Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits

Carl Sabottke, Octavian Suciu, and Tudor Dumitraş, ***University of Maryland***

https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sabottke

This paper is included in the Proceedings of the 24th USENIX Security Symposium

August 12–14, 2015 • Washington, D.C.

ISBN 978-1-939133-11-3

Open access to the Proceedings of the 24th USENIX Security Symposium is sponsored by USENIX

Vulnerability Disclosure in the Age of Social Media: Exploiting Twitter for Predicting Real-World Exploits

Carl Sabottke Octavian Suciu Tudor Dumitraș
University of Maryland

Abstract

In recent years, the number of software vulnerabilities discovered has grown significantly. This creates a need for prioritizing the response to new disclosures by assessing which vulnerabilities are likely to be exploited and by quickly ruling out the vulnerabilities that are not actually exploited in the real world. We conduct a quantitative and qualitative exploration of the vulnerability-related information disseminated on Twitter. We then describe the design of a Twitter-based exploit detector, and we introduce a threat model specific to our problem. In addition to response prioritization, our detection techniques have applications in risk modeling for cyber-insurance and they highlight the value of information provided by the victims of attacks.

1 Introduction

The number of software vulnerabilities discovered has grown significantly in recent years. For example, 2014 marked the first appearance of a 5 digit CVE, as the CVE database [46], which assigns unique identifiers to vulnerabilities, has adopted a new format that no longer caps the number of CVE IDs at 10,000 per year. Additionally, many vulnerabilities are made public through a *coordinated disclosure process* [18], which specifies a period when information about the vulnerability is kept confidential to allow vendors to create a patch. However, this process results in multi-vendor disclosure schedules that sometimes align, causing a flood of disclosures. For example, 254 vulnerabilities were disclosed on 14 October 2014 across a wide range of vendors including Microsoft, Adobe, and Oracle [16].

To cope with the growing rate of vulnerability discovery, the security community must prioritize the effort to respond to new disclosures by assessing the risk that the vulnerabilities will be exploited. The existing scoring systems that are recommended for this purpose, such as FIRST's Common Vulnerability Scoring System (CVSS) [54], Microsoft's exploitability index [21] and Adobe's priority ratings [19], err on the side of caution by marking many vulnerabilities as likely to be exploited [24]. The situation in the real world is more nuanced. While the disclosure process often produces *proof of concept* exploits, which are publicly available, recent empirical studies reported that only a small fraction of vulnerabilities are exploited *in the real world*, and this fraction has decreased over time [22,47]. At the same time, some vulnerabilities attract significant attention and are quickly exploited; for example, exploits for the Heartbleed bug in OpenSSL were detected 21 hours after the vulnerability's public disclosure [41]. To provide an adequate response on such a short time frame, the security community must quickly determine which vulnerabilities are exploited in the real world, while minimizing false positive detections.

The security vendors, system administrators, and hackers, who discuss vulnerabilities on social media sites like Twitter, constitute rich sources of information, as the participants in coordinated disclosures discuss technical details about exploits and the victims of attacks share their experiences. This paper explores the opportunities for *early exploit detection* using information available on Twitter. We characterize the exploit-related discourse on Twitter, the information posted before vulnerability disclosures, and the users who post this information. We also reexamine a prior experiment on predicting the development of proof-of-concept exploits [36] and find a considerable performance gap. This illuminates the threat landscape evolution over the past decade and the current challenges for early exploit detection.

Building on these insights, we describe techniques for detecting exploits that are active in the real world. Our techniques utilize *supervised machine learning* and ground truth about exploits from ExploitDB [3], OSVDB [9], Microsoft security advisories [21] and the descriptions of Symantec's anti-virus and intrusion-protection signatures [23]. We collect an unsampled cor-

pus of tweets that contain the keyword "CVE," posted between February 2014 and January 2015, and we extract features for training and testing a support vector machine (SVM) classifier. We evaluate the false positive and false negative rates and we assess the detection lead time compared to existing data sets. Because Twitter is an open and free service, we introduce a *threat model*, considering realistic adversaries that can poison both the training and the testing data sets but that may be resource-bound, and we conduct simulations to evaluate the resilience of our detector to such attacks. Finally, we discuss the implications of our results for building security systems without secrets, the applications of early exploit detection and the value of sharing information about successful attacks.

In summary, we make three contributions:

- We characterize the landscape of threats related to information leaks about vulnerabilities before their public disclosure, and we identify features that can be extracted automatically from the Twitter discourse to detect exploits.
- To our knowledge, we describe the first technique for early detection of real-world exploits using social media.
- We introduce a threat model specific to our problem and we evaluate the robustness of our detector to adversarial interference.

Roadmap. In Sections 2 and 3 we formulate the problem of exploit detection and we describe the design of our detector, respectively. Section 4 provides an empirical analysis of the exploit-related information disseminated on Twitter, Section 5 presents our detection results, and Section 6 evaluates attacks against our exploit detectors. Section 7 reviews the related work, and Section 8 discusses the implications of our results.

2 The problem of exploit detection

We consider a *vulnerability* to be a software bug that has security implications and that has been assigned a unique identifier in the CVE database [46]. An *exploit* is a piece of code that can be used by an attacker to subvert the functionality of the vulnerable software. While many researchers have investigated the techniques for creating exploits, the utilization patterns of these exploits provide another interesting dimension to their security implications. We consider *real-world exploits* to be the exploits that are being used in real attacks against hosts and networks worldwide. In contrast, *proof-of-concept (PoC) exploits* are often developed as part of the vulnerability disclosure process and are included in penetration testing suites. We further distinguish between *public PoC exploits*, for which the exploit code is publicly available, and *private PoC exploits*, for which we can find reliable information that the exploit was developed, but it was not released to the public. A PoC exploit may also be a real-world exploit if it is used in attacks.

The existence of a real-world or PoC exploit gives urgency to fixing the corresponding vulnerability, and this knowledge can be utilized for prioritizing remediation actions. We investigate the opportunities for *early detection* of such exploits by using information that is available publicly, but is not included in existing vulnerability databases such as the National Vulnerability Database (NVD) [7] or the Open Sourced Vulnerability Database (OSVDB) [9]. Specifically, we analyze the Twitter stream, which exemplifies the information available from social media feeds. On Twitter, a community of hackers, security vendors and system administrators discuss security vulnerabilities. In some cases, the victims of attacks report new vulnerability exploits. In other cases, information leaks from the *coordinated disclosure* process [18] through which the security community prepares the response to the impending public disclosure of a vulnerability.

The vulnerability-related discourse on Twitter is influenced by trend-setting vulnerabilities, such as Heartbleed (CVE-2014-0160), Shellshock (CVE-2014-6271, CVE-2014-7169, and CVE-2014-6277) or Drupalgeddon (CVE-2014-3704) [41]. Such vulnerabilities are mentioned by many users who otherwise do not provide actionable information on exploits, which introduces a significant amount of noise in the information retrieved from the Twitter stream. Additionally, adversaries may inject fake information into the Twitter stream, in an attempt to poison our detector. Our *goals* in this paper are (i) to identify the good sources of information about exploits and (ii) to assess the opportunities for early detection of exploits in the presence of benign and adversarial noise. Specifically, we investigate techniques for minimizing *false-positive detections*—vulnerabilities that are not actually exploited—which is critical for prioritizing response actions.

Non-goals. We do not consider the detection of *zero-day attacks* [32], which exploit vulnerabilities before their public disclosure; instead, we focus on detecting the use of exploits against known vulnerabilities. Because our aim is to assess the value of publicly available information for exploit detection, we do not evaluate the benefits of incorporating commercial or private data feeds. The design of a complete system for early exploit detection, which likely requires mechanisms beyond the realm of Twitter analytics (e.g., for managing the reputation of data sources to prevent poisoning attacks), is also out of scope for this paper.

2.1 Challenges

To put our contributions in context, we review the three primary challenges for predicting exploits in the absence of adversarial interference: class imbalance, data scarcity, and ground truth biases.

Class imbalance. We aim to train a classifier that produces binary predictions: each vulnerability is classified as either exploited or not exploited. If there are significantly more vulnerabilities in one class than in the other class, this biases the output of supervised machine learning algorithms. Prior research on predicting the existence of proof-of-concept exploits suggests that this bias is not large, as over half of the vulnerabilities disclosed before 2007 had such exploits [36]. However, few vulnerabilities are exploited in the real world and the exploitation ratios tend to decrease over time [47]. In consequence, our data set exhibits a severe class imbalance: we were able to find evidence of real-world exploitation for only 1.3% of vulnerabilities disclosed during our observation period. This class imbalance represents a significant challenge for simultaneously reducing the false positive and false negative detections.

Data scarcity. Prior research efforts on Twitter analytics have been able to extract information from millions of tweets, by focusing on popular topics like movies [27], flu outbreaks [20, 26], or large-scale threats like spam [56]. In contrast, only a small subset of Twitter users discuss vulnerability exploits (approximately 32,000 users), and they do not always mention the CVE numbers in their tweets, which prevents us from identifying the vulnerability discussed. In consequence, 90% of the CVE numbers disclosed during our observation period appear in fewer than 50 tweets. Worse, when considering the known real-world exploits, close to half have fewer than 50 associated tweets. This data scarcity compounds the challenge of class imbalance for reducing false positives and false negatives.

