The Role of the Underground Economy in Social Network Spam and Abuse

by

Kurt Thomas

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Fall 2013

The Role of the Underground Economy in Social Network Spam and Abuse

Abstract

The Role of the Underground Economy in Social Network Spam and Abuse

by

Kurt Thomas

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Vern Paxson, Chair

Online social networks have emerged as real-time communication platforms connecting billions of users around the globe. Implicit to the interactions within an online social network is the notion of trust; users create relationships with their friends and valued media outlets, in turn receiving access to content generated by each relationship. This trust however comes with a price. On the heels of the widespread adoption of online social networks, scams, phishing, and malware attacks conducted by criminals have become a regular occurrence. Such attacks exploit the trust users place in their relationships and the integrity of information found in online social networks.

The threat criminals pose to online social networks is exacerbated by the emergence of an *underground economy*—a digital network of criminals who buy and sell goods that directly enable the abuse of online social networks. Such services empower other miscreants to penetrate online social networks and engage with victims, while at the same time abstracting away the complexities of circumventing existing protection mechanisms employed by online social networks to hinder spam and abuse.

In this dissertation, we empirically analyze in both breadth and depth the range of threats currently targeting online social networks through the lens of Twitter. We map out the support infrastructure that is critical to online social network abuse, characterize the tools and techniques used to disseminate malignant content, and evaluate how such attacks ultimately realize a profit for the attackers involved. In the process, we argue that the for-profit infrastructure provided by the underground economy in the form of fake accounts and affiliate programs has become a fundamental weak point of abuse. Defenders should concentrate their efforts on disrupting these resources rather than fighting the subsequent, multifaceted abuse it enables such as scams, phishing, malware, and political attacks.

To aid in this effort, we develop two new strategies for preventing abuse in social networks. Our first defense identifies abusive links in online social networks (or any web service) before they are distributed to recipients. At its heart, this technique identifies common HTML content generated by affiliate programs and criminal hosting infrastructure which act as a buttress for the abuse ecosystem. Our second defense relies on directly engaging with the

underground economy that fuels online social network abuse to understand how millions of fake accounts are registered in an automated fashion. We leverage this understanding to detect abusive accounts at the time of their registration, preventing criminals from ever interacting with the legitimate users of online social networks.

In summary, this dissertation provides a data-driven analysis of spam and abuse on Twitter. We demonstrate that existing solutions for protecting online social networks fail to protect the millions of users that now rely on the technology as a global communication platform, exposing users to scams, phishing, malware, and even political censorship. By adopting the solutions presented in this dissertation, online social network operators can effectively defend both the ingress points of abuse—fraudulent and compromised accounts—and the egress points of abuse—spam links that direct victims to spamvertised products, fake software, clickfraud, banking theft, and malware that converts a victim's machine into a commodity for the underground economy. Such solutions afford online social network providers an opportunity to strike at the critical infrastructure that criminals rely on in order to monetize and abuse online social networks.

A creature who has spent his life creating one particular representation of his selfdom will die rather than become the antithesis of that representation.

- Scytale (Dune Messiah)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online social networks have emerged as real-time communication platforms connecting billions of users around the globe. As of December 2012, over 67% of adults in the United States participate in at least one online social network [23], while global adoption spans Europe, South America, Asia, and the Middle-East [94]. Popular web services such as Facebook and Twitter draw in over 1.1 billion [29] and 200 million [133] active users respectively, allowing participants to share stories, photos, and disseminate links. The ease of posting content to a global audience has democratized the spread of ideas and user-generated content [5, 6] with online social networks fueling political discourse [98, 93, 27], aiding in the organization of protests [108, 26], and facilitating access to critical information in the absence of credible news reports [85, 38].

Implicit to the interactions within an online social network is the notion of trust [79, 40]; users create relationships with their friends and valued media outlets, in turn receiving access to content generated by each relationship. This trust however comes with a price. On the heels of the widespread adoption of online social networks, scams, phishing, and malware attacks have become a regular occurrence. Such attacks exploit the trust users place in their relationships and the integrity of information found in online social networks. Salient examples of attacks include worms that hijack social networking accounts and spread to connected friends and followers [10, 117], phishing attacks that promise to increase a user's popularity [17, 84], and targeted attacks against news institutions and political dissidents [81, 64, 63].

The criminals operating in this space are financially motivated. Miscreants that disseminate spam for generic pharmaceuticals and knock-off designer products generate an estimated revenue between $12—$92 million each year [75], with similar criminals that dupe victim's into installing ineffectual software pulling in $5-116 million over the course of their operations [111]. The threat these criminals pose to online social networks is exacerbated by the emergence of an *underground economy*—a digital network of criminals who buy and sell goods that directly enable the abuse of online social networks. The miscreants operating in this space sell fake accounts, generate spurious relationships to inflate a user's popularity, as well as provide access to fundamental resources required for abuse such as compromised machines [11], CAPTCHA solving services [86], and proxies to diversify an attacker's IP addresses [49]; all for a small price. Such services empower other miscreants to penetrate online social networks and engage with victims, while at the same time abstracting away the complexities of circumventing existing protection mechanisms employed by online social networks to hinder spam and abuse.

Despite the gamut of threats facing online social networks, a comprehensive understanding of how criminals target online social networks and the role the underground economy plays in facilitating these attacks has remained elusive. The absence of this critical intelligence has forced online social network providers into an unending firefight. Operators deploy weak heuristics to detect abusive accounts (e.g., identifying the formation of too many relationships or sending unsolicited messages to victim's [122]), which criminals quickly adapt to in order to resume operation. This reactive development cycle never affords defenders an opportunity to step back and investigate the critical infrastructure that underpins the entire abuse ecosystem or to identify fundamental weaknesses in how criminals monetize online social networks that might otherwise fundamentally change the war against for-profit, criminal abuse.

In this dissertation, we empirically analyze in both breadth and depth the range of threats currently targeting online social networks through the lens of Twitter. We map out the support infrastructure that is critical to online social network abuse, characterize the tools and techniques used to disseminate malignant content, and evaluate how such attacks ultimately realize a profit for the attackers involved. In the process, we argue that the for-profit infrastructure provided by the underground economy in the form of fake accounts and affiliate programs has become a fundamental weak point of abuse. Defenders should concentrate their efforts on disrupting these resources rather than fighting the subsequent, multifaceted abuse it enables such as scams, phishing, malware, and political attacks.

To aid in this effort, we develop two new strategies for preventing abuse in social networks. Our first defense identifies abusive links in online social networks (or any web service) before they are distributed to recipients. At its heart, this technique identifies common HTML content generated by affiliate programs and criminal hosting infrastructure which act as a buttress for the abuse ecosystem [69]. Our second defense relies on directly engaging with the underground economy that fuels online social network abuse to understand how millions of fake accounts are registered in an automated fashion. We leverage this understanding to detect abusive accounts at the time of their registration, preventing criminals from ever interacting with the legitimate users of online social networks.

1.1 Characterizing Social Network Spam and Abuse

We begin with a characterization of spam and abuse targeting Twitter. Tapping into a real-time feed of content posted to Twitter, we find that 8% of 25 million URLs posted to the site over the course of a month direct to phishing, malware, and scams listed on popular domain blacklists. We find evidence that suggests the accounts involved in disseminating this nefarious content are in fact legitimate accounts that have been compromised and are now being puppeteered by criminals. Using clickthrough data, we analyze criminals' use of communication features unique to Twitter and the degree that they affect the success of spam. We find that Twitter is a highly successful platform for coercing users to visit spam pages, with a clickthrough rate of 0.13%, compared to much lower rates previously reported for email spam.

Given the absence of spam filtering on Twitter at the time of our study, we examine whether the use of URL blacklists would help to significantly stem the spread of abusive content. Our results indicate that blacklists are too slow at identifying new threats, allowing more than 90% of visitors to view a page before it becomes blacklisted. We also find that even if blacklist delays were reduced, the use by criminals of URL shortening services for obfuscation negates the potential gains unless tools that use blacklists develop more sophisticated spam filtering.

1.2 Social Spam: Tools, Techniques, and Monetization

We continue with an in-depth study of the tools, techniques, and support infrastructure that miscreants attacking online social networks rely upon. To perform our analysis, we identify over 1.1 million accounts suspended by Twitter for disruptive activities over the course of seven months. In the process, we collect a dataset of 1.8 billion tweets, 80 million of which belong to spam accounts. We use our dataset to characterize the behavior and lifetime of spam accounts, the campaigns they execute, and the wide-spread abuse of legitimate web services such as URL shorteners and free web hosting. We also identify an emerging marketplace of illegitimate programs operated by criminals that include Twitter account sellers, ad-based URL shorteners, and spam affiliate programs that help enable underground market diversification.

Our results show that 77% of spam accounts identified by Twitter are suspended within one day of their first tweet. Because of these pressures, less than 9% of accounts form social relationships with regular Twitter users. Instead, 17% of accounts rely on hijacking trends, while 52% of accounts use unsolicited mentions to reach an audience. In spite of daily account attrition, we show how five spam campaigns controlling 145 thousand accounts combined are able to persist for months at a time, with each campaign enacting a unique spamming strategy. Surprisingly, three of these campaigns send spam directing visitors to reputable store fronts, blurring the line regarding what constitutes spam on online social networks.

1.3 Emerging Threats: Censorship and Astroturfing

As online social networks emerge as an important tool for political engagement and dissent, services including Twitter have become regular targets of censorship. In the past, nation states have exerted their control over Internet access to outright block connections to social media during times of political upheaval. Parties without such capabilities may however still desire to control political expression. A striking example of such manipulation recently occurred on Twitter when an unknown attacker leveraged 25,860 fraudulent accounts to send 440,793 tweets in an attempt to disrupt political conversations following the announcement of Russia's parliamentary election results.

We undertake an in-depth analysis of the infrastructure and accounts that facilitated the attack. We find that miscreants leveraged the same spam-as-a-service market that fuels spam and abuse on Twitter to acquire thousands of fraudulent accounts which they used in conjunction with compromised hosts located around the globe to flood out political messages. Our findings demonstrate how malicious parties can adapt the services and techniques traditionally used by criminals to other forms of attack, including censorship. Despite the complexity of the attack, we show how Twitter's relevance-based search helped mitigate the attack's impact on users searching for information regarding the Russian election.

1.4 Developing Real-Time and Scalable Spam Detection

While spam in online social networks carries a number of similarities with traditional email-based spam, email-based spam filtering techniques generally fall short for protecting other web services, including Twitter. To better address this need, we present Monarch, a 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 Monarch and the fundamental challenges that arise due to the diversity of web service spam. We show that Monarch 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 redirector services. Finally, we demonstrate Monarch's scalability, showing our system could protect a service such as Twitter—which needs to process 15 million URLs/day—for a bit under $800/day.

1.5 Disrupting the Underground Account Marketplace

As web services such as Twitter, Facebook, Google, and Yahoo now dominate the daily activities of Internet users, cyber criminals have adapted their monetization strategies to engage users within these walled gardens. To facilitate access to these sites, an underground market has emerged where fraudulent accounts—automatically generated credentials used to perpetrate scams, phishing, and malware—are sold in bulk by the thousands. In order to understand this shadowy economy, we investigate the market for fraudulent Twitter accounts to monitor prices, availability, and fraud perpetrated by 27 merchants over the course of a 10-month period. We use our insights to develop a classifier to retroactively detect several million fraudulent accounts sold via this marketplace. During active months, the 27 merchants we monitor appeared responsible for registering 10—20% of all accounts later flagged for spam by Twitter, generating $127—459K for their efforts.

With Twitter's cooperation, we disable 95% of all fraudulent accounts registered by the merchants we track, including those previously sold but not yet suspended for spamming. Throughout the suspension process, we simultaneously monitor the underground market for any fallout. While we do not observe an appreciable increase in pricing or delay in merchant's delivering new accounts, we find 90% of all purchased accounts immediately after our actioning are suspended on arrival.

1.6 Summary

In summary, this dissertation provides a data-driven analysis of spam and abuse on Twitter. We demonstrate that existing solutions for protecting online social networks fail to protect the millions of users that now rely on the technology as a global communication platform, exposing users to scams, phishing, malware, and even political censorship. To overcome these limitations, we propose two new solutions: (1) analyzing the content that users post to social networks to detect abusive URLs (effectively the egress point of the majority of abuse in online social networks), and (2) infiltrating the underground economy to disrupt weak points of its organization, in particular depleting the stockpiles of fraudulent accounts registered by miscreants which would otherwise be sold and distributed to criminals (effectively the ingress point for the majority of abuse in online social networks).

Chapter 2

Social Networks, Abuse, and the Criminal Ecosystem

2.1 Social Network Organization and Interactions

We define online social networks as a graph consisting of *users* (e.g., nodes) who form *relationships* (e.g., edges) with each other. These relationships reflect both trust and common interests shared between close friends, mere acquaintances, and media outlets [79, 66]—often delineated as strong and weak *ties* [40, 36]. Such social connections are the primary means for *information diffusion*, the spread of information or the adoption of ideas throughout a social network [43, 6, 5]. Users share photos, videos, stories, and links with their immediate connections, which can then cascade to other users in the network via sharing from friends, friends of friends, or any number of hops away from the author of the original content.

Examples of large online social networks include Twitter and Facebook, which as of June 2013 have garnered an audience of 200 million [133] and 1.1 billion [29] active users respectively. In addition to providing the basic functionality of a social graph, these web services provide search capabilities, messaging applications, and other features which bypass the requirement of sharing a connection with a user in order to communicate or share content. Much of the research in this dissertation targets Twitter, and as such, we provide a brief overview of the online social networks core functionality.

Account: Any and all interactions within Twitter occur through a personalized account. Users can tailor basic properties of an account to suit their needs, such as including a profile image, providing a short biography, or listing a URL to an external webpage. Accounts are used to post and receive information, form relationships, and search Twitter, all of which we describe in detail. Accounts are uniquely identified by a *@username*, analogous to a email address. Accounts can be *public* and visible to any user within Twitter or *private*, where the account's controller maintains some privacy policy over who can access an account's associated content.

Tweets: A tweet is a colloquialism used by Twitter to describe a status update—analogous to an email's body—which is the primary means of sharing information within the web service. Twitter restricts these updates to 140 characters or less. To facilitate the posting of URLs in tweets, URL shortening services such as `t.co` and `bit.ly` are commonly used. These sites act as a redirectors, proxying requests from a short hashed code to a URL of arbitrary length.

Followers: An account's followers are the set of users that will receive a tweet once it is posted, akin to the `To` field of an email. Users must subscribe as an account's follower before receiving tweets; an account cannot force her messages to be viewed by other users.

Followings: Relationships in Twitter are not bidirectional, meaning a user can receive tweets from a friend without revealing their own tweets. Followings are the set of users an account subscribes to in order to obtain access to status updates.

Timelines: Each Twitter user is provided with a customized timeline of tweets aggregated from accounts a user follows. When accessing a timeline via the web, a single tweet contains the tweet text, an icon for the account that posted the tweet, the time posted, geolocation data, and the application used to post the tweet. The mobile experience for timelines provides the same basic information. Timelines are the primary means of receiving information from the Twitter network, with search, recommendations, and trends being the other mechanisms.

Mentions: Mentions, or replies, are a mechanism Twitter provides to address a particular user within the network. To mention a user, a tweet is composed that also contains the target's *@username*. Tweets that contain mentions are still publicly broadcast to all of a user's followers, but a notification is sent to the mentioned user (and the presence of the username makes it easier for the recipient to identify a message is directed at her). It is important to note that mentioning a user does not require the sender or receiver share a relationship in common; any party on Twitter can mention another party.

Example: `@justinbieber PLEASE FOLLOOWW MEEE!!! <3333`

Retweets: Retweets on Twitter are a form of attribution, where `RT @username` or `via @username` denote that the tweet text originally appeared from another account. Retweets build on the authority of another user and are used to increase the volume of followers who see a tweet, effectively forming an information cascade. Retweets are also used in calculating search result ranking, where more popular information cascades receive a higher ranking.

Example: `RT @JBieberCrewz: RT this if u <3 justin bieber`

Hashtags: In addition to mentioning users, tweets can include tags to arbitrary topics by including a hashtag *#topic*. If enough users pick up on the topic it will appear in the list of *trending topics*—popular breaking stories within Twitter—allowing tweets to be syndicated to all of Twitter. Hashtags can also be used as a seed to a search query. Hashtags are frequently used to organize conversations that defy social boundaries. The entire Twitter community can post to the same hashtag, allowing parties to communicate with one another without directly sharing social relationships.

Example: `Get free followers #FF #Follow Justin Bieber`

Direct Messages: Direct messages are private tweets sent between two users, effectively duplicating the functionality of email. Direct messages do not appear in a users timeline, but instead appear in a separate, private timeline. Direct messages require that the recipient follow the sender, in contrast to mentions which require no such relationship.

Example: `Be a BELIEBER, click on this URL: http:// ... <3333`

Favorites: Favorites are a way of positively ranking a tweet or bookmarking a tweet for later use (akin to Google's +1 or Facebook's Like button). When a user favorites another account's tweet, the other account's owner is notified. Similarly, Twitter provides *recommended content* to users, where favorited content is used to rank what tweets and accounts are recommended.

2.2 Components of Social Network Abuse

As web services such as Twitter and Facebook now dominate the daily activities of Internet users [2], cyber criminals have adapted their monetization strategies to engage users within these walled gardens. Attacks targeting online social networks require three components: (1) access to account credentials; (2) a mechanism to engage with legitimate users within the network (i.e. the victims that will be exploited to realize a profit); and (3) some form of monetizable content, typically in the form of a URL that directs a victim off of Twitter to a website that generates a profit via spamvertised products, fake software, clickfraud, banking theft, or malware that converts a victim's machine or assets (e.g., credentials) into a commodity for the underground economy. With respect to Twitter, the underpinnings of each of these components are outlined in Figure 2.1, which we use to guide discussion.

What becomes apparent from this taxonomy is that, while there are several ways to engage with victims (and more constantly emerge as new features are added), the ingress and egress points of abuse are much fewer. For this reason, this dissertation advocates the development of URL-based defenses and at-registration time defenses. Strangling those two choke points collapses all the other pain points of social network spam and abuse which are arguably harder to solve given the diverse ways legitimate users engage one another within social networks.

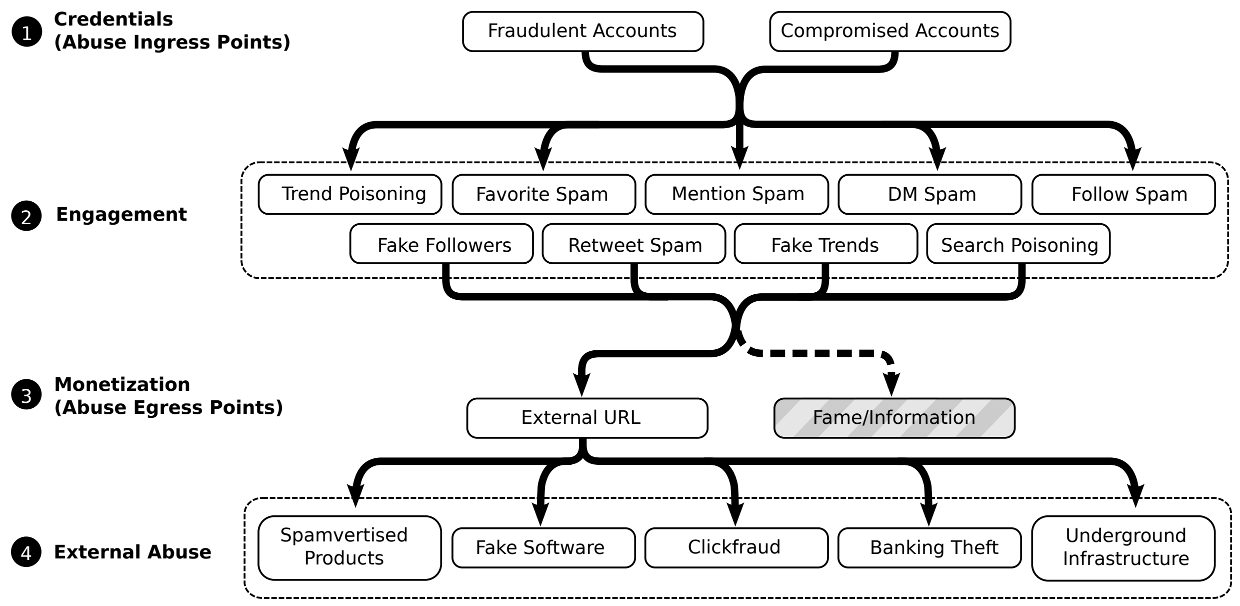


Figure 2.1: Components necessary to abuse an online social network. This includes credentials, a mechanism to engage with legitimate users, and finally a means of monetizing traffic, typically to some form of for-profit abuse.

2.2.1 Credentials

In order to interact with a site like Twitter, criminals must first obtain credentials for either new or existing accounts. This has lead to a proliferation of *fraudulent accounts*—automatically generated credentials used exclusively to disseminate scams, phishing, and malware [16]—as well as *compromised accounts*—legitimate credentials that have fallen into the hands of miscreants, which criminals repurpose for nefarious ends. Notable sources of compromise on Twitter include the brute force guessing of weak passwords [134], password reuse with compromised websites, as well as worms or phishing attacks that propagate within the network [46].

2.2.2 Engagement

Once an attacker has access to an online social network, they need some mechanism to engage with legitimate users. Any of the multitude of features on Twitter can be targets of abuse in a criminal's quest for drawing an audience. While its possible to solve one facet of abuse, criminals are constantly evolving how they engage with users to leverage new features added to social networks as well as to adapt to defense mechanisms employed by online social network operators. The result is a reactive development cycle that never affords defenders any reprieve. To illustrate this point, we list just some ways in which criminals engage with users.

Mention Spam: Mention spam consists of sending an unsolicited mention to a victim, bypassing any requirement of sharing a social connection with a victim. Spammers can either initiate a conversation or join an existing conversation a victim is having with her followers. The victim will then receive a notification she has been contacted, with the mention often containing a spam link.

Follow Spam: Follow spam occurs when criminals leverage an account to generate hundreds of relationships with legitimate users. The aim of this approach is to either have a victim reciprocate the relationship, in which case the criminals content will be syndicated to the victim, or at least have the victim view the criminal's account profile which often has a URL embedded in its biography section.

Direct Message Spam: Direct message spam (or DM spam) is identical to mention spam, but requires that a criminal's account be followed by a victim. As such, DM spam is typically used when an account has become compromised due to the challenges of fraudulent accounts obtaining relationships with legitimate users, discussed in Chapter 4.

Trending Poisoning: Rather than forming relationships with users or targeting single users, spammers can post tweets that contain popular keywords from trending topics. We call this form of attack trend poisoning. Trends result from spontaneous coordination between Twitter users as well as from breaking news stories. Users that explore these trends will receive a feed of legitimate tweets interspersed with spam.

Search Poisoning: Search poisoning is identical to trend poisoning, but instead of emerging topics typified by hashtags, spammers embed specific keywords or brands in their tweets such as "viagra" and "ipad". From there, users that search for information relevant to a keyword or brand will be exposed to spam.

Fake Trends: Fake Trends leverage the availability of thousands of accounts under the control of a single criminal to effectively generate a new trend. From there, victims looking at emerging content will be exposed to the criminal's message.

Favorite Spam: Favorite Spam relies on abusing functionality on Twitter which allows a user to favorite, or recommend, a tweet. Criminals will mass-favorite tweets from victims in the hopes they either reciprocate a relationship or view the criminal's account profile, just like follow spam.

Fake Followers: Fake Followers are distinct from follow spam, in that a criminal purchases relationships from the underground economy. The goal here is to inflate the popularity of a criminal's account, often for improving the ranking of the account's content in search results.

Retweet Spam: Retweet Spam entails hundreds of spam accounts all retweeting another account's tweet. As with fake followers, the goal is to inflate the importance of a tweet to improve its ranking in search results.

2.2.3 Monetization

In order to monetize a victim, users are typically funneled from Twitter to another website via a link. The landing page of this link attempts to sell victims a product, forward their traffic, steal the victims' credentials, or compromise the victims' machine. The exception to this approach is abuse that takes a more circumlocutious path towards profiting from victims. Examples of such exceptions are celebrities who buy—or inadvertently acquire—fake followers to inflate their popularity (thus achieving a payout from fame) [18] as well as politically-motivated attacks such as censoring speech or controlling the message surrounding emerging trends (where the payout is political capital or damage control) [64, 63]. While the latter attacks are realistic threats, as we will show, the vast majority of abuse currently targeting social networks is more criminal in nature.

2.3 Criminal Monetization of Abuse

Profit lies at the heart of the criminal abuse ecosystem. Monetization strategies form a spectrum between selling products to a user with the user's consent to stealing from a victim without consent. We provide a brief overview of these techniques, paying particular attention to the most notorious families of malware targeting each particular form of monetization.

2.3.1 Spamvertized Goods

Spam pertains to any form of bulk unsolicited messaging, including messages sent via email and social networks. Examples of well-understood email spam botnets include Rustock [56], Storm [65], and MegaD [15]. Monetization comes in many forms, including the sale of pharmaceuticals, replica goods, and pirated software [69].

2.3.2 Fake Software

Fake software (as opposed to pirated software and intellectual property theft) includes any malware that prompts a user to install or upgrade ineffectual software. The most prominent approach here is selling rogue antivirus [100, 111]. These programs prompt users to pay a one-time fee in order to remove non-existent malware infections, providing no protection in return.

2.3.3 Clickfraud

Clickfraud generates revenue by using automated bots (or by redirecting traffic generated by victims) to simulate legitimate traffic to pay-per-click advertisements [83, 113]. These ads typically appear on pages controlled by miscreants, while the ads are syndicated from advertising networks such as Google AdSense. Money is thus siphoned from advertisers into the hands of criminals.

2.3.4 Banking Theft

Information stealers such as Torpig [112], Zeus, and SpyEye harvest sensitive user data from compromised machines, including documents, passwords, and banking credentials. An attacker can then sell access to these accounts or liquidate the account's assets.

2.3.5 Commoditizing Compromised Hosts

Apart from directly monetizing victims or their traffic, a number of strategies have appeared where a victim's machine is compromised and then sold to other criminal activities.

Proxies & Hosting: For externally facing hosts, miscreants can convert compromised machines into bulletproof hosting services and proxy networks that are re-sold as tools for other miscreants [49]. These machines can in turn host vital C&C infrastructure, act as anonymizers, or simply provide a diverse pool of IP addresses to circumvent IP-based restrictions for sending spam or registering accounts.

Droppers: Rather than employing one of the aforementioned techniques, miscreants can sell successful infections to other parties on the underground market [11]. In some cases, this involves miscreants installing a *dropper* on a compromised host. This tool automatically contacts a *pay-per-install* provider for new binaries to install in exchange for a fee. We differentiate between *staged installers* where an initial infection is used to bootstrap updates or new features, and droppers, where control of a machine is re-sold to a second or even multiple parties.

2.4 Specialization Within the Criminal Ecosystem

At the center of the for-profit spam and malware ecosystem is an underground market that connects Internet miscreants with parties selling a range of specialized products and services including spam hosting [4, 49], CAPTCHA solving services [86], pay-per-install hosts [11], and exploit kits [42]. Even simple services such as garnering favorable reviews or writing web page content are for sale [128, 88]. Revenue generated by miscreants participating in this market varies widely based on business strategy, with spam affiliate programs generating

$12—$92 million [75] and fake anti-virus scammers $5-116 million [111] over the course of their operations.

Specialization within this ecosystem is the norm. Organized criminal communities include carders that siphon credit card wealth [32]; email spam affiliate programs [69]; and browser exploit developers and traffic generators [42]. These distinct roles allow miscreants to abstract away certain complexities of abuse, in turn selling their specialty to the underground market for a profit.

Within recent years a great deal of effort has been spent on studying the activities of underground economies, and in particular, the spam marketplace. Previous research has examined the hosting infrastructure of scams [4, 49], the organization of email spam campaigns [65], and the economic incentives of spam and malware [57, 111]. For online social networks, examinations of URL shorteners have focused on URL distribution [60], use for phishing [14], and as an aid to classification of spam [68].

2.5 Combating Social Network Spam and Abuse

2.5.1 Social Network Spam Analysis Strategies

The diverse array of social network spam and its evasive nature makes it difficult to obtain a comprehensive source of ground truth for measurement. Previous approaches include using blacklists to identify URLs on Facebook directing to spam content [34], deploying passive social networking accounts to act as spam traps [114, 67], and manually identifying spam tweets in trending topics [8]. Each of these approaches introduce a unique bias and error in the type of spam identified. For instance, blacklists preclude URLs that were not reported by users or that failed to appear in email spam traps. False positives and negatives also remain a flaw of social spam traps. Stringhini et al. found that passive accounts acting as spam traps received a surprising volume of legitimate traffic, with only 4.5% of friend requests on Facebook originating from spammers, compared to 90% on Twitter [114]. Equally problematic, samples generated from friend requests will omit spam from compromised accounts in addition to spammers who do not form social connections. While manual analysis by experts reduces the potential for error, it is prohibitively expensive for researchers to acquire a large data sample. Our approach of using Twitter's detection algorithm or blacklists is not without its own bias, which we show in Chapter 3 and Chapter 4. As such, we remain cautious of drawing conclusions for *all* spam on Twitter.

2.5.2 Detecting Scams, Phishing, and Malware Content

Detecting scams, phishing, and malware based on URL and page properties as we show in Chapter 6 has garnered a great deal of research interest. Researchers have paid particular attention to identifying phishing URLs, where a number of solutions rely on HTML forms, input fields, page links, URL features, and hosting properties for detection [142, 71, 35].

Malware, specifically drive-by-downloads, has also been the target of recent study, with most solutions relying on exposing sandboxed browsers to potentially malicious content [99, 129]. An exception is Wepawet, which relies on detecting anomalous arguments passed to plugins to prevent attacks [19]. Our own system generalizes to all forms of scams, phishing, and malware and allows for real-time URL submission by web services.

Of the closest works to our URL classification, Ma et al. show that one can classify spam URLs based on lexical structure and underlying hosting infrastructure including DNS and WHOIS information [72, 73]. We employ these same metrics in our system, but crawl URLs to resolve redirect URLs that would otherwise obscure the final landing page and its hosting infrastructure. A similar approach is taken by Wittaker et al. [132] for specifically classifying phishing pages. We expand upon their research and generalize Monarch to detect all forms of spam, adding features such as JavaScript behavior, redirect chains, and the presence of mashup content, while developing our own classification engine and collection infrastructure to fulfill real-time requirements.

2.5.3 Spam Filtering and Usability Challenges

Spam detection strategies tend to focus on providing accurate decisions, minimizing false positives—legitimate content incorrectly labeled as spam—and false negatives—illegitimate content that is incorrectly classified as benign and thus goes unfiltered. However, a second challenge remains for web services, as they must decide how to appropriately action spam content. Currently, Twitter and Facebook prevent messages containing known spam content from being posted [33, 28], while bit.ly implements a warning page that users must click past to access potentially harmful content [9]. Warnings provide users with an opportunity to bypass false positives, but burden users with making (un)informed security decisions.

The effectiveness of warnings in the context of phishing sites was examined in several studies [135, 25], the results of which showed that unobtrusive warning messages are ineffective compared to modal dialogs and active, full-screen warnings. These works lead to a discussion of the best approach for educating users of security practices and making informed decisions [48]. While advances in usability are orthogonal to Monarch, web services relying on Monarch's decisions can take heed of these studies when determining the best mechanism for conveying the potential harm of a URL to users.

2.5.4 Social Network-specific Spam Detection

The pervasive nuisance of spam in social networks has lead to a multitude of detection strategies specific to social network features. These include analyzing social graph properties of sybil accounts [138, 21, 139], characterizing the arrival rate and distribution of posts [34], analyzing statistical properties of account profiles [114, 8], and identifying common spam redirect paths for URLs posted in tweets [68]. While effective, all of these approaches rely on *at-abuse* time metrics that target strong signals such as sending a spam URL or forming hundreds of relationships in a short period. Consequently, at-abuse time classifiers delay detection until an attack is underway, potentially exposing legitimate users to spam activities before enough evidence of nefarious behavior triggers detection. Furthermore, dormant accounts registered by account merchants will go undetected until miscreants purchase the accounts and subsequently send spam. Overcoming these shortcomings requires *at-registration* abuse detection that flags fraudulent accounts during the registration process before any further interaction with a web service can occur.

Chapter 3

Characterizing Social Network Spam and Abuse

3.1 Introduction

As celebrities such as Oprah, Ashton Kutcher, and Justin Bieber attract throngs of Twitter followers, spammers have been quick to adapt their operations to target Twitter with scams, malware, and phishing attacks. Promising users great diets and more friends, or simply stealing accounts, spam has become a pervasive problem.

Despite an increase in volume of unsolicited messages, Twitter, at the time of our analysis, lacked a filtering mechanism to prevent spam, with the exception of malware, blocked using Google's Safebrowsing API [33]. Instead, Twitter has developed a loose set of heuristics to quantify spamming activity, such as excessive account creation or requests to befriend other users [122]. Using these methods along with user-generated reports of spamming and abusive behavior, the site suspends offending accounts, withdrawing their presence from the Twittersphere along with all of the account's messages.

In this chapter we describe our findings from a large scale effort to characterize spam on Twitter. After collecting a month-long sample of Twitter data, we examine over 400 million public tweets and crawl 25 million unique URLs. Using an assortment of URL blacklists to identify spam, we find over 2 million URLs that direct users to scams, malware, and phishing sites—roughly 8% of all links posted to Twitter. Analyzing the content of spam messages, we provide a breakdown of techniques employed by spammers to exhort Twitter users to click on links. By studying the accounts involved in spamming, we find evidence that spammers primarily abuse *compromised* accounts in their spamming activity, rather than accounts generated solely for the purpose of spamming, which are significantly less prevalent from the perspective of blacklist-identified spam.

Using clickthrough data generated from spam URLs, we examine the success of Twitter spam at enticing over 1.6 million users into visiting spam web pages. We find that the success of spam is directly tied to having a large audience and a variety of accounts to spam from, while use of certain Twitter-specific features also helps increase user traffic. Overall, we find that 0.13% of messages advertised on Twitter will be clicked, *almost two orders of magnitude higher than email spam* [57].

Given the limitations of existing spam filtering on Twitter, we examine whether the use of URL blacklists would help to significantly stem the spread of Twitter spam. By measuring the time period between a blacklist flagging a spam URL and its appearance on Twitter, we find that blacklists in fact lag behind Twitter, with the majority of spam messages appearing 4—20 days before the URLs embedded in the messages become flagged. In contrast, we find over 90% of visits to spam URLs occur within the first two days of posting, indicating that blacklist lag-time is too long to protect a significant number of users against spam. We also examine how spammers can employ URL shortening services to completely evade blacklists, a current problem for Twitter's malware detection.

In summary, the contributions of this chapter are:

- We present the first in-depth look at spam on Twitter, based on a detailed analysis of tweets containing over 2 million distinct URLs pointing to blacklisted scams, phishing and malware.
- We analyze the clickthrough rate for spam on Twitter, finding that 0.13% of users exposed to spam URLs click though to the spam web site.
- We identify a diversity of spam campaigns exploiting a range of Twitter features to attract audiences, including large-scale phishing attacks and targeted scams.
- We measure the performance of blacklists as a filter for URLs posted on Twitter, finding that blacklists are currently too slow to stop harmful links from receiving thousands of clicks.
- We develop techniques to identify and analyze two types of spamming accounts on twitter; fraudulent accounts created primarily for spamming and accounts compromised by spammers.

3.2 Blacklist-based Detection of Spam Content on Twitter

Understanding spam behavior on Twitter requires a large-scale, real-time framework for detecting and tracking spam accounts. In this section, we describe the development of our Twitter monitoring infrastructure and the use of URL blacklists to identify spam. Our infrastructure focuses on analyzing the techniques employed by spammers to generate click traffic and attract an audience, in addition to tracking the use of obfuscation and redirects to mask potentially suspicious web pages.

Within the broad spectrum of spam, we monitor three different categories: malware, phishing, and scams. A spam URL is classified as malware if the page hosts malicious software or attempts to exploit a user's browser. Phishing pages include any website attempting to solicit a user's account credentials, many of which specifically target Twitter credentials. Lastly, we define a scam as any website advertising pharmaceuticals, software, adult content, and a multitude of other solicitations.

