WARNINGBIRD: Detecting Suspicious URLs in Twitter Stream *

Sangho Lee† and Jong Kim‡
†Department of Computer Science and Engineering
‡Division of IT Convergence Engineering
Pohang University of Science and Technology (POSTECH)
Pohang, Republic of Korea
{sangho2,jkim}@postech.ac.kr

Abstract

Twitter can suffer from malicious tweets containing suspicious URLs for spam, phishing, and malware distribution. Previous Twitter spam detection schemes have used account features such as the ratio of tweets containing URLs and the account creation date, or relation features in the Twitter graph. Malicious users, however, can easily fabricate account features. Moreover, extracting relation features from the Twitter graph is time and resource consuming. Previous suspicious URL detection schemes have classified URLs using several features including lexical features of URLs, URL redirection, HTML content, and dynamic behavior. However, evading techniques exist, such as time-based evasion and crawler evasion. In this paper, we propose WARNINGBIRD*, a suspicious URL detection system for Twitter. Instead of focusing on the landing pages of individual URLs in each tweet, we consider correlated redirect chains of URLs in a number of tweets. Because attackers have limited resources and thus have to reuse them, a portion of their redirect chains will be shared. We focus on these shared resources to detect suspicious URLs. We have collected a large number of tweets from the Twitter public timeline and trained a statistical classifier with features derived from correlated URLs and tweet context information. Our classifier has high accuracy and low false-positive and false-negative rates. We also present* WARNINGBIRD *as a real-time system for classifying suspicious URLs in the Twitter stream.*

*This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2011-C1090-1131-0009) and World Class University program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea(R31-10100).

1 Introduction

Twitter is a well-known social networking and information sharing service [15] that allows users to exchange messages of fewer than 140-character, also known as *tweets*, with their friends. When a user Alice updates (or sends) a tweet, this tweet will be distributed to all of her *followers*, who have registered Alice as one of their friends. Instead of distributing her tweets to all of her followers, Alice can send a tweet to a specific twitter user Bob by mentioning this user by including *@Bob* in the tweet. Unlike status updates, mentions can be sent to users who do not follow Alice. When Twitter users want to share URLs with friends via tweets, they usually use URL shortening services [1] to reduce the length of these URLs, because tweets can only contain a restricted number of characters. `bit.ly` and `tinyurl.com` are widely used services, and Twitter also provides its own shortening service `t.co`.

Owing to the popularity of Twitter, malicious users often try to find a way to attack it. Most common forms of Web attacks, including spam, scam, phishing, and malware distribution attacks, have appeared on Twitter. Because tweets are short in length, attackers use shortened malicious URLs that redirect Twitter users to external attack servers [6, 11, 19, 23].

To cope with malicious tweets, many Twitter spam detection schemes have been proposed. These schemes can be classified into account feature-based [2, 16, 23, 28] and relation feature-based [21, 31] schemes. Account feature-based schemes use the distinguishing features of spam accounts such as the ratio of tweets containing URLs, the account creation date, and the number of followers and friends. However, malicious users can easily fabricate these account features. The relation feature-based schemes rely on more robust features that malicious users cannot easily fabricate such as the distance and connectivity apparent in the Twitter graph. Extracting these relation features from the Twitter graph, however, requires a significant amount of time and

resources, because the Twitter graph is tremendous in size.

A number of suspicious URL detection schemes [3, 17–19, 24, 30] have also been introduced. They use static or dynamic crawlers and may be executed in virtual machine honeypots, such as Capture-HPC [4], HoneyMonkey [29], and Wepawet [7], to investigate newly observed URLs. These schemes classify URLs according to several features including lexical features of URLs, DNS information, URL redirection, and the HTML content of the landing pages. Nonetheless, malicious servers can bypass investigation by selectively providing benign pages to crawlers. For instance, because static crawlers usually cannot handle JavaScript or Flash, malicious servers can use them to deliver malicious content only to normal browsers. Even if investigators use dynamic crawlers that have (almost) all the functionalities of real browsers, malicious servers may be able to distinguish them through IP address, user interaction, browser fingerprinting [8], or honeyclient detection techniques [14]. A recent technical report from Google has also discussed techniques for evading current Web malware detection systems [20]. Malicious servers can also employ temporal behaviors—providing different content at different times—to evade investigation [24].

In this paper, we propose WARNINGBIRD, a suspicious URL detection system for Twitter. Instead of investigating the landing pages of individual URLs in each tweet, which may not be successfully fetched, we considered correlated redirect chains of URLs included in a number of tweets. Because attackers' resources are limited and need to be reused, a portion of their redirect chains must be shared. We found a number of meaningful features of suspicious URLs derived from the correlated URL redirect chains and related tweet context information. We collected a large number of tweets from the Twitter public timeline and trained a statistical classifier with their features. The trained classifier has high accuracy and low false-positive and false-negative rates.

The contributions of this paper can be summarized as follows:

- We present a new suspicious URL detection system for Twitter that is based on correlations of URL redirect chains, which are difficult to fabricate. The system can classify suspicious URLs found in the Twitter stream in real time.
- We introduce new features of suspicious URLs: some of them are newly discovered and others are variations of previously discovered features.
- We present some investigation results regarding suspicious URLs that have been widely distributed through Twitter over the past several months and continue to remain active.

The remainder of this paper is organized as follows. In Section 2, we discuss case studies on suspicious URLs in Twitter. In Section 3, we introduce our system, WARNINGBIRD. In Section 4, we present the evaluation results. In Section 5, we discuss the limitations of the proposed system. In Section 6, we discuss related work. Finally, we conclude this paper in Section 7.

2 Case Study

2.1 blackraybansunglasses.com

We consider `blackraybansunglasses.com`, which is a suspicious site associated with spam tweets. We first encountered this site in April 2011 and it remains active. We use a one percent sample of tweets collected on July 11, 2011, to conduct an in-depth analysis of the site (see Figure 1). `blackraybansunglasses.com` has a page, `redirect.php`, that conditionally redirects users to random spam pages. It uses a number of different Twitter accounts and shortened URLs to distribute its URL to other Twitter users. According to our dataset, it uses $6,585$ different Twitter accounts and shortened URLs, and occupies about 2.83% of all the $232,333$ tweets with URLs that we sampled. When a user clicks on one of the shortened URLs, such as `bit.ly/raCz5i` distributed by `zarzuelavbafpv0`, he or she will be redirected to a private redirection site, such as `beginnersatlanta.tk`, which seems to be managed by the operator of `blackraybansunglasses.com`. The user will then be repeatedly redirected to `bestfreevideoonline.info` and `blackraybansunglasses.com`. The redirection site `blackraybansunglasses.com` evaluates whether its visitors are normal browsers or crawlers using several methods, including cookie or user-agent checking. When it is sure that a current visitor is a normal browser, it redirects the visitor to `forexstrategysite.com`, which then finally redirects him or her to random spam pages. When `blackraybansunglasses.com` determines that a current visitor is not a normal browser, it simply redirects the visitor to `google.com` to avoid investigation. Therefore, crawlers may not be able to see `forexstrategysite.com` or the further random spam pages.