Quality of ground truth. Prior work on Twitter analytics focused on predicting quantities for which good predictors are already available (modulo a time lag): the Hollywood Stock Exchange for movie box-office revenues [27], CDC reports for flu trends [45] and Twitter's internal detectors for highjacked accounts, which trigger account suspensions [56]. These predictors can be used as ground truth for training high-performance classifiers. In contrast, there is no comprehensive data set of vulnerabilities that are exploited in the real world. We employ as ground truth the set of vulnerabilities mentioned in the descriptions of Symantec's anti-virus and intrusion-protection signatures, which is, reportedly, the best available indicator for the exploits included in exploit kits [23, 47]. However, this dataset has coverage biases, since Symantec does not cover all platforms and products uniformly. For example, since Symantec does not provide a security product for Linux, Linux kernel vulnerabilities are less likely to appear in our ground truth dataset than exploits targeting software that runs on the Windows platform.

2.2 Threat model

Research in adversarial machine learning [28, 29], distinguishes between exploratory attacks, which poison the testing data, and causative attacks, which poison both the testing and the training data sets. Because Twitter is an open and free service, causative adversaries are a realistic threat to a system that accepts inputs from all Twitter users. We assume that these adversaries cannot prevent the victims of attacks from tweeting about their observations, but they can inject additional tweets in order to compromise the performance of our classifier. To test the ramifications of these causative attacks, we develop a threat model with three types of adversaries.

Blabbering adversary. Our weakest adversary is not aware of the statistical properties of the training features or labels. This adversary simply sends tweets with random CVEs and random security-related keywords.

Word copycat adversary. A stronger adversary is aware of the features we use for training and has access to our ground truth (which comes from public sources). This adversary uses fraudulent accounts to manipulate the word features and total tweet counts in the training data. However, this adversary is resource constrained and cannot manipulate any user statistics which would require either more expensive or time intensive account acquisition and setup (e.g., creation date, verification, follower and friend counts). The copycat adversary crafts tweets by randomly selecting pairs of non-exploited and exploited vulnerabilities and then sending tweets, so that the word feature distributions between these two classes become nearly identical.

Full copycat adversary. Our strongest adversary has full knowledge of our feature set. Additionally, this adversary has sufficient time and economic resources to purchase or create Twitter accounts with arbitrary user statistics, with the exception of verification and the account creation date. Therefore, the full copycat adversary can use a set of fraudulent Twitter accounts to fully manipulate almost all word and user-based features, which creates scenarios where relatively benign CVEs and real-world exploit CVEs appear to have nearly identical Twitter traffic at an abstracted statistical level.

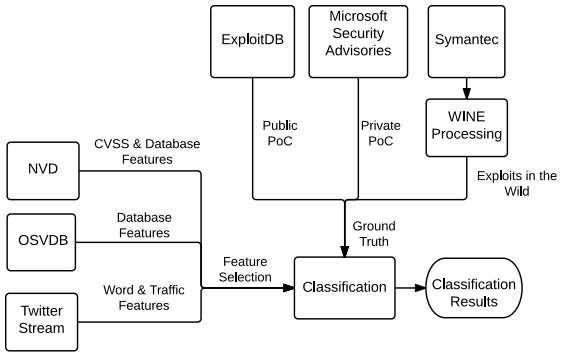


Figure 1: Overview of the system architecture.

3 A Twitter-based exploit detector

We present the design of a Twitter-based exploit detector, using supervised machine learning techniques. Our detector extracts vulnerability-related information from the Twitter stream, and augments it with additional sources of data about vulnerabilities and exploits.

3.1 Data collection

Figure 1 illustrates the architecture of our exploit detector. Twitter is an online social networking service that enables users to send and read short 140-character messages called "tweets", which then become publicly available. For collecting tweets mentioning vulnerabilities, the system monitors occurrences of the "CVE" keyword using Twitter's Streaming API [15]. The policy of the Streaming API implies that a client receives all the tweets matching a keyword as long as the result does not exceed 1% of the entire Twitter hose, when the tweets become samples of the entire matching volume. Because the CVE tweeting volume is not high enough to reach 1% of the hose (as the API signals rate limiting), we conclude that our collection contains all references to CVEs, except during the periods of downtime for our infrastructure.

We collect data over a period of one year, from February 2014 to January 2015. Out of the 1.1 billion tweets collected during this period, 287,717 contain explicit references to CVE IDs. We identify 7,560 distinct CVEs. After filtering out the vulnerabilities disclosed before the start of our observation period, for which we have missed many tweets, we are left with 5,865 CVEs.

To obtain context about the vulnerabilities discussed on Twitter, we query the National Vulnerability Database (NVD) [7] for the CVSS scores, the products affected and additional references about these vulnerabilities. Additionally, we crawl the Open Sourced Vulnerability Database (OSVDB) [9] for a few additional attributes, including the disclosure dates and categories of the vulnerabilities in our study.[1] Our data collection infrastructure consists of Python scripts, and the data is stored using Hadoop Distributed File System. From the raw data collected, we extract multiple features using Apache PIG and Spark, which run on top of a local Hadoop cluster.

Ground truth. We use three sources of ground truth. We identify the set of vulnerabilities *exploited in the real world* by extracting the CVE IDs mentioned in the descriptions of Symantec's anti-virus (AV) signatures [12] and intrusion-protection (IPS) signatures [13]. Prior work has suggested that this approach produces the best available indicator for the vulnerabilities targeted in exploits kits available on the black market [23, 47]. Considering only the vulnerabilities included in our study, this data set contains 77 vulnerabilities targeting products from 31 different vendors. We extract the creation date from the descriptions of AV signatures to estimate the date when the exploits were discovered. Unfortunately, the IPS signatures do not provide this information, so we query Symantec's Worldwide Intelligence Network Environment (WINE) [40] for the dates when these signatures were triggered in the wild. For each real-world exploit, we use the earliest date across these data sources as an estimate for the date when the exploit became known to the security community.

However, as mentioned in Section 2.1, this ground truth does not cover all platforms and products uniformly. Nevertheless, we expect that some software vendors, which have well established procedures for coordinated disclosure, systematically notify security companies of impending vulnerability disclosures to allow them to release detection signatures on the date of disclosure. For example, the members of Microsoft's MAPP program [5] receive vulnerability information in advance of the monthly publication of security advisories. This practice provides defense-in-depth, as system administrators can react to vulnerability disclosures either by deploying the software patches or by updating their AV or IPS signatures. To identify which products are well covered in this data set, we group the exploits by the vendor of the affected product. Out of the 77 real-world exploits, 41 (53%) target products from Microsoft and Adobe, while no other vendor accounts for more than 3% of exploits. This suggests that our ground truth provides the best coverage for vulnerabilities in Microsoft and Adobe products.

We identify the set of vulnerabilities with *public proof-of-concept exploits* by querying ExploitDB [3], a collaborative project that collects vulnerability exploits. We

[1] In the past, OSVDB was called the Open Source Vulnerability Database and released full dumps of their database. Since 2012, OSVDB no longer provides public dumps and actively blocks attempts to crawl the website for most of the information in the database.

identify exploits for 387 vulnerabilities disclosed during our observation period. We use the date when the exploits were added to ExploitDB as an indicator for when the vulnerabilities were exploited.

We also identify the set of vulnerabilities in Microsoft's products for which *private proof-of-concept* exploits have been developed by using the Exploitability Index [21] included in Microsoft security advisories. This index ranges from 0 to 3: 0 for vulnerabilities that are known to be exploited in the real world at the time of release for a security bulletin,[2] and 1 for vulnerabilities that allowed the development of exploits with consistent behavior. Vulnerabilities with scores of 2 and 3 are considered less likely and unlikely to be exploited, respectively. We therefore consider that the vulnerabilities with an exploitability index of 0 or 1 have an private PoC exploit, and we identify 218 such vulnerabilities. 22 of these 218 vulnerabilities are considered real-world exploits in our Symantec ground truth.

3.2 Vulnerability categories

To quantify how these vulnerabilities and exploits are discussed on Twitter, we group them into 7 categories, based on their utility for an attacker: Code Execution, Information Disclosure, Denial of Service, Protection Bypass, Script Injection, Session Hijacking and Spoofing. Although heterogeneous and unstructured, the summary field from NVD entries provides sufficient information for assigning a category to most of the vulnerabilities in the study, using regular expressions comprised of domain vocabulary.

Table 2 and Section 4 show how these categories intersect with POC and real-world exploits. Since vulnerabilities may belong to several categories (a code execution exploit could also be used in a denial of service), the regular expressions are applied in order. If a match is found for one category, the subsequent categories would not be matched.

Aditionally, the Unknown category contains vulnerabilities not matched by the regular expressions and those whose summaries explicitly state that the consequences are unknown or unspecified.