3.2.1 Twitter monitoring

To measure the pervasiveness of spam, we develop a Twitter monitoring framework that taps into Twitter's Streaming API[1] and collect roughly seven million tweets/day over the course of one month. We collect data from two separate taps. One targets a random sample of Twitter activity while the second specifically targets any tweets containing URLs. The random sample is used to generate statistics about the fraction of URLs in tweets and general Twitter trends, while the URL stream is used for all other measurements.

Once a tweet appears in the URL stream, we isolate the associated URL and use a custom web crawler to follow the URL through HTTP status codes and META tag redirects until reaching the final *landing page* at a rate of roughly ten landing pages per second; currently, JavaScript and Flash are not handled due to the sheer volume of traffic that must be processed and the complexity required to instrument these redirects. While crawling URLs, each redirect is logged, allowing us to analyze the frequency of cross-domain and local redirects, but more importantly, redirect resolution removes any URL obfuscation that masks the domain of the final landing page. We record the number of redirects and the URLs in each sequence.

3.2.2 Blacklist detection

To automatically identify spam, we use blacklists to flag known spam URLs and domains. We regularly check every landing page's URL in our data set against three blacklists: Google Safebrowsing, URIBL, and Joewein [39, 126, 54]. Each landing page must be rechecked multiple times since blacklists may be slow to update in response to new spam sites. URLs and domains blacklisted by Google indicate the presence of phishing or malware, while URIBL and Joewein specifically target domains present in spam email and are used by anti-spam software to classify email messages. Once a landing page is retroactively marked as spam, we analyze the associated spam tweets and users involved in the spam operation. We have found that URIBL and Joewein include domains that are not exclusively hosting spam; we created a white-list for popular domains that appear on these blacklists and verified that the domains primarily host non-spam content.

[1]`http://dev.twitter.com/docs/streaming-apis`

3.2.3 Data summary

Our data collection spans one month of Twitter activity from January to February, 2010. During this time we gathered over 200 million tweets from the stream and crawled 25 million URLs. Over three million tweets were identified as spam. Of the URLs crawled, two million were identified as spam by blacklists, 8% of all unique links. Of these blacklisted URLs, 5% were malware and phishing, while the remaining 95% directed users towards scams. To understand blacklist performance, we manually inspected a random sample of distinct URLs from tweets, finding that 26% of URLs pointed to spam content, with an error margin of 5% at 95% confidence. To manually classify tweets, one of the authors clicks on the URL in a tweet and decides if the URL is spam based on the content of the web page. Compared to the 8% detected by blacklists, a significant proportion of spam URLs are never seen in blacklists, a challenge discussed in greater detail in Section 3.5. Over 90% of Twitter users have public accounts [80], and we also collect the complete history for over 120,000 users with public accounts, half of which have sent spam identified by our blacklists; the history is an additional 150 million tweets sent by these users.

In the event *bit.ly* or an affiliated service is used to shorten a spam URL, we use the *bit.ly* API[2] to download clickthrough statistics and click stream data which allows us to identify highly successful spam pages and the rate of traffic. Of the spam links recovered, 245,000 had associated clickthrough data, totaling over 1.6 million clicks. Using all of the links recovered during crawling, we present an analysis of the techniques employed by spammers, using clickthrough statistics when available, to measure effectiveness.

3.3 Social Network Spam Content, Distribution, and Clickthrough

With over 3 million tweets posted to Twitter directing users to spam detected by popular blacklists, we present an analysis of the categories of spam appearing on Twitter and what techniques are being employed to reach audiences. To measure the success of Twitter spam, we analyze clickthrough statistics for spam URLs, estimating the likelihood a spam tweet will be clicked by a follower. Finally, as spammers must coerce Twitter members into following spam accounts, we analyze tweeting behavior to differentiate between automated spamming bots and compromised accounts that have been used to send spam, finding the vast majority of spammers appear to be compromised accounts or unwitting participants in spam distribution.

3.3.1 Spam breakdown

Aggregating all of the spam tweets identified by our system, we generate a list of the most frequent terms. We then manually classify each term into a spam category when a clear

[2]`http://dev.bitly.com/api.html`

Category	Fraction of spam
Free music, games, books, downloads	29.82%
Jewelery, electronics, vehicles	22.22%
Contest, gambling, prizes	15.72%
Finance, loans, realty	13.07%
Increase Twitter following	11.18%
Diet	3.10%
Adult	2.83%
Charity, donation scams	1.65%
Pharmacutical	0.27%
Antivirus	0.14%

Table 3.1: Breakdown of spam categories for spam on Twitter, based on tweet text.

distinction is possible, in turn using the terms to classify all of our spam tweets. Roughly 50% of spam was uncategorized due to using random terms; the breakdown of the remaining 50% of tweets is shown in Table 3.1. While the typical assortment of scams present in email carry over to Twitter, we also identify Twitter-specific advertisements that sell Twitter followers or purport to give an account free followers. This unique category makes up over 11% of categorized Twitter spam, while the remainder of spam is dominated by financial scams, games, sale advertisements, and free downloads.

With only 140 characters for spammers to present a message, we analyze what Twitter-specific features appear in tweets with blacklisted URLs compared to those of regular users. To act as a control, we select two samples of 60,000 tweets, one made up of any tweet appearing in our stream, while the second sample is generated from only tweets containing URLs. Each tweet is parsed for mentions, retweets, and hashtags, the results of which can be seen in Table 3.2.

The random sample of tweets is dominated by conversations between users, as indicated by 41% of sample tweets containing mentions. Compared to the sample of tweets containing URLs, spam tweets are only slightly less likely to use Twitter features, with the exception of malware and phishing tweets, where hashtags make up 70% of spam. To understand the motivation for spammers to use these features, we present an analysis of how hashtags, retweets, and mentions are being used by spammers.

Unsolicited Mentions: Mentions are used by spammers to personalize messages in an attempt to increase the likelihood a victim follows a spam link. Mentions can also be used to communicate with users that do not follow a spammer. In our data set, 3.5-10% of spam tweets rely on mentions to personalize messages, the least popular feature compared to hashtags and retweets.

Example: `Win an iTouch AND a $150 Apple gift card @victim! http://spam.com`

Source	#	@	RT	#,@	#,RT
Google	70.1%	3.5%	1.8%	0.1%	0.3%
Joewein	5.5%	3.7%	6.5%	0.2%	0.5%
URIBL	18.2%	10.6%	11.4%	1.5%	1.3%
Tweet	13.3%	41.1%	13.6%	1.8%	2.3%
Tweet, URL	22.4%	14.1%	16.9%	1.6%	2.4%

Table 3.2: Feature frequency by blacklist for mentions (@), retweets (RT), and hashtags (#), compared to a random sample of tweets and a random sample of tweets containing URLs.

Retweets: Of the spam tweets we observe, roughly 1.8-11.4% are retweets of blacklisted URLs. We identify four sources of spam retweets: retweets purchased by spammers from respected Twitter members, spam accounts retweeting other spam, hijacked retweets, and users unwittingly retweeting spam. Of the sources, we are able to differentiate instances of purchased tweets, discussed further in Section 3.4, and hijacked retweets which we discuss next.

Example: `RT @scammer:  check out the Ipads giveaway http://spam.com`

Tweet hijacking: Rather than coercing another account to retweet spam, spammers can hijack tweets posted by other users and retweet them, prepending the tweet with spam URLs. Currently, there are no restrictions on Twitter on who can retweet a message, allowing spammers to take tweets posted by prominent members, modify them, and repost with spam URLs. By hijacking tweets from prominent Twitter users, spammers can exploit user trust in retweets. Analyzing retweets for prepended text, we find hijacking constituted 23% of phishing and malware retweets, compared to 1% of scam retweets.

Example: `http://spam.com RT @barackobama A great battle is ahead of us`

Trend setting: Hashtags are used to simplify searches for content, and if enough users tweet the same hashtag, it becomes a *trending topic*. The anomaly of 70% of phishing and malware spam containing hashtags can be explained by spammers attempting to create a trending topic, generating over 52,000 tweets containing a single tag. Searching for hashtags that exclusively appear in spam tweets, we identify attempts to initiate a trend. Of the total trends we identify, roughly 14% appear to be generated exclusively by spammers.

Example: `Buy more followers!  http://spam.com #fwlr`

Trend hijacking: Rather than generating a unique topic, spammers can append currently trending topics to their own spam. Anyone who searches for the topic will then encounter the spam message, interspersed with other non-spam generated by Twitter users. Using this technique, spammers no longer need to obtain followers and instead ride on the success of other topics. Analyzing the list of trending topics from a set of random tweets, we find that

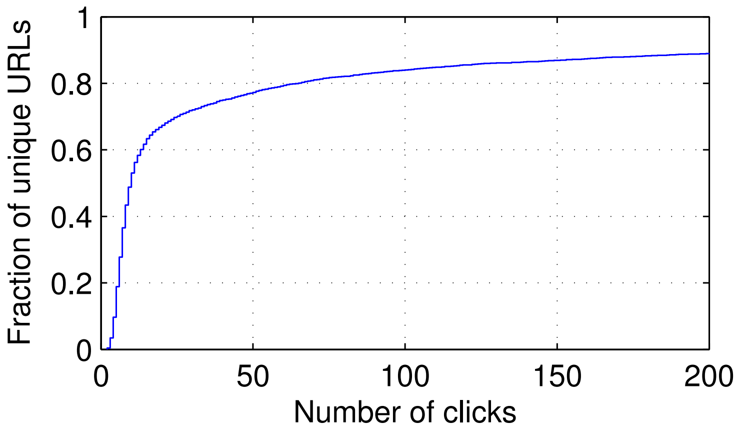


Figure 3.1: Clickthrough for spam URLs posted to Twitter. Only the 2.3% of URLs that generated any traffic are shown.

roughly 86% of trends used by spammers also appear in benign tweets, with popular trends at the time including #haiti, #iranelection, #glee, and the #olympics.

Example: `Help donate to #haiti relief: http://spam.com`

3.3.2 Spam Clickthrough

In the event an account spams URLs shortened with *bit.ly*, we can recover clickthrough statistics for the link and analyze the linear correlation of clickthrough with other features such as followers and tweet behavior. Of the blacklisted domains we identify, we observe the clickthrough data for nearly 245,000 URLs. Roughly 97.7% of URLs receive no clicks, but those that do accumulate *over 1.6 million visitors*, indicating that spam on Twitter is by no means unsuccessful. Of links that generate any traffic, 50% of the URLs receive fewer than 10 clicks, as shown in Figure 3.1, while the upper 10% of URLs account for 85% of the 1.6 million clicks we observe. These highly successful URLs are dominated by phishing scams that have pervaded Twitter in recent months [52], and we discuss this further in Section 3.4.

Using the 2.3% of URLs that receive any traffic, we calculate the linear correlation for clicks and the number of accounts tweeting a link, the aggregate followers that could view the link, and lastly the number of times the link was tweeted, broken down into disjoint combinations of features (*RT, @, #*). Unsurprisingly, the features with the largest coefficient of correlation ($\rho > 0.7$) are the number of accounts involved in spamming and the number of followers that receive a link, both of which directly impact the overall number of potential impressions. In addition to audience volume, we found that the use of hashtags ($\rho = .74$) and retweets with hashtags ($\rho = .55$) is correlated with higher clickthrough rates. In practice, the use of such features is rare, as previously shown in Table 3.2, but their dominance amongst 70% of phishing and malware tweets bolsters their correlation to successful clickthrough.

Surprisingly, the number of times spam is tweeted shows a low coefficient of correlation to clickthrough ($\rho = .28$), indicating that repeatedly posting a link does little to increase traffic.

To understand the effectiveness of tweeting to entice a follower into visiting a spam URL, we measure the ratio of clicks a link receives compared to the number of tweets sent. Given the broadcast nature of tweeting, we measure *reach* as a function of both the total tweets sent t and the followers exposed to each tweet f, where reach equals $t \times f$. In the event multiple accounts with potentially variable number of followers all participate in tweeting a single URL, we measure total reach as the sum of each individual account's reach. Averaging the ratio of clicks to reach for each of the 245,000 URLs in our *bit.ly* data set, we find roughly 0.13% of spam tweets generate a visit, orders of magnitude higher when compared to clickthrough rates of 0.003%—0.006% reported for spam email [57].

There are a number of factors which may degrade the quality of this estimate. First, our data set exclusively targets *bit.ly* URLs which may carry an inherent bias of trust as the most popular URL shortening service [105]. Secondly, click data from *bit.ly* includes the entire history of a link, while our observation of a link's usage only account for one month of Twitter activity. If a link is tweeted prior to our study, or all repeated tweets do not appear in our 10% sample, reach may be underestimated. We attempt to correct for this possibility by measuring the number of times a tweet is repeated using the entire history of 50,000 accounts, finding on average a tweet will appear 1.24 times, with 93% of tweets being unique. This adjustment is factored into the reach of our earlier calculations, but we still caution our estimate of tweet clickthrough as a rough prediction.

Twitter's improved clickthrough rate compared to email has a number of explanations. First, users are faced with only 140 characters in which to base their decision whether a URL is spam. Paired with an implicit trust for accounts users befriend, increased clickthrough potentially results from a mixture of naivety and lack of information. Alternatively, previous estimates of email clickthrough implicitly expect all emails to be viewed. In practice, this may not be the case, resulting in users never being presented the option to click on spam. This same challenge exists in identifying whether a tweet is viewed, but the rates that users view tweets versus emails may differ.

Regardless the underlying cause, Twitter's clickthrough rate makes the social network an attractive target for spammers; with only loose spam filtering in place, spammers are free to solicit throughout the Twittersphere. Furthermore, the computational time of broadcasting tweets is pushed off on Twitter's servers compared to email spam which requires access to large quantities of bots. After a spammer generates a Twitter following, messages can easily be distributed to thousands of followers with a minimal amount of effort.

3.3.3 Spam Accounts

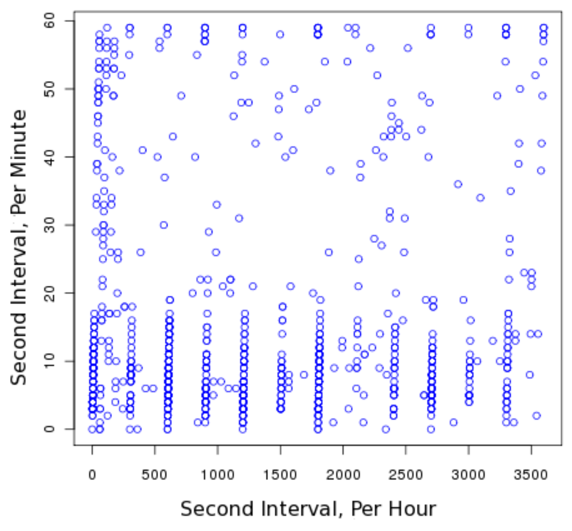
Without Twitter accounts, spammers are incapable of promoting their landing pages. To understand the types of accounts involved in spamming, we define two categories for users flagged as tweeting blacklisted links. The first is the *fraudulent* account created with the express purpose of promoting spam. In contrast, a *compromised* account was created by a legitimate user and at some point in time compromised through the use of phishing attacks, malware, or simple password guessing. To differentiate between the two, we develop an array of tests that analyze an account's entire tweet history, finding that the majority of spam on Twitter originates from compromised accounts, not a fraudulent account. It is important to note these tests are not designed to detect spamming accounts and replace blacklists as they can easily be evaded by an adversary. Instead, we rely on these classification techniques solely to help us understand the ecosystem of spam on Twitter.

Fraudulent Accounts: We develop two tests that indicate if an account is fraudulent and manually verifying the accuracy of each test on a random sample of both spam and likely non-spam accounts. The first test analyzes tweet timing, based on the assumption that legitimate account tweets overall reflect a uniform (Poisson) process. The second test measures the entropy of an account's tweets, identifying users that consistently tweet the same text or link.

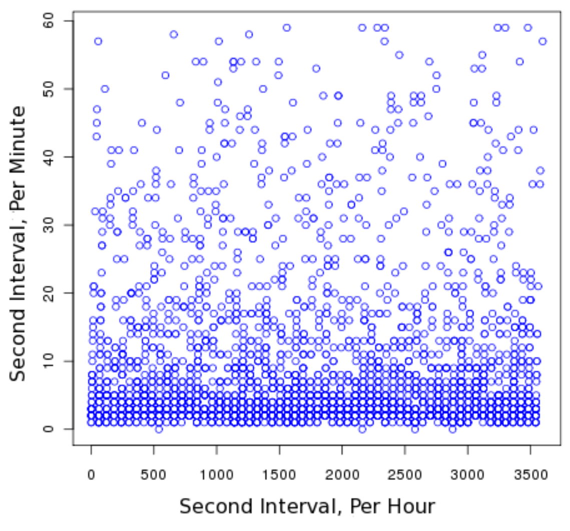
χ^2: Test on Timestamp: Our first test examines tweet timestamps to identify patterns in the minutes and seconds for when a tweet was posted. We represent timestamps for an individual account using vectors corresponding to the seconds value of each hour and seconds value of each minute. We then use a χ^2 test to compute the p-value for these vectors for their consistency with an underlying uniform distribution. For example, a p-value of less than 0.001 indicates less than 0.1% chance that a user posting as a Poisson process generated the sequence. For our evaluation, we treat a p-value of less than 0.001 for either vector as evidence that the user has demonstrably failed the test. Such user tweet patterns very likely reflect automation, leading to postings at regularized times. We deem such accounts as likely fraudulent. Figure 3.2 shows examples of the minutes and seconds for three accounts that fail the test. We manually assessed dozens of accounts that both passed and failed this test, including both inspecting the contents of their tweets and their tweeting patterns over time, finding that it is highly accurate in finding what appear to be fraudulent accounts.

Tweet text and link entropy: For each spam account, we examine the account's tweets history to identify instances where the text and links posted are dominated by repetition, which we measure by calculating entropy. The test begins by binning the text and URLs posted by an account into distinct bins and calculating the entropy of the resulting distribution for the text and URL. If there is no repetition, then the entropy is equivalent to a uniformly random set of the same size. We then calculate relative entropy as the ratio of observed entropy to the entropy of a uniformly random set of the same size, finding that a relative entropy value less than 0.5 indicates strong repetition. For users that do not repeatedly post the same tweet, relative entropy is close to one.

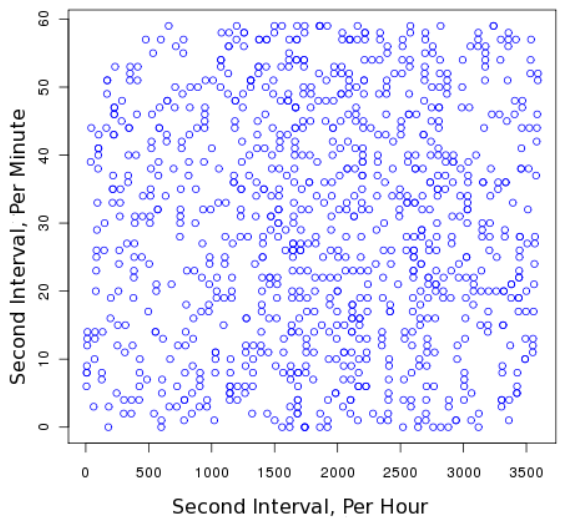
Using the entire tweet history of a sample of 43,000 spam accounts, each with over 100 tweets per user, we find that roughly 16% of accounts tweeting at least one blacklisted link are fraudulent. To gauge the false negative rate of our classification, we manually inspect



(a)



(b)



(c)

Figure 3.2: Scatter plots of times of tweets for three users deemed to not post uniformly. The x-axis gives the minutes value of each hour and y-axis gives seconds. In (a), the user posts at regular intervals—approximately every five minutes. The account in (b) tends to tweet toward the beginning of each minute, indicated by the prevalence of points low on the y-axis. For (c), the pattern is less obvious but still caught by the χ^2 test as indicating regularized tweeting with respect to the hour (x-axis).

99 accounts that passed both the χ^2 and entropy tests to determine a breakdown of the non-fraudulent spamming accounts. Of the 99 samples, 35 are not categorized due to tweets appearing in a foreign language and another 5 had been suspended, prohibiting our access to the account's tweet history and reducing our sample size to 59 accounts. Of these, 5 were clearly fraudulent accounts that had evaded detection, roughly 8.5% of accounts, with an error bound of 7% at 95% confidence.

To understand why the majority of spam accounts passed both tests, we perform a second test to determine how many blacklisted URLs an average account tweets. For each account in our sample of 43,000, we selected 10% of URLs from the account's history and crawled them to determine the final landing page. Using our blacklists, we identified 304,711 spam landing pages, roughly 26% of URLs crawled. The majority of spam accounts tweeted only 2 spam messages, while the remainder of their tweets appeared to be benign URLs and purely text tweets posted at random intervals. Given the low number of spam URLs, we believe the vast majority of accounts tweeting blacklisted URLs are not fraudulent accounts, indicating a potential for compromised accounts.

Compromised spamming accounts: With the majority of spamming accounts passing both the χ^2 and entropy tests used to identify automated behavior, we are left with two possibilities for non-fraudulent accounts. First, an account could have been compromised by means of phishing, malware, or simple password guessing, currently a major trend in Twitter [134]. As most non-fraudulent accounts tweet a limited number of spam URLs, the short lifetime of a compromise can result from Twitter detecting the compromise and notifying the user involved, as occurs with phishing attacks, or the user might identify suspicious activity within their tweet timeline and takes defensive action. Alternatively, given the limited number of spam URLs posted, an account's owner may have tweeted the URLs unintentionally, unaware that they were spam. Given that we expect a non-fraudulent spammer to tweet 20 spam URLs, it is unlikely an account mistakenly posts spam so frequently, leading us to believe accounts are in fact compromised.

Compromised accounts present spammers with an attractive means of exploiting trust, using a victim's credibility to push spam out to followers. Furthermore, by taking control of a victim's account, spammers are relieved of the effort of coercing users into following spam accounts. For non-fraudulent accounts that tweet malware and phishing URLs, we have strong evidence indicating the accounts involved are likely compromised users. In particular, we identify two major schemes to steal accounts, including phishing pages that purport to provide followers and the Koobface botnet which spreads through URLs in tweets [31]. For accounts identified as tweeting spam domains found in the URIBL and Joewein blacklists, we have less direct evidence indicating accounts were compromised, though there have been examples of such behavior reported [134].

Using a fake account to act as a spam trap, we entered our account information into one of the most frequently spammed phishing sites that was blacklisted by Google's Safebrowsing blacklist. Once phished, the account was used to further advertise the same phishing scam in addition to other spam domains. By searching for these spam URLs in our data set, we identified over 20,000 accounts that were affected, 86% of which passed our fraudulent account test.

Further evidence that Twitter accounts are being compromised comes from the spread of Koobface malware which hijacks a victim's Twitter account and tweets on his behalf. During a concerted effort to infiltrate the Koobface botnet, we constructed search queries to find compromised accounts on Twitter and monitored the spread on Twitter during a month of collection. We identified 259 accounts that had tweeted a link leading to a landing page that attempted to install Koobface malware, indicating that these accounts had already been compromised by the botnet and were being used to infect new hosts [117].

These two cases highlight that compromises are occurring on Twitter with the explicit purpose of spreading phishing, malware, and spam. With Twitter credentials being sold in the underground market [78], evidence is mounting that Twitter accounts with large followings are viewed as a commodity, giving access to a trusting audience more likely to click on links, as indicated by our clickthrough results.

Spam Tools: To understand how spammers are communicating with Twitter, we analyze the most popular applications amongst spam accounts used to post tweets. Using information embedded in each tweet, we aggregate statistics on the most popular applications employed by spammers, comparing these results to a random sample. Figure 3.3 shows that fraudulent account application usage is dominated by automation tools such as HootSuite[3] and twitterfeed[4] that allow users to pre-schedule tweets at specific intervals. These tools are not exclusive to spammers, as indicated by the presence in the random sample, though typical users are far more likely to interface with Twitter directly through the web. Interestingly, application usage amongst compromised accounts and a random sample are similar, supporting our claim that the majority of accounts that pass both automation tests are regular Twitter accounts that have been compromised.

Given our belief the majority of accounts are non-fraudulent spammers, we analyze anomalous application usage to identify instances of unauthorized third party access. For typical users, we expect tweets to originate from an array of desktop and phone applications, while spam tweets should appear from an independent application controlled by spammers. To identify this anomaly, we measure the frequency that an application is used to generate spam versus non-spam tweets on a per account basis. On average, 22% of accounts contain spam tweets that originate from applications that are *never* used for non-spam tweets. This pattern of unauthorized third party access further demonstrates that stolen Twitter accounts are being compromised and abused by spammers.

[3]`http://hootsuite.com/`
[4]`http://twitterfeed.com/`

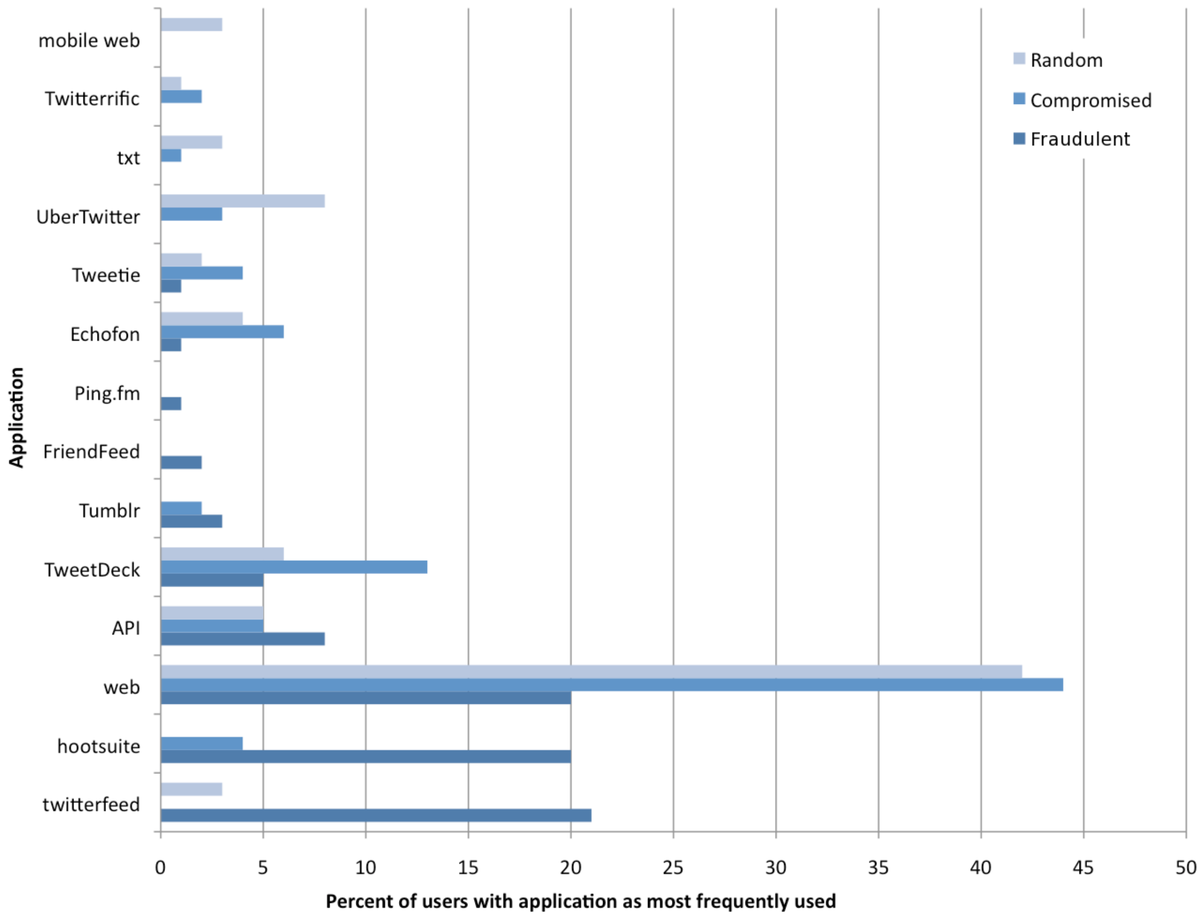


Figure 3.3: Most frequently used applications per-account for compromised, fraudulent, and a random sample of accounts. Fraudulent accounts use different applications than compromised users, which are closer to the random set.

3.4 Detecting, Clustering, and Analyzing Spam Campaigns

To aid in the propagation of products and malware, spammers manage multiple accounts in order to garner a wider audience, withstand account suspension, and in general increase the volume of messages sent. To understand the collusion of accounts towards advertising a single spam website, we develop a technique that clusters accounts into *campaigns* based on blacklisted landing pages advertised by each account. We define a campaign as the set of accounts that spam at least one blacklisted landing page in common. While at least 80% of campaigns we identify consist of a single account and landing page, we present an analysis of the remaining campaigns including the number of websites hosting spam content for the campaign and number of actors involved.

Cluster Statistic	Google	Joewein	URIBL
Maximum possible campaigns	6,210	3,435	383,317
Campaigns identified	2,124	1,204	59,987
Campaigns with more than one account	14.50%	20%	11.46%
Campaigns with more than one page	13.09%	18.36%	27.18%

Table 3.3: Campaign statistics after clustering

3.4.1 Clustering URLs into campaigns

To cluster accounts into campaigns, we first define a campaign as a binary feature vector $\mathbf{c} = \{0, 1\}^n$, where 1 indicates a landing page is present in the campaign and n is the total number of landing pages in our data set. When generating the feature vector for a campaign, we intentionally consider the full URL of a landing page and not its host name to allow for distinct campaigns that operate within the same domain space, such as on free web hosting, to remain separate.

Clustering begins by aggregating all of the blacklisted landing pages posted by an account i and converting them into a campaign $\mathbf{c_i}$, where each account is initially considered part of a unique campaign. Campaigns are clustered if for distinct accounts i, j the intersect $\mathbf{c_i} \cap \mathbf{c_j} \neq \emptyset$, indicating at least one link is shared by both accounts. The resulting clustered campaign $\mathbf{c_{(i,j)}} = \mathbf{c_i} \cup \mathbf{c_j}$. This process repeats until the intersection of all pairs of campaigns $\mathbf{c_i}, \mathbf{c_j}$ is empty. Once complete, clustering returns the set of landing pages for each campaign as well as the accounts participating in each campaign.

Due to our use of Twitter exclusively to identify campaigns, there are a number of limitations worth noting. First, if an account participates in multiple campaigns, the algorithm will automatically group the campaigns into a single superset. This occurs when an account is shared by two spammers, used for multiple campaigns over time by a single spammer, or compromised by different services. Alternatively, if each landing page advertised by a spammer is unique to each account, our algorithm has no means of identifying collusion and results in partitioning the campaign into multiple disjoint subsets.

3.4.2 Clustering results

The results of running our clustering technique on the accounts flagged by each blacklist are shown in Table 3.3. If there were an absence of accounts that tweet multiple scam pages, our clustering technique would return the maximum possible number of campaigns, where each landing page is considered a separate campaign. In practice this is not the case; we are able to identify multiple instances where spam advertised by a group of accounts span a number of distinct landing pages, and even domains.

Analyzing the membership of campaigns, we find that at least 10% of campaigns consist of more than one account. The membership breakdown of these campaigns is shown in

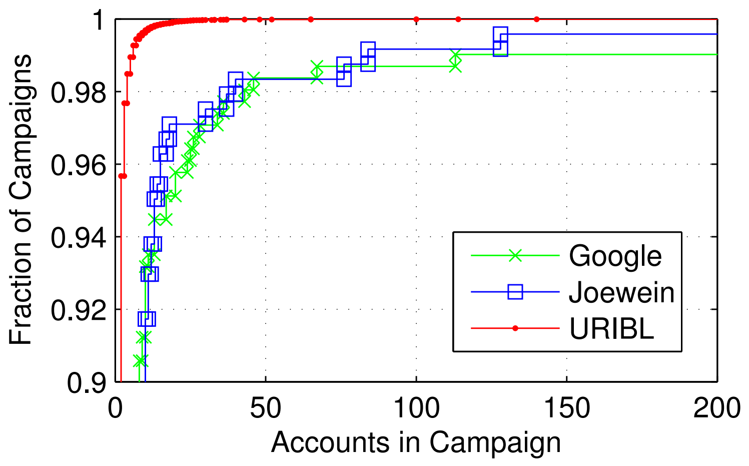


Figure 3.4: Number of accounts colluding in campaigns

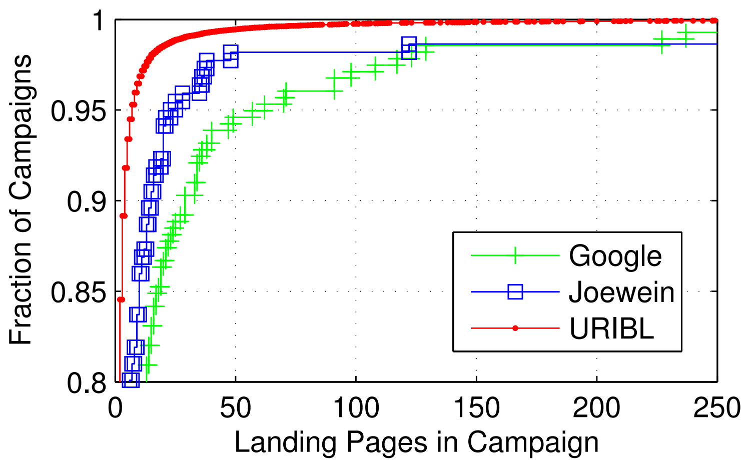


Figure 3.5: Number of landing pages targeted by campaigns

Figure 3.4. Diversity of landing pages within campaigns is slightly more frequent, as shown in Figure 3.5, where the use of affiliate links and multiple domains results in a greater volume of links that comprise a single campaign. While the vast majority of accounts do not collude with other Twitter members, there are a number of interesting campaigns at the tail end of these distributions that clustering helps to identify.

Phishing for followers: A particularly interesting phishing campaign that appeared during our monitoring period is websites purporting to provide victims with followers if they revealed their account credentials. In practice, these accounts are then used in a pyramid scheme to attract new victims and advertise other services.

Clustering returned a set of a 21,284 accounts that tweeted any one of 1,210 URLs associated with the campaign. These URLs directed to 12 different domains, while the full

URL paths contained affiliate information to keep track of active participants. To understand spamming behavior within the campaign, we fractured users into *subcampaigns*, where a subcampaign is a set of users that share identical feature vectors, rather than the original criteria of sharing at least one link in common. From the initial campaign, hundreds of subcampaigns appear. Of the 12 distinct domains, each has a independent subcampaign consisting of on average 1,400 participants, accounting for roughly 80% of the original campaign members. The remaining 20% of participants fall into multiple clusters due to signing up for multiple follower services, accounting for why the independent campaigns were initially merged.

Defining features.: This campaign makes up a significant portion of the tweets flagged by the Google blacklist, and shows surprisingly large user involvement and frequent tweeting. Using the χ^2 and entropy tests, we find that a large fraction of the users, 88% in our set, tweeting for this campaign are compromised users, adding to the evidence that phished accounts are used to further promote the phishing campaign. A defining feature of tweets in this campaign is the extensive use of hashtags, 73% of the tweets sent contained a hashtag. Hash tags are frequently reused and typically denote the subcampaign (such as #maisfollowers). For the URLs being tweeted, most have a redirect chain consisting of a single hop, from a shortened URL to the landing page, though affiliate tracking typically introduces a second hop (shortened URL -¿ affiliate link -¿ landing page). In some cases, the landing page itself appears in tweets. We have also observed that the phishing sites plainly advertise the service to get more followers.

Personalized mentions: Of the campaigns we identify as spam, one particular campaign run by *http://twitprize.com* uses Twitter to draw in traffic using thousands of fraudulent accounts that exclusively generate spam telling users they had won a prize. Clustering returns a set of 1,850 accounts and 2,552 distinct affiliate URLs that were all shortened with tinyurl. Spam within the campaign would target victims by using mentions and crafting URLs to include the victim's Twitter account name to allow for personalized greetings. Promising a prize, the spam page would take a victims address information, require a survey, list multiple mailers to sign up for, and finally request the user either sign up for a credit card or subscribe to a service.