Another interesting point about `blackraybansunglasses.com` is that it does not use Twitter APIs to distribute malicious tweets. Instead, it abuses the Twitter Web interface. Previous Twitter spam detection schemes usually assumed that many spammers would use Twitter APIs to distribute their spam tweets. Smart Twitter spammers, however, no longer rely on Twitter APIs, because they know that using APIs will

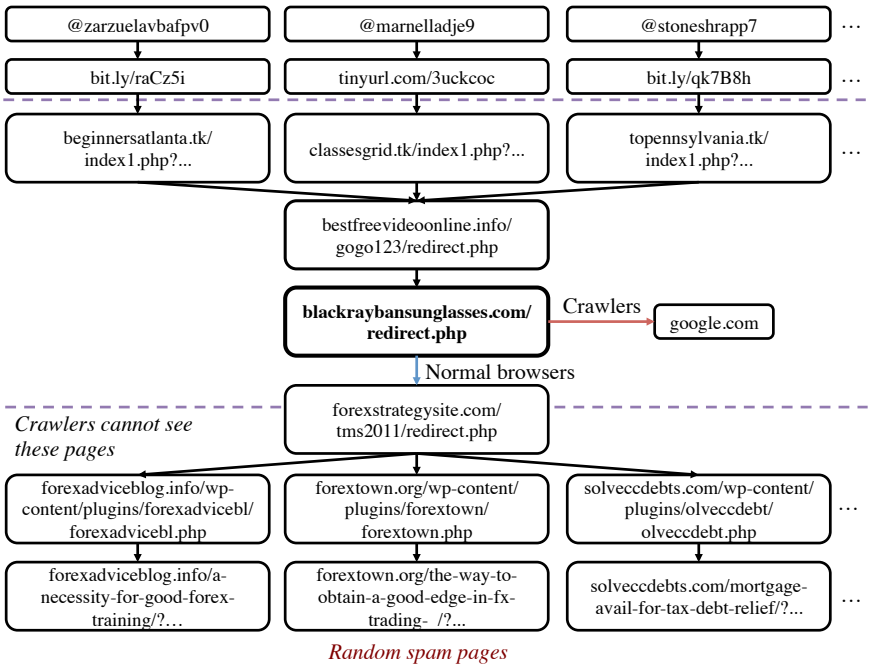

Figure 1. Redirect chains of blackraybansunglasses.com on July 11, 2011

distinguish their tweets from normal tweets. For instance, `tweetattacks.com` [26] sells a Twitter spam program that uses the Web interface instead of APIs to make spam receivers believe that the received tweets are not spam and to circumvent API limits.

2.2 24newspress.net

Let us also discuss `24newspress.net`, which is a suspicious site distributed via tweets. We first found this site at the end of June 2011 and it remains active. We use one percent of the tweet samples collected on July 23, 2011, to conduct an in-depth analysis of the page (see Figure 2). Unlike `blackraybansunglasses.com` `24newspress.net` does not perform conditional redirection to avoid investigation. Instead, it uses a number of IP addresses and domain names for cloaking like IP fast flux and domain flux methods [12, 22]. It has five other domain names: `24dailyreports.net`, `7reports.net`, `job365report.net`, `jobs-post.net`, and `week-job.net`. It also uses a number of different shortened URLs and different Twitter accounts to distribute tweets to Twitter users. In our dataset, we found $6,205$ tweets related to `24newspress.net`, which represent about 2.41% of all the $257,329$ tweets with URLs sampled. In addition, it abuses a mobile Twitter Web interface to distribute its spam tweets.

2.3 Frequent URL Redirect Chains

We performed a simple investigation on three days' worth of tweet samples culled from July 23 to 25, 2011. We extracted frequent URL redirect chains from the sample data and ranked them according to their frequency after removing whitelisted domain names. Many suspicious sites, such as `jbfollowme.com`, which attempts to attract Justin Bieber's fans, proved to be highly ranked (see Table 1).

2.4 Observations

From the previous examples, we can identify meaningful characteristics of suspicious URLs. They use a number of different Twitter accounts and shortened URLs, or a number of domain names and IP addresses to cloak the same suspicious URLs. They also use long redirect chains to avoid investigation. Moreover, they appear more frequently in the Twitter public timeline than benign URLs. These characteristics are the basis for the feature models we employ to classify suspicious URLs.

3 Proposed System

3.1 Motivation and Basic Idea

Our goal is to develop a suspicious URL detection system for Twitter that is robust enough to protect against con-

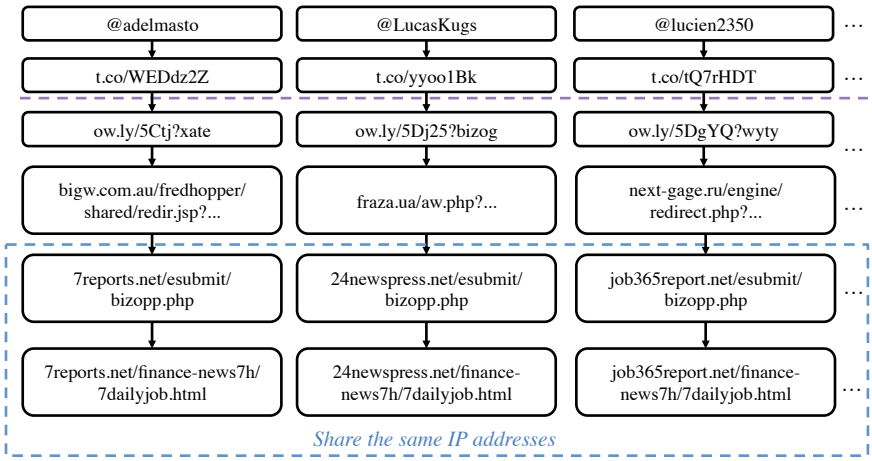

Figure 2. Redirect chains of 24newspress.net on July 23, 2011

Table 1. Domain names of frequent URL redirect chains from July 23 to 25, 2011

Rank	July 23	July 24	July 25
1	24newpress.net	24newspress.net	24newpress.net
2	blackraybansunglasses.com	blackraybansunglasses.com	blackraybansunglasses.com
3	software-spot.com	cheapdomainname.info	bigfollow.net
4	ustream.tv	ustream.tv	twitmais.com
5	10bit.info	twitmais.com	jbfollowme.com
6	blackreferrer.com	bigfollow.net	addseguidores.com.br
7	tweetburner.com	jbfollowme.com	elitebrotherhood.net
8	livenation.com	10bit.info	livenation.com
9	twitmais.com	addseguidores.com.br	naturesoundcds.com
10	bigfollow.net	wayjump.com	all-about-legal.net