3.3 Classifier feature selection

The features considered in this study can be classified in 4 categories: Twitter Text, Twitter Statistics, CVSS Information and Database Information.

For the Twitter features, we started with a set of 1000 keywords and 12 additional features based on the distribution of tweets for the CVEs, e.g. the total number of tweets related to the CVE, the average age of the accounts posting about the vulnerability and the number of retweets associated to the vulnerability. For each of these initial features, we compute the mutual information (MI) of the set of feature values X and the class labels $Y \in \{\text{exploited}, \text{not exploited}\}$:

$$MI(Y,X) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \ln \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Mutual information, expressed in nats, compares the frequencies of values from the joint distribution $p(x,y)$ (i.e. values from X and Y that occur together) with the product of the frequencies from the two distributions $p(x)$ and $p(y)$. MI measures how much knowing X reduces uncertainty about Y, and can single out useful features suggesting that the vulnerability is exploited as well as features suggesting it is not. We prune the initial feature set by excluding all features with mutual information below 0.0001 nats. For numerical features, we estimate probability distributions using a resolution of 50 bins per feature. After this feature selection process, we are left with 38 word features for real-world exploits.

Here, rather than use a wrapper method for feature selection, we use this mutual information-based filter method in order to facilitate the combination of automatic feature selection with intuition-driven manual pruning. For example, keywords that correspond to trend-setting vulnerabilities from 2014, like Heartbleed and Shellshock, exhibit a higher mutual information than many other potential keywords despite their relation to only a small subset of vulnerabilities. Yet, such highly

Keyword	MI Wild	MI PoC	Keyword	MI Wild	MI PoC
advisory	0.0007	0.0005	ok	0.0015	0.0002
beware	0.0007	0.0005	mcafee	0.0005	0.0002
sample	0.0007	0.0005	windows	0.0012	0.0011
exploit	0.0026	0.0016	w	0.0004	0.0002
go	0.0007	0.0005	microsoft	0.0007	0.0005
xp	0.0007	0.0005	info	0.0007	X
ie	0.0015	0.0005	rce	0.0007	X
poc	0.0004	0.0006	patch	0.0007	X
web	0.0015	0.0005	piyolog	0.0007	X
java	0.0007	0.0005	tested	0.0007	X
working	0.0007	0.0005	and	X	0.0005
fix	0.0012	0.0002	rt	X	0.0005
bug	0.0007	0.0005	eset	X	0.0005
blog	0.0007	0.0005	for	X	0.0005
pc	0.0007	0.0005	redhat	X	0.0002
reading	0.0007	0.0005	kali	X	0.0005
iis	0.0007	0.0005	0day	X	0.0009
ssl	0.0005	0.0003	vs	X	0.0005
post	0.0007	0.0005	linux	X	0.0009
day	0.0015	0.0005	new	X	0.0002
bash	0.0015	0.0009			

Table 1: Mutual information provided by the reduced set of keywords with respect to both sources of ground truth data. The "X" marks in the table indicate that the respective words were excluded from the final feature set due to MI below 0.0001 nats.

[2] We do not use this score as an indicator for the existence of real-world exploits because the 0 rating is available only since August 2014, toward the end of our observation period.

specific keywords are undesirable for classification due to their transitory utility. Therefore, in order to reduce susceptibility to concept drift, we manually prune these word features for our classifiers to generate a final set of 31 out of 38 word features (listed in Table 1 with additional keyword intuition described in Section 4.1).

In order to improve performance and increase classifier robustness to potential Twitter-based adversaries acting through account hijacking and Sybil attacks, we also derive features from NVD and OSVDB. We consider all 7 CVSS score components, as well as features that proved useful for predicting proof-of-concept exploits in prior work [36], such as the number of unique references, the presence of the token BUGTRAQ in the NVD references, the vulnerability category from OSVDB and our own vulnerability categories (see Section 3.2). This gives us 17 additional features. The inclusion of these non-Twitter features is useful for boosting the classifier's resilience to adversarial noise. Figure 2 illustrates the most useful features for detecting real-world or PoC exploits, along with the corresponding mutual information.

3.4 Classifier training and evaluation

We train linear support vector machine (SVM) classifiers [35, 38, 39, 43] in a feature space with 67 dimensions that results from our feature selection step (Section 3.3). SVMs seek to determine the maximum margin hyperplane to separate the classes of exploited and non-exploited vulnerabilities. When a hyperplane cannot perfectly separate the positive and negative class samples based on the feature vectors used in training, the basic SVM cost function is modified to include a regularization penalty, C, and non-negative slack variables, ξ_i. By varying C, we explore the trade-off between false negatives and false positives in our classifiers.

We train SVM classifiers using multiple rounds of stratified random sampling. We perform sampling because of the large imbalance in the class sizes between vulnerabilities exploited and vulnerabilities not exploited. Typically, our classifier training consists of 10 random training shuffles where 50% of the available data is used for training and the remaining 50% is used for testing. We use the `scikit-learn` Python package [49] to train our classifiers.

An important caveat, though, is that our one year of data limits our ability to evaluate concept drift. In most cases, our cross-validation data is temporally intermixed with the training data, since restricting sets of training and testing CVEs to temporally adjacent blocks confounds performance losses due to concept drift with performance losses due to small sample sizes. Furthermore, performance differences between the vulnerability database features of our classifiers and those explored in [36] emphasize the benefit of periodically repeating previous experiments from the security literature in order to properly assess whether the results are subject to long-term concept drift.

Category	# CVEs	Real-World	PoC	Both
	All Data / Good Coverage			
Code Execution	1249/322	66/39	192/14	28/8
Info Disclosure	1918/59	4/0	69/5	4/0
Denial Of Service	657/17	0/0	16/1	0/0
Protection Bypass	204/34	0/0	3/0	0/0
Script Injection	683/14	0/0	40/0	0/0
Session Hijacking	167/1	0/0	25/0	0/0
Spoofing	55/4	0/0	0/0	0/0
Unknown	981/51	7/0	42/6	5/0
Total	5914/502	77/39	387/26	37/8

Table 2: CVEs Categories and exploits summary. The first sub-column represents the whole dataset, while the second sub-column is restricted to Adobe and Microsoft vulnerabilities, for which our ground truth of real-world exploits provides good coverage.

Performance metrics. When evaluating our classifiers, we rely on two standard performance metrics: *precision* and *recall*.[3] Recall is equivalent to the true positive rate: Recall $= \frac{TP}{TP+FN}$, where TP is the number of true positive classifications and FN is the number of false negatives. The denominator is the total number of positive samples in the testing data. Precision is defined as: Precision $= \frac{TP}{TP+FP}$ where FP is the total number of false positives identified by the classifier. When optimizing classifier performance based on these criteria, the relative importance of these quantities is dependent on the intended applications of the classifier. If avoiding false negatives is priority, then recall must be high. However, if avoiding false positives is more critical, then precision is the more important metric. Because we envision utilizing our classifier as a tool for prioritizing the response to vulnerability disclosures, we focus on improving the precision rather than the recall.

4 Exploit-related information on Twitter

Table 2 breaks down the vulnerabilities in our study according to the categories described in Section 3.2. 1249 vulnerabilities allowing code execution were disclosed during our observation period. 66 have real-world exploits, and 192 have public proof-of-concept exploits; the intersection of these two sets includes 28 exploits. If we consider only Microsoft and Adobe vulnerabilities, for which we expect that our ground truth has good coverage (see Section 3.1), the table shows that 322 code-

[3] We choose precision and recall because Receiver Operating Characteristic (ROC) curves can present an overly optimistic view of a classifier's performance when dealing with skewed data sets [**?**].

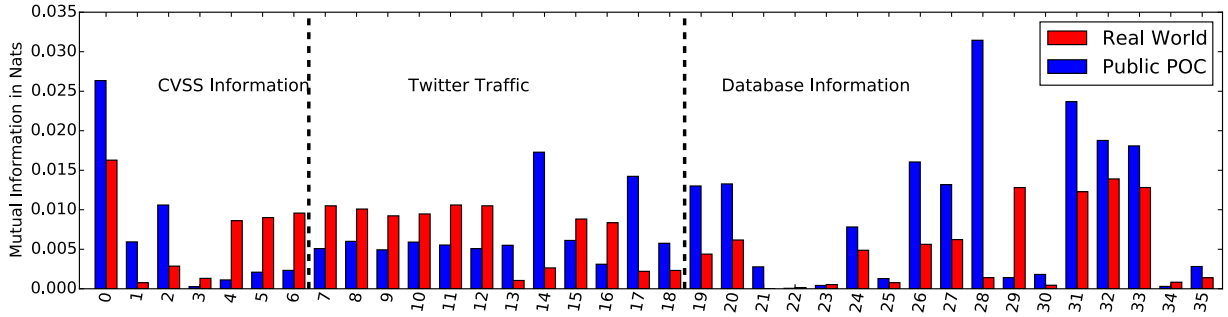


Figure 2: Mutual information between real world and public proof of concept exploits for CVSS, Twitter user statistics, NVD, and OSVDB features. **CVSS Information** - 0: CVSS Score, 1: Access Complexity, 2: Access Vector, 3: Authentication, 4: Availability Impact, 5: Confidentiality Impact, 6: Integrity Impact. **Twitter Traffic** - 7: Number of tweets, 8/9: # users with minimum T followers/friends, 10/11: # retweets/replies, 12: # tweets favorited, 13/14/15: Avg # hashtags/URLs/user mentions per tweet, 16: # verified accounts, 17: Avg age of accounts, 18: Avg # of tweets per account. **Database Information** - 19: # references in NVD, 20: # sources in NVD, 21/22: BUGTRAQ/SECUNIA in NVD sources, 23: allow in NVD summay, 24: NVD last modified date - NVD published date, 25: NVD last modified date - OSVDB disclosed date, 26: Number of tokens in OSVDB title, 27: Current date - NVD last modified date, 28: OSVDB in NVD sources, 29: code in NVD summay, 30: # OSVDB entries, 31: OSVDB Category, 32: Regex Category, 33: First vendor in NVD, 34: # vendors in NVD, 35: # affected products in NVD.

execution vulnerabilities were disclosed, 39 have real-world exploits, 14 have public PoC exploits and 8 have both real-world and public PoC exploits.