Defining features.: This campaign is dominated by tweet URLs from tinyurl pointing to unique, victim-specific, landing pages at *http://twitprize.com* with no intermediate redirects. Of the tweets containing URLs in this campaign, 99% are a retweet or mention. The heavy use of usernames in tweets is an interesting characteristic, unique to this type of campaign. Unlike the previous phishing campaign, we find infrequent use of hashtags, with only 2% of tweets containing a hashtag. The accounts that tweet URLs in this campaign pass the entropy tests since each tweet contains a different username and the links point to distinct twitprize URLs. Of the accounts participating, 25% have since been suspended by Twitter.

Buying retweets: One of the primary challenges for spammers on Twitter is to gain a massive following in order to increase the volume of users that will see a spam tweet. To circumvent this challenge, a number of services have appeared that sell access to followers. One such service, *retweet.it*, purports to retweet a message 50 times to 2,500 Twitter followers for $5 or 300 times to 15,000 followers for $30. The accounts used to retweet are other Twitter members (or bots) who sign up for the retweet service, allowing their accounts to be used to generate traffic.

Defining features.: While the service itself does not appear to be a scam, it has been employed by spammers. Using a unique feature present in all *retweet.it* posts to generate a cluster, we identify 55 accounts that retweeted a spam post soliciting both malware and scams. The χ^2 test indicate that 84% of the accounts are fraudulent.

Distributing malware: Using clustering, we identified the largest campaign pushing malware in our data set, consisting 113 accounts used to propagate 57 distinct malware URLs. The content of the sites include programs that bring satellite channels to a computer that are "100% adware and spyware free" and an assortment of other scams. In addition to serving malware, some sites advertised by the campaign were reported by Google's blacklist for drive by downloads.

Defining features.: The top malware campaign is significantly different than other campaigns, with a relatively small account base and few tweets. The accounts that tweet links in this cluster tend to be fraudulent, indicating that the malware is not compromising Twitter accounts in order to self propagate, a feature found among Twitter phishing URLs. One difference from other campaigns is this use of redirects to mask the landing page. Since both Twitter and shortening services such as *bit.ly* use the Google Safebrowsing API to filter links, if a *bit.ly* URL is to be placed in tweets, the redirect chain must at least be two hops (*bit.ly* → intermediate → malware landing site). Two hops is not enough though, as the Safebrowsing list contains both sites that serve as well as sites that redirect to malware, requiring at least an additional hop to be used to mask it from initial filtering.

Nested URL shortening: In addition to locating large campaigns, clustering helps to identify instances of URL compression where multiple links posted in tweets all resolve to the same page. One such campaign consisted of 14 accounts soliciting a financial scam. While unremarkable for its size, the campaign stands out for its use of multiple redirector services, totaling 8 distinct shortening domains that appear in tweets. In turn, each initial link triggers a long chain of nested redirects that leads our crawler through *is.gd* → *short.to* → *bit.ly* before finally resolving to the scam page. While the motivation for nested redirects is unclear, it may be a result of spam filtering done on the part of shortening services. By nesting URLs, filtering based on domains or full URLs is rendered obsolete less the final URL is resolved, which we discuss further in Section 3.5

Link Statistics	URIBL	Joewein	Google Malware	Google Phishing
Flagged before posting	27.17%	3.39%	7.56%	1.71%
Flagged after posting	72.83%	96.61%	92.44%	98.29%
Avg. lead period (days)	29.40	13.41	29.58	2.57
Avg. lag period (days)	-21.93	-4.28	-24.90	-9.01
Overall avg. (days)	-12.70	-3.67	-20.77	-8.82

Table 3.4: Blacklist performance, measured by the number of tweets posted that lead or lag detection. Positive numbers indicate lead, negative numbers indicate lag.

3.5 Limitations of Domain Blacklists for Social Networks

Given the prevalence of spam throughout Twitter, we examine the degree to which blacklists could stem the spread of unsolicited messages. Currently, Twitter relies on Google's SafeBrowsing API to block malicious links, but this filtering only suppresses links that are blacklisted at the time of its posting; Twitter does not retroactively blacklist links, allowing previously undetected malicious URLs to persist. To measure how many tweets slip through Twitter's defenses, and whether the same would be true for URIBL and Joewein, we examine a number of blacklist characteristics, including delay, susceptibility to evasion, and limitations that result if we restrict filtering to considering only domains rather than the full paths of spam websites.

3.5.1 Blacklist delay

Using historical data for the URIBL, Joewein, and Google blacklists, we can measure the delay between a tweet's posting and the time of its subsequent blacklisting. For cases where a spam URL embedded in a tweet appeared on a blacklist prior to appearing on Twitter, we say that the blacklist *leads* Twitter. Conversely, a blacklist *lags* Twitter if posted URLs reach the public before becoming blacklisted. Lead and lag times play an important role in determining the efficiency of blacklists. For example, for long lag periods spam filters must maintain a large index of URLs in stale tweets to retroactively locate spam. Furthermore, depending on the rate at which users click on spam links, long lag periods can result in little protection unless spammers reuse links even after they appear on blacklists.

We begin measuring blacklist delay by gathering the timestamps for each tweet of a blacklisted URL. For URLs spammed in multiple tweets, we consider each posting as a unique, independent event. Table 3.4 shows the lead and lag times for tweets, where we see that the majority of spam tweets appear on Twitter multiple days prior to being flagged in blacklists, and in the case of URIBL and Google, multiple weeks. A more extensive presentation of blacklist delay can be seen in Figure 3.6, showing the volume of tweets per

Link Statistics	URIBL	Joewein	Google Malware	Google Phishing
Flagged before posting	50.19%	20.31%	18.89%	15.38%
Flagged after posting	49.81%	79.69%	81.11%	84.62%
Avg. lead period (days)	50.53	15.51	28.85	2.50
Avg. lag period (days)	-32.10	-5.41	-21.63	-10.48
Overall avg. (days)	9.36	-1.16	-12.10	-8.49
Total domains flagged	1620	128	625	13

Table 3.5: Blacklist performance, measured by lead and lag times for unique domains posted.

lead and lag day. It is important to note that Twitter use of Google's Safebrowsing API to filter links prior to their posting biases our analysis towards those links that pass through the filter, effectively masking the lead time apart from URLs that spammers obfuscated with shorteners to avoid blacklisting.

Table 3.5 shows the same lead and lag periods but weighted by unique domains rather than by individual tweets. While blacklisting timeliness improves from this perspective, this also indicates that domains previously identified as spam are less likely to be re-posted, limiting the effectiveness of blacklisting.

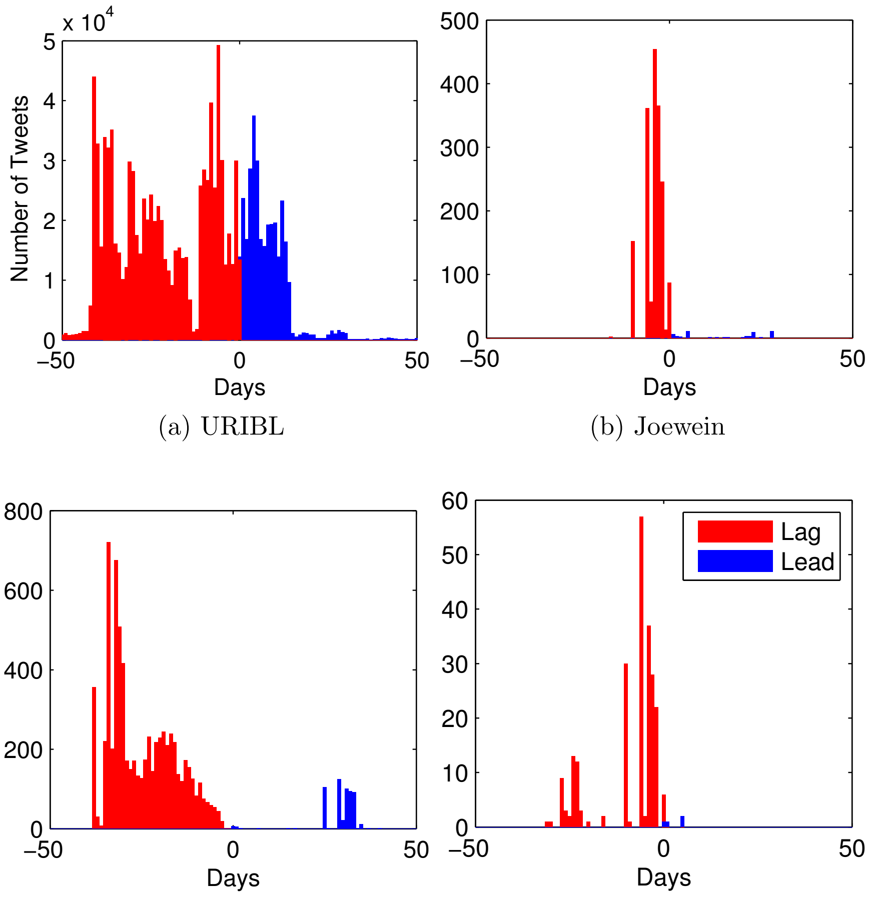
To understand the exposure of users due to blacklist lag, we measured the rate that clicks arrived for spam links. Using daily clickthrough data for a random sample of 20,000 spam links shortened with *bitly*, we found that 80% of clicks occur within the first day of a spam URL appearing on Twitter, and 90% of clicks within the first two days. Thus, for blacklisting to be effective in the context of social networks, lag time must be effectively zero in order to prevent numerous users from clicking on harmful links.

3.5.2 Evading blacklists

The success of blacklists hinges on the reuse of spam domains; if every email or tweet contained a unique domain, blacklists would be completely ineffective. While the registration of new domains carries a potentially prohibitive cost, URL shortening services such as *bitly*, *tinyurl*, *is.gd*, and *ow.ly* provide spam orchestrators with a convenient and free tool to obfuscate their domains.

By following shortened URLs, we found over 80% of distinct links contained at least one redirect, as shown in a breakdown in Figure 3.7. In particular, redirects pose a threat to blacklisting services when they cross a domain boundary, causing a link to appear from a non-blacklisted site as opposed to a blacklisted landing page. Figure 3.8 shows the cross-domain breakdown for distinct URLs seen in links containing at least one redirect. Roughly 55% of blacklisted URLs cross a domain boundary.

The effect of shortening on Twitter's malware defenses (filtering via Google's Safebrowsing API) appears quite clearly in our data set. Disregarding blacklist delay time, 39% of



(a) URIBL (b) Joewein

(c) Google - Malware (d) Google - Phishing

Figure 3.6: Volume of spam tweets encountered, categorized by either lagging or leading blacklist detection

distinct malware and phishing URLs evade detection via use of shorteners. Despite the small fraction, these links make up *over 98% of malicious tweets* identified by our system. Even in the event a shortened URL becomes blacklisted, generating a new URL comes at effectively no cost. Without the use of crawling to resolve shortened URLs, blacklists become much less effective.

3.5.3 Domain blacklist limitations

For blacklists based only on domains rather than full URLs, such as URIBL and Joewein, false positives pose a threat of blacklisting entire sites. Looking through the history of URIBL and Joewein, we identified multiple mainstream domains that were blacklisted prior to our study, including *ow.ly*, *tumblr*, and *friendfeed*. Each of these services allow users to

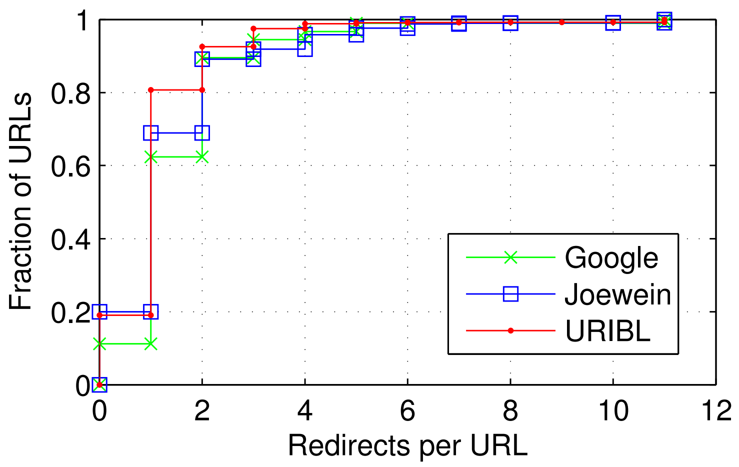


Figure 3.7: Frequency of redirects and nested redirects amongst distinct spam URLs

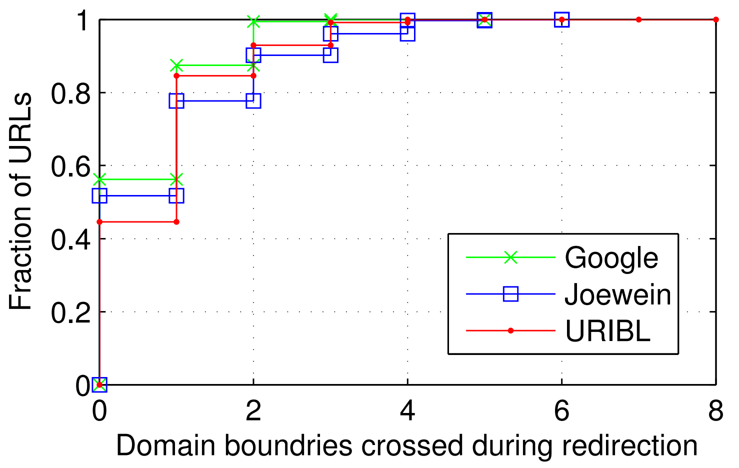


Figure 3.8: Frequency of cross-domain redirects amongst distinct spam URLs containing at least one hop

upload content, giving rise to the potential for abuse by spammers.

The presence of user-generated content and mashup pages presents a unique challenge for domain blacklists. For instance, while *ow.ly* merely acts as a redirector, the site embeds any spam pages to which it redirects in an iFrame, causing a browser's address bar to always display *ow.ly*, not the spam domain. When faced with mashup content, individual cross-domain components that make up a page must be blacklisted rather than the domain hosting the composite mashup. This same challenge exists for Web 2.0 media where content contributed by users can affect whether a domain becomes blacklisted as spam. For *tumblr* and *friendfeed*, we identified multiple cases in our data set where the domains were used by spammers, but the majority of accounts belong to legitimate users. The appearance and subsequent deletion of social media domains within URIBL and Joewein disguises the fact that the domains are being abused by spammers. To address the issue of spam in social media, individual services can either be left to tackle the sources of spam within their own sites, or new blacklists must be developed akin to Google's Safebrowsing API that go beyond domains and allow for fine-grained blacklisting.

3.6 Summary of Results

This chapter presents the first study of spam on Twitter including spam behavior, clickthrough, and the effectiveness of blacklists to prevent spam propagation. Using over 400 million messages and 25 million URLs from public Twitter data, we find that 8% of distinct Twitter links point to spam. Of these links, 5% direct to malware and phishing, while the remaining 95% target scams. Analyzing the account behavior of spammers, we find that only 16% of spam accounts are clearly automated bots, while the remaining 84% appear to be compromised accounts being puppeteered by spammers. Even with a partial view of tweets sent each day, we identify coordination between thousands of accounts posting different obfuscated URLs that all redirect to the same spam landing page. By measuring the clickthrough of these campaigns, we find that Twitter spam is far more successful at coercing users into clicking on spam URLs than email, with an overall clickthrough rate of 0.13%.

Finally, by measuring the delay before blacklists mark Twitter URLs as spam, we have shown that if blacklists were integrated into Twitter, they would protect only a minority of users. Furthermore, the extensive use of URL shortening services masks known-bad URLs, effectively negating any potential benefit of blacklists. We directly witness this effect on Twitter's malware and phishing protection, where even if URLs direct to sites known to be hostile, URL shortening allows the link to evade Twitter's filtering. To improve defenses for Twitter spam, URLs posted to the site must be crawled to unravel potentially long chains of redirects, using the final landing page for blacklisting. While blacklist delay remains an unsolved challenge, retroactive blacklisting would allow Twitter to suspend accounts that are used to spam for long periods, forcing spammers to obtain new accounts and new followers, a potentially prohibitive cost.

Chapter 4

Social Spam: Tools, Techniques, and Monetization

4.1 Introduction

As Twitter continues to grow in popularity, a spam marketplace has emerged that includes services selling fraudulent accounts, affiliate programs that facilitate distributing Twitter spam, as well as a cadre of spammers who execute large-scale spam campaigns despite Twitter's efforts to thwart their operations. While social network spam has garnered a great deal of attention in the past year from researchers, most of the interest has involved developing tools to detect spam. These approaches rely on URL blacklists [34], passive social networking spam traps [114, 67], and even manual classification [8] to generate datasets of Twitter spam for developing a classifier that characterizes abusive behavior. These spam detection approaches however have not yet been used to analyze the tools and techniques of spammers, leaving the underground marketplace that capitalizes on Twitter largely obscure.

In this chapter we characterize the illicit activities of Twitter accounts controlled by spammers and evaluate the tools and techniques that underlie the social network spam distribution chain. This infrastructure includes automatically generated accounts created for the explicit purpose of soliciting spam; the emergence of *spam-as-a-service* programs that connect Twitter account controllers to marketers selling products; and finally the techniques required to maintain large-scale spam campaigns despite Twitter's counter-efforts. To perform the study, we aggregate over 1.8 billion messages on Twitter sent by 32.9 million accounts during a seven month period from August 17, 2010 to March 4, 2011. Within this period, we identify accounts suspended by Twitter for abusive behavior, including spam, aggressive friending, and other non-spam related offenses. Manual analysis indicates that an estimated 93% of suspended accounts were in fact spammers, with the remaining 7% suspended for mimicking news services and aggressive marketing. In total, our dataset consists of over 1.1 million suspended accounts that we show to be spammers, and 80 million spam tweets from these accounts. In contrast to the previous chapter and previous studies [34], only 8% of the URLs we examine were ever caught by blacklists, and the accounts within our dataset are largely fraudulent, as opposed to compromised users. This enables us to provide a unique perspective on a subset of Twitter spammers not previously examined.

At the heart of the of the Twitter spam craft is access to hundreds of accounts capable of reaching a wide audience. We find that 77% of accounts employed by spammers are suspended within a day of their first post, and 92% of accounts within three days. The countermeasures imposed by Twitter's suspension algorithm preclude the possibility of attempting to form meaningful relationships with legitimate users, with 89% of spam accounts having fewer than 10 followers. In place of distributing messages over the social graph, we find that 52% of spam accounts turn to *unsolicited mentions*, whereby a personalized message is sent to another account despite the absence of a social relationship. Another 17% of accounts rely on embedding *hashtags* in their messages, allowing spam to garner an audience from users who view popular Twitter discussions via search and *trending topics*.

Beyond the characteristics of spam accounts, we explore five of the largest Twitter spam campaigns that range from days to months in duration, weaving together fraudulent accounts, diverse spam URLs, distinct distribution techniques, and a multitude of monetization approaches. Together, these campaigns control 145 thousand account that generate 22% of spam on Twitter. Surprisingly, three of the largest campaigns direct users to legitimate products appearing on *amazon.com* via affiliate links that generate income on a purchase, blurring the line regarding what constitutes spam. Indeed, only one of the five campaigns we analyze advertises content generally found in email spam [69], revealing a diverse group of miscreants in the underground space that go beyond email spammers.

Finally, within the amalgam of spam on Twitter, we identify an emerging market of *spam-as-a-service*. This marketplace includes affiliate programs that operate as middlemen between spammers seeking to disseminate URLs and affiliates who control hundreds of Twitter accounts. The most prominent affiliate program, called Clickbank, appeared in over 3.1 million tweets sent from 203 affiliates participating in the program. Other services include ad-based URL shorteners as well as account arbiters who sell the ability to tweet from thousands of accounts under a single service's control. Each of these services enables a diversification in the social network spam marketplace, allowing spammers to specialize exclusively in hosting content or acquiring Twitter accounts.

In summary, we frame our contributions as followers:

- We characterize the spamming tools and techniques of 1.1 million suspended Twitter accounts that sent 80 million tweets.

- We examine a number of properties pertaining to fraudulent accounts, including the formation of social relationships, account duration, and dormancy periods.

- We evaluate the wide-spread abuse of URLs, shortening services, free web hosting, and public Twitter clients by spammers.

- We provide an in-depth analysis of five of the largest spam campaigns targeting Twitter, revealing a diverse set of strategies for reaching audiences and sustaining campaigns in Twitter's hostile environment.
- We identify an emerging marketplace of social network *spam-as-a-service* and analyze its underlying infrastructure.

4.2 Retroactive Detection of Spam Accounts on Twitter

In order to characterize the tools and services that Twitter spammers rely on, we aggregate a dataset of nearly 1.8 billion tweets sent by 32.9 million Twitter accounts over a 7 month period. Of these, we identify 1.1 million accounts suspended by Twitter for abusive behavior. Combined, these accounts sent over 80 million tweets containing 37 million distinct URLs. We manually verify a sample of suspended accounts and find the vast majority were suspended for spamming, providing us with a rich source of ground truth for measuring spam. In addition to our Twitter dataset, we resolve the first redirect of 15 million URLs to deobfuscate a layer of shortening. Finally, for 10 million URLs shortened by *bit.ly*, we download multiple statistics provided by *bit.ly* including clickthrough and, when available, the *bit.ly* account that shortened the URL. A summary of our dataset can be found in Table 4.1.

4.2.1 Twitter Dataset

Our Twitter dataset consists of over 1.8 billion tweets collected from Twitter's streaming API [124] during a seven month period from August 17, 2010 to March 4, 2011. We access Twitter's API through a privileged account, granting both increased API requests per hour and a larger sample than would be conferred to a default account. We rely on the `statuses/filter` method to collect a sample of public tweets conditioned to contain URLs. For each tweet, we have the associated text of the tweet, the API client used to post the tweet (e.g., web, third-party client), as well as statistics tied to the account who posted the tweet including the account's number of friends, followers, and previous posts. On average, we receive 12 million tweets per day, with a ceiling imposed by Twitter capping our collection at 150 tweets per second. We lack data for some days due to network outages, updates to Twitter's API, and instability of our collection infrastructure. A summary of tweets collected each day and outage periods is shown in Figure 4.1.

In order to label spam within our dataset, we first identify accounts suspended by Twitter for abusive behavior. This includes spam, aggressive friending, and other non-spam related offenses. Upon suspension, all of an accounts tweets and profile data become restricted and relationships disappear from the social graph. While this provides a clear signal to identify suspended accounts, it also eliminates any possibility of simply downloading an account's

Data Source	Sample Size
Tweets	1,795,184,477
Accounts	32,852,752
Distinct URLs	1,073,215,755
Tweets from Suspended Accounts	80,054,991
Suspended Accounts	1,111,776
Distinct URLs from Suspended Accounts	37,652,300
Resolved URLs	15,189,365
Bit.ly URLs	10,092,013
Bit.ly Accounts	23,317

Table 4.1: Summary of data collected from Twitter, Bit.ly, and from resolving the first redirect of URLs

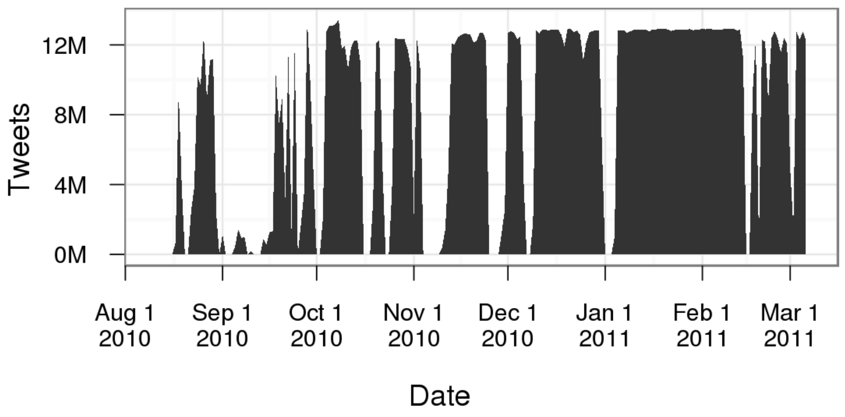


Figure 4.1: Tweets containing URLs received per day. On average, we receive 12 million tweets per day, with a ceiling imposed by Twitter.

history upon suspension. Nevertheless, we are able to reconstruct a composite of a suspended account's activities from all of the tweets in our sample set.

Due to delays in Twitter's account suspension algorithm, we wait two weeks from the last day of data collection before we determine which accounts were suspended by Twitter. This process consists of a bulk query to Twitter's API to identify accounts that no longer have records, either due to deletion or suspension, followed by a request to access each missing account's Twitter profile via the web to identify requests that redirect to *http://twitter.com/suspended*. Of 32.9 million accounts appearing in our sample, 1.1 million were subsequently suspended by Twitter, roughly 3.3% of accounts. These accounts posted over 80 million tweets, with their daily activity shown in Figure 4.2. We sample a portion of these accounts and manually verify that the vast majority are suspended for spam

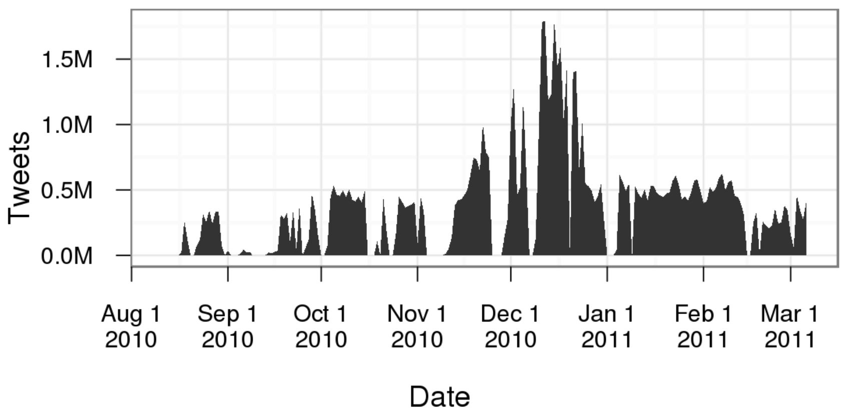


Figure 4.2: Daily tweet activity of suspended users. Peak activity preceded the holiday season in December.

behavior. Furthermore, we provide an estimate for what fraction of each spam account's tweets appear in our sample, as well as provide an estimate for how many spam accounts go uncaught by Twitter and are thus unlabeled in our data set.

Validating Suspended Accounts are Spammers: When we identify a suspended account, we retroactively label all of the account's tweets in our sample as spam. In doing so, we make an assumption that suspended accounts are predominantly controlled by spammers and are not valid accounts performing unrelated abusive behaviors. To validate this assumption, we draw a random sample of 100 suspended accounts and aggregate every tweet posted by the account appearing in our dataset. We then analyze the content of each tweet to identify common spam keywords, frequent duplicate tweets, and tweet content that appears across multiple accounts. Additionally, we examine the landing page of each tweet's URL, if the URL is still accessible, and the overall posting behavior of each account to identify automation.

Of the 100 accounts, 93 were suspended for posting scams and unsolicited product advertisements; 3 accounts were suspended for exclusively retweeting content from major news accounts, and the remaining 4 accounts were suspended for aggressive marketing and duplicate posts. None of the accounts appeared to be legitimate users who were wrongfully suspended. Presumably, any such false positives would later be resolved by the user requesting their account be unsuspended. From these results, we can discern that the majority of accounts we examine are *fraudulent* accounts created by spammers, though the URLs posted by some of these accounts may direct to legitimate content. We provide further evidence that the accounts in our dataset are created explicitly for spamming rather than compromised or legitimate when we examine the relationships and duration of suspended accounts in Section 4.3.1. From here on out, we refer to suspended accounts as spam accounts interchangeably.

Validating Active Accounts are Non-spammers: False negatives from Twitter's detection algorithm result in omitting a portion of spam accounts from our analysis. To measure what fraction of spam accounts are missed by Twitter, we randomly sample 200 active accounts and evaluate each account's tweet history using the same criteria we applied to validate spam accounts. Of the 200 accounts, 12 were clearly spammers, from which we can estimate that 6% of active accounts are in fact spammers, with an error bound of ±3.3% at 95% confidence. Consequently, many of our measurements may underestimate the total volume of spam on Twitter and the number of accounts colluding in spam campaigns. For the accounts we manually identified as overlooked spammers, we found no significant distinction between their behavior and that of suspended accounts, leading us to believe they fall below some classification or heuristic threshold that bounds false positives.

Estimating the Likelihood Spammers are Caught: Using our estimates of the number of false positives and false negatives that result from Twitter's spam detection algorithm, we can approximate the algorithm's sensitivity, or the likelihood that a spam account posting URLs will be caught. Of the 31 million accounts that were not suspended, 6% are false negatives, amounting to roughly 1.9 million spam accounts that are overlooked. Another 1 million spam accounts were correctly identified by Twitter's algorithm. Applying the metric for sensitivity:

$$sensitivity = \frac{true\ positives}{true\ positives + false\ negative}$$

we find that only 37% of spam accounts posting URLs on Twitter are caught by the suspension algorithm during the period of our measurement. We note that this estimate is sensitive to the error bound of the false negative rate. Despite the potential for omitting a class of spam from our analysis, we show in Section 4.2.2 that alternative approaches such as using blacklists to identify spam accounts net an entirely different class of spammers. As such, while our sample may be biased, our analysis provides insights into a large population of spam accounts that have previously been uncharacterized.

Sample Rate: As a final validation step, we measure the fraction of URLs posted to Twitter that we receive in our sample. Our daily sample remains roughly constant at 12 million tweets even though Twitter reports exponential growth [120]. After October 12, 2010, Twitter began to impose a rate limit of 150 tweets per second regardless the actual rate they receive tweets. To measure how this impacts our sample, we take a random sample of 1,600 non-suspended accounts that appear in our dataset and download the entirety of their account history. Of these accounts, 1,245 were still publicly accessible, providing a sample of 798,762 tweets. We then filter out tweets that appear during an outage in our collection or that do not contain URLs, leaving a sample of 32,142 tweets, with roughly 465 samples per day. The daily fraction of these tweets that appear in our sample can be seen in

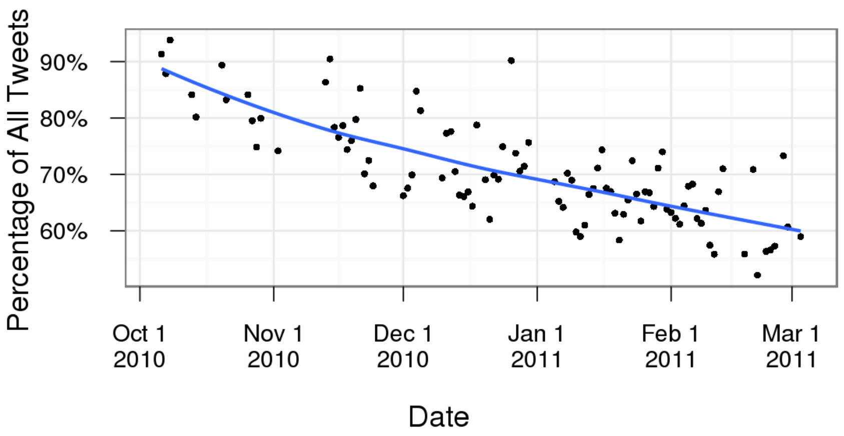


Figure 4.3: Estimated percentage of all tweets containing URLs we receive per day. Due to Twitter's cap of 12 million tweets per day, we receive a smaller sample size as Twitter grows.

Figure 4.3 along with a fit curve. At our peak collection, we received 90% tweets containing URLs posted to Twitter. The sample rate has since decreased to nearly 60%.

4.2.2 Spam URL Dataset

From the 80 million tweets we label as spam, we extract 37.7 million distinct URLs pointing to 155,008 full domains (e.g.,
an.example.com) and 121,171 registered domains (e.g., *example.com*). Given the multitude of shorteners that obfuscate landing pages and no public listing, we attempt to fetch each URL and evaluate whether the HTTP response includes a server-side redirect to a new URL. We only resolve the first such redirect, making no attempt at subsequent requests. In total, we are able to resolve 15.2 million URLs, the remainder of which were shortened by services that have since been deactivated, or were inaccessible due to rate limiting performed by the shortening service.

Blacklist Overlap: In prior work, we examined millions of URLs posted to Twitter that appeared in blacklists [41]. To determine whether the spam we identify from suspended accounts differs significantly from spam detected by blacklists, we examine the overlap of our spam URL dataset with blacklists. We take a sample of 100,000 URLs appearing in tweets regardless of whether they are shortened and a second sample of 100,000 unshortened URLs. We consult three blacklist families: SURBL and all its affiliated lists (e.g., SpamCop, Joewein); Google Safebrowsing, both malware and phishing; and URIBL. If a URL was flagged at any any point in the history of these blacklists from August, 2010 till May, 2011, we consider the URL to be blacklisted. We find only 8% of spam tweet URLs appeared in blacklists and only 5% of unshortened URLs. As such, we believe we present an entirely un-explored subset of spam on social networks from both the perspective of fraudulent accounts as well as non-blacklisted spam.

4.2.3 Bit.ly URL and Account Dataset

Of the URLs associated with spam tweets in our dataset, over 10 million direct to *bit.ly* , a popular shortening service, or one of its multiple affiliated services (e.g., *j.mp*, *amzn.to*). From *bit.ly*'s public API [124] we are able to download clickthrough statistics for a subset of these URLs found in prominent spam campaigns, and when available, the registered *bit.ly* account that shortened the URL. Roughly 47% of *bit.ly* URLs in our dataset had an associated *bit.ly* account, of which 23,317 were unique.

4.3 Tools and Techniques of Spammers

Within the amalgam of spam activities on Twitter, we identify a diverse set of tools and strategies that build upon access to hundreds of fraudulent accounts, an array of spam URLs and domains, and automation tools for interacting with Twitter. We explore each of these areas in depth and present challenges facing both spammers and Twitter in the arms race of social network spam.

4.3.1 Accounts

At the heart of the Twitter spam craft are thousands of fraudulent accounts created for the explicit purposes of soliciting products. 77% of these accounts are banned within a day of their first post, and 89% acquire less than 10 followers at the height of their existence. Yet, within Twitter's hostile suspension environment, spammers are still capable of reaching millions of users through the use of unsolicited mentions and trending topics. We examine a range of properties surrounding spam accounts, including the length of their activity, the rate they send tweets, the social relationships they form, and the stockpiling of accounts.

Active Duration: Spam accounts are regularly suspended by Twitter, but it takes time for Twitter to build up a history of mis-activity before taking action. We measure this window of activity for a sample of 100,000 spam accounts created after our measurement began. We omit accounts that were created during one of our outage periods to reduce bias, though an account appearing at the cusp of an outage period will have its activity window underestimated. For each of these accounts, we calculate the difference between the timestamp of an account's first tweet and last tweet within our dataset, after which we assume the account was immediately suspended. Figure 4.4 shows a CDF of account activity. 77% of accounts were suspended within a day of their first tweet, with 92% of accounts surviving only three days. The longest lasting account was active for 178 days before finally being suspended. While a minority of accounts are able to persist, we show

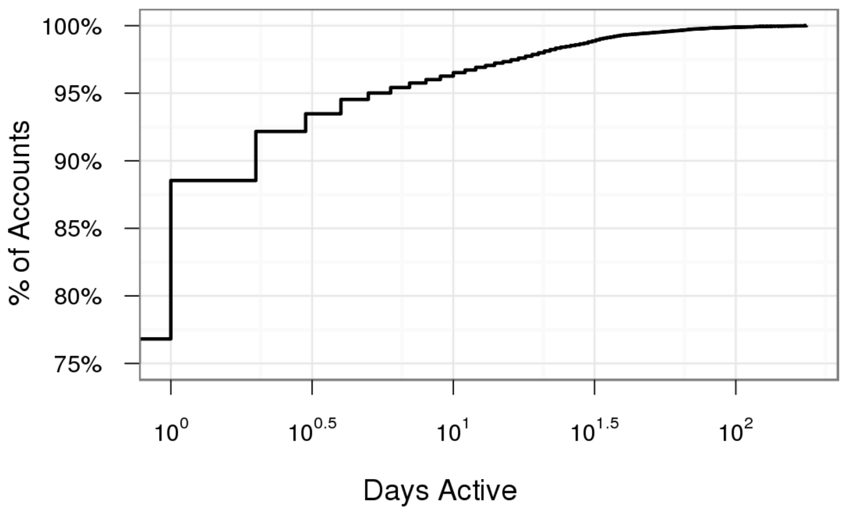


Figure 4.4: Duration of account activity. 77% of accounts are suspended within a day of their first tweet and 92% within three days.

that rapid suspension impacts both the volume of tweets spammers can disseminate and the relationships they can form.

