WARNINGBIRD: A Near Real-time Detection System for Suspicious URLs in Twitter Stream

Sangho Lee, *Student Member, IEEE,* and Jong Kim, *Member, IEEE*

Abstract—Twitter is prone to malicious tweets containing URLs for spam, phishing, and malware distribution. Conventional Twitter spam detection schemes utilize account features such as the ratio of tweets containing URLs and the account creation date, or relation features in the Twitter graph. These detection schemes are ineffective against feature fabrications or consume much time and resources. Conventional suspicious URL detection schemes utilize several features including lexical features of URLs, URL redirection, HTML content, and dynamic behavior. However, evading techniques such as time-based evasion and crawler evasion exist. In this paper, we propose WARNINGBIRD, a suspicious URL detection system for Twitter. Our system investigates correlations of URL redirect chains extracted from several tweets. Because attackers have limited resources and usually reuse them, their URL redirect chains frequently share the same URLs. We develop methods to discover correlated URL redirect chains using the frequently shared URLs and to determine their suspiciousness. We collect numerous tweets from the Twitter public timeline and build a statistical classifier using them. Evaluation results show that our classifier accurately and efficiently detects suspicious URLs. We also present WARNINGBIRD as a near real-time system for classifying suspicious URLs in the Twitter stream.

Keywords—Suspicious URL, Twitter, URL redirection, conditional redirection, classification

1 INTRODUCTION

Twitter is a famous social networking and information sharing service [2] 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, it will be distributed to all of her *followers* who have registered Alice as one of their friends. Instead of distributing a tweet to all of her followers, Alice can also 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 a URL with friends via tweets, they usually use URL shortening services [3] to reduce the URL length since tweets can contain only a restricted number of characters. `bit.ly` and `tinyurl.com` are widely used services, and Twitter also provides a shortening service `t.co`.

Owing to the popularity of Twitter, malicious users often try to find a way to attack it. The most common forms of Web attacks, including spam, scam, phishing, and malware distribution attacks, have also appeared on Twitter. Because tweets are short in length, attackers use shortened malicious URLs that redirect Twitter users to external attack servers [4]–[9].

To cope with malicious tweets, several Twitter spam detection schemes have been proposed. These schemes can be classified into account feature-based [6], [10]–[12], relation feature-based [13], [14], and message feature-based [15] 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 a Twitter graph, however, requires a significant amount of time and resources as a Twitter graph is tremendous in size. The message feature-based scheme focused on the lexical features of messages. However, spammers can easily change the shape of their messages.

A number of suspicious URL detection schemes [4], [16]–[20] have also been introduced. They use static or dynamic crawlers, and they may be executed in virtual machine honeypots, such as Capture-HPC [21], HoneyMonkey [22], and Wepawet [23], to investigate newly observed URLs. These schemes classify URLs according to several features including lexical features of URLs, DNS information, URL redirections, and the HTML content of the landing pages. Nevertheless, malicious servers can bypass an 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

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-2012-H0301-12-3002) and World Class University program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea(R31-10100).
A preliminary version of this paper was presented at NDSS 2012 [1].

- *S. Lee is with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea. e-mail: sangho2@postech.ac.kr.*
- *J. Kim is with the Division of IT Convergence Engineering, Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea. e-mail: jkim@postech.ac.kr.*

malicious content only to normal browsers. Even if investigators use dynamic crawlers with (almost) all of the functionalities of real browsers, malicious servers may be able to recognize them through their IP addresses, user interaction, browser fingerprinting [24], or honeyclient detection techniques [25]. A recent technical report from Google has also discussed techniques for evading current Web malware detection systems [26]. Malicious servers can also employ temporal behaviors—providing different content at different times—to evade an investigation [19].

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 correlations of URL redirect chains extracted from a number of tweets. Because attacker's resources are generally limited and need to be reused, their URL redirect chains usually share the same URLs. We therefore created a method to detect correlated URL redirect chains using such frequently shared URLs. By analyzing the correlated URL redirect chains and their tweet context information, we discover several features that can be used to classify suspicious URLs. We collected a large number of tweets from the Twitter public timeline and trained a statistical classifier using the discovered features. The trained classifier is shown to be accurate and has low false positives and negatives.

The contributions of this paper are as follows:

- We present a new suspicious URL detection system for Twitter that is based on the correlations of URL redirect chains, which are difficult to fabricate. The system can find correlated URL redirect chains using the frequently shared URLs and determine their suspiciousness in almost real time.
- We introduce new features of suspicious URLs: some of which are newly discovered and while others are variations of previously discovered features.
- We present the results of investigations conducted on suspicious URLs that have been widely distributed through Twitter over several months.

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

2 CASE STUDY AND DATA ANALYSIS

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 was active until August 2011. We used a one percent of a sample of tweets collected on July 11, 2011, to conduct an in-depth analysis of the site (Fig. 1). `blackraybansunglasses.com` has a page, `redirect.php`, which 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 the sampled 232,333 tweets with URLs. 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 and 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 spam pages.

Another interesting point about `blackraybansunglasses.com` is that it uses the Twitter Web interface. Conventional Twitter spam detection schemes usually assumed that many spammers would use Twitter APIs to distribute their spam tweets. Advanced Twitter spammers, however, no longer rely on Twitter APIs, because they know that using APIs will distinguish their tweets from normal tweets. For instance, `tweetattacks.com` [27] sells a Twitter spam program that uses the Web interface to deceive spam receivers and to circumvent API limits.

2.2 24newspress.net

We also consider a suspicious site `24newspress.net`. We first found this site at the end of June 2011 and it was active until October 2011. We used one percent of the tweet samples collected on July 23, 2011, to conduct an in-depth analysis of the page (Fig. 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 [28], [29]. 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 the mobile Twitter Web interface to distribute its spam tweets.

