Evaluating the Performance of Twitter-based Exploit Detectors

Daniel Alves de Sousa[1], **Elaine Ribeiro de Faria**[1], **Rodrigo Sanches Miani**[1]

[1]School of Computer Science (FACOM) – Federal University of Uberlândia (UFU)
Uberlândia, MG, Brazil

{danielsousa,elaine,miani}@ufu.br

Abstract. *Patch prioritization is a crucial aspect of information systems security, and knowledge of which vulnerabilities were exploited in the wild is a powerful tool to help systems administrators accomplish this task. The analysis of social media for this specific application can enhance the results and bring more agility by collecting data from online discussions and applying machine learning techniques to detect real-world exploits. In this paper, we use a technique that combines Twitter data with public database information to classify vulnerabilities as exploited or not-exploited. We analyze the behavior of different classifying algorithms, investigate the influence of different antivirus data as ground truth, and experiment with various time window sizes. Our findings suggest that using a Light Gradient Boosting Machine (LightGBM) can benefit the results, and for most cases, the statistics related to a tweet and the users who tweeted are more meaningful than the text tweeted. We also demonstrate the importance of using ground-truth data from security companies not mentioned in previous works.*

1. Introduction

Exploit detection is an essential application for system administrators as system updates sometimes involve rigorous impact analysis and even severe adaptations or migrations. An exploited vulnerability on a particular system should demand urgent responses from its administrator in the sense of prioritizing patches. Given this scenario and the fact that many vulnerabilities might never be exploited in real-world attacks [Nayak et al. 2014], knowledge of which vulnerabilities are more likely to be exploited in the wild can be an excellent tool for system administrators to prioritize patch deployments. There are a few metrics that could be used with this purpose (such as the Common Vulnerability Score System - CVSS base scores and the Microsoft Update Severity Rating System), but they err on the side of caution [Younis and Malaiya 2015]. The analysis of social media data can leverage this process by taking advantage of the community's discussions on the topic [Shrestha et al. 2020].

The work presented in [Sabottke et al. 2015] has shown that hackers, system administrators, and software vendors discuss vulnerabilities on social media like Twitter. The authors also showed, for the first time, that this information could be used to create a framework for predicting exploits using machine learning techniques. Several works published after that investigated the feasibility of such *Twitter-based early exploit detectors* ([Bullough et al. 2017], [Queiroz et al. 2017], and [Chen et al. 2019]). Despite these works shown the potential of using Twitter data to detect exploits, they still have

limitations. References [Bullough et al. 2017] and [Queiroz et al. 2017], for example, uses only Support Vector Machine (SVM) as its classifier algorithm. [Chen et al. 2019] partially solves this issue by adding several other classifiers in their study. However, they do not provide a performance comparison using the original dataset described in [Sabottke et al. 2015]. Another issue is related to the use of a single source, Symantec Intrusion Protection Signature, for building the ground-truth of real-world exploits. As discussed in [Sabottke et al. 2015], this is a notable limitation since Symantec does not cover all platforms and products uniformly.

Moreover, none of the previous work evaluates the impact of training Twitter-based exploit detectors using past data to predict the future. For example, suppose that an exploit detector was trained using data from 2017. What happens if only data from 2018 would be presented to this model? Would add more training data gives better performance? This discussion is important to evaluate the practical implications of using Twitter data to build exploit detectors and consequently helping prioritize which vulnerabilities to patch. Based on these analyses, we propose to evaluate the extent to which different factors influence the performance of Twitter-based exploit detectors. We focus on exploring some machine learning characteristics, training classifiers with different time-window sizes, and evaluating the impact of ground-truth labels from different sources.

The paper has four main contributions. First, we identify a suitable classifier for building Twitter-based exploit detectors using a five-year dataset composed of tweets and vulnerability information. Second, we develop a ground-truth for labeling real-world exploits using data from sources other than Symantec. Third, we provide empirical evidence that using ground-truth information from a single vendor can bias the model toward some vulnerabilities and induce to non-optimal performance on real-world scenarios. Fourth, we examine if the performance of an exploit detector model is affected along the time. Our results suggest that models trained and tested using data from a single calendar year outperform those trained with data from previous years. This indicates that selecting the right amount of past information that will feed the model is decisive to improve its performance.

The rest of this paper is organized as follows. Section 2 presents the basic terminology about security vulnerabilities. Section 3 reviews and compares related studies with this work. Section 4 details the dataset, features and classifiers that are used in our system architecture. Section 5 presents the results and shows some threats to validity. Finally, Section 6 concludes the paper and suggests future work.

