Twitter-Enhanced Android Malware Detection

Jordan DeLoach and Doina Caragea
Department of Computer Science
Kansas State University
{jdeloach,dcaragea}@ksu.edu

***Abstract*—In data-driven Android malware detection, large numbers of both malicious and benign apps are used to train machine learning classifiers to detect malware. Existing approaches have nearly exclusively focused on app contents to extract features for classification. We seek to understand if auxiliary data, specifically Twitter data, can be used to improve the performance of existing approaches for Android malware detection. Throughout the course of our research, we collected over 50 million tweets potentially related to Android apps. We propose to link tweets with apps using approaches inspired from the standard vector space model, and subsequently study the usefulness of the linked tweets in malware detection. We find that Twitter data accurately linked to apps through HTTP links can be used to improve the machine learning classifier performance across a variety of common malware detection classifiers. However, classification experiments with Twitter data automatically linked to apps reveal the need for future work on more robust linking approaches.**

I. Introduction

Mobile phones and their corresponding applications have proliferated in the past ten years. Alongside the creation of a variety of benign mobile apps, including revolutionary apps having impacts in areas from health to finance and communication, the number of malicious apps has also increased significantly. The inclusion of malicious apps into an app store allows them to be downloaded and spread to unknowing and unsuspecting users' mobile devices across the globe. These malicious apps can then infect their targets, having undesirable impacts ranging from personal data leakage to financial losses. In 2016 alone, over 4,000 apps were removed from the Google Play Store, likely due to their maliciousness [16]. Kapersky [12] found that many malicious apps had been downloaded more than 100,000 times, with one malicious Pokemon Go guidebook app being downloaded more than 500,000 times in the Google Play Store. To combat this increase in the prevalence of malware, various methods have been proposed to detect Android malware at the app store level.

Many of these approaches leverage machine learning by extracting "cheap features" from an app's binary [19], [4]. These "cheap features," often as simple as just searching the decompressed text for the presence or non-presence of a certain API or permission, are computationally very inexpensive to extract. The goal of such an approach is to leverage a large number of cheap features, sometimes upwards of hundreds of thousands, as in the case of Drebin [4]. More robust methods employ data and control flows, or API dependency graphs, as a means to get better perspective and richer, more predictive features from app binaries [24], [5]. Even then, challenges like obfuscation remain a problem for effectively determining the benign or malicious nature of an app based solely upon the contents of the app binary [22].

Regardless of the depth to which approaches dive within the app binary, the contents of the binary are used, almost exclusively, as a basis for malware detection. Outside of purely app binary-based approaches, one work studied leveraging user reviews to predict security-related behaviors [13], and another used app descriptions to infer description-to-permission fidelity [17]. Each of these attempted to link certain user or app-provided text with certain behaviors of apps. Both works focused on predicting their respective links, but did not explore how this information could affect the performance of an automated malware detection solution.

The aim of this work is to explore whether additional information about an app can be useful in creating more accurate machine learning classifiers to detect Android malware. Namely, we investigate whether adding social media data, as represented by Twitter, into the machine learning process can improve existing approaches for Android malware detection.

Intuitively, there are two main ways in which Twitter data can help. First, the text of a genuine tweet referring to a particular app may contain words that are informative with respect to the benign or malicious nature of the app. In particular, the text of a tweet may express a positive sentiment, possibly associated with a benign app, or a negative sentiment associated with a malicious app. Second, metadata about a tweet referring to a particular app may also contain useful information for determining if the app is malicious or benign. The metadata contains information about the tweet (e.g., number of favorites or retweets), and also about the user who posted the tweet (e.g., the number of followers). Together, these two pieces of information are indicative of spam tweets/users, which in turn is indicative of the corresponding app being associated with malicious behavior. Specifically, an app with tweets from users with small Twitter footprints and no genuine favorable peer reviews (in terms of favorites and retweets) has a higher probability of being associated with malicious behavior, as compared to an an app with tweets from users who have common peer following and retweet patterns.

In Figure 1, we provide examples of tweets referencing a benign app (top), and a malicious app (bottom), respectively. Corresponding to the tweets in Figure 1, the percentiles of a few key tweet metrics, extracted from the metadata associated with the tweets, are shown the Table I. These percentiles emphasize the differences that exist in the types of users that

Fig. 1. Tweet examples. (Top) Tweet that references a benign app. (Bottom) Tweet that references a malicious app.

tweet about apps, *i.e.* tweets about malicious apps tend to come from users with less credibility and validity (i.e., spam users), as compared to those of users who tweet about benign apps (i.e., genuine users).

TABLE I
PERCENTILES OF SELECTED USER METRICS, EXTRACTED FROM TWEET METADATA, FOR A TWEET REFERENCING A MALICIOUS APP, VERSUS A TWEET REFERENCING A BENIGN APP. AS CAN BE SEEN, THE VALUES ARE HIGHER FOR THE BENIGN APP AS COMPARED TO THE MALICIOUS APP.

Class	Statuses	Followers	Friends	Favorites
Malicious	0.15	0.30	0.38	0.56
Benign	0.51	0.98	0.66	0.68

Our main goal is to study the usefulness of Twitter data when detecting Android malware using machine learning techniques. While this problem is obviously of practical interest in Android security, linking tweets to apps is a challenging problem in itself, unless the tweet contains an explicit link to the corresponding app, and can be directly linked that way. As a first step towards addressing this challenge, we propose to use approaches inspired from the standard vector space model (VSM) to match the text that describes an app, usually available from the app store, with the text of tweets crawled from Twitter using security-related keywords. For apps that are linked to tweets, we augment the app features extracted from binaries with features extracted from tweet text or tweet metadata, and learn classifiers for malware detection.