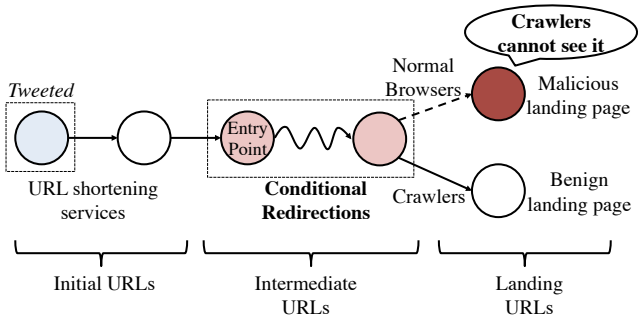

Figure 3. Conditional redirection

ditional redirections. Let us consider a simple example of conditional redirections (see Figure 3). In this example, an attacker creates a long URL redirect chain by using public URL shortening services, such as `bit.ly` and `t.co`, and his or her own private redirection servers to redirect visitors to a malicious landing page. The attacker then uploads a tweet including the initial URL of the redirect chain to Twitter. Later, when a user or a crawler visits the initial URL, he or she will be redirected to an *entry point* of intermediate URLs that are associated with private redirection servers. Some of these redirection servers will check whether the current visitor is a normal browser or a crawler. If the current visitor seems to be a normal browser, they will redirect the visitor to a malicious landing page. If not, they will redirect the visitor to a benign landing page. Therefore, the attacker can selectively attack normal users while deceiving investigators.

The above example shows us that, as investigators, we cannot fetch malicious landing URLs, because attackers do not reveal them to us. We also cannot rely on the initial URLs, because attackers can generate a large number of different initial URLs by abusing URL shortening services. Fortunately, the case study on `blackraybansunglasses.com` shows that attackers *reuse some of their redirection servers* when creating a number of redirect chains, because they do not have infinite redirection servers (see Section 2). Therefore, if we analyze a number of *correlated redirect chains* instead of an individual redirect chain, we can find an entry point of

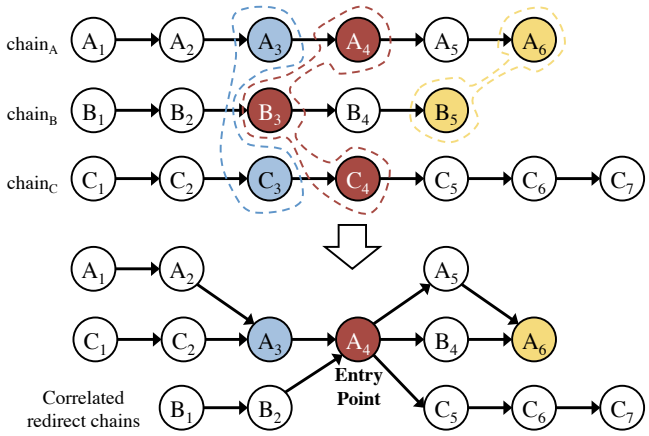

Figure 4. Redirect chains and their correlation

the intermediate URLs in the correlated redirect chains. Let us consider the three redirect chains shown in the top half of Figure 4. These three redirect chains share some URLs: A_3=C_3, A_4=B_3=C_4, and A_6=B_5. By combining the three redirect chains using these shared URLs, we can generate the correlated redirect chains (the bottom half of Figure 4) that share the same entry point URL, A_4 (because A_4 is the most frequent URL in the chains). The correlated redirect chains show that the entry point has three different initial URLs and two different landing URLs, and participates in redirect chains that are six to seven URLs long. These are the characteristics of the suspicious URLs that we considered in Section 2. Therefore, this correlation analysis can help to detect suspicious URLs even when they perform conditional redirections, because the suspiciousness of the two landing URLs is not important to the correlation analysis.

3.2 System Details

WARNINGBIRD is composed of four major components: data collection, feature extraction, training, and classification (see Figure 5).

Data collection: The data collection component has two subcomponents: the collection of tweets with URLs and crawling for URL redirections. To collect tweets with URLs and their context information from the Twitter public timeline, this component uses Twitter Streaming APIs [27]. Whenever this component receives a tweet with a URL from Twitter, it executes a crawling thread that follows all redirections of the URL and looks up the corresponding IP addresses. The crawling thread appends these retrieved URL and IP chains to the tweet information and pushes this extended tweet information into a tweet queue. As we have seen, our crawler cannot reach malicious landing URLs when they use conditional redirections to evade crawlers. However, because our detection system does not rely on the features of landing URLs, it works independently of such crawler evasions.

Feature extraction: The feature extraction component has three subcomponents: grouping identical domains, finding entry point URLs, and extracting feature vectors. This component monitors the tweet queue to check whether a sufficient number of tweets have been collected. Specifically, our system uses a tweet window instead of individual tweets. When more than w tweets are collected (w is $10,000$ in the current implementation), it pops w tweets from the tweet queue. First, for all URLs in the w tweets, this component checks whether they share the same IP addresses. If some URLs share at least one IP address, it replaces their domain names with a list of those with which they are grouped. For instance, when `http://123.com/hello.html` and `http://xyz.com/hi.html` share the same IP address, this component replaces these URLs with `http://['123.com','xyz.com']/hello.html` and `http://['123.com','xyz.com']/hi.html`, respectively. This grouping process allows the detection of suspicious URLs that use several domain names to bypass blacklisting.

Next, the component tries to find the entry point URL for each of the w tweets. First, it measures the frequency with which each URL appears in the w tweets. It then discovers the most frequent URL in each URL redirect chain in the w tweets. The URLs thus discovered become the entry points for their redirect chains. If two or more URLs share the highest frequency in a URL chain, this component selects the URL nearest to the beginning of the chain as the entry point URL.

Finally, for each entry point URL, this component finds URL redirect chains that contain the entry point URL, and extracts various features from these URL redirect chains and the related tweet information (details of these features will be given in Subsection 3.3). These feature values are then turned into real-valued feature vectors.

When we group domain names or find entry point URLs, we ignore whitelisted domains to reduce false-positive rates. Whitelisted domains are not grouped with other domains and are not selected as entry point URLs. Our whitelisted domain names include the Alexa Top 1000 sites, some famous URL shortening sites, and some domains that we have manually verified.

