IJDCST @ Dec, Issue- V-3, I-1, SW-39
ISSN-2320-7884 (Online)
ISSN-2321-0257 (Print)

ALERTBIRD: A System for Detecting Suspicious URLs in Twitter Stream

Shaik Shaheda[1], V. Padmaja[2]

[1]M.Tech (CS), Nimra College of Engineering and Technology, A.P., India.

[2]Asst. Professor, Dept. of Computer Science & Engineering, Nimra College of Engineering and Technology, A.P., India.

Abstract—Twitter is vulnerable 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 ALERTBIRD, 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 ALERTBIRD as a near real-time system for classifying suspicious URLs in the Twitter stream.

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

I. INTRODUCTION

Ever since Social Media introduced, it revolutionized the communication system. Among them, Twitter is one of the famous social networking and information sharing services [1] 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 [2] 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 [3].

To cope with malicious tweets, several Twitter spam detection schemes have been proposed. These schemes can be classified into account feature-based [4], relation feature-based [5] and message feature-based [6] 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 [7] have also been introduced. They use static or dynamic crawlers, and they may be executed in virtual machine honeypots, such as Capture-HPC, HoneyMonkey, and Wepawet, 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 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, or honeyclient detection techniques. A recent technical report from Google has also discussed techniques for evading current Web malware detection

systems. Malicious servers can also employ temporal behaviors—providing different content at different times—to evade an investigation [8].

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.

II. Related Work

A. 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 [9], [10], some use honey-profiles to lure spammers [3], [11], some monitor the Twitter public timeline to detect accounts that post tweets with blacklisted URLs [12], and yet others monitor Twitter's official account for spam reporting, @spam [13].

Many preliminary studies [3], [4], [5]–[8] 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] relies on more robust features extracted from the Twitter graph. Yang et al. [12] 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. [7] 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 and link farming attacks for increasing spammers' social influences have been conducted.

B. 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 [3], and also consider underlying DNS and WHOIS information [7]. More sophisticated static detection systems, such as Prophiler, 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 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.

C. ARROW: Generating Signatures to Detect Drive by Downloads

Zhang et al. have developed ARROW [14], 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. 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.

III. Proposed Work

D. 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. 1), 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 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.

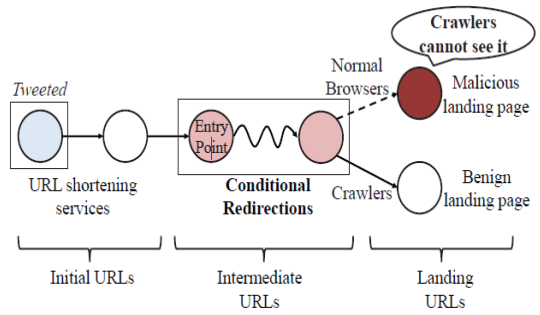

Figure 1 Conditional Redirection

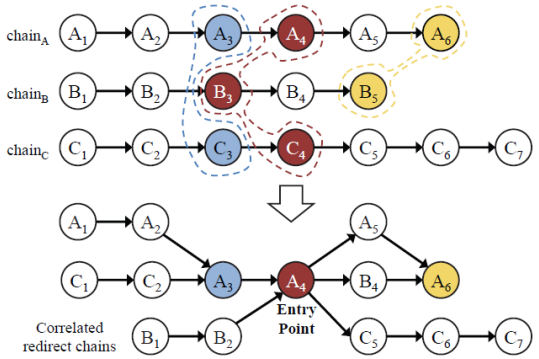

Figure 2 Redirect chains and their correlation

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. 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. 2 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.2) that share the same entry point URL, A4 (because A4 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 already considered. Even the entry point, A4, 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.

E. System Details

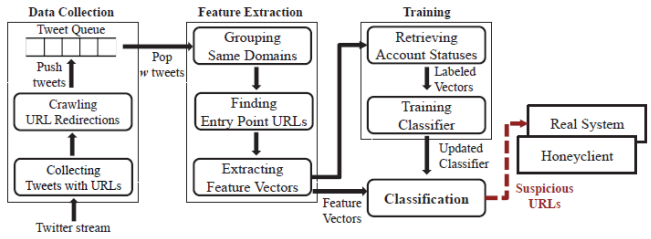

Figure 3 System Overview

Our system consists of four components: data collection, feature extraction, training, and classification (Fig. 3).

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.

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.

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.

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.

III. 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 ALERTGBIRD. Unlike the conventional systems, ALERTBIRD 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 ALERTBIRD to process all tweets from the Twitter public timeline.

References

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

[2] 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.

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

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

[5] J. Song, S. Lee, and J. Kim, "Spam filtering in Twitter using senderreceiver relationship," in Proc. RAID, 2011.

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

[7] 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.

[8] 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.

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

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

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

[12] 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.

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

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