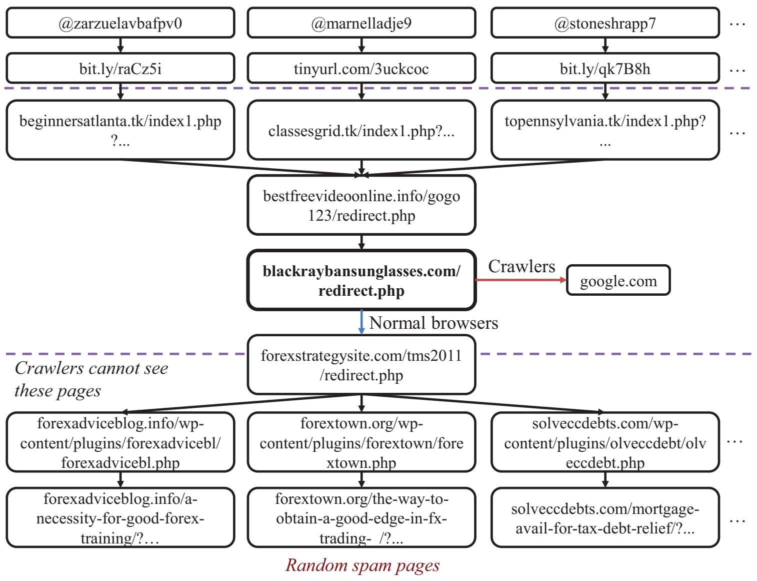

@zarzuelavbafpv0
@marnelladje9
@stoneshrapp7
bit.ly/raCz5i
tinyurl.com/3uckcoc
bit.ly/qk7B8h
beginnersatlanta.tk/index1.php?...
classesgrid.tk/index1.php?...
topennsylvania.tk/index1.php?...
bestfreevideoonline.info/gogo123/redirect.php
blackraybansunglasses.com/redirect.php
Crawlers
google.com
Normal browsers
Crawlers cannot see these pages
forexstrategysite.com/tms2011/redirect.php
forexadviceblog.info/wp-content/plugins/forexadvicebl/forexadvicebl.php
forextown.org/wp-content/plugins/forextown/forextown.php
solveccdebts.com/wp-content/plugins/olveccdebt/olveccdebt.php
forexadviceblog.info/a-necessity-for-good-forex-training/?...
forextown.org/the-way-to-obtain-a-good-edge-in-fx-trading-/?...
solveccdebts.com/mortgage-avail-for-tax-debt-relief/?...
Random spam pages

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

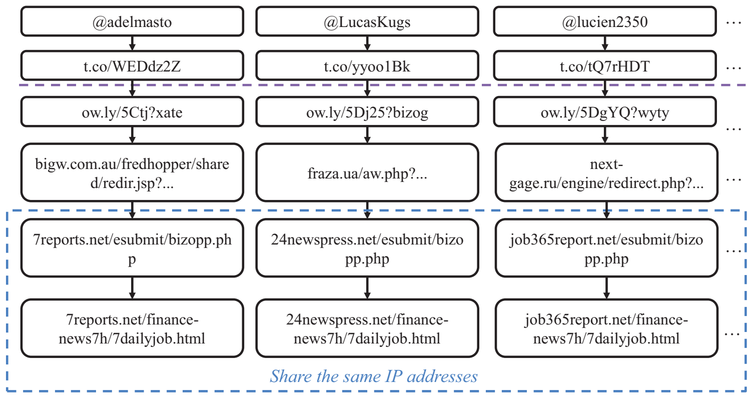

@adelmasto
@LucasKugs
@lucien2350
t.co/WEDdz2Z
t.co/yyoo1Bk
t.co/tQ7rHDT
ow.ly/5Ctj?xate
ow.ly/5Dj25?bizog
ow.ly/5DgYQ?wyty
bigw.com.au/fredhopper/shared/redir.jsp?...
fraza.ua/aw.php?...
next-gage.ru/engine/redirect.php?...
7reports.net/esubmit/bizopp.php
24newspress.net/esubmit/bizopp.php
job365report.net/esubmit/bizopp.php
7reports.net/finance-news7h/7dailyjob.html
24newspress.net/finance-news7h/7dailyjob.html
job365report.net/finance-news7h/7dailyjob.html
Share the same IP addresses

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

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 (Table 1).

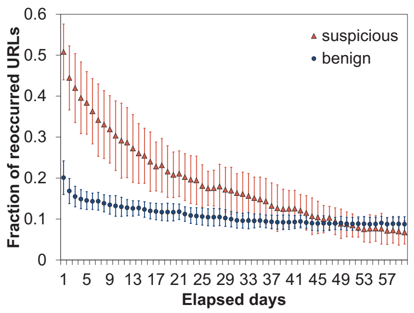


Fig. 3. Average fraction of reoccurred URL redirect chains in September 2011 during the next 60 days

2.4 Reoccurrences of URL Redirect Chains

We also discovered that suspicious URL redirect chains have frequently reoccur in the Twitter public timeline over several days. To verify the reoccurrence of suspicious URL redirect chains, we extracted the benign (posted by active accounts) and suspicious (posted by suspended accounts [30]) URL redirect chains for each day in September 2011, and checked the average number of repetitions of the extracted URL redirect chains during the ensuing 60 days. Let $\mathcal{D}$ denotes a set of days in September 2011, $\mathbf{B}(d_i)$ denotes a set of benign entry point URLs on d_i, $\mathbf{S}(d_i)$ denotes a set of suspicious entry point URLs on $d_i \in \mathcal{D}$, and $\mathbf{A}(d_i)$ denotes a set of all entry point URLs on d_i. For each $d_{i,j} \in \{j \text{ days later from } d_i : 1 \leq j \leq 60\}$, we compute

TABLE 1
Domain names of frequent URL redirect chains on July 23–25, 2011

Rank	July 23	July 24	July 25
1	24newpress.net	24newspress.net	24newspress.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

the following equations:

$$\sum_{d_i \in \mathcal{D}} \left(\frac{|\mathbf{B}(d_i) \cap \mathbf{A}(d_{i,j})|}{|\mathbf{B}(d_i)|} \right) / |\mathcal{D}|,$$

$$\sum_{d_i \in \mathcal{D}} \left(\frac{|\mathbf{S}(d_i) \cap \mathbf{A}(d_{i,j})|}{|\mathbf{S}(d_i)|} \right) / |\mathcal{D}|,$$

$d_{i,j}$'s are between September 2 and November 29, 2011.

The results are shown in Fig. 3. On average, 27% of the suspicious and 13% of the benign URL redirect chains re-appeared during the next 30 days, and 19% of the suspicious and 11% of the benign URL redirect chains re-appeared during the next 60 days. This confirms that suspicious URLs are repeated more often over a longer period than benign URLs are. We also found that about 50 days later, suspicious URLs were repeated less frequently than benign URLs; we think that Twitter or other investigators eventually detected and blocked most of them at around that time. These frequent repetitions and long lifetimes of the suspicious URLs show that conventional methods cannot efficiently detect them.