Training: The training component has two subcomponents: retrieval of account statuses and the training classifier. Because we use an offline supervised learning algorithm, the feature vectors for training are relatively old values than feature vectors for classification. To label the training vectors,

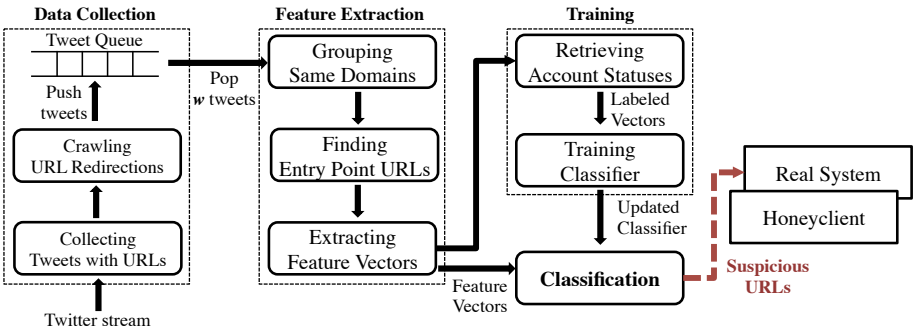

Figure 5. System overview

we use the Twitter account status; URLs from suspended accounts are considered malicious and URLs from active accounts are considered benign. We periodically update our classifier by using labeled training vectors.

Classification: The classification component executes our classifier using input feature vectors to classify suspicious URLs. When the classifier returns a number of malicious feature vectors, this component flags the corresponding URLs and their tweet information as suspicious. These URLs, detected as suspicious, will be delivered to security experts or more sophisticated dynamic analysis environments for in-depth investigation.

3.3 Features

We introduce 12 features for classifying suspicious URLs on Twitter. These features can be classified as features derived from correlated URL redirect chains and features derived from the related tweet context information. We also describe how we normalize these feature values to real values between zero and one.

3.3.1 Features Derived from Correlated URL Redirect Chains

URL redirect chain length: Attackers usually use long URL redirect chains to make investigations more difficult and avoid the dismantling of their servers. Therefore, when an entry point URL is malicious, its chain length may be longer than those of benign URLs. To normalize this feature, we choose an upper-bound value of 20, because most of the redirect chains we have seen over the four-month period have had fewer than 20 URLs in their chains. If the length of a redirect chain is l, this feature can be normalized as $\min(l, 20)/20$.

Frequency of entry point URL: The number of occurrences of the current entry point URL within a tweet window is important. Frequently appearing URLs that are not whitelisted are usually suspicious, as discussed in Section 2. When the window size is w and the number of occurrences is n, this feature can be normalized as n/w.

Position of entry point URL: Suspicious entry point URLs are not usually located at the end of a redirect chain, because they have to conditionally redirect visitors to different landing URLs. If the position of an entry point of a redirect chain of length l is p, this can be normalized as p/l.

Number of different initial URLs: The initial URL is the beginning URL that redirects visitors to the current entry point URL. Attackers usually use a large number of different initial URLs to make their malicious tweets, which redirect visitors to the same malicious URL, look different. If the number of different initial URLs redirecting visitors to an entry point URL that appears n times is i, this feature can be normalized as i/n.

Number of different landing URLs: If the current entry point URL redirects visitors to more than one landing URL, we can assume that the current entry point URL performs conditional redirection behaviors and may be suspicious. If an entry point URL that appears n times redirects visitors to λ different landing URLs, this feature can be normalized as λ/n.

3.3.2 Features Derived from Tweet Context Information

The features derived from the related tweet context information are variations of previously discovered features. Our variations focused on the similarity of tweets that share the same entry point URLs.

Number of different sources: Sources are applications that upload the current entry point URL to Twitter. Attackers usually use the same source application, because maintaining a number of different applications is difficult. Benign users, however, usually use various Twitter applications, such as TweetDeck and Echofon. Therefore, the number

of different sources may be small when the current entry point URL is suspicious. If the number of different sources of an entry point URL that occurs n times is s, this feature can be normalized as s/n.

Number of different Twitter accounts: The number of different Twitter accounts that upload the current entry point URL can be used to detect injudicious attackers who use a small number of Twitter accounts to distribute their malicious URLs. If the number of Twitter accounts uploading an entry point URL that occurs n times is α, this feature can be normalized as α/n.

Standard deviation of account creation date: Attackers usually create a large number of Twitter accounts within a relatively short time period. Therefore, if the creation dates of the accounts that upload the same entry point URL are similar, it might indicate that the current entry point URL is suspicious. We use the standard deviation of account creation date as a similarity measure. To normalize the standard deviation, we assume that the time difference between any account creation dates is less than or equal to one year. Therefore, this feature can be normalized as

$$\min\left(\frac{\text{std(a set of account creation date)}}{\text{(1 year)}\sqrt{n}}, 1\right).$$

Standard deviation of the number of followers and number of friends: The numbers of followers and friends of attackers' accounts are usually similar, because attackers use certain programs to increase their numbers of followers and friends. We again use standard deviations to check for similarities in the numbers of followers and friends. To normalize the standard deviations, we assume that the number of followers and friends is usually less than or equal to $2,000$, which is the restricted number of accounts Twitter allows one can to follow. Therefore, these features can be normalized as

$$\min\left(\frac{\text{std(\#followers or \#friends)}}{2000\sqrt{n}}, 1\right).$$

Standard deviation of the follower-friend ratio: We define the follower-friend ratio as below:

$$\frac{\text{min(\#followers, \#friends)}}{\text{max(\#followers, \#friends)}}.$$

Like the numbers of followers and friends, the follower-friend ratios of attackers' accounts are similar. We use a normalized standard deviation to check the similarity as

$$\min\left(\frac{\text{std(a set of follower-friend ratios)}}{\sqrt{n}}, 1\right).$$

Because attackers' accounts usually have more friends than followers, the follower-friend ratios of malicious accounts are usually different from the follower-friend ratios of benign accounts. Attackers, however, can fabricate this ratio, because they can use Sybil followers or buy followers. Therefore, instead of using an individual follower-friend ratio, we use the standard deviation of follower-friend ratios of accounts that post the same URLs and assume that fabricated ratios will be similar.

Tweet text similarity: The texts of tweets containing the same URL are usually similar (e.g., retweets). Therefore, if the texts are different, we can assume that those tweets are related to suspicious behaviors, because attackers usually want to change the appearance of malicious tweets that include the same malicious URL. We measure the similarity between tweet texts as

$$\sum_{t,u \in \text{a set of pairs in tweet texts}} \frac{J(t,u)}{|\text{a set of pairs in tweet texts}|},$$

where $J(t,u)$ is the Jaccard index [13], which is a famous measure that determines the similarity between two sets t and u, and is defined as below:

$$J(t,u) = \frac{|t \cap u|}{|t \cup u|}.$$

We remove mentions, hashtags, retweets, and URLs from the texts when we measure their similarity, so that we only consider the text features.