To summarize, the contributions of our work are as follows:

- We use security keywords to crawl a large Twitter dataset that can be used to augment existing Android malware detection datasets with social media data.
- We identify tweets that can be linked to apps based on HTTP links, and use them to augment an existing Android malware detection feature representation of an app with features extracted from the tweet text or the metadata associated with the tweets, respectively.
- We train machine learning classifiers on the subset of apps with HTTP-linked tweets, and show the usefulness of Twitter data for detecting Android malware.
- To account for tweets that do not have direct HTTP links to apps, we propose and compare two simple VSM approaches that achieve the linking by matching the tweet text with the text that describes an app.

A. Overview of the Proposed Approach

As opposed to prior work, our proposed solution aims to combine knowledge that may be learned about an app from its binary, with information that users online may be posting about the app, to create a more verbose representation for discriminating between malicious and benign behaviors. In a sense, this allows us to not only examine what an app does (as represented by the code), but also what users say that the app does (as represented by tweets about the app). This two-pronged approach, we speculate, may prove effective at improving existing techniques for malware detection.

To utilize social media data to augment existing malware detection approaches, two sets of data are needed: a set of apps and a set of tweets about the apps. While collection of both Android app binaries and tweets is relatively trivial, the concept of putting a specific app and a specific tweet together to create a common feature vector representation, while simple, it is challenging, and represents the main novelty presented in this paper.

We show a pipeline representation of our solution in Figure 2. We start by assembling a set of Android apps, including app binaries and app descriptions from the app store, and a set of tweets (text and metadata) that potentially reference Android apps in relation to security issues. Details about the assembly of the datasets can be found in Section III-A.

Next, we link tweets to apps by directly using HTTP links present in tweets, or by matching tweet text to app description using approaches inspired from the vector space model. This is possible as tweets and app descriptions often share common terms, as can be seen from Table II, which shows an example of an Android app description as represented in the Google Play Store, alongside some tweets that are likely talking about it. The approaches for linking an app with potentially related tweets are presented in Section II. Each approach presents a particular tradeoff between confidence in a given link (precision), and the general applicability of the approach over a wide corpus of tweets (recall).

Tweets and apps that are linked together are subsequently provided to a module that extracts features from app binaries and from Twitter data, as explained in Section III-B1. The features are further used to learn machine learning classifiers as described in Section III-B2. The details about the metrics used to evaluate our approaches are provided in Section III-C, and the results are presented in Section IV.

II. TWEET-TO-APP LINKING APPROACHES

Linking methods are perhaps the most critical element in successfully leveraging social media data to aid existing malware detection approaches. While it may be intuitive for a human to understand that a tweet is talking about the "YouTube" app and know exactly which app that is, for a computer, a quick scan of the text "youtube" in the Google Play Store returns many more results than the official "YouTube" app built by Google. Produced by Google alone there is "YouTube," "YouTube for Android TV," "YouTube for Google TV," "YouTube Kids," "YouTube Gaming," "YouTube Music,"

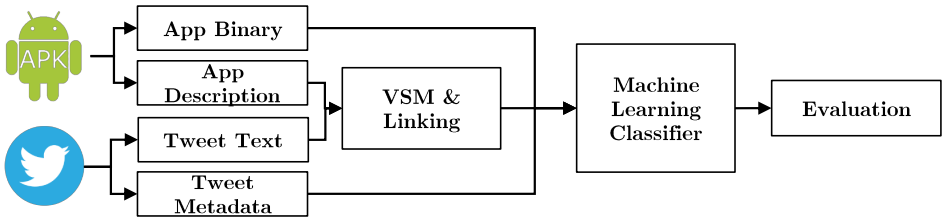


Fig. 2. Pipeline Overview

TABLE II
TEXT SIMILARITY BETWEEN THE DESCRIPTION OF AN APP AND TWEETS THAT REFER TO THAT APP. WORDS THAT APPEAR BOTH IN THE APP DESCRIPTION AND IN THE TWEETS ARE HIGHLIGHTED WITH DIFFERENT COLORS.

App Description	Tweet
Millions of users have made Ookla ***Speedtest*** the #1 ***app*** for ***testing*** ***Internet*** speeds, and it's trusted daily by professionals throughout the industry! - Discover your ***Download***, Upload and Ping - Real-time graphs show ***connection*** consistency - Troubleshoot or verify the speed you were promised - Track past tests with detailed reporting - Easily share your results	RT @Speedtest: One-tap ***internet*** ***connection*** ***testing*** on the go... ***download*** Speedtest for Android today https://t.co/HutGDx5Ag3 https://t.co/b
	RT @Speedtest: Get the free ***Speedtest*** ***app*** for Android today. https://t.co/HutGDwNYRt https://t.co/ShAT8xpDKW
	@Sebianoti Yup! Get the ***app*** here: https://t.co/HutGDwNYRt One-tap ***internet*** ***connection*** testing on the go... ***download*** ***Speedtest*** for Android today https://t.co/HutGDx5Ag3

"YouTube Creator Studio," and "YouTube VR," let alone the many more third-party apps using "YouTube" branding. Thus, the process of attempting to match a tweet with *exactly* one Android app represents a challenging problem.

Very precise approaches allow us to be highly certain of the exact app being discussed by a tweet. However, if the approaches lack broad applicability, there may be only a relatively small number of tweets that would meet such stringent linking requirements. As such, we present three approaches, each having varying levels of effectiveness and applicability. The first approach is very precise, but not widely applicable, while the other two approaches are less precise, but have higher recall, as they can be applied to any pair of tweet and app descriptions.

A. HTTP-based Linking

Our first tweet-to-app linking approach relies on the existence, in the tweet, of a link to the app in the Google Play Store. Such links to apps can be tokenized in a way that allows the extraction of the exact APP ID. For instance, the page showing the Speedtest.net app, a popular app to test internet speeds, is accessible at: *http://play.google.../details?id=* ***org.zwanoo.android.speedtest***. On May 16, 2017, Twitter user @Speedtest tweeted: "*Download Speedtest for Android and test your internet connection on the go! https://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest*."
This tweet is representative of one way in which Twitter users talk about apps, with explicit links. In this instance, we are very certain that the user and the tweet text are indeed referring to the app represented by the APP ID **org.zwanoo.android.speedtest**. While this approach is highly precise, the number of tweets that use an exact HTTP link is low. Additionally, many tweets that include a Google Play Store link are automatically tweeted and may not be representative of authentic end-user tweets about an app.