2.5 Observations

From the 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 over several days. These characteristics form the basis for the features 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 conditional redirections. Consider a simple example of conditional redirections (Fig. 4), in which an attacker creates a long URL redirect chain using a public URL shortening service, such as `bit.ly` and `t.co`, as well

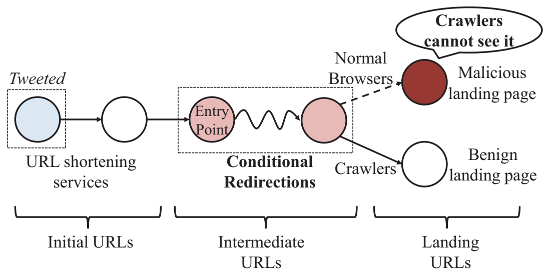


Fig. 4. Conditional redirection

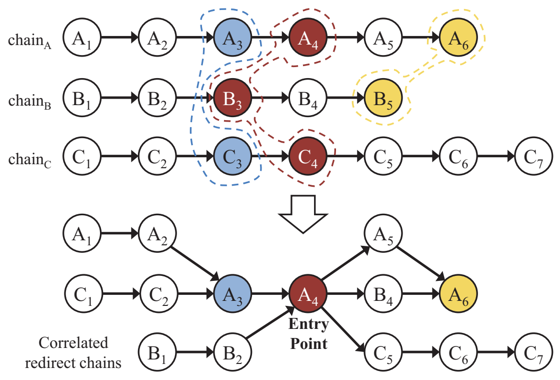


Fig. 5. Redirect chains and their correlation

as the attacker's own private redirection servers used 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 the intermediate URLs that are associated with private redirection servers. Some of these redirection servers check whether the current visitor is a normal browser or a crawler. If the current visitor seems to be a normal browser, the servers 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 that, as investigators, we cannot fetch the content of malicious landing URLs, because attackers do not reveal them to us. We also cannot rely on the initial URLs, as 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 may *reuse some of their redirection servers* when creating their redirect chains because they do not have infinite redirection servers (Section 2). Therefore, if we analyze several *correlated redirect chains* instead of an individual redirect chain, we can find the entry point of the intermediate URLs in these chains. Consider the three redirect chains shown in the top half of Fig. 5 which 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 Fig. 5) that share the same entry point URL, A_4 (because A_4 is the most frequent URL in these 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. Even the entry point, A_4, does not allow our crawler to visit the latter URLs, we could infer that the chains are suspicious because it has many initial URLs for the same landing (entry point in reality) URLs. Therefore, this correlation analysis can help in detecting suspicious URLs even when they perform conditional redirections.

3.2 System Details

Our system consists of four components: data collection, feature extraction, training, and classification (Fig. 6).

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 [31]. Whenever this component obtains a tweet with a URL, 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 it 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 of identical domains, finding entry point URLs, and extracting feature vectors. This component monitors the tweet queue to determine 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 several URLs share at least one IP address, it replaces their domain names with a list of domains 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 enables the detection of suspicious URLs that use several domain names to bypass the blacklisting.

Next, this 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 these tweets. It then discovers the most frequent URL in each URL redirect chain in the w tweets. The discovered URLs thus 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, the component finds URL redirect chains that contain the entry point URL, and extracts various features from these URL redirect chains along with the related tweet information (further details of these features are described 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 popular URL shortening sites, and some domains that we have manually verified.

Training: The training component has two subcomponents: retrieval of account statuses and training of the classifier. Because we use an offline supervised learning algorithm, the feature vectors for training are relatively older than feature vectors for classification. To label the training vectors, we use the Twitter account status; URLs from suspended accounts are considered malicious whereas URLs from active accounts are considered benign. We periodically update our classifier 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 an in-depth investigation.

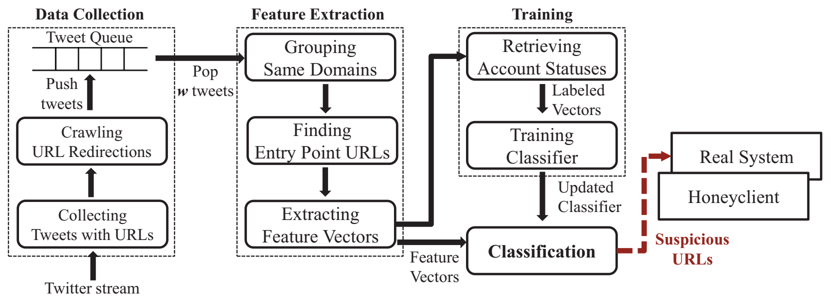


Fig. 6. System overview

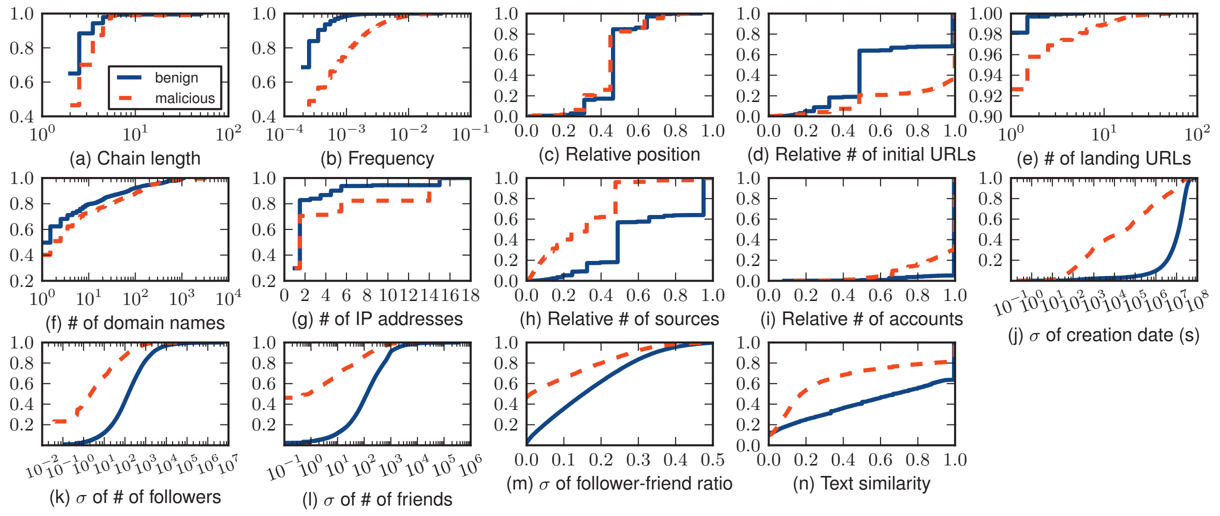


Fig. 7. CDF of features of entry point URLs that appear between September 2011 and October 2011. σ means standard deviation and it is a measure for checking the similarities between feature values.

3.3 Features

We introduce 14 features for classifying suspicious URLs on Twitter. [1] These features can be classified as features derived from correlated URL redirect chains and features derived from the related tweet context information. All statistics were checked in every 10,000 tweets and we only consider URLs had appeared more than once.

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 adismantling of their servers. Therefore, when an entry point URL is malicious, its chain length l may be longer than those of benign URLs. Fig. 7a shows that the lengths of the redirect chains of malicious entry point URLs are longer than those of benign URLs.