4 Evaluation

4.1 System Setup and Data Collection

Our system uses two Intel Quad Core Xeon E5530 2.40GHz CPUs and 24 GiB of main memory. To collect tweets, we use Twitter Streaming APIs [27]. Our accounts have the Spritzer access role; thus, we can collect about one percent of the tweets from the Twitter public timeline as samples. From April 8 to August 8, 2011 (122 days), we collected 27,895,714 tweet samples with URLs.

4.2 Feature Selection

To evaluate and compare the features of our scheme, we use the F-score [5]. The F-score of a feature represents the degree of discrimination of the feature. Features with large F-scores can split benign and malicious samples better than features with small F-scores. The F-score shows that the redirect chain length is the most important feature, followed by the number of different sources and the standard deviation of the account creation date (see Table 2). We also verify that the number of different Twitter accounts that upload an entry point URL is a less important feature.

Table 2. F-score of our features

Feature	F-score
URL redirect chain length	0.0963
Number of different sources	0.0798
Standard deviation of account creation date	0.0680
Frequency of entry point URL	0.0374
Position of entry point URL	0.0353
Standard deviation of friends-followers ratio	0.0321
Number of different landing URLs	0.0150
Number of different initial URLs	0.0117
Standard deviation of the number of followers	0.0085
Tweet text similarity	0.0060
Standard deviation of the number of friends	0.0050
Number of different Twitter accounts	0.0008

Table 3. Training and test datasets

Dataset	Period	Benign	Malicious	Total
Training	5/10–7/8	183,113	41,721	224,834
$\text{Test}_{\text{past}}$	4/8–5/9	71,220	6,730	77,950
$\text{Test}_{\text{future}}$	7/9–8/8	91,888	4,421	96,309

This result implies that attackers use a large number of different Twitter accounts to distribute their malicious URLs. The similarity of tweet texts is also less important, because many attackers currently do not use different tweet texts to cloak their malicious tweets. In addition, the standard deviations of the number of followers and number of friends are less important, because benign users' numbers of followers and friends are also similar. Interestingly, the standard deviation of the number of followers has a higher F-score value than that of the number of friends, because fabricating the number of followers is more difficult than fabricating the number of friends.

4.3 Training and Testing Classifiers

We use 60 days of tweet samples from May 10–July 8 for training the classification models and 62 days of tweet samples from April 8–May 9 and July 9–August 8 to test the classifier with older and newer datasets, respectively. For training and testing, we need to label the datasets. Unfortunately, we cannot find suitable blacklists for labeling our datasets, because many URLs in our datasets, such as `blackraybansunglasses.com`, are still not listed on public URL blacklists such as the Google Safe Browsing API [10]. Therefore, instead of URL blacklists, we use Twitter account status information to label our datasets. Namely, if some URLs are from suspended accounts, we treat the URLs as malicious. If not, we treat the URLs as benign. Recently, Thomas *et al.* [25] figured out that most suspended accounts are spam accounts. Therefore, our treatment of URLs is valid. From the training dataset, we found 4,686,226 accounts that were active and 263,289 accounts that were suspended as of August 11, 2011. We also found 224,834 entry point URLs that appear *more than once* in some windows of 10,000 sample tweets. Among them, 183,113 entry point URLs are from active accounts and 41,721 entry point URLs are from suspended accounts. Therefore, we designated the 183,113 entry point URLs from active accounts as benign samples and the remaining 41,721 entry point URLs as malicious samples. We also use the account status information to label the test datasets; the results are shown in Table 3.

Table 5. Comparing classifier accuracy while varying training weights of benign samples within a 10-fold cross validation (cost 1.6)

		%		
Weight	AUC	Accuracy	FP	FN
1.0	0.8312	87.66	1.67	10.67
1.2	0.8310	87.51	1.31	11.18
1.4	**0.8310**	**87.09**	**1.03**	**11.88**
1.6	0.8309	86.39	0.83	12.78
1.8	0.8310	86.15	0.71	13.14
2.0	0.8308	85.99	0.61	13.39

We used the LIBLINEAR library [9] to implement our classifier. We compared seven classification algorithms with our training dataset and selected an L2-regularized logistic regression algorithm with a primal function, because it shows the best accuracy values with our dataset (see Table 4). We also tested a number of learning cost values and chose a cost value of 1.6. Because our dataset is unbalanced—the number of benign samples is 4.4 times larger than that of malicious samples—we must choose a good weight value to give a penalty to benign samples. We compared six weight values and selected a weight value of 1.4 for benign samples, because this value produces good accuracy values and relatively low false-positive and false-negative rates (see Table 5). All the training and 10-fold cross validation can be done in less than three seconds in our evaluation environment. Therefore, the training time is negligible.

We use two test datasets that represent past and future values, to evaluate the accuracy of our classifier (see Table 3). Whether the test datasets regard the past or future ones, our classifier achieves high accuracy, and low false-positive and false-negative rates (see Table 6). Therefore, our features do not tightly depend on specific time periods and, hence, can be used generally.

Table 4. Comparing classifiers within a 10-fold cross validation (learning cost 1.0 and weight 1.0). Logistic regression (LR), support vector classification (SVC), area under the ROC curve (AUC), false positive (FP), false negative (FN), and Lagrange primal and dual maximal violation functions that determine termination of training.

		%		
Classifier	AUC	Accuracy	FP	FN
L2-regularized LR (primal)	**0.8312**	**87.67**	**1.64**	**10.69**
L2-regularized L2-loss SVC (dual)	0.8267	86.93	1.40	11.67
L2-regularized L2-loss SVC (primal)	0.8268	86.95	1.38	11.67
L2-regularized L1-loss SVC (dual)	0.8279	87.50	1.38	11.67
L1-regularized L2-loss SVC (primal)	0.8269	86.74	1.40	11.86
L1-regularized LR (primal)	0.8312	87.64	1.68	10.67
L2-regularized LR (dual)	0.8310	87.63	1.69	10.67

Table 6. Classification accuracy of test datasets

		%		
Dataset	AUC	Accuracy	FP	FN
$\text{Test}_{\text{past}}$	0.7113	91.10	1.32	7.57
$\text{Test}_{\text{future}}$	0.7889	93.11	3.67	3.21

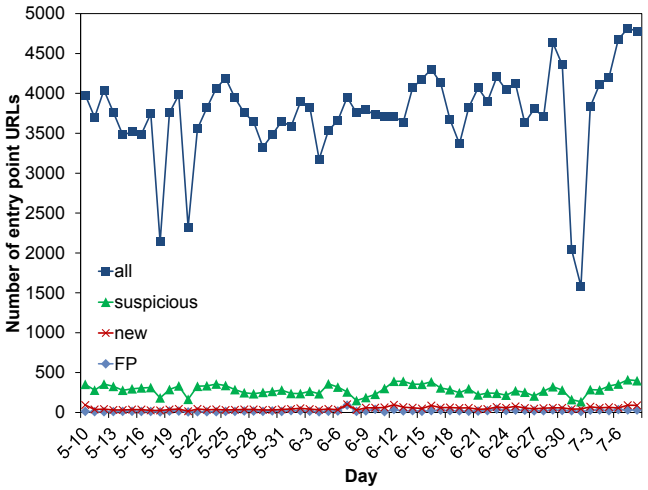

Figure 6. Daily analysis on training datasets (60 days: May 10–July 8)