Information disclosure is the largest category of vulnerabilities from NVD and it has a large number of PoC exploits, but we find few of these vulnerabilities in our ground truth of real-world exploits (one exception is Heartbleed). Instead, most of the real-world exploits focus on *code execution* vulnerabilities. However, many proof-of-concept exploits for such vulnerabilities do not seem to be utilized in real-world attacks. To understand the factors that drive the differences between real-world and proof-of-concept exploits, we examine the CVSS base metrics, which describe the characteristics of each vulnerability. This analysis reveals that most of the real-world exploits allow *remote* code execution, while some PoC exploits require local host or local network access. Moreover, while some PoC vulnerabilities require authentication before a successful exploit, real-world exploits focus on vulnerabilities that do not require *bypassing authentication* mechanisms. In fact, this is the only type of exploit we found in the segment of our ground truth that has good coverage, suggesting that remote code-execution exploits with no authentication required are strongly favored by real-world attackers. Our ground truth does not provide good coverage of web exploits, which explains the lack of Script Injection, Session Highjacking and Spoofing exploits from our real-world data set.

Surprisingly, we find that, among the remote execution vulnerabilities for which our ground truth provides good coverage, there are *more real-world exploits than public PoC exploits*. This could be explained by the increasing prevalence of obfuscated disclosures, as reflected in NVD vulnerability summaries that mention the possibility of exploitation "via unspecified vectors" (for example, CVE-2014-8439). Such disclosures make it more difficult to create PoC exploits, as the technical information required is not readily available, but they may not thwart determined attackers who have gained experience in hacking the product in question.

4.1 Exploit-related discourse on Twitter

The Twitter discourse is dominated by a few vulnerabilities. Heartbleed (CVE-2014-0160) received the highest attention, with more than 25,000 tweets (8,000 posted in the first day after disclosure). 24 vulnerabilities received more than 1,000 tweets. 16 of these vulnerabilities were exploited: 11 in the real-world, 12 in public proofs of concept and 8 in private proofs of concept. The median number of tweets across all the vulnerabilities in our data set is 14.

The terms that Twitter users employ when discussing exploits also provide interesting insights. Surprisingly, the distribution of the keyword "0day" exhibits a high mutual information with public proof-of-concept exploits, but not with real-world exploits. This could be explained by confusion over the definition of the term *zero-day vulnerability*: many Twitter users understand this to mean simply a new vulnerability, rather than a vulnerability that was exploited in real-world attacks before its public disclosure [32]. Conversely, the distribution of the keyword "patch" has high mutual information only

with the real-world exploits, because a common reason for posting tweets about vulnerabilities is to alert other users and point them to advisories for updating vulnerable software versions after exploits are detected in the wild. Certain words like "exploit" or "advisory" are useful for detecting both real-world and PoC exploits.

4.2 Information posted before disclosure

For the vulnerabilities in our data set, Figure 3 compares the dates of the earliest tweets mentioning the corresponding CVE numbers with the public disclosure dates for these vulnerabilities. Using the disclosure date recorded in OSVDB, we identify 47 vulnerabilities that were mentioned in the Twitter stream before their public disclosure. We investigate these cases manually to determine the sources of this information. 11 of these cases represent misspelled CVE IDs (e.g. users mentioning 6172 but talking about 6271 – Shellshock), and we are unable to determine the root cause for 5 additional cases owing to the lack of sufficient information. The remaining cases can be classified into 3 general categories of information leaks:

Disagreements about the planned disclosure date. The vendor of the vulnerable software sometimes posts links to a security bulletin ahead of the public disclosure date. These cases are typically benign, as the security advisories provide instructions for patching the vulnerability. A more dangerous situation occurs when the party who discovers the vulnerability and the vendor disagree about the disclosure schedule, resulting in the publication of vulnerability details a few days before a patch is made available [6, 14, 16, 17]. We have found 13 cases of disagreements about the disclosure date.

Coordination of the response to vulnerabilities discovered in open-source software. The developers of open-source software sometimes coordinate their response to new vulnerabilities through social media, e.g. mailing lists, blogs and Twitter. An example for this behavior is a tweet about a `wget` patch for CVE-2014-4877 posted by the patch developer, followed by retweets and advice to update the binaries. If the public discussion starts before a patch is completed, then this is potentially dangerous. However, in the 5 such cases we identified, the patching recommendations were first posted on Twitter and followed by an increased retweet volume.

Leaks from the coordinated disclosure process. In some cases, the participants in the coordinated disclosure process leak information before disclosure. For example, security researchers may tweet about having confirmed

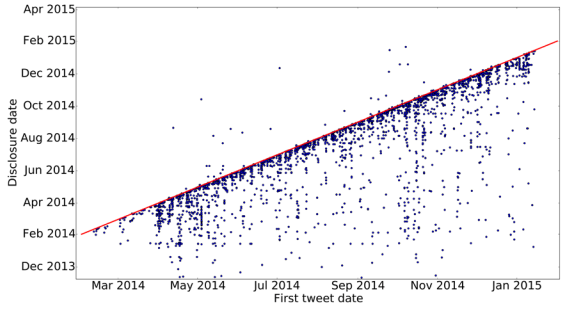


Figure 3: Comparison of the disclosure dates with the dates when the first tweets are posted for all the vulnerabilities in our dataset. Plotted in red is the identity line where the two dates coincide.

that a vulnerability is exploitable, along with the software affected. This is the most dangerous situation, as attackers may then contact the researcher with offers to purchase the exploit, before the vendor is able to release a patch. We have identified 13 such cases.

4.3 Users with information-rich tweets

The tweets we have collected were posted by approximately 32,000 unique users, but the messages posted by these users are not equally informative. Therefore, we quantify utility on a per user basis by computing the ratio of CVE tweets related to real-world exploits as well as the fraction of unique real-world exploits that a given user tweets about. We rank user utility based on the harmonic mean of these two quantities. This ranking penalizes users that tweet about many CVEs indiscriminately (e.g. a security news bot) as well as the thousands of users that only tweet about the most popular vulnerabilities (e.g. Shellshock and Heartbleed). We create a whitelist with the top 20% most informative users, and we use this whitelist in our experiments in the following sections as a means of isolating our classifier from potential adversarial attacks. Top ranked whitelist users include computer repair servicemen posting about the latest viruses discovered in their shops and security researchers and enthusiasts sharing information about the latest blog and news postings related to vulnerabilities and exploits.

Figure 4 provides an example of how the information about vulnerabilities and exploits propagates on Twitter amongst all users. The "Futex" bug, which enables unauthorized root access on Linux systems, was disclosed on June 6 as CVE-2014-3153. After identifying users who posted messages that had retweets counting for at least 1% of the total tweet counts for this vulnerability and applying a thresholding based on the number of retweets,

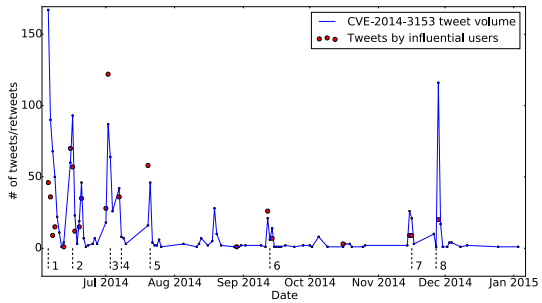


Figure 4: Tweet volume for CVE-2014-3153 and tweets from the most influential users. These influential tweets shape the volume of Twitter posts about the vulnerability. The 8 marks represent important events in the vulnerability's lifecycle: 1 - disclosure, 2 - Android exploit called Towelroot is reported and exploitation attempts are detected in the wild, 3,4,5 - new technical details emerge, including the Towelroot code, 6 - new mobile phones continue to be vulnerable to this exploit, 7 - advisory about the vulnerability is posted, 8 - exploit is included in ExploitDB.

we identify 20 influential tweets that shaped the volume of Twitter messages. These tweets correspond to 8 important milestones in the vulnerability's lifecycle, as marked in the figure.