Tweet Rates: Given the threat of account suspension, we examine whether the rate that spammers send tweets impacts when they are suspended. In order to calculate the total number of tweets sent by an account, we rely on a statistical summary embedded by Twitter in each tweet that includes the *total* number posts made by an account (independent of our sampling). Using a sample of 100,000 accounts, we calculate the maximum tweet count embedded for each account by Twitter and compare it against the account's active duration.

The results of our calculation are shown in Figure 4.5 along with a fit curve. We identify three clusters in the figure, outlined in ovals, that represent two distinct spamming strategies. The first strategy **(I)** relies on short-lived accounts that flood as many tweets as possible prior to being suspended, representing 34% of our sample. These accounts last a median of 3 days and send 98 tweets. In contrast, a second strategy **(II)** relies on longer lasting accounts that tweet at a modest rate, representing 10% of our sample. While these accounts last a median of 7 days, in the end, they send a nearly equal volume of tweets; a median of 97 tweets per account. The final cluster **(III)** consists of 56% of accounts that are suspended within a median of 1 day and send 5 tweets on average. The reason behind these accounts' suspension is unclear, but it is likely tied to rules beyond tweet count, such as sending URLs duplicated from previously suspended accounts or sharing an email address or IP address with other suspended accounts. While an individual account sending 100 tweets will not reach a large audience, we show in Section 4.5 that actual spam campaigns coordinate thousands of accounts yielding hundreds of thousands of tweets.

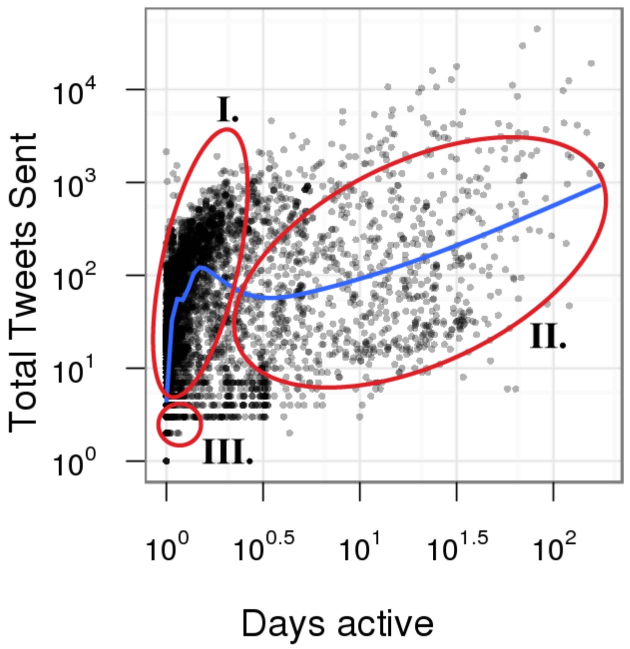


Figure 4.5: Active duration vs. tweets sent for spam accounts. Two strategies appear: **(I)** burst accounts and **(II)** long-lived, low-daily volume accounts

Relationships: The social graph is the focus of regular user interaction on Twitter, yet we find that most spammers fail to form social connections and instead leverage other social networking features to reach an audience. Due to the disappearance of spam accounts from the social graph upon suspension, we are unable to retroactively perform a detailed analysis of the accounts spammers befriended (or whether spammers befriend one another). Nevertheless, each tweet in our dataset includes a snapshot of the number of friends and followers an account held at the time of the tweet. For clarity, we define a *friend* as a second user that an account receives content from, while a *follower* is a second user that receives an account's content. With respect to distributing spam URLs, only followers are important, though spammers will acquire friends in the hope that the relationship will be reciprocated.

To compare relationships formed by both spam and non-spam accounts, we aggregate friend and follower data points for a sample of 100,000 active and suspended users. Figure 4.6 shows a CDF of the maximum number of followers a spam account acquires prior to suspension. Surprisingly, 40% of spam accounts acquire no followers, while 89% of accounts have fewer than 10 followers. We believe this is due both to the difficulty of forming relationships with legitimate users, as well as a result of the hostile environment imposed by Twitter, where the effort and time required to acquire followers is outpaced by the rate of suspension.

With no followers, spam accounts are unable to distribute their content along social

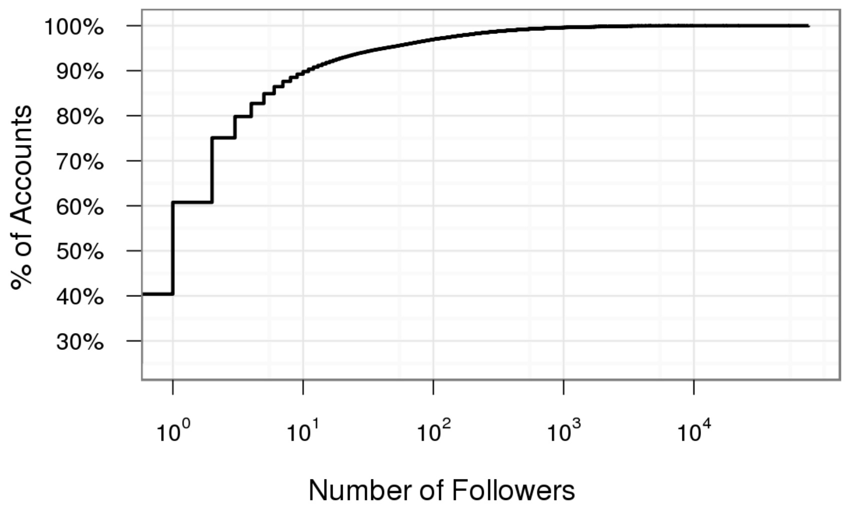
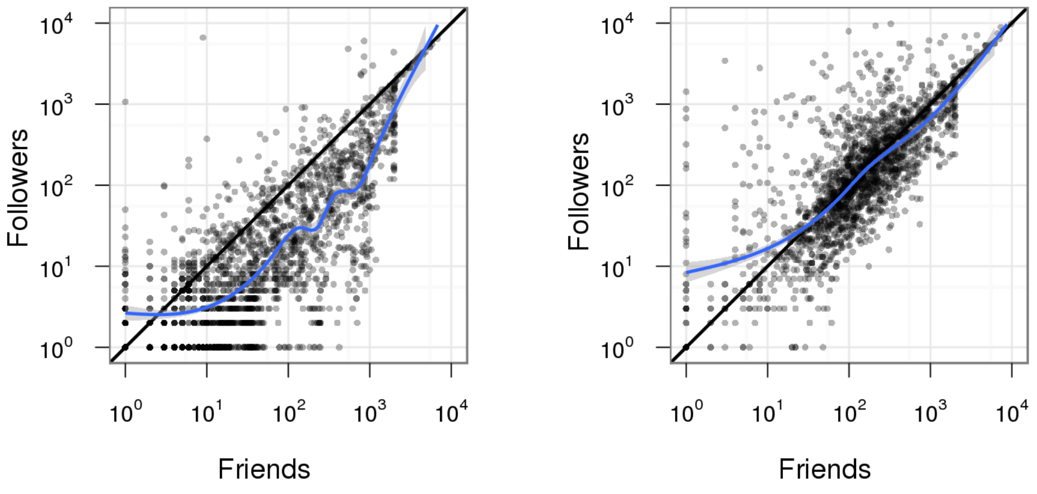


Figure 4.6: Users following spam accounts. 89% of accounts have fewer than 10 followers; 40% have no followers.

connections. Instead, we find that 52% of accounts with fewer than 10 followers send unsolicited mentions, whereby a personally tailored message is sent to an unsuspecting account that shares no relation with the spammer. Another 17% of accounts rely on embedding hashtags in their spam tweets, allowing spam content to appear in the stream of popular Twitter discussions and through search queries. We examine the success of each of these approaches in Section 4.5 for a subset of spam campaigns.

For those spam accounts that do form social relationships, their relationships are heavily skewed towards friends rather than followers, indicating a lack of reciprocated relationships. Figure 4.7 shows the number of friends and followers for spam accounts as well as active accounts presumed to be non-spammers. An identity line in both plots marks equal friends and followers, while a trend line marks the areas of highest density in the scatter plot. Relationships of non-spam accounts center around the identity, while spam accounts are shifted right of the identity due to the lack of reciprocated relationships. The modality in both graphs at 2,000 friends results from Twitter imposing a limit on the number of friends possible, after which an account must have more followers than friends to grow their social graph. While 11% of spam accounts attempt to befriend users, either for the purpose of acquiring followers or for obtaining the privilege to direct messages, it is clear that legitimate Twitter users rarely respond in kind.

Dormancy: Long dormancy periods where a spam account is registered but never used until a later date hint at the possibility of stockpiling accounts. To measure account dormancy, we select a sample of 100,000 accounts created during one our active collection periods and measure the difference between the account's creation date (reported in each tweet) versus



(a) Spam Accounts (b) Non-spam Accounts

Figure 4.7: Friends vs. followers for spam and non-spam accounts. Spammers are skewed towards forming relationships that are never reciprocated.

the account's first post in our sample. The results in Figure 4.8 show that, unsurprisingly, 56% of accounts are activated within a day of their registration. This indicates most spammers create accounts and immediately add them to the pool under their control. However, 12% of spam accounts remain inactive for over a week and 5% for over one month. We highlight this phenomenon further in Section 4.5 when we present a number of campaigns that stockpile accounts and activate them simultaneously to generate hundreds of thousands of tweets.

4.3.2 URLs and Domains

Beyond the necessity of multiple accounts to interact with social networks, spammers also require a diverse set of URLs to advertise. We find that individual spam accounts readily post thousands of unique URLs and domains, simultaneously abusing URL shorteners, free domain registrars, and free web hosting to support their endeavors. Of the 37.7 million spam URLs in our dataset, 89.4% were tweeted *once*. These unique URLs account for 40.5% of spam tweets, while the remaining 10.6% of URLs are massively popular and account for the 59.5% of spam tweets. To understand how spammers are generating URLs, we examine a breadth of properties from the abuse of free services, the diversity of domains, and the overlap of spam URLs with those posted by non-suspended Twitter accounts.

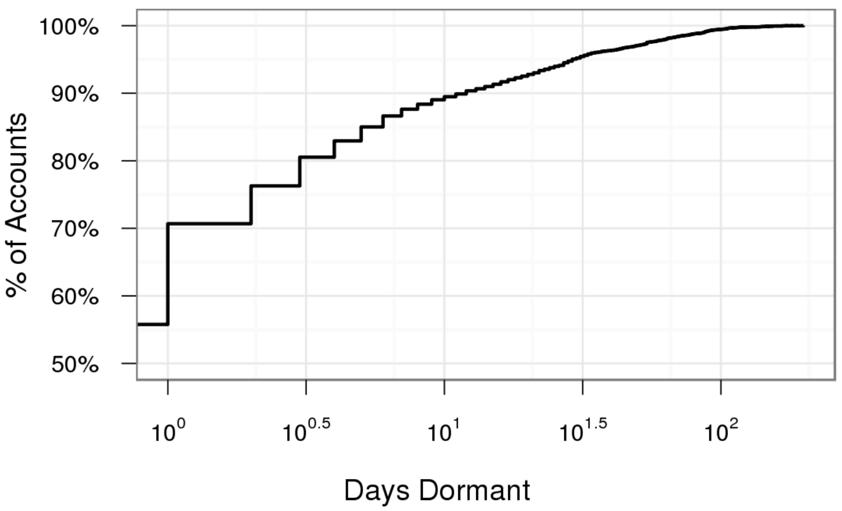


Figure 4.8: Dormancy duration of accounts. 56% of accounts begin tweeting within the same day the account is created, while 12% lay dormant for over one week, allowing for account stockpiling.

Abusing Shorteners: We find that URL shortening services, such as *bit.ly*, are frequently abused by spammers despite their use of blacklists and spam detection algorithms [9]. In general, URL shorteners simplify the process of generating a variety of unique URLs without incurring a cost to the spammer. URL shorteners also obfuscate the destination of a URL that might otherwise look suspicious to visitors and decrease clickthrough.

Given that any domain can operate a shortening service, we develop a heuristic to identify shorteners used by spammers. Using the first-hop resolution data for 15 million URLs, we identify domains that respond with a server-side redirect (HTTP status code 30x). If a single domain redirects to at least five distinct registered domains, that domain is considered to be a shortening service. Using this criteria, we identify *317 services* that are used in *60% of spam tweets*.

The most popular shorteners abused by spammers are shown in Table 4.2; 35% of spam tweets are shortened by *bit.ly*, followed in popularity by *tinyurl.com* and a variety of other shorteners with low spam volumes that make up a long tail. For each shortener we compute the bias spammers have towards using the service compared to regular users. First, we calculate $p_1 = p(shortener|spam)$, the probability a spam tweet uses the shortener, and $p_2 = p(shortener \mid nonspam)$, the probability a non-spam tweet uses the shortener. We then compute the likelihood ratio p_1/p_2. This result is strictly a lower bound as our non-spam dataset contains uncaught spam.

As Table 4.2 shows, all of the top ten shortening services are preferred by spammers, with *3.ly* over 65 times more likely to be used by spammers. The likelihood ratio of a shortener does *n*ot indicate that more spam URLs are shortened than non-spam URLs. Instead, a

Service Name	% of Tweets	Likelihood Ratio
bit.ly	34.86%	1.41
tinyurl.com	6.88%	2.61
is.gd	2.45%	3.01
goo.gl	2.45%	1.14
ow.ly	2.32%	1.40
dlvr.it	1.99%	1.66
tiny.cc	1.38%	12.36
tiny.ly	1.34%	5.23
3.ly	1.14%	65.55
dld.bz	1.10%	3.71

Table 4.2: Top 10 public shortening services abused by spammers. Likelihood ratio indicates the likelihood a spammer will use the service over a regular user.

likelihood ratio greater than one simply indicates that given the choice of domains available to both spammers and regular Twitter users, spammers are more likely to choose shorteners. Even if popular URL shortening services deployed stronger spam filtering, the presence of hundreds of alternative shorteners and the ease with which they are created makes it simple for spammers to obfuscate their URLs.

Domain Names: Spammers who host their own content require access to hundreds of domains in order to counteract attrition resulting from takedown and blacklisting. Where traditional domain registration carries a cost, we find that Twitter spammers cleverly obtain free hosting and subdomains through the abuse of public services.

Figure 4.9 visualizes the number of subdomains and the number of registered domains tweeted by each of the 1.1 million spam accounts in our dataset, along with a fit curve showing the densest regions. If an account exclusively posts unique registered domains (e.g., *greatpills.com*), it will appear along the identity line, while accounts that rely on multiple subdomains (e.g., *greatpills.com, my.greatpills.com*) will appear below the identity line. Three distinct approaches to spamming are apparent, outlined in ovals.

The first approach (I) consists of 0.13% of spam accounts that abuse free subdomain registration services and blog hosting. These accounts post over 10 subdomains tied to fewer than 10 registered domains, with 0.04% of spam accounts tweeting over 250 subdomains. A second spamming strategy (II) consists of using multiple unique domains; we find 1.4% of users tweet over 10 unique registered domains with no additional subdomains, represented by the points scattered along the identify line. The remaining 98.56% of accounts, labeled as (III), tweet fewer than 10 domains in their entire lifetime, appearing as a dense cluster near the origin. Of these clusters, we explore the abuse of free domain registrars and hosting services.

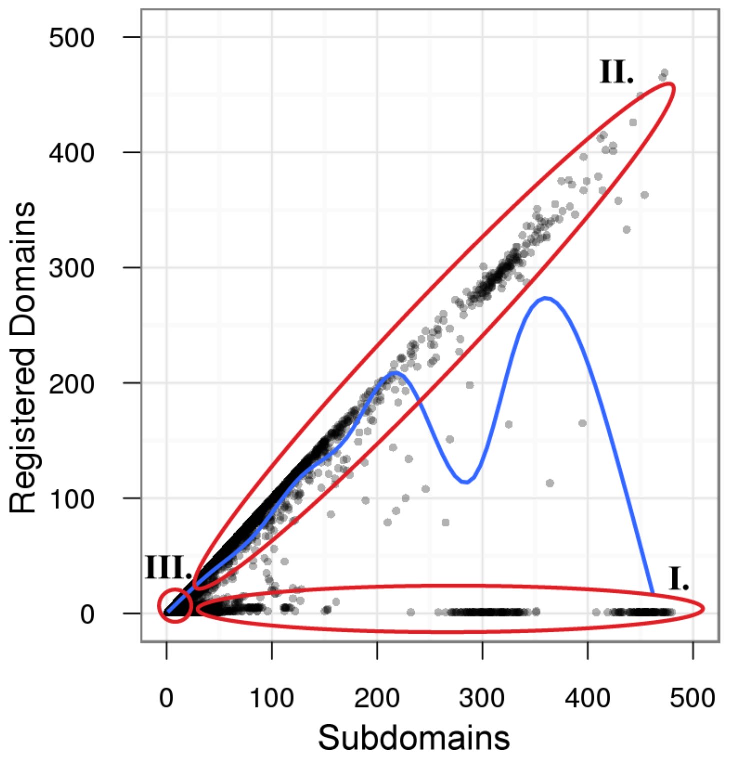


Figure 4.9: The number of subdomains versus the number of registered domains that URLs posted by a spam account resolve to. Each point corresponds to a single account.

Blog Program	Subdomains	Unique URLs
Blogspot	18,364	249,589
LiveJournal	15,327	54,375
Wordpress	4,290	58,727

Table 4.3: Top three free blog hosting sites, the number of blogs registered, and the number of unique URLs pointing to the blogs.

Subdomains: We identify a number of spammers that rely on free subdomains to avoid registration costs. Services including *co.cc*, *co.tv*, *uni.cc*, and *dot.tk* all allow anyone to register a subdomain that directs to an arbitrary IP address. In total, we find over 350,000 spam URLs directing to these services. The majority of these URLs belong to accounts shown in Figure 4.9 that post over 250 subdomains.

One particular group of 376 spam accounts advertised over 1,087 subdomains located at *co.cc* with another 1,409 accounts advertising a smaller subset of the same domains. With no limits on subdomain registration services, this *co.cc* campaign displays how spammers can easily circumvent the requirement of domain registration.

Blog Hosting: Free blog pages from Blogspot, LiveJournal, and Wordpress account for nearly 363,000 URLs; roughly 0.1% of the URLs that appear in our dataset after shortening is resolved. While this may seem minute, Blogspot is the third most popular domain for shortened URLs, highlighting the huge variety of domains used by spammers.

To understand how many blog accounts spammers register, we extract the blog subdomain from each URL and calculate the total number of unique account names that appear, shown in Table 4.3. Over 18,000 accounts were registered on Blogspot and another 15,000 on LiveJournal. As a whole, we identified 7,500 Twitter spam accounts that advertised one of the three blog platforms, indicating a small collection of spammers who abuse both Twitter and blogging services.

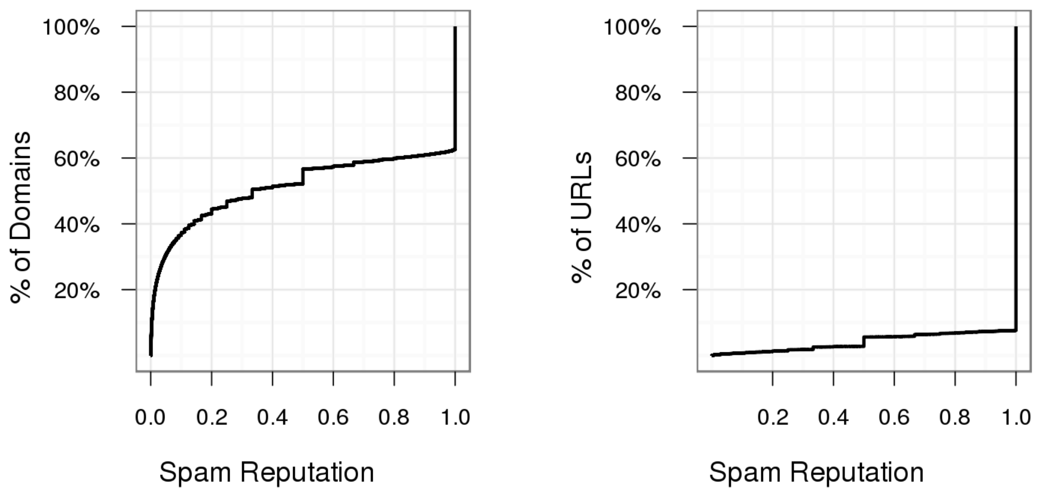
URL and Domain Reputation: Many of the URLs and domains used by spammers also appear in tweets published by non-suspended users. For instance, of the 121,171 registered domains that appear in spam tweets, 63% also appear in tweets posted by active accounts. To understand whether this overlap is the result of a single retweet of a popular URL or a regular occurrence, we calculate a reputation score for each domain and URL in our dataset. Using all 1.8 billion tweets, we calculate the frequency that a domain or URL appears in spam tweets, and repeat this process for all tweets. The fraction of these two values offers a reputation score in the range $(0, 1)$. A reputation of one indicates a domain or URL appears exclusively in spam, while a reputation near zero indicates the domain or URL was rarely found in spam tweets. We note that due to some spam accounts going unsuspended, our reputation scores underestimate how frequently some domains are used by spammers.

Figure 4.10 shows the reputation scores for both domains and URLs. We find 53% of domains appear more frequently in non-spam tweets than spam, compared to 2.8% of URLs. This indicates that attempting to build a domain blacklist from URLs appearing in spam tweets would be highly ineffective, and more so, attempting to detect unsuspended accounts based on their posting a duplicate spam URLs requires explicit knowledge the URL, not the account posting it, was spam. In fact, 11,573,273 active accounts posted at least one URL also posted by spammers.

To break down this phenomenon further, we examine both the reputation of URLs as well as the frequency that both spam accounts and non-spam accounts post them. Figure 4.11 shows the median spam reputation of all the URLs posted by an account as well as the number URLs that an account shares in common with spammers. A trend line marks the clusters with the highest density. Spammers are clearly identified by their rarely duplicated, high-spam reputation scores compared to non-spam accounts that post low-spam reputation URLs, though in low frequency. As such, both account behavior as well as the URLs posted would be required to deploy blacklists.

4.3.3 API Clients

Along with accounts and URLs, spammers require a client to interact with Twitter's API or their web portal. The overall application usage of spam is shown in Table 4.4. We find 64%



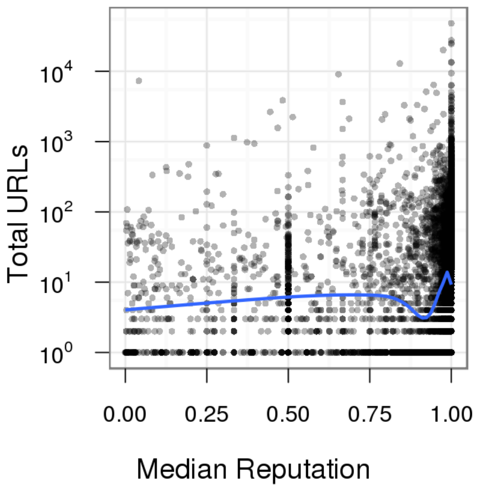
(a) Domain Reputation (b) URL Reputation

Figure 4.10: Reputation of spam URLs and domains. 53% of domains appear more frequently in non-spam tweets than spam tweets, though only 2.8% of URLs.

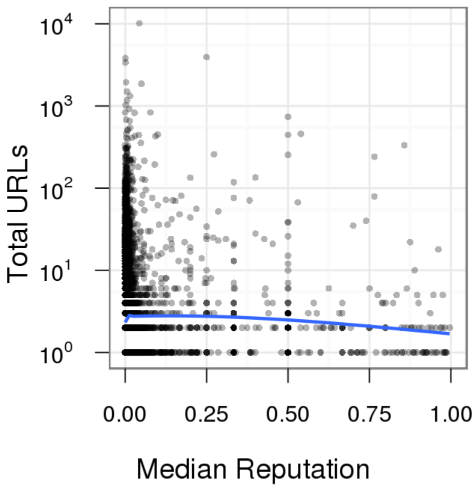
of spam originates from the web and mobile web interface, while the remainder of spam is sent from over 10,544 public and custom API clients. We find spammers are nearly three times more likely to use the web interface compared to regular users, and six times more likely to use the mobile web interface. The remaining top 10 applications are automation frameworks that allow scheduled tweets and connecting blog posts to Twitter. Of the API clients we identify, we find over 6,200 are used *exclusively* to send spam, indicating a number of spammers are readily developing custom clients to access their accounts.

4.4 The Evolution of Monetization Towards Spam-as-a-Service

We find evidence of an emerging spam-as-a-service market that capitalizes on Twitter, including affiliate programs, ad-based shortening services, and account sellers. Each of these services allow spammers to specialize their efforts, decoupling the process of distributing spam, registering domains and hosting content, and if necessary, product fulfillment. Each of the services we identify reveals a targeted approach to monetizing social networks.



(a) Spam Accounts



(b) Non-spam Accounts

Figure 4.11: Comparison of URL reputation and the total spam URLs posted by spam and non-spam accounts, where each point represents a distinct account.

API Name	% of Tweets	Likelihood Ratio
web	58.30%	2.98
twitterfeed	12.39%	1.06
Mobile Web	5.40%	6.07
dlvr.it	2.95%	2.01
hellotxt.com	1.14%	7.89
twittbot.net	1.05%	1.50
EasyBotter	0.98%	4.86
Google	0.83%	0.30
API	0.73%	1.32
www.movatwi.jp	0.72%	1.29
HootSuite	0.71%	0.41

Table 4.4: Top 10 Twitter clients used by spammers.

4.4.1 Affiliate Programs

One aspect of diversification we identify within the underground marketplace is the adoption of affiliate programs, both legitimate and otherwise. From the 15 million URLs we unshortened, we identify two prominent affiliate programs used by spammers: *clickbank.com* and *amazon.com*. Clickbank acts as a middleman, connecting vendors seeking to distribute

Service	Twitter Accounts	Tweets	Type
Clickbank	16,309	3,128,167	Affiliate
Amazon	8,129	1,173,446	Affiliate
Eca.sh	343	352,882	Shortener
Vur.me	72	9,339	Shortener
Spn.tw	905	87,757	Account
Assetize	815	120,421	Account

Table 4.5: Programs enabling spam-as-a-service. These include affiliate programs that connect vendors to affiliate advertisers, shorteners that embed ads, as well as account arbitration services that sell access to accounts.

URLs with affiliates willing to advertise the URLs. Clickbank affiliates are paid based on clickthrough, while vendors are charged a fee. In contrast, Amazon's affiliate program offers up to a 15% commission on purchases made after visitors click on an affiliate's URL. The use of Amazon's affiliate program by spammers blurs the line between what constitutes legitimate advertisement and abuse.

Table 4.5 shows that over 3.1 million spam tweets directed to Clickbank and nearly 1.2 million to Amazon. While the total number of accounts involved in both affiliate programs is a small fraction of the spam accounts we identify, the abuse of these services hint at an emerging spam-as-a-service market that allows Twitter spammers to exclusively spend their effort on generating accounts and tweets, leaving the task of domain registration and content generation to other parties.

Affiliate programs provide a unique opportunity to explore how individuals are earning a profit by sending spam on Twitter. Assuming each affiliate ID uniquely maps to one spammer, we can group an affiliate's Twitter accounts and the tweets the account's send based on the affiliate ID embedded in each tweet's URL. The results, shown in Table 4.6, offer a glimpse at the spam infrastructure each affiliate controls. Our analysis reveals a heavily biased environment where a small number of affiliates account for the vast majority of spam. We repeat this same experiment using the 47% of *bit.ly* URLs that contain an associated *bit.ly* account ID. We find that 50% of the 23,317 *bit.ly* accounts control only two or fewer Twitter accounts. Yet, one *bit.ly* account acquired over 5,000 Twitter accounts and sent over 550,000 tweets, revealing again the same biased marketplace that contains thousands of small actors alongside a few major players.

4.4.2 Ad-Based Shorteners

A second form of monetization we identify is the use of syndicated ads from existing ad networks. Ad-based shortening services such as *eca.sh* and *vur.me* provide public URL shortening services, but in return, embed the destination page for shortened URLs in an `IFrame` and display advertisements alongside the original content. Anyone can register an

		Tweets		Tw. Accts	
Service	Affiliates	Med	Max	Med	Max
Clickbank	203	565	217,686	2	151
Amazon	919	2	324,613	1	848
Bit.ly	23,317	2	551,200	1	5318

Table 4.6: Affiliates identified for Clickbank and Amazon along with Twitter accounts they control and the volume of spam they send. Bit.ly accounts reveal a similar result. Both show a biased environment where a small number of spammers account for the vast majority of spam.

Campaign	Tweets	Accounts	URLs	Hashtags	Mentions	Med. Followers	Med. Tweets
Afraid	14,524,958	124,244	14,528,613	-	11,658,859	2	130
Clickbank	3,128,167	16,309	1,432,680	3,585	542,923	9	108
Yuklumdegga	130,652	2,242	24	11	-	3	83
Amazon	129,602	848	1	-	118,157	22	123
Speedling	118,349	1,947	89,526	4674	870	95	190

Table 4.7: Summary of major spam campaigns on Twitter. This includes the number of tweets, accounts, unique URLs, unique hashtags, and unique mentions. In addition, we include the median number of followers and tweets for accounts in the campaign.

account with the service and will receive a portion of the revenue generated by the additional advertisements. For ad-based URL shorteners, spammers need not control the content they shorten; any major news outlet or popular URL can be shortened, requiring the spammer only handle distribution on Twitter. Within our set of spam, there are over 362,000 tweets sent by 415 accounts using ad-based shorteners, a breakdown of which is provided in Table 4.5.

4.4.3 Account Sellers and Arbiters

The final monetization technique we find in our spam dataset are services that sell *control* of accounts as well as sell *access* to accounts. One particular service, called Assetize (since disabled), allowed Twitter users to sell access to their accounts. Assetize drew in over 815 accounts, in turn composing tweets and sending them on each account's behalf. In return, the account's owner would be paid. A similar service called Sponsored Tweets (*http://spn.tw*) is currently in existence and allows anyone to register to have advertisements posted to their account, with 905 such accounts appearing in our spam dataset.

A second form of spam-as-a-service includes programs that specialize in the sale of Twitter accounts, violating Twitter's Terms of Service [121]. A number sites including *xgcmedia.com* and *backlinksvault.com* purport to register accounts with unique email addresses and create accounts with custom profile images and descriptions. While we cannot directly measure the popularity or impact of these services on Twitter, previous work has examined advertisements for these programs and their associated costs [88]. Both account arbiters and sellers reveal a fledgling market where spammers with content to advertise can obtain access to Twitter accounts without requiring CAPTCHA solvers or other tools to enable automated account creation.

4.5 Persistence of Spam Campaigns Despite Detection

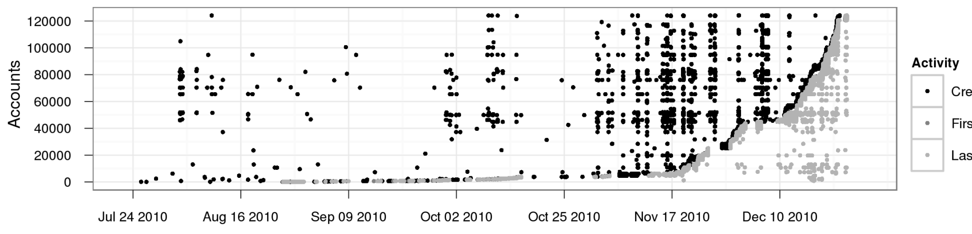
In this section, we explore five major spam campaigns executed on Twitter that highlight the breadth of tools employed by spammers and the ingenuity of their approaches. Some campaigns are executed by centralized controllers orchestrating thousands of accounts, while others exhibit a decentralized spamming infrastructure enabled by spam-as-a-service programs. Only one of the five campaigns advertises content also found in major email spam campaigns [69], leading us to believe some of the actors in the Twitter spam market are separate from the email marketplace dominated by botnets. A summary of each campaign can be found in Table 4.7. Due to the multitude of obfuscation techniques used by spammers, there is no simple mechanism to cluster tweets and accounts into campaigns. As such, we describe our methodology for generating campaigns on a per-campaign basis.

4.5.1 Afraid

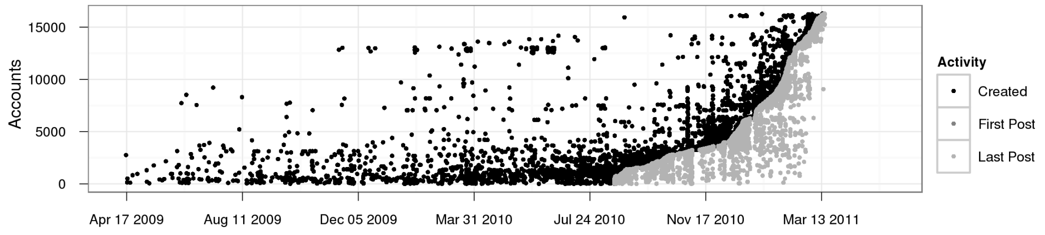
The largest campaign in our dataset consists of *over 14 million tweets and 124,000 accounts.* During a period in December when we first identified the campaign, accounts were distributing Amazon affiliate URLs linking to a variety of products. All the URLs distributed by the campaign directed to custom shorteners that have since disappeared, making further analysis impossible. The sheer volume of spam directing to Amazon underscores the blurred line between what constitutes legitimate content compared to traditional email pharmaceuticals and replica goods. As we will show with two other campaigns, spammers are readily capitalizing on the ability to send unsolicited tweets to large audiences on Twitters to push legitimate goods for their own profit.

Despite regular account suspensions, the campaign sustained itself over a 6 month period, relying on unsolicited mentions to reach out to over 11.7 million distinct Twitter users. As Table 4.7 shows, accounts in the campaign completely ignore the social graph, acquiring a median of two followers throughout their lifetime. Figure 4.12a shows the creation time, activation time, and suspension time for each of the accounts in the campaign. Most dates for activation and suspension overlap due to the short-lived nature of accounts. Accounts are clearly registered in bulk (as indicated by the vertical lines resulting from duplicate registration dates), sometimes *months* in advance of their final activation, leading us to believe accounts were controlled in a centralized fashion.