4.4 Data Analysis

We performed a daily analysis on the training dataset. On average, 3756.38 entry point URLs appear more than once in each tweet window during a given day (with a window size of $10,000$). Among them, on average, 282.93 suspicious URLs are detected, where 19.53 URLs are false positives and 30.15 URLs are newly discovered (see Figure 6). This relatively small number of new suspicious URLs implies that many suspicious URLs repeatedly detected by WARNINGBIRD are not detected or blocked by other existing detection systems. To verify the reoccurrence of suspicious URLs, we grabbed May 10's entry point URLs and checked how many times these URLs had appeared in the Twitter public timeline during the next two months (see Figure 7). On average, 17% of suspicious URLs and 5.1% of benign URLs of May 10 were observed during the two months; thus, suspicious URLs are more repeated than benign URLs. Interestingly, 38.9% of the suspicious URLs had appeared again on July 4, 55 days later. Therefore, existing detection schemes cannot detect or block a portion of suspicious URLs that can be detected by WARNINGBIRD.

We also determine whether the domain groupings allow us to detect a larger number of suspicious URLs. We compare grouped and ungrouped URLs by using 16 days of tweet samples from between July 23 and August 8 (except July 31 owing to a local power blackout). On average, we find 334.94 unique suspicious entry point URLs when we group URLs and 79.88 suspicious URLs when we do not group URLs (see Figure 8). Therefore, the domain groupings give us about 4.19 times better detection rates.

4.5 Running Time

We evaluated the running time of our system. First, we compared the running time of each component of our system—domain grouping; feature extraction, including the detection of entry points; and classification—in a single window of collected tweets that varies in size. Even if the

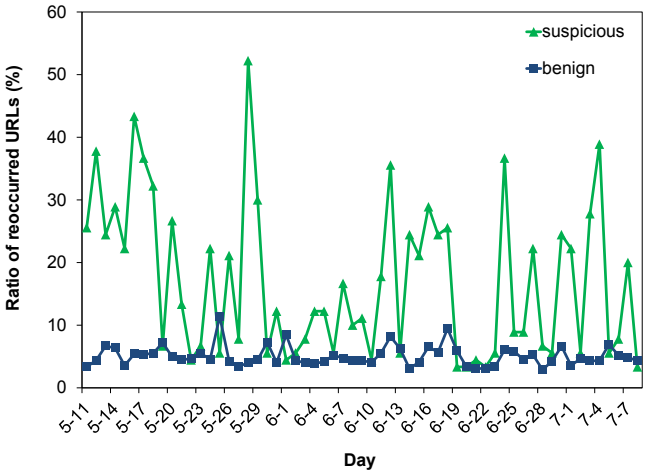

Figure 7. Reoccurrence of May 10's entry point URLs (59 days: May 11–July 8)

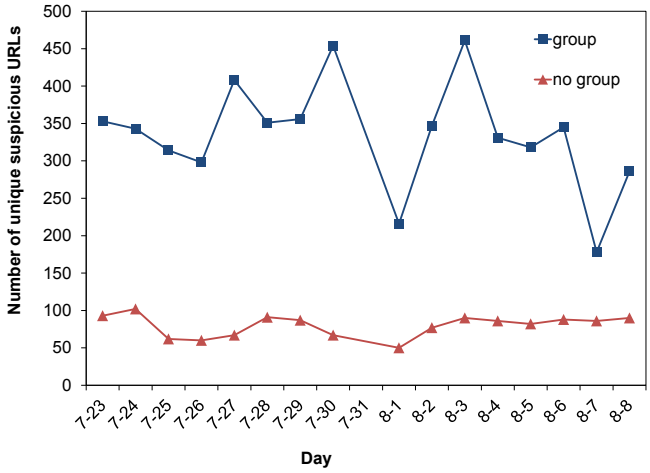

Figure 8. Comparing domain grouping results (16 days: July 23–August 8 excluding July 31)

Table 7. Required time to classify a single URL when a window size is 100,000 and using 100 concurrent connections for crawling

Component	Avg. running time (ms)
Redirect chain crawling	24.202
Domain grouping	2.003
Feature extraction	1.618
Classification	0.484
Total	**28.307**

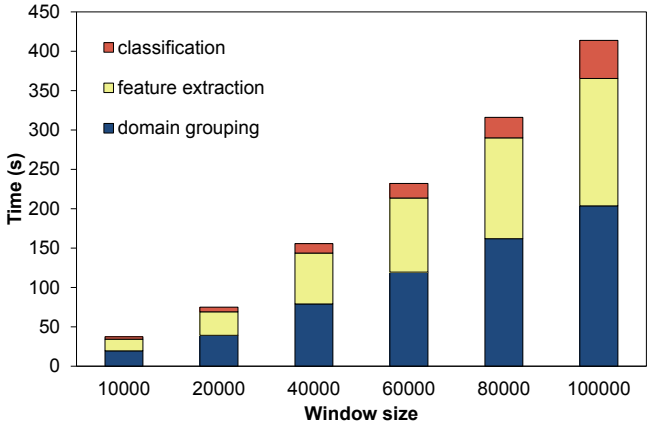

Figure 9. Running time for each component to process a tweet window

window size becomes $100,000$, which contains of about 10% of all tweets with URLs per hour, the running time is only 6.9 minutes (see Figure 9). Next, we estimate the time required to classify a single URL. Our system currently uses 100 crawling threads to *concurrently* visit URL redirect chains; on average, each thread requires 2.42 s to visit a single URL redirect chain. If the window size is $100,000$, we need 28.307 ms to process a single URL (see Table 7); thus, our system can process about $127,000$ URLs per hour. Therefore, our system can handle 10% of the tweet samples, the level provided by the Gardenhose access role, in real time. By increasing the number of crawling threads, we can process more than 10% of the tweet samples. For instance, if we use $1,000$ crawling threads, we can process about $576,000$ URLs per hour. Even if we do this, the current implementation cannot process all the tweets, because we would have to process a single URL in less than 3.6 ms to handle $1,000,000$ URLs per hour.

