hackers aiming to poison and evade the machine learning models has been discussed. Hao et al. [54] proposed a prediction model to predict the web domain abuse based on features derived from the behavior of users at the time of registration. They study different evading strategies an attacker can use and demonstrate that evading attempts are expensive to attackers, and their model's reliance on different sets of features allows for limiting the decrease in false positive rate. Other researchers have also discussed the robustness of their models against adversarial attacks such as [50, 1]. In this chapter, we run simulation experiments and demonstrate that the impact is very limited, as discussed in Section 2.7.

2.3 Background

2.3.1 Supervised Learning Approaches

A brief explanation of the machine learning approaches that are used in this study is provided below:

Support Vector Machine (SVM). Support vector machine (SVM) was proposed by Vapnik [55]. SVM works by finding a separating margin that maximizes the geometric distance between classes (in our case exploited and not exploited). The separating margin is termed as hyperplane. When a separating plane cannot be found to distinguish between the two classes, the SVM cost function includes a regularization

penalty and a slack variable for the misclassified samples. Varying these parameters, trade-off between precision and recall can be observed.

Naïve Bayes Classifier (NB). Naïve Bayes is a probabilistic classifier, which uses Bayes theorem with independent attribute assumption. During training, we compute the conditional probabilities of a sample of a given class having a certain attribute. We also compute the prior probabilities for each class, i.e., fraction of the training data belonging to each class. Naïve Bayes assumes that the attributes are statistically independent; therefore the likelihood for a sample S represented with a set of attributes a associated with a class c is given as $\Pr(c|S) = P(c) \times \prod_{i=1}^{d} \Pr(a_i|c)$.

Bayesian Network (BN). BN is a generalized form of NB such that not all features are assumed to be independent. Instead, variable dependencies are modeled in a graph learned from the training data.

Decision Tree (DT). Decision tree is a hierarchical recursive partitioning algorithm. We build the decision tree by finding the best split attribute, i.e., the attribute that maximizes the information gain at each split of a node. To avoid overfitting, the terminating criteria is set to less than 5% of the total samples.

Logistic Regression (LOG-REG). Logistic regression classifies samples by computing the odds ratio. The odds ratio gives the strength of the association between the attributes and the class.

2.3.2 Challenges

Previous work has pointed out methodological issues with exploit prediction studies [30]. We also note that there is a balance between ensuring an evaluation is conducted under real-world conditions and conducting an evaluation on an adequate sample size. Some of those challenges are reviewed below.

Class imbalance. As mentioned earlier, evidence of real-world exploits is found for only around 2.4% of the reported vulnerabilities. This skews the distribution towards one class in the prediction problem (i.e., *not exploited*). In such cases, standard machine learning approaches favor the majority class, leading to poor performance on the minority class. Some of the prior work in predicting the likelihood of exploitation considered the existence of PoCs as an indicator of real-world exploit weaponization, which substantially increases the number of exploited vulnerabilities in the studies adopting this assumption [52, 40, 30]. However, out of the PoC exploits that are identified, only a small fraction are ever used in real-world attacks [41]—a result confirmed in this chapter (e.g., only about 4.5% of the vulnerabilities having PoCs were subsequently exploited in the wild.) Other prior work used class balancing techniques on both training and testing datasets and reported performance achieved using metrics like TPR, FPR, and accuracy [22, 50].[23] Resampling the data to balance both classes in the dataset leads to training the classifier on a data distribution that is highly different from the underlying distribution. The impact of this manipulation, whether positive or negative, cannot be observed when testing the same classifier on a manipulated dataset, e.g., a testing set with the same rebalancing ratio. Hence, the prediction performance of the proposed models in deployed, real-world settings is questionable. In this chapter, oversampling techniques (in particular SMOTE [56]) are examined to confirm the impact of highly imbalanced dataset used on the machine learning models. Note that the testing dataset is not manipulated because we aim to observe a performance that can be reproduced under the settings of a model running on real-world deployment (e.g., streaming predictions). Doing so, only marginal improvement is observed for some classifiers as reported in Section 2.6.2, while other

[23]Note that these metrics are sensitive to the underlying class distribution and sensitive to the ratio of class rebalancing

classifiers have shown a slightly negative impact when they are trained on oversampled dataset.

Evaluating models on temporal data. Machine learning models are evaluated by training the model on one set of data and then testing the model on another set that is assumed to be drawn from the same distribution. The data split can be done randomly or in a stratified manner, where the class ratio is maintained in both training and testing. A key aspect of the exploit prediction task is that it is time-dependent [32, 30]. Randomly splitting data violates this aspect because events that happen in the future would be used to predict events that happened in the past. Prior research has ignored this aspect while designing their experiments [1, 52]. In this work, this temporal mixing is avoided in most experiments. However, experiments with a very small sample size, in which this is not controlled, are included (this is because one of the used ground truth sources does not have date/time information). It is explicitly noted when this is the case.

Limitations of ground truth. As mentioned, attack signatures reported by Symantec are used as the ground truth of the exploited vulnerabilities, similar to previous work [1, 35]. This ground truth is not comprehensive because the distribution of the exploited vulnerabilities over software vendors is found to differ from that for overall vulnerabilities (i.e., vulnerabilities that affect Microsoft products have a good coverage compared to products of other OS vendors.) Although this source is limited in terms of coverage [1], it is still the most reliable source for labeling exploited vulnerabilities because it reports attack signatures of exploits detected in the wild for known vulnerabilities. Other sources either report whether a piece of software is malicious without proper mapping to the exploited CVE number (e.g., VirusTotal[24]) or rely on online blogs and social media sites to identify exploited vulnerabilities (e.g.,

[24] `https://www.virustotal.com`

SecurityFocus[25]). In this chapter, Symantec data is used while taking into account the false negatives. To avoid overfitting the machine learning model on this not-so-representative ground truth, we omit the software vendor from the set of examined features.

2.4 Exploit Prediction Model

Using machine learning models to address this problem has interesting security implications in terms of prioritizing which vulnerabilities need to be patched first to minimize the risk of cyberattacks. Figure 2.1 gives an overview of the proposed exploit prediction model. It consists of the following phases:

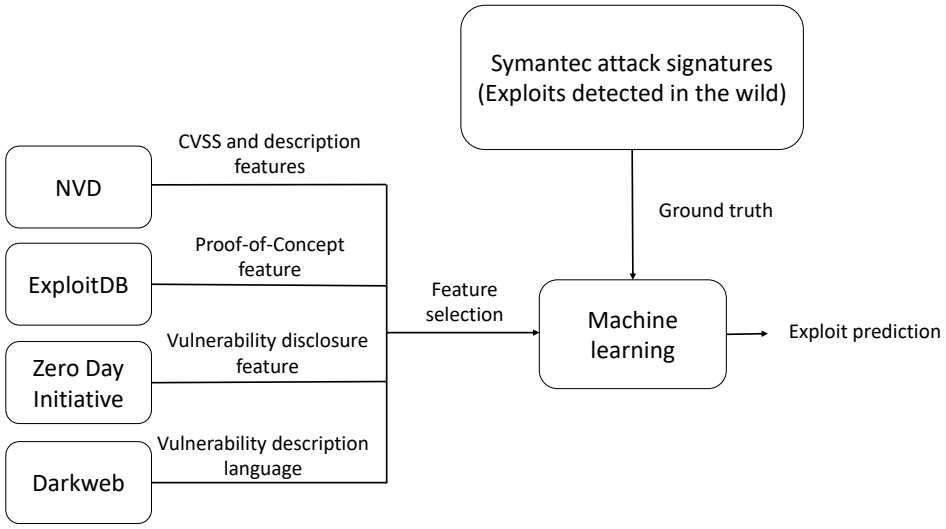


Figure 2.1: Exploit Prediction Model.

- **Data collection:** Three data sources are used in addition to the NVD. These data sources are EDB (ExploitDB), ZDI (Zero Day Initiative) and data mined

[25] `https://www.securityfocus.com`. There are many examples where attack signatures are reported by Symantec, but not reported by SecurityFocus. Also, there are vulnerabilities SecurityFocus reports as exploited, and those exist in software whose vendors are well-covered by Symantec, yet Symantec does not report them.

from D2web markets and forums, focusing on malicious hacking. Ground truth is assigned to the binary classification problem studied in this chapter using Symantec attack signatures of exploits detected in the wild. The data sources are discussed in Section 2.4.1.

- **Feature selection:** Features are extracted from each of the data sources. The features include bag-of-words features for vulnerability description and discussions on the D2web, binary features that check for the presence of PoC exploits in EDB, vulnerability disclosures in ZDI and the D2web. Additional features are also included from the NVD, including CVSS score and CVSS vector.

- **Prediction:** Binary classification is performed on the selected features to determine whether the vulnerability will likely be exploited or not. To address this classification problem, several standard supervised machine learning approaches are evaluated.

2.4.1 Data Sources

Our analysis combines vulnerability and exploit information from multiple open source databases, namely: the NVD, EDB, ZDI, and the D2web. The D2web database is obtained from an API maintained by CYR3CON. Our experiments cover vulnerabilities that were published in 2015 or 2016. Table 2.1 shows the vulnerabilities identified from each of the data sources between 2015 and 2016 as well as the number of vulnerabilities that were exploited in real-world attacks. A brief overview of each of the data sources, including ground truth, is provided below.

NVD. The National Vulnerability Database maintains a database of publicly disclosed vulnerabilities. Each vulnerability is identified using a unique CVE number. Our dataset contains 12,598 vulnerabilities. Figure 2.2 shows the disclosure of vulner-

Table 2.1: Number of Vulnerabilities (2015–2016).

Database	Vulnerabilities	Exploited	% Exploited
NVD	12,598	306	2.4%
EDB	799	74	9.2%
ZDI	824	95	11.5%
D2web	378	52	13.8%

abilities per month. At the time of data collection, there were only 30 vulnerabilities disclosed in December 2016, hence the small bar at the end of 2016. For each vulnerability, the description, the CVSS score and vector, and the publication date are collected. Organizations often use the CVSS score to prioritize which vulnerabilities to patch. The CVSS vector lists the components from which the score is computed. More details about CVSS components are provided in Section 2.4.2.

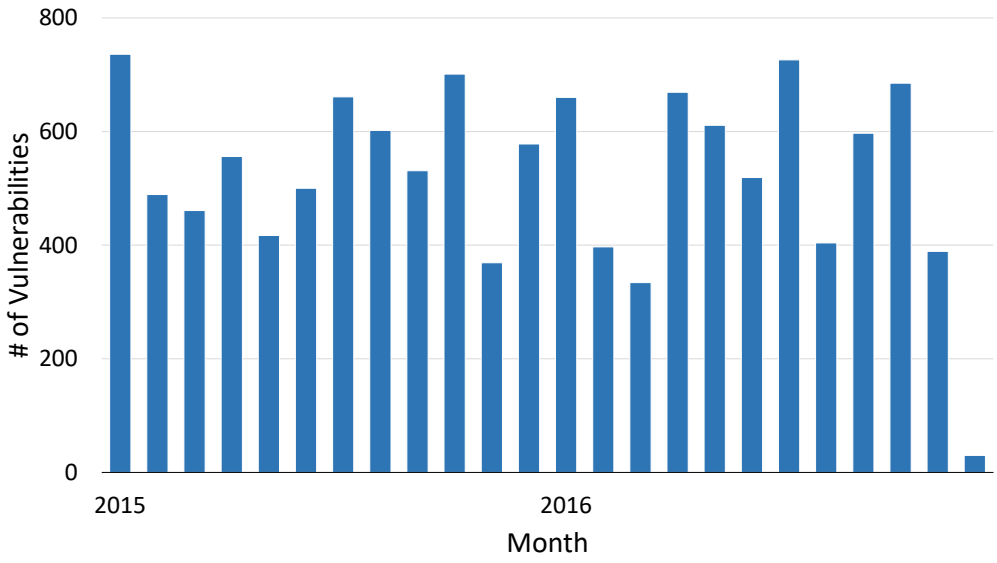


Figure 2.2: Vulnerabilities Disclosed per Month.

EDB (white hat community). The Exploit Database is an archive of PoC exploits maintained by Offensive Security.[26] PoC exploits for known vulnerabilities

[26] `https://www.offensive-security.com`

are reported with CVE numbers of target vulnerabilities. Using the unique CVE numbers from the NVD for the period between 2015 and 2016, EDB was queried to find out whether a PoC exploit is available. The availability date of PoCs has also been recorded. By querying EDB, verified PoCs have been found for 799 of the vulnerabilities studied.

ZDI (vulnerability detection community). Zero Day Initiative maintains a database of vulnerabilities that are identified and reported by security researchers. Reported software flaws are first verified by ZDI before disclosure. Monetary incentives are provided to researchers who report valid vulnerabilities. Before ZDI publicly discloses a vulnerability, the software vendors of the target products are notified and allowed time to implement patches. The ZDI database has been queried for the vulnerabilities, and 824 common CVE numbers between the NVD and ZDI were found. The publication date has also been noted.

D2web (hacker community). The data collection infrastructure maintained by CYR3CON was originally described in [18]. In this paper, the authors built a system that crawls sites on the D2web, both marketplaces and forums, to collect data relating to malicious hacking. They first identify sites before developing scripts for automatic data collection. A site is put forward to script development after it has been determined whether the content is of interest (i.e., hacking-related) and is relatively stable. The population size of a site is observed, though not much decisive power is assigned to it. While a large population is an indicator of the age and stability of a site, a small population number can be associated with higher-value information (i.e., closed forums).

D2web users advertise and sell their products on marketplaces. D2web marketplaces provide a new avenue to gather information about vulnerabilities and exploits. Forums, on the other hand, feature discussions on newly discovered vulnerabilities and

exploit kits. Data related to malicious hacking is filtered from the noise and added to a database using a machine learning approach with high precision and recall. Not all exploits or vulnerability items in the database have CVE numbers associated with them. First, the database was queried to extract all items with CVE mentions. Some vulnerabilities are mentioned in the D2web using their Microsoft Security Bulletin Number[27] (e.g., MS16-006). Every bulletin number was mapped to its corresponding CVE number, making ground truth assignment easy. These items can be products sold on markets or posts extracted from forums discussing topics relating to malicious hacking. We found 378 unique CVE mentions between 2015 and 2016 on more than 120 D2web websites, much more than what a previous work discovered [35]. We also queried the posting date and descriptions associated with all the CVE mentions including product title and description, vendor information, entire discussion with the CVE mention, author of the posts, and the topic of the discussion.

Attack signatures (ground truth). For our ground truth, we identified vulnerabilities that were exploited in the wild using Symantec anti-virus attack signatures[28] and Intrusion Detection Systems (IDS) attack signatures.[29] The attack signatures are associated with the CVE number of the vulnerability that was exploited. We mapped these CVEs to the CVEs mined from the NVD, EDB, ZDI, and the D2web. This ground truth indicates actual exploits that were used in the wild and are not just PoC exploits. For the NVD, around 2.4% of the disclosed vulnerabilities were exploited, which is consistent with previous studies. We do not have data regarding the volume and frequency of attacks leveraging the detected exploits; hence we consider all exploited vulnerabilities to have equal importance. This assumption has been adopted

[27] `https://technet.microsoft.com/en-us/security/bulletins.aspx`

[28] `https://www.symantec.com/security_response/landing/azlisting.jsp`

[29] `https://www.symantec.com/security_response/attacksignatures`

by previous works as well [35, 1]. Additionally, we define the *exploitation date* of a vulnerability as the date it was first detected in the wild. Symantec IDS attack signatures are reported without recoding the dates when they were first detected, but its anti-virus attack signatures are reported with their *exploitation date*. Between 2015 and 2016, 112 attack signatures have been reported without and 194 with their exploitation date.

2.4.2 Feature Description

We combine features from all the data sources discussed in Section 2.4.1. Table 2.2 gives a summary of these features. We now discuss each of them.

Table 2.2: Summary of Features.

Feature	**Type**
NVD and D2web description	TF-IDF on bag of words
CVSS	Numeric and categorical
D2web Language	Categorical
Presence of PoC	Binary
Vulnerability mention on ZDI	Binary
Vulnerability mention on the D2web	Binary

NVD and D2web description. The NVD description provides information on the vulnerability and what it allows hackers to do when they exploit it. D2web description often provides rich context on what the discussion is about, and is often synthesized from forums rather than marketplaces since items are described in fewer words. Patterns can be learned based on this textual content. We obtained the description of published vulnerabilities from the NVD, and we queried the D2web database for CVE mentions between 2015 and 2016. This description was appended to the NVD description with the corresponding CVE. We observed that some of

the descriptions on the D2web are in a foreign language as discussed in Section 2.5. We first translated the foreign text to English using the Google Translate API.[30] We then vectorized the text features using the *term frequency-inverse document frequency* (TF-IDF) model that is learned from the training set and used to vectorize the testing set. TF-IDF creates a vocabulary of all the words in the description. The importance of a word feature increases with the number of times it occurs, but is normalized by the total number of words in the description. This eliminates common words from being important features. We limited our TF-IDF model to the 1,000 most frequent words (using more word features has no benefit in terms of performance, however, it would increase the computational cost.)

CVSS. The NVD provides us with a CVSS score and the CVSS vector from which the score is computed, indicating the severity of each disclosed vulnerability. We used the CVSS base metric version 2.0 rather than version 3.0 because the latter is only present for a fraction of the vulnerabilities we study. The CVSS vector lists the components from which the score is computed. The components of the vector include Access Complexity, Authentication, Confidentiality, Integrity, and Availability. Access Complexity indicates how difficult it is to exploit the vulnerability once the attacker has gained access. It is defined in terms of three levels: High, Medium and Low. Authentication indicates whether authentication is required by the attacker to exploit the vulnerability. It is a binary identifier taking the values Required and Not Required. Confidentiality, Integrity, and Availability indicate what loss the system would incur if the vulnerability is exploited. They take the values None, Partial and Complete. All the CVSS vector features are categorical. We vectorize these features by building a vector with all possible categories. Then if that category is present, we insert "1", otherwise "0".

[30] `https://cloud.google.com/translate/docs`

Language. D2web feeds are posted in different languages. We found 4 languages that are used in the D2web posts that reference vulnerabilities. These languages are English, Chinese, Russian, and Swedish. Since there is a limited number of non-English postings, giving the model little chance to learn proper representation for each language, we opted to use the text translation as described. To this end, we believe translation can result in a loss of important information, although we can retain the impact of knowing the language by using it as a feature. We show analysis on the languages of D2web feeds and their variation in the exploitation rate in Section 2.5.

Presence of proof-of-concept. The presence of PoC exploits in EDB increases the likelihood of a vulnerability being exploited. We treat it as a binary feature indicating whether a PoC is present for a vulnerability or not.

Vulnerability mention on ZDI. ZDI acts similar to the NVD, i.e., both disclose software vulnerabilities. Given that a vulnerability is disclosed on ZDI, its exploitation likelihood raises. Similar to the presence of PoCs, we use a binary feature to denote whether a vulnerability was disclosed in ZDI before it is exploited.

Vulnerability mention on the D2web. We use a binary feature indicating whether a vulnerability is mentioned on the D2web.

2.5 Vulnerability and Exploit Analysis

To assess the importance of aggregating different data sources for early identification of threatened vulnerabilities, we first analyze the likelihood of exploitation given that a vulnerability is mentioned on each data source. Time-based analysis is then provided for the exploited vulnerabilities that have reported *exploitation dates* (n = 194) to show the difference in days between when vulnerabilities are exploited and when they are mentioned online. In our time-based analysis, we ignore the exploited vulnerabilities without reported dates because we cannot make any assump-

tions regarding their exploitation dates. Furthermore, we analyze our ground truth and compare it to other sources to identify the vendors of highly vulnerable software and systems. As previous works reported, Symantec reports attack signatures for vulnerabilities of certain products [35, 1]. We study the distribution of affected software vendors by vulnerabilities from each data source. We base this analysis on the vendor mentions by CPE data from the NVD. Finally, we provide a language-based analysis on the D2web data to reveal some socio-cultural factors present in the D2web that seem to affect the likelihood of exploitation.