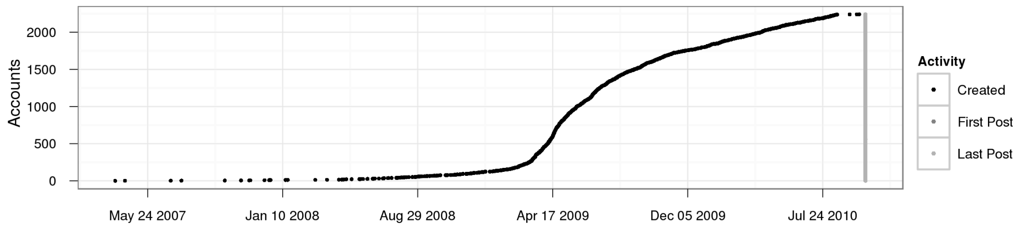
Every tweet of the campaign included at least one unique URL along with a random amalgamation of tweet content stolen from other users' tweets, making text-based clustering



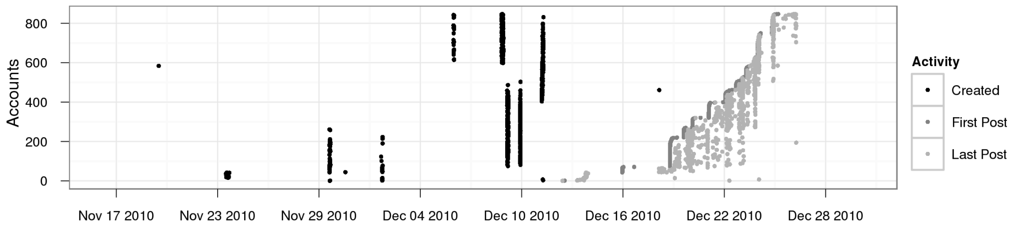
(a) Afraid Campaign



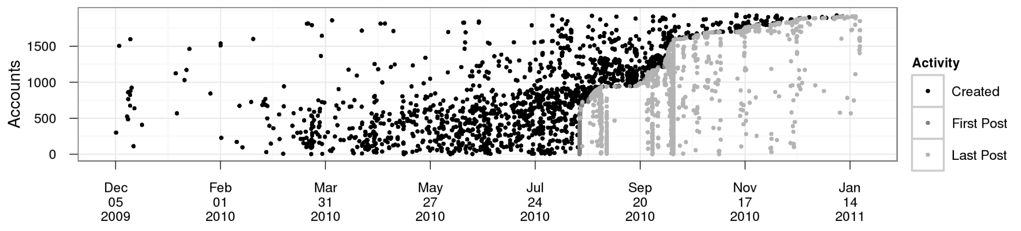
(b) Clickbank Campaign



(c) Yuklumdegga Campaign



(d) Amazon Campaign



(e) Speedling Campaign

Figure 4.12: Prominent spam campaigns on Twitter

difficult. Tweets belonging to the Afraid campaign share no textual similarity other than a rare artifact that multiple retweets are included in a single tweet, violating the definition and functionality of a retweet (i.e. exposing a tweet to an account's audience while maintaining attribution). Additionally, many tweets share the same full domain, though domains alone are not enough to capture all tweets belonging to the campaign. We employ a regular expression to identify tweets with numerous embedded retweets and then group them by the domain advertised. Domain clusters with fewer than tens of thousands of tweets are omitted. The subclusters are finally merged, revealing the full scope of the Afraid campaign. A sample of the campaign's tweets are provided below.

```
@Aguirre_5030 Haha yes for u RT nikivic i love him and i care lol RT dhegracia:  I
don't love you http://ciqf.t6h.ru/HENGK

@mahi58 RT PoiintGod11:  Didn't you just tweet about bad english?  Lol
RT ashLeyGaneshx3:  I didnt get no text http://boo.lol.vc/3GbPH
```

In total, we find over 178 unique domains used exclusively by the campaign, 140 of which rely on *afraid.org* for nameservers. Those domains still being hosted can be resolved, but do not forward traffic to the original campaign landing page.

4.5.2 Clickbank

Clickbank is one of the highest volume spam-as-a-service programs we identify within our dataset, consisting of over 16,000 Twitter accounts each operating in a decentralized fashion controlled by over 200 affiliates. Nearly 13% of *bit.ly* URLs redirect to Clickbank, making Clickbank the most frequent spam domain directed to by the shortener. Figure 4.12b shows the prevalence of accounts tweeting Clickbank throughout time. Clickbank URLs appear consistently from the onset of our collection window to its completion, despite accounts being suspended at regular intervals.

Due to the multiple actors within the campaign, a variety of spamming approaches appear, including the use of unsolicited mentions as well as popular trends. To understand the effectiveness of Clickbank spammers, we take a sample of 20,000 *bit.ly* URLs that direct to the affiliate program and examine their clickthrough. Over 90% of URLs received no clicks at all, though a total of 4,351 clicks were generated for all 20,000 URLs.

We identify tweets belonging to Clickbank participants based on whether the URLs advertised direct to *cbfeed.com* or *clickbank.com*, which serve as intermediate hops for all URLs associated with the service. Our criteria matches both the raw URL appearing in a tweet as well as resolved URLs, provided first-hop data is available.

4.5.3 Yuklemdegga

The Yuklumdegga campaign consists of over 2,200 accounts that pushed pharmacy goods from one of the largest email spam affiliate programs called Eva Pharmacy [61]. Each of the URLs in the campaign resolved to *yuklumdegga.com*, where we identified the HTML store front associated with the Eva Pharmacy program, previously documented by Levchenko et al. [69]. The presence of well-known pharmacy scams on Twitter indicates that some email spam programs are being carried over to Twitter, though we still identify a variety of Twitter spam that does not have a prominent email equivalent.

The Yuklumdegga campaign relies exclusively on hijacking trending topics to reach an audience with its tweets, embedding one of eleven trends that existed on the single day of the campaign's operation. We find 6 of the campaigns 24 URLs directed to *bit.ly* , with an aggregate clickthrough of 1,982 visitors. Assuming the number of visitors was identical for all 24 URLs of the campaign, we can estimate a total of 8,000 users accessed the pharmacy site, all from reading popular trends.

The preparation of the Yuklumdegga campaign provides a stark comparison to other Twitter campaigns. Figure 4.12c shows the creation times of the campaigns' accounts, with activation and suspension times overlapped for the single day of the campaign's existence. The bulk of accounts were created *nearly a year* in advance of the campaign's onset. This can result either from the campaign purchasing accounts from a service, or simply creating accounts in a serial fashion until their final activation.

Given that many of the campaign's accounts were created prior to our collection window, we may incorrectly estimate when an account was activated. We determine this is in fact not the case. We calculate the number of posts sent prior to an account's first tweet in our dataset using statistics embedded in each tweet (described in Section 4.3.1). We find accounts posted a median of 4 posts prior to the campaign's onset, indicating thousands of accounts were stockpiled and then used to send hundreds of tweets.

4.5.4 Amazon

Like the Afraid campaign, a second Amazon affiliate spam campaign appeared simultaneously during the holiday season. With only 848 accounts, the campaign relied on unsolicited mentions to reach out to over 118,000 Twitter users, each pushing a single URL shortened by *bit.ly*. *Bit.ly* reports the URL received an astounding *107,380 clicks* during the course of the URL's existence. Using our estimate of our sample size for that day (derived in Section 4.2.1), we can generate an estimate for how many unsolicited mentions result in a visit. Given we received 70% of URLs posted to Twitter in December, roughly 185,000 tweets would have been sent by the campaign. This would indicate over 58% of users clicked on their unsolicited mention, assuming that no other channels advertised the URL and the absence of automated crawlers visiting the URL.

With respect to the accounts participating in the campaign, Figure 4.12d shows the majority of accounts were created in bulk prior to the holiday and activated in rolling fashion around December 18th. As Twitter banned the active accounts, dormant accounts were then deployed to keep the total accounts active at any point roughly constant. This technique highlights how stockpiled accounts can sustain a campaign through a high-value period.

4.5.5 Speedling

Speedling (*http://www.speedlings.com/*) is a software program used to generate thousands of blogs that embed advertisements as well as Amazon affiliate URLs generated from a product theme such as cookbooks, games, or any arbitrary keyword. Due to the decentralized nature of the Speedling product, where anyone can purchase the software program, multiple approaches to spamming appear. As a result, text-based clustering is impossible. Nevertheless, many Speedling participants rely on a Twitter application provided by the software that is uniquely identified through the Twitter API in the source field as **Go Speedling** or **Speedlings** depending the software version. Other participants do not rely on these APIs, but instead use a shortener that only appears in our spam dataset from Speedling participants. Tweets that satisfy any of these criteria are included in the cluster. We note that this provides a strict lower bound on the presence of Speedling spam as some tweets may not match any of these criteria. A more sophisticated approach would be to cluster based on the HTML template of Speedling-generated blogs. However, as we lack HTML due to the retroactive nature of our analysis and link rot, this is impossible in the context of our current study. In total, we find over 89,526 URLs directing to 1,971 domains, all registered to *speedeenames.com*.

As with Clickbank, Speedling represents a decentralized spam campaign consisting of multiple users of the Speedling software. Figure 4.12e shows the creation and activation time of accounts within the campaign. The vertical lines of final posts indicate mass suspensions on the part of Twitter, yet new accounts are registered, sustaining Speedling's presence on Twitter from the start of our collection period till mid-January.

The monetization approach of Speedling is one of the most interesting compared to the other Twitter campaigns we examine. Visitors are directed to template blog pages listing thousands of Amazon products catering to a specific interest, in addition to a live stream of recommendations from Twitter users that are in fact Twitter accounts controlled by the Speedling operator. Visitors that click on an ad or purchase an Amazon product directly translate into profit, allowing Speedling participants to operate purely through legitimate advertisements. As with the other Amazon affiliate spam we identified, this approach highlights the semi-legitimate nature of some spam campaigns on Twitter, where the distribution approach rather than the landing page or products sold are what distinguish spam from legitimate advertisements.

4.6 Improving Spam Detection in Social Networks

In this section, we discuss the implications of our analysis and how our results compare to previous work on social network spam. We also present potential directions for new spam defenses based on our results, describing both in-network solutions as well as addressing the services that facilitate spamming.

Compromised vs. Fraudulent Accounts: Earlier studies of social networks found that 97% of accounts sending spam on Facebook were compromised [34], compared to 84% of accounts on Twitter (discussed in Chapter 3). In contrast, we find a majority of suspended accounts in our dataset were fraudulent and created for the explicit purpose of spamming. We believe this disparity results from how the datasets for each study were generated. As we showed in Section 4.2.2, only 8% of the URLs posted by fraudulent accounts appeared in blacklists. These same blacklists served as the source of identifying social network spam in the previous studies, with an apparent bias towards compromised accounts. Our results should thus be viewed in conjunction with these previous studies, offering a wider perspective of the multitude of spamming strategies in social networks.

Blacklists and Spam Traps: With Twitter catching an estimated 37% of spam accounts, a number of studies have examined how to improve this accuracy. These approaches include account heuristics that identify newly created accounts and the lack of social relationships and the identification of unsolicited mentions based on the social graph [114, 67, 8, 109]. Each of these approaches hinges on access to accurate training data, which in practice is often difficult to acquire.

One potential solution for aggregating training data and improving spam detection is to develop Twitter-specific blacklists and spam traps. As previous research has shown, existing blacklists are too slow at identifying threats appearing on social networks, as well as often inaccurate with respect to both false positives and negatives [101, 107]. Even though only 10.6% of URLs in our dataset appear in multiple spam tweets, they account for 59.5% of spam. To capture this re-use, as soon as an account is suspended, the URLs it posted and their associated final landing pages could be added to a blacklist along with the frequency they appeared. If Twitter consulted this blacklist prior to posting a URL, services such as Clickbank would be taken offline, while campaigns that persist despite account suspension would be forced to diversify the URLs they post.

Additionally, rather than suspended accounts outright, Twitter could quarantine tweets from known spam accounts, obtaining access to a steady stream of spam URLs for both classification and blacklisting. While such quarantine is standard practice for email, Twitter has the added difficulty that spammers can easily observe the system to confirm the delivery of their tweets. They can do so by either forming relationships between their accounts to monitor tweet delivery (though this risks Sybil detection [139, 21]), or, alternatively, polling the search API confirm whether their spam tweets were indexed. These approaches however incur an additional burden to operating fraudulent accounts.

Beyond Social Networks: The Twitter spam marketplace relies on a multitude of services that include popular URL shorteners, free web hosting, legitimate affiliate programs like Amazon, and illegitimate programs such as Clickbank, Assetiz, and account sellers. While the vast majority of research efforts have targeted spam as it appears on social networks, solutions that disincentivize the abuse of these individual programs would be equally viable. Shortening services, including *bit.ly* and HootSuite, already employ blacklists before URLs are shortened [51, 9]. By monitoring which services underpin the spam ecosystem on Twitter as we do in this study, we can deploy customized countermeasures for each service, reducing the support infrastructure available to spammers.

4.7 Summary of Results

This chapter presents a unique look at the behaviors of spammers on Twitter by analyzing the tweets sent by suspended users in retrospect. We found that the current marketplace for Twitter spam uses a diverse set of spamming techniques, including a variety of strategies for creating Twitter accounts, generating spam URLs, and distributing spam. We highlighted how these features are woven together to form five of the largest spam campaigns on Twitter accounting for nearly 20% of the spam in our dataset. Furthermore, we found an emerging *spam-as-a-service* market that includes reputable and not-so-reputable affiliate programs, ad-based shorteners, and Twitter account sellers.

In particular, we found that 89% of fraudulent accounts created by spammers forgo participation in the social graph, instead relying on unsolicited mentions and trending topics to attract clicks. Surprisingly, 77% of accounts belonging to spammers were suspended within one day, yet despite this attrition rate, new fraudulent accounts are created to take their place, sustaining Twitter spam throughout the course of our seven month measurement. By examining the accounts controlled by individual spammers as revealed by affiliate programs, we find a handful of actors controlling thousands of Twitter accounts, each pushing a diverse strategy for monetizing Twitter. As a whole, our measurements expose a thriving spam ecosystem on Twitter that is unperturbed by current defenses. Our findings highlight the necessity of better spam controls targeting both abusive accounts as well as the services that support the spam marketplace.

Chapter 5

Emerging Threats: Censorship and Astroturfing

5.1 Introduction

Social networks have emerged as a significant tool for both political discussion and dissent. Salient examples include the use of Twitter, Facebook, and Google+ as a medium for connecting United States government officials with citizens to drive public discourse [98, 93, 27]. The Arab Spring that swept over the Middle-East also embraced Twitter and Facebook as a tool for organization [108], while Mexicans have adopted social media as a means to communicate about violence at the hands of drug cartels in the absence of official news reports [38]. Yet, the response to the growing importance of social networks in some countries has been chilling, with the United Kingdom threatening to ban users from Facebook and Twitter in response to rioting in London [45] and Egypt blacking out Internet and cell phone coverage during its political upheaval [103].

While nation states can exert their control over Internet access to outright block connections to social media [130], parties without such capabilities may still desire to control political expression. An example of this recently occurred on Twitter during protests tied to Russia's parliamentary elections [26]. The protests began in Moscow's Triumfalnaya Square and quickly moved online as both pro-Kremlin and anti-Kremlin parties posted to Twitter to express their opinions on Russia's election outcome. In response to these discussions, a wave of bots swarmed the hashtags that legitimate users were using to communicate in an attempt to control the conversation and stifle search results related to the election [63]. This attack highlights the possibility of manipulating social networks for partisan goals through the nefarious use of sybil accounts, sidestepping any requirement for controlling Internet access.

In this chapter we present an in-depth analysis of how unknown parties attempted to control the political conversations surrounding Russia's disputed election. We examine the accounts and infrastructure the attackers relied upon, as well as the impact of their efforts on Twitter users searching for information pertaining to the election and protests. While previous researchers have explored the potential of using posts from sybil accounts to skew product ratings and to generate fake content [53, 89, 128], we show that the attackers specifically adapted spam infrastructure to manipulate political speech. These events demonstrate that malicious parties are now using the *spam-as-a-service* marketplace that has emerged for social networks for multiple ends beyond spam.

The attack consisted of 25,860 fraudulent Twitter accounts used to inject 440,793 tweets into legitimate conversations about the election. We find evidence that these accounts originated from a pool of 975,283 fraudulent accounts, 80% of which remain dormant in preparation for use in future spam campaigns. We contrast the geolocation of logins for legitimate users and those of bots, finding that 56% of logins tied to users discussing the Russian election were located in Russia, compared to just 1% of spam accounts. Equally striking, the attack relied on machines distributed across the globe, 39% of which appear in IP blacklists, a strong indicator that the miscreants involved relied on compromised hosts.

Despite the volume of traffic generated by the attack, its impact was partially mitigated by relevance rankings integrated in search results that aim to filter out spam tweets. On average, search results that used relevance metrics returned 53% fewer bot-generated tweets. These techniques highlight how personalized search results can defend against censorship-based attacks, even in the presence of thousands of fake accounts.

In summary, we frame our contribution as follows:

- We present an in-depth analysis of the profiles, tweets, login behavior, and social graph of accounts attempting to censor political discussion.
- We explore the infrastructure required to carry out such an attack, finding that spam services were re-purposed to enable censorship.
- We characterize the impact of the attack on legitimate users searching for information regarding the election and protests.

5.2 Detecting Political Attacks on Twitter

Before analyzing the attack, we discuss our technique for identifying automated accounts that posted to twenty distinct topics pertaining to the Russian election between December 5—6, 2011. In total, 46,846 accounts participated in discussions of the disputed election results, 25,860 of which we identify as bots. Although these accounts do not fit with traditional views of spam where an account advertises a product or scam, we refer to these accounts as *spam accounts* in this chapter. The other accounts were legitimate users on both sides of the political spectrum.

Hashtag	Translation	Accounts
чп	Catastrophe	23,301
6дек	December 6th	18,174
5дек	December 5th	15,943
выборы	Election	15,082
митинг	Rally	13,479
триумфальная	Triumphal	10,816
победазанами	Victory will be ours	10,380
5dec	December 5th	8,743
навальный	Alexey Navalny [1]	8,256
ridus	Ridus [2]	6,116

Table 5.1: Top 10 hashtags related to the Russian election used between December 5—6.

Statistic	Spam	Nonspam
Accounts	25,860	20,986
Tweets (Dec 5—6, 2011)	440,793	876,774
Tweets (May, 2011—Jan, 2012)	2,445,382	- -

Table 5.2: Summary of accounts who participated in hashtags pertaining to the Russian election (December 5—6) and the activities of spam accounts outside of the election period.

5.2.1 Attacked Hashtags

To characterize the attack, we begin by identifying all of the accounts that posted a tweet containing the hashtag #триумфальная between December 5—6, 2011. This hashtag corresponds with the protests at Moscow's Triumfalnaya Square and was previously reported to be a target of spam accounts [118]. Due to the possibility that the attack targeted multiple hashtags, we take the set of users who tweeted #триумфальная and aggregate all the other hashtags they tweeted with during the attack window, filtering out hashtags with fewer than 1,500 participants. In total, we identify twenty hashtags posted by 46,846 accounts. Table 5.1 shows the top ten of these hashtags and their translation.

In order to identify which accounts were bots, we rely on Twitter's internal spam detection algorithm that monitors abusive behavior including accounts that excessively post to multiple hashtags. At the time of our analysis, the algorithm had suspended 24,203 of the accounts that posted to at least one of the twenty hashtags. In addition to suspended accounts, we include 1,657 accounts that were not suspended but exhibit patterns akin to the bots including similar-looking automatically generated email addresses and creation times correlated with a burst in spam account creation. We discuss the details of how we generate these criteria further in Section 5.3.

[1]Prominent blogger arrested during protest in Moscow

[2]Russian media outlet

5.2.2 Dataset

After identifying all of the accounts that tweeted with hashtags pertaining to the Russian election and labeling them as spam or nonspam, we aggregate all of the tweets sent by both spam and legitimate accounts during December 5—6, 2011. Furthermore, we aggregate all of the tweets sent by spam accounts between May, 2011 when attackers registered their first account until January, 2012 when we started our analysis. In summary, our dataset consists of over 2.4 million spam tweets, 440,793 of which were posted during the attack, as shown in Table 5.2.

In addition to the posting activity of accounts, we build our analysis on registration data and periodic logging tied to each account. This data includes an account's email address, the IP address used to register the account, and all subsequent IP addresses used to access the account between November, 2011—January, 2012. This information allows us to analyze how attackers registered thousands of accounts and the hosts they used to access Twitter.

Finally, in order to gauge the impact of the attack on users searching for information related to the Russian election, we aggregate all of the tweets returned by search queries conducted between December 5—6 that correspond with one of the twenty hashtags attacked. We subsequently identify all of the tweets tied to bots and label them as spam, allowing us to measure the attack's success at diluting search results. We provide a more detailed summary of the search queries performed in Section 5.4.

5.3 Overlap of Attack Infrastructure with Spam-as-a-Service Assets

We deconstruct the attack into three components: the tweets sent prior to and during the attack; the registration data tied to the accounts involved; and the IP addresses that attackers used to access Twitter. We find that the accounts used in the attack generated politically-motivated tweets long before December 5—6, 2011. In order to spread these messages, the attackers acquired thousands of accounts from spam-as-a-service markets that controlled nearly a million fraudulent accounts. Similarly, the attack relied on tens of thousands of compromised machines located around the globe, 39% of which were blacklisted for email spam and malware distribution.

5.3.1 Tweets

In order to control the information users' found when they accessed hashtags pertaining to the Russian election, the attackers posted 440,793 tweets that targeted 20 hashtags organically adopted by users. At its height, the attack generated 1,846 tweets per minute, as shown in Figure 5.1. The entire attack consisted of a short burst in traffic on the first day, followed by a sustained flow of incomprehensible tweets interspersed with partisan jeers the following

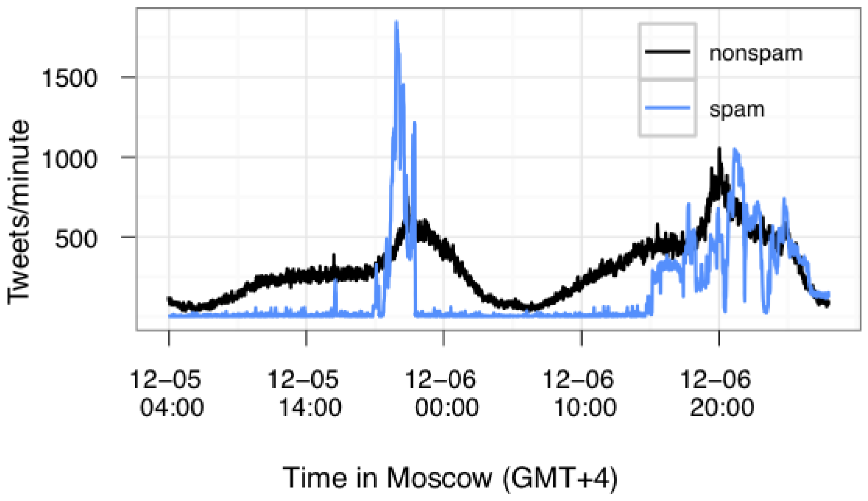


Figure 5.1: Number of tweets sent per minute during the attack on December 5—6. Tweets generated by bots appear in two large spikes beginning around 8PM the first day and 3PM the second day.

day, effectively diluting the content available to Twitter users who were following the election discussions.

For the month prior to the attack, the majority of spam accounts that existed at the time remained dormant, in contrast to legitimate users, per Figure 5.2. However, over the entire course of May, 2011 up until the attack, the bots generated nearly 1.8 million tweets during sporadic periods of activity. The first salvo of coordinated tweets appeared in May, when 4,215 accounts tweeted for the first time with content deriding a prominent Russian anti-corruption blogger. Similar examples occur throughout the dataset when thousands of accounts activate to promote one-sided political opinions interspersed with unrelated news headlines, as determined by Google Translate. Yet, with no legitimate followers or hashtags tied to the early tweets, there was no one to see the content. The uniformity in the types of spam tweets sent, even months before the December 5—6 attack, implies that the accounts were under the sole control of miscreants rather than leased at different intervals. Otherwise, we would expect to observe mismatching messages from competing spammers renting access to the accounts.

5.3.2 Accounts

Registration & Profile

Manipulating Twitter search results using bots requires the acquisition of thousands of accounts that are either fraudulent or compromised. In order to understand where the bot

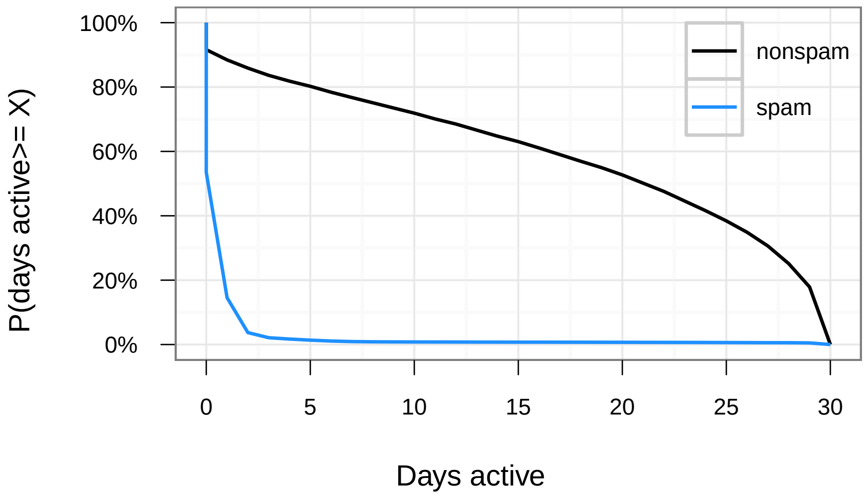
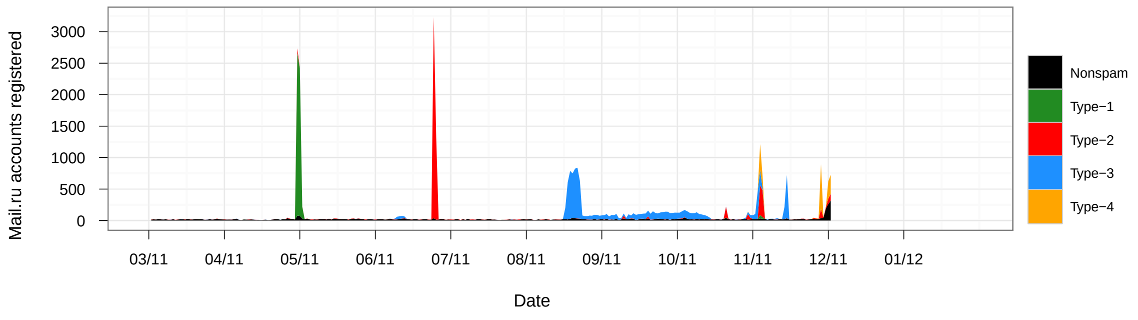


Figure 5.2: Total number of days in November, just prior to the attack, that an account tweets at least once. Nonspam users were frequently active, while spam accounts remained dormant.

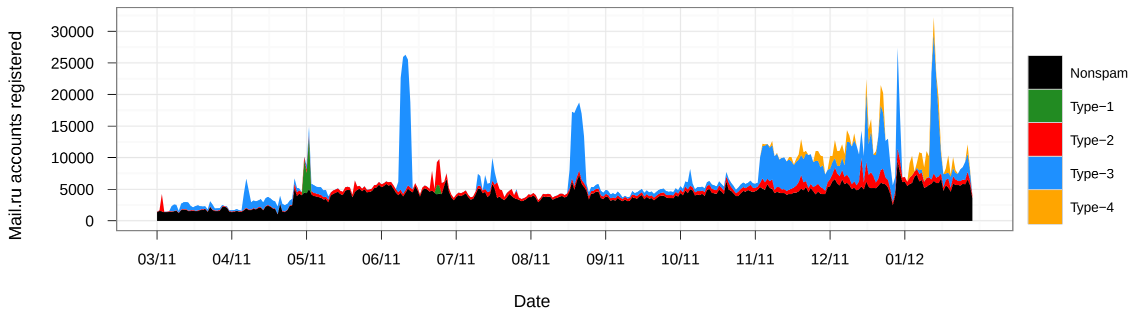
accounts originated from, we begin by analyzing the profiles and registration data tied to each suspended account. We find that 99.5% of the accounts were registered with a distinct mail.ru email address. 95% of these mail.ru email accounts were valid and belonged to the attacker, as indicated by the account's controller clicking on a URL sent to the email addresses after registration. The remaining 5% of mail.ru email addresses were awaiting verification before the account tied to the email was suspended.

Looking into account registration further, we examine the naming conventions used for the screennames, real names, and email addresses of each spam account. We identify a number of patterns tied to bot accounts with mail.ru emails that regularly repeat, but are absent from legitimate users with mail.ru email addresses. Due to the adversarial nature of identifying spam accounts, we do not reveal the patterns, but discuss their accuracy and importance. In total, we identify four distinct patterns of account registrations which we codify into regular expressions which we denote Type-1 through Type-4.

In order to evaluate the accuracy of our expressions at identifying spam accounts, we apply each regex to the 46,846 accounts in our December 5—6 dataset. While this classification approach is simple, our expressions identify an additional 1,657 spam accounts posting to election-based hashtags that were uncaught by Twitter's suspension algorithm. We manually validate the labels for 150 of the newly labeled spam accounts and find only 2% are false positives. Even more impressive, when we apply the expressions to all mail.ru registrations in the past year, we identify *975,283 spam accounts*, only 20% of which Twitter's algorithm had suspended at the time of our analysis. Furthermore, 80% of these accounts have no friends, followers, or tweets despite existing for months. We repeat our manual validation



(a) Registration times of spam accounts used in the attack. Miscreants registered accounts with four distinct profile conventions in noticeable bursts.



(b) Registration times of all mail.ru accounts (note that the scale is 10x of that in the previous plot). Due to a lack of diversity in account profiles, we can readily detect other fraudulent accounts based on the same account signatures as those used in the attack.

Figure 5.3: Pattern of registrations for accounts used in the attack and other accounts registered by the same spam-as-a-service programs where the attackers purchased accounts from.

for 150 of the flagged mail.ru accounts and find only 4% are false positives. Due to the false positive rate, the number of accounts we identify should be treated as a rough estimate of the number of spam accounts registered with mail.ru emails that mirror the accounts used in the attack.

We further validate our classification approach both on accounts within the attack and for all accounts tied to mail.ru email addresses. Figure 5.3a shows the registration dates of bots from March 2011 up until the date of the attack. Miscreants registered accounts in bulk, with account types rarely overlapping during the same period. In contrast, legitimate account registrations are uniformly distributed during the entire period. The registration times for the accounts used in the attack overlap with an abnormal volume of Twitter accounts registered to mail.ru email addresses, shown in Figure 5.3b. The registration spikes in June, August, and January are labeled exclusively as spam, while legitimate registrations remain roughly stable throughout the entire period.

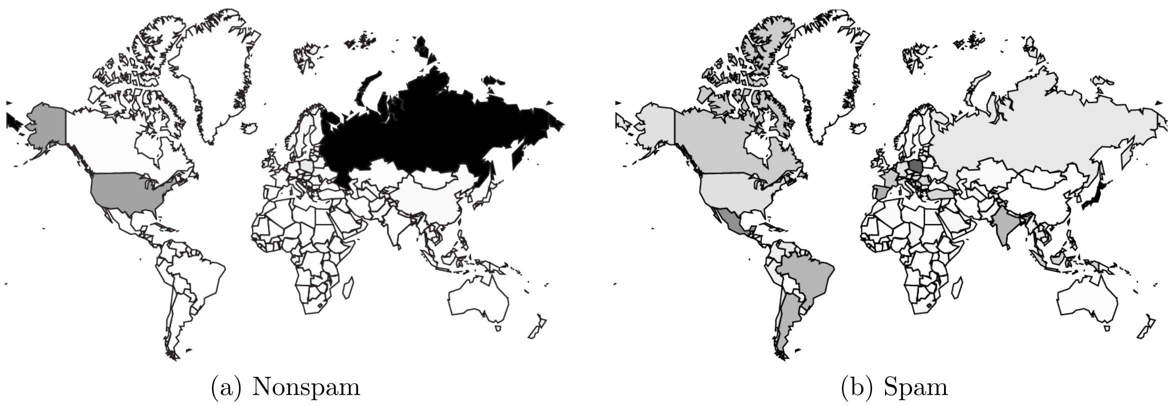


Figure 5.4: Geolocation of user logins. Higher density regions are shown in black. Over 56% of logins tied to legitimate users originate from Russia, compared to only 1% of logins for spam accounts.

In total, the accounts used in the attack represent only 3% of all the mail.ru accounts that our expressions flag as Type 1—4. This indicates the accounts were likely purchased from a spam-as-a-service marketplace that registers and sells accounts in bulk, such as buyaccs.com. These markets have an incredible negative impact on Twitter. For instance, the software registering these Type 1—4 accounts is responsible for *over 80% of fraudulent accounts* tied to mail.ru email addresses suspended by Twitter within the last year. With accounts readily available to any party willing to pay, spam-as-a-service shops simplify the re-purposing of spam infrastructure to whatever end, be it traditional scams or politically motivated censorship.

Social Graph

While most automatically generated accounts rarely engage in forming relationships with other Twitter accounts [116], the spam accounts involved in the attack attempted to simulate a social graph. A median account had 121 following—or outbound relationship—76% of which terminated at other bots. Similarly, a median account had 122 follower—or inbound relationship—85% of which originated from accounts involved in the attack. Even though the attackers acquired accounts registered across multiple months, all of the accounts were used to form a complete sybil network. As a result, all spam accounts involved in the attack that were not singletons were reachable via only spam relationships in an average of 3 hops. The motivation for an attacker to interconnect spam accounts is unclear, but may be a result of assumptions that the presence of social connections will make accounts less susceptible to suspension or improve the relevance ranking of content posted by spam accounts, discussed in Section 5.4.

The presence of a sybil graph is interesting for two reasons. First, it indicates that a single party controlled all of the accounts used in the attack. Relationships between the accounts were formed as far back as May, 2011, requiring coordination between the accounts long before the attack. Furthermore, 80% of the nearly 1 million fraudulent mail.ru accounts we identify have 0 friends and followers and remain dormant. It does not appear that building social relationships is the responsibility of the account creator, providing further evidence that control of the accounts changed hands at some point. We conclude that the miscreants who launched the attack adapted the accounts to their needs and generated social connections, while the party registering the accounts provided them without tweets or relationships.

5.3.3 IP Addresses

Diversity and Lifetime

In addition to acquiring thousands of spam accounts, the attack relied on a diverse body of IP addresses to circumvent Twitter's IP-based restrictions. We find that miscreants registered 84% of the bots with unique IP addresses. After sign up, this diversity decreases; only 49% of 110,189 IP addresses used to access spam accounts between November, 2011—January, 2012 were unique across accounts.

To translate this into the number of machines under the attacker's control, we first examine the lifetime of IP addresses used to access accounts. We find that 80% of the IP addresses tied to the spam accounts were present in our logs from November—January for only a single day. This same phenomenon is true for the 20,986 legitimate accounts, where 84% of IP addresses used to access the accounts persist for one day. We performed a reverse DNS lookup on all the IPs tied to the bots and find that each of the IP addresses belongs to ISP address pools. The hosts tied to these IP addresses are likely residential, as indicated by the presence of `dsl`, `cable`, `dynamic` and a number of other heuristics in the reverse lookup's naming convention.

Due to heavy churn in IP addresses over time, it is difficult to estimate the number of unique hosts ever used by attackers. Instead, we limit our analysis to a single day. At the height of the attack on December 6th, 11,356 unique IP addresses were used to access spam accounts. If we assume that IP addresses are stable for at least a day, then tens of thousands of hosts were available to the attackers.

Geolocation and Origin

In order to understand where the hosts controlled by the attackers originate from and how they compare to legitimate users, we examine the geolocation of IP addresses used by both types of parties. To start, we generate a list of the unique IP addresses used to access each of the 46,846 accounts between November, 2011—January, 2012. We then map these to their country of origin using the MaxMind database [74] and aggregate the totals across spam and legitimate accounts.

Figure 5.4 shows our results. 56% of all legitimate logins originate from Russia, compared to only 1% of logins tied to spam accounts. The IPs used by the attack are located around the globe, with Japan accounting for the largest set of logins (14%). These results imply that the bots are using compromised machines or proxy services to access Twitter.

Blacklist Membership

If attackers relied on compromised machines, there is a possibility that the hosts used to access Twitter were also used by other parties—or the same party—for other malicious behavior such as distributing email spam [11]. To this end, we used a list of 47 million suspicious IP addresses taken from the CBL blacklist [12], which contains IPs flagged for email spam and spreading malware. We then tested whether an IP addresses used to access any of the 25,860 accounts employed in the attack ever appears in the blacklist between October 2011—January 2012. When we perform our analysis, we ignore the timestamp for when an IP address is listed and unlisted to account for any delay between an attacker using an IP address and its subsequent blacklisting. However, this approach may also overestimate the number of malicious hosts.

We find that the CBL blacklist contains 39% of the IP addresses tied to bots. This indicates that hosts used to attack Twitter are also used by a malicious party to generate spam or distribute malware. However, in order to judge the accuracy of the blacklists, we repeat the same experiment using a list of IP addresses used to access legitimate accounts. We find that 21% of benign IP addresses are also listed. While we cannot definitively determine why blacklists are flagging IPs tied to legitimate users, it may result from blacklists biasing their classification of Russian IPs, or arise due to DHCP churn causing aliasing with other infected hosts. The imprecision we detect in blacklists reiterates previous research on the limitations of blacklists [107], especially in the context of social networks as we discussed in Chapter 3 and Chapter 4. Nevertheless, because IPs used in the attack are more likely to be listed, we can infer that some of the attack's hosting infrastructure was simultaneously used for more traditional spam/malware activities.

5.4 Impact of Political Attacks on Free Access to Information

The attack on Twitter is a compelling example of how miscreants can adapt spam infrastructure to censor legitimate access to relevant information surrounding controversial events. However, even with control of thousands of fraudulent accounts and compromised machines, the attack was partly mitigated by Twitter's relevance ranking of tweets, which personalizes search results and emphasizes popular content. We provide a brief overview on the different search mechanisms available to Twitter users before evaluating the fraction of search results that were affected by politically-slanted spam.

