A SURVEY ON WARNING BIRD: A NEAR REAL-TIME DETECTION SYSTEM FOR SUSPICIOUS URLS IN TWITTER STREAM

DR. MD.ALI HUSSAIN, N.NUTHAN, K.SUDHEER, CH.UDAYA MANI TEJA

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: suspiciously, Twitter, URL redirection, conditional redirection, classification.

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 Nuthan by mentioning this user by including @Nuthan 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] 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 honey client detection techniques. Malicious servers can also employ temporal behaviors providing different content at different times—to evade an investigation .In this paper, we propose WARNINGBIRD, the heels of the widespread adoption of web services such as social networks and URL shorteners, scams, phishing and malware have become regular threats. Despite

extensive research, email-based spam filtering techniques generally fall short for protecting other web services. To better address this need, we present Novel real-time system that crawls URLs as they are submitted to web services and determines whether the URLs direct to spam. We evaluate the viability of novel and the fundamental challenges that arise due to the diversity of web service spam. We show that Novel approach can provide accurate, real-time protection, but that the underlying characteristics of spam do not generalize across web services. In particular, we find that spam targeting email qualitatively differs in significant ways from spam campaigns targeting Twitter. We explore the distinctions between email and Twitter spam, including the abuse of public web hosting and redirect or services. Finally, we demonstrate Monarch's scalability, showing our system could protect a service such as Twitter—which needs to process 1 million URLs/day

Case Study : 2.1 blackraybansunglasses.com

We consider blackraybansunglasses.com, which is a suspicious site associated with spam tweets. We first encountered this site in April 2011 and it 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 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 Zarzu he or she will be redirected to a private redirection, such as beginnersatlanta.tk, which seems to be managed by the operator of blackraybansunglasses.com. The user will then be repeatedly goes to bestfreevideoonline.info. 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

Design Goals: To provide URL spam filtering as a service, we adopt six design goals targeting both efficiency and accuracy:

1) Real-time results. Social networks and email operate as near-interactive, real-time services. Thus, significant delays in filtering decisions degrade the protected service.

2) Readily scalable to required throughput. We aim to provide viable classification for services such as Twitter those receive over 15 million URLs a day.

3) Accurate decisions. We want the capability to emphasize low false positives in order to minimize mistaking non- spam URLs as spam.

4) Fine-grained classification. The system should be capable of distinguishing between spam hosted on Public services alongside non-spam content (i.e., classification of individual URLs rather than coarser-grained domain names).

5) Tolerant to feature evolution. The arms-race nature of spam leads to ongoing innovation on the part of spammers efforts to evade detection. Thus, we require the ability to easily retrain to adapt to new features.

6) Context-independent classification. If possible, decisions should not hinge on features specific to a particular service, allowing use of the classifier for different types of web services.

Proposed Mechanism

In this work we present the design and implementation of Novel system for filtering spam URLs in real-time as they are posted to web applications. Classification operates independently of the context where a URL appears (e.g., blog comment, tweet, or email), giving rise to the possibility of spam URL filtering as a service. We intend the system to act as a first layer of defense against spam content targeting web services, including social networks, URL shorteners, and email.

4.1 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 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.

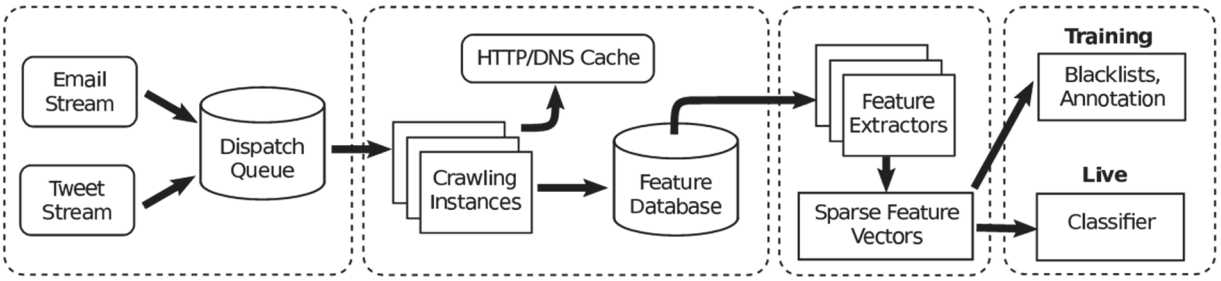
4.2 URL Aggregation: Our current architecture aggregates URLs from two sources for training and testing purposes: links emailed to spam traps operated by a number of major email providers and links appearing in Twitter's streaming API. In the case of Twitter, we also have contextual information about the account and tweet associated with a URL. However, we hold to our design goal of remaining agnostic to the source of a URL and omit this information during classification. We examine how removing Twitter-specific features affects accuracy as shown in figure 1.