4.6 Real-time Detection and Sliding Window

The real-time version of WARNINGBIRD uses a sliding window technique for achieving good latency and detection coverage. A small window gives immediate results; however, it cannot catch suspicious URLs that repeat after long-time intervals. A large window has good detection coverage; however, its latency is bad. A sliding window is a well-known technique for taking advantage of both small and large windows. Let w denote the window size and s denote the sliding size ($s \leq w$). Whenever a sliding window system receives s new items, it processes the previous $w - s$ items and the s new items at the same time. Therefore, the latency of this method depends on s and its detection coverage depends on w. Currently, we have set w at

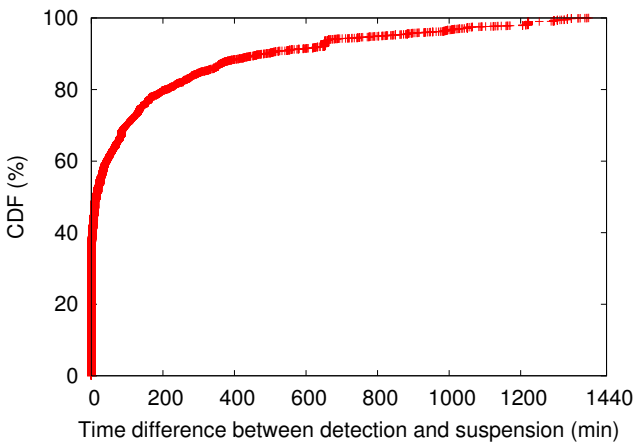

Figure 10. Time difference between WarningBird's detection of suspicious accounts and Twitter's suspension within a day

$10,000$ and s at $2,000$. Every 12 minutes, the real-time version of WARNINGBIRD returns suspicious URLs that have appeared in the previous hour. Because our system can process $10,000$ collected tweets in less than one minute (see Figure 9), we can detect suspicious URLs with only one-minute time lags. Between August 6 and August 18, 2011, the real-time WARNINGBIRD reported $4,336$ unique suspicious URLs without system errors.

4.7 Comparison with Twitter

We compare the efficiency of WARNINGBIRD with that of Twitter's detection system. For the comparison, we sampled $14,905$ accounts detected by the real-time WARNINGBIRD between September 1 and October 22, 2011. To compare their efficiencies, we measured the time difference between WARNINGBIRD's detection and Twitter's suspension of the accounts. We monitored the WARNINGBIRD to obtain newly detected suspicious accounts and then checked the status of each account every 15 s until it was suspended, within a day. Among the sampled accounts, $5,380$ accounts were suspended within a day; 37.3% of them was suspended within a minute, another 42.5% of them was suspended within 200 minutes, and the remaining 20.7% of them was suspended within a day (see Figure 10). The average time difference is 13.5 minutes; therefore, our detection system is more efficient than that of Twitter. We also checked the statuses of the sampled accounts on October 28, 2011 to verify the accuracy of our system. Among the $14,905$ accounts, $9,250$ accounts were suspended. We then randomly selected 500 accounts from the remaining $5,655$ active accounts to manually check their suspiciousness. Among the 500 accounts, 320 accounts were suspicious. Therefore, the detection accuracy of our system given the sample data is about 86.3%.

5 Discussion

In this section, we discuss some limitations of our system and possible evasion techniques.

Dynamic redirection: Currently, WARNINGBIRD uses a static crawler written in Python. Because it can handle only HTTP redirections, it will be ineffective on pages with embedded dynamic redirections such as JavaScript or Flash redirection. Therefore, WARNINGBIRD will designate pages with embedded dynamic redirection as entry point URLs. This determination causes inaccuracy in some of the feature values, including the redirect chain lengths, positions of the entry point URLs, and the number of different landing URLs. Therefore, in the future we will use customized Web browsers to retrieve redirect chains fully.

Multiple redirections: Web pages can embed several external pages and different content. Therefore, some pages can cause multiple redirections. Because our system currently only considers HTTP redirection and does not consider page-level redirection, it cannot catch multiple redirections. Therefore, we need customized browsers to catch and address multiple redirections.

Coverage and scalability: Currently, our system only monitors one percent of the samples from the Twitter public timeline, because our accounts have the Spritzer access role. As shown in Section 4, if our accounts were to take on the Gardenhose access role, which allows the processing of 10% of the samples, our system could handle this number of samples in real time. The current implementation, however, cannot handle 100% of the Twitter public timeline. Therefore, we must extend WARNINGBIRD to a distributed detection system, for instance, Monarch [24], to handle the entire Twitter public timeline.

Feature evasion methods: Attackers can fabricate the features of their attacks to evade our detection system. For instance, they could use short redirect chains, change the position of their entry point URLs, and reuse initial and landing URLs. These modifications, paradoxically, would allow previous detection systems to detect their malicious URLs. Attackers may also be able to reduce the frequency of their tweets to bypass our detection system. However, this will also reduce the number of visitors to their malicious pages. Features derived from tweet information, however, are relatively weak at protecting against forgery, as many researchers have already pointed out [21, 24, 31]. Attackers could use a large number of source applications and Twitter accounts, use similar tweet texts, and carefully adjust the numbers of followers and friends of their accounts

to increase the standard deviation values. In addition, they could increase the standard deviation of their account creation date if they own or have compromised older accounts. Although these features are weak, attackers have to consume their resources and time to fabricate these features. Therefore, using these features is still meaningful. The strongest evasion method is definitely to increase the number of redirect servers. This method, however, would require many resources and large financial investment on the part of the attackers.

6 Related Work

6.1 Twitter Spam Detection

Many Twitter spam detection schemes have been introduced. Most have focused on how to collect a large number of spam and non-spam accounts and extract the features that can effectively distinguish spam from non-spam accounts. To detect spam accounts, some schemes investigate collected data manually [2, 28], some use *honey-profiles* to lure spammers [16, 23], some monitor the Twitter public timeline to detect accounts that post tweets with blacklisted URLs [11, 31], and some monitor Twitter's official account for spam reporting, *@spam* [21].

Much preliminary work [2, 11, 16, 23, 28] relies on account features including the numbers of followers and friends, account creation dates, URL ratios, and tweet text similarities, which can be efficiently collected but easily fabricated. To avoid feature fabrication, recent work [21,31] relies on more robust features extracted from the Twitter graph. Yang *et al.* [31] focused on relations between spam nodes and their neighboring nodes such as a bi-directional link ratio and betweenness centrality, because spam nodes usually cannot establish strong relationships with their neighboring nodes. They also introduced other features based on timing and automation. Song *et al.* [21] considered the relations between spam senders and receivers such as the shortest paths and minimum cut, because spam nodes usually cannot establish robust relationships with their victim nodes. The extraction of these robust features, however, is time and resource consuming.