Search Mode	Tweets Returned
Real-time	2,923,022
Relevance	17,276,281
Relevance (top 5 most recent)	3,743,919

Table 5.3: Number of tweets returned to users searching for hashtags related to the Russian election.

5.4.1 Search: Relevance vs. Real-time

Twitter search offers two modes of operation: *real-time* and *relevance* mode. Real-time mode returns tweets in order of most recently posted first. This type of indexing is susceptible to message dilution attacks as spammers merely need to outproduce legitimate content that users post to hashtags. In contrast, the relevance search mode incorporates signals that capture the popularity of a tweet while at the same time surfacing content from accounts whose social graph and interests overlap those of the account submitting a search query [125]. As a result, the algorithm ranks content by its importance, reducing the impact of mass producing tweets on a single topic. However, to add some dynamism to search results to prevent popular content from being locked at the top, the freshness of a tweet is also considered in the ordering of the most relevant tweets. By default, Twitter returns relevance-ranked searches.

5.4.2 Search Pollution

To measure the frequency of spam in search results, we aggregate all of the tweets returned by queries performed between December 5—6, 2011 related to one of the attacked hashtags. If a bot posted a tweet that appears in the search results, we assume that tweet was spam. We assume all other accounts and tweets are legitimate. On average, searches return 15 tweets per query. We consider all of these tweets in our analysis even though users may only view a fraction when searching. Consequently, our analysis may overestimate a user's perception of spam in search results.

Table 5.3 shows a summary of the data used in our analysis. Twitter users generated over 233,000 real-time search queries related to the election. In aggregate, these search results contained 2.9 million tweets. Users relied on the default relevance-ranked search far more frequently. In total, users performed 1.1 million relevance searches which returned 17 million tweets. Analyzing the fraction of spam in each search query, we find that relevance-based searches returned 53% fewer spam tweets compared to real-time searches, a testament to the volume of tweets produced by bots in the real-time feed. If we restrict our analysis to the five most recent relevance-ranked tweets that were returned in search—those that appear at the top of the page and are most likely to be seen—we find that relevance-mode returned 64% fewer spam tweets. These results highlight that integrating a user's social graph and interests into the tweets returned by searches can to a degree mitigate the impact of message dilution attacks.

5.5 Summary of Results

We have analyzed how attackers adapted fraudulent accounts and compromised host—traditionally tied to spamming—in order to control political speech on Twitter. In particular, we examined an attack launched by unknown miscreants that leveraged 25,860 accounts to send 440,793 tweets in order to disrupt conversations about the Russian election, protests, and purported fraud. We showed that the accounts used in the attack were likely purchased from a spam-as-a-service program that controlled at least 975,283 Twitter accounts and mail.ru email addresses. In contrast to legitimate Russian users participating in discussions of the election, only 1% of the IP addresses used by attackers originated in Russia. Instead, the attackers controlled hosts located around the globe, over 39% of which were blacklisted for involvement in more classic spam activities. Despite the large volume of malicious tweets, Twitter's search relevance algorithm, which personalizes search results and weights popular content, eliminated 53% of the tweets sent during the attack compared to the real-time search results with no protections, offering a promising approach for defending against future censorship attacks.

Chapter 6

Developing Real-Time and Scalable Spam Detection

6.1 Introduction

Following on the heels of the massive proliferation of web services—including social networks, video sharing sites, blogs, and consumer review pages that draw in hundreds of millions of viewers— phishing, malware, and scams have become a regular threat [17, 84, 55]. Bypassing protection mechanisms put in place by service operators, scammers are able to distribute harmful content through the use of compromised and fraudulent accounts. As spam evolves beyond email and becomes a regular nuisance of web services, new defenses must be devised to safeguard what is currently a largely unprotected space.

While email spam has been extensively researched, many of the solutions fail to apply to web services. In particular, recent work as well as our findings in Chapter 3 have shown that domain and IP blacklists currently in use by social network operators and by URL shortening services [59, 110, 51, 9] perform too slowly (high latency for listing) and inaccurately for use in web services [101, 107]. Alternative solutions, such as account-based heuristics that are specifically designed to identify automated and suspicious behavior in web services [8, 67, 114], focus on identifying accounts generated by spammers, and thus have limited utility in detecting misuse of compromised accounts. They also can incur delays between a fraudulent account's creation and its subsequent detection due to the need to build a history of abuse. Given these limitations, we seek to design a system that operates in *real-time* to limit the period users are exposed to spam content; provides *fine-grained* decisions that allow services to filter individual messages posted by users; but functions in a manner *generalizable* to many forms of web services.

To this end we design Monarch, a real-time system that crawls URLs as they are submitted to web services and determines whether the URLs direct to spam content. For our study, we define spam to include scams advertising pharmaceuticals, adult content, and other solicitations, phishing that attempts to capture account credentials, and pages attempting to distribute malware. By restricting our analysis to URLs, Monarch can provide spam protection regardless of the context in which a URL appears, or the account from which it originates. This gives rise to the notion of *spam URL filtering as a service*. Monarch frees other web services from the overhead of reinventing spam classifiers and the accompanying infrastructure components.

The architecture of Monarch consists of three core elements: a front-end that accepts URLs submitted by web services seeking a classification decision, a pool of browsers hosted on cloud infrastructure that visits URLs to extract salient features, and a distributed classification engine designed to scale to tens of millions of features that rapidly returns a decision for whether a URL leads to spam content. Classification builds upon a large foundation of spam characteristics [71, 77, 127, 4, 96, 30, 137, 72, 73, 132] and includes features drawn from the lexical properties of URLs, hosting infrastructure, and page content (HTML and links). We also collect new features including HTTP header content, page frames, dynamically loaded content, page behavior such as JavaScript events, plugin usage, and a page's redirection behavior. Feature collection and URL classification occur at the time a URL is submitted to our service, with the overall architecture of Monarch scaling to millions of URLs to satisfy the throughput expected of large social networks and web mail providers.

In this chapter, we evaluate the viability of Monarch as a real-time filtering service and the fundamental challenges that arise from the diversity of web service spam. We show that Monarch can provide accurate, real-time protection, but that the underlying characteristics of spam do not generalize across web services. In particular, we leverage Monarch's feature collection infrastructure to study distinctions between 11 million URLs drawn from email and Twitter. We find that spam targeting email is qualitatively different from Twitter spam, requiring classifiers to learn two distinct sets of rules to ensure accuracy. A basic reason for this distinction is that email spam occurs in short-lived campaigns that quickly churn through spam domains, while spam on Twitter consists of long lasting campaigns that often abuse public web hosting, generic redirectors, and URL shortening services.

Our evaluation also includes an analysis of which URL features serve as the strongest indicators of spam and their persistence as spam evolves. We find that classification requires access to every URL used to construct a landing page, HTML content, and HTTP headers to ensure the best accuracy. In contrast, relying solely on DNS entries or the IP address of spam infrastructure achieves much less accuracy. Furthermore, without regular retraining and access to new labeled spam samples, accuracy quickly degrades due to the ephemeral nature of spam campaigns and their hosting infrastructure.

We deploy a full-fledged implementation of Monarch to demonstrate its scalability, accuracy, and run-time performance at classifying tweet and email spam URLs. Using a modest collection of cloud machinery, we process 638,000 URLs per day. Distributed classification achieves an accuracy of 91% (0.87% false positives) when trained on a data set of nearly 50 million distinct features drawn from 1.7 million spam URLs and 9 million non-spam URLs, taking only one hour to produce a model. While the current false positive rate is not optimal, we discuss several techniques that can either lower or ameliorate their impact in Section **??**. During live classification, each URL takes on average 5.54 sec to process from start to finish.

This delay is unavoidable and arises from network requests made by the browser, which is difficult to speed up; only 1% of overhead comes from instrumenting the browser for feature collection. The cloud infrastructure required to run Monarch at this capacity costs $1,587 for a single month. We estimate that scaling to 15 million URLs per day would cost $22,751 per month, and requires no changes to Monarch's architecture.

In summary, we frame our contributions as:

- We develop and evaluate a real-time, scalable system for detecting spam content in web services.
- We expose fundamental differences between email and Twitter spam, showing that spam targeting one web service does not generalize to other web services.
- We present a novel feature collection and classification architecture that employs an instrumented browser and a new distributed classifier that scales to tens of millions of features.
- We present an analysis of new spam properties illuminated by our system, including abused free hosting services and redirects used to mask spam web content.
- We examine the salience of each feature used for detecting spam and evaluate their performance over time.

6.2 Design Goals for Browser-based Spam Detection

In this work we present the design and implementation of Monarch, a 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.

We show the overall intended operation of Monarch in Figure 6.1. Monarch runs as an independent service to which any web service can provide URLs to scan and classify. During the period it takes for Monarch's classification to complete, these services can either delay the distribution of a URL, distribute the URL and retroactively block visitors if the URL is flagged as spam (risking a small window of exposure), or employ a heavier-weight verification process to enforce even stricter requirements on false positives than are guaranteed by classification.

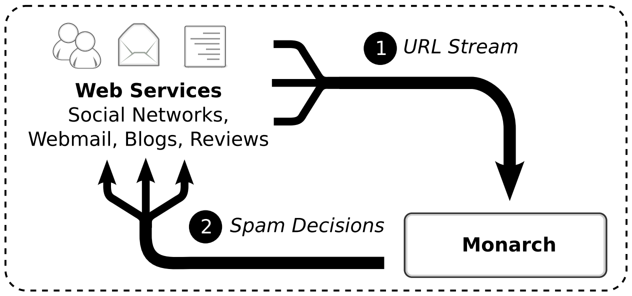


Figure 6.1: Intended operation of Monarch. Web services provide URLs posted to their sites for Monarch to classify. The decision for whether each URL is spam is returned in real-time.

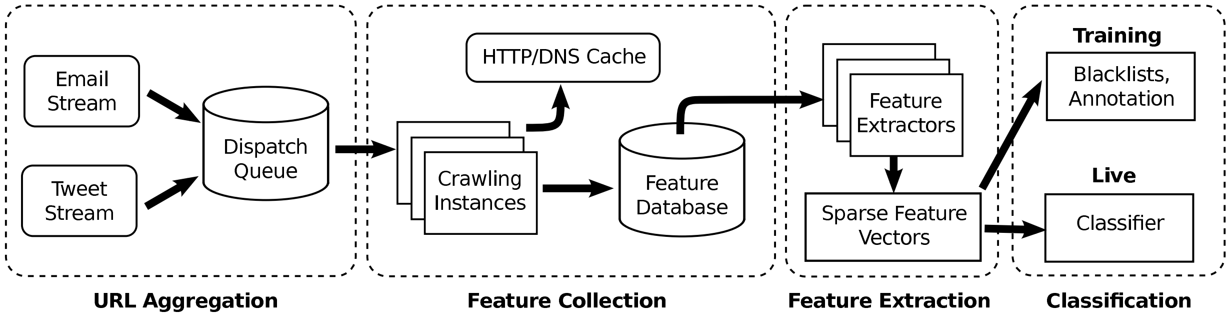


Figure 6.2: System flow of Monarch. URLs appearing in web services are fed into Monarch's cloud infrastructure. The system visits each URL to collect features and stores them in a database for extraction during both training and live decision-making.

6.2.1 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 that 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.

6.2.2 System Flow

Figure 6.2 shows Monarch's overall internal system flow. URLs posted to web services are fed into a dispatch queue for classification. The system visits each URL to collect its associated raw data, including page content, page behavior, and hosting infrastructure. It then transforms these raw features into meaningful boolean and real-valued features and provides these results to the classifier for both training and live decision-making. During live classification, Monarch's final decision is returned to the party that submitted the URL; they can then take appropriate action based on their application, such as displaying a warning that users can click through, or deleting the content that contained the URL entirely. We now give an overview of each component in this workflow.

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 in Section 6.6.

Feature Collection: During feature collection, the system visits a URL with an instrumented version of the Firefox 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 in Section 6.3. 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.

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.

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 in Section 6.4.

6.3 Content and Hosting Feature Collection

Classification hinges on having access to a robust set of features derived from URLs to discern between spam and non-spam. Previous work has shown that lexical properties of URLs, page content, and hosting properties of domains are all effective routes for classification [72, 73, 132, 77, 71]. We expand upon these ideas, adding our own sources of features collected by one of three components: a *web browser*, a *DNS resolver*, and *IP address analysis*. A comprehensive list of features and the component that collects them can be found in Table 6.1. A single monitor oversees multiple copies of each component to aggregate results and restart failed processes. In turn, the monitor and feature collection components are bundled into a *crawling instance* and replicated in the cloud.

6.3.1 Web Browser

Within a crawling instance, a web browser provides the primary means for collecting features for classification. Due to real-time requirements, a trade-off arises between expedited load times and fidelity to web standards. Given the adversarial nature of spam, which can exploit poor HTML parsing or the lack of JavaScript and plugins in a lightweight browser [13, 37], our system employs an instrumented version of Firefox with JavaScript enabled and plugin applications installed including Flash and Java. As a URL loads in the browser, we monitor a multitude of details, including redirects, domains contacted while constructing a page, HTML content, pop-up windows, HTTP headers, and JavaScript and plugin execution. We now explain the motivation behind each of these raw features and the particulars of how we collect them.

Initial URL and Landing URL: As identified by earlier research [73, 77], the lexical features surrounding a URL provide insight into whether it reflects spam. The length of a URL, the number of subdomains, and terms that appear in a URL all allow a classifier to discern between *get.cheap.greatpills.com* and *google.com*. However, given the potential for nested URLs and the frequent use of shortening services, simply analyzing a URL presented to our service does not suffice. Instead, we fetch each URL provided to the browser, allowing

Source	Features	Collected By
Initial URL, Final URL	Domain tokens, path tokens, query parameters, is obfuscated?, number of subdomains, length of domain, length of path, length of URL (From here on out, we denote this list as URL features)	Web browser
Redirects	URL features for each redirect, number of redirects, type of redirect	Web browser
Frame URLs	URL features for each embedded IFrame	Web browser
Source URLs	URL features for every outgoing network request; includes scripts, redirects, and embedded content	Web browser
HTML Content	Tokens of main HTML, frame HTML, and script content	Web browser
Page Links	URL features for each link, number of links, ratio of internal domains to external domains	Web browser
JavaScript Events	Number of user prompts, tokens of prompts, onbeforeunload event present?	Web browser
Pop-up Windows	URL features for each window URL, number of windows, behavior that caused new window	Web browser
Plugins	URL features for each plugin URL, number of plugins, application type of plugin	Web browser
HTTP Headers	Tokens of all field names and values; time-based fields are ignored	Web browser
DNS	IP of each host, mailserver domains and IPs, nameserver domains and IPs, reverse IP to host match?	DNS resolver
Geolocation	Country code, city code (if available) for each IP encountered	IP analysis
Routing Data	ASN/BGP prefix for each IP encountered	IP analysis

Table 6.1: List of features collected by Monarch

the browser to log both the initial URL provided as well as the URL of the final landing page that results after executing any redirects.

Redirects: Beyond the initial and final landing URL, the redirect chain that occurs in between can provide insight into whether a final page is spam. Suspiciously long redirect chains, redirects that travel through previously known spam domains, and redirects generated by JavaScript and plugins that would otherwise prevent a lightweight browser from proceeding all offer insight into whether the final landing page reflects spam. To capture each of these behaviors, the web browser monitors each redirect that occurs from an initial

URL to its final landing page. This monitoring also includes identifying the root cause of each redirect; whether it was generated by a server 30X HTTP response, meta refresh tag, JavaScript event, or plugin (e.g., Flash).

Sources and Frames: In the case of mashup pages with spam content embedded within a non-spam page, the URL of a final page masks the presence of spam content. This is particularly a problem with URL shortening services, including *ht.ly* and *ow.ly*, which embed shortened URLs as IFrames. To recover information about embedded content, the web browser monitors and logs all frames, images, and ad URLs it contacts during the construction of a page. The browser also collects a list of all outgoing network requests for URLs, regardless whether the URL is for a top level window or frame, and applies a generic label called *sources*.

HTML Content: Beyond features associated with URLs, the content of a page often proves indicative of the presence of spam [92, 142, 132]. This includes the terms appearing on a page and similar layout across spam webpages. To capture page content, the web browser saves a final landing page's HTML in addition to the HTML of all subframes on the page. Naturally, we cannot collect HTML features for image-based spam or for media content such as PDFs.

Page Links: The links appearing on a final landing page offer some insight into spam. While the web browser only follows URLs that automatically load (it does not crawl embedded links such as HREFs), if a page contains a URL to a known spam page, then that can help to classify the final landing page. Similarly, search engine optimization techniques where a page comes stuffed with thousands of URLs to an external domain also suggests misbehavior. To capture both of these features, the web browser parses all links on a final landing page. Each link is subjected to the same analysis as frames and redirects. Afterwards, we compute the ratio of links pointing at internal pages versus external domains.

JavaScript Events: In addition to the content of a page, observing an attempt to force the user to interact with a page—such as pop-up boxes and prompts that launch before a user navigates away from a page—strongly indicates spam. To identify this behavior, the web browser instruments all dialog messages that would normally require some user action to dismiss, including alerts, input boxes, and onbeforeunload events. When a dialog box occurs, the browser silently returns from the event, logging the text embedded in the dialog. If a return value is expected such as with an input box, the browser provides a random string as a response. The browser saves as features the number of dialogs that occur, the text of the dialogs, and the presence of an onbeforeunload event.

Pop-up Windows: As with pop-up dialogs, pop-up windows are a common feature of spam. Whenever a pop-up window occurs, the browser allows the window to open, instrumenting the new page to collect all the same features as if the URL had originated from the dispatch queue. It records the parent URL that spawned the pop-up window, along with whether the page was launched via JavaScript or a plugin. After all windows have finished loading (or upon a timeout), the browser saves the total number of pop-up windows spawned and the features of each window and associates them with the parent URL.

Plugins: Previous reports have shown that spammers abuse plugins as a means to redirect victims to a final landing page [117, 37]. To capture such plugin behavior, our browser monitors all plugins instantiated by a page, the application type of each plugin (e.g., Java, Flash), and finally whether a plugin makes any request to the browser that leads to an outgoing HTTP request, causes a page to redirect, or launches a new window.

HTTP Headers: The HTTP headers that result as the browser loads a landing page provide a final source of information. Header data offers insight into the servers, languages, and versions of spam hosts, in addition to cookie values and custom header fields. We ignore HTTP fields and values associated with timestamps to remove any bias that results from crawling at particular times.

6.3.2 DNS Resolver

While spammers rapidly work through individual domain names during the course of a campaign, they often reuse their underlying hosting infrastructure for significant periods [30, 4, 127, 73]. To capture this information, once the web browser finishes processing a page, the crawler instance manager forwards the initial, final, and redirect URLs to a DNS resolver. For each URL, the resolver collects hostnames, nameservers, mailservers, and IP addresses associated with each domain. In addition, we examine whether a reverse lookup of the IP addresses reported match the domain they originated from. Each of these features provides a means for potentially identifying common hosting infrastructure across spam.

6.3.3 IP Address Analysis

Geolocation and routing information can provide a means for identifying portions of the Internet with a higher prevalence of spam [127]. To extract these features, we subject each IP address identified by the DNS resolver to further analysis in order to extract geolocation and routing data. This includes identifying the city, country, ASN, and BGP prefix associated with each address.

6.3.4 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.

6.3.5 Feature Extraction

In preparation for classification, we transform the unprocessed features gathered during feature collection into a meaningful feature vector. We first canonicalize URLs to remove obfuscation, domain capitalization, and text encoding. Obfuscation includes presenting IP addresses in hex or octet format, or embedding path-traversal operations in the URL path. By reducing URLs to a canonical form, we can assure that common URL features match consistently across occurrences and cannot be masked by lexical transformations that would otherwise result in the same browser behavior. To compensate for potential lost information, we also include a boolean feature reflecting the presence of an obfuscated URL.

Once canonicalized, we split a URL into its basic components of domain, path, and query parameters, each of which we tokenize by splitting on non-alphanumeric characters. We apply a similar process to HTML and any text strings such as HTTP headers, where we tokenize the text corpus into individual terms and treat them as an unsorted bag of words. We then convert the results of tokenization into a binary feature vector, with a flag set for each term present. Rather than obscuring the origin of each token, we construct separate feature groups to indicate that a feature appeared in a redirect versus HTML content. Given the potential for millions of features, we represent feature vectors as sparse hash maps, which only indicate the presence of a feature. Finally, we provide these sparse maps to the classifier for training and decision making.

6.4 Parallelizing Classification as a Distributed Logistic Regression

For Monarch, we want a classifier that we can train quickly over large sets of data. Our first design decision along these lines was to use linear classification, where the output classification model is a weight vector $\vec{w}$ that describes a hyperplane that separates data points placed in a high-dimensional space. We choose linear classification because of its simplicity, scalability and interpretability—for these reasons, linear classifiers have become common-place for web service providers interested in large-scale anti-phishing and anti-spam classification [132, 131].

In addition to quick training, we want the classifier to fit in memory. With these goals in mind, we design our training algorithm as a parallel online learner with regularization to yield a sparse weight vector. In particular, we combine the strategies of iterative parameter mixing [76] and subgradient L1-regularization [22]. Although more sophisticated algorithms exist that could yield higher accuracy classifiers at the expense of more training time, we favor a design that can yield favorable classification accuracies with less training time.

6.4.1 Notation

The problem of identifying spam URLs is an instance of binary classification. For a given URL, the data point $\vec{x} \in \mathbb{R}^d$ represents its feature vector in a high-dimensional space with d features. Because $\vec{x}$ is sparse (typically 1,000—1,500 nonzero entries out of an extremely large feature space of $d > 10^7$), we represent the feature vector as a hash map. We label the data point with an accompanying class label $y \in \{-1, +1\}$. $y = -1$ represents a non-spam site, and a $y = +1$ represents a malicious site.

To predict the class label of a previously unseen example $\vec{x}$ (representing the URL's feature vector), we train a linear classifier characterized by weight vector $\vec{w}$ trained offline on a labeled set of training data. During testing or deployment, we compute a predicted label as the sign of the dot product between the weight vector and the example: $\hat{y} = \text{sign}(\vec{x} \cdot \vec{w})$. If the predicted label $\hat{y} = +1$ but the actual label $y = -1$, then the error is a false positive. If $\hat{y} = -1$ but $y = +1$, then the error is a false negative.

6.4.2 Logistic Regression with L1-regularization

For training the weight vector $\vec{w}$, we use logistic regression (LR) with L1-regularization [47]. Given a set of n labeled training points $\{(\vec{x}_i, y_i)\}_{i=1}^n$, the goal of the training process is to find $\vec{w}$ that minimizes the following objective function:

$$f(\vec{w}) = \sum_{i=1}^{n} \log(1 + \exp[-y_i(\vec{x}_i \cdot \vec{w}_i)]) + \lambda \|\vec{w}\|_1. \tag{6.1}$$

The first component of the objective constitutes the log-likelihood of the training data as a function of the weight vector—it is an upper bound on the number of mistakes made during training, and solving this proxy objective is a tractable alternative to directly minimizing the training error. For a single example $(\vec{x}_i, y_i)$, we can minimize the value of $\log(1 + \exp[-y_i(\vec{x}_i \cdot \vec{w}_i)])$ if the classification margin $y_i(\vec{x}_i \cdot \vec{w}_i)$ is a large positive value. (The margin is proportional to the distance between $\vec{x}_i$ and the classifier's decision boundary—a positive value means it is on the correct side of the boundary, and a negative value means it is on the incorrect side.) Thus, a solution that minimizes $f(\vec{w})$ would ideally yield a positive classification margin for as many examples as possible.

The second component is the regularization—it adds a penalty to the objective function for values where the L1 norm of $\vec{w}$ is large ($\|\vec{w}\|_1 = \sum_{j=1}^{d} w_j$). L1-regularization tends to

Algorithm 1 Distributed LR with L1-regularization

```
  Input: Data $\mathbf{D}$ with $m$ shards
  Parameters: $\lambda$ (regularization factor), $I$ (no. of iterations)
  Initialize: $\vec{w} = \vec{0}$
  **for** $i = 1$ to $I$ **do**
      (gradient) $\vec{g}^{(j)} = \text{LRsgd}(\vec{w}, \mathbf{D}_j)$ for $j = 1..m$
      (average) $\vec{w} = \vec{w} - \frac{1}{m}\sum_{j=1}^{m} \vec{g}^{(j)}$
      (shrink) $w_\alpha = \text{sign}(w_\alpha) \cdot \max(0, |w_\alpha| - \lambda)$ for $\alpha = 1..d$
  **end for**
```

Algorithm 2 Stochastic gradient descent for LR (LRsgd)

```
  Input: $\vec{w}$ (weight vector), $\mathbf{D}_j$ (data shard)
  Parameters: $\eta$ (learning rate)
  Initialize: $\vec{g}_0 = \vec{w}$
  **for** $t = 1$ to $|\mathbf{D}_j|$ **do**
      Get data point $(\mathbf{x}_t, y_t)$ from $\mathbf{D}_j$
      Compute margin $z = y_t(\vec{x}_t \cdot (\vec{w} - \vec{g}_{t-1}))$
      Compute partial gradient $\vec{h} = y_t([1 + e^{-z}]^{-1} - 1)\vec{x}_t$
      $\vec{g}_t = \vec{g}_{t-1} + \eta\vec{h}$
  **end for**
  Return: $\vec{g}_{|\mathbf{D}_j|}$
```

yield sparse weight vector—where there are relatively few nonzero feature weights. This is useful for applications that may be memory-constrained and require sparse solutions. (By contrast, using the L2 norm, another popular form of regularization, would yield solutions whose weights have small magnitudes but that tend to be non-sparse.) The parameter λ governs the amount of regularization: a higher λ gives the second component of Equation 6.1 more weight relative to the first component and will yield a more sparse solution.

Many optimization strategies exist for minimizing the objective in Equation 6.1. We had particular interest in a strategy amenable to learning over large-scale data sets in a short amount of time. We settled on a combination of recently-developed distributed learning and regularization techniques, which we describe in the next section.

6.4.3 Training Algorithm

We first divide the training data into m shards (which occurs by default storing data on certain distributed file systems such as the Hadoop Distributed File System [44]). Then, we distribute the initial model weight vector $\vec{w}$ to the m shards for training by stochastic gradient descent (Algorithm 1, "gradient" step).

Within each shard, we update the weight vector using a stochastic gradient descent for logistic regression (Algorithm 2). We update the weight vector one example at a time as we read through the shard's data (this is also known as online learning). Under the constraint where we can only read each data item once within a shard, updating the model incrementally after every example typically has good convergence properties. As a performance optimization, we return the sum of the partial gradients rather than the updated weight vector itself.

After the m shards update their version of the weight vector, we collect the partial gradients $\vec{g}^{(1)}..\vec{g}^{(m)}$ and average them (Algorithm 1, "average" steps). Then, we perform L1-regularization (Algorithm 1, "shrink" step) on the averaged weight vector using a truncation function with threshold λ —this only applies to feature weights corresponding to binary features. In particular, all feature weights w_i with magnitude less than or equal to λ are set to 0, and all other weights have their magnitudes reduced by λ. This procedure reduces the number of nonzero weight vector entries, allowing the resulting weight vector to occupy less memory. Because there are fewer real-valued features (about 100) than binary features (about 10^7), we do *not* regularize the feature weights corresponding to real-valued features.

After the shrinkage step, we distribute the new weight vector $\vec{w}$ to the m shards again to continue the training process. The process repeats itself for I iterations.

A number of practical issues arise in getting the distributed logistic regression to scale to large-scale data. We describe how we implement our classifier in Section 6.5.4.

6.4.4 Data Set and Ground Truth

Our data set for training and testing the classifier consists of three sources: URLs captured by spam traps operated by major email providers, blacklisted URLs appearing on Twitter, and non-spam URLs appearing on Twitter that are used to represent a non-spam data sample. In total, we use Monarch's feature collection infrastructure over the course of two months to crawl 1.25 million spam email URLs, roughly 567,000 blacklisted Twitter URLs, and over 9 million non-spam Twitter URLs. Due to blacklist delay, generating our spam set of Twitter URLs requires retroactively checking all our Twitter URLs against 5 blacklists: Google Safebrowsing, SURBL, URIBL, Anti-Phishing Work Group (APWG), and Phishtank. If at any point after a URL is posted to Twitter its landing page, any of its redirects, frame URLs, or any of its source URLs become blacklisted, we treat the sample as spam. A breakdown of the categories of spam identified on Twitter can be seen in Table 6.2; 36% of blacklisted URLs were flagged as scams, 60% as phishing, and 4% as malware. A breakdown for email categories is not available, but the sample is known to contain scams, phishing, and malware.

In general, we lack comprehensive ground truth, which complicates our overall assessment of Monarch's performance. We may misclassify some true spam URLs as nonspam given absence of the URL in our spam-trap and blacklist feeds. Thus, we may somewhat underestimate false negatives (spam that slips through) seen in live operation, and overestimate false positives (legitimate URLs tagged as spam). In practice, building a training set of spam and non-spam samples remains a challenge for Monarch, requiring either user reports or spam traps operated by the web services seeking protection. However, for the purposes

Blacklist	Detected URLs
Anti-Phishing Work Group	350,577
Google Safebrowsing (Phishing)[1]	12
Google Safebrowsing (Malware)[1]	22,600
Phishtank	46,203
SURBL (Scams)	51,263
SURBL (Malware, Phishing)	7,658
URIBL (Scams)	189,047
Total Samples	667,360
Total Unique	567,784

Table 6.2: Blacklist results for URLs appearing on Twitter that were flagged as spam.

of evaluating Monarch's effectiveness at identifying spam, blacklists and email spam traps provide a suitable source of ground truth.

6.5 Implementation Details

We implement each of the four components of Monarch as independent systems operating on Amazon Web Services (AWS) cloud infrastructure. For exact specifications of the hardware used for our system, we refer readers to the AWS EC2 instance documentation [3].

6.5.1 URL Aggregation

URL aggregation and parsing is written in Scala and executes on a single EC2 Extra Large instance running Ubuntu Linux 10.04. The aggregation phase for Twitter URLs parses tweets from the Twitter Streaming API [124] and extracts URLs from the tweet text. The email spam URLs we process are provided to us post processing, and require no additional parsing. We place incoming URLs into a Kestrel queue [119] that keeps the most recent 300,000 URLs from the Twitter stream and email URLs to supply feature collection with a steady workload of fresh URLs. A full-fledged implementation of our system would require that the queue keeps every submitted URL, but for the purposes of evaluating Monarch, we only need enough URLs to scale to the system's throughput and to generate large data sets for classification.

[1]Twitter uses Google's Safebrowsing API to filter URLs appearing in tweets. URLs in our data set were either obfuscated to prevent detection, or were not present in the blacklist at the time of posting.

6.5.2 Feature Collection

As previously framed, feature collection consists of four components: a web browser, DNS resolver, IP address analysis, and a monitor to handle message passing and aggregate results. Feature collection runs in parallel on 20 EC2 High-CPU instances each running Ubuntu Linux 10.04 and executing 6 browsers, DNS resolvers, and IP analyzers each. For web browsing we rely on Firefox 4.0b4 augmented with a custom extension written in a combination of XML and JavaScript to tap into Firefox's API [90] which exposes browser-based events. We collect plugin-related events not exposed to the API by instrumenting Firefox's NPAPI [91] with hooks to interpose on all message passing between plugins and the browser. If a URL takes more than 30 seconds to load, we enforce a timeout to prevent delaying classification for other URLs. DNS resolution occurs over Linux's native `host` command, while geolocation and route lookups use the MaxMind GeoIP library [74] and Route Views [1] data respectively. A monitor written in Python aggregates the results from each of these services, generating its output as JSON text files stored in AWS S3.

6.5.3 Feature Extraction

Feature extraction is tightly coupled with the classification and training phase and does not run on separate hardware. Until the extraction phase, we store features in raw JSON format as key-value pairs. During extraction, we load the JSON content into a Scala framework, transform each into meaningful binary and real-valued features, and produce a sparse hash map stored in memory.

6.5.4 Classifier

Before training begins, we copy the raw feature data from Amazon S3 to a Hadoop Distributed File System (HDFS) [44] residing on the 50-node cluster of Amazon EC2 Double-Extra Large instances. Files in HDFS are automatically stored in shards of 128 MB, and we use this pre-existing partitioning (as required in the "Input" line of Algorithm 1). Within each shard, we randomize the order of the positive and negative example—this gives the stochastic gradient descent in Algorithm 2 (which incrementally computes its partial gradient) better convergence rates compared to the situation of processing a long, contiguous block of positive examples followed by a long, contiguous block of negative examples.

We implement Algorithms 1 and 2 using Spark, a distributed computation framework that provides fault-tolerant distributed collections [141]. Spark provides map-reduce and shuffle operations, allow us to cache the data in memory across the cluster between iterations. We take advantage of these capabilities to construct an efficient distributed learner.

The first step of the training implementation is to normalize the real-valued features. In particular, we project real values to the $[0, 1]$ interval—doing so ensures that real-valued features do not dominate binary features unduly (a common practice in classification). We perform a map-reduce operation to compute the ranges (max/min values) of each real-valued feature. Then, we broadcast the ranges to the slave nodes. The slaves read the raw JSON data from HDFS, perform feature extraction to convert JSON strings into feature hash maps, and use the ranges to complete the normalization of the data vectors. At this point, the data is ready for training.

For the "gradient" step in Algorithm 1, we distribute m tasks to the slaves, whose job is to map the m shards to partial gradients. The slaves then compute the partial gradients for their respective shards using Algorithm 2.

Because the number of features is quite large, we want to avoid running Algorithm 1's "average" and "shrink" steps entirely on the master—the amount of master memory available for storing the weight vector $\vec{w}$ constitutes a resource bottleneck we must tend to.[2] Thus, we must avoid aggregating all of the partial gradients at the master at once and find an alternate implementation that exploits the cluster's parallelism.

To achieve this, we partition and shuffle the partial gradients across the cluster so that each slave is responsible for a computing the "average" and "shrink" steps on a disjoint subset of the feature space. We split each gradient into P partitions (not to be confused with the initial m data shards). Specifically, we hash each feature to an integer key value from 1 to P. Then, we shuffle the data across the cluster to allow the slave node responsible for feature partition $p \in \{1..P\}$ to collect its partial gradients. At this point the slave responsible for partition p performs the "average" and "shrink" steps. When these computations finish, the master collects the P partitions of the weight vector (which will have a smaller memory footprint than before shrinking) and joins them into the final weight vector $\vec{w}$ for that iteration.

6.6 Accuracy, Run-time Performance, and Cost

In this section we evaluate the accuracy of our classifier and its run-time performance. Our results show that we can identify web service spam with 90.78% accuracy (0.87% false positives), with a median feature collection and classification time of 5.54 seconds. Surprisingly, we find little overlap between email and tweet spam features, requiring our classifier to learn two distinct sets of rules. We explore the underlying distinctions between email and tweet spam and observe that email is marked by short lived campaigns with quickly changing domains, while Twitter spam is relatively static during our two month-long analysis. Lastly, we examine our data set to illuminate properties of spam infrastructure including the abuse of popular web hosting and URL shorteners.

[2]In our application, the slaves are able to compute the partial gradient over their respective shards without memory exhaustion. However, if the partial gradient computation were to bottleneck the slave in the future, we would have to add a regularization step directly to Algorithm 2.

Training Ratio	Accuracy	FP	FN
1:1	94.14%	4.23%	7.50%
4:1	90.78%	0.87%	17.60%
10:1	86.61%	0.29%	26.54%

Table 6.3: Results for training on data with different non-spam to spam ratios. We adopt a 4:1 ratio for classification because of its low false positives and reasonable false negatives.

6.6.1 Classifier Performance

We train our classifier using data sampled from 1.2 million email spam URLs, 567,000 blacklisted tweet URLs, and 9 million non-spam URLs. In all experiments, we use the following parameters for training: we set the number of iterations to $I = 100$, the learning rate to $\eta = 1$, and the regularization factor to $\lambda = \frac{10\eta}{m}$ (where m is the number of data shards).