2. Terminology

Common Vulnerabilities and Exposures (CVE) is a list maintained by MITRE which assigns a unique number to each disclosed vulnerability. Meltdown vulnerability, for instance, is identified by the number CVE-2017-5754. Aside from the official channels, some vulnerabilities may also be disclosed through forums, social media, or blogs, which may lead to a situation where the CVE of a non-patched flaw can be published. In either scenario, official and non-official disclosure, malicious hackers can take advantage of a vulnerability to harm unpatched systems. *Exploit* is the term used to define the techniques or tools developed with this goal.

Exploits can be divided into two categories: proof of concept (PoC) and real-

world (RW) exploits [Sabottke et al. 2015]. While the first is developed as part of the disclosure process to demonstrate a particular vulnerability, the latest is created to perform real attacks. Although some PoC may be used in real-world scenarios, others are too impractical to be. Therefore, vulnerabilities with exploits in the wild are a subset of the ones with PoC exploits.

3. Related Work

Many previous works have addressed the task of using machine learning to predict whether a vulnerability will be exploited or not. [Bozorgi et al. 2010] trained an SVM classifier using features extracted from the Open Source Vulnerability Database (OSVDB) and the NVD to predict if a vulnerability is likely to be exploited. As ground truth, the authors used a metric called "Exploit Classification" from the OSVDB, no longer available since 2016. Despite getting nearly 90% of accuracy, the ground truth used presents a very high positive rate (exploited vulnerabilities), contrasting with most related works ([Nayak et al. 2014], and [Bilge and Dumitras 2012], for example).

[Sabottke et al. 2015] introduced the use of Twitter to help classifying vulnerabilities as exploited or not exploited. Like [Bozorgi et al. 2010], the authors acquired data from the NVD and the OSVDB, but they added an extra set of features extracted from Twitter, including text and statistics about tweets and users who tweeted. The work divides the ground truth into two groups, PoC and RW exploits, using the Exploit Database (EDB) [1] as a source for PoC and Symantec's antivirus and IPS signatures for RW. They collected tweets from February 2014 and January 2015 and found evidence that the use of Twitter data could increase the classifier's overall performance. The paper, however, does not explore other classifiers options (only SVM was used) or methods to overcome dataset imbalance, it uses only one year's worth of data, and relies on a single antivirus vendor for RW ground truth information. [Queiroz et al. 2017] used a similar approach to detect useful information about security vulnerabilities using Twitter data. They collected posts from security specialists from March 2016 to early March 2017 and manually labeled training data. Despite not being the focus of the paper, the approach was able to identify useful alerts about vulnerability exploits.

[Bullough et al. 2017] raised questions about prior work's methodology and highlighted how small changes in using the dataset could affect the performance of predictive models. The authors have been especially critical about temporal intermixing caused by random splitting data for train and test. Such ideas are valuable and should be considered when planning new models, but their conclusion about using a temporal split may not be accurate. In Section 5.4, we reproduce this test in a variety of ways, and our results suggest that performance differences may have different reasons. Furthermore, the work uses only PoC ground truth from EDB and found 18% of their CVEs exploited, a value significantly above those presented in works about RW exploits.

[Chen et al. 2019] used an ensemble of regression algorithms to predict when a vulnerability will be exploited, both for PoC and RW scenarios. As features, the authors created a graph-based model relating CVE-Authors-Tweet and, for ground truth, only Symantec's data was used. The authors also approached the temporal intermixing issue,

[1] https://www.exploit-db.com

demonstrating how, in some cases, the CVSS is not available at the time of the vulnerability's disclousure. In our work, we will demonstrate in section 5.1 that the CVSS plays a small part in the classifier's performance. Nonetheless, those issues may indicate that more relevant NVD data could be affected similarly and should also be studied.

4. Proposal and Experiments

In this work, we evaluate and propose improvements in the Twitter-based exploit detection method presented in [Sabottke et al. 2015]. We chose to use that paper as our baseline because other related works were not entirely comparable to the best of our knowledge, displaying very different rates of exploited vulnerabilities or using completely different data sources. We start by using the same dataset from the mentioned work, which contains messages posted on Twitter, together with public data, and we experiment with different machine learning techniques to classify if a vulnerability will be exploited. We then extend the ground truth with information from different anti-malware software, and finally, we extend the dataset with data from 2015 to 2018. In all cases, we only consider already cataloged vulnerabilities (those to which a CVE number was assigned). To summarize, this paper has four main goals:

- Compare classification algorithms: we want to compare the performance of four well-known algorithms on classifying vulnerabilities as "exploited" or "not exploited" based on data from Twitter and public vulnerabilities databases. Namely, we compared Support Vector Machines, Logistic Regression, XGBoost, and LightGBM. We also intend to evaluate how different groups of features impact each algorithm. In our method, we divide features into four groups: Twitter text, Twitter metadata, CVSS score and subscores, and a set of data from public vulnerabilities databases (mainly from the NVD).
- Compare Multiple Ground Truth: previous works rely only on Symantec's antivirus and intrusion protection system (IPS) signatures to indicate real-world exploits. We want to evaluate if other antivirus databases can provide useful insight and improve prediction performance.
- Class Balancing: we want to verify if class balancing methods can improve the overall results, given that class imbalance is one of the main challenges of this task.
- Updated Data and Different Time Window Sizes: we want to evaluate how the method behaves on more recent data and understand how the classifier is affected by temporal splits and changes in the volume of training and testing data. To do that, we create time windows with different sizes covering different periods to train and test our model.