Likelihood of exploitation. For each data source, Table 2.3 shows the vulnerability exploitation probability for the vulnerabilities mentioned in that data source. This analysis emphasizes the value of open data sources in supplementing the NVD data. As mentioned in Section 2.4.1, about 2.4% of the vulnerabilities in NVD are exploited in the wild. Hence, including other sources can increase the likelihood of correctly identifying the vulnerabilities that will be exploited.

Table 2.3: Number of Vulnerabilities, Number of Exploited Vulnerabilities, Percentage of Exploited Vulnerabilities That Appeared in Each Source, and Percentage of Total Vulnerabilities That Appeared in Each Source.

	EDB	**ZDI**	**D2web**	**ZDI ∨ D2web**	**EDB ∨ ZDI ∨ D2web**
Number of vulnerabilities	799	824	378	1,180	1,791
Number of exploited vulnerabilities	74	95	52	140	164
Percentage of exploited vulnerabilities	21%	31%	17%	46%	54%
Percentage of total vulnerabilities	6.3%	6.5%	3.0%	9.3%	14.2%

Time-based analysis. Most software systems are attacked repeatedly using vulnerabilities after exploits to such vulnerabilities have been detected in the wild [57]. As a matter of fact, vulnerabilities may take a long time between the date they are disclosed and the date they are patched by vulnerability management teams. Here,

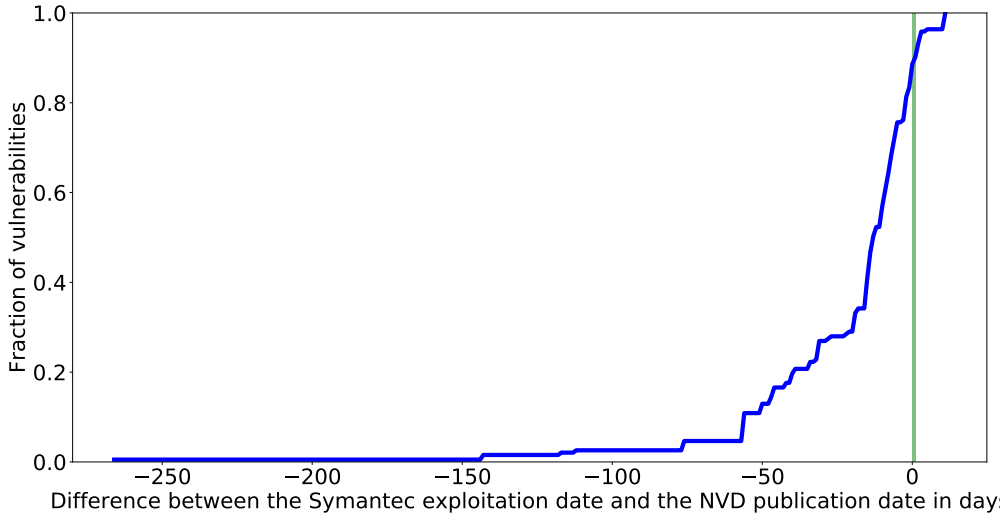


Figure 2.3: Day Difference Between CVE First Published in the NVD and the Symantec Attack Signature Date vs. the Fraction of Exploited CVEs on the NVD Reported (Cumulative).

we only analyze the population of exploited vulnerabilities that are reported with their exploitation date (194 vulnerabilities).

Figure 2.3 shows that for more than 93% of the vulnerabilities, they are disclosed by NIST before any real-world attacks are detected. In the other few cases, attacks were detected in the wild before NIST published the vulnerabilities (i.e., zero-day attacks). This could be caused by too many reasons: (1) the vulnerability information is sometimes leaked before the disclosure, (2) by the time NIST disclosed a vulnerability in the NVD, other sources have already validated and published it, then exploits rapidly started using it in real-world attacks, or (3) the attacker knew that what they were doing was successful and continued to exploit their targets until discovered [36]. Additionally, ZDI and NVD have limited variation on the vulnerability disclosure dates (median is 0 days). It is important to note that because ZDI disclosures come from the industry, reserved CVE numbers are shown earlier here than in other sources.

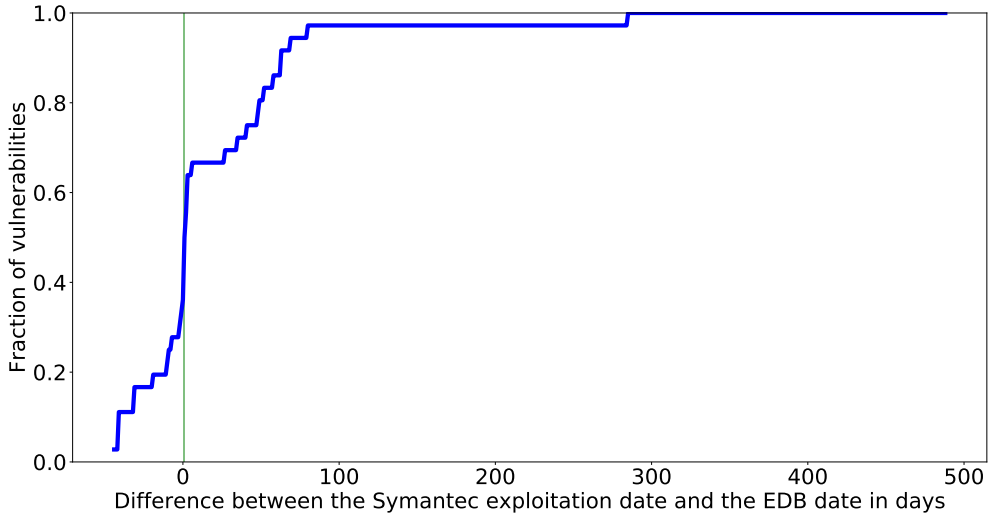


Figure 2.4: Day Difference Between the Date of Availability of PoC on EDB and the Symantec Attack Signature Date vs. the Fraction of Exploited CVEs With PoCs on EDB Reported (Cumulative).

In the case of EDB database, almost all of the exploited vulnerabilities that have PoCs archived in EDB were found in the wild within the first 100 days of the PoCs availability. Such a short period between the availability of PoCs and actual attacks in the real-world indicates that having a template for exploits (in this case PoCs) makes it easy for hackers to configure and use in real-world attacks. Figure 2.4 shows the difference in days between the availability of PoCs and *exploitation dates*.

In the case of the D2web database, more than 60% of the first-time mentions to the exploited vulnerabilities are within 100 days before or after the *exploitation dates*. The remaining mentions are within the 18 months time frame after the vulnerability *exploitation date* (see Figure 2.5).

Vendor/system-based analysis. As noted, Symantec reports vulnerabilities that attack the systems and software configurations used by their customers. For the vulnerabilities we studied, more than 84% and 36% of the exploited vulnerabilities reported by Symantec exist in products solely from, or run on, Microsoft and Adobe

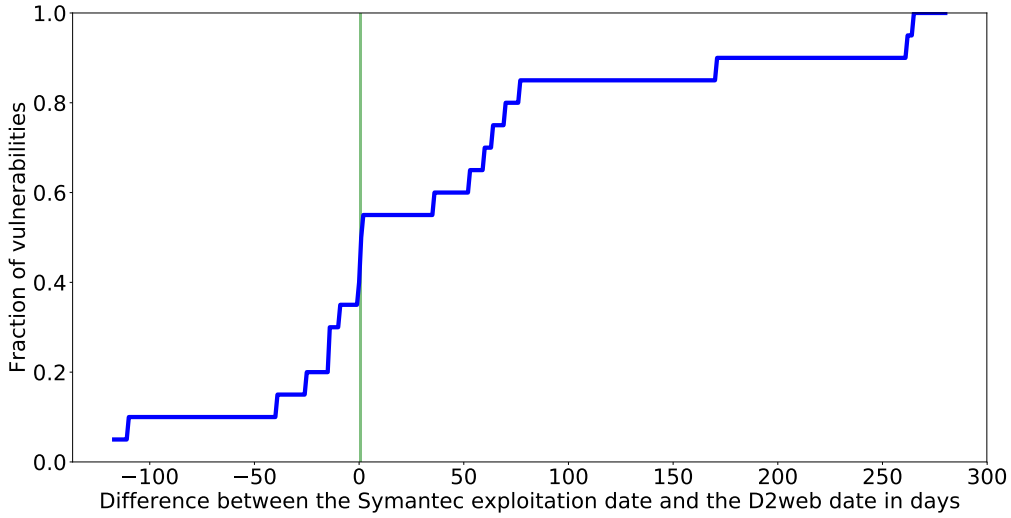


Figure 2.5: Day Difference Between CVE First Mentioned in D2web and Symantec Attack Signature Date vs. the Fraction of Exploited CVEs on D2web Reported (Cumulative).

products, respectively; whereas less than 16% and 8% of vulnerabilities published in the NVD are related to Microsoft and Adobe, respectively. Figure 2.6 shows the percentage from the exploited vulnerabilities that can affect each of the top five vendors in every data source. It is important to note that a vulnerability may affect more than one vendor (e.g., CVE-2016-4272 exists in Adobe Flash Player,[31] and it allows attackers to execute arbitrary code via unspecified vectors and can affect products from all five vendors.) This explains why some operating systems (e.g., Linux) with less coverage from Symantec data are targeted by vulnerabilities reported by Symantec.

In addition, D2web data appears to have more uniform vendor coverage. Only 30% and 6.2% of the vulnerabilities mentioned in the D2web during the period we study are related to Microsoft and Adobe, respectively. Additionally, ZDI favors products from these two vendors (57.8% for Microsoft and 35.2% for Adobe). This

[31] `https://www.adobe.com/products/flashplayer.html`

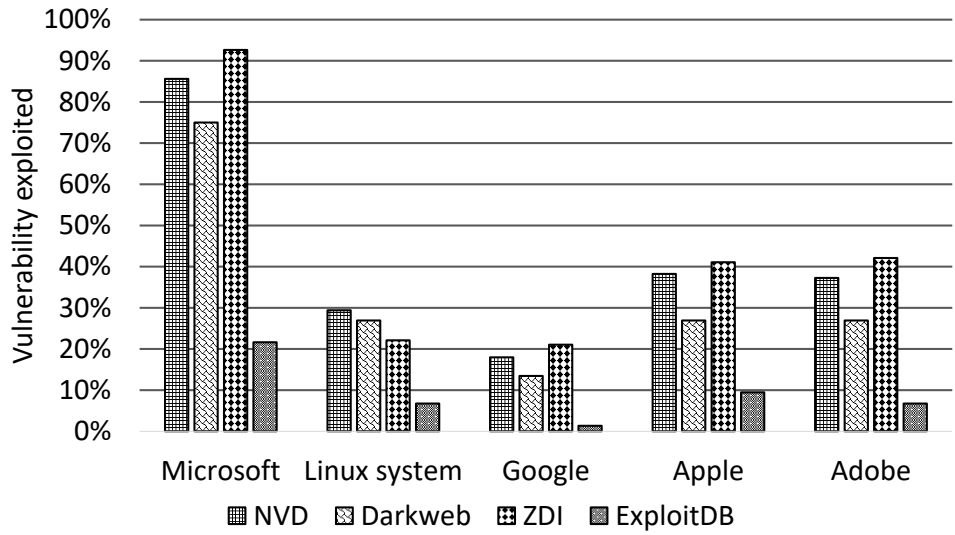


Figure 2.6: Most Exploited Vendors/Systems in Each Data Source.

provides evidence that each data source covers vulnerabilities targeting varying sets of software vendors.

Language-based analysis. Interestingly, we found notable variations on the exploitation likelihood depending on the language used on D2web data feeds that reference CVEs. In D2web feeds, four languages are detected with different vulnerability post and item distributions. Not surprisingly, English and Chinese have far more vulnerability mentions (242 and 112, respectively) than Russian and Swedish (13 and 11, respectively). However, vulnerabilities mentioned in Chinese postings are characterized by the lowest exploitation rate. For example, of those vulnerabilities, only 12 are exploited (about 10%), while 32 of the vulnerability mentioned on English postings are exploited (about 13%). Although vulnerability mentions in Russian or Swedish postings are few, these vulnerabilities have very high exploitation rates. For example, about 46% of the vulnerabilities mentioned in Russian were exploited (6), and about 19% for vulnerabilities mentioned in Swedish (2). Figure 2.7 shows the number of vulnerability mentions by each language as well as the number of *exploited* vulnerabilities mentioned by each language.

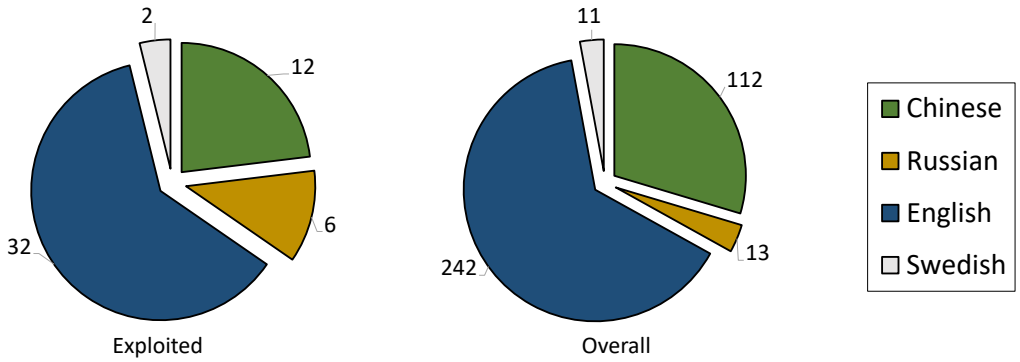


Figure 2.7: Number of *Exploited* Vulnerabilities Mentioned by Each Language (Left), and the Number of Vulnerabilities Mentions in Each Language (Right).

2.6 Experimental Setup

We perform a series of experiments with our models to evaluate our approach. First, we compare our model to a benchmark work presented in [1]. For our model, we found that Random Forest (RF) gives us the best F1 measure.[32] Random forest is an ensemble method proposed by Breiman [58]. It is based on the idea of generating multiple predictors (decision trees in this case), which are then used in combination to classify a newly disclosed vulnerability. The strength of the random forest lies in introducing randomness to build each classifier and using random, low-dimensional subspaces to classify the data at each node in a classifier. We use a random forest that combines bagging [58] for each tree with random feature selection [59] at each node to split the data. The final result is, therefore, an ensemble of decision trees, each having their own independent opinion on class labels (i.e., *exploited* or *not exploited* for a given disclosed vulnerability). Therefore, a new vulnerability is classified independently by each tree and assigned a class label best fit for it. Multiple decision trees

[32]The harmonic mean of precision and recall.

may result in having multiple class labels for the same data sample; hence, a majority vote is taken and the class label with most votes is assigned to the vulnerability.

In [1], the authors temporally mix their samples i.e., vulnerabilities exploited in the future are used to predict vulnerabilities exploited in the past, a practice that is discussed in [30]. Additionally, to account for the severe class imbalance, only vulnerabilities that occur in Microsoft or Adobe products were used in training and testing (477 vulnerabilities, 41 of which were exploited). We compare our model to [1] under the same conditions.

For the second experiment, we restrict the training samples to the vulnerabilities published before any of the vulnerabilities in the testing samples were published. Also, we only use data feeds that are present before the *exploitation date*. Thus, we guarantee that our experimental settings resemble the real-world case. Since we cannot make any assumptions regarding the sequence of events for the exploited vulnerabilities reported by Symantec without the *exploitation date* ($n = 112$), we remove these vulnerabilities from our experiments. The fraction of exploited vulnerabilities becomes 1.2%. We compare the performance of our model to the CVSS score.

The goal for exploit prediction is to predict whether a disclosed vulnerability will likely be exploited in the future or not. Few vulnerabilities are exploited before they are published [36]. Prediction for such vulnerabilities does not add any value to the goal of the prediction task, considering that they had been already exploited by the time those vulnerabilities are revealed. That being said, knowing what vulnerabilities are exploited in the wild can help organizations with their cyber defense strategies, but this is out of the scope of this chapter.

2.6.1 Performance Evaluation

We evaluate our classifiers based on two classes of metrics that have been used in previous work. The first class is used to demonstrate the performance achieved in the minority class (in our case 1.2%). The metrics under this class are *precision* and *recall*. They are computed as reported in Table 2.4. *Precision* is defined as the fraction of vulnerabilities that were exploited from all vulnerabilities predicted to be exploited by our model. It highlights the effect of mistakingly flagging non-exploited vulnerabilities. *Recall* is defined as the fraction of correctly predicted exploited vulnerabilities from the total number of exploited vulnerabilities. It highlights the effect of unflagging important vulnerabilities that were used later in attacks. For highly imbalanced data, these metrics give us an intuition regarding how well the classifier performed on the minority class (i.e., exploited vulnerabilities). The *F1* measure is the harmonic mean of precision and recall. It summarizes precision and recall in a common metric. The F1 measure can be varied based on the trade-off between precision and recall. This trade-off is dependent on the priority of the applications. If keeping the number of incorrectly flagged vulnerabilities to a minimum is a priority, then high precision is desired. To keep the number of undetected vulnerabilities that are later exploited to a minimum, high recall is desired. We further report the *Receiver Operating Characteristics* (ROC) curve as well as the *Area Under Curve* (AUC) of the classifier. ROC graphically illustrates the performance of our classifier by plotting the true positive rate against the false positive rate at various thresholds of the confidence scores the classifier outputs. In binary classification problems, the overall TPR is always equivalent to recall for the positive class, while FPR is the number of *not exploited* vulnerabilities that are incorrectly classified as *exploited* from all *not exploited* samples. ROC is a curve, thus, AUC is the area under ROC.

The higher the AUC value, the closer the model to perfection (i.e., a classifier with an AUC of 1 is a perfect classifier).

Table 2.4: Evaluation Metrics. TP: True Positives, FP: False Positives, FN: False Negatives, and TN: True Negative.

Metric	**Formula**
Precision	$\frac{TP}{TP+FP}$
TPR (recall in case of binary classification)	$\frac{TP}{TP+FN}$
F1	$2 \cdot \frac{precision \cdot recall}{precision+recall}$
FPR	$\frac{FP}{FP+TN}$

2.6.2 Results

We utilize and compare the performance of several standard supervised machine learning approaches for exploit prediction. Parameters for all approaches were set in a way to provide the best performance. We use the *scikit-learn* Python package [60].

Examining Classifiers. We maintain the temporal information for all the classifiers. The vulnerabilities are sorted according to the date they were posted on the NVD. The first 70% are reserved for training, along with the features that are available by the end of the training period. The remaining vulnerabilities are used for testing.

Table 2.5 shows a comparison between the classifiers with respect to precision, recall, and F1 measure. Random forest performs the best with F1 measure of 0.4 as compared to Support Vector Machine (SVM): 0.34, Bayesian Network (BN): 0.34, Logistic Regression (LOG-REG): 0.33, Decision Tree (DT): 0.25, and Naïve Bayes (NB): 0.27. Note that even though RF has the best F1 measure, it does not have the best recall—NB does. We choose RF with high precision, which makes the model reliable as compared to low precision, which results in a lot of false positives.

Table 2.5: Precision, Recall, F1 Measure for RF, SVM, LOG-REG, DT and NB to Predict Whether a Vulnerability Will Likely Be Exploited.

Classifier	**Precision**	**Recall**	**F1 measure**
RF	**0.45**	0.35	**0.40**
BN	0.31	0.38	0.34
SVM	0.28	0.42	0.34
LOG-REG	0.28	0.4	0.33
DT	0.25	0.24	0.25
NB	0.17	**0.76**	0.27

Benchmark test. We compare our model to a recent work that uses vulnerability mentions on Twitter to predict the likelihood of exploitation [1]. In the study, the authors use SVM as their classifier, while our model works best with a Random Forest classifier. Although it would be straightforward to think that our approach would achieve better performance than the work proposed in [1], we only compare to this work because: (1) to the best of our knowledge, there is no existing work on predicting exploits in the wild using D2web data, and (2) we compare all major approaches, and currently using feeds from social media is the best one.

In [1], the authors restricted the training and evaluation of their classifier to vulnerabilities targeting Microsoft and Adobe products, because Symantec does not have attack signatures for all the targeted platforms. They performed a 10-fold stratified cross-validation, where the data is partitioned into 10 parts while maintaining the class ratio in each part. They trained on 9 parts and tested on the remaining one. The experiment was repeated for all 10 parts. Hence, each sample gets tested once.