Overall Accuracy: In order to avoid mistaking benign URLs as spam, we tune our classifier to emphasize low false positives and maintain a reasonable detection rate. We use a technique from Zadrozny et al. [140] to adjust the ratio of non-spam to spam samples in training to tailor false positive rates. We consider non-spam to spam ratios of 1:1, 4:1, and 10:1, where a larger ratio indicates a stronger penalty for false positives. Using 500,000 spam and non-spam samples each, we perform 5-fold validation and randomly subsample within a fold to achieve the required training ratio (removing spam examples to increase a fold's non-spam ratio), while testing always occurs on a sample made up of equal parts spam and non-spam. To ensure that experiments over different ratios use the same amount of training data, we constrain the training set size to 400,000 examples.

Table 6.3 shows the results of our tuning. We achieve lower levels of false positives as we apply stronger penalties, but at the cost of increased false negatives. We ultimately chose a 4:1 ratio in training our classifier to achieve 0.87% false positives and 90.78% overall accuracy. This choice strikes a balance between preventing benign URLs from being blocked, but at the same time limits the amount of spam that slips past classification. For the remainder of this evaluation, we execute all of our experiments at a 4:1 ratio.

To put Monarch's false positive rate in perspective, we provide a comparison to the performance of mainstream blacklists. Previous studies have shown that blacklist false positives range between 0.5—26.9%, while the rate of false negatives ranges between 40.2—98.1% [107]. Errors result from a lack of comprehensive spam traps and from low volumes of duplicate spam across all traps [106]. These same performance flaws affect the quality of our ground truth, which may skew our estimated false positive rate.

For web services with strict requirements on false positives beyond what Monarch can guarantee, a second tier of heavier-weight verification can be employed for URLs flagged by Monarch as spam. Operation can amortize the expense of this verification by the relative infrequency of false positives. Development of such a tool remains for future work.

Feature Type	Unfiltered	Filtered	Non-spam	Spam
HTML terms	20,394,604	50,288	22,083	28,205
Source URLs	9,017,785	15,372	6,782	8,590
Page Links	5,793,359	10,659	4,884	5,775
HTTP Headers	8,850,217	9,019	3,597	5,422
DNS records	1,152,334	5,375	2,534	2,841
Redirects	2,040,576	4,240	2,097	2,143
Frame URLs	1,667,946	2,458	1,107	1,351
Initial/Final URL	1,032,125	872	409	463
Geolocation	5,022	265	116	149
AS/Routing	6,723	352	169	183
All feature types	49,960,691	98,900	43,778	55,122

Table 6.4: Breakdown of features used for classification before and after regularization.

Accuracy of Individual Components: Classification relies on a broad range of feature categories that each affect the overall accuracy of our system. A breakdown of the features used for classification before and after regularization can be found in Table 6.4. From nearly 50 million features we regularize down to 98,900 features, roughly half of which are each biased towards spam and non-spam. We do not include JavaScript pop-ups or plugin related events, as we found these on a negligible number of pages.

To understand the most influential features in our system, we train a classifier exclusively on each feature category. For this experiment, we use the data set from the previous section, applying 10-fold validation with training data at a 4:1 non-spam to spam ratio and the testing set again at a 1:1 ratio. Any feature category with an accuracy above 50% is considered better than a classifier that naively guesses the majority population. The results of per-feature category training are shown in Table 6.5. Source URLs, which is an amalgamation of every URL requested by the browser as a page is constructed, provides the best overall performance. Had our classifier relied exclusively on initial URLs or final landing page URLs, accuracy would be 7% lower and false negatives 10% higher. Surprisingly, DNS and redirect features do not perform well on their own, each achieving approximately 72% accuracy. The combination of all of these features lowers the false positive rate while maintaining high accuracy.

Accuracy Over Time: Because criminals introduce new malicious websites on a continual basis, we want to determine how often we need to retrain our classifier and how long it takes for the classifier to become out of date. To answer these questions, we evaluate the accuracy of our classifier over a 20 day period where we had continuous spam and non-spam samples. We train using two different training regimens: (1) training the classifier once over four days' worth of data, then keeping the same classification model for the rest of the experiment; (2) retraining the classifier every four days, then testing the model on the subsequent four

Feature Type	Accuracy	FP	FN
Source URLs	89.74%	1.17%	19.38%
HTTP Headers	85.37%	1.23%	28.07%
HTML Content	85.32%	1.36%	28.04%
Initial URL	84.01%	1.14%	30.88%
Final URL	83.59%	2.34%	30.53%
IP (Geo/ASN)	81.52%	2.33%	34.66%
Page Links	75.72%	15.46%	37.68%
Redirects	71.93%	0.85%	55.37%
DNS	72.40%	25.77%	29.44%
Frame URLs	60.17%	0.33%	79.45%

Table 6.5: Accuracy of classifier when trained on a single type of feature. Sources, headers, and HTML content provide the best individual performance, while frame URLs and DNS data perform the worst.

days of data. The data for each four-day window consists of 100,000 examples sampled at a 4:1 non-spam to spam ratio. We repeat this experiment four times by resampling each window's data, and take the average result.

Figure 6.3 shows the results for our time-sensitive evaluations. The error of the statically trained classifier gradually increases over time, whereas the classifier retrained daily maintains roughly constant accuracy. This indicates that in a deployment of Monarch, we will need to retrain the classifier on a continual basis. We explore the temporal nature of features that cause this behavior further in Section 6.6.3.

Training Across Input Sources: One of the primary challenges of training a classifier is obtaining labeled spam samples. Consequently, if a single labeled data set generalized to all web services, it would alleviate the problem of each web service being required to obtain their own spam samples. For instance, a great deal of time and effort could be saved if spam caught by passive email spam traps were applicable to Twitter where we currently are forced to crawl every link and retroactively blacklist spam URLs. However, spam targeting one web service is not guaranteed to be representative of spam targeting all web services. To this end we ask: how well can an email-trained classifier perform on Twitter data? How well can a Twitter-trained classifier perform on email data?

Table 6.6 displays the results of an experiment where we train our classifier on matching and mismatched data sources. We construct a 5-fold data set containing 400,000 non-spam samples and 100,000 tweet spam samples. Then, we copy the 5 folds but replace the 100,000 tweet spam samples with 100,000 email spam examples. We perform 5-fold cross validation to obtain classification rates. For a given testing fold, we test on both the tweet spam and email spam version of the fold (the non-spam samples remain the same in both version to ensure comparable results with respect to false positives).

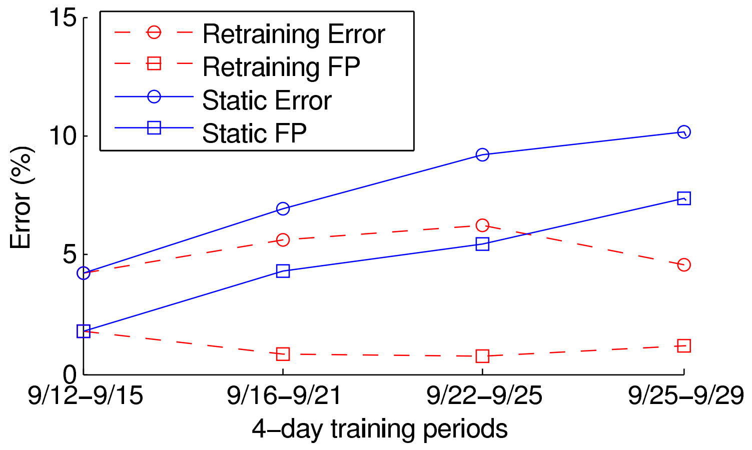


Figure 6.3: Performance of classifier over time. Regular retraining is required to guarantee the best accuracy, else error slowly increases.

Using a mixture of Twitter spam and non-spam samples, we are able to achieve 94% accuracy, but let 22% of spam tweets slip past our classifier. This same training regimen utterly fails on email, resulting in 88% of email spam going uncaught. These results are mirrored on a mixed data set of email spam and non-spam samples. We can achieve an accuracy of 98.64% with 4.47% false negatives when we train a classifier to exclusively find email spam. When we apply this same classifier to a testing set of Twitter spam, 98% of spam samples go uncaught.

These results highlight a fundamental challenge of spam filtering. Within the spam ecosystem, there are a variety of actors that each execute campaigns unique to individual web services. While Monarch's infrastructure generalizes to any web service, training data is not guaranteed to do the same. We require individual labeled data sets from each service in order to provide the best performance. A second unexpected result is the difficulty of identifying tweet spam compared to email spam. On matched training and testing sets, email spam classification achieves half the false negatives of tweet spam classification and a fifth of the false positives. We explore the underlying reason for this discrepancy in Section 6.6.3.

Context vs. Context Free Training: Because spam URLs can appear on different web services such as email, social networks, blogs, and forums, the question arises whether using context-aware features can improve classification accuracy at the cost of generalizability. To investigate this issue, we compare the error rate of classifying Twitter spam URLs (we exclude email spam) with and without account-based features. These features include account creation time, a tokenized version of tweet text, a tokenized version of an account's profile

Training Set	Testing Set	Accuracy	FP	FN
Tweet spam	Tweet spam	94.01%	1.92%	22.25%
Tweet spam	Email spam	80.78%	1.92%	88.14%
Email spam	Tweet spam	79.78%	0.55%	98.89%
Email spam	Email spam	98.64%	0.58%	4.47%

Table 6.6: Effects of training and testing on matching and mismatching data sets. Email and tweet spam are largely independent in their underlying features, resulting in low cross classification accuracy.

Training Method	Accuracy	FP	FN
With Tweet Features	94.15%	1.81%	22.11%
Without Tweet Features	94.16%	1.95%	21.38%

Table 6.7: Effects of including contextual Twitter information. Omitting account and tweet properties from classification has no statistically significant effect on accuracy (the error rates are within one standard deviation of each another).

description, the number of friends and followers an account has, the number of posts made by an account, a tokenized screen name, the account's unique Twitter ID, the application used to access Twitter (e.g., web, Twitter's API, or a third-party application), hashtags present in the tweet, and "mentions" present in the tweet. Comprehensive historical data such as the ratio of URLs to posts is unavailable.

We perform 5-fold cross validation over a data set containing 400,000 non-spam samples and 100,000 tweet spam samples. The results of the experiment are shown in Table 6.7. Even if Twitter account features are included, accuracy is statistically identical to training without these features. This contrasts with previous results that rely on account-based features to identify (fraudulent) spam accounts [8, 67, 114], but agrees with recent studies that have shown compromised accounts are the major distributors of spam [41, 107] which would render account-based features obsolete.

While this result is not guaranteed to generalize to all web services, we have demonstrated that strong performance for filtering email and Twitter spam is achievable without any requirement of revealing personally identifiable information. Omitting contextual information also holds promise for identifying web spam campaigns that cross web service boundaries without significant loss of accuracy due to disparate contextual information.

6.6.2 Run Time Performance

In addition to Monarch's accuracy, its overall performance and cost to execute are important metrics. In this section we measure the latency, throughput, and the cost of Monarch, finding

Component	Median Run Time (seconds)
URL aggregation	0.005
Feature collection	5.46
Feature extraction	0.074
Classification	0.002
Total	5.54

Table 6.8: Breakdown of the time spent processing a single URL.

a modest deployment of our system can classify URLs with a median time of 5.54 seconds and a throughput of 638,000 URLs per day, at a monthly cost of $1,600 on cloud machinery.

Latency: We measure latency as the time delta from when we receive a tweet or email URL until Monarch returns a final decision. Table 6.8 shows a breakdown of processing time for a sample of 5,000 URLs. URL aggregation takes 5 ms to parse a URL from Twitter's API format (email requires no parsing) and to enqueue the URL. Feature collection represents the largest overhead in Monarch, accounting for a median run time 5.46 seconds. Within feature collection, crawling a URL in Firefox consumes 3.13 seconds, while queries for DNS, geolocation and routing require 2.33 seconds. The majority of the processing time in both cases occurs due to network delay, not execution overhead. The remaining 70ms are spent extracting features and summing weight vectors for a classification decision.

Given that Firefox browsing incurs the largest delay, we investigate whether our instrumentation of Firefox for feature collection negatively impacts load times. We compare our instrumented Firefox against an uninstrumented copy using a sample of 5,000 URLs on a system running Fedora Core 13 machine with a four core 2.8GHz Xeon processor with 8GB of memory. We find instrumentation adds 1.02% overhead, insignificant to the median time it takes Firefox to execute all outgoing network requests which cannot be reduced. Instrumentation overhead results from interposing on browser events and message passing between the browser and monitoring service, accounting on average 110KB of log files.

Throughput: We measure the throughput of Monarch for a small deployment consisting of 20 instances on Amazon's EC2 infrastructure for crawling and feature collection. The crawling and feature extraction execute on a high-CPU medium instance that has 1.7GB of memory and two cores (5 EC2 compute units), running a 32-bit version of Ubuntu Linux 10.04. Each instance runs 6 copies of the crawling and feature collection code. We determined that the high-CPU medium instances have the lowest dollar per crawler cost, which make them the most efficient choice for crawling. The number of crawlers that each instance can support depends on the memory and CPU the machine. Using this small deployment, we can process 638,000 URLs per day.

Component	AWS Infrastructure	Monthly Cost
URL aggregation	1 Extra Large	$178
Feature collection	20 High-CPU Medium	$882
Feature extraction	—	$0
Classification	50 Double Extra Large	$527
Storage	700GB on EBS	$70
Total		$1,587

Table 6.9: Breakdown for the cost spent for Monarch infrastructure. Feature extraction runs on the same infrastructure as classification.

Training Time: For the experiments in Section **??**, we trained over data sets of 400,000 examples (80 GB in JSON format). The training time for 100 iterations of the distributed logistic regression took 45 minutes. Although we do not fully explore the effects of different data sizes or algorithm parameters on training time, we note that the following factors can increase the training time: a higher number of iterations, a larger training set (both with respect to number of examples and total number of nonzero features), a smaller regularization factor λ (which increases the amount of data communicated throughout the cluster by decreasing the sparsity of the partial gradients and weight vectors), and a smaller number of cluster machines.

For example, if we wanted to train on a larger number of examples, we could lower the number of iterations and increase the regularization factor to limit the training time. Being aware of these tradeoffs can help practitioners who want to retrain the classifier daily.

Cost: Using our deployment of Monarch as a model, we provide a breakdown of the costs associated with running Monarch on AWS for a month long period, shown in Table 6.9. Each of our components executes on EC2 spot instances that have variable prices per hour according to demand, while storage has a fixed price. URL aggregation requires a single instance to execute, costing $178 per month. For a throughput of 638,000 URLs per day, 20 machines are required to constantly crawl URLs and collect features, costing $882 per month. Besides computing, we require storage as feature data accumulates from crawlers. During a one month period, we collected 1TB worth of feature data, with a cost of $.10 per GB. However, for live execution of Monarch that excludes the requirement of log files for experimentation, we estimate only 700GB is necessary to accommodate daily re-training at a monthly cost of $70. We can discard all other data from the system after it makes a classification decision. Finally, daily classifier retraining requires a single hour of access to 50 Double-Extra Large instances, for a total of $527 per month. In summary, we estimate the costs of running a URL filtering service using Monarch with a throughput of 638,000 URLs per day to be approximately $1,600 per month. We can reduce this cost by limiting our use of cloud storage (switching from JSON to a compressed format), as well as by reducing the processing time per URL by means of better parallelism and code optimizations.

We estimate the cost of scaling Monarch to a large web service, using Twitter as an example. Twitter users send 90 million tweets per day, 25% (22.5 million) of which contain URLs [102]. After whitelisting, deploying Monarch at that scale requires a throughput of 15.3 million URLs per day. The URL aggregation component is already capable of processing incoming URLs at this capacity and requires no additional cost. The crawlers and storage scale linearly, requiring 470 instances for feature collection and approximately 15 TB of storage for a week's worth of data, costing \$20,760 and \$1,464 per month respectively. The classifier training cost remains \$527 per month so long as we use the same size of training sample. Alternatively, we could reduce the number of training iterations or increase the regularization factor λ to train on more data, but keep training within one hour. This brings the total cost for filtering 15.3 million URLs per day to \$22,751 per month.

6.6.3 Comparing Email and Tweet Spam

We compare email and tweet spam features used for classification and find little overlap between the two. Email spam consists of a diverse ecosystem of short-lived hosting infrastructure and campaigns, while Twitter is marked by longer lasting campaigns that push quite different content. We capture these distinctions by evaluating two properties: feature overlap between email and tweet spam and the persistence of features over time for both categories. Each experiment uses 900,000 samples aggregated from email spam, tweet spam, and non-spam, where we use non-spam as a baseline.

Overlap: We measure feature overlap as the log odds ratio that a feature appears in one population versus a second population. Specifically, we compute $|log(p_1 q_2 / p_2 q_1)|$, where p_i is the likelihood of appearing in population i and $q_i = 1 - p_i$. A log odds ratio of 0 indicates a feature is equally likely to be found in two populations, while an infinite ratio indicates a feature is exclusive to one population. Figure 6.4 shows the results of the log odds test (with infinite ratios omitted). Surprisingly, 90% of email and tweet features *never overlap*. The lack of correlation between the two indicates that email spammers are entirely separate actors from Twitter spammers, each pushing their own campaigns on distinct infrastructure. Consequently, the classifier must learn two separate sets of rules to identify both spam types.

Equally problematic, we find 32% of tweet spam features are shared with non-spam, highlighting the challenge of classifying Twitter spam. In particular, 41% of IP features associated with tweet spam are also found in nonspam, a result of shared redirects and hosting infrastructure. In contrast, only 16% of email spam IP features are found in non-spam, allowing a clearer distinction to be drawn between the two populations.

Persistence: We measure feature persistence as the time delta between the first and last date a feature appears in our data set, shown in Figure 6.5. Email spam is marked by much shorter lived features compared to tweet spam and non-spam samples. Notably, 77% of initial URL features appearing in email disappear after 15 days. The same is true for 60% of

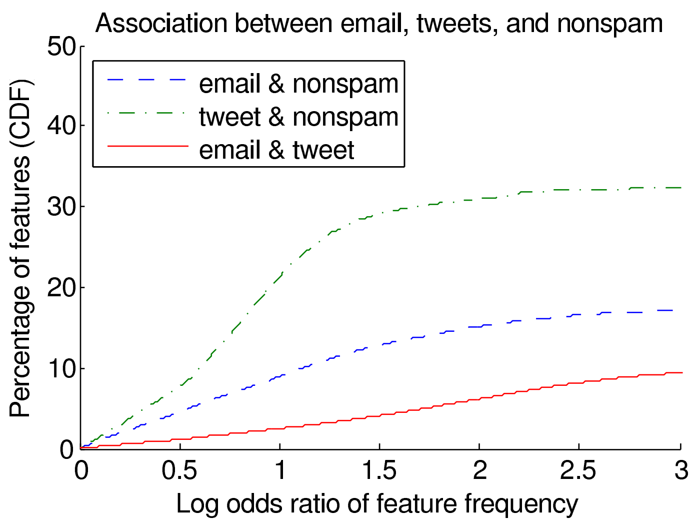


Figure 6.4: Overlap of features. Email and Twitter spam share only 10% of features in common, indicating that email spammers and Twitter spammers are entirely separate actors.

email DNS features, compared to just 30% of IP features associated with email spam hosting. Each of these results highlights the quick churn of domains used by email campaigns and the long lasting IP infrastructure controlled by email spammers. This same sophistication is unnecessary in Twitter, where there is no pressure to evade blacklists or spam filtering.

6.6.4 Spam Infrastructure

Email spam has seen much study towards understanding the infrastructure used to host spam content [4, 49]. From our feature collection, we identify two new properties of interest that help to understand spam infrastructure: redirect behavior used to lead victims to spam sites, and embedding spam content on benign pages.

Redirecting to Spam: Both Twitter and email spammers use redirects to deliver victims to spam content. This mechanism is dominated by tweet spam where 67% of spam URLs in our data set use redirects, with a median path length of 3. In contrast, only 20% of email spam URLs contain redirects, with a median path length of 2. Further distinctions between email and tweet spam behavior can be found in the abuse of public URL shorteners. Table 6.10 shows the top ten URL shortening services used for both email and tweet spam. The majority of email spam in our data set redirects through customized infrastructure hosted on arbitrary domains, while Twitter spammers readily abuse shortening services provided by *bit.ly* , Twitter, Google, and Facebook. Despite efforts by URL shorteners to block spam [51, 9], we find that widespread abuse remains prevalent.

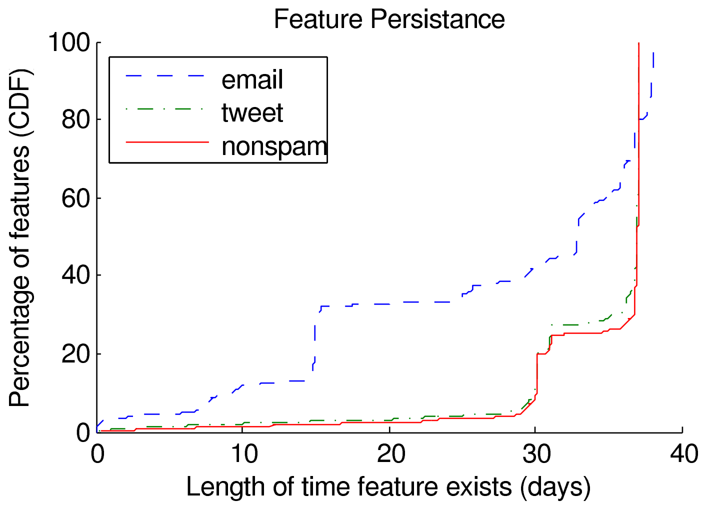


Figure 6.5: Persistence of URL features. Email spam features are shorter lived compared to tweet spam, a result of short-lived campaigns and domain churn.

Apart from the use of redirectors to mask initial URLs, we also examine domains that are commonly traversed as shortened URLs resolve to their final landing page. The top two destinations of URLs shortened by *bit.ly* are publicly available services provided by *google.com* and *blogspot.com*. Together, these two domains account for 24% of the spam first shortened by *bit.ly* . In the case of *google.com*, spam URLs embed their final landing page behind an arbitrary redirector operated by Google. This masks the final spam landing site from *bit.ly* , rendering blacklisting performed by the service obsolete. The second most common service, *blogspot.com*, is abused for free spam hosting rather than as a redirector. Each blog contains scam advertisements and other solicitations. By relying on Blogspot, spammers can evade domain-based blacklists that lack the necessary precision to block spam hosted alongside benign content.

Each of these are prime examples of web services currently being abused by spammers and serve as a strong motivation for the need of a system like Monarch.

Page Content: Another phenomenon we frequently observe in Twitter spam is the blacklisting of content within a page. For the majority of sites, this is a web advertisement from a questionable source. We have observed popular news sites with non-spam content displaying ads that cause a variety of spam popups, sounds, and video to play. Table 6.11 shows a breakdown of the locations containing blacklisted URLs specifically for Twitter. The column labeled exclusive indicates the percent of URLs that can be blacklisted exclusively based on a URL in that location. For example, 0.05% of Twitter spam can be blacklisted using only an initial URL posted to the site. Since the category source URLs is a superset of all other URLs, 100% of pages can be blacklisted; however, looking exclusively at URLs which are not

Domain	Email spam	Twitter spam
bit.ly	1%	41%
t.co	0%	4%
tinyurl.com	3%	4%
ow.ly	0%	4%
goo.gl	0%	3%
su.pr	0%	3%
fb.me	0%	2%
dlvr.it	0%	2%
os7.biz	0%	1%
is.gd	0%	1%

Table 6.10: Top 10 URL shortening services abused by spammers.

Feature Category	% Blacklisted	% Exclusive
Initial URL	16.60%	0.05%
Final URL	23.33%	2.62%
Top-level Window Redirect URL	34.25%	4.41%
Content Redirect URL	3.99%	1.35%
Frame Content URL	14.85%	6.87%
Link URLs	28.28%	7.03%
Source URLs	100%	42.51%

Table 6.11: Breakdown of the locations of blacklisted URLs. We mark a page as spam if it makes any outgoing request to a blacklisted URL.

part of other categories, we find that 42.51% of source URLs lead to blacklisting. This indicates a page included an image, stylesheet, plugin, script, or dynamically retrieved content via JavaScript or a plugin that was blacklisted. These scenarios highlight the requirement of analyzing *all* of a webpages content to not overlook spam with dynamic page behavior or mash-up content that includes known spam domains.

6.7 Evading Browser-based Spam Detection

In this section we discuss potential evasive attacks against Monarch that result from running a centralized service. While we can train our system to identify spam and have shown the features we extract are applicable over time, classification exists in an adversarial environment. Attackers can tune features to fall below the spam classification threshold, modify content after classification, and block our crawler. We do not propose solutions to these attacks; instead, we leave to future work an in depth study of each attack and potential solutions.

Feature Evasion: When Monarch provides a web service with a classification decision, it also provides attackers with immediate feedback for whether their URLs are blocked. An attacker can use this feedback to tune URLs and content in an attempt to evade spam classification, as discussed in previous studies [20, 70, 7], but not without consequences and limitations. The simplest changes an attacker can make are modifications to page content: HTML, links, and plugins. Known spam terms can be transformed into linguistically similar, but lexically distinct permutations to avoid detection, while links and plugins can be modified to imitate non-spam pages. Page behavior poses a more difficult challenge; by removing pop-up windows and alert prompts, a spammer potentially reduces the effectiveness of eliciting a response from victims. Finally, hosting infrastructure, redirects, and domains, while mutable, require a monetary expense for dynamism. We leave evaluating how susceptible our classification system is to evasion to future work, but note that email spam classification and intrusion prevention systems both exist in adversarial environments and maintain wide-spread adoption.

Time-based Evasion: In Monarch's current implementation, feature collection occurs at the time a URL is submitted to our system; URLs are not re-crawled over time unless they are resubmitted. This raises the potential for an attacker to change either page content or redirect to new content after a URL has been classified. For this attack to succeed, a URL's redirects and hosting infrastructure must appear benign during classification and allow subsequent modification. An attacker that simply masks his final landing page, but re-uses known hostile redirect infrastructure may still be identified by the classifier. Furthermore, static shorteners such as *bit.ly* cannot be used because the landing page cannot be changed after shortening. To circumvent both of these limitations, an attacker can rely on mutable content hosted on public infrastructure typically associated with non-spam pages, such as Blogspot, LiveJournal, and free web hosting. In this scenario, an attacker's blog contains non-spam content during classification and is subsequently modified to include spam content or a JavaScript redirect to a new hostile landing page.

Crawler Evasion: Rather than an attacker modifying content to evade classification, an adversary can alter HTTP and DNS behavior to prevent our crawler from ever reaching spam pages. Potential attacks include relying on browser user-agent detection or other forms of browser fingerprinting [24] to forward our crawler to non-hostile content and regular users to a hostile copies. Alternatively, the IP addresses of Monarch's crawlers can be learned by an attacker repeatedly posting URLs to our service and tracking the IPs of visitors. A list of crawler IP addresses can then be distributed as a blacklist, with attackers either blocking access or redirecting our crawlers to non-spam content.

6.8 Summary of Results

Monarch is a real-time system for filtering scam, phishing, and malware URLs as they are submitted to web services. We showed that while Monarch's architecture generalizes to many web services being targeted by URL spam, accurate classification hinges on having an intimate understanding of the spam campaigns abusing a service. In particular, we showed that email spam provides little insight into the properties of Twitter spammers, while the reverse is also true. We explored the distinctions between email and Twitter spam, including the overlap of spam features, the persistence of features over time, and the abuse of generic redirectors and public web hosting. We have demonstrated that a modest deployment of Monarch on cloud infrastructure can achieve a throughput of 638,000 URLs per day with an overall accuracy of 91% with 0.87% false positives. Each component of Monarch readily scales to the requirements of large web services. We estimated it would cost $22,751 a month to run a deployment of Monarch capable of processing 15 million URLs per day.

Chapter 7

Disrupting the Underground Account Marketplace

7.1 Introduction

As web services such as Twitter, Facebook, Google, and Yahoo now dominate the daily activities of Internet users [2], cyber criminals have adapted their monetization strategies to engage users within these walled gardens. This has lead to a proliferation of *fraudulent accounts*—automatically generated credentials used to disseminate scams, phishing, and malware. Our own work in Chapter 4 estimates at least 3% of active Twitter accounts are fraudulent. Facebook estimates its own fraudulent account population at 1.5% of its active user base [58], and the problem extends to major web services beyond just social networks [62].

The complexities required to circumvent registration barriers such as CAPTCHAs, email confirmation, and IP blacklists have lead to the emergence of an underground market that specializes in selling fraudulent accounts in bulk. *Account merchants* operating in this space brazenly advertise: a simple search query for "`buy twitter accounts`" yields a multitude of offers for fraudulent Twitter credentials with prices ranging from $10—200 per thousand. Once purchased, accounts serve as stepping stones to more profitable spam enterprises that degrade the quality of web services, such as pharmaceutical spam [75] or fake anti-virus campaigns [111].

In this chapter we describe our investigation of the underground market profiting from Twitter credentials to study how it operates, the impact the market has on Twitter spam levels, and exactly how merchants circumvent automated registration barriers.[1] In total, we identified and monitored 27 account merchants that advertise via web storefronts, blackhat forums, and freelance labor sites. With the express permission of Twitter, we conducted a longitudinal study of these merchants and purchased a total of 121,027 fraudulent Twitter ac-

[1]Our study is limited to Twitter, as we were unable to acquire permission to conduct our research from other companies we saw being abused.

counts on a bi-weekly basis over ten months from June, 2012—April, 2013. Throughout this process, we tracked account prices, availability, and fraud in the marketplace. Our findings show that merchants thoroughly understand Twitter's existing defenses against automated registration, and as a result can generate thousands of accounts with little disruption in availability or instability in pricing.

In order to fulfill orders for fraudulent Twitter accounts, we find that merchants rely on CAPTCHA solving services; fraudulent email credentials from Hotmail, Yahoo, and mail.ru; and tens of thousands of hosts located around the globe to provide a diverse pool of IP addresses to evade blacklisting and throttling. In turn, merchants stockpile accounts months in advance of their sale, where "pre-aged" accounts have become a selling point in the underground market. We identify which registration barriers effectively increase the price of accounts and summarize our observations into a set of recommendations for how web services can improve existing automation barriers to increase the cost of fraudulent credentials.

Finally, to estimate the overall impact the underground market has on Twitter spam we leveraged our understanding of how merchants abuse the registration process in order to develop a classifier that retroactively detects fraudulent accounts. We applied our classifier to all accounts registered on Twitter in the last 10 months and identify several million suspected fraudulent accounts generated and sold via the underground market. During active months, the 27 merchants we monitor appeared responsible for registering 10—20% of all accounts later flagged by Twitter as spam. For their efforts, the merchants generated an estimated total revenue between \$127,000—\$459,000 from the sale of accounts.

With Twitter's cooperation, we disable 95% of all fraudulent accounts registered by the merchants we track, including those previously sold but not yet suspended for spamming. Throughout the suspension process, we simultaneously monitor the underground market for any fallout. While we do not observe an appreciable increase in pricing or delay in merchants delivering new accounts, we find 90% of all purchased accounts immediately after our actioning are suspended on arrival. We are now actively working with Twitter to integrate our defense into their real-time detection framework to help prevent abusive signups.

In summary, we frame our contributions as follows:

- We perform a 10 month longitudinal study of 27 merchants profiting from the sale of Twitter accounts.
- We develop a classifier based on registration signals that detects several million fraudulent accounts that merchants sold to generate \$127,000—\$459,000 in revenue.
- We investigate the impact that the underground market has on Twitter spam levels and find 10—20% all spam accounts originate from the merchants we study.
- We investigate the failures of existing automated registration barriers and provide a set of recommendations to increase the cost of generating fraudulent accounts.

7.2 Legal and Ethical Guidelines

To minimize the risk posed to Twitter or its users by our investigation of the account market, we follow a set of policies set down by our institutions and Twitter, reproduced here to serve as a note of caution to other researchers conducting similar research.

Twitter & Users: Some of the account merchants we deal with work in an on-demand fashion, where purchases we place directly result in abusive registrations on Twitter (e.g., harm) in violation of the site's Terms of Services. Even purchases from existing stockpiles might be misconstrued as galvanizing further abuse of Twitter. As such, we directly contacted Twitter to receive permission to conduct our study. In the process, we determined that any interactions with the underground market should not result in harm to Twitter's user base. In particular, accounts we purchased should *never* be used to tweet or form relationships while under our control. Furthermore, we take no special action to guarantee our accounts are not *suspended* (e.g disabled) by Twitter; our goal is to observe the natural registration process, not to interact with or impede Twitter's service in any way.

Account Merchants: We do not interact with merchants anymore than necessary to perform transactions. To this end, we only purchased from merchants that advertise their goods publicly and never contact merchants outside the web sites or forums they provide to conduct a sale (or to request replacement accounts in the event of a bad batch). Our goal is not to study the merchants themselves or to collect personal information on them; only to analyze the algorithms they use to generate accounts.

Sensitive User Data: Personal data logged by Twitter is subject to a multitude of controls, while user names and passwords sold by merchants also carry controls to prevent fraud, abuse, and unauthorized access. First, we *never* log into accounts; instead, we rely on Twitter to verify the authenticity of credentials we purchase. Furthermore, all personal data such as IP addresses or activities tied to an account are never accessed outside of Twitter's infrastructure, requiring researchers involved in this study to work on site at Twitter and to follow all relevant Twitter security practices. This also serves to remove any risk in the event an account is *compromised* rather than registered by an account merchant, as no personal data ever leaves Twitter. To our knowledge, we never obtained credentials for compromised accounts.

7.3 Infiltrating the Marketplace for Twitter Accounts

We infiltrate the market for Twitter accounts to understand its organization, pricing structure, and the availability of accounts over time. Through the course of our study, we identify 27 account merchants (or sellers) whom we purchase from on a bi-weekly basis from June, 2012 —April, 2013. We determine that merchants can provide thousands of accounts within 24 hours at a price of \$0.10—\$0.02 per account.

7.3.1 Identifying Merchants

With no central operation of the underground market, we resort to investigating common haunts: advertisements via search engines, blackhat forums such as *blackhatworld.com*, and freelance labor pages including Fiverr and Freelancer [87, 88]. In total, we identify a disparate group of 27 merchants. Of these, 10 operate their own websites and allow purchases via automated forms, 5 solicit via blackhat forums, and 12 advertise via freelance sites that take a cut from sales. Advertisements for Twitter accounts range in offerings from credentials for accounts with no profile or picture, to “pre-aged” accounts[2] that are months old with unique biographies and profile data. Merchants even offer 48 hours of support, during which miscreants can request replacements for accounts that are dysfunctional. We provide a detailed breakdown of the merchants we identify and their source of solicitation in Table 7.1. We make no claim our search for merchants is exhaustive; nevertheless, the sellers we identify provide an insightful cross-section of the varying levels of sophistication required to circumvent automated account registration barriers, outlined in detail in Section 7.4.

7.3.2 Purchasing from Merchants

Once we identify a merchant, we place an initial test purchase to determine the authenticity of the accounts being sold. If genuine, we then determine whether to repeatedly purchase from the merchant based on the quality of accounts provided (discussed in Section 7.4) and the overall impact the seller has on Twitter spam (discussed in Section 7.6). As such, our purchasing is an iterative process where each new set of accounts improves our understanding of the market and subsequently directs our investigation.

Once we vet a merchant, we conduct purchases on a bi-weekly basis beginning in June, 2012 (at the earliest) up to the time of our analysis in April, 2013, detailed in Table 7.1. We note that purchasing at regular intervals is not always feasible due to logistical issues such as merchants delaying delivery or failing to respond to requests for accounts. In summary, we place 144 orders (140 of which merchants successfully respond to and fulfill) for a total of 120,019 accounts. Purchases typically consist of a bulk order for 1,000 accounts, though sellers on Fiverr operate in far less volume.

Throughout this process, we protect our identity from merchants by using a number of email and Skype pseudonyms. We conduct payments through multiple identities tied to PayPal, WebMoney, and pre-paid credit cards. Finally, we access all web content on a virtual machine through a network proxy.

[2]Pre-aged accounts allow miscreants to evade heuristics that disable newly minted accounts based upon weak, early signs of misbehavior. In contrast, much more robust signals of maleficence must be satisfied to action older accounts to limit the impact on legitimate users.