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 deemed suspicious, as discussed in Section 2. If the size of a tweet window is w and an entry point URL appears n times in the windows, this feature can be computed as $\frac{n}{w}$. Fig. 7b shows that the frequency of malicious entry point URLs is higher than that of benign entry point URLs.

Relative position of an entry point URL: Suspicious entry point URLs are not usually located at the end of a redirect chain since they have to conditionally redirect visitors to different landing URLs. Their positions are relative to the lengths of their redirect chains. Therefore, if the position of an entry point of a redirect chain of length l is p, this feature can be computed as $\frac{p}{l}$. Fig. 7c shows that the relative positions of malicious entry point URLs tend to be around the front part of the redirect chains.

Relative 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. The number of different initial URLs cannot exceed the number of times that their entry point URLs appear. Therefore, if the number of different initial

1. Compared with our previous study [1], we introduced two new features and used relative numbers for some features, e.g., the number of initial URLs, instead of absolute numbers.

URLs redirecting visitors to an entry point URL that appears n times is i, this feature can be computed as $\frac{i}{n}$. Fig. 7d shows that the relative number of initial URLs of malicious entry point URLs is larger than that of benign entry point URLs.

Number of different landing URLs: If the current entry point URL redirects visitors to more than one landing URLs, we can assume that the current entry point URL performs conditional redirection activities and may be suspicious. Unlike the initial URLs, we use an absolute number of different landing URLs as a feature since the existence of more than one landing URL is a suspicious sign regardless of the frequency of the entry point URL. Fig. 7e shows that the number of landing URLs of malicious entry point URLs is larger than that of benign entry point URLs.

Numbers of different domain names and IP addresses: As explained in Section 2.2, some spam sites use a large number of domain names and IP addresses to avoid blacklisting. Therefore, we use the number of different domain names and the number of different IP addresses of the entry point URLs as features. Figs. 7f and 7g show that the numbers of domain names and IP addresses of malicious entry point URLs are larger than those of benign entry point URLs.

3.3.2 Features Derived from Tweet Context Information

The features derived from the related tweet context information are variations of previously discovered features. However, unlike previous studies that have focused on the differences between malicious and benign accounts, we focused on the *similarity* of the features of accounts distributing the same entry point URLs. Preparing a large number of dissimilar Twitter accounts for distributing spam URLs becomes a burden to attackers; therefore, similarity checking is effective.

Relative number of different source applications: Sources are applications that upload the current entry point URL to Twitter. Attackers usually use the same source application as maintaining a number of different applications is difficult. Benign users, however, typically 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 source applications that post the same entry point URL that occurs n times is s, this feature can be computed as $\frac{s}{n}$. Fig. 7h shows that the relative number of source applications of malicious entry point URLs is smaller than that of benign URLs.

Relative 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 computed as $\frac{\alpha}{n}$. Fig. 7i shows that the relative number of accounts of malicious entry point URLs is smaller than that of benign URLs.

Similarity in the account creation dates: Attackers generally create a large number of Twitter accounts within a short period of time. Therefore, if the creation dates of the accounts that have uploaded the same entry point URL are similar, it may indicate that the current entry point URL is suspicious. We use the standard deviation of the account creation dates as a similarity measure (utilizing their Unix time for a comparison). Fig. 7j shows that the creation dates of accounts of malicious entry point URLs are more similar than those of benign entry point URLs.

Similarity in the number of followers and number of friends: The number of followers and number of friends associated with attacker's accounts are usually similar, because attackers use certain programs to manipulate these numbers. We again use the standard deviation to check for such numerical similarities. Figs. 7k and 7l show that the number of followers and the number of friends of accounts of malicious entry point URLs are more similar than those of benign entry point URLs.

Similarity in the follower-friend ratio: We define the follower-friend ratio as below:

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

As with the number of followers and number of friends, the follower-friend ratios of attacker accounts are similar. We thus use the similarity (standard deviation) of these ratios as a feature. Fig. 7m shows that the follower-friend ratios of accounts with malicious entry point URLs are more similar than those with benign entry point URLs.

Similarity of tweet texts: Tweeted texts containing the same URL are usually similar (e.g., retweets). Therefore, if the tweet texts associated with the same URL are different, we can assume that these tweets are related to suspicious behaviors because attackers usually want to change the appearance of malicious tweets that include the same malicious URL to evade detection. 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 [32], 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. Fig. 7n shows that the tweet texts of malicious entry point URLs are less similar than those of benign entry point URLs.

3.4 Feature Normalization

The normalization of features is important when the ranges of the feature values are different. For example, the similarity of tweet texts will lie between 0 and 1. However, the similarity of account creation dates (standard deviation) may be larger than 31,536,000 s (or two years). Without normalization, the similarity of account creation dates will cancel out the similarity of tweet texts. Among the various normalization methods available, we opted for z-score normalization owing to its robustness agianst unbounded ranges and outliers. [2] In z-score normalization, a value v_i of a feature F could be normalized as $v_i' = \frac{v_i - \bar{F}}{\sigma_F}$, where $\bar{F}$ and σ_F are the mean and standard deviation of F.

4 Evaluation

4.1 System Setup and Data Collection

Our system consists of two Intel Quad Core Xeon E5530 2.40GHz CPUs and 24 GiB of main memory. To collect the tweets, we used Twitter Streaming APIs [31]. Our accounts have aSpritzer access role, and thus we can collect about one percent of all tweets from the Twitter public timeline as samples. From April 8 to December 8, 2011 (245 days in total), we collected 59,056,761 samples of tweets with URLs. We observed about 240,000 tweets daily on average. Our system visited all the URLs in the tweets to collect the URL redirect chains. In addition, starting on July 23, our system collected the IP addresses of all URLs for the domain grouping. From the collected tweets, we found 13,261,069 unique Twitter accounts. Among them, 1,339,496 accounts (10.1%) were suspended as of January 15, 2012.

Twitter announced that it had started to wrap URLs with lengths longer than 19 characters using its URL shortening service `t.co` [33] from August 15, 2011 and that it started to wrap all URLs regardless of their length from October 10, 2011 [34]. We noticed that this additional layer of URL redirections affects our classification results; therefore, from August 15, 2011, we decided to remove the first `t.co` URLs in redirect chains.

4.2 Labeling Threshold

Labeling is essential for classification. Unfortunately, we were unable to find a suitable source for labeling our datasets, as many of the URLs in our datasets have not been listed on a public URL blacklist, such as the Google Safe Browsing API [35]. Therefore, instead of URL blacklists, we used Twitter account status information to label our datasets. That is, if some accounts had posted the same URLs and Twitter suspended the accounts later, we regarded the URLs as malicious. Otherwise, we regarded

2. We used min-max normalization that normalize features using the minimum and maximum values of them previously [1]. However, it was not a good choice because some features such as the similarity of account creation dates are unbounded.