While CVE-2014-3153 is known to be exploited in the wild, it is not included in our ground truth for real-world exploits, which does not cover the Linux platform. This example illustrates that monitoring a subset of users can yield most of the vulnerability- and exploit-related information available on Twitter. However, over reliance on a small number of user accounts, even with manual analysis, can increase susceptibility to data manipulation via adversarial account hijacking.

5 Detection of proof-of-concept and real-world exploits

To provide a baseline for our ability to classify exploits, we first examine the performance of a classifier that uses only the CVSS score, which is currently recommended as the reference assessment method for software security [50]. We use the total CVSS score and the exploitability subscore as a means of establishing baseline classifier performances. The exploitability subscore is calculated as a combination of the CVSS access vector, access complexity, and authentication components. Both the total score and exploitability subscore range from 0-10. By varying a threshold across the full range of values for each score, we can generate putative labels

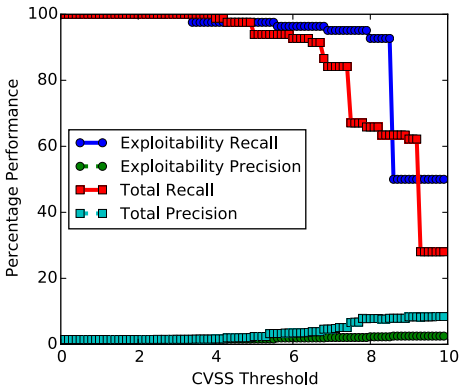
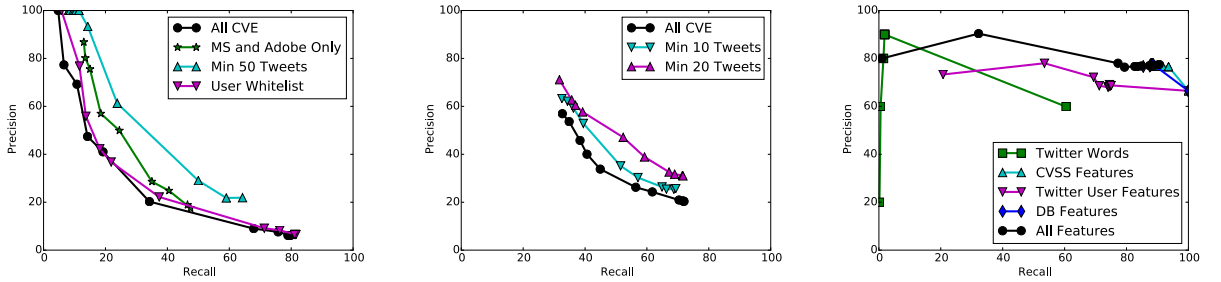


Figure 5: Precision and recall for classifying real world exploits with CVSS score thresholds.

where vulnerabilities with scores above the threshold are marked as "real-world exploits" and vulnerabilities below the threshold are labeled as "not exploited". Unsurprisingly, since CVSS is designed as a high recall system which errs on the side of caution for vulnerability severity, the maximum possible precision for this baseline classifier is less than 9%. Figure 5 shows the recall and precision values for both total CVSS score thresholds and CVSS exploitability subscore thresholds.

Thus, this high recall, low precision vulnerability score is not useful by itself for real-world exploit identification, and boosting precision is a key area for improvement.

Classifiers for real-world exploits. Classifiers for real-world exploits have to deal with a severe class imbalance: we have found evidence of real-world exploitation for only 1.3% of the vulnerabilities disclosed during our observation period. To improve the classification precision, we train linear SVM classifiers on a combination of CVSS metadata features, features extracted from security-related tweets, and features extracted from NVD and OSVDB (see Figure 2). We tune these classifiers by varying the regularization parameter C, and we illustrate the precision and recall achieved. Values shown are for cross-validation testing averaged across 10 stratified random shuffles. Figure 6a shows the average cross-validated precision and recall that are simultaneously acheivable with our Twitter-enhanced feature set. These classifiers can achieve higher precision than a baseline classifier that uses only the CVSS score, but there is still a tradeoff between precision and recall. We can tune the classifier with regularization to decrease the number of false positives (increasing precision), but this comes at the cost of a larger number of false negatives (decreasing recall).



(a) Real-world exploits.

(b) Public proof-of-concept exploits.

(c) Private proof-of-concept exploits for vulnerabilities in Microsoft products.

Figure 6: Precision and recall of our classifiers.

This result partially reflects an additional challenge for our classifier: the fact that our ground truth is imperfect, as Symantec does not have products for all the platforms that are targeted in attacks. If our Twitter-based classifier predicts that a vulnerability is exploited and the exploit exists, but is absent from the Symantec signatures, we will count this instance as a false positive, penalizing the reported precision. To assess the magnitude of this problem, we restrict the training and evaluation of the classifier to the 477 vulnerabilities in Microsoft and Adobe products, which are likely to have a good coverage in our ground truth for real-world exploits (see Section 3.1). 41 of these 477 vulnerabilities are identified as real-world exploits in our ground truth. For comparison, we include the performance of this classifier in Figure 6a. Improving the quality of the ground truth allows us to bolster the values of precision and recall which are simultaneously achievable, while still enabling classification precision an order of magnitude larger than a baseline CVSS score-based classifier. Additionally, while restricting the training of our classifier to a whitelist made up of the top 20% most informative Twitter users (as described in Section 4.3) does not enhance classifier performance, it does allow us to achieve a precision comparable to the previous experiments (Figure 6a). This is helpful for preventing an adversary from poisoning our classifier, as discussed in Section 6.

These results illustrate the current potential and limitations for predicting real-world exploits using publicly-available information. Further improvements in the classification performance may be achieved through a broader effort for sharing information about exploits active in the wild, in order to assemble a high-coverage ground truth for training of classifiers.

Classifiers for proof-of-concept exploits. We explore two classification problems: predicting *public proof-of-concept exploits*, for which the exploit code is publicly available, and predicting *private proof-of-concept exploits*, for which we can find reliable information that the exploit was developed, but it was not released to the public. We consider these problems separately, as they have different security implications and the Twitter users are likely to discuss them in distinct ways.

First, we train a classifier to predict the availability of exploits in ExploitDB [3], the largest archive of *public* exploits. This is similar to the experiment reported by Bozorgi et al. in [36], except that our feature set is slightly different—in particular, we extract word features from Twitter messages, rather than from the textual descriptions of the vulnerabilities. However, we include the most useful features for predicting proof-of-concept exploits, as reported in [36]. Additionally, Bozorgi et al. determined the availability of proof-of-concept exploits using information from OSVDB [9], which is typically populated using references to ExploitDB but may also include vulnerabilities for which the exploit is rumored or private (approximately 17% of their exploit data set). After training a classifier with the information extracted about the vulnerabilities disclosed between 1991–2007, they achieved a precision of 87.5%.

Surprisingly, we are not able to reproduce their performance results, as seen in Figure 6b, when analyzing the vulnerabilities that appear in ExploitDB in 2014. The figure also illustrates the performance of a classifier trained with exclusion threshold for CVEs that lack sufficient quantities of tweets. These volume thresholds improve performance, but not dramatically.[4] In part, this is due to our smaller data set compared to [36], made up of vulnerabilities disclosed during one year rather than a 16-year period. Moreover, our ground truth for public proof-of-concept exploits also exhibits a high class im-

[4] In this case, we do not restrict the exploits to specific vendors, as ExploitDB will incorporate any exploit submitted.

balance: only 6.2% of the vulnerabilities disclosed during our observation period have exploits available in ExploitDB. This is in stark contrast to the prior work, where more than half of vulnerabilities had proof of concept exploits.

This result provides an interesting insight into the evolution of the threat landscape: today, proof-of-concept exploits are less centralized than in 2007, and ExploitDB does not have total coverage of public proof-of-concept exploits. We have found several tweets with links to PoC exploits, published on blogs or mailing lists, that were missing from ExploitDB (we further analyze these instances in Section 4). This suggests that information about public exploits is increasingly dispersed among social media sources, rather than being included in a centralized database like ExploitDB,[5] which represents a hurdle for prioritizing the response to vulnerability disclosures.

To address this concern, we explore the potential for predicting the existence of *private* proof-of-concept exploits by considering only the vulnerabilities disclosed in Microsoft products and by using Microsoft's Exploitability Index to derive our ground truth. Figure 6c illustrates the performance of a classifier trained with a conservative ground truth in which we treat vulnerabilities with scores of 1 or less as exploits. This classifier achieves precision and recall higher than 80%, even when only relying on database feature subsets. Unlike in our prior experiments, for the Microsoft Exploitability Index the classes are more balanced: 67% of the vulnerabilities (218 out of 327) are labeled as having a private proof-of-concept exploit. However, only 8% of the Microsoft vulnerabilities in our dataset are contained within our real-world exploit ground truth. Thus, by using a conservative ground truth that labels many vulnerabilities as exploits we can achieve high precision and recall, but this classifier performance does not readily translate to real-world exploit prediction.