For comparison, we also perform the same experiment, under highly similar assumptions. We use all exploited vulnerabilities regardless of whether the date is reported by Symantec or not. In our case, we have 2,056 vulnerabilities targeting

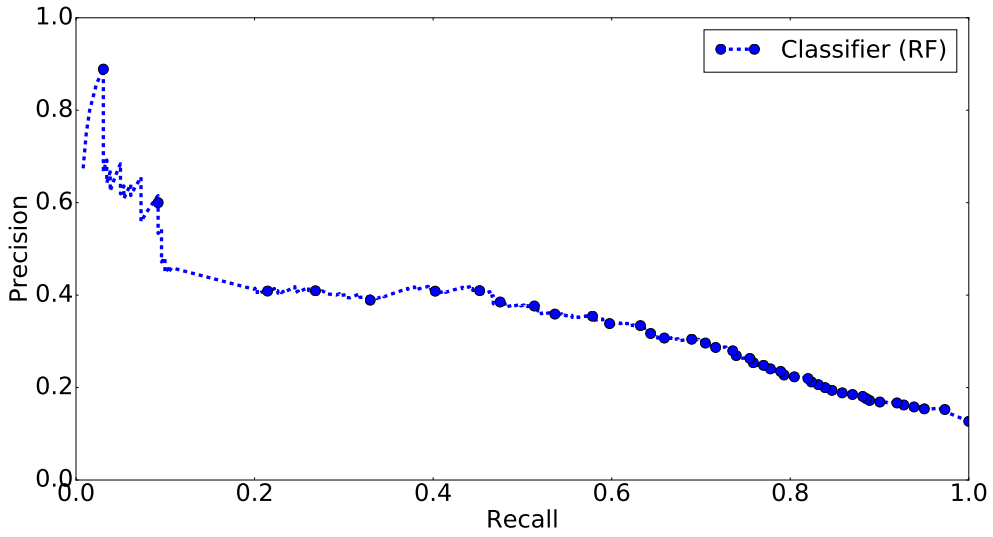


Figure 2.8: Precision-Recall Curve for Proposed Features of Microsoft-Adobe Vulnerabilities (RF).

Microsoft and Adobe products. Out of 2,056 vulnerabilities, 261 are exploited, a fraction that is consistent with previous works. We perform a 10-fold stratified cross-validation. We plot the precision-recall curve for our model (see Figure 2.8). The precision-recall curve shows us the trade-off between precision and recall for different decision thresholds. Since the F1 measure is not reported in [1], we use the precision-recall curve reported for comparison. By maintaining the recall value constant, we compare how precision varies. Table 2.6 shows the comparison between the two models by comparing precision for different values of recall. For a threshold of 0.5 we get an F1 measure of 0.44 with precision 0.53 and recall 0.3. Maintaining recall, the precision value displayed in the graph in [1] is 0.3, significantly lower than 0.4. We perform the same experiment on different recall values to compare precision. At each point, we obtain higher precision than the previous approach.

Baseline comparison. In [30], the authors argue that the problem of predicting the likelihood of exploitation is sensitive to the sequence of vulnerability-related events. Temporally mixing such events leads to future events being used to predict past ones,

Table 2.6: Precision Comparison between [1] and the Proposed Model While Maintaining Recall.

Recall	Precision*	Precision (our work)
0.20	0.30	**0.41**
0.40	0.18	**0.40**
0.70	0.10	**0.29**

*Numbers derived from Figure 6.a. from [1].

resulting in inaccurate prediction results. To avoid the temporal mixing of events, we create time-based splits, as described in this section.

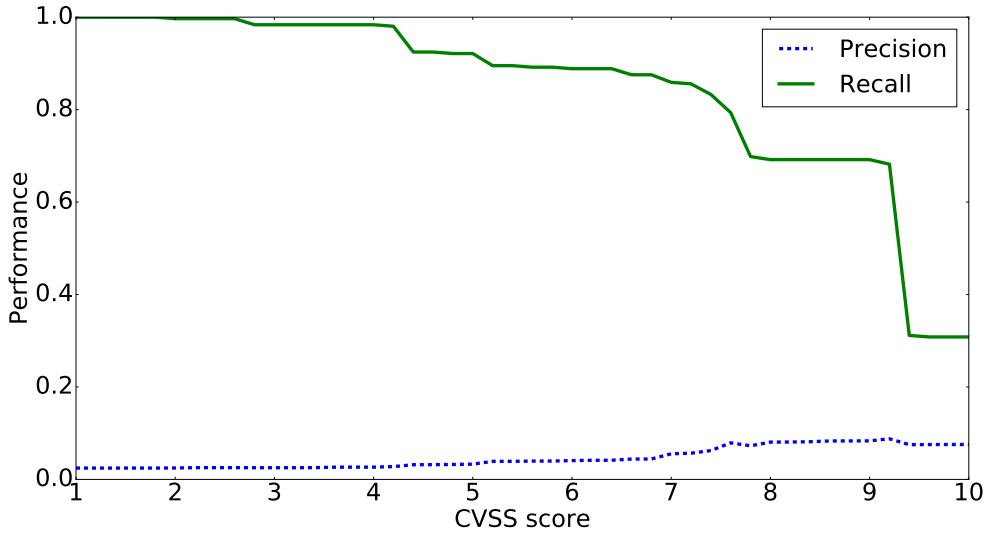


Figure 2.9: Precision and Recall for Classification Based on CVSS Base Score Version 2.0 Threshold.

For a baseline comparison, we use the CVSS version 2.0 base score to classify whether a vulnerability will be exploited or not based on the severity score assigned to it. CVSS score has been used as a baseline in previous studies [35, 1]. CVSS tends to be overly cautious, i.e., it tends to assign high scores to many vulnerabilities, resulting in many false positives. Figure 2.9 shows the precision-recall curve for the CVSS score. It is computed by varying the decision threshold (x-axis), on which

we determine the class label of each vulnerability. CVSS gives high recall with very low precision, which is not desired for real-world patch prioritization tasks. The best F1 measure that could be obtained is 0.15. Figure 2.10 shows the performance comparison between our proposed RF model and the CVSS-based model that yields the highest F1 score. Our model outperforms the baseline with an F1 measure of 0.4, a precision of 0.45, and a recall of 0.35. Additionally, our classifier shows very high TPR (90%) at low FPR (13%), with an AUC of 94%, as depicted in Figure 2.11.

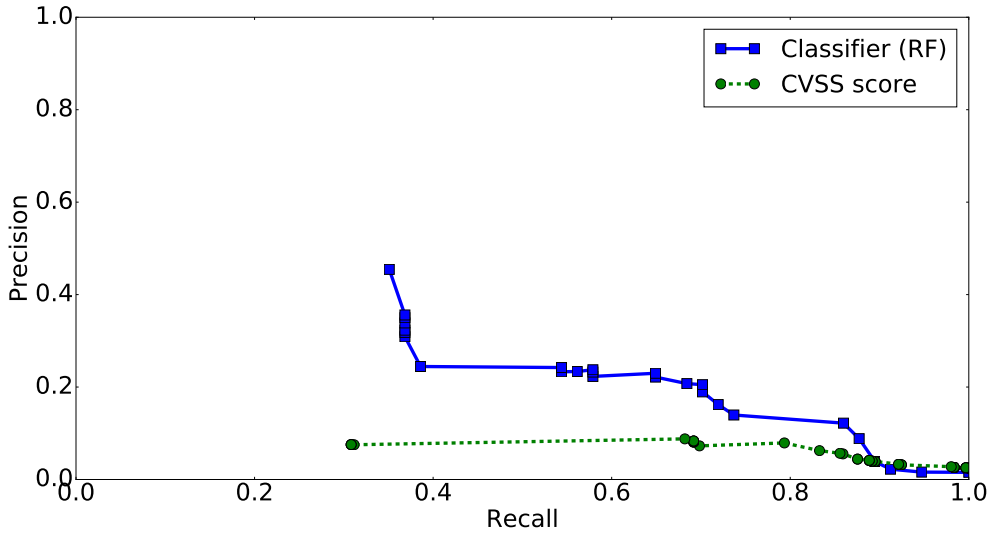


Figure 2.10: Precision-Recall Curve for Classification Based on CVSS Score Threshold (RF).

Evaluation with individual data sources. We study what effect introducing each data source has on the prediction of vulnerabilities mentioned in that source. This is important for understanding if the addition of a particular data source benefits the vulnerabilities that have been mentioned in that data source. We find that the time-based split used in the previous experiments leaves very few vulnerabilities mentioned in these data sources in the test set (ZDI: 18, D2web: 4, EDB: 2). Hence, we increase the numbers by (1) performing a 10-fold cross-validation without sorting the vulnerabilities (2) increasing the ground truth by considering the exploited vulnerabilities

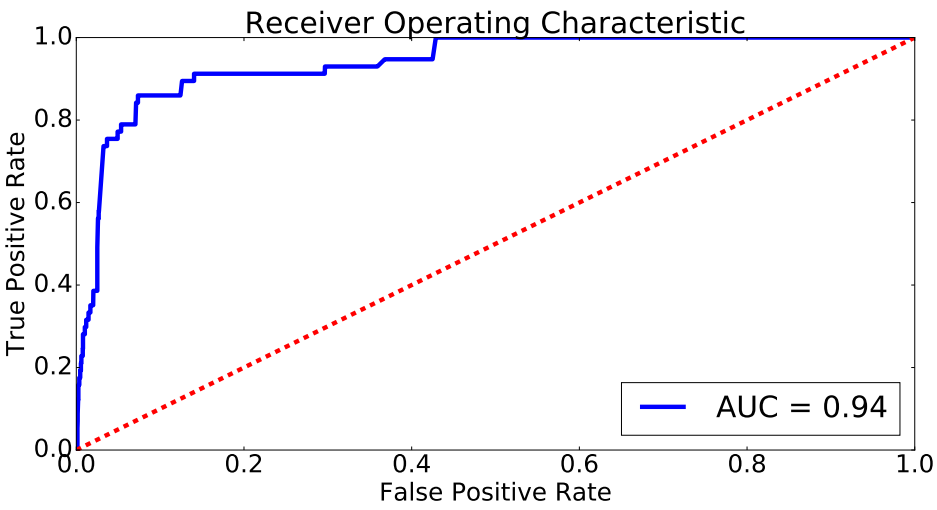


Figure 2.11: ROC Curve for Classification Based on Random Forest Classifier.

that did not have exploit date (these were removed from earlier experiments because it was not clear whether these were exploited before or after the vulnerability was exploited). Using these two techniques, we have 84 vulnerabilities mentioned in ZDI that have been exploited, 57 in EDB, and 32 in the D2web. We report the results (precision, recall, and F1) for the vulnerabilities mentioned in each data source. Also, we mention the prediction of these vulnerabilities by using only the NVD features. For the vulnerabilities mentioned in the D2web, we only consider the D2web features along with the NVD features. The model predicts 12 vulnerabilities as exploited with a precision of 0.67 and a recall of 0.38. By only considering the NVD features, the model predicts 12 vulnerabilities as exploited with a precision of 0.23 and a recall of 0.38. Consequently, using the D2web features, the precision improved significantly from 0.23 to 0.67. Table 2.7 shows the precision-recall with corresponding the F1 measure. D2web information was thus able to correctly identify positive samples mentioned in the D2web with higher precision.

Table 2.7: Precision, Recall, F1 Measure for Vulnerabilities Mentioned on D2web, ZDI, and EDB.

Source	Case	Precision	Recall	F1 measure
D2web	NVD	0.23	0.38	0.27
	NVD + D2web	0.67	0.375	0.48
ZDI	NVD	0.16	0.54	0.25
	NVD + ZDI	0.49	0.24	0.32
EDB	NVD	0.15	0.56	0.24
	NVD + EDB	0.31	0.40	0.35

For ZDI, we have 84 vulnerabilities mentioned. By just utilizing the NVD features, we get an F1 measure of 0.25 (precision: 0.16, recall: 0.54) as compared to adding the ZDI feature with F1 measure of 0.32 (precision: 0.49, recall: 0.24)—a significant improvement in precision. Table 2.7 also shows the precision-recall with corresponding F1 measure for samples mentioned on ZDI.

We perform a similar analysis for the vulnerabilities that have PoCs available on EDB. For EDB, there are 57 vulnerabilities with PoCs. With only the NVD features, the model scored an F1 measure of 0.24 (precision: 0.15, recall: 0.56); while adding the EDB feature boosted the F1 measure to 0.35 (precision: 0.31, recall: 0.40)—a significant improvement in precision, as shown in Table 2.7.

Feature importance. To better explain our choices to the features, we examine and provide an understanding on where our prediction power primarily derives from. We report the features that have the most contribution to the prediction performance. A feature vector for a sample has 28 features computed from the non-textual data (summarized in Table 2.2) as well as the textual features—TF-IDF computed from the bag of words for the 1,000 words that have the highest frequency in the NVD description and the D2web. For each of the features, we compute the Mutual Information (MI) [61], which expresses how much a variable (here a feature x_i) tells about

another variable (here the class label $y \in \{exploited,\ not\ exploited\}$). The features that contribute the most from the non-textual data are $\{language\_Russian = true,\ has\_D2web = true,\ has\_PoC = false\}$. In addition, the features that contribute the most from the textual data are the words $\{demonstrate,\ launch,\ network,\ xen,\ zdi,\ binary,\ attempt\}$. All of these features received MI scores over 0.02.

Addressing class imbalance The problem of class imbalance has gained a lot of research interest (see [62] for a survey). Since our dataset is highly imbalanced, we use SMOTE [56] and measure the improvement in classification performance. SMOTE oversamples the minority class by creating synthetic samples with features similar to those of the exploited vulnerabilities. This data manipulation is only applied to the training set. Using SMOTE, no performance improvement is achieved for our RF classifier. However, SMOTE introduces a considerable improvement with a Bayesian Network (BN) classifier. Table 2.8 reports different oversampling ratios and the change in performance. The best oversampling ratio is experimentally determined, i.e., high oversampling ratios lead the model to learn from a distribution that differs significantly from the real distribution.

Table 2.8: Performance Improvement Attained by Applying SMOTE for the BN Classifier Using Different Oversampling for the Exploited Samples.

Oversampling percentage	**Precision**	**Recall**	**F1 measure**
100%	0.37	0.42	0.39
200%	0.40	**0.44**	**0.42**
300%	**0.41**	0.40	0.40
400%	0.31	0.40	0.35

2.7 Adversarial Data Manipulation

We study the effects of adversarial data manipulation only on D2web data. For the presence of PoCs, we only consider PoCs that are verified by EDB. Adversaries need to hack into EDB to add noise or remove PoCs from the EDB database. In this analysis, we assume such action cannot be taken by adversaries. Similarly, ZDI publishes only vulnerabilities that are verified by its researchers; hence there is a very small chance of manipulating these data sources.

On the other hand, the public nature of D2web marketplaces and forums gives an adversary the ability to poison the data used by the classifier. They can achieve this by adding vulnerability discussions on these platforms with the intent of fooling the classifier to make it produce high false positives. Previous works discuss how an adversary can influence a classifier by manipulating the training data [63, 64, 65].

In our prediction model, we use the presence of the vulnerability in the D2web, the language of the market/forum on which it was mentioned, and the vulnerability description as features. An adversary could easily post discussions regarding vulnerabilities he does not intend to exploit, nor does he expect that they will be exploited. To study the influence of such noise on the performance of the model, we experiment with two strategies:

1. Adversary adding random vulnerability discussions. In this strategy, the adversary initiates random vulnerability discussions on the D2web and reposts them with different CVE numbers. So the CVE mentions on the D2web increases. For our experiment, we consider two cases with different amounts of noise added. In case 1, we assume that the noise is present in both the training and the testing data. We consider varying fractions of noise (5%, 10%, 20% of the total data samples) randomly distributed in training and testing data. The experimental setup follows conditions

discussed in Section 2.6. Vulnerabilities are first sorted according to time, and the first 70% are reserved for training and the remaining for testing. Figure 2.12 shows the ROC curve showing the false positive rate (FPR) vs the true positive rate (TPR). For different amounts of noise introduced, our model still maintains a high TPR with low FPR and AUC of at least 0.94, a performance similar to the experiment without adding noise. This shows that the model is highly robust against noise such that it learns a good representation of the noise in the training dataset and can distinguish them in the testing dataset.

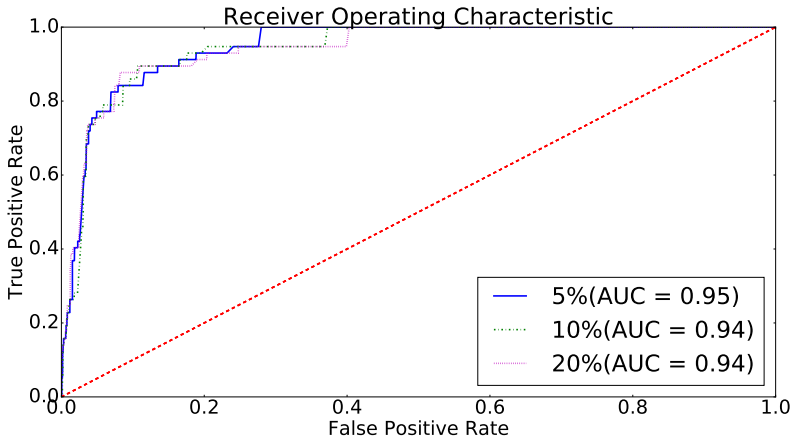


Figure 2.12: ROC Curves for Adversary Adding Noise to Both the Training and the Testing Data.

For case 2, we randomly add vulnerability discussion found on the D2web with different CVE numbers to only the test data and repeat the same experiment. Figure 2.13 shows the ROC plot. In the case, even though there is a slight increase in the FPR, the performance is still on par with the experiment without noise (AUC of 0.87 or more). Hence, noisy samples affect the prediction model slightly, if no noise was introduced in the training data.

2. Adversary adding vulnerability discussion similar to the NVD description. In the previous strategy, the adversary adds vulnerability discussions randomly

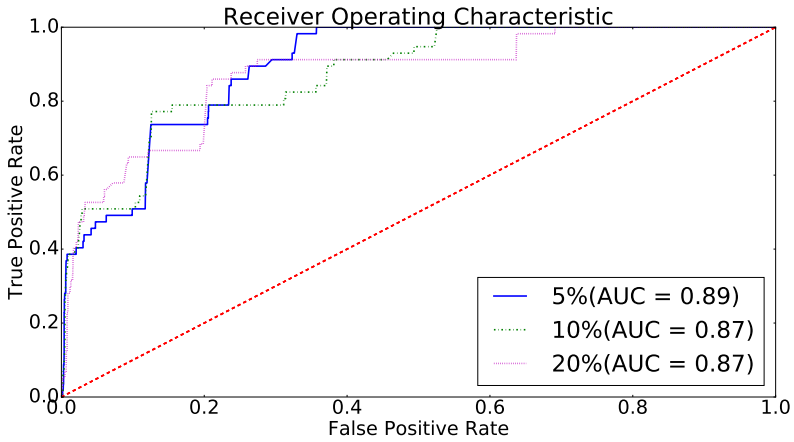


Figure 2.13: ROC Curves for Adversary Adding Noise to Only the Testing Data.

without taking into account the actual capability of the vulnerability. For instance, CVE-2016-3350[33] is a vulnerability in Microsoft Edge, as reported by the NVD. If the vulnerability is mentioned on the D2web as noise by an adversary but targeting Google Chrome, then it might be easy for the prediction model to detect it as seen in previous experiments. But what if the adversary crafts the vulnerability discussion such that it is a copy of the NVD description or consistent with the NVD description? In this strategy, the adversary posts the NVD description with the CVE number on the D2web. For case 1, we consider this noise to be randomly distributed in both training and testing. Figure 2.14 shows the ROC curves for different levels of noise. The performance decreases as the number of noisy samples increases, but there is no significant decline (AUC of 0.88 or more).