4.3 Feature Collection: During feature collection, the system visits a URL with an instrumented version of the Google Chrome web browser to collect page content including HTML and page links, monitor page behavior such as pop-up windows and JavaScript activity, and discover hosting infrastructure. We explore the motivation behind each of these features. To ensure responsiveness and adhere to our goal of real-time, scalable execution, we design each process used for feature collection to be self-contained and parallelizable. In our current architecture, we implement feature collection using cloud machinery, allowing us to spin up an arbitrary number of collectors to handle the system's current workload.

4.4 Feature Extraction: Before classification, we transform the raw data generated during feature collection into a sparse feature vector understood by the classification engine. Data transformations include tokenizing URLs into binary features and converting HTML content into a bag of words. We permanently store the raw data, which allows us to evaluate new transformations against it over time.

4.5 Classification: The final phase of the system flow produces a classification decision. Training of the classifier occurs off-line and independent of the main system pipeline, leaving the live decision as a simple summation of classifier weights. During training, we generate a labeled data set by taking URLs found during the feature collection phase that also appear in spam traps or blacklists. We label these samples as spam, and all other samples as non-spam. Finally, in order to handle the millions of features that result and re-train daily to keep pace with feature evolution, we develop a distributed logistic regression, as discussed.

FIGURE 1



5.4 Proxy and Whitelist

To reduce network delay, Monarch proxies all outgoing network requests from a crawling instance through a single cache containing previous HTTP and DNS results. In addition, we employ a whitelist of known good domains and refrain from crawling them further if they appear during a redirect chain as a top-level window; their presence in IFrames or pop-up windows does not halt the surrounding collection process. Whitelists require manual construction and include trusted, high-frequency domains that do not support arbitrary user content. Our current whitelist contains 200 domains, examples of which include nytimes.com, flickr.com, and youtube.com. Whitelisted content accounts for 32% of URLs visited by our crawlers. The remaining content falls into a

long tail distribution of random hostnames, 67% of which appear once and 95% of which appear at most 10 times in our system. While we could expand the whitelist, in practice this proves unnecessary and provides little performance improvement.

Discussion

6.1Dynamic 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.

6.2 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.

6.3 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],. 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.

6.6 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 URLbased 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+.

Related Work

7.1 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 nonspam 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 and link farming attacks for increasing spammers' social influences have been conducted.

7.2 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 . 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.

7.3 Arrow: Generating Signatures to Detect Driveby Downloads

Zhang et al. have developed ARROW, which also considers a number of correlated URL redirect chains to generate signatures of drive-by download attacks. It uses honey clients to detect drive-by download attacks and collect logs of HTTP redirection traces from the compromised honey clients. 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.

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.
6. C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: The underground on 140 characters or less," in Proc. ACM CCS, 2010.
7. S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru,
8. "Phi.sh/$0CiaL: the phishing landscape through short URLs," in Proc. CEAS, 2011.
9. 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.

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

11. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on Twitter," in Proc. CEAS, 2010.
12. J. Song, S. Lee, and J. Kim, "Spam filtering in Twitter using sender receiver
13. 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 Twitterspammer s," in Proc. RAID, 2011.
15. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," in Proc. NDSS, 2012.

***

Assoc. Professor, Department of ECM, KL University, Guntur, A.P., India.
B.Tech, Department of ECM, KL University, Guntur, A.P., India.