Our classifier considers each CVE as an instance to which should be assigned *true*, if the vulnerability was exploited, or *false* otherwise. The features used to characterize each instance summarize the data collected about a specific CVE. They contain a Bag-of-Words (BoW) representation of tweets mentioning that CVE, Twitter statistics and metadata related to those tweets, and public database information about the vulnerability. In Section 4.3 we detail these features.

To train a classifier, we need a way to resolve if a vulnerability has any known exploit. On the first test, we use the same ground truth data as defined in [Sabottke et al.

2015]: the ExploitDB (EDB) and Symantec's antivirus and intrusion protection system (IPS) signatures. For all remaining tests, we improve the ground truth with data from Avast[2], ESET[3], and Trend Micro[4]. The EDB is an online resource of known exploits that also provides information about the vulnerabilities affected. While the EDB is an excellent resource of PoC exploits, an antivirus signature is probably the best indicator that an exploit has been spotted in the wild. Past works only included Symantec's database as a source for such information, but we were able to find similar data from other vendors. In Section 5.2, we analyze the quality of data and how they can impact past results.

4.1. System's Architecture

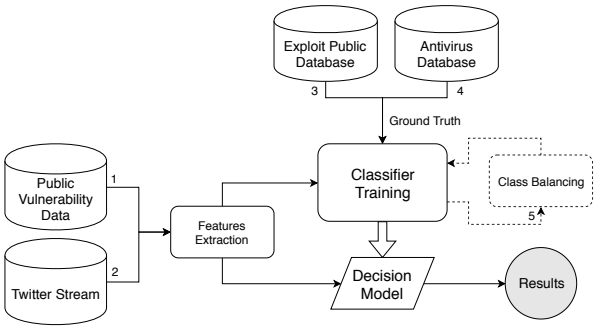


Figure 1. System Architecture

Fig. 1 represents the system used in this work. Data gathering process and its sources (numbers 1 to 4) are detailed in subsection 4.2. Feature extraction process is presented in 4.3 while the machine learning techniques and the balancing phase (number 5) are clarified in 4.4. We developed several Python scripts for supporting the data gathering and feature extraction process. We also use the *scikit-learn* library [Pedregosa et al. 2011] for conducting the classification tasks.

Table 1. Vulnerabilities mentioned on Twitter

	# of Vulnerabilities	Mentioned on Twitter	%
2015	6484	822	13%
2016	6447	776	12%
2017	14714	1292	9%
2018	16556	3753	23%

4.2. Dataset

For the first part of this work, we used the same dataset described in [Sabottke et al. 2015] as it was a good baseline for comparison. In essence, it consists of tweets collected using Twitter's Streaming API from February 2014 to January 2015 using the keyword "CVE". Each tweet was automatically associated with a vulnerability using the CVE number, and then, to each vulnerability, detailed data (such as vendor, CVSS, and description) were collected from the NVD and the Open Source Vulnerability Database (OSVDB). The authors were able to collect 287,717 tweets containing explicit references to CVE numbers and referencing 5,865 different CVEs from the period studied.

[2]https://www.avast.com/exploit-protection.php
[3]https://www.virusradar.com/en/threat_encyclopaedia
[4]https://www.trendmicro.com/vinfo/us/threat-encyclopedia

For the second part of our research, we collected tweets containing the word "CVE" from January 2015 to December 2018, filtered out those referring to vulnerabilities outside our test period and those not mentioning a valid CVE number. We were able to find 44,570 messages, mentioning 6,643 vulnerabilities discussed by 4,033 users. Table 1 shows the number of CVEs mentioned on Twitter by year. In 2018, for example, 23% of all CVEs disclosed in that year were mentioned at least one time on Twitter.