We repeat the experiment by adding noise only in the test data for case 2. In this experiment, we observe that the biggest drop in performance results in an AUC of 0.78 when 20% of the samples are noise (see Figure 2.15). This shows that adding correct vulnerability discussions does affect the prediction model, except if a large

[33] `https://nvd.nist.gov/vuln/detail/CVE-2015-3350`

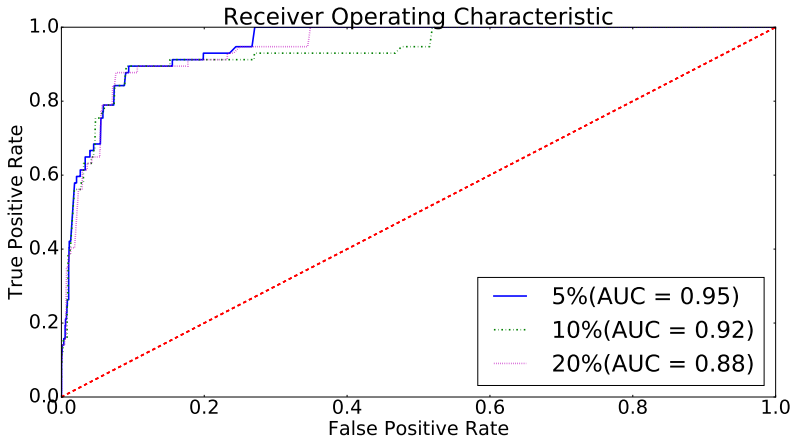


Figure 2.14: ROC Curves for Adversary Adding Noise to Both the Training and the Testing Data.

number of such samples are added. Also, the effect can be countered by also adding such noisy samples to the training data for the model to learn from.

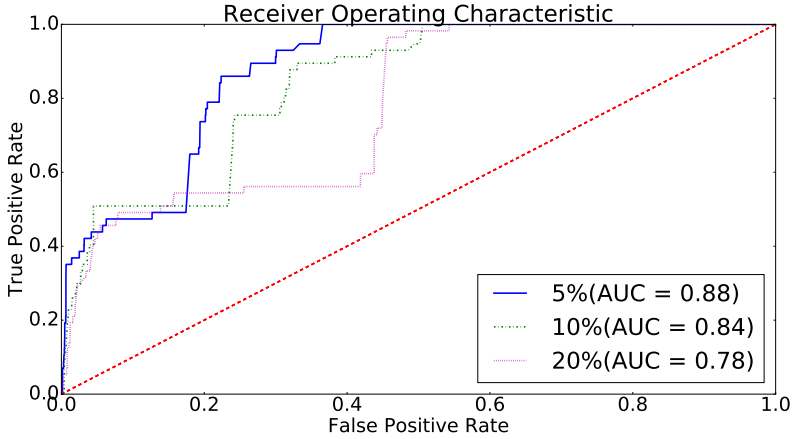


Figure 2.15: ROC Curves for Adversary Only Adding Noise to the Testing Data.

Note that an adversary would need to add a large number of noisy samples to lower the performance of the prediction model. Previous research on using data feeds like Twitter for exploit prediction mentions that an adversary can register a large number of Twitter accounts and flood Twitter with vulnerability mentions [1]. In the D2web markets and forums, the creation of accounts needs verification and, in some cases, technical demonstration of skills. While fake accounts are often themselves sold

on the D2web, it is difficult to purchase and maintain thousands of such fake accounts to post with them. Also, if one person is posting a large volume of discussions with CVE mentions, he/she can be identified from their *username* or can be removed from the market/forum if many of their posts get downvoted for being irrelevant. It is also important to note that such forums also function as a meritocracy [48], where users who contribute more are held in higher regard (which also makes it difficult to flood discussions with such information).

2.8 Discussion

Viability of the model and cost of misclassification. The performance achieved by our model as a first-line defense layer is very promising. Recall that a random forest classifier outputs a confidence score for every testing sample. A threshold can be set to identify the decision boundary. We shall note that all the results we report in this chapter are achieved based on hard-cut thresholds, such that all samples that are assigned confidence scores greater than a threshold thr are predicted as to be exploited. Relying solely on a hard-cut threshold may not be a good practice in real-world threat assessments; rather, thr should be varied in accordance to other variables within an organization such that different thresholds can be set to different vendors (i.e., thr_{ven1}, thr_{ven2}), or information systems (i.e., thr_{sys1}, thr_{sys2}). For instance, if an organization hosts an important website on an Apache server, and the availability of this site is of top priority for the organization, then any vulnerability of the Apache server should receive high attention and put forward to remediation plans regardless of other measures. Other vulnerabilities, tens of which are disclosed on a daily bases, may exist in many other systems within the organization.

Since it is very expensive to be responsive to that many security advisories (e.g., some patches may be unavailable, some systems may need to be taken offline to apply

patches), assessing the likelihood of exploitation can help in quantifying risk and planning mitigation. Risk is always thought of as a function of likelihood (exploitation) and impact. The cost of classifying negative samples as *exploited* is the effort made to have them fixed. This involves patching or other remediation such as controlling access or blocking network ports. Similarly, the cost of falsely classifying threatened vulnerability as *not to be exploited* depends on the impact it will cause if exploited. For example, if two companies run the same database management system, and one hosts a database with data about all business transactions, while the other hosts a database with data that is of little value to the company, the resulting cost of a data breach would be significantly different.

Model success and failure cases. By analyzing the false negatives and false positives, we gain an understanding of why and where our model performs well and where it is ineffective. We first look into false negatives. The 10 exploited vulnerabilities (about 18% of the exploited samples in the testing dataset) that received the lowest confidence scores seem to have common features. For example, 9 of these appear in Adobe products, namely Flash Player (5 vulnerabilities) and Acrobat Reader (4 vulnerabilities). Flash Player vulnerabilities seem to have very similar description from the NVD. The same is observed for Acrobat Reader. We also observe that they were assigned CVE numbers on the same day (April 27, 2016), and 7 out of these 9 were published on the same day as well (July 12, 2016), and assigned a CVSS base score of 10.0 (except for one, assigned 7.0). The other vulnerability exists in the Windows Azure Active Directory (CVSS score of 4.3). Out of these 10 vulnerabilities, one had a verified PoC archived on EDB before it was detected in the wild, and another one had a ZDI mention, while none were mentioned in the D2web. We attribute misclassifying these vulnerabilities to the limited representation of these samples in

the training dataset. Moreover, this observation signifies the importance of avoiding experiments on time-intermixed data, a point discussed in Section 2.3.2.

We also study false positive samples that receive high confidence score—samples our model predicted as exploited while they are not. For our random forest classifier, all false positives we examine exist in products affecting Microsoft products although we do not use vendor as a feature. Our model is able to infer the vendor from other textual features. We assume that this level of overfitting is unavoidable and marginal, and we attribute this largely to the limitations on our ground truth. The model is highly generalizable though. We find examples of vulnerabilities from other vendors with confidence scores close to *thr* we set; however, we cannot assume that they are exploited.

2.9 Conclusion

In this chapter, we proposed an approach that aggregates early signs of vulnerability exploitation from various online sources to predict *if* vulnerabilities will be exploited, a problem that is directly related to patch prioritization. Our machine learning models not only outperform existing severity scoring standards but also outperform approaches that combine information from social media sites like Twitter for exploit prediction. Existing scoring standards suffer high false positive rates: too many vulnerabilities are overrated, making it impossible for the patching teams to prioritize what to patch first. Our approach; however, achieves a high true positive rate while maintaining false positive rate low. In the future, we look to use other

intelligence sources, including penetration testing platforms such as Metasploit[34] and hacker community in social media platform such as Twitter and Chan sites.[35]

[34] `https://www.metasploit.com`

[35] A type of Internet forums, mostly image boards, that encourage visitors to anonymously post content. Some Chan sites are found to be leveraged by activists, such as the well-known hacking activist group Anonymous.

Chapter 3

COMBINING SOCIAL NETWORK ANALYSIS WITH ML TECHNIQUES TO PREDICT EXPLOITS IN THE WILD

3.1 Introduction

Existing studies suggest that D2web sites are among the best sources for gathering cyberthreat intelligence supporting cyber defense operations [46, 35, 48, 31]. Until recently, the value of such intelligence in predicting exploits in the wild was not well studied. In Chapter 2, we show that information about *what* activity emerges following public disclosure is a key to predict exploits in the wild. We also illustrate the importance of the content of malicious hacker discussions on the D2web, i.e., when added to other threat intelligence sources, it significantly improved the performance of ML models. Yet, the utility of information about *who* discusses vulnerabilities is to be discovered in this chapter. Here, we continue to address the same task, but with a different approach. This approach focuses on harnessing ML techniques on features derived from the social network of hackers participating in D2web forums, as well as features derived from the NVD. We demonstrate the viability of such data in predicting exploits in the wild through a series of experiments. We believe this is because social network structures related to certain exploit authors is indicative of their ability to write exploits that are subsequently employed in an attack. Specific contribution of this chapter includes:

- Introducing an approach that uses the users' post/reply discussions in D2web to generate a directed social graph. We also provide observations on the connectivity measures for the individuals who authored postings preceding exploits

in the wild. We specifically show that the users who discussed vulnerabilities that are subsequently exploited in the wild are about 4 times more active in posting than other users.

- Empirically demonstrating the viability of the approach through a suite of experiments on real-world hackers and exploits data. We specifically demonstrate that the addition of social network data improves the value of recall by about 19%, F1 score by about 6% while maintaining precision.

The remainder of this chapter is organized as follows. In Section 3.2, we present our data and modeling approach. Section 3.3 describes the hacker social network generation approach. Section 3.4 presents analysis and observations from the hacker social network. Empirical prediction results are presented in Section 3.5. The implications of the results for vulnerability management tasks are discussed in Section 3.6. Section 3.7 provides an overview about the works related to ours, and Section 3.8 concludes this chapter.

3.2 Approach

Similar to Chapter 2, we view the problem of predicting exploits in the wild as a binary classification problem, where the positive class is *exploited*, and the negative class is *not exploited*. Figure 3.1 gives an overview of our proposed approach.

3.2.1 Data Collection

Our machine learning models use features derived from two sources: vulnerability data feeds collected from the NVD and a D2web database of posts with cybersecurity-related content collected and filtered from 151 D2web forums [18]. The class labels are determined based on a ground truth set of attack signatures of exploits detected

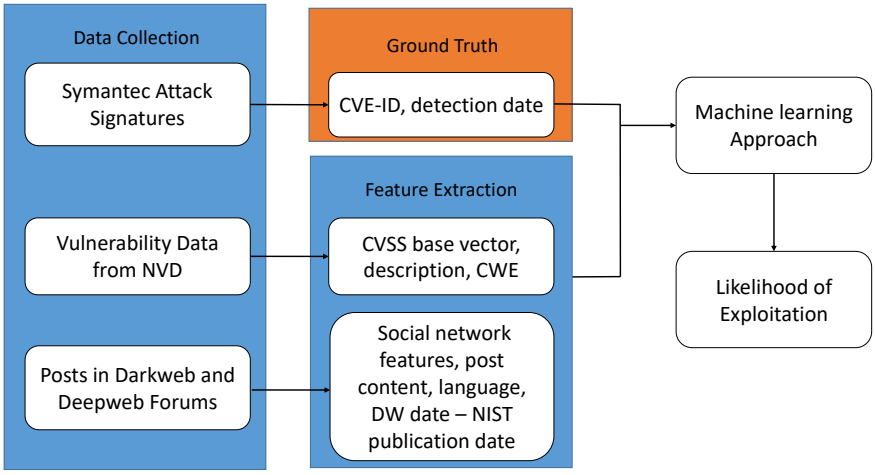


Figure 3.1: An Overview of the Predictive Model.

in the wild and reported by Symantec. For this chapter, we focus our study on data collected from all the mentioned sources between January 2010–March 2017. Figure 3.2 shows the number of vulnerabilities published on NVD per year, the number of vulnerabilities mentioned on D2web, and the number of vulnerabilities reported by Symantec. Next, we provide details on the data collection process for the three sources we use.

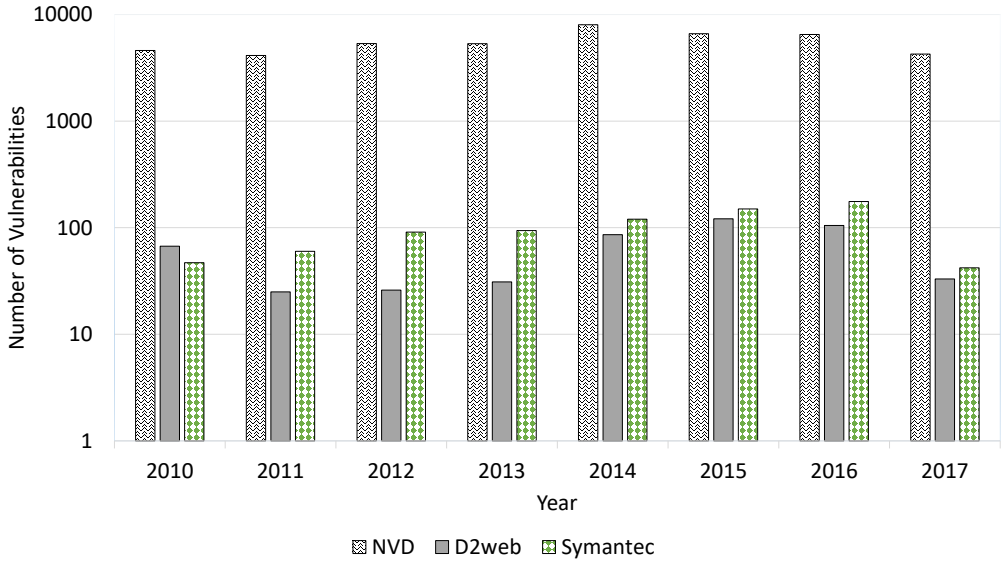


Figure 3.2: Vulnerabilities Reported per Year From NVD (per Disclosure Year), D2web (per Disclosure Year), and Symantec (per Exploitation Year).

NVD. Every vulnerability in the NVD is assigned a unique CVE number. To collect vulnerability data, we use the JSON data feeds provided by the NVD and extract information about the vulnerabilities we study (extracted information is discussed in Section 3.2.2). Further, a web scraper is developed to retrieve from the NVD's vulnerability webpages the data elements that are missing in the JSON files (e.g., the disclosure date).

D2web forums. We use a database of posts collected from 151 darkweb and deep-web forums. The data collection system is described in Chapter 2, and originally introduced in [18]. In summary, the system extracts the hacking-related content from D2web sites using a learning algorithm with high accuracy. The database we use contains over 2,290,000 posts under 223,074 distinct forum topics and from 151 distinct forums. From these posts, only 3,082 have explicitly mentioned 624 distinct vulnerabilities by referencing their CVE numbers. Of those, 502 are within the period we focus on, and they are found in 46 different forums. We only use these forums to build the social network of users, as described in Section 3.3.

Symantec attack signatures. We label vulnerabilities as *exploited* in the wild if CVE numbers are identified in the description of the attack signatures reported by Symantec's anti-virus[1] or Intrusion Detection Systems' attack signatures.[2] The fraction of exploited vulnerabilities is found to be very small as shown in Figure 3.2, and it varies from one year to another. The maximum value for the fraction of exploited vulnerabilities is 2.7% (for vulnerabilities published in 2016), and minimum value is less than 1% (for vulnerabilities published in 2017)—previous work has reported fractions comparable to our findings [1, 35].

[1]A complete list is found here `https://www.symantec.com/security_response/landing/azlisting.jsp`. The detection date is labeled with "Discovered".

[2]`https://www.symantec.com/security_response/attacksignatures`

3.2.2 Feature Description

Table 3.1 summarizes the features we extract from both data sources discussed in Section 3.2.1. Here we provide discussions on each of these sets of features.

Table 3.1: Summary of Features.

Source	Feature Set	Type
NVD	CVSS base score	Numeric and Categorical
	Description	TF-IDF word uni-grams
	CWE	Categorical
D2web	Social Network Features	Numeric
	Post content	TF-IDF uni-grams
Combined	D2web date - NVD date	Numeric

CVSS base score. CVSS is a vulnerability severity scoring framework designed to measure the exploitability and impact of software vulnerabilities. We use the version v2.0 CVSS base score. There are two components for this set of features: (1) the base score (numeric): a given severity score ranges from 0 to 10 (10 is the most severe), and (2) the CVSS vector: a vector of the metrics that determine the base score (categorical). The measures in the vector are *Access Vector, Access Complexity, Authentication, Confidentiality Impact, Integrity Impact*, and *Availability Impact*. Each one of these measures can take one value from a predetermined set of possible values. For example, *Access Vector* indicates how the vulnerability is exploited. It can take one of these three possible values: *Local (L), Adjacent Network (A) and Network (N)*.[3]

NVD description. NIST provides a textual description of the vulnerability when it is released. The description summarizes the system/software in which the flaw exists and gives information on how it can be exploited. Each vulnerability description undergoes a preprocessing pipeline including stemming (reducing the words to their root

[3]See `https://www.first.org/cvss/v2/guide` for a complete documentation.

forms) and stop word removal (e.g., *and, or, then*). The preprocessed documents are then turned into numerical feature vectors using Term Frequency-Inverse Document Frequency (TF-IDF) computed for the 250 most frequent words in the collection of documents. Informally, TF-IDF assigns higher values to the words that appeared in a document with higher frequency while rarely did these words appear in other documents in the corpus.

CWE. It is a community-effort project comprising enumerating common software security weaknesses (categorical). These are categories of flaws that can be unintentionally made during software development and can exist in software architecture, design, or code.

D2web social network features. Contains measures computed from the social connections of users posting hacking-related content. The basic social network features (e.g., in-degree and out-degree) indicate how active a user is in the social graph. More advanced features measure the centrality of users in the social graph. Highly central users are more important; thus the vulnerability mentions should take more consideration. We compute the features for the set of users who explicitly mentioned one or more CVE numbers in their posts.

Post content. We found evidence for many vulnerability mentions with content ranging from exploit offers to content irrelevant of the mentioned vulnerability. This set of features is extracted the same way as the NVD description features, except that for non-English posts, we automatically translate the content to English using Google Translate API,[4] then the TF-IDF is computed over the translated corpus.

[4] `https://cloud.google.com/translate`

3.2.3 Classifier Training and Prediction

We use a supervised machine learning approach to train classifiers on the presented features. The output of the classifiers is confidence score. A threshold can be set on confidence score to determine the best decision boundary. The experimental settings and results are described in Section 3.5.

3.3 Hacker Social Network

In Section 3.2.2, we describe the features we use, while here, we elaborate on the social network creation approach. We then provide our observations on the created graphs in Section 3.4. In this work, we adopt the same assumption made in much of the previous work on D2web data, where they consider the same usernames (case insensitive) across different D2web sites to belong to the same person(s) [49]. This allows for generating one network comprising a large number of D2web sites as opposed to a social network for each site [66].

3.3.1 Social Graph

Formally, the user social graph $G = (V, E)$ is a weighted, directed graph with no self-loops (i.e., every edge has a weight, every edge points away from one node to another node, and there exists at most one edge between any pair of nodes). V is the set of vertices (D2web users) and E is the set of edges.

The graph is created using posts between January 2010 and June 2016 (747,351 posts under 109,413 forum topics), and for every topic t_x, posts under t_x are grouped in a list l_x ordered by the date and time of posts. Then, an edge is created (with weight = 1) from user v_i to v_j and labeled with the date of v_i's posting date only if: (1) $v_i \neq v_j$, (2) both v_i and v_j have posts in l_x, and v_i has posted after v_j, (3) the

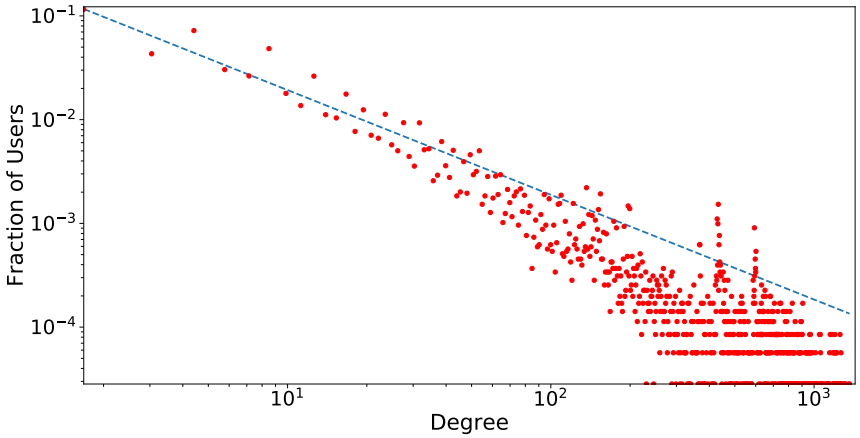


Figure 3.3: Degree Distribution—a Scale-Free Network With an Exponent γ of 1.07.

number of posts between v_i's post and v_j's post in l_x is less than thr (it is set to be 10 in all experiments in this chapter), and (4) there is no existing edge originating from v_i to v_j and labeled with the same date. Once the edges are created, they are added to a multi-directed graph with parallel edges of weights = 1. The multi-graph is then transformed to a directed graph G by summing the weights of the parallel edges pointing to the same direction.