Contribution of various feature groups to the classifier performance. To understand how our features contribute to the performance of our classifiers, in Figure 7 we compare the precision and recall of our real-world exploit classifier when using different subgroups of features. In particular, incorporating Twitter data into the classifiers allows for improving the precision beyond the levels of precision achievable with data that is currently available publicly in vulnerability databases. Both user features and word features generated based on tweets are capable of bolstering classifier precision in comparison to CVSS and features extracted from NVD and OSVDB. Consequently, the analysis of social media streams like Twitter is useful for boosting identification of exploits active in the real-world.

[5] Indeed, OSVDB no longer seems to provide the exploitation availability flag.

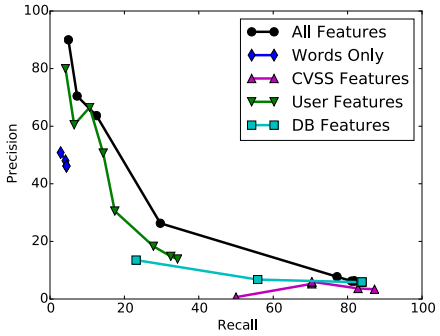


Figure 7: Precision and recall for classification of real world exploits with different feature subsets. Twitter features allow higher-precision classification of real-world exploits.

5.1 Early detection of exploits

In this section we ask the question: *How soon can we detect exploits active in the real world by monitoring the Twitter stream?* Without rapid detecton capabilities that leverage the real-time data availability inherent to social media platforms, a Twiter-based vulnerability classifier has little practical value. While the first tweets about a vulnerability precede the creation of IPS or AV signatures by a median time of 10 days, these first tweets are typically not informative enough to determine that the vulnerability is likely exploited. We therefore simulate a scenario where our classifier for real-world exploits is used in an online manner, in order to identify the dates when the output of the classifier changes from "not exploited" to "exploited" for each vulnerability in our ground truth.

We draw 10 stratified random samples, each with 50% coverage of "exploited" and "not exploited" vulnerabilities, and we train a separate linear SVM classifier with each one of these samples ($C = 0.0003$). We start testing each of our 10 classifiers with a feature set that does not include features extracted from tweets, to simulate the activation of the online classifier. We then continue to test the ensemble of classifiers incrementally, by adding one tweet at a time to the testing set. We update the aggregated prediction using a moving average with a window of 1000 tweets. Figure 8a highlights the tradeoff between precision and early detection for a range of aggregated SVM prediction thresholds. Notably, though, large precision sacrifices do not necessarily lead to large

gains for the speed of detection. Therefore, we choose an aggregated prediction threshold of 0.95, which achieves 45% precision and a median lead prediction time of two days before the first Symantec AV or WINE IPS signature dates. Figure 8b shows how our classifier detection lags behind first tweet appearances. The solid blue line shows the cumulative distribution function (CDF) for the number of days difference between the first tweet appearance for a CVE and the first Symantec AV or IPS attack signature. The green dashed line shows the CDF for the day difference between 45% precision classification and the signature creation. Negative day differences indicate that Twitter events occur before the creation of the attack signature. In approximately 20% of cases, early detection is impossible because the first tweet for a CVE occurs after an attack signature has been created. For the remaining vulnerabilities, Twitter data provides valuable insights into the likelihood of exploitation.

Figure 8c illustrates early detection for the case of Heartbleed (CVE-2014-1060). The green line indicates the first appearance of the vulnerability in ExploitDB, and the red line indicates the date when a Symantec attack signature was published. The dashed black line represents the earliest time when our online classifier is able to detect this vulnerability as a real-world exploit at 45% precision. Our Twitter-based classifier provides an "exploited" output 3 hours after the first tweet appears related to this CVE on April 7, 2014. Heartbleed exploit traffic was detected 21 hours after the vulnerability's public disclosure [41]. Heartbleed appeared in ExploitDB on the day after disclosure (April 8, 2014), and Symantec published the creation of an attack signature on April 9, 2014. Additionally, by accepting lower levels of precision, our Twitter-based classifiers can achieve even faster exploit detection. For example, with classifier precision set to approximately 25%, Heartbleed can be detected as an exploit within 10 minutes of its first appearance on Twitter.

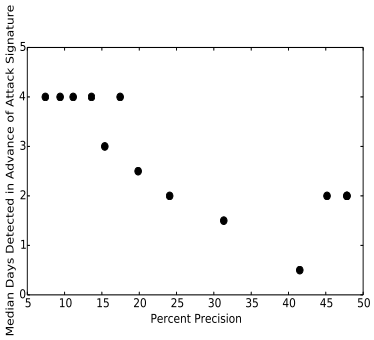
6 Attacks against the exploit detectors

The public nature of Twitter data necessitates considering classification problems not only in an ideal environment, but also in an environment where adversaries may seek to poison the classifiers. In causative adversarial machine learning (AML) attacks, the adversaries make efforts to have a direct influence by corrupting and altering the training data [28, 29, 31].

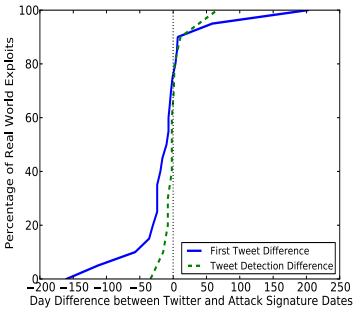
With Twitter data, learning the statistics of the training data is as simple as collecting tweets with either the REST or Streaming APIs. Features that are likely to be used in classification can then be extracted and evaluated using criteria such as correlation, entropy, or mutual information, when ground truth data is publicly available. In this regard, the most conservative assumption for security is that an adversary has complete knowledge of a Twitter-based classifier's training data as well as knowledge of the feature set.

When we assume an adversary works to create both false negatives and false positives (an availability AML security violation), practical implementation of a basic causative AML attack on Twitter data is relatively straightforward. Because of the popularity of spam on Twitter, websites such as `buyaccs.com` cheaply sell large volumes of fraudulent Twitter accounts. For example, on February 16, 2015 on buyaccs.com, the baseline price for 1000 AOL email-based Twitter accounts was $17 with approximately 15,000 accounts available for purchase. This makes it relatively cheap (less than $300 as a base cost) to conduct an attack in which a large number of users tweet fraudulent messages containing CVEs and keywords, which are likely to be used in a Twitter-based classifier as features. Such an attacker has two main limitations. The first limitation is that, while the attacker can add an extremely large number of tweets to the Twitter stream via a large number of different accounts, the attacker has no straightforward mechanism for removing legitimate, potentially informative tweets from the dataset. The second limitation is that additional costs must be incurred if an attacker's fraudulent accounts are to avoid identification. Cheap Twitter accounts purchased in bulk have low friend counts and low follower counts. A user profile-based preprocessing stage of analysis could easily eliminate such accounts from the dataset if an adversary attempts to attack a Twitter classification scheme in such a rudimentary manner. Therefore, to help make fraudulent accounts seem more legitimate and less readily detectable, an adversary must also establish realistic user statistics for these accounts.

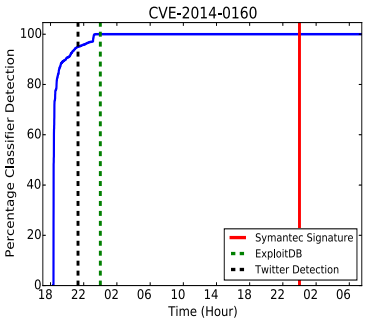
Here, we analyze the robustness of our Twitter-based classifiers when facing three distinct causative attack strategies. The first attack strategy is to launch a causative attack without any knowledge of the training data or ground truth. This *blabbering adversary* essentially amounts to injecting noise into the system. The second attack strategy corresponds to the *word-copycat adversary*, who does not create a sophisticated network between the fraudulent accounts and only manipulates word features and the total tweet count for each CVE. This attacker sends malicious tweets, so that the word statistics for tweets about non-exploited and exploited CVEs appear identical at a user-naive level of abstraction. The third, most powerful adversary we consider is the *full-copycat adversary*. This adversary manipulates the user statistics (friend, follower, and status counts) of a large number of fraudulent accounts as well as the text content of these CVE-related tweets to launch a more sophisticated Sybil attack. The only user statis-



(a) Tradeoff between classification speed and precision.



(b) CDFs for day differences between first tweet for a CVE, first classification, and attack signature creation date.



(c) Heartbleed detection timeline between April 7th and April 10th, 2014.

Figure 8: Early detection of real-world vulnerability exploits.

tics which we assume this full copycat adversary cannot arbitrarily manipulate are account verification status and account creation date, since modifying these features would require account hijacking. The goal of this adversary is for non-exploited and exploited CVEs to appear as statistically identical as possible on Twitter at a user-anonymized level of abstraction.