Unlike [Sabottke et al. 2015], which collected tweets through Twitter Stream API, we collected messages by searching old tweets using the GetOldTweets3 tool [Mottl 2018]. We believe this may be the cause of the difference in information volume since accounts and messages deleted will not be found in our method. We also collected data from the NVD for feature extraction. Section 4.3 contains the list of features to each CVE. As discussed in [Sabottke et al. 2015], we also used two different ground truth for the dataset: one for proof-of-concept (PoC) and another for real-world (RW) exploits. Therefore some of the experiments are also divided into two categories, one for each type of exploit. For PoC we used data from ExploitDB. For the real-worlds exploits, most related studies use signatures from Symantec's antivirus and IPS. We included information from four other vendors: Avast, ESET, Trend Micro, and Kaspersky. We believe that relying on a single vendor can lead to biased results and less efficient learning from the model. In all cases, not all signatures mention the CVE exploited, imposing some limitations on the results.

4.3. Features

For all of our tests, we divided the features used by the classifier into four categories: Twitter text, Twitter statistics and metadata, CVSS score, and Public Vulnerabilities Databases. Twitter text represents a combination of all messages tweeted about a CVE. Table 2 shows a summary of our features. We represent the text with the Bag of Word (BoW) model, and the words chosen were the same as in [Sabottke et al. 2015]. The keyword dataset comprises 36 words and some examples include: *0day*, *advisory*, *beware*, *ssl*, and *fix*. Twitter statistics contain data such as the number of retweets related to a CVE and the combined number of followers from all users who tweeted a CVE. Public Vulnerabilities Database originally included information from the NVD and the OSVDB, but we used only the former since the latter is no longer available. The CVSS category contains features representing the CVSS vectors. For the 2015 to 2018 data, we included features for the CVSS 3.0. We also included impact and exploitability subscores for both versions 2.0 and 3.0.

Table 2. Summarized List of Features

Position	Category	Data type
0 - 35	Twitter text	BoW
36 - 47	Twitter metadata	All Numeric
48 - 56	CVSS Score 2.0	3 Numeric, 6 Categorical Ordinal
57 - 67	CVSS Score 3.0	3 Numeric, 8 Categorical Ordinal
68 - 78	Public Vulnerabilities Databases	6 Numeric, 5 Binary

4.4. Classifiers

We used the SVM classifier as our baseline since it is well suitable for text categorization [Joachims 1998] and because it was used in most related works. We tested several supervised classification algorithms, but we will approach the ones which stood out

	2014*	2015	2016	2017	2018
Mentioned CVEs	5,865	822	776	1292	3753
Exploited	77 (1.3%)	71 (8.6%)	31 (4.0%)	61 (4.7%)	133 (4.2%)
PoC	383 (6.5%)	115 (14.0%)	90 (11.6%)	220 (17.0%)	257 (11.9%)

*Sabottke dataset

Table 3. Number of CVEs exploited compared to total

best: Logistic Regression, XGBoost, and Light Gradient Boosting Machine (LightGBM). We also tried several class balancing algorithms available on Python's *imbalanced-learn API* [imbalanced-learn API documentation 2019]. In this paper, we will cover only the ones which performed best with our application: Synthetic Minority Over-sampling Technique (SMOTE), Adaptive Synthetic (ADASYN), Nearest-Neighbor (AllKNN), and the Random Under Sampler (RUS). The first two being over-sampling techniques and the other two under-sampling algorithms.

We used the stratified 10-fold cross-validation and averaged the results. When testing the balancing algorithms, we applied the methods on the training set of each fold. Because our dataset contains features with different data types, and the experimented algorithms also use different types as input, we tested multiple scaling and feature representation methods for each algorithm and used the one which performed best. All categorical-ordinal data (mostly related to the CVSS score) or binary features were first converted to numeric values. Standardization was done through sklearn's *StandardScaler*.

5. Experimental Results

To compare our results with the original work ([Sabottke et al. 2015]), we use as our baseline an SVM classifier with no class balancing and Symantec as the single source of information about real-world exploits. Results are shown using the values of precision ($\frac{TP}{TP+FP}$, the fraction of correct positive prediction), recall ($\frac{TP}{TP+FN}$, the fraction of positive cases that were correctly predicted), and F-score ($2 * \frac{precision*recall}{precision+recall}$, the harmonic mean of precision and recall). We use the precision-recall (PR) curve as the visual representation for all experiments, considering the database is highly imbalanced. Table 3 shows how imbalanced the classes are. Since we already have conservative exploits indicators on the CVSS scores, we prioritize the increase of the precision over the recall without sacrificing the F-score. For each algorithm, we tested two scenarios: PoC and RW exploits.

To extract more significant conclusions, we plotted the classifier results when training and testing with each subset of features. We also plot a line for the results using all features combined. By adopting this strategy, it is possible to see how different features may favor a particular algorithm. Furthermore, we ran tests with combinations of subsets, e.g., CVSS and Twitter Statistics, and eventually concluded that certain features are not suitable for some of the algorithms. We have shortened the names of the categories: *Words* stands for Twitter text, *Twitter Stats* stands for Twitter statistics and metadata, *CVSS* stands for the CVSS score and, *Database* stands for Public Vulnerabilities Databases. A description of each subset can be found in Section 4.3.