Degree distributions, for both incoming and outgoing edges, of G are found to resemble the power-law distribution as depicted in Figure 3.3. This means that there exist very few users with a very large number of connections, and many users with few connections - this observation is known to be common for social media sites [67][5].

3.3.2 Social Network Measures

After creating the social network, we compute measures derived from the network structure. In this chapter, we consider three categories of social network measures:

[5]In the D2web database, there are few users, each has exactly one post. Some of these posts are found in topics with a number of postings greater than thr; hence, the little up-tick in Figure 3.3.

Network structure measures: the measures under this category are: (1) *In-degree*: the number of edges pointing to the user, (2) *Out-degree*: the number of edges originated from the user, (3) *Sum of In-degree weights*: the sum of the weights for all edges pointing to the user, (4) *Sum of out-degree weights*: the sum of the weights for all edges pointing away from the user. These measures describe the type of activities in which the user engages. For example, higher in-degree than out-degree may indicate the user tendency towards creating new topics or posting under topics shortly after they are created.

Centrality measures: three measures are computed: (1) *In-degree centrality*: it measures the popularity of a user v_i by normalizing v_i's in-degree by the maximum possible in-degree, (2) *Out-degree centrality*: measures how actively a user v_i replies to others by normalizing v_i's out-degree measure by the maximum possible out-degree, (3) *Betweenness centrality*: for a user v_i, Betweeness centrality measures the importance of v_i by computing the fraction of shortest paths between all pairs of users that pass through v_i.

Importance measures: the number of connections user v_i has with other users, by itself, may not be indicative of importance; rather, v_i is important if his/her posts make other important users reply. Hence, influence metrics incorporate the centrality of users with outgoing edges to v_i into v_i's centrality (i.e., if an important user v_j replies to v_i, then the importance of v_i increases). Two measures are computed under this category: (1) *Eigenvector centrality*: measures the importance of v_i by assigning a centrality proportional to the sum of in-neighbors' centralities. Eigenvector centrality of v_i is the i^{th} value of the eigenvector C_e corresponding to the largest eigenvalue of the network adjacency matrix A^t, and (2) *Pagerank centrality*: measures the centrality of v_i by incorporating fractions of the centralities of in-neighbors, such that each of v_i's

in-neighbors passes the value of his/her centrality divided by the number of outgoing edges.[6]

3.4 Social Network Analysis

In this section, we report our observations on the computed measures. Table 3.2 shows statistics for the D2web social graph G created according to our description, as well as statistics for (1) the subset of users who have discussed vulnerabilities in D2web (i.e., $vulUsers \subset V$), (2) the subgraph of G induced by the $vulUsers$, $G_{vulns}(vulUsers, vulEdges)$, (i.e., $vulEdges \subset E$), and (3), a subgraph induced by $vulUsers$ as well as all their in- and out- neighbors $G_{vulNei}(vulNeis, vulNeiEdges)$, (i.e., $vulNeis \subset V$ and $vulNeiEdges \subset E$). We create these three graphs to understand the differences in the user connectivity for the subset of users mentioned vulnerabilities (i.e., $vulUsers$, less than 1% of the total population) as opposed to the total user population.

[6]In all the experiments, the damping factor is set to 0.85.

Table 3.2: Statistics for the Graph G With All Users With at Least One Edge (an In-Edge or an Out-Edge), the Subset of Users That Have Discussed Vulnerabilities *vulUsers*, Subgraph ($G_{vulns}(vulUsers, vulEdges)$), and Subgraph ($G_{vulNei}(vulNeis, vulNeiEdges)$).

Property	**D2web users**	**Users mentioning vulnerabilities**		
		as a subset of nodes	as a subgraph	as a subgraph with all 1-hop neighborers
Nodes	53,178	365	365	10,469
Edges	730,740	undefined	1,492	202,070
The average of:				
In-degree	13.74	64.90	4.08	19.30
Out-degree	13.74	51.45	4.08	19.30
Sum of in-degree weights	35.80	250	48.07	75.73
Sum of out-degree weights	35.80	215	48.07	75.73
In-degree centrality	$2.59e^{-4}$	$1.22e^{-3}$	$1.12e^{-2}$	$1.84e^{-3}$
Out-degree centrality	$2.59e^{-4}$	$9.68e^{-4}$	$1.12e^{-2}$	$1.84e^{-3}$
Betweeness centrality	$7.2e^{-5}$	$1.29e^{-3}$	$1.89e^{-4}$	$2.35e^{-4}$
Eigenvector centrality	$2.18e^{-4}$	$4.68e^{-4}$	$5.95e^{-3}$	$8.21e^{-4}$
Pagerank	$1.90e^{-5}$	$1.27e^{-4}$	$2.74e^{-3}$	$9.6e^{-5}$

These individuals, or *vulUsers*, are spread across many D2web forums (46 forums out of 151). Further, they are generally more active than other users in the same forums. As depicted in Table 3.2, the average in-degree (64.90) and out-degree (51.45) for the subset of *vulUsers* are orders of magnitude higher than the same measures for all users in the graph (13.74)—about 5 times higher in-degree and 4 times higher out-degree. This shows that the average hacker in *vulUsers* is exposed to a larger population of hackers than a normal hacker. Expectedly, we observe that a hacker from *vulUsers* is more likely to engage in discussions with another hacker from the same group than he/she does with others. For example, the average in-degree and out-degree (4.08) for the subgraph G_{vulns} indicates that, on average, a user has connections with about 4 other *vulUsers* (about 1% of the population of *vulUsers*); whereas he/she would reply to about 51 users and make about 64 other users reply to his/her post, less than 0.1% of the total population. For these reasons, *vulUsers* generally, exhibit significantly higher centrality and importance measures as compared to normal users in G. However, we observe that the distribution of network measures vary largely within this group of users. We also observe that about 30% of the $vulUsers$ have no communication history with other users within the same group. Figure 3.4 shows a visualization for the subgraph G_{vulns}, which confirms the two observations. Finally, about 25% of $vulUsers$ joined the D2web community less than three days before their first vulnerability mention.

3.5 Experimental Setup

We perform experiments on the set of vulnerabilities mentioned on D2web forums in the period from January 2010 to March 2017. We exclude the vulnerabilities that were mentioned by users with no communication history. We also exclude the vulnerabilities that had been detected in the wild by Symantec before they were

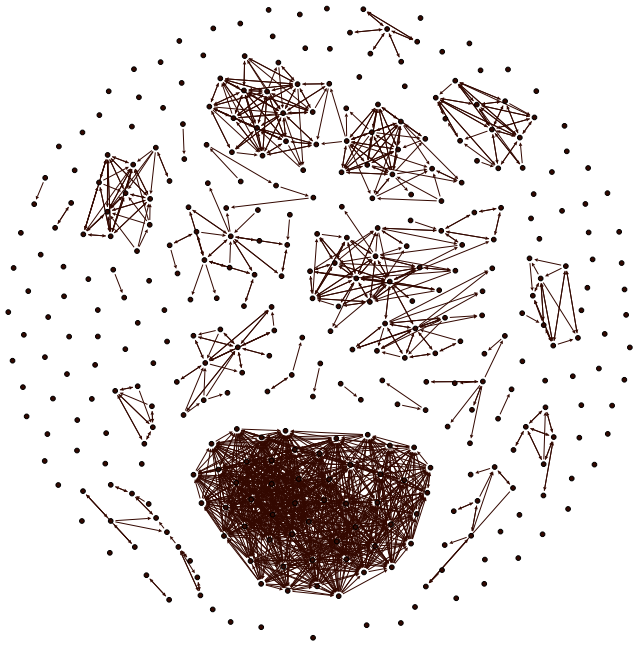


Figure 3.4: The Subgraph of G That Is Induced by the Set of Users Who Have Mentioned Vulnerabilities in Their Postings.

mentioned in any of the D2web posts since those vulnerabilities could be retrieved by querying the database without prediction. Our resultant dataset contains 157 distinct vulnerabilities, 24 of which have the class label *exploited*.

Different machine learning classifiers are compared,[7] but we only report the performance achieved by the best performing classifier, which is Random Forest (RF) [58]. RF is an ensemble of decision trees generated from the training data. Each tree produces a prediction independent of other trees, with the algorithm taking the majority vote at the end. In each of the experiments, we report an averaged performance of five runs.

[7]Classifiers used include: Support Vector Machine (SVM), Logistic Regression, and Naive Bayes.

3.5.1 Performance Evaluation

We evaluate the performance of the predictors using precision, recall, and the harmonic mean of precision and recall, which is F1 measure. As explained in Chapter 2, precision is the fraction of vulnerabilities that were exploited from all vulnerabilities predicted as being exploited, and recall is the fraction of correctly predicted exploited vulnerabilities from the total number of exploited vulnerabilities.

We also compute Receiver Operating Characteristics (ROC) for each run and report Area Under Curve (AUC) of the classifier. ROC graphically illustrates the classification performance by plotting the true positive rate (TPR) against the false positive rate (FPR) at different points of the decision boundary.

3.5.2 Results

Experiments under real-world conditions. In this set of experiments, we sort the vulnerabilities by their D2web date, then we train our classifiers on the vulnerabilities mentioned before June 2016 (125 vulnerabilities), and test on the vulnerabilities from June 2016 to March 2017 (32 vulnerabilities, only 3 are *exploited*). The classification performance achieved by our RF model has an average precision of 0.57, recall of 0.93, and F1 of 0.67. The same classifier is able to achieve on average AUC of 0.95.[8] The lower score of precision is attributed to the fact that Symantec's data is biased towards reporting exploits targeting vulnerabilities that exist in software products from certain software vendors such as Microsoft and Adobe [1]. Since our model is found to predict vulnerabilities as being exploited from other vendors as well, we believe that some false positives were exploited in the wild but never detected by Symantec.

[8]The results of 5 runs show relatively high variance due to the small number of samples on which the models are tested.

Ablation test and cross-validation. Since the number of vulnerabilities in our testing dataset in the previous experiment is relatively small, we further apply stratified 5-fold cross-validation on the whole dataset. In this experiment, the samples are intermixed; hence, these conditions do not reflect the conditions of real-world streaming prediction (i.e., predicting the likelihood of exploitation at the time of the vulnerability mention). The average F1 achieved is 0.72, with a precision of 0.61, a recall of 0.89, and an AUC of 0.88.[9]

To measure the impact of individual feature sets on the overall classification performance, we apply two tests: (1) an ablation test (depicted in Figure 3.5) where the change in precision, recall, F1, and AUC is recorded when each set of features is removed from the prediction model, and (2) a test on individual feature sets (depicted in Figure 3.6) where the classification performance is reported for models trained on only one set of features at a time. In the ablation test, when the set of social network features is removed from the model, some decrease in performance was recorded as depicted in Figure 3.5. In the individual feature tests, the social network measures resulted in an improvement in performance that is significantly higher than the improvement recorded from the inclusion of other feature sets. We note that the simple classifier - which labels all vulnerabilities as being *exploited*, results in a precision of 0.16, a recall of 1, at an F1 of 0.27 and an AUC of 0.5.

3.6 Discussion

In the ablation test, the largest drop in F1 occurred when the CVSS set of features was removed, followed by the removal of social network measures with a comparable drop in F1. It is important to note that CVSS vector is designed to assess the

[9]Standard deviation for the evaluation metrics across the five runs is between 0.02 (F1 and AUC) and 0.06 (recall).

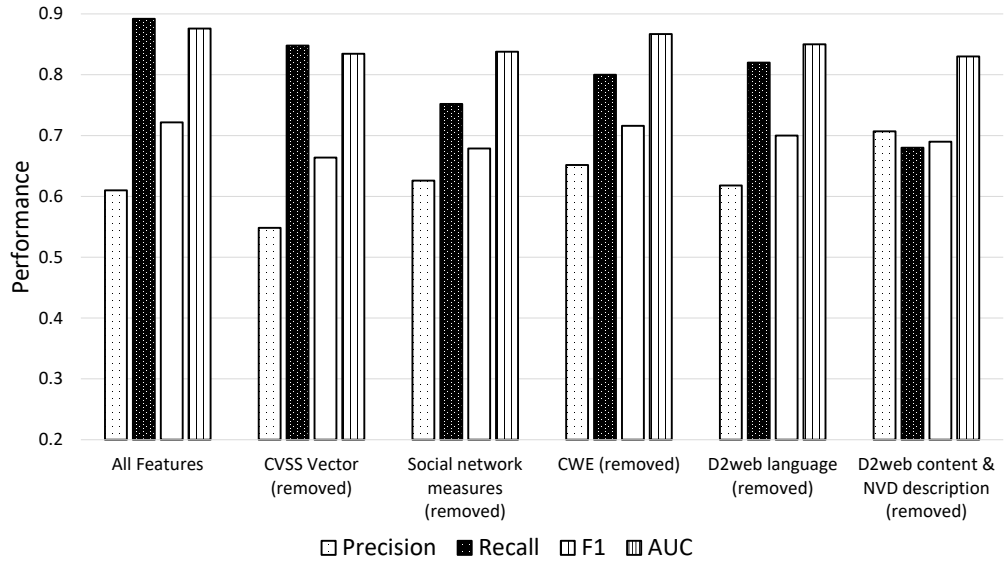


Figure 3.5: The Classification Performance Achieved by Applying Ablation Test With 5-Fold Cross-Validation.

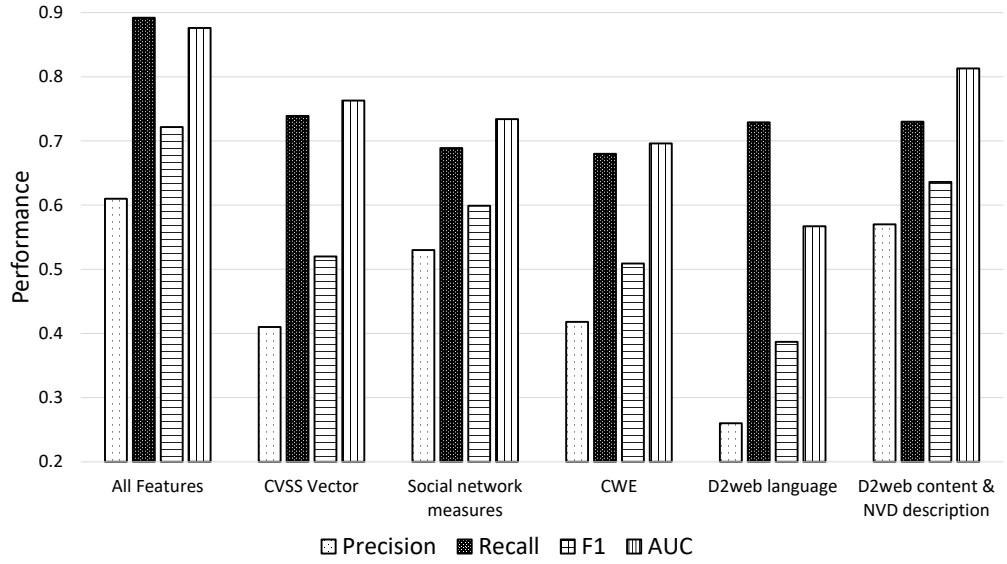


Figure 3.6: The Classification Performance Achieved by Individual Feature Sets.

vulnerability exploitability and impact, and the assignment of values to the different components of the vector is manually done by domain experts.[10] Therefore, being able to achieve performance comparable to the expensive CVSS vector when social network features are used is very promising. However, we do note that even the experiments here where we only use the CVSS vector differ substantially from previous work. As we have selected a-priori on vulnerabilities that have appeared on D2web, we observe a superior performance of CVSS score in this chapter when compared to previous work.

Additionally, when the individual feature sets were examined, the best performing features were the TF-IDF computed from both the content of D2web postings and the vulnerability description retrieved from the NVD. The second-best performing features were the social network measures, scoring F1 that is significantly higher than F1 scores achieved by the other individual feature sets, excluding the textual features. The D2web textual content provides rich information about the context in which the vulnerability is discussed. Furthermore, the software vendor (e.g., Microsoft, Adobe) can be easily derived from the NVD description; leading the model to potentially overfit the biased ground truth. In all experiments, social network measures demonstrated their viability as predictors of potential cyber threats.

3.7 Related Work

3.7.1 Vulnerability Exploitation Prediction

A recent study found that CVSS base score metrics are poor indicators of exploitation [35]. Many vulnerabilities are assigned high scores, resulting in very high false positive rates. Sabottle et al. [1] proposed an SVM classifier that leverages data feeds

[10] `https://www.first.org/cvss/v2/guide`

from Twitter with explicit mentions to legitimate CVE identifiers to predict whether a vulnerability will have proof-of-concepts available in one experimental setup, and predict whether a vulnerability will be detected in the wild in other experimental setup. Results in their work are reported on time-intermixed samples (i.e., samples in testing set may have appeared before samples in training set), and including samples where the exploitation date is before any of the tweets are posted. Both practices have been discussed in [30] to measure their impact on real-world proactive exploitation prediction settings. The work in [30] replicates the experiments done in [1] (with data feeds spanning different periods) to predict the existence of proof-of-concept exploits from EDB. They found that the experimental methodology highly influence the results. In both papers, the dynamics of user connectivity are not studied.

3.7.2 Social Network Analysis

An extensive amount of work has focused on the usage of measures computed from a social network of actors to identify malicious actions [68, 69, 70]. For example, Cao et al. [70] proposed a method called *SybilRank* that relies on social network measures to identify fake accounts (Sybils). Across multi-disciplines, hacker communities in underground hacking forums have been widely studied to understand the dissemination of information among hackers, the motives for hacking, and the reputation and skill level of hackers to detect threats [71, 72, 73]. However, the dynamics of connections within hacker communities have not been quantified as predictors for vulnerability exploitation. We find the measures computed from social connections to be promising predictors of future cyber attacks.

3.8 Conclusion

Our work in this chapter contributes toward understanding user connectivity in D2web forums and extracting measures that serve as predictors of cyberattacks. Using these measures, the developed models predict *if* exploits are going to be detected in the wild. Our experimental results demonstrate that the addition of predictors derived from the social network structure improves recall by about 19% while maintaining precision. Besides, these predictors support models that generalize well with biased ground truth (i.e., under-representing vulnerabilities affecting software from certain vendors).

Chapter 4

A RULE LEARNING-BASED APPROACH TO PREDICT ORGANIZATION-TARGETED EXTERNAL THREATS

4.1 Introduction

The majority of recent cyber incidents, such as data breaches at Equifax, Verizon, Gmail, Instagram, and others [74, 75], are believed to be originated from threat actors sending malicious emails—emails with malicious attachments or with links to destinations that serve malicious content [76, 77]. A 2017 Verizon investigation report stated that 75% of breaches were perpetrated by outsiders exploiting known vulnerabilities [78]. Although the cybersecurity research community constantly demonstrates that these incidents could have been avoided, proactively identifying and systematically understanding *when* and *why* such events are likely to occur is still challenging. We demonstrate in Chapters 2 and 3 how cyberthreat intelligence, collected from various sources such as D2web, is useful for predicting *if* exploits are going to be circulated in the wild. In this chapter, we present an approach to predict *when*, in the future, certain organization-targeted cyberattacks are likely to occur, an important prediction task for cybersecurity analysts, i.e., allowing to prioritize what defense measures to deploy first.