For all strategies, we assume that the attacker has purchased a large number of fraudulent accounts. Fewer than 1,000 out of the more than 32,000 Twitter users in our CVE tweet dataset send more than 20 CVE-related tweets in a year, and only 75 accounts send 200 or more CVE-related tweets. Therefore, if an attacker wishes to avoid tweet volume-based blacklisting, then each account cannot send a high number of CVE-related tweets. Consequently, if the attacker sets a volume threshold of 20-50 CVE tweets per account, then 15,000 purchased accounts would enable the attacker to send 300,000-750,000 adversarial tweets.

The blabbering adversary, even when sending 1 million fraudulent tweets, is not able to force the precision of our exploit detector below 50%. This suggests that Twitter-based classifiers can be relatively robust to this type of random noise-based attack (black circles in Fig. 9). When dealing with the word-copycat adversary (green squares in Fig. 9), performance asymptotically degrades to 30% precision. The full-copycat adversary can cause the precision to drop to approximately 20% by sending over 300,000 tweets from fraudulent accounts. The full-copycat adversary represents a practical upper bound for the precision loss that a realistic attacker can inflict on our system. Here, performance remains above baseline levels even for our strongest Sybil attacker due to our use of non-Twitter features to increase classifier robustness. Nevertheless, in order to recover performance, implementing a Twitter-based vulnerability classifier in a realistic setting is likely to require curation of whitelists and blacklists for informative and adversarial users. As shown in figure 6a, restricting the classifier to only consider the top 20% of users with the most relevant tweets about real-world exploits causes no performance degradation and fortifies the classifier against low tier adversarial threats.

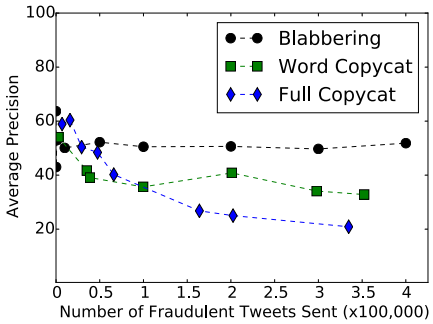


Figure 9: Average linear SVM precision when training and testing data are poisoned by the three types of adversaries from our threat model.

7 Related work

Previous work by Allodi et al. has highlighted multiple deficiencies in CVSS version 2 as a metric for predicting whether or not a vulnerability will be exploited in the wild [24], specifically because predicting the small fraction of vulnerabilities exploited in the wild is not one of the design goals of CVSS. By analyzing vulnerabilities in exploit kits, work by Allodi et al. has also established

that Symantec threat signatures are an effective source of information for determining which vulnerabilities are exploited in the real world, even if the coverage is not complete for all systems [23].

The closest work to our own is Bozorgi et al., who applied linear support vector machines to vulnerability and exploit metadata in order predict the development of proof-of-concept exploits [36]. However, the existence of a POC exploit does not necessarily mean that an exploit will be leveraged for attacks in the wild, and real world attacks occur in only of a small fraction of the cases for which a vulnerability has a proof of concept exploit. [25, 47]. In contrast, we aim to detect exploits that are active in the real world. Hence, our analysis expands on this prior work [36] by focusing classifier training on real world exploit data rather than POC exploit data and by targeting social media data as a key source of features for distinguishing real world exploits from vulnerabilities that are not exploited in the real world.

In prior analysis of Twitter data, success has been found in a wide variety of applications including earthquake detection [53], epidemiology [26, 37], and the stock market [34, 58]. In the security domain, much attention has been focused on detecting Twitter spam accounts [57] and detecting malicious uses of Twitter aimed at gaining political influence [30, 52, 55]. The goals of these works is distinct from our task of predicting whether or not vulnerabilities are exploited in the wild. Nevertheless, a practical implementation of our vulnerability classification methodology would require the detection of fraudulent tweets and spam accounts to prevent poisoning attacks.

8 Discussion

Security in Twitter analytics. Twitter data is publicly available, and new users are free to join and start sending messages. In consequence, we cannot obfuscate or hide the features we use in our machine learning system. Even if we had not disclosed the features we found most useful for our problem, an adversary can collect Twitter data, as well as the data sets we use for ground truth (which are also public), and determine the most information rich features within the training data in the the same way we do. Our exploit detector is an example of a *security system without secrets*, where the integrity of the system does not depend on the secrecy of its design or of the features it uses for learning. Instead, the security properties of our system derive from the fact that the adversary can inject new messages in the Twitter stream, but cannot remove any messages sent by the other users. Our threat model and our experimental results provide practical bounds for the damage the adversary can inflict on such a system. This damage can be reduced further by incorporating techniques for identifying adversarial Twitter accounts, for example by assigning a reputation score to each account [44, 48, 51].

Applications of early exploit detection. Our results suggest that, the information contained in security-related tweets is an important source for timely security-related information. Twitter-based classifiers can be employed to guide the prioritization of response actions after vulnerability disclosures, especially for organizations with strict policies for testing patches prior to enterprise-wide deployment, which makes patching a resource-intensive effort. Another potential application is modeling the risk associated with vulnerabilities, for example by combining the likelihood of real-world exploitation, produced by our system, with additional metrics for vulnerability assessment, such as the CVSS severity scores or the odds that the vulnerable software is exploitable given its deployment context (e.g. whether it is attached to a publicly-accessible network). Such models are key for the emerging area of cyber-insurance [33], and they would benefit from an evidence-based approach for estimating the likelihood of real-world exploitation.

Implications for information sharing efforts. Our results highlight the current challenges for the early detection of exploits, in particular the fact that the existing sources of information for exploits active in the wild do not cover all the platforms that are targeted by attackers. The discussions on security-related mailing lists, such as Bugtraq [10], Full Disclosure [4] and oss-security [8], focus on disclosing vulnerabilities and publishing exploits, rather than on reporting attacks in the wild. This makes it difficult for security researchers to assemble a high-quality ground truth for training supervised machine learning algorithms. At the same time, we illustrate the potential of this approach. In particular,our whitelist identifies 4,335 users who post information-rich messages about exploits. We also show that the classification performance can be improved significantly by utilizing a ground truth with better coverage. We therefore encourage the victims of attacks to share relevant technical information, perhaps through recent information-sharing platforms such as Facebook's ThreatExchange [42] or the Defense Industrial Base voluntary information sharing program [1].

9 Conclusions

We conduct a quantitative and qualitative exploration of information available on Twitter that provides early warnings for the existence of real-world exploits. Among the products for which we have reliable ground truth, we identify more vulnerabilities that are *exploited in*

the real-world than vulnerabilities for which *proof-of-concept exploits* are available publicly. We also identify a group of 4,335 users who post information-rich messages about real-world exploits. We review several unique challenges for the exploit detection problem, including the skewed nature of vulnerability datasets, the frequent scarcity of data available at initial disclosure times and the low coverage of real world exploits in the ground truth data sets that are publicly available. We characterize the threat of information leaks from the coordinated disclosure process, and we identify features that are useful for detecting exploits.

Based on these insights, we design and evaluate a detector for real-world exploits utilizing features extracted from Twitter data (e.g., specific words, number of retweets and replies, information about the users posting these messages). Our system has fewer false positives than a CVSS-based detector, boosting the detection precision by *one order of magnitude*, and can detect exploits a median of *2 days ahead of existing data sets*. We also introduce a threat model with three types of adversaries seeking to poison our exploit detector, and, through simulation, we present practical bounds for the damage they can inflict on a Twitter-based exploit detector.

Acknowledgments

We thank Tanvir Arafin for assistance with our early data collection efforts. We also thank the anonymous reviewers, Ben Ransford (our shepherd), Jimmy Lin, Jonathan Katz and Michelle Mazurek for feedback on this research. This research was supported by the Maryland Procurement Office, under contract H98230-14-C-0127 and by the National Science Foundation, which provided cluster resources under grant CNS-1405688 and graduate student support with GRF 2014161339.

References

[1] Cyber security / information assurance (CS/IA) program. `http://dibnet.dod.mil/`.

[2] Drupalgeddon vulnerability - cve-2014-3704. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3704`.

[3] Exploits database by offensive security. `https:http://exploit-db.com/`.

[4] Full disclosure mailing list. `http://seclists.org/fulldisclosure/`.

[5] Microsoft active protections program. `https://technet.microsoft.com/en-us/security/dn467918`.

[6] Microsoft's latest plea for cvd is as much propaganda as sincere. http://blog.osvdb.org/2015/01/12/microsofts-latest-plea-for-vcd-is-as-much-propaganda-as-sincere/.

[7] National vulnerability database (NVD). `https://nvd.nist.gov/`.

[8] Open source security. `http://oss-security.openwall.org/wiki/mailing-lists/oss-security`.

[9] Open sourced vulnerability database (OSVDB). `http://www.osvdb.org/`.

[10] Securityfocus bugtraq. `http://www.securityfocus.com/archive/1`.

[11] Shellshock vulnerability - cve-2014-6271. `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271`.