5.1. Analyzing the Performance of Different Algorithms and Groups of Features

Table 4 shows the overall performance of the tested algorithms using the same dataset described in [Sabottke et al. 2015] which encompasses data from February 2014 to January 2015. Significance was calculated using a 10-fold cross-validated paired t-test between the algorithm's and the baseline's f-score. P-values greater than 0.05 implies no significant improvement. We also used the f-score because, in our tests, the SVM was the only algorithm which had recall greater than the precision, so using either of these measures could lead to incorrect conclusions. The results reveal that, regardless of the algorithm, there is still room for improvement. We believe this can be achieved with modifications to the preprocessing and the data-gathering technique. We'll discuss that in Section 6. Next, we analyze each of the used algorithms.

	Precision	**Recall**	**F-score**	**P-value**
Baseline - SVM (PoC)	0.2075	0.7053	0.3199	—
LR (PoC)	0.6678	0.2434	0.3568	0.252
XGBoost (PoC)	0.7454	0.2746	0.4014	0.078
LightGBM (PoC)	0.7170	0.3293	0,4513	< 0.001
Baseline - SVM(RW)	0.0632	0.7660	0.1166	—
LR (RW)	0.7	0.1857	0.2935	0,007
XGBoost (RW)	0.4916	0.1535	0.2340	0,080
LightGBM (RW)	0.5219	0.2196	0.3091	0,004

Table 4. Overall results

5.1.1. SVM (Baseline)

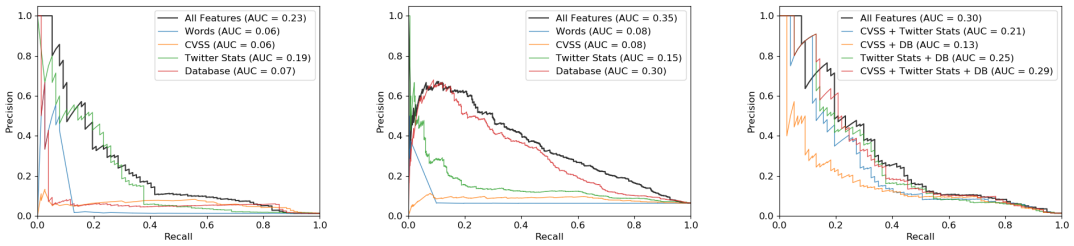
Fig. 2 shows the difference between the Precision-recall curves for PoC and real-world exploits. The first characteristic we would like to highlight is how the classifier performs better with PoC data, probably because vulnerabilities with exploits published on the ExploitDB are likely to have references on their description on NVD or OSDB. Therefore the subset of features extracted from public vulnerabilities databases plays a big part in the overall performance for PoC (this holds for all other algorithms).

Secondly, it is possible to notice how, in real-world scenarios, the Twitter Statistics subset has an essential contribution to the overall result. As will be demonstrated here, this characteristic depends on the algorithm. However, our results indicate that the "who said it" might be a more relevant question than "what was said" to determine if a tweet indicates an exploited vulnerability.

Another characteristic is how the CVSS tends to be a conservative indicator. In other words, using just that score leads to labeling most of the vulnerability as *exploited*, hence the low precision and high recall. On the other hand, Words subset tends to be the other way around: higher precision but very low recall. This behavior reflects how certain words on the BoW representation, such as "exploit" or "beware", are particularly efficient in detecting exploits but are generally related to only a subset of the exploits.

5.1.2. Logistic Regression (LG)

Considering the precision, the Logistic Regression (LG) was the best performing algorithm for the RW scenario. When looking at F-score, the Logistic Regression was the



(a) SVM - real-world exploits **(b) SVM - PoC exploits** **(c) LG - real-world exploits**

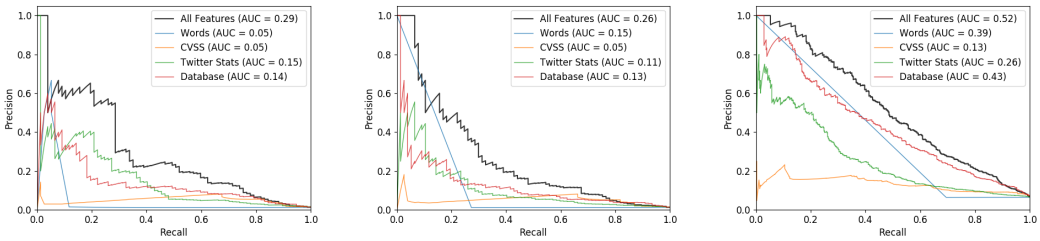
Figure 2. Precision-recall curves for SVM and Logistic Regression

second best. Fig. 2 c) shows the PR curves for our test. In it, we display different combinations of features. We can highlight how Twitter statistics was useful for all the tests it was included, once again. It is worth mentioning that the model was one of the fastest to train and test.