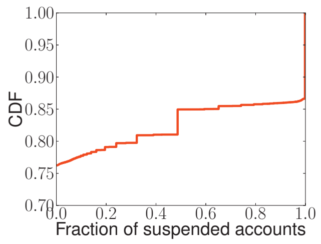


Fig. 8. Fraction of suspended accounts distributing the same URLs

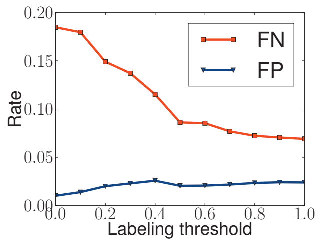


Fig. 9. False positive and negative rates according to labeling threshold

them as benign. Our treatment of URLs is acceptable as Thomas *et al.* have recently confirmed that most suspended accounts are spam accounts [30].

Since we rely on the results of Twitter's spam account detection system to label the collected datasets, one can argue that it just mimics the Twitter's detection system at most. However, most of our features are independent of the Twitter's rules [36] that focus on the suspicious characteristics of individual accounts, such as aggressive following, many tweets with (blacklisted) URLs, a small number of followers compared to the number of followings, and frequently blocked or reported by other users. Twitter can know whether an account violates the rules or not only after the account have performed a series of activities. However, unlike the rules, we focus on the characteristics of URL redirect chains and the similarity of a group of users who uploaded the same URL redirect chains; our system can immediately check them. We also verified that our system can detect suspicious accounts that Twitter cannot detect even several days later (Subsection 4.7). Therefore, we can say that our system is not a simple mimic of the Twitter's detection system. Because the Twitter's detection system had a time delay for suspicious account detection, we checked the status information of accounts at least one month later from their posting of tweets.

The remaining challenge is that of how to reduce the possibility of false labeling. For instance, let us assume that 30 suspended accounts and 20 active accounts distribute the same URL U_1, and the other 20 suspended accounts and 30 active accounts distribute the same URL U_2. Can we be assured that U_1 is malicious and U_2 is benign? If we treat a URL as malicious if at least one suspicious account posted it [3], we can capture many suspicious URLs but false positives increase. In contrast, if we treat a URL as benign if at least one active account posted it, we can reduce false positives but we could miss many real suspicious URLs. Moreover, unlike suspended accounts, we could not guarantee that all of the active accounts are not spam accounts because the Twitter's detection system is not perfect (Subsection 4.7).

To solve this problem, we need to define a reasonable threshold value that decides whether a URL is malicious

3. As our previous study [1] does.

TABLE 2
Training and test datasets

Dataset	Period	Benign	Malicious	Total
Training	September	78,982	14,885	93,867
	October	77,914	12,065	89,979
Total		156,896	26,950	183,846
$\text{Test}_{\text{past}}$	August	89,543	21,368	110,911
$\text{Test}_{\text{future}}$	November	81,742	13,132	94,874

TABLE 3
Comparing classifiers within a 10-fold cross validation

		%		
Classifier	AUC	Accuracy	FP	FN
L2R LR (primal)	0.9000	91.90	1.56	6.54
L2R L2-loss SVC (dual)	0.8995	91.79	1.49	6.72
L2R L2-loss SVC (primal)	0.8973	91.76	1.50	6.74
L2R L1-loss SVC (dual)	**0.9028**	**91.87**	**1.13**	**7.01**
L1R L2-loss SVC (primal)	0.8984	91.78	1.56	6.10
L1R LR (primal)	0.9007	91.91	1.27	6.52
L2R LR (dual)	0.9020	91.96	1.54	6.51

or benign. We used tweets collected between July 23 and August 8, 2011 to ascertain the threshold value. For each group of accounts that distributed the same entry point URLs, we determined what portion were suspended accounts (Fig. 8). Approximately 76% of the entry point URLs had no relationship with the suspended accounts while another 13% of them were distributed solely by suspended accounts. We thus need to assess the remaining (approximately) 11% of entry point URLs. Fig 8 shows that the CDF sharply increases when the fraction of suspended accounts is 50%. Therefore, intuitively, we can assume that 50% is a good candidate. To verify this assumption, we trained an L1-regularized logistic regression algorithm [37] with the dataset where its labels were determined according to the fractions of suspended accounts. We then checked the false-positive and false-negative rates within 10-fold cross validation (Fig 9). The false-positive rates slightly increased according to the fraction of suspended accounts. In contrast, the false-negative rates substantially decreased in accordance with the fraction of suspended accounts, especially when the fraction was 50%. As a result, we choose 50% as the value for the labeling threshold.

4.3 Training and Testing Classifiers

We used sample tweets collected between September 2011 and October 2011 to train the classification models and sample tweets collected during August 2011 and during November 2011 for testing the classifier using older and newer datasets, respectively. From the training dataset, we found 183,846 entry point URLs that appeared *more than once* in every 10,000 consecutive sample tweets. Among them, 156,896 entry point URLs were benign and 26,950 entry point URLs were malicious. We also used the account status information to label the test dataset; the results are shown in Table 2.

TABLE 4
Classification accuracy of test datasets

		%		
Dataset	AUC	Accuracy	FP	FN
$\text{Test}_{\text{past}}$	0.8937	88.00	0.83	11.16
$\text{Test}_{\text{future}}$	0.8960	91.53	1.23	7.24

We used the LIBLINEAR library [37] to implement our classifier. We compared seven classification algorithms, and selected an L2-regularized L1-loss support vector classification (SVC) algorithm, since it shows the highest AUC and the lowest FP with the training dataset, experimentally. [4] Table 3 shows the results; here, LR is an abbreviation of logistic regression, SVC is support vector classification, AUC is area under the ROC curve, FP is false positive, FN is false negative, L1R and L2R are L1- and L2-regularized, and primal and dual represent functions that determine termination of training. Standard deviations of the AUC were 0.0029–0.0032, those of the accuracy were 0.17%–0.20%, those of the FP were 0.05%–0.09%, and those of the FN were 0.18%–0.19%.

We could further reduce the value of the FP by increasing the weight value of the benign samples to penalize them (since the number of benign samples is fairly larger than the number of malicious samples). We used a weight value of 1.1 for benign samples; finally, we obtained 0.95% FP, 7.33% FN, 91.71% accuracy, and 0.9027 AUC. All the training and 10-fold cross validation could be done in less than several seconds in our system. Therefore, the training time is negligible.