6.2 Suspicious URL Detection

Many suspicious URL detection schemes have been proposed. They can be classified into either static or dynamic detection systems. Some lightweight static detection systems focus on the lexical features of a URL such as its length, the number of dots, or each token it has [19], and also consider underlying DNS and WHOIS information [17, 18]. More sophisticated static detection systems, such as Prophiler [3], additionally extract features from HTML content and JavaScript codes to detect drive-by download attacks. However, static detection systems cannot detect suspicious URLs with dynamic content such as obfuscated JavaScript, Flash, and ActiveX content. Therefore, we need dynamic detection systems [4, 7, 24, 29, 30] that use virtual machines and instrumented Web browsers for in-depth analysis of suspicious URLs. Nevertheless, all of these detection systems may still fail to detect suspicious sites with conditional behaviors.

6.3 ARROW: Generating Signatures to Detect Drive-by Downloads

Zhang *et al.* have developed ARROW [32], which also considers a number of correlated URL redirect chains to generate signatures of drive-by download attacks. It uses honeyclients to detect drive-by download attacks and collect logs of HTTP redirection traces from the compromised honeyclients. From these logs, it identifies central servers that are contained in a majority of the HTTP traces to the same binaries and generates regular expression signatures using the central servers' URLs. ARROW also groups domain names with the same IP addresses to avoid IP fast flux and domain flux [12, 22].

Although the methods for detecting central servers in ARROW and for detecting entry point URLs in WARNINGBIRD are similar, there are three important differences between these two systems. First, ARROW's HTTP traces are redirect chains between malicious landing pages and malware binaries. Therefore, ARROW cannot be applied to detect other Web attacks, such as spam, scam, and phishing attacks, which do not have such redirect chains to enable the downloading of malware binaries. Moreover, if honeyclients cannot access malicious landing pages owing to conditional redirections, ARROW cannot obtain any HTTP traces. Second, ARROW focuses on how to generate the signatures of central servers that redirect visitors to the same malware binaries, whereas WARNINGBIRD focuses on how to measure the suspiciousness of entry point URLs. Third, ARROW relies on logs of HTTP traces to detect central servers. Therefore, it cannot detect suspicious URLs in real time. In contrast, WARNINGBIRD is a real-time system.

7 Conclusion

Previous suspicious URL detection systems are weak at protecting against conditional redirection servers that distinguish investigators from normal browsers and redirect them to benign pages to cloak malicious landing pages. In this paper, we propose a new suspicious URL detection system for Twitter, WARNINGBIRD. Unlike the previous systems, WARNINGBIRD is robust when protecting against

conditional redirection, because it does not rely on the features of malicious landing pages that may not be reachable. Instead, it focuses on the correlations of multiple redirect chains that share redirection servers. We introduced new features on the basis of these correlations, implemented a real-time classification system using these features, and evaluate the system's accuracy and performance. The evaluation results showed that our system is highly accurate and can be deployed as a real-time system to classify large samples of tweets from the Twitter public timeline. In the future, we will extend our system to address dynamic and multiple redirections. We will also implement a distributed version of WARNINGBIRD to process all tweets from the Twitter public timeline.

References

[1] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis. we.b: The web of short URLs. In *Int. World Wide Web Conf. (WWW)*, 2011.

[2] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on Twitter. In *Collaboration, Electronic messaging, Anti-Abuse and Spam Conf. (CEAS)*, 2010.

[3] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: A fast filter for the large-scale detection of malicious web pages. In *Int. World Wide Web Conf. (WWW)*, 2011.

[4] Capture-HPC. `https://projects.honeynet.org/capture-hpc`.

[5] Y.-W. Chen and C.-J. Lin. Combining SVMs with various feature selection strategies. In *Feature Extraction*, volume 207 of *Studies in Fuzziness and Soft Computing*, pages 315–324. 2006.

[6] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia. Who is tweeting on Twitter: Human, bot, or cyborg? In *Annual Computer Security Applications Conf. (ACSAC)*, 2010.

[7] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Int. World Wide Web Conf. (WWW)*, 2010.

[8] P. Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies (PET)*, 2010.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[10] Google. Google safe browsing API. `http://code.google.com/apis/safebrowsing`.

[11] C. Grier, K. Thomas, V. Paxson, and M. Zhang. @spam: The underground on 140 characters or less. In *ACM Conf. Computer and Communications Security (CCS)*, 2010.

[12] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *Network and Distributed System Security Symp. (NDSS)*, 2008.

[13] P. Jaccard. The distribution of flora in the alpine zone. *The New Phytologist*, 11(2):37–50, 1912.

[14] A. Kapravelos, M. Cova, C. Kruegel, and G. Vigna. Escape from monkey island: Evading high-interaction honeyclients. In *SIG SIDAR Conf. Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2011.

[15] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *Int. World Wide Web Conf. (WWW)*, 2010.

[16] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: Social honeypots + machine learning. In *ACM SIGIR Conf.*, 2010.

[17] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD)*, 2009.

[18] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious URLs: An application of large-scale online learning. In *Int. Conf. Machine Learning (ICML)*, 2009.

[19] D. K. McGrath and M. Gupta. Behind phishing: An examination of phisher modi operandi. In *USENIX Workshop Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[20] M. A. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt. Trends in circumventing web-malware detection. Technical report, Google, 2011.

[21] J. Song, S. Lee, and J. Kim. Spam filtering in Twitter using sender-receiver relationship. In *Int. Symp. Recent Advances in Intrusion Detection (RAID)*, 2011.

[22] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: Analysis of a botnet takeover. In *ACM Conf. Computer and Communications Security (CCS)*, 2009.

[23] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Annual Computer Security Applications Conf. (ACSAC)*, 2010.

[24] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and evaluation of a real-time URL spam filtering system. In *IEEE Symp. Security and Privacy (Oakland)*, 2011.

[25] K. Thomas, C. Grier, V. Paxson, and D. Song. Suspended accounts in retrospect: An analysis of twitter spam. In *Internet Measurement Conf. (IMC)*, 2011.

[26] TweetAttacks. Twitter marketing software that breaks the limits. `http://tweetattacks.com`.

[27] Twitter Developers. Streaming API. `https://dev.twitter.com/docs/streaming-api`.

[28] A. Wang. Don't follow me: Spam detecting in Twitter. In *Int. Conf. Security and Cryptography (SECRYPT)*, 2010.

[29] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with Strider HoneyMonkeys: Finding web sites that exploit browser vulnerabilities. In *Network and Distributed System Security Symp. (NDSS)*, 2006.

[30] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phising pages. In *Network and Distributed System Security Symp. (NDSS)*, 2010.

[31] C. Yang, R. Harkreader, and G. Gu. Die free or live hard? empirical evaluation and new design for fighting evolving Twitter spammers. In *Int. Symp. Recent Advances in Intrusion Detection (RAID)*, 2011.

[32] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee. ARROW: Generating signatures to detect drive-by downloads. In *Int. World Wide Web Conf. (WWW)*, 2011.