With predictive features and enough data, statistical learning approaches often provide accurate predictions in terms of standard performance measures, such as precision and recall, which, for many prediction tasks, are most desirable. For other tasks, generating transparent and explainable predictions that allow human experts to understand the reasoning that leads to certain predictions could be more valuable

than the predictions themselves [79]. These two notions, accuracy and interpretability, are equally considered in this work.

In this chapter, we describe the technical approach that identifies indicators of certain enterprise-targeted cyberattacks from unconventional sources of threat intelligence (D2web) and uses them to predict attacks in the future. In doing so, we use concepts from logic programming, in particular, the concepts of Point Frequency Function (pfr) from Annotated Probabilistic Temporal Logic (APT- logic) [80, 81, 82]. The rules it learns are of the form "if certain hacker activity is observed in a given time point, then there will be an x number of attacks of type y, targeting organization o in exactly Δt time points, with probability p." Similar to the previous chapters, we obtain real-world hacker discussion data from a commercially available API, maintained by a cyberthreat intelligence firm (called CYR3CON)[1]. We also obtain over 600 historical records of targeted real-world cyberattack incidents. These incidents are recorded from the logs of two large enterprises participating in the IARPA Cyberattack Automated Unconventional Sensor Environment (CAUSE) program.[2]

During the execution of the CAUSE program, our approach was integrated into a deployed system that submitted real-time warnings that originated from multiple predictive models. Our original approach—explained in this chapter and originally developed and presented in [28]—produced warnings often connected to a single source. However, the ephemeral nature of many D2web sources led to challenges in modeling and predicting over an extended period of time. Therefore, we extend the capabilities of the original approach using indicators that capture aggregated discussion trends

[1] `https://cyr3con.ai`

[2] `https://www.iarpa.gov/index.php/research-programs/cause`

across multiple and additional hacker community platforms (see Section 4.9). These platforms include D2web as well as environments such as Chan sites[3] and social media.

The main goal that devised the design of the current approach was to generate warnings that predict *when* cyberattacks that are likely to occur. These warnings are required to be:

- **Timely**: to indicate the exact point in time when a predicted attack will occur;
- **Actionable**: to provide metadata/warning details, i.e., the target enterprise, type of attack, volume, software vulnerabilities and tags identified from the hacker discussions;
- **Accurate**: to predict unseen real-world attacks with an average increase in F1 of over 45% for one enterprise and 57% for the other, compared to a baseline approach; and
- **Transparent**: to allow analysts to easily trace the warnings back to the rules triggered, discussions that fired the rules, etc.

The rest of the chapter is organized as follows. Section 4.2 presents related work. Section 4.3, presents technical preliminaries formally explaining our logic programming approach. Section 4.4 provides detailed explanation of the data. Processing flow is described in Section 4.5. Section 4.6 offers an overview of the design and components of the developed approach. Empirical self-assessment results are provided in Section 4.7.3. Section 4.8 introduces technical challenges that arose with predictions over an extended period of time. Section 4.9 provides a technical extension over the developed approach to address these challenges. Section 4.10 concludes this chapter.

[3]A type of Internet forums, mostly image boards, that encourage visitors to anonymously post content. Some Chan sites are found to be leveraged by activists, such as the well-known hacking activist group Anonymous.

4.2 Related Work

The task of selecting and deploying cybersecurity countermeasures is generally expensive [83, 84, 85]. Therefore, the problem of predicting cybersecurity-related events has gained a growing interest. Yet, much of the current literature focuses on producing accurate predictions. Our work, however, considers other goals, such as producing interpretable predictions that support human-in-the-loop-driven decisions. This section reviews works that are related to both these goals.

Predicting cyberattack events. Recently, predicting cybersecurity events has received an increasing attention [86, 22, 87]. For example, Soska and Christin [22] developed an ML-based approach that predicted whether a given website will turn malicious in the future using features derived from the webpage structure as well as content and traffic statistics. Their approach was evaluated on a corpus of 444,519 websites (highly imbalanced, with only about 1% of the sites belonging to the positive class). The approach achieved a true positive rate of 66% and a false positive rate of 17%. Although they used the C4.5 decision tree classifier, the predictions were made by a single-layered ensemble approach using 100 features. The authors reported that the classification of non-malicious sites was generally less trivial. Other studies focused on predicting cybersecurity events of certain types, such as vulnerability exploitation [1, 31, 30, 29]. In [31], the authors proposed ML classifiers that predicted the likelihood of vulnerability exploitations in the future. They tested a population of over 12,000 software vulnerabilities using features computed from the activities of white-hat and black-hat hacking communities following official vulnerability disclosures. The proposed method outperformed the widely-used standard severity scoring system (a.k.a., CVSS[4]), with F1 more than doubled. These studies

[4] `https://www.first.org/cvss`

focused on vulnerability-targeted attacks, whereas our focus is on attacks targeting particular commercial enterprises. Similar to our prediction task, the works presented in [26, 27, 88] focused on (1) identifying and analyzing enterprise-targeted attack indicators from online cybersecurity-related discussions, and (2) producing predictions of possible future events. These studies identified attack indicators from (1) hacker sentiments from posts in hacking forums [26], (2) word-counts from hacker discussions on D2web, blogs, and Twitter [27], or (3) social network structure generated from D2web forum discussions [88]. All these works used ML approaches solely focusing on producing accurate predictions, while we consider predictions that are accurate and transparent.

Supporting interpretable decisions. Knowledge representation and reasoning (KRR) supports formally explainable reasoning, which is desired for many applications, including cybersecurity incidents prediction [89, 90]. Nunes et al. [15] developed an argumentation model for cyber-attribution using a dataset from the capture-the-flag event held at DEFCON,[5] a famous hacking conference. The model was based on a formal reasoning framework called Defeasible Logic Programming [91]. Using a two-layered hybrid KRR-ML approach, the ML classification accuracy increased from 37% to 62%. While their approach supported automated reasoning, it was used for cyber-attribution only *after* the attacks were observed. Moreover, human-driven classification was not a desirable propriety. Instead, the reasoning framework was used to reduce the search space, thereby improving accuracy. Furthermore, Marin et al. [92] investigated user adoption behavior to predict in which topic of a darkweb hacker forum will users post in the future, given the influence of their peers. The authors formulated the problem as a sequential rule mining task [93], where the goal is to mine for the user posting rules through sequences of user posts and produce

[5] `https://www.defcon.org`

predictions. Each rule of the form $X \Rightarrow Y$ is interpreted as follows "if X (a set of hackers) engages in a given forum topic, Y (a single hacker) is likely to engage in the same topic (or adopt it) with a given confidence afterward, mainly because of the influence of X." They obtained prediction precision results of up to 0.78, with a precision gain approaching 800%, compared to a baseline created with the prior probabilities of hacker posts. While their approach is rather impressive, they addressed a prediction task that is different from ours.

4.3 Preliminaries

In this section, we define the syntax and semantics of Annotated Probabilistic Temporal Logic (APT-logic) applied to our domain, which is built upon the earlier work of Shakarian et al. [81].

4.3.1 Syntax

Herbrand base. We use $B_{\mathcal{L}}$ to denote the Herbrand base (a finite set of ground atoms) of a first order logical language $\mathcal{L}$. Then, we divide $B_{\mathcal{L}}$ into two disjoint sets: $B_{\mathcal{L}\{conditions\}}$ and $B_{\mathcal{L}\{actions\}}$, so that $B_{\mathcal{L}} \equiv B_{\mathcal{L}\{conditions\}} \cup B_{\mathcal{L}\{actions\}}$. $B_{\mathcal{L}\{conditions\}}$ comprehends the atoms allowed only in the premise of APT rules, representing *conditions* or user activity performed on hacker community websites, e.g.,

$$mention\_on(forum\_1, debian)$$

On the other hand, $B_{\mathcal{L}\{actions\}}$ comprehends the atoms allowed only in the conclusion of APT rules, representing *actions* or malicious activities reported by the data

providers in their own facilities, e.g.,

$$attack(data\_provider, malicious\_email, x)$$

Formulas. Complex sentences (formulas) are constructed recursively from atoms, using parentheses and the logical connectives ($\neg$ negation, $\vee$ disjunction, $\wedge$ conjunction).

Time formulas. If F is a formula, t is a time point, then F_t is a time formula, which states that F is true at time t.

Probabilistic time formulas. If ϕ is a time formula and $[l, u]$ is a probability interval $\subseteq [0, 1]$, then $\phi : [l, u]$ is a probabilistic time formula *(ptf)*. Intuitively, $\phi : [l, u]$ says ϕ is true with a probability in $[l, u]$, or using the complete notation, $F_t : [l, u]$ says F is true at time t with a probability in $[l, u]$.

APT rules. Suppose condition F and action G are formulas, t is a natural number, $[l, u]$ is a probability interval and $fr \in \mathcal{F}$ is a frequency function symbol that we will define later. Then $F \overset{fr}{\rightsquigarrow} G : [t, l, u]$ is an APT (Annotated Probabilistic Temporal) rule, which informally saying, computes the probability that G is true in exactly Δt time units after F becomes true. For instance, the APT rule below informs that the probability the data provider is being attacked by a malicious email, in exactly 3-time units after users mention *"debian"* on *forums_1*, is between 44% and 62%.

$$\begin{aligned} &mention\_on(set\_forum\_1, debian) \overset{pfr}{\rightsquigarrow} \\ &\quad attack(data\_provider, malicious\_email) \ : \ [3, 0.44, 0.62] \end{aligned} \tag{4.1}$$

4.3.2 Semantics

World. In general, a world is a set of ground atoms that belongs to $B_{\mathcal{L}}$. It describes a possible state of the (real) world being modeled by an APT-logic program. Some possible worlds in our context are:

- $\{spike(Amazon\_AWS)\}$,
- $\{mention\_on(forum\_1, debian), attack(data\_provider, malicious\_email, x)\}$,
- $\{attack(data\_provider, malicious\_email, x)\}$,
- $\{\}$

Thread. A thread is a series of worlds that models the domain over time, where each world corresponds to a discrete time-point in $\mathcal{T} = \{1, ..., t_{max}\}$. $Th(i)$ specifies that according to the thread Th, the world at time i will be $Th(i)$. Given a thread Th and a time formula ϕ, we say Th satisfies ϕ (denoted $Th \vDash \phi$) iff:

- If $\phi \equiv F_t$ for some ground time formula F_t, then $Th(t)$ satisfies F;
- If $\phi \equiv \neg\rho$ for some ground time formula ρ, then Th does not satisfy ρ;
- If $\phi \equiv \rho_1 \wedge \rho_2$ for some ground time formulas ρ_1 and ρ_2, then Th satisfies ρ_1 and Th satisfies ρ_2;
- If $\phi \equiv \rho_1 \vee \rho_2$ for some ground time formulas ρ_1 and ρ_2, then Th satisfies ρ_1 or Th satisfies ρ_2;

Frequency functions. A frequency function represents temporal relationships within a thread, checking how often a world satisfying formula F is followed by a world satisfying formula G. Formally, a frequency function fr belonging to $\mathcal{F}$ maps quadruples

of the form (Th, F, G, t) to [0,1] of real numbers. Among the possible ones proposed in [80], we investigate here alternative definitions for the *point frequency function (pfr)*, which specifies how frequently action G follows condition F in "exactly" Δt time points. To support ongoing security operations, we need to relax the original assumption of a finite time horizon t_{max} in [80, 81]. Therefore, we introduce here a different but equivalent formulation for *pfr* that does not rely on a finite time horizon. To accomplish that, we first need to define how a *ptf* can be satisfied in our model. If we consider A as the set of all ptf's satisfied by a given thread Th, then we say that Th satisfies $F_t : [l, u]$ (denoted $Th \vDash F_t : [l, u]$) iff:

- If $F = a$ for some ground a, then $\exists\, a_t : [l', u'] \in A$ *s.t.* $[l', u'] \sqsupseteq [l, u]$;
- If $F_t : [l, u] = \neg F'_t : [l, u]$ for some ground formula F', then $Th \vDash F'_t : [1-u, 1-l]$;
- If $F_t : [l, u] = F'_t : [l, u] \wedge F''_t : [l, u]$ for some ground formulas F' and F'', then $Th \vDash F'_t : [l, u]$ and $Th \vDash F''_t : [l, u]$;
- If $F_t : [l, u] = F'_t : [l, u] \vee F''_t : [l, u]$ for some ground formulas F' and F'', then $Th \vDash F'_t : [l, u]$ or $Th \vDash F''_t : [l, u]$;

The resulting formulation of *pfr* is shown in Equation 4.2, which is equivalent to the original one proposed in [80] when t_{max} comprises the whole thread Th (all time points):

$$pfr(Th, F, G, \Delta t) =$$

$$\left[\frac{\sum\limits_{t|Th \vDash F_t:[l,u] \wedge Th \vDash G_{t+\Delta t}:[l',u']} l'}{\sum\limits_{t|Th \vDash F_t:[l,u]} u}, \frac{\sum\limits_{t|Th \vDash F_t:[l,u] \wedge Th \vDash G_{t+\Delta t}:[l',u']} u'}{\sum\limits_{t|Th \vDash F_t:[l,u]} l} \right] \tag{4.2}$$

Satisfaction of APT rules and programs. Th satisfies an APT rule $F \overset{pfr}{\rightsquigarrow} G : [\Delta t, l, u]$ (denoted $Th \vDash F \overset{pfr}{\rightsquigarrow} G : [\Delta t, l, u]$) iff:

$$pfr(Th, F, G, \Delta t) \subseteq [l, u] \tag{4.3}$$

Probability intervals. For this application, the possible values for l, l', u, and u' are either 0 or 1. Therefore, the rules learned using Equation 4.2 always have point probabilities. To derive a probability interval $[l, u]$ corresponding to a point probability p of rule r, we use standard deviation (i.e., σ) computed from the binomial distribution—remember that the possible outcome of event G following event F is either 0 or 1. We subtract/add one standard deviation from/to the point probability to determine the lower/upper bounds of the probability range, i.e., $[p - \sigma, p + \sigma]$. The standard deviation is computed as follows:

$$\sigma = \frac{\sqrt{support\_F * p * (1 - p)}}{support\_F} \tag{4.4}$$

where $support\_F$ is the number of times the precondition or F is observed. For example, the precondition of rule 4.1 was satisfied by the thread 32 times. Of these, 17 times the postcondition of the rule was also satisfied, resulting in a point probability of approximately 0.53. The value of σ is approximately 0.09, hence the probability range $[0.44, 0.62]$.

4.4 Dataset Description

This section explains the ground truth data, obtained from the data providers, and provides discussions about the data collection infrastructure that supplies hacker discussion data feeds.

4.4.1 D2web Crawling Infrastructure

Similar to the data used in Chapters 2 and 3, here we summarize the D2web crawling infrastructure that CYR3CON maintains—originally introduced in [18]. Customized lightweight crawlers and parsers were built for each site to collect and extract data. Data is collected from more than 300 platforms (forums and marketplaces). To ensure the collection of relevant data, machine learning models are used to only retain discussions related to cybersecurity and omit irrelevant data.

4.4.2 Enterprise-Relevant External Threats

To construct rules and evaluate the performance of the learned model, we use data from historical records of attack attempts that are recorded from the logs of two enterprises participating in the IARPA CAUSE program. One of the two enterprises is a defense industrial base (referred to as Armstrong) while the other is a financial services organization (referred to as Dexter). The database is distributed to the CAUSE performers in increments, once every few months. Each data point is a record of a detected deliberate, malicious attempt to gain unauthorized access, alter or destroy data, or interrupt services or resources in the environment of the participating organizations. Those malicious attempts were detected in an uncontrolled environment and by different security defense commercial products such as anti-virus, intrusion detection systems, and hardware controls. Each ground truth (GT) record includes *ID, Format Version, Reported Time, Occurrence Time, Event Type, and Target Industry*[6]. The types of attacks included in the GT dataset are:

[6]We intentionally skip some details about other fields of the GT records due to the limitation in space and irrelevance to the scope of this chapter.

- **Malicious email (M-E).** A malicious attempt is identified as a Malicious Email event if an email is received by the organization, and it either contains a malicious email attachment, or a link (embedded URL or IP address) to a known malicious destination.

- **Malicious destination (M-D).** A malicious attempt is identified as a visit to a Malicious Destination if the visited URL or IP address hosts malicious content.

- **Endpoint malware (E-M).** A Malware on Endpoint event is identified if malware is discovered on an endpoint device. This includes, but not limited to, ransomware, spyware, and adware.

Other events related to insider threats are out of the scope of this work. A summary of the time periods and the number of records for each attack type is provided in Section 4.7.3.

4.5 Extracting Indicators of Cyberthreat

Common Vulnerabilities and Exposures (CVE) numbers are unique identifiers assigned to software vulnerabilities reported in the National Vulnerability Database (NVD). Common Platform Enumeration (CPE) is a list of software/hardware products that are vulnerable to a given CVE. CPE data can be obtained from the NVD. We query the database using API calls to look for postings with software vulnerability mentions (in terms of CVE numbers). Regular expressions are used to identify CVE mentions.[7] We map each CVE to pre-identified groups of CPEs. Each is a set of CPEs belonging to similar software vendors and/or products. We identified over 100 groups of CPEs, e.g., *Microsoft Office*, *Apache Tomcat*, and *Intel*. Moreover, CVEs

[7]For more explanation, see `https://cve.mitre.org/cve/identifiers/syntaxchange.html`.

are mapped to some nation-state threat actors who are known to leverage certain CVEs as part of their attack tactics—perhaps among the most well-known threat actors is the North Korean group *HIDDEN COBRA*, which was recently identified to be responsible for an increasing number of cyberattacks to US targets [94]. This mapping is determined based on an encoded list of threat actors along with vulnerabilities they favor. The list is compiled by manually analyzing cyberthreat reports that were published by cybersecurity firms.[8] The final CPE groupings and nation-state actor mappings are used as preconditions by the *learner*.

4.6 A Novel Logic Programming-Based Cyberthreat Prediction System

This section provides discussions about the components of our state-of-the-art prediction system, as well as the input and output data. Fig. 4.1 shows the system design, which has two main components: the *learner* and the *predictor*.