We also used two test datasets representing past and future values to evaluate the accuracy of our classifier (Table 2). Regardless of whether the test datasets represented past or future values, our classifier achieved a relatively high accuracy, and few false positives and false negatives (Table 4). As a result, we concluded that our features could endure about one month time differences.

4.4 Feature Comparisons and Variations

We used the F-score to evaluate and compare the features of our scheme [38]. The F-score of a feature represents its degree of discrimination. Features with large F-scores can split benign and malicious samples better than features with small F-scores. Although the F-score does not reveal the mutual information between features, it is simple and quite effective in many cases. The F-scores show that the similarity of the account creation dates, the relative number of source applications, and the relative number of initial URLs are important features (Table 5). We also verified that the similarity of the number of friends and followers, and the relative number of

4. Since all classifiers we tested have the similar results, choosing one of them is acutally not an important problem.

TABLE 5
F-scores of training dataset

Feature	F-score
Similarity of account creation dates	0.1360
Relative number of sources	0.1278
Relative number of initial URLs	0.0659
Similarity of tweet texts	0.0547
Similarity of follower-friend ratios	0.0531
Frequency of entry point URL	0.0393
Number of landing URLs	0.0277
URL redirect chain length	0.0142
Number of IP addresses	0.0091
Relative position of entry point URL	0.0043
Similarity of the number of friends	0.0029
Similarity of the number of followers	0.0008
Number of domain names	0.0006
Number of accounts	0.0005

Twitter accounts are less important since attackers can manipulate the number of their friends and use a large number of bot accounts to distribute URLs. Moreover, we noticed that the number of different domain names is not important since we had already grouped domain names that share the same IP addresses.

Since Twitter is an evolving system, the features of accounts and URLs on the system could change with time. To know how they had been changed during our data collection periods, we checked the F-scores of our features in each month between May 2011 and November 2011, and compared six features that had high F-scores in some of the months (Fig. 10). [5] The F-scores of the similarity of account creation dates and the relative number of initial URLs had not much changed during the months. This is because the differences between the average feature values of them had not much changed (Fig. 11). On the other hand, the F-scores of the relative number of source applications and the frequency of entry point URLs had increased during the months owing to the reduced number of malicious applications and the reduced frequency of benign URLs. We think the reasons why they reduced are Twitter's efforts to reduce the number of malicious applications and less sampled tweets containing the same benign URLs due to the continuous growth of the number of tweets. Unfortunately, the F-scores of the similarity of the follower-friend ratios and the length of URL redirect chains had decreased during the months. Fig. 11 shows that the standard deviation of the follower-friend ratios of malicious accounts had increased during the months; it implies that attackers had changed the characteristics of their accounts to avoid detection. The figure also shows that the lengths of malicious URL redirect chains had decreased during the months; two possible explanations are i) attackers really had reduced the lengths of redirect chains because too long chains could be treated as malicious, or ii) they had applied dynamic redirections to prevent simple static

5. Since we use different datasets, labeling method, normalization method, and modified features, the resulting F-scores differ from the old results [1].

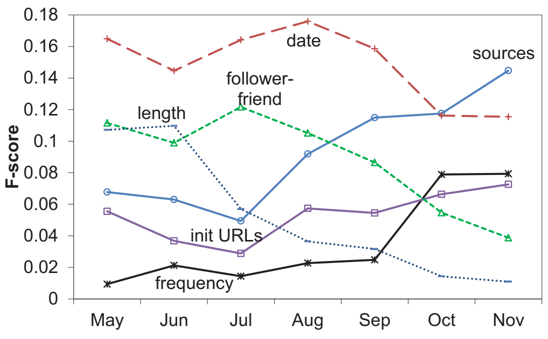


Fig. 10. F-score variations

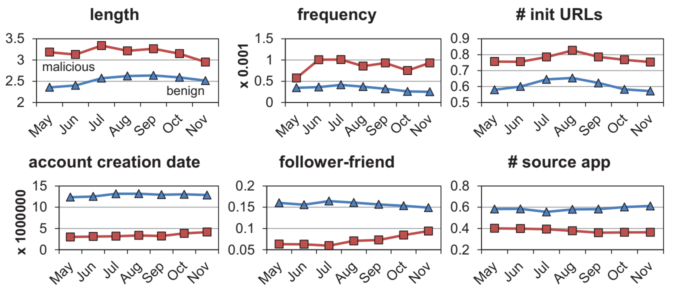


Fig. 11. Variations of average values of features

crawlers such as ours. The variations of F-scores and average feature values show that we need to periodically update our classifiers to cope with continuously changing circumstance of Twitter.

4.5 Running Time

We evaluated the running time of our system. First, we compared the running time of each component of the system—domain grouping; feature extraction, including the detection of entry points; and classification—in a single window of collected tweets that varied in size. Even if the window size grew to 100,000, which can contain of about 10% of all tweets with URLs per hour, the running time was only 6.9 min (Fig. 12). Next, we estimated 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. For a window size of 100,000, we needed 28.309 ms to process a single URL (Table 6)—indicating that 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.

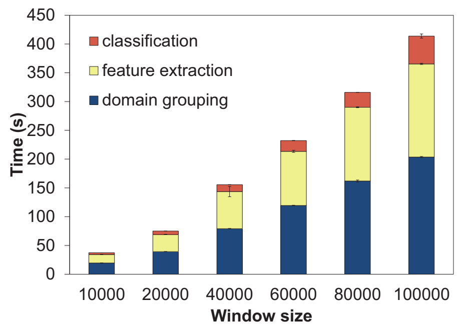


Fig. 12. Running time for each component to process a tweet window

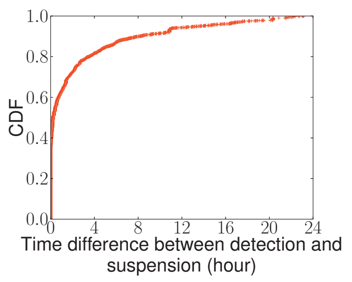


Fig. 13. Time difference between WarningBird's detection of suspicious accounts and Twitter's suspension within a day

TABLE 6
Required time to classify a single URL (a window size is 100,000 and 100 concurrent crawling threads)

Component	Avg. running time (ms)
Redirect chain crawling	24.202
Domain grouping	2.003
Feature extraction	1.620
Classification	0.484
Total	**28.309**

4.6 Online Detection

We also developed an online WarningBird system. The online 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 10,000 and s at 2,000. About every 12 min, the online version of WarningBird returns suspicious URLs that have appeared in the previous hour—near real time detection. Because our system can process 10,000 collected tweets in less than one minute (Fig. 12), we can detect suspicious URLs with only one-minute time lags. In addition, we could set s at 200 to detect suspicious URLs about every 1.2 min. However, because we do not want to make our system heavily burdened, we have not use such parameter.