5.1.3. XGBoost and LightGBM

Both of the ensembles tested had good results, but only one of them, the LightGBM, had a P-value of less than 0.05. Besides, the XGBoost demonstrated the second-longest execution time, ahead of only the SVM. The LightGBM, on the other hand, was the fastest algorithm to execute in our tests. Figure 3 shows the PR curves for both algorithms.

In conclusion, the LightGBM was the best performing algorithm, considering the F-score, during our initial tests by a small margin, followed by the Logistic Regression. Throughout our other tests, the LightGBM was able to improve more than the Logistic Regression, as will be shown in Section 5.3. Finally, we would like to point out how the CVSS subset of features plays a more significant part when testing with PoC exploits. For RW exploits, the Twitter statistics and the NVD data become more relevant.



(a) PR Curves for the XGBoost with RW exploits **(b) PR Curves for the LightGBM with RW exploits** **(c) PR Curves for the LightGBM with PoC exploits**

Figure 3. Precision-recall curves for XGBoost and LightGBM

5.2. Multiple Ground Truth

In this experiment, we investigate how using different antivirus signatures as ground truth interferes with the classifier efficiency. In our research, we were able to find public databases with lists and descriptions of signatures from the following vendors: Avast,

	2014	2015	2016	2017	2018	Total
Symantec Tweeted	**77***	**43**	**29**	**58**	**131**	**267**
- Symantec Total	*90**	*261*	*247*	*219*	*364*	***1.228***
Avast Tweeted	**112**	**33**	**3**	**4**	**4**	**44**
- Avast Total	*123*	*220*	*97*	*21*	*8*	***346***
Other Tweeted	**11**	**5**	**2**	**14**	**7**	**28**
- Others Total	*14*	*19*	*3*	*15*	*7*	***45***
PoC Tweeted	**383**	**115**	**90**	**220**	**257**	**699**
- PoC Total	*823*	*721*	*549*	*1124*	*981*	***3.984***

* [Sabottke et al. 2015] dataset

Table 5. Number of exploited vulnerabilities mentioned by vendors

	SVM			LR			LightGBM		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Baseline	0.2244	0.8278	0.3531	0.5850	0.1094	0.1844	0.5458	0.2844	0.3740
ADASYN[I]	0.2271	0.8285	0.3565	0.1936	0.8403	0.3147	0.4640	0.3774	0.4162
SMOTE[I]	0.2391	0.8044	0.3686	0.2200	0.8181	0.3467	0.4867	0.3785	0.4258
AllKNN[II]	0.2271	0.8285	0.3565	0.4955	0.3007	0.3743	0.4956	0.5267	**0.5107**
RUS[II]	0.2330	0.8303	0.3639	0.1322	0.8921	0.2302	0.2212	0.8642	0.3523

[I]Over-sampling technique
[II]Under-sampling technique

Table 6. Results for class balancing for RW exploits

ESET, Symantec, and Trend Micro. We then developed web crawlers to collect this information, searched for CVEs mentions, and created an expanded ground truth that will be shared with the general public.

To our knowledge, all previous works use only Symantec to tell if a vulnerability has been exploited in the wild. Our findings indicate that, from 2015 to 2018, at least 248 of the 1,338 real-world exploited vulnerabilities are not mentioned by Symantec. Furthermore, considering years before 2015, Symantec's database misses 518 real-world exploited vulnerabilities in a total of 2,655 mentioned in signatures descriptions. Notice that this does not necessarily mean their antivirus has no signature for those exploits, but it indicates that no reference was made in the description. Table 5 shows the amount of exploited vulnerabilities mentioned by each vendor from 2015 to 2018. ESET and Trend Micro contain limited information compared to Avast and Symantec, so we combined their numbers and labeled them "Other". Also, notice how the quantities change drastically over time, we will discuss that at the end of this section.

One of our concerns about using public signature descriptions was to avoid vendors that would only include PoC information and count it as "exploit detection". To verify if we were getting new information, for each vendor, we compared the lists of exploited vulnerabilities to our list of PoC. We found that a significant amount of exploited vulnerabilities were nor mentioned by Symantec or EDB, so it is reasonable to consider them as real-world exploits. Figure 4 shows the intersection of our new list of exploited vulnerabilities with Symantec's and EDB's (from 2015 to 2018).

To check how the new ground truth affects our classifier, we conducted a series of tests where the model was trained using labels from a single vendor but tested with the combination of all vendors (the combined ground truth). Table 7 contains the precision, recall, and F-score for each test with our best performing algorithm, the LightGBM.