B. Linking Using the TF-IDF Representation

To account for tweets that don't include an HTTP link, we propose a method inspired from the standard vector space model (VSM) that is widely used in information retrieval [14]. Assuming a vocabulary over a collection of documents, the VSM makes the assumption that terms in a document are independent. Each dimension in the vector space corresponds to one term. Documents, i.e., tweets and app descriptions, can be represented as vectors in this space. Furthermore, similarity between two documents is computed as the cosine of the angle between the vectors corresponding to the documents.

To identify relationships between tweets and apps, we consider the merged collection of apps, represented by the text of their descriptions in the app store, and tweets, represented using the actual text of the tweets. We construct the vocabulary for our collection of app descriptions and tweets using common text processing techniques, such as stemming using the Porter stemmer, stop-word removal, and replacing non-semantic terms like usernames and links with generalized strings. The Porter stemmer reduces words that have a common stem, e.g. *practically* and *practical*, down to the common stem, such that small grammatical differences do not impact similarity. Stop-word removal is a simple technique that removes around 500 non-descriptive terms such as "the," "a" and "an." Finally, we replace usernames and web links with placeholder texts, such that we can capture more of the meaning of the text, and less of the specifics of the exact username or the exact HTTP link. In addition, in preliminary experimentation, the best performance was obtained when utilizing tri-grams as terms, as opposed to uni-grams, or single words. For a text sequence "a b c d e," the uni-grams are "a,"

"b," "c," "d," and "e," while the tri-grams include "a b c," "b c d," and "c d e." Intuitively, tri-grams allow us to better capture the meaning and common grammar used to describe an app, as compared to just constructing a vector space model based upon uni-grams. Additionally, we require a term to be present in at least two different documents for the word to be included in the vocabulary.

While there are various approaches for computing the vector corresponding to the text in an app description or in a tweet, given a vocabulary, we utilize Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is a standard way to capture the comparative importance of a term in a text as measured by its term frequency (TF), while also regularizing the value based upon how common or uncommon the term is in the rest of the collection via the Inverse Document Frequency (IDF). Both app descriptions and tweets are represented as vectors, with the TF-IDF of every vocabulary term (tri-gram) in the respective text. The resulting vectors are further used to determine the cosine similarity between a tweet and an app.

Every app description and tweet are similar to some degree if they share at least one common term. However, we focus on pairs of app descriptions and tweets that are highly similar. Thresholding on a sorted list of scaled similarity scores allows us to yield different link sets that we are variably confident in. Through this thresholding, we can intentionally balance between having a widely-applicable approach, by allowing less certain links (based on smaller similarity scores), and a more precise approach, by allowing only higher confidence links (with higher similarity scores).

C. Linking Using a Reduced WMF Representation

Twitter text is quite distinct from the text of other domains, in the sense that its corresponding vectors are exceptionally sparse. With the 140-character requirement, users are forced to be quite terse in their statements. While the vocabulary used across many tweets can be just as large as in other domains, the number of terms used in a specific tweet is inherently much lower, lending towards an extremely sparse vector. Dimensionality reduction techniques allow high-dimensional vector spaces to be distilled down to a smaller number of components or concepts, ideally creating a more equitable means for comparison of strings in a short-text domain.

We should note that the common belief that if a word isn't in the text, it isn't relevant, may be invalid in short-text domains. The insight that an omitted word shouldn't mean that the implied concept is disregarded, is inspired from recommender systems: the absence of a rating of a movie in relation to a user does not imply that the user dislikes the movie, just that the user has not seen the movie yet. Likewise, just as other similar movies can be used to predict the potential rating of that movie for that user, the presence of a concept in the text can be predicted or implied by the other terms in the domain.

Using this insight, we leverage a Weighted Matrix Factorization (WMF) approach as a means for dimensionality reduction [21]. A key benefit of WMF over other common dimensionality reduction methods applied in the text domains, such as Principal Components Analysis [11] or Singular Value Decomposition [7], is that a non-existent word is given a small, tunable value in the original matrix to be factorized. This allows us to tune for how forgiving the algorithm should be on the absence of a term when computing the weight for a certain concept. We use the Alternating Least Squares Weighted Regularization (ALS-WR) implementation for matrix factorization [25], available in Spark (originally for collaborative filtering with implicit feedback). This method has three tunable parameters: the number of latent components, k; α, a multiplier used to vary the weight given to ratings, in our case terms, so that unseen ratings/terms get a small non-zero weight; and λ, a regularization parameter, which is standard in ALS optimization.

III. Experimental Setup

A. Datasets

We use two large base datasets in our work. The first dataset is an Android app dataset and the second is a tweet dataset. Furthermore, to evaluate our proposed app-to-tweet linking approaches, we use the HTTP linking approach to derive a ground truth dataset.

1) Android App Dataset: Our base Android app dataset is a hybrid of both the PlayDrone [23] and AndroZoo [3] datasets, supplemented with additional malware from industry collaborators. In 2014, the PlayDrone work made most of the free apps on the Android market available to researchers. The AndroZoo dataset is newer and makes over 5.3 million apps from a wide range of app stores available to researchers. Starting with these apps, we then take the MD5 checksums of the respective apps and submit them to VirusTotal [1], a multiple anti-virus scanning solution, that allows us to garner multiple expert (anti-virus) opinions for the class label of each app. While past works employing a similar methodology use a standard of around 10 out of the approximately 50 Android anti-virus solutions available [19], we utilize a lower threshold of 3 to determine an app as either malicious or benign. We do this for two reasons: first, it gives us more potentially malicious apps in our dataset. Second, when we are using Twitter to aid malware detection, those apps may often have not been widely detected as malicious yet, hence, a lower threshold will allow us to work with a larger set of apps, including apps that are possibly, but not certainly malicious. Overall, our base app set includes 1,385,058 apps, of which 158,108 were considered malicious (based on at least 3 scanners), and 939,608 were benign. However, in our experiments, we do not use all of these apps, we use only those for which we could create a link to a tweet, as described below.