4.7 Comparison with Twitter's Detection System

We compared the efficiency of WarningBird with that of Twitter's detection system. For the comparison, we sampled 14,905 accounts detected by our online WarningBird system between September 1, 2011 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, for one day, until it was suspended. Among the sampled accounts, 5,380 accounts were suspended within a day; 37.3% of them were suspended within a minute, another 44.3% of them were suspended within four hours, and the remaining 18.4% of them were suspended within a day (Fig. 13). The average time difference was 13.5 min, which shows that our detection system is more efficient than that of Twitter. We also checked the status of the sampled accounts on October 28, 2011 to verify the accuracy of our system. Among the 14,905 accounts, Twitter had suspended 9,250 accounts. We then randomly selected 500 accounts from the remaining 5,655 active accounts to manually check how suspect they were. Among the 500 accounts, 320 accounts were suspicious. Therefore, the detection accuracy of our system given the sample data is about 86.3%.

4.8 Considerations of Compromised Accounts

Although WarningBird is suitable for detecting frequent suspicious URLs distributed by bot accounts (which are common on Twitter [30]), we need to consider more advanced attacks using compromised accounts [7]. We can classify compromised Twitter accounts into two types: (i) accounts authorizing malicious applications and (ii) accounts stolen by attackers. Twitter users may accidently (or intentionally) authorize malicious applications luring them with interesting advertisements, such as enticements to increase the number of their followers or notify them regarding their unfollowers. User accounts may also be stolen by attackers guesing or stealing their passwords. In such cases, five account-similarity-based features, i.e., the number of source applications and the similarities in the account creation dates, the number of followers, the number of friends, and the follower-friend ratio, are no longer effective.

We considered an extreme case: training classifiers while excluding the ineffective features from the training dataset. When the tweet window size is 10,000 and we consider URLs appeared $\geq$ 2 times in each window

TABLE 7
Top three F-scores considering compromised accounts and the number of repetitions

Repetition	Ben. vs. Mal.	F-score		
		# init. URLs	Sim. texts	# accounts
≥ 3	$39,728 : 15,399$	0.2337	0.0976	0.0110
≥ 4	$21,496 : 11,484$	0.2131	0.1215	0.1033

(the same parameters as the current implementation), FN is about 94.47% and FP is about 1.59%; it misses almost all suspicious URLs. However, when we consider URLs appeared ≥ 3 times in each window, FN becomes 41.49% and FP becomes 8.04%, and when we consider URLs appeared ≥ 4 times in each window, FN becomes 39.66% and FP becomes 7.94%. These results imply that WarningBird can detect suspicious URLs from compromised accounts when their frequency is high. We also verified that the F-scores of some features increase as the number of repetitions increase (Table 7). Surely, we could bring back these results—only consider highly frequent URLs—to increase the accuracy of the current WarningBird system; however, it will pass over low-rate attacks. For example, the number of suspicious URLs appeared exactly two times in each window is 11,551, which is about 42.86% of all malicious samples in the training dataset. We could not ignore such a large number of URLs. To detect the low-rate attacks, we need to increase the size of the tweet window or decrease the size of the sliding; however, they demand more resources. Therefore, we should consider the tradeoff.

5 Discussion

Dynamic redirection: Currently, WarningBird uses a static crawler written in Python. Because it can only handle HTTP redirections, it is ineffective on pages that have 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 fully retrieve redirect chains.

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 only 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 almost real time. The current implementation, however, cannot handle 100% of the Twitter public timeline. Therefore, we need to extend WarningBird to a distributed detection system, for instance, Monarch [19], 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 can use short redirect chains, change the position of their entry point URLs, reuse initial and landing URLs, or use a small number of different domain names and IP addresses. These modifications, paradoxically, would allow conventional 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 would 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 [13], [14], [19]. 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 a lot of resource and large financial investment on the part of the attackers.

Adaptation to the other services: Although WarningBird is designed for Twitter, with some simple modifications it can also be applied to other services that can monitor a continuous URL stream. For example, we can consider an e-mail service that continuously processes a large number of e-mails for its users. Its operators can collect and investigate e-mails containing URLs. When a proper number of such e-mails are collected, the URL-based features can be extracted, such as the length of the URL redirect chain, the frequency of entry point URLs, and the number of different initial and landing URLs. The operators can also extract other features from e-mail context information such as the number of senders and receivers, the number of mail servers and relay servers, and similarities in e-mail messages. Web forum services are also similar; as their operators can collect all posts and comments of users containing URLs and can extract URL-based features as well as other features including user IDs, IP addresses, and message similarities. We can modify WarningBird to use the above features for detecting suspicious URLs on those systems. A similar method can also be applied to other social networking services such as Facebook and Google+.

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 manually analyze the collected data [11], [12], some use *honey-profiles* to lure spammers [6], [10], some monitor the Twitter public timeline to detect accounts that post tweets with blacklisted URLs [7], [14], and yet others monitor Twitter's official account for spam reporting, *@spam* [13].

Many preliminary studies [6], [7], [10]–[12] rely 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 [13], [14] relies on more robust features extracted from the Twitter graph. Yang *et al.* [14] 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.* [13] 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. Account and relation feature-based schemes cannot detect spam messages from compromised accounts, because the compromised accounts have benign features. To solve this problem, Gao *et al.* [15] proposed a spam detection scheme using message-based features. They focused on the syntactic similarity of spam messages. Spammers, however, can easily fabricate syntactical features of their spam messages. In addition, studies on the ecosystem of Twitter spammers [39] and link farming attacks for increasing spammers' social influences [40] have been conducted.

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 [4], and also consider underlying DNS and WHOIS information [16], [17]. More sophisticated static detection systems, such as Prophiler [18], 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 [19]–[23] 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 [41], 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 merges domain names with the same IP addresses to avoid IP fast flux and domain flux [28], [29].

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 near real-time system.

7 Conclusion

Conventional suspicious URL detection systems are ineffective in their protection 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 proposed a new suspicious URL detection system for Twitter, called WarningBird. Unlike the conventional 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 the same redirection servers. We introduced new features on the basis of these correlations, implemented a near real-time classification system using these features, and evaluated the system's accuracy and performance. The evaluation results show that our system is highly accurate and can be deployed as a near 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] S. Lee and J. Kim, "WarningBird: Detecting suspicious URLs in Twitter stream," in *Proc. NDSS*, 2012.

[2] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. WWW*, 2010.

[3] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis, "we.b: The web of short URLs," in *Proc. WWW*, 2011.

[4] D. K. McGrath and M. Gupta, "Behind phishing: An examination of phisher modi operandi," in *Proc. USENIX LEET*, 2008.