The results suggest that in 2014, [Sabottke et al. 2015] could have achieved equal

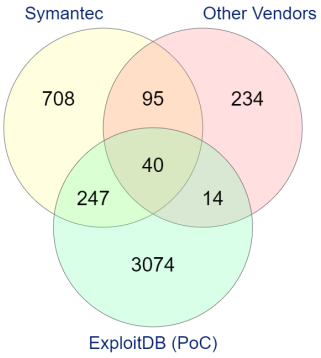


Figure 4. The intersection of the lists of exploited vulnerabilities

	Precision	Recall	F-score	P-value
Symantec	0.5605	0.1233	0.2021	–
Avast	0.5242	0.2209	0.3109	0.086
Sym + Avast	0.5238	0.2795	0.3645	0.046
Sym + Avast + Others	0.5458	0.2844	0.3740	0.032

Table 7. Results with combined ground truth

or better results using Avast's database instead of Symantec's (the P-value larger than 0.05 means the improvement is not significant enough for us to declare it better, despite the F-score). Most importantly, we conclude that using information from a single vendor can bias the model toward some vulnerabilities and lead to non-optimal performance on real-world scenarios. As we will explain in Section 5.3, the gains of a combined ground truth can even be emphasized when using a class balancing algorithm. Unfortunately, as shown in Table 5, since 2015, Avast, has gradually decreased the volume of information about its signatures.

From this experiment, we conclude that using multiple ground truth sources is crucial for any machine learning application dealing with exploit detectors. Despite that, we see less of this information available in recent years, which may indicate less effective classifiers on future works.

5.3. Class-balancing

In another experiment, we compared methods for dealing with class imbalance. The severe imbalance in our application motivated this test, as shown in Table 3. In this test, we ran class balancing methods with all four classification algorithms, and once again, the LighGBM outperformed the others. We used a Python library called "imbalanced-learn" and tested several algorithms of both undersampling and oversampling. In our tests, we used the combined ground truth from Section 5.2 on real-world exploits and executed these algorithms for the training set from each fold of the 10-fold cross-validation. Table 6 shows the results for the best-performing ones.

As we can see, the SVM and the LightGBM behave differently with the balancing algorithms. While the baseline could only get similar F-score values and no statistically significant improvement, the LightGBM showed a considerable increase in performance, with a P-value of 0.001, when used with the AllKNN, an under-sampling technique. Other algorithms were not able to obtain statistically different results, although some achieved superior F-score.

In general, the AllKNN outperformed the original experiments for all classification algorithms, both for PoC and real-world exploits. It is worth mentioning that when using only Symantec's ground truth, improvements were more modest, highlighting the importance of using multiple sources for ground truth.

5.4. Updated Data and Time Window Sizes

With our last test, we wanted two things. First, to understand if the results obtained using data from 2014 would remain with an updated dataset. Second, to answer if training with data from a more extended period would influence the results. To do that, we first trained and tested the model separating our data by year, from 2015 to 2018. We then ran a series of experiments where the model trained with different time windows from years before 2018 but tested with data from 2018. More specifically, the model trained with the following groups of years: 2017, then from 2016 to 2017, and finally, from 2015 to 2017. Here, we would like to evaluate whether adding more training data would positive influence the performance of the model.In this experiment, we used the LightGBM with the combined ground truth. The results are shown in Table 8 and Table 9.

Year	Precision	Recall	F-score
2015	0.6048	0.6282	0.6163
2016	0.2089	0.2333	0.2204
2017	0.5419	0.5681	0.5547
2018	0.6999	0.5529	0.6178

Table 8. Results for updated data - Training and testing using a one-year window

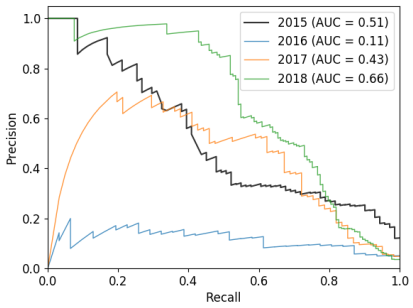


Figure 5. Precision-recall curves for each year on RW exploits (2015 - 2018)

Train	Test	Precision	Recall	F-score
2018 (baseline)	2018	0.6999	0.5529	0.6178
2017	2018	0.3333	0.1654	0.2211
2016 2017	2018	0.3051	0.1353	0.1875
2015 2016 2017	2018	0.3208	0.1278	0.1828

Table 9. Results for training with different time windows

We can see from Table 9 that results remain reasonably constant throughout the years, except for 2016, probably due to the small number of exploited vulnerabilities mentioned on Twitter (see Table 5). Moreover, as mentioned in Section 5.2, the contribution of Avast and other antiviruses to our ground truth diminished in recent years, making the model more dependent on Symantec as its only source of information.