2) Twitter Dataset: For our base tweet dataset, we connected to the Twitter Firehose API and listened for keywords such as "Android," "app," "mobile," and "malware," almost the entire months of November and December 2016. While that provided a substantial number of tweets, we further sampled down and kept tweets that contained keywords from the apps in our app datasets (developer names, application names, etc.), which we aimed to link tweets to, as well as any tweets that

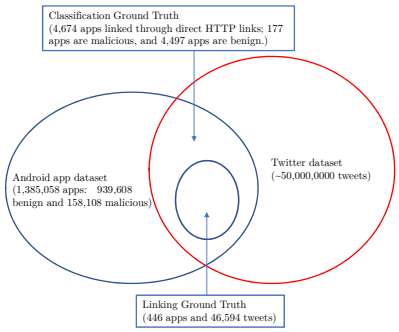
contained a play.google.com link. This two-month collection yielded nearly 50 million tweets that could be about a specific app. Additionally, there were many tweets not about a specific app, and some that weren't even about Android or malware.

3) Ground Truth Datasets: We used the HTTP-based linking to create ground truth datasets for evaluating the classification improvements gained by using Twitter data, and also for evaluating our proposed linking approaches. Our Twitter dataset contains links to over 26,000 different Play Store apps, many of them being paid apps. However, there are only 4,674 apps for which we had both the app binary in our dataset, as well as a definitive label from VirusTotal (as per Section III-A1). Of those 4,674 apps, 177 apps are malicious, and 4,497 apps are benign. We use this dataset for the classification experiment in Section IV-B. While this dataset is highly imbalanced in terms of benign to malware ratio, it closely matches the distribution observed in the real world, as emphasized in [19].

To construct the ground truth dataset for evaluating the VSM linking approaches, we further filtered down this dataset to only apps for which we have the app store description, as the description is required to calculate the cosine similarity scores. This decreases the size of our dataset to 4,075 apps and 346,796 tweets. Furthermore, we also filter down our subset such that, after standard text processing techniques, at least 10 terms exist in both the tweet(s) and the app description for a given app. We do this to ensure that a vector has sufficient non-zero components when attempting to map the similarities between the corresponding strings. With fewer non-zero components, we see performance decline, e.g., if an app and tweet only had three terms, there is a chance for those terms to exactly match and yield a high confidence score. In reality, matching strings based on three terms is not a strong indicator of similarity. After the aforementioned filters, we have a set of 46,594 tweets that link to 446 apps that serve as our ground truth for the experiment in Section IV-A.

Our datasets, including the Android app dataset, Twitter datasets and ground truth datasets (ground truth for classification and ground truth for linking) are summarized in Figure 3. While the Android dataset has been used in prior work, e.g. [19], there are no other datasets in the literature that can be used to validate tweet-to-app linking approaches, or approaches that make use of Twitter data to enhance Android malware detection.

Fig. 3. Data Sets: Android app dataset, Twitter dataset, ground truth for classification and ground truth for linking, respectively.



B. Classification Algorithms and Features Used

Given a training dataset, there are two main components that determine a classification approach: the set of features used to represent the instances in the training set (in our case, Android apps), and the algorithm used to learn a classifier. Below, we describe the features and the classification algorithms that we used to evaluate the usefulness of Twitter data for Android malware detection.

1) Classification Features: Our classification features used to represent an app can be divided in two categories: features extracted from app binary, and features extracted from the tweets linked to an app.

For the features extracted from an app binary, we use a set of 471 features that have been defined previously [19], and known to give state-of-the-art results in Android malware detection. The features mainly relate to the different APIs an app may use that are associated with either malicious or benign behavior, in addition to the standard permissions and intents present in the Android operating system. For those 471 features, we simply check for their presence or non-presence in the app binary and record either a "1" or a "0" in the feature vector.

Features that we extract from the linked tweets can be divided into two categories: features constructed from the text of the linked tweets, which we refer to as "text," and features constructed from the metadata associated with the linked tweets, which we refer to as "metrics." The text features correspond to the vectors used in the linking approaches, specifically TF-IDF features and WMF features. Also based on text, we construct an additional feature corresponding to the sentiment of the tweet as identified by the Stanford NLP library [15], which we refer to as "sentiment."

The set of metric features includes 7 features, available as part of the tweet metadata:

- *followersCount* - number of followers of the tweet's author
- *friendsCount* - number of users the tweet's author follows
- *favouritesCount* - number of tweets the tweet's author has favorited
- *statusesCount* - number of statuses for the tweet's author
- *linksCount* - number of HTTP links in the tweet
- *mediaCount* - number of media entities (*photos*) in the tweet
- *isReply* - 1 if the tweet is in response to another tweet, 0 otherwise

To construct a single feature vector, irrespective of the number of tweets that may be linked to an app, we experimented with several different techniques to combine values from multiple tweets into one feature vector, including just taking one of the tweets at random, summing up the values coming from different tweets, averaging the values and taking the median. We found that averaging the values garnered the

best performance as it summarized the perspective of multiple tweets, but didn't sway feature values due to the number of tweets, as summing would do.

2) Classification Algorithms: The classification algorithms that were used to separate apps into malicious or benign classes include Naïve Bayes, Logistic Regression, and Support Vector Machines (SVM). Each of these algorithms is relatively standard in the Android malware detection field and is commonly used in many other similar works [19], [4], [24], [8]. Naïve Bayes is a simple statistical technique that predicts the probability of an instance being in a particular class, conditioned upon the features, assumed to be independent from each other given the class. Logistic Regression is a method able to take a variety of independent features to predict a dependent class variable, based upon the logistic function. SVM is a state-of-the-art max-margin approach that attempts to find support vectors that create a separating hyperplane between the two classes, by maximizing the margin between the two classes. We should note that our goal is not to establish that one classification approach is better than another. Instead, we aim to show that Twitter data can be beneficial and aid several state-of-the-art classification approaches.