[12] Symantec a-z listing of threats & risks. http://www.symantec.com/security_response/landing/azlisting.jsp.

[13] Symantec attack signatures. `http://www.symantec.com/security_response/attacksignatures/`.

[14] Technet blogs - a call for better coordinated vulnerability disclosure. http://blogs.technet.com/b/msrc/archive/2015/01/11/a-call-for-better-coordinated-vulnerability-disclosure.aspx/.

[15] The twitter streaming api. `https://dev.twitter.com/streaming/overview/`.

[16] Vendors sure like to wave the "coordination" flag. http://blog.osvdb.org/2015/02/02/vendors-sure-like-to-wave-the-coordination-flag-revisiting-the-perfect-storm/.

[17] Windows elevation of privilege in user profile service - google security research. https://code.google.com/p/google-security-research/issues/detail?id=123.

[18] Guidelines for security vulnerability reporting and response. Tech. rep., Organization for Internet Safety, September 2004. `http://www.symantec.com/security/OIS_Guidelines%20for%20responsible%20disclosure.pdf`.

[19] Adding priority ratings to adobe security bulletins. http://blogs.adobe.com/security/2012/02/when-do-i-need-to-apply-this-update-adding-priority-ratings-to-adobe-security-bulletins-2.html, 2012.

[20] ACHREKAR, H., GANDHE, A., LAZARUS, R., YU, S.-H., AND LIU, B. Predicting flu trends using twitter data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on* (2011), IEEE, pp. 702–707.

[21] ALBERTS, B., AND RESEARCHER, S. A Bounds Check on the Microsoft Exploitability Index The Value of an Exploitability Index Exploitability. 1–7.

[22] ALLODI, L., AND MASSACCI, F. A Preliminary Analysis of Vulnerability Scores for Attacks in Wild. In *CCS BADGERS Workshop* (Raleigh, NC, Oct. 2012).

[23] ALLODI, L., AND MASSACCI, F. A preliminary analysis of vulnerability scores for attacks in wild. *Proceedings of the 2012 ACM Workshop on Building analysis datasets and gathering experience returns for security - BADGERS '12* (2012), 17.

[24] ALLODI, L., AND MASSACCI, F. Comparing vulnerability severity and exploits using case-control studies. *(Rank B)ACM Transactions on Embedded Computing Systems 9*, 4 (2013).

[25] ALLODI, L., SHIM, W., AND MASSACCI, F. Quantitative Assessment of Risk Reduction with Cybercrime Black Market Monitoring. *2013 IEEE Security and Privacy Workshops* (2013), 165–172.

[26] ARAMAKI, E., MASKAWA, S., AND MORITA, M. Twitter Catches The Flu : Detecting Influenza Epidemics using Twitter The University of Tokyo. In *Proceedings of the 2011 Conference on Emperical Methods in Natural Language Processing* (2011), pp. 1568–1576.

[27] ASUR, S., AND HUBERMAN, B. A. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on* (2010), vol. 1, IEEE, pp. 492–499.

[28] BARRENO, M., BARTLETT, P. L., CHI, F. J., JOSEPH, A. D., NELSON, B., RUBINSTEIN, B. I., SAINI, U., AND TYGAR, J. Open problems in the security of learning. In *Proceedings of the 1st ...* (2008), p. 19.

[29] BARRENO, M., NELSON, B., JOSEPH, A. D., AND TYGAR, J. D. The security of machine learning. *Machine Learning 81* (2010), 121–148.

[30] BENEVENUTO, F., MAGNO, G., RODRIGUES, T., AND ALMEIDA, V. Detecting spammers on twitter. *Collaboration, electronic messaging, anti-abuse and spam conference (CEAS) 6* (2010), 12.

[31] BIGGIO, B., NELSON, B., AND LASKOV, P. Support Vector Machines Under Adversarial Label Noise. *ACML 20* (2011), 97–112.

[32] BILGE, L., AND DUMITRAȘ, T. Before we knew it: An empirical study of zero-day attacks in the real world. In *ACM Conference on Computer and Communications Security* (2012), T. Yu, G. Danezis, and V. D. Gligor, Eds., ACM, pp. 833–844.

[33] BÖHME, R., AND SCHWARTZ, G. Modeling Cyber-Insurance: Towards a Unifying Framework. In *WEIS* (2010).

[34] BOLLEN, J., MAO, H., AND ZENG, X. Twitter mood predicts the stock market. *Journal of Computational Science 2* (2011), 1–8.

[35] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory* (1992), pp. 144–152.

[36] BOZORGI, M., SAUL, L. K., SAVAGE, S., AND VOELKER, G. M. Beyond Heuristics : Learning to Classify Vulnerabilities and Predict Exploits. In *KDD '10 Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (2010), p. 9.

[37] BRONIATOWSKI, D. A., PAUL, M. J., AND DREDZE, M. National and local influenza surveillance through twitter: An analysis of the 2012-2013 influenza epidemic. *PLoS ONE 8* (2013).

[38] CHANG, C.-C., AND LIN, C.-J. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1—-27:27.

[39] CORTES, C., CORTES, C., VAPNIK, V., AND VAPNIK, V. Support Vector Networks. *Machine Learning 20* (1995), 273˜–˜297.

[40] DUMITRAȘ, T., AND SHOU, D. Toward a Standard Benchmark for Computer Security Research: The {Worldwide Intelligence Network Environment (WINE)}. In *EuroSys BADGERS Workshop* (Salzburg, Austria, Apr. 2011).

[41] DURUMERIC, Z., KASTEN, J., ADRIAN, D., HALDERMAN, J. A., BAILEY, M., LI, F., WEAVER, N., AMANN, J., BEEKMAN, J., PAYER, M., AND PAXSON, V. The matter of Heartbleed. In *Proceedings of the Internet Measurement Conference* (Vancouver, Canada, Nov. 2014).

[42] FACEBOOK. ThreatExchange. `https://threatexchange.fb.com/`.

[43] GUYON, I., GUYON, I., BOSER, B., BOSER, B., VAPNIK, V., AND VAPNIK, V. Automatic Capacity Tuning of Very Large VC-Dimension Classifiers. In *Advances in Neural Information Processing Systems* (1993), vol. 5, pp. 147–155.

[44] HORNG, D., CHAU, P., NACHENBERG, C., WILHELM, J., WRIGHT, A., AND FALOUTSOS, C. Polonium : Tera-Scale Graph Mining and Inference for Malware Detection. *Siam International Conference on Data Mining (Sdm)* (2011), 131–142.

[45] LAZER, D. M., KENNEDY, R., KING, G., AND VESPIGNANI, A. The parable of Google Flu: traps in big data analysis.

[46] MITRE. Common vulnerabilities and exposures (CVE). urlhttp://cve.mitre.org/.

[47] NAYAK, K., MARINO, D., EFSTATHOPOULOS, P., AND DUMITRAȘ, T. Some Vulnerabilities Are Different Than Others: Studying Vulnerabilities and Attack Surfaces in the Wild. In *Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses* (Gothenburg, Sweeden, Sept. 2014).

[48] PANDIT, S., CHAU, D., WANG, S., AND FALOUTSOS, C. Netprobe: a fast and scalable system for fraud detection in online auction networks. *Proceedings of the 16th ... 42* (2007), 210, 201.

[49] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[50] QUINN, S., SCARFONE, K., BARRETT, M., AND JOHNSON, C. Guide to Adopting and Using the Security Content Automation Protocol {(SCAP)} Version 1.0. NIST Special Publication 800-117, July 2010.

[51] RAJAB, M. A., BALLARD, L., LUTZ, N., MAVROMMATIS, P., AND PROVOS, N. CAMP: Content-agnostic malware protection. In *Network and Distributed System Security (NDSS) Symposium* (San Diego, CA, Feb 2013).

[52] RATKIEWICZ, J., CONOVER, M. D., MEISS, M., GONC, B., FLAMMINI, A., AND MENCZER, F. Detecting and Tracking Political Abuse in Social Media. *Artificial Intelligence* (2011), 297–304.

[53] SAKAKI, T., OKAZAKI, M., AND MATSUO, Y. Earthquake shakes Twitter users: real-time event detection by social sensors. In *WWW '10: Proceedings of the 19th international conference on World wide web* (2010), p. 851.

[54] SCARFONE, K., AND MELL, P. An analysis of CVSS version 2 vulnerability scoring. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009* (2009), pp. 516–525.

[55] THOMAS, K., GRIER, C., AND PAXSON, V. Adapting Social Spam Infrastructure for Political Censorship. In *LEET* (2011).

[56] THOMAS, K., LI, F., GRIER, C., AND PAXSON, V. Consequences of Connectivity: Characterizing Account Hijacking on Twitter. ACM Press, pp. 489–500.

[57] WANG, A. H. Don't follow me: Spam detection in Twitter. *2010 International Conference on Security and Cryptography (SECRYPT)* (2010), 1–10.

[58] WOLFRAM, M. S. A. Modelling the Stock Market using Twitter. *iccsinformaticsedacuk Master of* (2010), 74.