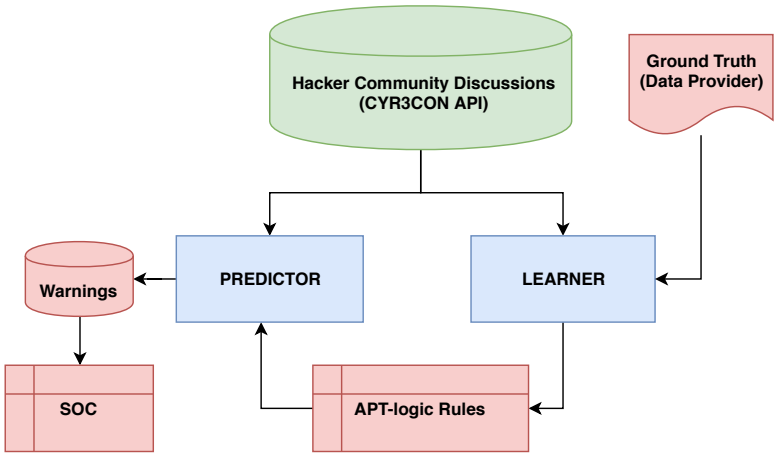


Figure 4.1: Logic Programming-Based Cyberthreat Prediction System

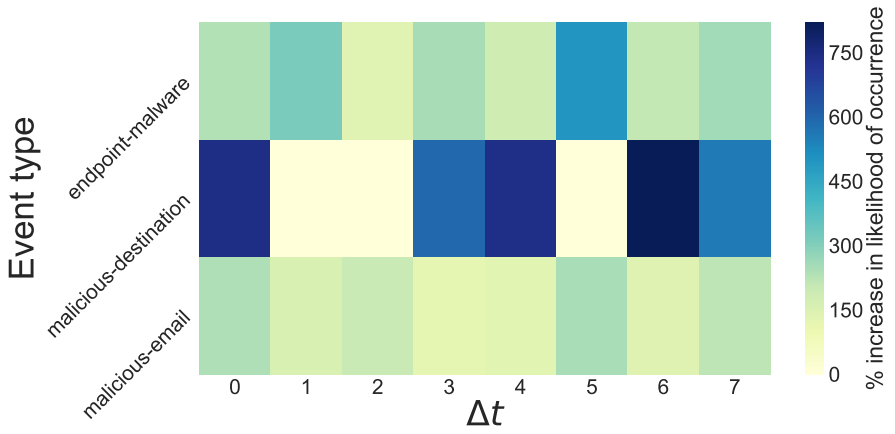
[8]See the Kaspersky Lab's 2016 report as an example, `https://media.kaspersky.com/en/business-security/enterprise/KL_Report_Exploits_in_2016_final.pdf`

4.6.1 Learner

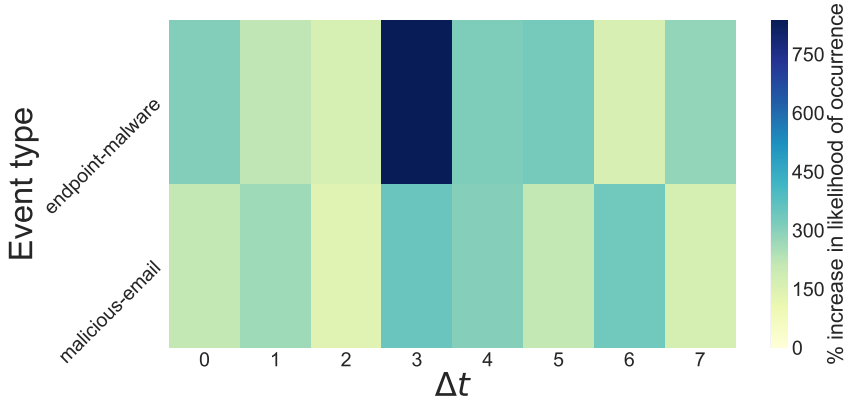
The *learner* learns APT-logic rules that link indicators of cyberthreats and real-world attack events. The indicators of threats are annotated from a collection of hacker discussions with vulnerability mentions, following the approach discussed in Section 4.5. The real-world attack events are cyberattack attempts observed by the data providers. Each event, a CPE group or an attack attempt, is annotated with the date when the corresponding vulnerability was mentioned or when the incident was recorded. These dates are mapped to discrete time-points to construct the *thread*, which is used in the rule learning approach discussed in Section 4.3. The output of the *learner* is an APT-logic program, i.e., a set of APT rules. These rules, along with indicators annotated from the hacker community discussions are used by the *predictor* to produce warnings. Figure 4.2 depicts the percentage increase in the likelihood of attacks, from the rules learned, compared to the prior probability of attacks—the probability of attacks with no knowledge of hacker activity. The increase is significant, which is a promising observation for accurate predictions

4.6.2 Predictor

The *predictor* uses the output of the *learner*, i.e., the APT-logic program and the indicators annotated from D2web hacker discussions. It triggers rules if any indicators are observed that match the preconditions of the rules in the APT-logic program [28]. If a match exists, the system generates a warning with metadata, such as the probability, event type, and target organization. This allows analysts to easily trace the warnings back to the rules that were triggered, discussions that fired the rules, etc. Figure 4.3 shows two screenshots taken from a deployed system that uses our approach.



(a) Armstrong



(b) Dexter

Figure 4.2: Percentage Increase in Attack Likelihood Over Attack Prior Probability for the Learned Rules, per Δt per Event Type, and for the Two Companies (a) Armstrong, and (B) Dexter.

Figure 4.3: Two Screenshots From a Deployed System That Uses Our Approach.

4.7 Predicting Enterprise-Targeted Attacks

During phase 2 of the IARPA CAUSE program, our approach was integrated into a deployed system that generates, fuses, and actively submits warnings to a SOC for performance evaluation purposes. Details about the deployed system, provided in [28], are out of the focus of this chapter, but we follow some of the evaluation practices adopted by the CAUSE program (i.e., using the subset of the evaluation metrics that are relevant to the prediction task). Furthermore, the warnings that are submitted by each system are evaluated by the SOC on a monthly basis. However, the said evaluations are aggregated for all models belonging to a single system. To evaluate our approach, it needs to be isolated from other models used by the same system. In this section, we internally evaluate our approach before the warnings are fused with warnings from other models.

4.7.1 Experimental Settings

We perform evaluations on the warnings targeting Armstrong that were submitted during July, August, and September of 2017. The results are aggregated by months for the experiments on Armstrong data while aggregating by periods of 7 days for Dexter. The latter starts from July 1 to July 28, 2016. These time windows differ because the Armstrong dataset covers a longer period than the one covered by Dexter, and there is no more Dexter data provided or evaluated by the program. The reported records of Malicious Destination for Dexter only cover a period that ends before the testing period starts, hence they are not evaluated. For each data provider, rules are learned from all the records preceding the testing period.

4.7.2 Evaluation

Pairing ground truth events with warnings. To receive a score, each warning needs to be paired up with a single ground truth event occurring within the same day, or one day after the attack prediction date, i.e., a 1-to-1 relationship—this is a requirement by the CAUSE program.[9] To do so, we use the Hungarian assignment algorithm [95] to solve the warning-to-ground truth assignment problem, with the objective to maximize warning-to-attack lead time. The results of the Hungarian algorithm (i.e., warning-to-ground truth assignments) are used to evaluate the performance of the system. The same approach is used with predictions produced by the baseline model.

Evaluation metrics. We use the standard evaluation metrics: precision, recall, and F1. Precision is the fraction of warnings that match ground truth events, recall is the fraction of ground truth events that are matched, and F1 is the harmonic mean of precision and recall. Using these metrics, we present a performance comparison between our approach and a baseline model. Additionally, we show that a fused model can benefit from the temporal correlations and statistical characteristics captured by the system and the baseline model, respectively.

4.7.3 Results

We found that our approach outperforms a baseline system that randomly generates a x number of warnings on each day such that each value of x has a chance proportional to its frequency of occurrence in the historical data. We repeat the baseline for 100 runs and take the average of each metric. In the real-time deployment of our approach, human experts can evaluate the warnings by leveraging the other capa-

[9] See [28] for an elaborate explanation.

bilities of the system, i.e., *transparency* and *actionable* through a Web UI dashboard. However, in those experiments, any triggered rule is counted, which is not necessarily important given other details. That said, our approach scored significantly higher than the baseline system as shown in Table 4.1.

Table 4.1: Evaluation Results.

Dataset	type	Testing starts	#GT-events	Our approach				Baseline* (average of 100 runs)				%increase in F1
				#warnings	Precision	Recall	F1	#warnings	Precision	Recall	F1	
Armstrong	M-E	Jul-17	24	32	0.313	**0.417**	**0.357**	11.759	**0.417**	0.205	0.271	32%
		Aug-17	11	3	**1.000**	0.273	**0.429**	11.966	**0.289**	0.315	0.299	43%
		Sep-17	13	18	0.167	0.231	0.194	12.793	**0.249**	**0.249**	**0.247**	-21%
	M-D	Jul-17	4	12	**0.167**	**0.500**	**0.250**	3.534	0.099	0.091	0.090	178%
		Aug-17	9	23	0.174	**0.444**	**0.250**	3.121	**0.232**	0.086	0.120	108%
		Sep-17	3	10	**0.100**	**0.333**	**0.154**	2.948	0.071	0.075	0.068	126%
	E-M	Jul-17	14	10	0.300	**0.214**	**0.250**	8.552	**0.326**	0.200	0.242	3%
		Aug-17	18	45	0.200	**0.500**	**0.286**	9.155	**0.324**	0.168	0.217	32%
		Sep-17	17	21	**0.286**	**0.353**	**0.316**	8.879	0.247	0.127	0.164	93%
Dexter	M-E	1-Jul-16	2	13	0.150	**1.000**	**0.267**	2.720	**0.157**	0.205	0.169	58%
		8-Jul-16	7	10	0.500	**0.714**	**0.588**	2.610	**0.655**	0.253	0.348	69%
		15-Jul-16	9	6	0.333	**0.222**	0.267	2.770	**0.619**	0.188	**0.276**	-3%
		22-Jul-16	4	2	**0.500**	0.250	0.333	3.050	0.469	**0.355**	**0.385**	-14%
	E-M	1-Jul-16	1	2	**0.500**	**1.000**	**0.667**	1.790	0.189	0.330	0.226	195%
		8-Jul-16	3	4	**0.250**	**0.333**	**0.286**	1.750	0.245	0.167	0.186	54%
		15-Jul-16	3	1	**1.000**	**0.333**	**0.500**	1.740	0.281	0.190	0.217	130%
		22-Jul-16	4	2	**0.500**	**0.250**	**0.333**	1.780	0.383	0.208	0.257	30%

*A Simple Baseline Model That Generates x Number of Warnings on Each Day Based on the Prior Probability of Each Possible Value of x That Was Seen in the Training Data.

The desired non-functional requirements related to the generated warnings (i.e., timely, actionable, accurate, and transparent, as discussed in Section 4.1), need to be maintained over time. Due to various factors related to both intelligence data (the ephemeral nature of D2web sites) and enterprise data (data from a Security Information Event Manager or SIEM, which can be subject to schema differences due to policy changes over time), we examine further requirements for our approach.

Changing volume of cyberthreat intelligence data. In many applications of event prediction, the volume of signals from the monitored sensors is assumed to remain the same across the learning and the predictive phases. However, this assumption does not hold for cyberthreat intelligence data. This is mainly because of the ephemeral nature of D2web sites, which is caused by reasons such as law enforcement actions, malicious hackers going "dark", operational security measures employed by cybercriminals, and differences induced by adding newer data sources. In the approach discussed so far, changes in the volume of incoming cyberthreat intelligence data would have a direct impact on the number of warnings, affecting the system's performance.

Concept drift. Hacking tactics advance very rapidly to react to the latest advances in cybersecurity, i.e., new vulnerabilities are discovered, new exploits are integrated with malware kits, attack signatures are identified, etc. Likewise, the attacks observed in the wild and the activities of hackers on hacker community websites, including social media, are always evolving [30]. This change in the underlying data distribution for both the hacker discussions and the predicted events is known as "concept drift" [96].

4.9 An Extension to the Current Approach

The approach presented thus far produced warnings connected to single sources. To account for the challenges presented in Section 4.8, we extend upon the variety of sources used by considering cyberthreat intelligence sources from sources spanning hacker communities around the globe, including environments such as Chan sites, social media, paste sites,[10] Gray hat communities,[11] D2web, and even surface web. This includes over 400 platforms and over 18 languages. Non-English postings are translated to English using various language translation services.

As noted, changes in the volume of the incoming cyberthreat intelligence data can impact the volume of warnings produced. In this extension, we consider indicators that are evaluated based on the volume of hacker discussions, i.e., capturing discussion trends exceeding thresholds computed from a sliding time window—this approach is further discussed in this section. To account for the potential impact of concept drift, in each month we run our learner on data from the previous 6 months, and use the resulting rules to predict events in the examined month, as explained in Section 4.7.3.

4.9.1 Extracting Entity Tags

The threat intelligence sources we use supply a vast amount of textual content over time. We utilize a commercial natural language processing API, TextRazor,[12] which leverages a wide range of machine learning techniques (including Recurrent Neural Networks) to recognize entities from the context of postings. Each extracted entity

[10]Online text-hosting services that allow users to host content in plain text, such as source code snippets and data dumps, and obtain links to the content, often called pastes, to share them on other online platforms. Pastes are often found in hacker discussions.

[11]Hackers who exploit software weaknesses in a computer system without the owner's permission or knowledge. The goal is often to bring the weakness to the owner's attention.

[12]`https://www.textrazor.com`

is associated with a confidence score quantifying the confidence in the annotation. We set a lower bound on the confidence score to retain only those entities that are relevant. Two steps are taken to extract the final indicators: (1) annotating spikes in the volume of individually extracted tags, and (2) for those tags, identifying sets that frequently spike together.

Annotating spiking tags. We seek to gain an understanding of abnormal hacker activities that could correlate with attack events. To do so, we define what abnormal activities are, and use them as preconditions of APT-logic rules. They may or may not correlate with actual attack events, but the APT-logic program will only contain the rules whose precondition is found to correlate with the attack events. To identify such abnormalities, we consider common entity tags that appear on most days, i.e., on 90 days or more, because training periods are always 180 days. An entity is regarded as abnormal if it is observed on a given day with a spiking volume. Spikes are determined using statistical control charts [97], i.e., when the count of times an entity is observed exceeds a moving median added to a multiplier of a moving standard deviation.[13]

For instance, let F be an itemset, i.e.,

$$F = \{spike(f_1), \ldots, spike(f_n) \mid \forall i \in \{1, \ldots, n\} : f_i \in A_{var}\}$$

We assume the existence of three utility functions:

1. $count(f, t)$, which returns the number of time an entity f is extracted on day t,
2. $median(f, t, window)$, which returns the median of set S:

$$S = \{count(f, i) \mid i \in \{t - window, \ldots, t\}\}$$

[13] We use a sliding window of 20 days.

3. $stDiv(f, t, window)$, which returns the standard deviation of S.

The thread Th satisfies a predicate $spike(f)$ at some time point t, denoted $Th(t) \vDash spike(f)$ iff:

$$count(f, t) > (median(f, t, window) + (multiplier \times stDiv(f, t, window)))$$

Frequent itemset mining. As explained, preconditions could be atoms or formulas (i.e., an itemset). We only consider those formulas that are frequently satisfied in the historical data. To do so, we use the Apriori algorithm [98]. The Apriori algorithm takes as input a database of transactions—the annotated spiking tags are grouped by days, each day corresponds to a transaction. The algorithm then produces all itemsets of hacker activities that are frequently observed together. The identified itemsets are considered as preconditions and used by both the learner and the predictor.

4.9.2 Results

Fusion. For these experiments, we use a simple combining strategy to test the performance of a fused model. We first combine the warnings from the two models, i.e., our approach and the baseline. The warnings are grouped by their generation date and prediction data. Then, half of the warnings are removed from each group. The goal is to leverage the power of the individual approaches while limiting their intersection, i.e., removing half of the duplicate warnings.

Performance comparison. Fig. 4.4 shows the precision-recall curve for each of the testing months. By itself, our approach performs comparably to the baseline in terms of F1—specifically providing higher precision in the case of lower recall. We note that when our approach is combined with the baseline, the results improve further. The combined approach can significantly outperform the baseline in terms of

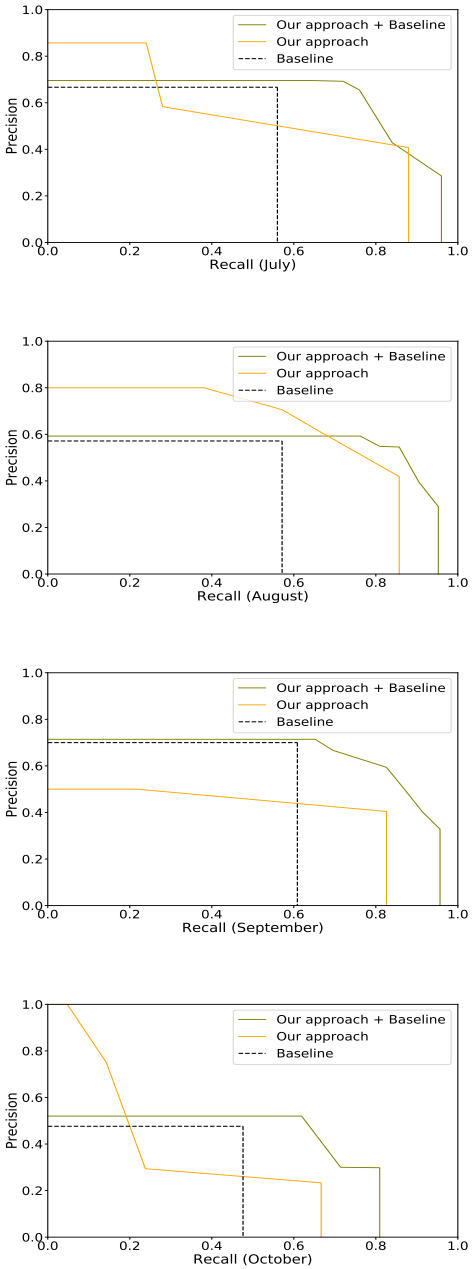


Figure 4.4: Precision-Recall Curves for the Fused Approach, Our Approach, and the Baseline Model, Respectively for Four Months: July, August, September, and October.

both precision and recall, yielding a recall increase of at least 14%, while maintaining precision. Furthermore, the baseline does not allow for a tradeoff between precision and recall while our approach produces warnings with probability values—as discussed in Section 4.3, enabling not only better tradeoff between performance metrics, but also a metric approximating the importance of each warning and allowing human analysts to prioritize investigation.

Transparent predictions. Our approach supports transparent predictions so that the user knows why certain warnings are generated. The user can trace back to the rule corresponding to a warning, and view its precondition. Table 1 shows a few examples of preconditions of rules that generated warnings preceding attack incidents. The user can further pinpoint the collection of hacker discussions that are responsible for the warning. For example, Fig. 4.5 shows a word cloud generated from the collection of posts resulting in a warning submitted on August 23. The warning predicts an event on August 25, i.e., Δt of 2. An event of malicious email is then observed by Armstrong on August 26.

Table 4.2: Examples of Preconditions of Rules That Would Have Generated Warnings Preceding Attack Incidents.

Precondition	Probability	σ	Warning date	Lead time [days]
spike(Credit card) $\wedge$ spike(Gmail)	0.88	0.07	Aug. 26	1
spike(Email) $\wedge$ spike(Security hacker)	0.86	0.08	Aug. 16	1
spike(Google Play)	0.92	0.04	Aug. 13	2

4.10 Conclusion

This chapter presents a novel approach used in a system that predicts certain types of cyberattacks targeting specific commercial enterprises. The chapter explains the underlying logic programming framework (APT-logic) used to model the probabilistic temporal relationships between hacker activities (from hacking community online platforms) and attack incidents (recorded by the SIEM the data providers).

Figure 4.5: A Word Cloud Generated From the Text of Postings that Resulted in a Positive Warning on August 23.

The developed approach uses APT-logic to first learn such relationships, captured in annotated rules, then use the learned rules in a deductive approach to reason about the possibility of future cyberattacks and generate warnings.

Moreover, this chapter addresses limitations of the original version of the approach, which used indicators of future attacks connected to single D2web sources—an approach no longer optimal to use because of the changing volume of intelligence data and the ephemeral nature of D2web sites. There are multiple reasons behind the changing landscape of D2web sites, such as law enforcement actions, malicious hackers going "dark", operational security measures employed by cyber-criminals, and differences caused by the newly added data sources. Therefore, this chapter (1) extends the sources used in [28] by using sources from other platforms such as social media and surface web, and (2) introduces an alternative approach considering indicators that are evaluated based on volume of discussion trends exceeding a threshold computed from a sliding time window.

We demonstrate the viability of our approach by comparing it to a time series prediction baseline model. Specifically, we show that our approach performs comparably to the baseline model while supporting a favorable precision-recall tradeoff and

transparent predictions. Additionally, our system can benefit from the predictions produced by the baseline model. With the combined approach, recall improves by at least 14% compared to the baseline model. Finally, we looked into using the system for data recorded by other data providers, and using intelligence data gathered not only from expert-hunted sources, but also from sources gathered by web spiders.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

This dissertation addresses limitations in the current widely-adopted, industry-standard systems for prioritizing tasks related to cyber-defense. Our goal is to proactively identify cyberthreats. This requires the ability to predict attacks, which is achieved by developing AI techniques that harness cyberthreat intelligence. We accomplished these goals by developing three pieces of research work: predicting if attacks will occur by examining what cyberthreat intelligence feeds are observed, predicting if attacks will occur by examining who generates these cyberthreat intelligence feeds, and predicting when attacks will likely occur in the future.

In Chapters 2 and 3, we consider the problem of predicting exploits in the wild for known software vulnerabilities. Chapter 2 presents ML-based models that leverage cyberthreat intelligence collected from four data sources: vulnerability databases, proof-of-concept exploit archives (White-hat community), vulnerability disclosures on enterprise websites (commercial firms), and over 150 malicious hacking forums and marketplaces in the darkweb and deepweb (D2web—Black-hat community). We view the problem as a binary classification problem (i.e., the positive class is *will be exploited*). The developed models outperformed the standard vulnerability severity scoring system (CVSS base score[1]) with an F1 score more than doubled.