C. Evaluation Metrics

Our proposed solution to Android malware detection is based on machine learning classifiers that leverage tweets linked to apps, in addition to information extracted directly from app binaries. Therefore, our evaluation is two-fold: first, we evaluate our proposed linking approaches to understand how accurately they can match tweets to apps, and secondly we evaluate classifiers that leverage tweets to understand the usefulness of the information extracted from Twitter. The metrics used to evaluate the linking and classification approaches are described below.

1) Linking Evaluation: To evaluate the effectiveness of our proposed linking approaches based on the TF-IDF and WMF representations, respectively, we utilize a metric that can take into account our relative certainty of a given link. For instance, if we have a potential link between a tweet and an app, but we know we are highly uncertain of it, it should not be penalized for being incorrect as much as a link between an app and a tweet that we believed with high certainty. More specifically, we leverage the Precision-Recall Curve (PRC), as it allows us not only to evaluate the precision of our linking strategy versus the broader applicability, or recall, but also to do this at various confidence levels, by varying a threshold on the ranked list of similarity scores between app descriptions and tweets.

Different points on the Precision-Recall Curve correspond to different linking results. High similarity thresholds result in points on the curve that have high precision and, implicitly, links that are very likely to be correct, while many correct links are not identified. On the other hand, low similarity thresholds provide a very imprecise route, but garner many links and have a wide recall. Thus, the Precision-Recall Curve allows us to visualize the trade-off between a higher precision result set requiring a higher threshold, and that of a less precise, higher recall result set using a lower threshold.

2) Classification Evaluation: To evaluate our classification algorithms, we leverage the standard 5-fold cross-validation strategy to account for variability and get more reliable performance estimates. In general, the k-fold cross-validation allows a classifier to train over $k-1$ folds of the data, and then evaluate over the remaining test fold. This is done k times, so that all k folds are tested over. Through this, the data is trained and tested upon in k iterations, producing results that are generalizable to other datasets.

Additionally, as for the linking approaches, we use the Precision Recall Curve (PRC) to evaluate our classification approaches. The PRC can be represented in a numeric form by taking the area under the curve, yielding the area under Precision-Recall Curve (auPRC) metric, which has been shown to be a strong metric for evaluating classifiers on the highly imbalanced datasets (i.e., a relatively large number of benign apps as compared to the number of malicious apps) that exist in Android malware detection [19]. The auPRC values reported in our experiments are averaged over 5-folds.

IV. Results

In attempting to discern if Twitter data can aide in malware detection, we examine three points. First, we compare different approaches for linking tweets to apps and examine their comparative accuracies. Second, we examine, using HTTP-based links, if and how the additional data affects classification performance. Finally, we compare the classification performance using a variety of different types of features that can be extracted from a tweet to determine the best way to leverage the tweet-to-app links.

A. Tweet-to-App Linking Performance

In this section, we aim to explore how accurately we can link a given tweet to the app it is referencing. We compare the linking approaches based on the TF-IDF and WMF representations inspired from the vector space model. As described in Section III-A3, the HTTP linking approach is used to create the ground truth dataset for evaluating the TF-IDF and WMF linking approaches. In particular, to be able to link tweets to apps based on app descriptions, in this experiment we use the ground truth dataset that is filtered to ensure that an app description exits and that at least 10 terms exist in both the tweet(s) and the app description for a given app. As a reminder, this dataset contains 46,594 tweets that link to 446 apps. We should note that both TF-IDF and WMF linking approaches are "unsupervised" approaches, as they do not require any "training" data or prior knowledge, in addition to the app descriptions and tweets that need to be matched.

The Precision-Recall Curves for both TF-IDF and WMF linking approaches, constructed using the ground truth dataset, are shown in Figure 4. As can be seen, the WMF representation does have better performance at most thresholds, as well as better overall performance, as indicated by the area under the curve. An additional note not visualized in the graph is that

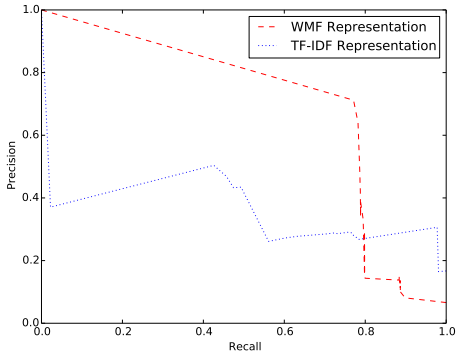


Fig. 4. Precision-Recall Curves corresponding to the TF-IDF and WMF linking approaches, evaluated on the ground truth dataset for linking. This ground truth dataset includes only the apps for which a description is available in the app store. Furthermore, only the apps for which the descriptions and the corresponding tweets have at least 10 words are included.

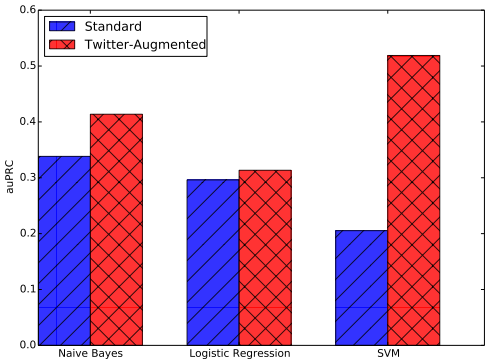


Fig. 5. Classification with Ground Truth Data, when two sets of features are used: standard features, meaning features extracted from app binaries, and Twitter augmented features, in particular metrics extracted from tweet metadata. Results are shown for three classification algorithms: Naïve Bayes, Logistic Regression and SVM.