[5] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on Twitter: Human, bot, or cyborg?" in *Proc. ACSAC*, 2010.

[6] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. ACSAC*, 2010.

[7] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: The underground on 140 characters or less," in *Proc. ACM CCS*, 2010.

[8] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru, "Phi.sh/$oCiaL: the phishing landscape through short URLs," in *Proc. CEAS*, 2011.

[9] F. Klien and M. Strohmaier, "Short links under attack: geographical analysis of spam in a URL shortener network," in *Proc. ACM HT*, 2012.

[10] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," in *Proc. ACM SIGIR*, 2010.

[11] A. Wang, "Don't follow me: Spam detecting in Twitter," in *Proc. SECRYPT*, 2010.

[12] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on Twitter," in *Proc. CEAS*, 2010.

[13] J. Song, S. Lee, and J. Kim, "Spam filtering in Twitter using sender-receiver relationship," in *Proc. RAID*, 2011.

[14] C. Yang, R. Harkreader, and G. Gu, "Die free or live hard? empirical evaluation and new design for fighting evolving Twitter spammers," in *Proc. RAID*, 2011.

[15] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," in *Proc. NDSS*, 2012.

[16] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious web sites from suspicious URLs," in *Proc. ACM KDD*, 2009.

[17] ——, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. ICML*, 2009.

[18] D. Canali, M. Cova, G. Vigna, and C. Kruegel, "Prophiler: A fast filter for the large-scale detection of malicious web pages," in *Proc. WWW*, 2011.

[19] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time URL spam filtering service," in *Proc. IEEE S&P*, 2011.

[20] C. Whittaker, B. Ryner, and M. Nazif, "Large-scale automatic classification of phising pages," in *Proc. NDSS*, 2010.

[21] Capture-HPC, "https://projects.honeynet.org/capture-hpc."

[22] 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 *Proc. NDSS*, 2006.

[23] M. Cova, C. Kruegel, and G. Vigna, "Detection and analysis of drive-by-download attacks and malicious JavaScript code," in *Proc. WWW*, 2010.

[24] P. Eckersley, "How unique is your web browser?" in *Proc. PET*, 2010.

[25] A. Kapravelos, M. Cova, C. Kruegel, and G. Vigna, "Escape from monkey island: Evading high-interaction honeyclients," in *Proc. DIMVA*, 2011.

[26] M. A. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt, "Trends in circumventing web-malware detection," Google, Tech. Rep., 2011.

[27] TweetAttacks, "Twitter marketing software that breaks the limits," http://tweetattacks.com.

[28] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *Proc. NDSS*, 2008.

[29] 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 *Proc. ACM CCS*, 2009.

[30] K. Thomas, C. Grier, V. Paxson, and D. Song, "Suspended accounts in retrospect: An analysis of twitter spam," in *Proc. ACM IMC*, 2011.

[31] Twitter Developers, "Streaming API," https://dev.twitter.com/docs/streaming-api.

[32] P. Jaccard, "The distribution of flora in the alpine zone," *The New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[33] Twitter Developers, "Next steps with the t.co link wrapper," https://dev.twitter.com/blog/next-steps-with-the-tco-link-wrapper.

[34] ——, "The t.co URL wrapper," https://dev.twitter.com/docs/tco-url-wrapper.

[35] Google, "Google safe browsing API," http://code.google.com/apis/safebrowsing.

[36] Twitter Help Center, "The Twitter rules," https://support.twitter.com/articles/18311-the-twitter-rules.

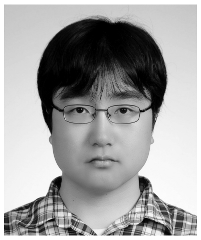
[37] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[38] Y.-W. Chen and C.-J. Lin, "Combining SVMs with various feature selection strategies," in *Feature Extraction*, ser. Studies in Fuzziness and Soft Computing, 2006, vol. 207, pp. 315–324.

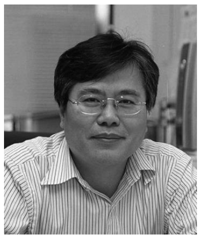
[39] C. Y. R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammers' social networks for fun and profit—a case study of cyber criminal ecosystem on Twitter," in *Proc. WWW*, 2012.

[40] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly, and K. P. Gummadi, "Understanding and combating link farming in the twitter social network," in *Proc. WWW*, 2012.

[41] J. Zhang, C. Seifert, J. W. Stokes, and W. Lee, "ARROW: Generating signatures to detect drive-by downloads," in *Proc. WWW*, 2011.



Sangho Lee received the B.S. in computer engineering from Hongik University, Korea, in 2006 and received the M.S. degree in computer science and engineering from Pohang University of Science and Technology (POSTECH), Korea, in 2008. He will receive the Ph.D. degree in computer science and enginnering from POSTECH in 2013. His research interests include Web and social network service security, applied cryptography, and privacy protection.



Jong Kim received the B.S. in electronic engineering from Hanyang University, Korea, in 1981, the M.S. degree in computer science from the Korean Advanced Institute of Science and Technology, Korea, in 1983, and the Ph.D. degree in computer engineering from Pennsylvania State University in 1991. He is currently a professor in the Division of IT Convergence Engineering, Pohang University of Science and Technology (POSTECH), Korea. From 1991 to 1992, he was a research fellow in the Real-Time Computing Laboratory of the Department of Electrical Engineering and Computer Science, University of Michigan. His major areas of interest are fault-tolerant computing, parallel and distributed computing, and computer security.

APPENDIX A
CONSIDERATIONS OF T.CO

Twitter had started to wrap long URLs in tweets using its URL shortening service `t.co` from August 15, 2011 (longer than 19-character URLs only) and all URLs from October 10, 2011. The path name of a `t.co` URL is composed of ≥ 8 alphanumeric characters, and depends on the original URL and the account who posts it. For example, if Alice posts two different tweets containing the same URL `http://abc.com/wow.php`, the two tweets will contain the same `t.co` URL (such as `http://t.co/wxyz5678`) that leads visitors to the original URL. When Bob retweets one of them, the `t.co` URL will be preserved. However, if Bob posts a different tweet containing the same URL, the tweet will contain a different `t.co` URL (such as `http://t.co/kl12oIz0`) for the original URL. This brings a bad effect to WARNINGBIRD because although several Twitter accounts post tweets containing the same benign URL, all of them have different initial `t.co` URLs; it could be misjudged as malicious. Therefore, we decided to exclude the initial `t.co` URLs from URL redirect chains from October 10, 2011. For the tweets between August 15 and October 9, 2011, we checked the character lengths of the second URLs in each redirect chain to determine whether the first `t.co` URLs had been added by Twitter or not.