In Chapter 3, we provide analysis from measures derived from the social network structure of the forum users in the D2web. We particularly show that users who dis-

[1] `https://nvd.nist.gov/vuln-metrics/cvss`

cuss software vulnerabilities in their postings are significantly more active in posting and replying to posts than other individuals. The social network measures are then used as a set of features to predict exploits in the wild. They are compared to the different feature sets presented in Chapter 2 in terms of their impact on F1 score. Two experiments are presented: one tests the performance using individual sets of features, and the other is an ablation test—which checks the decrease on F1 score after removing each set of features. Compared to other feature sets, the set of social network features provides the second-best F1 score, and it resulted in the second-highest F1 score drop when it is eliminated from the features used in the models. The addition of these features improved recall by 19% while maintaining precision.

In Chapter 4, we develop a novel approach used in a system that predicts certain types of cyberattacks targeting specific commercial enterprises. We explain the underlying logic programming framework (APT-logic) used to model the probabilistic temporal relationships between hacker activities (from online hacking communities) and attack incidents (recorded by the SIEM of the data providers). To support ongoing security operations, we relax the constraint of a finite time horizon t_{max} in the original rule-learning approach in [80, 81]. Our approach outperformed a statistical forecasting baseline with an average F1 score value increase of over 45%.

5.2 Future Work

The contribution of this dissertation focuses on addressing principal limitations in the current literature. Yet, it has several limitations that may be addressed in the future. For example:

- **Improving ground truth data.** We show in Chapter 2 that Symantec's data underrepresents vulnerabilities that do not affect products from vendors other than Microsoft and Adobe. To address such limitations, we may con-

sider other sources of cyberthreat intelligence that supply attack signatures of exploits in the wild, such as SecurityFocus[2] and IBM X-Force.[3] The credibility of such sources is yet a concern, i.e., many sources have high false positive rate since they consider proof-of-concept exploits as exploits in the wild. One way to address this challenge is to develop crowdsourcing mechanisms, possibly supporting human-in-the loop capabilities, that are able to reason about the correctness of reported exploits.

- **Revisiting assumptions about hacker post-reply relationships.** Some of the assumptions made to create edges in the hacker social graph approach developed in Chapter 3 may not precisely capture the post-reply relationships between the hackers in the D2web forums. For example, we use a heuristic that is based on an assumption that is adopted in much of the existing literature: a post under a given hacker discussion is a reply to all posts preceding it. We adopt this assumption because the forums we use do not provide information about which post a given reply is directed to. In the future, we may consider revising this assumption by leveraging natural language processing techniques to annotate, from the content of a post, the most related posts. Then, we can generate edges according to the confidence of this annotation. This should affect the centrality and influence measures computed from the social graph, which will possibly improve the prediction performance of the ML models.

- **Human-computer interaction (HCI) limitations.** Across all the pieces of work in this dissertation, the focus has been on developing approaches that: (1) offer specific capabilities to the human analysts (e.g., understand the reasoning

[2] `https://www.securityfocus.com`

[3] `https://www.ibm.com/security/xforce`

that led to certain predictions), and (2) outperform the existing baseline methods and the benchmark approaches, if there is any. We accomplish that by: (1) formally explaining the techniques and providing examples of predictions, and (2) empirically demonstrating the improvement in performance compared to standard baseline methods. However, the usability of the developed approaches to human analysts has not been tested. For example, the ML models developed in Chapter 2 reduced false positives by over 75% compared to the best CVSS-based patching strategy. Yet, we do not know whether this is enough for human analysts to trust these models and use them in production settings. We also do not know what would be the best precision-recall tradeoff that is favorable to the user, i.e., one that improves the patch prioritization process but does not lead to "alert fatigue", where users become desensitized to predictions of exploits in the wild. The same human-computer interaction (HCI) limitations apply to the work developed in Chapter 4. For example, we show how the human analyst can understand why a warning is generated, the corresponding rule that is triggered, the D2web discussions, etc. However, we have not tested the user acceptance to this system after it is deployed. Unlike most cybersecurity AI applications, which are designed to support automated actions, such as spam filtering and intrusion detection, our system is designed to ultimately support human decisions, and its acceptance by the users could be an interesting research direction to pursue in the future.

In addition to addressing the aforementioned limitations, the developed work could be applied to other problems or be extended in ways that could improve the results, such as:

- Considering extra features for the ML models analyzed in Chapter 2 and 3. For example, metadata about the website in which vulnerabilities are discussed such as age, and size of community (or number of users); and metadata related to users such as the number communities to which they are connected, the programming languages they use, and the type of weaknesses they focus on.

- Testing the approach developed in Chapter 4 on other types of rule postconditions, such as postconditions representing attacks observed by a group of organizations in a given industry, or postconditions representing types of attacks that are often conducted using a large pool of resources (e.g., DDoS attacks and 51% cryptocurrency attacks[4]). D2web sites often feature recruiting services for such attacks.

[4] "DDoS" refers to distributed Denial of Service, where typically a group of compromised computers are used to flood the victim's servers with too many requests causing them to go offline. "51% cryptocurrency attacks" refers to a type of cryptocurrency attack that is carried out by a group of hackers controlling more than 50% of the computing power.

REFERENCES

[1] Carl Sabottke, Octavian Suciu, and Tudor Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *USENIX Security*, volume 15, 2015.

[2] Aparna Banerjea. Notpetya: How a russian malware created the world's worst cyberattack ever.

[3] CLIFFORD COLBY. Equifax data breach: How to claim as much as $125 with the ftc settlement.

[4] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707. IEEE, 2002.

[5] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176, 2016.

[6] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336, 2014.

[7] Eric Nunes, Casey Buto, Paulo Shakarian, Christian Lebiere, Stefano Bennati, Robert Thomson, and Holger Jaenisch. Malware task identification: A data driven approach. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*, pages 978–985. IEEE, 2015.

[8] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.

[9] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, 50(3):41, 2017.

[10] Mohamed Alsharnouby, Furkan Alaca, and Sonia Chiasson. Why phishing still works: User strategies for combating phishing attacks. *International Journal of Human-Computer Studies*, 82:69–82, 2015.

[11] B Brij Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharma P Agrawal. Fighting against phishing attacks: state of the art and future challenges. *Neural Computing and Applications*, 28(12):3629–3654, 2017.

[12] Kamaldeep Singh, Sharath Chandra Guntuku, Abhishek Thakur, and Chittaranjan Hota. Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences*, 278:488–497, 2014.

[13] Kamal Alieyan, Ammar ALmomani, Ahmad Manasrah, and Mohammed M Kadhum. A survey of botnet detection based on dns. *Neural Computing and Applications*, 28(7):1541–1558, 2017.

[14] Jesus Mena. *Machine learning forensics for law enforcement, security, and intelligence*. Auerbach Publications, 2016.

[15] E. Nunes, P. Shakarian, G. I. Simari, and A. Ruef. Argumentation models for cyber attribution. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 837–844, Aug 2016.

[16] Ericsson Marin, Ahmad Diab, and Paulo Shakarian. Product offerings in malicious hacker markets. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, pages 187–189. IEEE, 2016.

[17] Hsinchun Chen. *Dark web: Exploring and data mining the dark side of the web*, volume 30. Springer Science & Business Media, 2011.

[18] Eric Nunes, Ahmad Diab, Andrew Gunn, Ericsson Marin, Vineet Mishra, Vivin Paliath, John Robertson, Jana Shakarian, Amanda Thart, and Paulo Shakarian. Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, pages 7–12. IEEE, 2016.

[19] Shalini Ghosh, Ariyam Das, Phil Porras, Vinod Yegneswaran, and Ashish Gehani. Automated categorization of onion sites for analyzing the darkweb ecosystem. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1793–1802. ACM, 2017.

[20] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[21] Xinbo Liu, Yapin Lin, He Li, and Jiliang Zhang. Adversarial examples: Attacks on machine learning-based malware visualization detection methods. *arXiv preprint arXiv:1808.01546*, 2018.

[22] Kyle Soska and Nicolas Christin. Automatically detecting vulnerable websites before they turn malicious. In *Usenix Security*, pages 625–640, 2014.

[23] Mark Felegyhazi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. *LEET*, 10:6–6, 2010.

[24] Luca Melis, Apostolos Pyrgelis, and Emiliano De Cristofaro. On collaborative predictive blacklisting. *ACM SIGCOMM Computer Communication Review*, 48(5):9–20, 2019.

[25] Amirreza Niakanlahiji, Mir Mehedi Pritom, Bei-Tseng Chu, and Ehab Al-Shaer. Predicting zero-day malicious ip addresses. In *Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense*, pages 1–6. ACM, 2017.

[26] Ashok Deb, Kristina Lerman, and Emilio Ferrara. Predicting cyber-events by leveraging hacker sentiment. *Information*, 9(11):280, Nov 2018.

[27] Palash Goyal, KSM Hossain, Ashok Deb, Nazgol Tavabi, Nathan Bartley, Andr'es Abeliuk, Emilio Ferrara, and Kristina Lerman. Discovering signals from web sources to predict cyber attacks. *arXiv preprint arXiv:1806.03342*, 2018.

[28] M. Almukaynizi, E. Marin, E. Nunes, P. Shakarian, G. I. Simari, D. Kapoor, and T. Siedlecki. Darkmention: A deployed system to predict enterprise-targeted external cyberattacks. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 31–36, Nov 2018.

[29] N Tavabi, P Goyal, M Almukaynizi, P Shakarian, and K Lerman. Darkembed: Exploit prediction with neural language models. In *Proceedings of AAAI Conference on Innovative Applications of AI (IAAI2018)*, 2018.

[30] Benjamin L. Bullough, Anna K. Yanchenko, Christopher L. Smith, and Joseph R. Zipkin. Predicting exploitation of disclosed software vulnerabilities using open-source data. In *Proceedings of the 2017 ACM International Workshop on Security and Privacy Analytics*. ACM, 2017.

[31] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. Patch before exploited: An approach to identify targeted software vulnerabilities. In *AI in Cybersecurity*, pages 81–113. Springer, 2019.

[32] Haipeng Chen, Rui Liu, Noseong Park, and VS Subrahmanian. Using twitter to predict when vulnerabilities will be exploited. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3143–3152. ACM, 2019.

[33] Charles P Pfleeger and Shari Lawrence Pfleeger. *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

[34] Stefan Frei, Dominik Schatzmann, Bernhard Plattner, and Brian Trammell. Modeling the security ecosystem-the dynamics of (in) security. In *Economics of Information Security and Privacy*, pages 79–106. Springer, 2010.

[35] Luca Allodi and Fabio Massacci. Comparing vulnerability severity and exploits using case-control studies. *ACM Transactions on Information and System Security (TISSEC)*, 17(1):1, 2014.

[36] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844. ACM, 2012.

[37] Verizon. 2015 verizon data breach investigations report, 2015.

[38] CISCO. Cisco 2016 midyear cybersecurity report, 2016.

[39] Zakir Durumeric, James Kasten, David Adrian, J Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, et al. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 475–488. ACM, 2014.

[40] Michel Edkrantz and Alan Said. Predicting cyber vulnerability exploits with machine learning. In *SCAI*, pages 48–57, 2015.

[41] Kartik Nayak, Daniel Marino, Petros Efstathopoulos, and Tudor Dumitraş. Some vulnerabilities are different than others. In *International Workshop on Recent Advances in Intrusion Detection*, pages 426–446. Springer, 2014.

[42] Luca Allodi and Fabio Massacci. A preliminary analysis of vulnerability scores for attacks in wild: the ekits and sym datasets. In *Proceedings of the 2012 ACM Workshop on Building analysis datasets and gathering experience returns for security*, pages 17–24. ACM, 2012.

[43] Sudip Mittal, Prajit Kumar Das, Varish Mulwad, Anupam Joshi, and Tim Finin. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 860–867. IEEE, 2016.

[44] Sagar Samtani, Kory Chinn, Cathy Larson, and Hsinchun Chen. Azsecure hacker assets portal: Cyber threat intelligence and malware analysis. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, pages 19–24. IEEE, 2016.

[45] Luca Allodi. Economic factors of vulnerability trade and exploitation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1483–1499. ACM, 2017.

[46] Marti Motoyama, Damon McCoy, Kirill Levchenko, Stefan Savage, and Geoffrey M. Voelker. An analysis of underground forums. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 71–80, New York, NY, USA, 2011. ACM.

[47] Thomas J. Holt and Eric Lampke. Exploring stolen data markets online: products and market forces. *Criminal Justice Studies*, 23(1):33–50, 2010.

[48] Jana Shakarian, Andrew T Gunn, and Paulo Shakarian. Exploring malicious hacker forums. In *Cyber Deception*, pages 261–284. Springer, 2016.

[49] John Robertson, Ahmad Diab, Ericsson Marin, Eric Nunes, Vivin Paliath, Jana Shakarian, and Paulo Shakarian. *Darkweb Cyber Threat Intelligence Mining*. Cambridge University Press, 2017.

[50] Yang Liu, Armin Sarabi, Jing Zhang, Parinaz Naghizadeh, Manish Karir, Michael Bailey, and Mingyan Liu. Cloudy with a chance of breach: Forecasting cyber security incidents. In *Usenix Security*.

[51] Mohammed Almukaynizi, Eric Nunes, Krishna Dharaiya, Manoj Senguttuvan, Jana Shakarian, and Paulo Shakarian. Proactive identification of exploits in the wild through vulnerability mentions online. In *Cyber Conflict (CyCon US), 2017 International Conference on*, pages 82–88. IEEE, 2017.

[52] Mehran Bozorgi, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–114. ACM, 2010.

[53] Su Zhang, Doina Caragea, and Xinming Ou. An empirical study on using the national vulnerability database to predict software vulnerabilities. In *International Conference on Database and Expert Systems Applications*, pages 217–231. Springer, 2011.

[54] Shuang Hao, Alex Kantchelian, Brad Miller, Vern Paxson, and Nick Feamster. Predator: Proactive recognition and elimination of domain abuse at time-of-registration. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1568–1579. ACM, 2016.

[55] Corinna Cortes and Vladimir Vapnik. Support-vector networks. pages 273–297, 1995.

[56] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[57] Luca Allodi, Fabio Massacci, and Julian M Williams. The work-averse cyber attacker model: Theory and evidence from two million attack signatures. *Available at SSRN 2862299*, 2017.

[58] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[59] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[60] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[61] Dongning Guo, Shlomo Shamai, and Sergio Verdú. Mutual information and minimum mean-square error in gaussian channels. *IEEE Transactions on Information Theory*, 51(4):1261–1282, 2005.

[62] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, July 2012.

[63] Marco Barreno, Peter L Bartlett, Fuching Jack Chi, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, Udam Saini, and J Doug Tygar. Open problems in the security of learning. In *Proceedings of the 1st ACM workshop on Workshop on AISec*, pages 19–26. ACM, 2008.

[64] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[65] Battista Biggio, Blaine Nelson, and Pavel Laskov. Support vector machines under adversarial label noise. *ACML*, 20:97–112, 2011.

[66] Elizabeth Phillips, Jason RC Nurse, Michael Goldsmith, and Sadie Creese. Extracting social structure from darkweb forums. 2015.

[67] Reza Zafarani, Mohammad Ali Abbasi, and Huan Liu. *Social media mining: an introduction*. Cambridge University Press, 2014.

[68] Mariam Nouh and Jason RC Nurse. Identifying key-players in online activist groups on the facebook social network. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 969–978. IEEE, 2015.

[69] Alex Beutel, Leman Akoglu, and Christos Faloutsos. Fraud detection through graph-based user behavior modeling. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1696–1697. ACM, 2015.

[70] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro. Aiding the detection of fake accounts in large scale social online services. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 15–15. USENIX Association, 2012.

[71] Thomas J Holt, Deborah Strumsky, Olga Smirnova, and Max Kilger. Examining the social networks of malware writers and hackers. *International Journal of Cyber Criminology*, 6(1):891, 2012.

[72] Victor Benjamin, Weifeng Li, Thomas Holt, and Hsinchun Chen. Exploring threats and vulnerabilities in hacker web: Forums, irc and carding shops. In *Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on*, pages 85–90. IEEE, 2015.

[73] Sagar Samtani, Ryan Chinn, and Hsinchun Chen. Exploring hacker assets in underground forums. In *Intelligence and Security Informatics (ISI), 2015 IEEE International Conference on*, pages 31–36. IEEE, 2015.

[74] IdentityForce. Data breaches - the worst breaches, so far, Last Accessed: June 2019.

[75] IdentityForce. Data breaches - the worst breaches, so far, Last Accessed: June 2019.

[76] GOV.UK. 2019 cyber security breaches survey., 2019.

[77] Symantec. 2019 internet security threat report, Last Accessed: June 2019.

[78] Verizon. 2017 data breach investigations report, 2017.

[79] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM.

[80] Paulo Shakarian, Austin Parker, Gerardo Simari, and Venkatramana V. S. Subrahmanian. Annotated probabilistic temporal logic. *ACM Trans. Comput. Logic*, 12(2):14:1–14:44, January 2011.

[81] Paulo Shakarian, Gerardo I. Simari, and V. S. Subrahmanian. Annotated probabilistic temporal logic: Approximate fixpoint implementation. *ACM Trans. Comput. Logic*, 13(2):13:1–13:33, April 2012.

[82] Andrew Stanton, Amanda Thart, Ashish Jain, Priyank Vyas, Arpan Chatterjee, and Paulo Shakarian. Mining for causal relationships: A data-driven study of the islamic state. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2137–2146. ACM, 2015.

[83] P. Nespoli, D. Papamartzivanos, F. G. Mrmol, and G. Kambourakis. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys Tutorials*, 20(2):1361–1396, Secondquarter 2018.

[84] Arpan Roy, Dong Seong Kim, and Kishor S Trivedi. Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pages 1–12. IEEE, 2012.

[85] C. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang. Nice: Network intrusion detection and countermeasure selection in virtual network systems. *IEEE Transactions on Dependable and Secure Computing*, 10(4):198–211, July 2013.

[86] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, and Y. Xiang. Data-driven cybersecurity incident prediction: A survey. *IEEE Communications Surveys Tutorials*, 21(2):1744–1772, Secondquarter 2019.

[87] Anna Sapienza, Sindhu Kiranmai Ernala, Alessandro Bessi, Kristina Lerman, and Emilio Ferrara. Discover: Mining online chatter for emerging cyber threats. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 983–990, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[88] Soumajyoti Sarkar, Mohammad Almukaynizi, Jana Shakarian, and Paulo Shakarian. Predicting enterprise cyber incidents using social network analysis on the darkweb hacker forums. *CoRR*, abs/1811.06537, 2018.

[89] Leslie F. Sikos, Dean Philp, Catherine Howard, Shaun Voigt, Markus Stumptner, and Wolfgang Mayer. *Knowledge Representation of Network Semantics for Reasoning-Powered Cyber-Situational Awareness*, pages 19–45. Springer International Publishing, Cham, 2019.

[90] Matt Turek. Explainable artificial intelligence (XAI)., Last Accessed: June 2019.

[91] Alejandro Javier García and Guillermo Ricardo Simari. Defeasible logic programming: An argumentative approach. *Theory Pract. Log. Program.*, 4(2):95–138, January 2004.

[92] E. Marin, M. Almukaynizi, E. Nunes, J. Shakarian, and P. Shakarian. Predicting hacker adoption on darkweb forums using sequential rule mining. In *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1183–1190, Dec 2018.

[93] Philippe Fournier-Viger, Cheng-Wei Wu, Vincent S. Tseng, and Roger Nkambou. Mining sequential rules common to several sequences with the window size constraint. In Leila Kosseim and Diana Inkpen, editors, *Advances in Artificial Intelligence*, pages 299–304, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[94] CISA. Hidden cobra north koreas ddos botnet infrastructure., 2017.

[95] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[96] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, Apr 1996.

[97] Douglas C Montgomery. *Introduction to statistical quality control*. John Wiley & Sons, 2007.

[98] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.