WMF has better performance on a larger set of tweet-app pairs, as explained in what follows. Specifically, the performance is only evaluated on pairs for which the similarity score is greater than 0. When using the TF-IDF representation, many of the tweet-app pairs yielded a similarity of zero as the tweets had no terms in common with the apps. This was not an issue for the WMF representation, as in the reduced dimensionality of 800 (which we used), most tweet-app pairs had at least some terms in common and, therefore, most pairs had a non-zero similarity score.

We should note that there are no tunable parameters for the TF-IDF representation. For WMF, we used the ALS-WR [25] implementation available in Spark, which has three tunable parameters: the number of latent components k; the multiplier α, used to vary the weight given to terms; the regularization parameter λ. We experimented with several values for the tunable parameters. For latent components, we found the best performance with $k = 800$ dimensions. For α, we used a value of 50, and for λ, a value of 0.05. One interesting note is that while the default α in the Spark implementation is 1, we find a much higher value to be most effective in our case. This is intuitive as our α, a constant that controls the relative weight given to terms, including non-seen terms, is much higher than normal due to the fact that we are working with a TF-IDF matrix, with normalized values within the [0,1] interval, as opposed to an implicit feedback matrix in collaborative filtering, which captures the number of times a user interacted with an item.

In conclusion, we can say that the approach based on the WMF representation is the stronger of the two VSM-based approaches for linking tweets to apps. As it reduces the dimensionality to a select number of concepts and computes similarity on concepts, the WMF representation helps alleviate much of the term sparsity issues that are prevalent in Twitter texts. However, there is still room for improvement, as can be seen from the PRC corresponding to the WMF approach. That being said, WMF does provide a relatively accurate approach for matching tweets with apps.

B. *Twitter-Augmented Classification with Ground Truth*

We want to investigate how performance can be improved on the apps in the classification ground truth dataset (assembled based on HTTP linking) when using the additional linked Twitter data. The reason for experimenting only with the apps in the classification ground truth dataset first is that we are highly confident about the linking of tweets to apps in this case, and we can better judge the usefulness of the Twitter data based on the classification results with and without tweet features.

For this experiment, we use the classification ground truth dataset that contains 4,674 apps for which we have both the app binary as well as tweets with an HTTP link in reference to an app. The number of apps in this experiment is substantially higher than the number of apps used in the dataset for linking evaluation, as here we have no requirements about the content of the text (*e.g.*, requiring 10 or more terms), in either the app description or in the tweets.

Out of 4,674 apps used in this experiment, 177 were positive or malicious, and 4,497 were negative or benign. The imbalance between positives and negatives is in line with standard Android malware detection approaches and as seen in the real world [19].

In Figure 5, we show the results of classification for the three algorithms that we consider, both with feature augmentation based on tweet metrics (i.e., features extracted from tweet metadata) and without feature augmentation. In all instances, and to varying degrees, we find additional Twitter data to be beneficial. As a reminder, in this experiment, we know with high certainty that the tweets are linked to the correct app binary. In practice, this may not always be the case and mis-linked tweet data may likely impact classifier performance.

Also from Figure 5, we can see that different classification algorithms benefit from Twitter features to varying degrees, with the SVM algorithm benefiting the most, while it also benefits the least from the features extracted from app binaries

alone. We should note that we did not perform any parameter tuning for the SVM and Logistic Regression algorithms (Naïve Bayes does not have any tunable parameters in the first place). The reason for using Spark-provided default parameters for these algorithms is to save computation required to tune parameters, provided that our goal is to study the potential benefits provided by Twitter data to classification algorithms, as opposed to getting the best results for a particular algorithm or comparing the algorithms. The results in Figure 5 show that there is indeed some difference in the tweets/users who are tweeting about benign apps and those tweeting about malicious apps. The metrics contained in the metadata, such as followers and number of previous tweets, are indeed discriminative with regards to malware detection.

While overall classification performance may appear low in comparison to state-of-the-art methods like Drebin [4], there are several possible reasons for this, including having a much smaller dataset and having a real-world dataset class imbalance which has been previously shown to degrade performance [19]. However, we should note that the general concept of a feature vector expanded with social media data could be applied to Drebin and other approaches, provided Twitter data was available, to further improve results.

C. Informative Twitter Features

In our last exploration point, we want to study the best ways to leverage linked tweets to aid in the malware detection process. Tweets can be represented in many ways, from the text of the tweet, the associated metrics or metadata of the tweet, to derived representations like tweet sentiment. Thus, in this experiment, we compare four different means for utilizing a linked tweet: TF-IDF Representation, WMF Representation, Tweet Text Sentiment, and Associated Twitter Metrics.

For each of these four sets of features, we compare the resulting auPRC from ground truth classification with the same base datasets as in Section IV-B. To construct the feature vectors, we use our standard 471 binary features, augmented with just those features from the set being compared. The results are shown in Table III. As can be seen, the Metrics extracted from tweet metadata result in the best performance overall (which is why we only used Metrics in the previous experiment). SVM is the only algorithm that benefits almost equally from all sets of features, while Naïve Bayes benefits the most from the metrics, and the least from the TF-IDF featyres. We also considered combining the tweet metrics with another set of features but the performance did not improve significantly (results not shown due to space limitations).

Ranked Twitter Metrics: Given that the Twitter metadata turned out to provide the most effective set of features for augmenting binary-based detection, we also wanted to study which of the tweet metrics are the most informative. To do this, we computed the Mutual Information Gain [6] for each of the seven features. Mutual Information Gain allows one to study how the values of a specific feature correlate with the class variable, thereby showing how much each metric actually contributes to the discriminative power in a classifier. By far, *favouritesCount* and *friendsCount* are the most discriminative metrics of the seven compared; *statusesCount* and *followersCount* are about half as discriminative, but still substantially higher than the bottom three features of *linksCount*, *mediaCount*, and *isReply*.