When training with different time windows, our results suggest that there may be some relation between CVEs from the same year that allows better learning from the model. When training with data from past years, F-score remained around 0.2. From our perspective, this difference was not expected since all data used for the model's features were equally available for all of the years. To check if information learned by the model was getting old overtime or if it was missing new information, we ran tests with multiple combinations of time windows (e.g., we would train just with 2015 or with 2015 and 2016, then with 2016 and 2017; we also tested with multiple years, and even trained with years ahead of the testing one). In all cases, the performance was significantly lower than the single year approach. We believe this behavior can relate to the fact that some vulnerabilities are disclosed together and be part of the same malware issue. The WannaCry ransomware outbreak, for example, was related to six different CVEs (CVE-2017-0143, CVE-2017-0144, CVE-2017-0145, CVE-2017-0146, CVE-2017-0147, and CVE-2017-0148), all with a fairly similar description, CVSS, and affecting the same products. If that is the case, maybe it is necessary to detect these groups of CVEs and treat them as a single vulnerability. To conclude, we found no empirical evidence that using more than one year of data can benefit a model that uses a temporal batch split.

5.5. Threats to validity

Our experiments were conducted with tweets acquired using a tool called GetOldTweets3 [Mottl 2018], which can search for Twitter messages regardless of their posting time but cannot obtain deleted tweets or tweets from deleted users. In contrast, if we were using the Twitter Stream API, messages would be obtained and stored in our database when posted. This limitation leads to a diminished tweet volume. We believe our contributions can apply for batch training with or without a temporal split, but it is necessary to mention that time intermixing can be a limitation on our tests using single year data. In our method, we also discarded tweets referencing older vulnerabilities (for example, we only consider CVEs disclosed in 2018 when collecting Twitter mentions in 2018), which may prevent the system from detecting new exploits on old vulnerabilities.

6. Conclusion and Future Work

In this paper, we explored several aspects of Twitter-based exploit detectors. We have compared the performance of four commonly used classification algorithms, conducted tests with different data sources for ground truth to real-world exploits, applied balancing algorithms, and trained and tested our classifier with different time windows.

By selecting a suitable algorithm, managing class imbalance issues, and adding new ground truth, we were able to outperform the baseline and evaluate the performance of exploit detectors using updated data. Some of our results raised questions that we would like to tackle in future work, some of them involve: monitoring the NVD data to check which of our features is available when a CVE is published; understanding how related CVEs interfere on our classifier and if that is the real cause of inferior performance using time windows; enhance the tweet searching by using keywords other than "CVE". We also want to experiment with different feature selection and text representations.

References

Bilge, L. and Dumitras, T. (2012). Before we knew it: An empirical study of zero-day attacks in the real world. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 833–844.

Bozorgi, M., Saul, L., Savage, S., and Voelker, G. (2010). 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.

Bullough, B. L., Yanchenko, A. K., Smith, C. L., and Zipkin, J. R. (2017). Predicting exploitation of disclosed software vulnerabilities using open-source data. *IWSPA 2017 - Proceedings of the 3rd ACM International Workshop on Security and Privacy Analytics, co-located with CODASPY 2017*, pages 45–53.

Chen, H., Liu, R., Park, N., and Subrahmanian, V. (2019). 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.

imbalanced-learn API documentation (2019). imbalanced-learn api. `https://imbalanced-learn.readthedocs.io/en/stable/api.html`.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Mottl, D. (2018). Getoldtweets3. `https://pypi.org/project/GetOldTweets3`.

Nayak, K., Marino, D., Efstathopoulos, P., and Dumitraş, T. (2014). Some vulnerabilities are different than others. In *International Workshop on Recent Advances in Intrusion Detection*, pages 426–446. Springer.

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. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Queiroz, A., Keegan, B., and Mtenzi, F. (2017). Predicting software vulnerability using security discussion in social media. *European Conference on Information Warfare and Security, ECCWS*, pages 628–634.

Sabottke, C., Suciu, O., and Dumitras, T. (2015). Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *USENIX Security Symposium*, pages 1041–1056.

Shrestha, P., Sathanur, A., Maharjan, S., Saldanha, E., Arendt, D., and Volkova, S. (2020). Multiple social platforms reveal actionable signals for software vulnerability awareness: A study of github, twitter and reddit. *PLOS ONE*, 15(3):1–28.

Younis, A. A. and Malaiya, Y. K. (2015). Comparing and evaluating cvss base metrics and microsoft rating system. In *2015 IEEE International Conference on Software Quality, Reliability and Security*, pages 252–261.