TABLE III
COMPARISON OF DIFFERENT SETS OF FEATURES (TF-IDF, WMF, SENTIMENT, AND METRICS) IN TERMS OF AUPRC VALUES OBTAINED WHEN USING A PARTICULAR SET, IN ADDITION TO THE STANDARD BINARY FEATURES.

Approach	Naïve Bayes	Logistic Regression	SVM
TF-IDF	0.1851	0.2342	0.5189
WMF	0.2667	0.2665	0.5168
Sentiment	0.3470	0.3275	0.3672
Metrics	0.4137	0.3134	0.5189

D. Insights into Classification with Automatically Linked Data

All our classification experiments thus far have focused on data for which we have reliable classification ground truth, and strongly suggest that Twitter data can be a useful addition to traditional Android security data. Furthermore, the linking experiments suggest that our proposed approaches based on TF-IDF and WMF can be used to link tweets to apps, although the overall linking performance can be improved. To understand the effect of the linking noise on classification, and to evaluate the need for more robust linking approaches, we present the results of an experiment where we applied both the linking approach and the classification approach on top of it, for data where we do not have any ground truth (instead we automatically link tweets to apps using the WMF approach). We used a subset of our dataset of roughly 45,000 app descriptions and 1.6 million tweets for automated linking. Using WMF as optimized previously, we calculate the tweet-to-app description similarities, and return any links thresholded at 0.6 similarity or above. Using this threshold, we find 315 app-tweet links that can be used in the classification approach (out of which 26 apps are malicious and 289 apps are benign).

The results across the three classification algorithms are shown in Figure 6. We see mixed results in this experiment. Naïve Bayes performs better with the addition of Twitter data, however SVM and Logistic Regression do not. We speculate one reason for decreased performance in Figure 6 as opposed to Figure 5 is that the inaccuracy of some app-tweet links impairs classification (in addition to the smaller size of the dataset). While our previous experiments clearly show that Twitter data helps in the classification process, this experiment with automatically identified links suggests that further research into accurate linking may be necessary to make the approach more widely applicable for classification.

Regarding the dataset size, initially, yielding only 315 links from a dataset including over 1.6 million tweets appears low. Upon consideration of the fact that our Twitter and app datasets are subsampled, however, this number is in line with expectations. The 315 links correspond to nearly 1% of the 45,000 app descriptions. As we use 45,000 app descriptions, that number represents 50% of our valid app descriptions. As

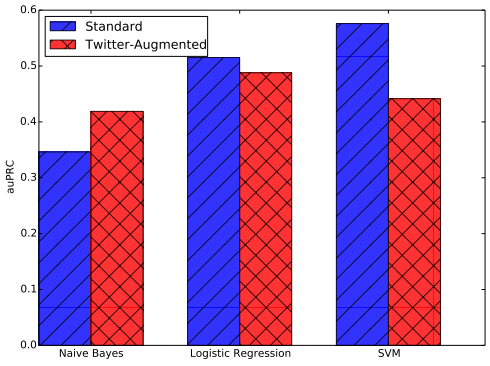


Fig. 6. Classification with Automatically Linked Data, when two sets of features are used: standard features, meaning features extracted from app binaries, and Twitter augmented features, in particular metrics extracted from tweet metadata. Results are shown for three classification algorithms: Naïve Bayes, Logistic Regression and SVM

we use 20% of the valid tweets, we would expect 10% of the valid links in our entire dataset to exist in our subsampled set. Given that not all apps have links/tweets about them, our results are in line with a hypothesis of 1 out of 10 apps have tweets about them, a.k.a. 10% of the 10%, or equivalently, the 1% links we yield.

V. Related Work

We group related works in several categories and review them in what follows.

A. Machine learning-based Android Malware Detection

Existing machine learning approaches to malware detection almost exclusively focus on detection based upon the contents of the app binary [22], [19], [24], [4]. The binary file, upon being decompressed, contains a slightly modified version of the code of the app, among other resources, including metadata about the app showing the exact permissions being requested by the app. From this binary file, common features used in the literature include app permissions, APIs used, and the presence of various other features associated with maliciousness.

More robust machine learning methods, borrowing concepts from static analysis, will take the collection of app contents one step further and construct meta-information on-top of the raw features, often in the form of API dependency or control flow graphs [24], [5]. However robust the app binary-based representations, previous works have failed to utilize information that may be present about an app on social media.

B. Usages of Alternative Data Sources for Android Security

AUTOREB [13] is a technique for automatically understanding review-to-behavior fidelity in Android apps, specifically correlating user reviews with security-related behaviors. While *AUTOREB* does not use these correlations for explicitly attempting to detect maliciousness, the results show a correlation between what a review says, and what an application does.

AUTOCOG [17] has a similar goal as *AUTOREB* except that it attempts to correlate app store descriptions of an app with the actual permissions used. Through linking the concepts, for instance the text "takes a photo" with a Camera permission, *AUTOCOG* could also then be used to realize when permissions are requested but not justified.

FeatureSmith [26] uses scholarly knowledge sources to automatically learn new types of features for Android malware detection. While external knowledge sources are used, the features they are targeting to learn strictly relate to the contents of the app binary, and do not leverage external information about specific applications.

C. Twitter-Augmented Security

Although we are the first to use Twitter data in the Android malware detection field, Twitter has been leveraged recently in other security spaces to aid in detection of security events.

Sabottke et al. [20] crawled Twitter for information about general security exploits as designated by specific vulnerability numbers in the CVE database. They require that the exact CVE number exist in the tweet to be included in their study, similar to our HTTP link-based approach. They note that data sparsity is a problem that affects their domain as 90% of the CVE numbers appear in less than 50 tweets. This finding is generally consistent with our findings that there are not many tweets for a specific app. As Twitter is an open data source (*i.e.*, anyone can post on any topic), malicious figures could intentionally inject misleading information to mislead the tool. As such, they utilize a Threat Model-based approach such that they can be more resilient to potential data poisoning attacks. The goal of their tool is to provide an exploit detector as well as to be able to assist with response prioritization.

Previously, methods for detecting malicious links in social media sites have been studied [2]. In this work, the authors discuss the sorts of features that are useful in detecting whether a social media post may have a malicious intent behind it. Their findings on the usefulness of the tweet metrics are consistent with our findings that metrics about a tweet and its author are highly informative. Ritter et al. [18] investigated the use of semi-supervised learning for security event detection from Twitter. They utilized a semi-supervised variant of Logistic Regression to learn to detect tweets representing security events such as a Distributed Denial of Service attack or data breaches by using only positive and unlabeled data.

D. Text Linking

Attempting to link together two different types of texts has been studied at length. Previous work has introduced an approach that leverages Weighted Textual Matrix Factorization (WTMF) as a means for computing sentence similarity [9]. WTMF, working with texts of sentence length, also deals with a similar issue in data sparsity.

A further work by Guo et al. [10] leverages WTMF in addition to other semantic knowledge elements from Twitter in an attempt to link tweets with news articles. In addition to WTMF, concepts like temporal similarity and common hashtags are used as edges in a graph. Unfortunately, many of the concepts including common hashtags and temporal

similarity cannot be utilized as readily in linking tweets to Android app descriptions.

VI. Conclusions & Future Work

In this paper, we presented a novel approach for augmenting machine learning approaches with Twitter data to improve Android malware detection. We introduced three linking techniques, which allow us to make connections between tweets and the apps that they reference. Our preliminary findings in Section IV-B show that Twitter data is a beneficial added dimension for Android app classification and malware detection. We believe that as we scale our methods up to larger datasets, performance will continue to improve. While there is more work to be done in both techniques for linking the tweets and the apps, as well as how to leverage the resulting linked Twitter data, our work, the first of its kind integrating social media data with Android malware detection, proves to be a promising avenue for future research.

For malware to be effective, they must not only be present in the app store, but they must be downloaded and consumed by end users. Social media links represent a popular way to spread to new devices. As such, our approach allows a more comprehensive look at the Android malware as a whole. We can observe not only what a malware may do on a device through our app binary-based features, but also how it attempts to spread through online social media.

We identify three main avenues for potential future work. First, it would be useful to leverage Twitter data to a larger extent to garner a better perspective from the tweets that we can link to apps. This could potentially imply looking at more advanced methods for Twitter spam detection. Additionally, crawling a graph of Twitter users talking about Android apps could help to establish webs of credibility. A second direction would be to use a Threat Model as applied to Twitter [20] to make it more resilient to potential data source poisoning attacks. Finally, it would be interesting to expand the social media data sources beyond Twitter, potentially considering blogs, other social networks, etc.

VII. Acknowledgements

Part of the computing for this project was performed on the Beocat Research Cluster at Kansas State University, which is funded in part by the NSF grants MRI-1429316 and CC-IIE-1440548. This project is partially supported by the National Science Foundation under Grant No. 1717871. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Virus Total. https://www.virustotal.com/, May 2017.

[2] B. Alghamdi, J. Watson, and Y. Xu. Toward detecting malicious links in online social networks through user behavior. In *IEEE/WIC/ACM Int. Conf. on Web Intelligence Workshops (WIW)*, pages 5–8. IEEE, 2016.

[3] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon. Androzoo: Collecting millions of android apps for the research community. In *Proc. of the 13th Int. Conference on Mining Software Repositories*, pages 468–471. ACM, 2016.

[4] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *NDSS*, 2014.

[5] V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, S. Arzt, S. Rasthofer, and E. Bodden. Mining apps for abnormal usage of sensitive data. In *Proc. of the 37th Int. Conf. on Software Eng.*, pages 426–436, 2015.

[6] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[8] J. DeLoach, D. Caragea, and X. Ou. Android malware detection with weak ground truth data. In *IEEE Int. Conf. on Big Data*, pages 3457–3464, 2016.

[9] W. Guo and M. Diab. A simple unsupervised latent semantics based approach for sentence similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 586–590. Association for Computational Linguistics, 2012.

[10] W. Guo, H. Li, H. Ji, and M. Diab. Linking tweets to news: A framework to enrich short text data in social media. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, 2013.

[11] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[12] Kapersky. Kaspersky security bulletin 2016 review of the year. 2016.

[13] D. Kong, L. Cen, and H. Jin. Autoreb: Automatically understanding the review-to-behavior fidelity in android applications. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 530–541. ACM, 2015.

[14] C. D. Manning, P. Raghavan, et al. *Introduction to information retrieval*, volume 1.

[15] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[16] McAfee. Mobile threat report 2017. https://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2017.pdf, 2016.

[17] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1354–1365. ACM, 2014.

[18] A. Ritter, E. Wright, W. Casey, and T. Mitchell. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th International Conference on World Wide Web*, pages 896–905. ACM, 2015.

[19] S. Roy, J. DeLoach, Y. Li, N. Herndon, D. Caragea, X. Ou, V. P. Ranganath, H. Li, and N. Guevara. Experimental study with real-world data for android app security analysis using machine learning. In *Proc. of the 31st ACSAC*, pages 81–90. ACM, 2015.

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

[21] N. Srebro, T. Jaakkola, et al. Weighted low-rank approximations. In *Icml*, volume 3, pages 720–727, 2003.

[22] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro. Droidsieve: Fast and accurate classification of obfuscated android malware. In *Proc. of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 309–320. ACM, 2017.

[23] N. Viennot, E. Garcia, and J. Nieh. A measurement study of google play. In *ACM SIGMETRICS Performance Evaluation Review*, volume 42, pages 221–233. ACM, 2014.

[24] M. Zhang, Y. Duan, H. Yin, and Z. Zhao. Semantics-aware android malware classification using weighted contextual api dependency graphs. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1105–1116. ACM, 2014.

[25] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Int. Conf. on Algorithmic Applications in Management*, pages 337–348. Springer, 2008.

[26] Z. Zhu and T. Dumitras. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 767–778. ACM, 2016.