Merchant	Period	#	Accts	Price
alexissmalley†	06/12—03/13	14	13,000	$4
naveedakhtar†	01/13—03/13	4	2,044	$5
truepals†	02/13—03/13	3	820	$8
victoryservices†	06/12—03/13	15	15,819	$6
webmentors2009†	10/12—03/13	9	9,006	$3—4
buuman$^{\text{II}}$	10/12—10/12	1	75	$7
danyelgallu$^{\text{II}}$	10/12—10/12	1	74	$7
denial93$^{\text{II}}$	10/12—10/12	1	255	$20
formefor$^{\text{II}}$	09/12—11/12	3	408	$2—10
ghetumarian$^{\text{II}}$	09/12—10/12	3	320	$4—5
jackhack08$^{\text{II}}$	09/12—09/12	2	755	$1
kathlyn$^{\text{II}}$	10/12—10/12	1	74	$7
smokinbluelady$^{\text{II}}$	08/12—08/12	1	275	$2
twitfollowers$^{\text{II}}$	10/12—10/12	1	80	$6
twitter007$^{\text{II}}$	10/12—10/12	1	75	$7
kamalkishover$^{\diamond}$	06/12—03/13	14	12,094	$4—7
shivnagsudhakar$^{\diamond}$	06/12—06/12	1	1,002	$4
accs.biz‡	05/12—03/13	15	17,984	$2—3
buyaccountsnow.com‡	06/12—11/12	8	7,999	$5—8
buyaccs.com‡	06/12—03/13	14	13,794	$1—3
buytwitteraccounts.biz‡	09/12—10/12	3	2,875	$5
buytwitteraccounts.info‡	10/12—03/13	9	9,200	$3—4
dataentryassistant.com‡	10/12—03/13	9	5,498	$10
getbulkaccounts.com‡	09/12—09/12	1	1,000	$2
quickaccounts.bigcartel.com‡	11/12—11/12	2	1,501	$3
spamvilla.com‡	06/12—10/12	3	2,992	$4
xlinternetmarketing.com‡	10/12—10/12	1	1,000	$7
Total	05/12—03/13	140	120,019	$1—20

Table 7.1: List of the merchants we track, the months monitored, total purchases performed (#), accounts purchased, and the price per 100 accounts. Source of solicitations include blackhat forums†, Fiverr$^{\text{II}}$, and Freelancer$^{\diamond}$ and web storefronts‡.

7.3.3 Account Pricing & Availability

Prices through the course of our analysis range from $0.01 to $0.20 per Twitter account, with a median cost of $0.04 for all merchants. Despite the large overall span, prices charged by individual merchants remain roughly stable. Table 7.1 shows the variation in prices for six merchants we tracked over the longest period of time. Price hikes are a rare occurrence and no increase is more than $0.03 per account. So long as miscreants have money on hand,

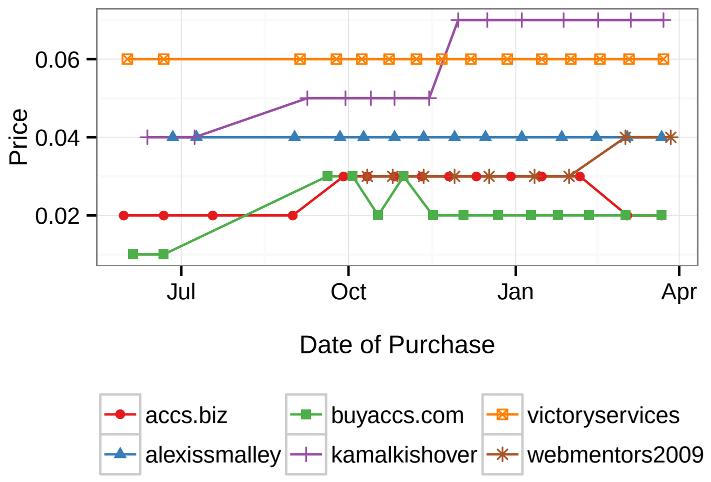


Figure 7.1: Variation in prices over time for six merchants we track over the longest period of time.

availability of accounts is a non-issue. Of the orders we placed, merchants fulfilled 70% in a day and 90% within 3 days. We believe the stable pricing and ready availability of fraudulent accounts is a direct result of minimal adversarial pressures on account merchants, a hypothesis we explore further in Section 7.4.

7.3.4 Other Credentials For Sale

Our permission to purchase accounts is limited to Twitter credentials, but many of the merchants we interact with also sell accounts for Facebook, Google, Hotmail, and Yahoo. We compare prices between web services, but note that as we cannot vet non-Twitter credentials, some prices may represent scams.

Facebook: Prices for Facebook accounts range from \$0.45—1.50 per *phone verified account* (PVA) and \$0.10 for non-PVA accounts. Phone verification requires that miscreants tie a SIM card to a newly minted Facebook account and verify the receipt of a text message, the complexities of which vastly increase the price of an account.[3] For those sellers that advertise their registration process, SIM cards originate from Estonia or Ukraine.

Google: Prices for Google PVA accounts range from \$0.03—0.50 per account.

[3] Advertisements that we encountered for phone verification services ranged in price from \$.10—\$.15 per verification for bulk orders of 100,000 verifications and \$.25 per verification for smaller orders.

Hotmail: Prices for Hotmail accounts cost $0.004—0.03 per account, a steep reduction over social networking or PVA credentials. We see similar prices for a multitude of web mail providers, indicating that email accounts are in demand and cheaper to create.

Yahoo: Yahoo accounts, like Hotmail, are widely available, with prices ranging from $0.006—0.015 per account.

7.3.5 Merchant Fraud

Operating in the underground market is not without risk of fraud and dishonesty on the part of account merchants. For instance, eight of the merchants we contacted attempted to sell us a total of 3,317 duplicate accounts. One merchant even schemed to resell us the same 1,000 accounts three times. For those merchants willing to honor their "48 hours of support", we requested replacement accounts for duplicates, bringing our account total up to 121,027 unique credentials.

Apart from duplicate credentials, some merchants were quick to resell accounts we purchased to third parties. In order to detect resales, we coordinate with Twitter to monitor all successful logins to accounts we purchase after they come under our control. We denote these accounts *reaccessed*. We repeat this same process to detect new tweets or the formation of relationships. Such behaviors should only occur when an account changes hands to a spammer, so we denote these accounts as *resold*. Such surreptitious behavior is possible because we make a decision not to change the passwords of accounts we purchase.

Table 7.2 shows the fraction of purchased accounts per seller that merchants reaccessed and resold. A total of 10% of accounts in our dataset were logged into (either by the seller or a third party; it is not possible to distinguish the two) within a median of 3 days from our purchase. We find that 6% of all accounts go on to be resold in a median of 5 days from our purchase. This serves to highlight that some merchants are by no means shy about scamming potential customers.

7.4 Characterizing How Criminals Automate Fraudulent Account Creation

Account merchants readily evade existing abuse safeguards to register thousands of accounts on a recurring basis. To understand these failings, we delve into the tools and techniques required to operate in the account marketplace. We find that merchants leverage thousands of compromised hosts, CAPTCHA solvers, and access to fraudulent email accounts. We identify what registration barriers increase the price of accounts and summarize our observations into a set of recommendations for how web services can improve existing automation barriers to increase the cost of fraudulent credentials in the future.

Merchant	Reaccessed	Resold
getbulkaccounts.com	100%	100%
formefor	100%	99%
denial93	100%	97%
shivnagsudhakar	98%	98%
quickaccounts.bigcartel.com	67%	64%
buytwitteraccounts.info	39%	31%
ghetumarian	30%	28%
buytwitteraccounts.biz	20%	18%
jackhack08	12%	11%
buyaccountsnow.com	10%	1%
kamalkishover	8%	0%
buyaccs.com	7%	4%
alexissmalley	6%	0%
victoryservices	3%	2%
Total	10%	6%

Table 7.2: List of dishonest merchants that reaccessed and resold credentials we purchased to other parties.

7.4.1 Dataset Summary

To carry out our analysis, we combine intelligence gathered from the underground market with private data provided through a collaboration with Twitter. Due to the sensitivity of this data, we strictly adhere to a data policy set down by Twitter, documented in Appendix 7.2. In total, we have the credentials for 121,027 purchased accounts, each of which we annotate with the seller and source of solicitation. Furthermore, we obtain access to each account's associated email address; login history going back one year including IP addresses and timestamps; signup information including the IP and user agent used to register the account; the history of each account's activities including tweeting or the formation of social connections, if any; and finally whether Twitter has flagged the account as spam (independent of our analysis).

7.4.2 Circumventing IP Defenses

Unique IP addresses are a fundamental resource for registering accounts in bulk. Without a diverse IP pool, fraudulent accounts would fall easy prey to network-based blacklisting and throttling [82, 143, 50]. Our analysis leads us to believe that account merchants either own or rent access to thousands of compromised hosts to evade IP defenses.

Registration Origin	Unique IPs	Popularity
India	6,029	8.50%
Ukraine	6,671	7.23%
Turkey	5,984	5.93%
Thailand	5,836	5.40%
Mexico	4,547	4.61%
Viet Nam	4,470	4.20%
Indonesia	4,014	4.10%
Pakistan	4,476	4.05%
Japan	3,185	3.73%
Belarus	3,901	3.72%
Other	46,850	48.52%

Table 7.3: Top 10 most popular geolocations of IP addresses used to register fraudulent accounts.

IP Address Diversity & Geolocation: As a whole, miscreants registered 79% of the accounts we purchase from unique IP addresses located across the globe. No single subnet captures the majority of abused IPs; the top ten /24 subnets account for only 3% of signup IPs, while the top ten /16 subnets account for only 8% of registrations. We provide a breakdown of geolocations tied to addresses under the control of merchants in Table 7.3. India is the most popular origin of registration, accounting for 8.5% of all fraudulent accounts in our dataset. Other 'low-quality' IP addresses (e.g., inexpensive hosts from the perspective of the underground market [11]) follow in popularity. In summary, registrations come from 164 countries, the majority of which serve as the origin of fewer than 1% of accounts in our dataset. However, in aggregate, these small contributors account for 48.5% of all registered accounts.

Merchants that advertise on blackhat forums or operate their own web storefronts have the most resources at their disposal, registering all but 15% of their accounts via unique IPs from hundreds of countries. Conversely, merchants operating on Fiverr and Freelancer tend to operate solely out of the United States or India and reuse IPs for at least 30% of the accounts they register.

Long-term IP Abuse: To understand the long-term abuse of IP addresses, we analyze data provided by Twitter that includes *all* registered accounts (not just our purchases) from June, 2012—April, 2013. From this, we select a random sample of 100,000 unique IPs belonging to accounts that Twitter has disabled for spamming (e.g., suspended) and an equally sized sample of IPs used to register legitimate Twitter accounts. We add a third category to our sample that includes all the unique IP addresses used by merchants to register the accounts we purchased. For each of these IPs, we calculate the total number of Twitter accounts registered from the same IP.

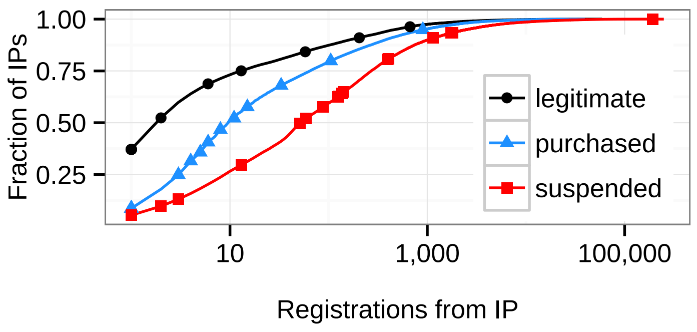


Figure 7.2: CDF of registrations per IP tied to purchased accounts, legitimate accounts, and suspended (spam) accounts.

A CDF of our results, shown in Figure 7.2, indicates merchants use the IP addresses under their control to register an abnormal number of accounts. Furthermore, the merchants we track are more cautious than other Twitter spammers who register a larger volume of accounts from a single IP address, making the merchants harder to detect. In total, merchants use 50% of the IP addresses under their control to register fewer than 10 accounts, compared to 73% of IPs tied to legitimate users and only 26% for other spammers. We note that the small fraction of legitimate IP addresses used to register thousands of accounts likely belong to mobile providers or other middleboxes.

IP Churn & Pool Size: In order to sustain demand for new accounts without overextending the abuse of a single IP address, merchants obtain access to tens of thousands of IP addresses that change over time. Figure 7.3 shows the fraction of accounts we purchase that appear from a unique IP address[4] as a function of time. We restrict our analysis to the six merchants we track over the longest period. Despite successive purchases of 1,000 accounts, all but one seller maintains IP uniqueness above roughly 80% of registered accounts, indicating that the IPs available to merchants change over time.

We calculate the number of IP addresses under each merchant's control by treating IP reuse as a *closed capture-recapture* problem. Closed capture-recapture measurements—used to estimate an unknown population size— require (1) the availability of independent samples and (2) that the population size under study remains fixed. To begin, we assume each purchase we make is an independent sample of the IP addresses under a merchant's control, satisfying the first requirement. The second requirement is more restrictive. If we assume that merchants use IP addresses tied to compromised hosts, then there is an inherent instability in the population size of IPs due to hosts becoming uninfected, new hosts becoming infected, and ISPs reallocating dynamic IPs. As such, comparisons over

[4]We calculate uniqueness over the IP addresses in our dataset, not over all IPs used to register accounts on Twitter.

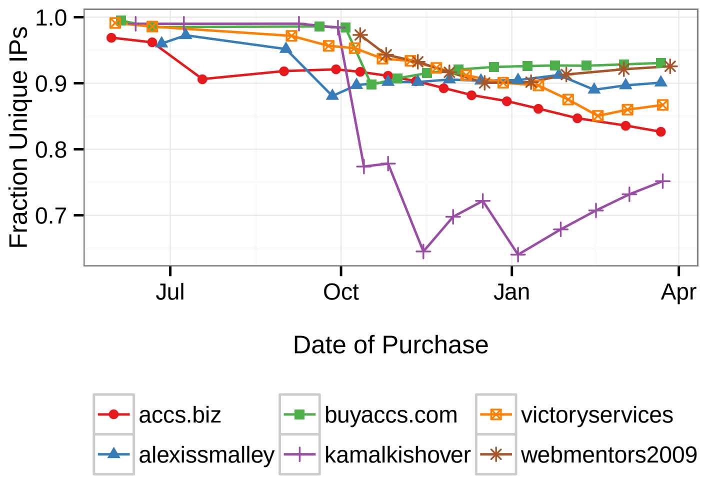


Figure 7.3: Availability of unique IPs over time for the six merchants we track over the longest period. All but one seller we repeatedly purchase from are able to acquire new IP address to register accounts from over time.

long periods are not possible. Nevertheless, if we restrict our analysis to batches of accounts from a single seller that were all registered within 24 hours, we can minimize the imprecision introduced by IP churn.

To this end, we select clusters of over 300 accounts registered by merchants within a 24 hour window. We split each cluster in half by time, with the first half m acting as the set of marked IPs and the second set c as the captured IPs, where there are r overlapping, or recaptured, IPs between both sets. We can then estimate the entire population size $\hat{N}$ (e.g., the number of unique IPs available to a merchant) according to the Chapman-Petersen method [104]:

$$\hat{N} = \frac{(m+1)(c+1)}{(r+1)} - 1$$

And standard error according to:

$$SE = \sqrt{\frac{\hat{N}^2(c-r)}{(c+1)(r+2)}}$$

For 95% confidence intervals, we calculate the error of $\hat{N}$ as $\pm 1.96 \times SE$. We detail our results in Table 7.4. We find that sellers like *accs.biz* and *victoryservices* have tens of thousands of IPs at their disposal on any given day, while even the smallest web storefront merchants have thousands of IPs on hand to avoid network-based blacklisting and throttling.

Merchant	$\hat{N}$ Estimate	± Error
accs.biz	21,798	4,783
victoryservices	17,029	2,264
dataentryassistant.com	16,887	4,508
alexissmalley	16,568	3,749
webmentors2009	10,019	2,052
buyaccs.com	9,770	3,344
buytwitteraccounts.info	6,082	1,661
buyaccountsnow.com	5,438	1,843
spamvilla.com	4,646	1,337
kamalkishover	4,416	1,170

Table 7.4: Top 10 merchants with the largest estimated pool of IP addresses under their control on a single day.

7.4.3 CAPTCHAs & Email Confirmation

Web services frequently inhibit automated account creation by requiring new users to solve a CAPTCHA or confirm an email address. Unsurprisingly, we find neither of these barriers are insurmountable, but they *do* impact the pricing and rate of generation of accounts, warranting their continued use.

Email Confirmation: All but 5 of the merchants we purchase from readily comply with requirements to confirm email addresses through the receipt of a secret token. In total, merchants email confirm 77% of accounts we acquire, all of which they seeded with a unique email. The failure of email confirmation as a barrier directly stems from pervasive account abuse tied to web mail providers. Table 7.5 details a list of the email services frequently tied to fraudulent Twitter accounts. Merchants abuse Hotmail addresses to confirm 60% of Twitter accounts, followed in popularity by Yahoo and mail.ru. This highlights the interconnected nature of account abuse, where credentials from one service can serve as keys to abusing yet another.

While the ability of merchants to verify email addresses may raise questions of the processes validity, we find that email confirmation positively impacts the price of accounts. Anecdotally, Hotmail and Yahoo accounts are available on *blackhatworld.com* for $6 per thousand, while Twitter accounts from the same forum are $40 per thousand. This is also true of web storefront such as *buyaccs.com* where mail.ru and Hotmail accounts are $5 per thousand, compared to $20 per thousand for Twitter accounts. Within our own dataset, we find that Twitter accounts purchased without email confirmation cost on average $30 per thousand compared to $47 per thousand for accounts with a confirmed email address. This difference likely includes the base cost of an email address and any related overhead due to the complexity of responding to a confirmation email.

Email Provider	Accounts	Popularity
hotmail.com	64,050	60.08%
yahoo.com	12,339	11.57%
mail.ru	12,189	11.43%
gmail.com	2,013	1.89%
nokiamail.com	996	0.93%
Other	2,157	0.14%

Table 7.5: Top 5 email providers used to confirm fraudulent Twitter accounts.

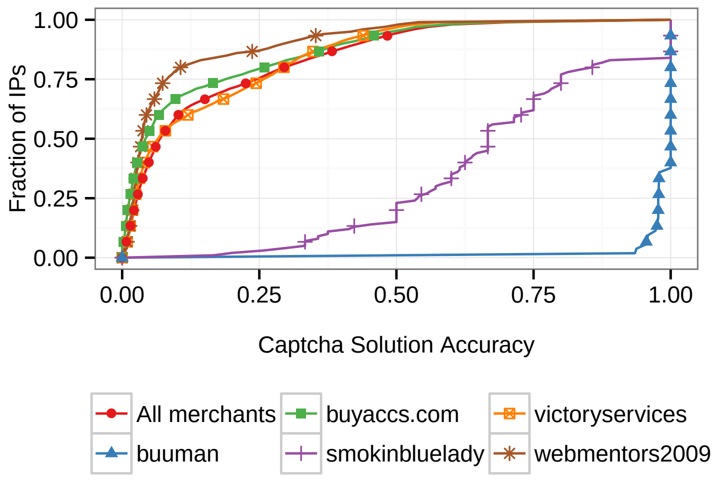


Figure 7.4: CAPTCHA solution rates per each IP address abused by a variety of merchants as well as the rates for all merchants combined.

CAPTCHA Solving: Twitter throttles multiple registrations originating from a single IP address by requiring a CAPTCHA solution. Merchants solved a CAPTCHA for 35% of the accounts we purchase; the remaining accounts were registered from fresh IPs that did not trigger throttling. While there are a variety of CAPTCHA solving services available in the underground market [86], none are free and thus requiring a CAPTCHA slightly increases the cost of creating fraudulent accounts.

A second aspect of CAPTCHAs is the success rate of automated or human solvers. By virtue of only buying successfully registered accounts, we cannot exactly measure CAPTCHA failure rates (unless account sellers fail and re-try a CAPTCHA during the same registration session, something we find rare in practice). However, we can examine registration attempts that occur from the same IPs as the accounts we purchase to estimate the rate of failure. To carry out this analysis, we examine all registrations within the previous year, calculating the fraction of registrations that fail due to incorrect CAPTCHA solutions per IP address.

We show a CDF of CAPTCHA solution rates for a sample of merchants in Figure 7.4. The median CAPTCHA solution rate for all sellers is 7%, well below estimates for automated CAPTCHA solving software of 18—30% [86], a discrepancy we currently have no explanation for. For two of the Fiverr sellers, *buuman* and *smokinbluelady*, the median CAPTCHA solution rate per IP is 100% and 67% respectively, which would indicate a human solver. In total, 92% of all throttled registration attempts from merchants fail. Despite this fact, account sellers are still able to register thousands accounts over the course of time, simply playing a game of odds.

7.4.4 Stockpiling & Suspension

Without effective defenses against fraudulent account registration, merchants are free to stockpile accounts and sell them at a whim. For many solicitations, merchants consider "pre-aged" accounts a selling point, not a detraction. To highlight this problem, we examine the failure of at-abuse time metrics for detecting dormant accounts and the resulting account stockpiles that occur.

Account Suspension: Twitter suspends (e.g., disables) spam accounts due to at-abuse time metrics such as sending spam URLs or generating too many relationships, as outlined in Twitter's rules [123]. In our case, we are interested in whether fraudulent accounts that do *not* perform visible spam actions (e.g., are dormant) nevertheless become suspended. While for miscreants this should ideally be impossible, there are multiple avenues for guilt by association, such as clustering accounts based on registration IP addresses or other features. As such, when Twitter suspends a large volume of active fraudulent accounts for spamming, it is possible for Twitter to catch dormant accounts in the same net.

Of the dormant accounts we purchase, only 8% are eventually detected and suspended. We exclude accounts that were resold and used to send spam (outlined in Section 7.3.5) from this metric in order to not skew our results. Of the merchants we track, Fiverr sellers take the least caution in registering unlinkable accounts, resulting in 57% of our purchases becoming suspended by the time of our analysis. In contrast, web storefronts leverage the vast resources at their disposal to create unlinkable accounts, where only 5% of our purchased accounts are eventually detected as fraudulent. These poor detection rates highlight the limitation of at-abuse time metrics against automated account registration. Without more sophisticated at-registration abuse signals, merchants are free to create thousands of accounts with minimal risk of Twitter suspending back stock.

Account Aging & Stockpiling: We examine the age of accounts, measured as the time between their registration and subsequent date of purchase, and find that accounts are commonly stockpiled for a median of 31 days. While most merchants deal exclusively in back stock, some merchants operate in an on-demand fashion. At the far end of this spectrum is a merchant *spamvilla.com* that sold us accounts registered a median of 323 days ago—nearly a year in advance of our purchase. In contrast, webstores such as *buyaccs.com* and Fiverr merchants including *smokinbluelady* sell accounts less than a day old. Even though these merchants operate purely on-demand, they are still able to fulfill large requests in short order (within a day in our experience). Both modes of operation illustrate the ease that merchants circumvent existing defenses and the need for at-registration time abuse detection.

7.4.5 Recommendations

Web services that rely on automation barriers must strike a tenuous balance between promoting user growth and preventing the proliferation of fraudulent accounts and spam behavior. We summarize our findings in this section with a number of potential improvements to existing barriers that should not impede legitimate users. While we draw many of our observations from the Twitter account abuse problem, we believe our recommendations should generalize across web services.

Email Confirmation: While account merchants have cheap, disposable emails on hand to perform email confirmation, confirmation helps to increase the cost of fraudulent accounts. In the case of Twitter, email confirmation raises the cost of accounts by 56%. Furthermore, in the absence of clear abuse signals, services can use email *reconfirmation* as a *soft action* against automation, similar to requiring a CAPTCHA before sending an email or tweet. Of the Twitter accounts we purchased, only 47% included the email address and password used to confirm the account. Merchants will sometimes re-appropriate these email addresses and sell them as "second-hand" at a discount of 20%. Without the original credentials, miscreants will be unable to perform email reconfirmation. Even if merchants adapt and begin to provide email credentials as part of their sale, the possibility of reselling email addresses disappears, cutting into a merchant's revenue.

CAPTCHAs CAPTCHA:s serve to both increase the cost of accounts due to the requirement of a CAPTCHA solving service as well as to throttle the rate of account creation. In our experience, when required, CAPTCHAs prevent merchants from registering 92% of fraudulent accounts. Services could also leverage this failure rate as a signal for blacklisting an IP address in real-time, cutting into the number of accounts merchants can register from a single IP.

IP Blacklisting: While miscreants have thousands of IP addresses at their disposal that rapidly change, IP blacklisting is not without merit. Our results show that merchants use a small fraction of IPs to register tens of thousands of accounts, which services could curb with real-time blacklisting. While public and commercial IP blacklists exist such as CBL [12], previous work has shown these generate too many false positives in the case of social spam [115], requiring service providers to generate and maintain their own blacklists.

Phone Verification: While Twitter does not require phone verification, we observe the positive impact phone verification has on increasing the cost of fraudulent accounts for other services. Facebook and GMail accounts that are phone verified cost up to 150x more than their Twitter, non-PVA counterpart. As with CAPTCHAs or email reconfirmation, phone verification can serve as a soft action against spammers who do not clearly fall into the set of accounts that should be automatically disabled.

7.5 Developing a Classifier to Detect Fraudulent Registrations

To understand the impact account merchants have on Twitter spam, we develop a classifier trained on purchased accounts to *retroactively* identify abusive registrations. Our technique relies on identifying patterns in the naming conventions and registration process used by merchants to automatically generate accounts. We apply our classifier to *all* Twitter accounts registered in the last year (overlapping with our investigation) and identify several million accounts which appear to be fraudulent. We note this approach is *not* meant to sustain accuracy in an adversarial setting; we only apply it to historical registrations where adaptation to our signals is impossible.

7.5.1 Automatic Pattern Recognition

Our detection framework begins by leveraging the limited variability in naming patterns used by account generation algorithms which enables us to automatically construct regular expressions that fingerprint fraudulent accounts. Our approach for generating these expressions is similar to previous techniques for identifying spam emails based on URL patterns [136] or spam text templates [95, 97]. However, these previous approaches fail on small text corpuses (e.g., screennames), especially when samples cannot be linked by repeating substrings. For this reason, we develop a technique explicitly for account naming patterns. Algorithm 3 shows a sketch of our approach which we use to guide our discussion.

Algorithm 3 Generate Merchant Pattern

```
 Input: List of accounts for a single merchant
 Parameters: $\tau$ (minimum cluster size)
clusters ← GROUP accounts BY
        ($\Sigma$-Seq, repeatedNames, emailDomain)
for all cluster ∈ clusters do
    if cluster.size() > $\tau$ then
        patterns ← MinMax$\Sigma$-Seq(cluster)
        OutputRegex(patterns, repeatedNames)
    end if
end for
```

Common Character Classes: To capture accounts that all share the same naming structure, we begin by defining a set of character classes:

$$\Sigma = \{p\{Lu\}, p\{Ll\}, p\{Lo\}, d, \dots\}$$

composed of disjoint sets of characters including uppercase Unicode letters, lowercase Unicode letters, non-cased Unicode letters (e.g., Arabic). and digits.[5] We treat all other characters as distinct classes (e.g., +, -, _). We chose these character classes based on the naming patterns of accounts we purchase, a sample of which we show in Table 7.6. We must support Unicode as registration algorithms draw account names from English, Cyrillic, and Arabic.

From these classes we define a function Σ-*Seq* that captures transitions between character classes and produces an ordered set $\sigma_1\sigma_2\dots\sigma_n$ of arbitrary length, where σ_i represents the i-th character class in a string. For example, we interpret the account `Wendy Hunt` from *accs.biz* as a sequence $p\{Lu\}p\{Ll\}\ p\{Lu\}p\{Ll\}$. We repeat this process for the name, screenname, and email of each account. We note that for emails, we strip the email domain (e.g., @hotmail.com) prior to processing and use this as a separate feature in the process for pattern generation.

Repeated Substrings: While repeated text stems between multiple accounts are uncommon due to randomly selected dictionary names, we find the algorithms used to generate accounts often reuse portions of text for names, screennames, and emails. For instance, all of the accounts in Table 7.6 from *victoryservices* have repeated substrings between an account's first name and screenname.

To codify these patterns, we define a function *repeatedNames* that canonicalizes text from an account's fields, brute forces a search of repeated substrings, and then codifies the resulting patterns as invariants. Canonicalization entails segmenting a string into multiple substrings based on Σ-Seq transitions. We preserve full names by ignoring transitions between upper and lowercase letters; spaces are also omitted from canonicalization. We then convert all

[5]We use Java character class notation, where p{*} indicates a class of letters and Lu indicates uppercase, Ll lowercase, and Lo non-case.

substrings to their lowercase equivalent, when applicable. To illustrate this process, consider the screenname `WendyHunt5`. Canonicalization produces an ordered list [`wendy`,`hunt`,5], while the name `Wendy Hunt` is converted to [`wendy`,`hunt`].

The function repeatedNames proceeds by performing a brute force search for repeated substrings between all canonicalized fields of an account. For our previous example of `WendyHunt5`, one successful match exists between name[1] and screenname[1], where [i] indicates the i-th position of a fields substring list; this same pattern also holds for the name and screenname for `Kristina Levy`. We use this positional search to construct invariants that hold across accounts from a single merchant. Without canonicalization, we could not specify what relationship exists between `Wendy` and `Kristina` due to differing text and lengths. When searching, we employ both exact pattern matching as well as partial matches (e.g., `neff` found in `brindagtgneff` for *buyaccs.com*). We use the search results to construct invariants for both strings that must repeat as well as strings that never repeat.

Clustering Similar Accounts: Once we know the Σ-Seq, repeatedNames, and email domain of every account from a merchant, we cluster accounts into non-overlapping groups with identical patterns, as described in Algorithm 3. We do this on a per-merchant basis rather than for every merchant simultaneously to distinguish which merchant an account originates from. We prune small clusters based on a empirically determined τ to reduce false positives, with our current implementation dropping clusters with fewer than 10 associated accounts.

Bounding Character Lengths: The final phase of our algorithm strengthens the invariants tied to Σ-Seq transitions by determining a minimum length $min(\sigma_i)$ and maximum length $max(\sigma_i)$ of each character class σ_i. We use these to define a bound $\{l_{min}, l_{max}\}$ that captures all accounts with the same Σ-Seq. Returning to our examples in Table 7.6, we group the account names from *accs.biz* and produce an expression:

$$p\{Lu\}\{1,1\}p\{Ll\}\{5,8\}\ \{1,1\}p\{Lu\}\{1,1\}p\{Ll\}\{4,4\}$$

We combine these patterns with the invariants produced by repeatedNames to construct a regular expression that fingerprints a cluster. We refer to these patterns for the rest of this chapter as *merchant patterns.*

7.5.2 Pattern Refinement

We refine our merchant patterns by including abuse-orientated signals that detect automated signup behavior based on the registration process, user-agent data, and timing events.

Signup Flow Events: We begin our refinement of merchant patterns by analyzing the activities of purchased accounts during and immediately after the signup work flow. These activities include events such as a user importing contacts and accessing a new user tutorial.

Seller	Popularity	Name	Screenname	Email
victoryservices	57%	`Trstram Aiken`	`Trstramsse912`	`KareyKay34251@hotmail.com`
		`Millicent Comolli`	`Millicentrpq645`	`DanHald46927@hotmail.com`
accs.biz	46%	`Wendy Hunt`	`WendyHunt5`	`imawzgaf7083@hotmail.com`
		`Kristina Levy`	`KristinaLevy6`	`exraytj8143@hotmail.com`
formefor	43%	`ola dingess`	`olawhdingess`	`TimeffTicnisha@hotmail.com`
		`brinda neff`	`brindagtgneff`	`ScujheShananan@hotmail.com`
spamvilla.com	38%	`Kiera Barbo`	`Kierayvydb`	`LinJose344@hotmail.com`
		`Jeannine Allegrini`	`Jeanninewoqzg`	`OpheliaStar461@hotmail.com`

Table 7.6: Obfuscated sample of names, screennames, and emails of purchased accounts used to automatically generate seller patterns. Popularity denotes the fraction of accounts that match the pattern for an individual merchant.

The complete list of these events is sensitive information and is omitted from discussion. Many of these events go untriggered by the automated algorithms used by account sellers, allowing us to distinguish automated registrations from legitimate users.

Given a cluster of accounts belonging to a single merchant, we generate a binary feature vector $e_{sig} = \{0, 1\}^n$ of the n possible events triggered during signup. A value of 1 indicates that at least ρ accounts in the cluster triggered the event e. For our experiments, we specify a cutoff $\rho = 5\%$ based on reducing false positives. Subsequently, we determine whether a new account with event vector e matches a seller's signup flow signature e_{sig} by computing whether $e \subseteq e_{sig}$ holds. The majority of legitimate accounts have $|e| \gg |e_{sig}|$, so we reject the possibility they are automated even though their naming conventions may match a merchant's.

User Agents: A second component of signups is the user agent associated with a form submission. Direct matching of user agents used by a seller with new subsequent signups is infeasible due to sellers randomizing user agents. For instance, *buytwitteraccounts.info* uses a unique (faked) agent for every account in our purchased dataset. Nevertheless, we can identify uniformity in the naming conventions of user agents just as we did with account names and screennames.

Given a cluster of accounts from a single seller, we generate a *prefix tree* containing every account's user agent. A node in the tree represents a single character from a user agent string while the node's depth mirrors the character's position in the user agent string. Each node also contains the fraction of agents that match the substring terminated at the given node. Rather than find the longest common substring between all accounts, we prune the tree so that every substring terminating at a node has a fraction of at least ϕ accounts in the cluster (in practice, 5%). We then generate the set of all substrings in the prefix tree and use them to match against the agents of newly registered accounts. The resulting substrings include pattens such as `Mozilla/5.0 (X11; Linux i686` which, if not truncated, would include multiple spurious browser toolbars and plugins and be distinct from subsequent signups. While in theory the resulting user agent substrings can be broad, in practice we find they capture browser variants and operating systems before being truncated.

Form Submission Timing: The final feature from the signup process we use measures the time between Twitter serving a signup form to the time the form is submitted. We then compute a bound $\{min_{ts}, max_{ts}\}$ for each seller to determine how quickly a seller's algorithm completes a form. To counter outliers, we opt for the 99% for both minimum and maximum time. For instance, the Fiverr merchant *kathlyn* registers accounts within $\{0, 1\}$ seconds. A newly minted account can match a seller's algorithm if its form completion time is within the sellers bound.

7.5.3 Alternative Signals

There were a number of alternative signals we considered, but ultimately rejected as features for classification. We omitted the delay between an account's registration and subsequent activation as we lacked training data to measure this period; all our accounts remain dormant after purchase (minus the small fraction that were resold). We also analyzed both the timing of registrations as well as the interarrival times between successive registrations. We found that merchants sell accounts in blocks that sometimes span months, preventing any interarrival analysis. Furthermore, merchants register accounts at uniformly random hours and minutes. Finally, as merchants create accounts from IP addresses around the globe, no subnet or country accurately captures a substantive portion of abusive registrations.

7.5.4 Evaluation

To demonstrate the efficacy of our model, we retroactively apply our classifier to *all* Twitter accounts registered in the last year. In total, we identify several million[6] distinct accounts that match one of our merchant patterns and thus are potentially fraudulent. We validate these findings by analyzing both the *precision* and *recall* of our model as well measuring the impact of time on the model's overall accuracy.

Precision & Recall: Precision measures the fraction of identified accounts that are in fact fraudulent (e.g., not misclassified, legitimate users), while recall measures the fraction of all possible fraudulent accounts that we identify, limited to the merchants that we study. To estimate the precision of each merchant pattern, we select a random sample of 200 accounts matching each of our 26 merchant patterns,[7] for a total of 4,800 samples. We then manually analyze the login history, geographic distribution of IPs, activities, and registration process tied to each of these accounts and label them as spam or benign. From this process, we estimate our overall precision at 99.99%, with the breakdown of the 10 most popular merchant pattern precisions shown in Table 7.7. In a similar vein, we estimate recall by

[6]Due to operational concerns, we are unable to provide exact numbers on the volume of spam accounts registered. As such, we reference merchants and the impact they have on Twitter as a *relative volume* of all several million accounts that we detect.

[7]We omit accounts purchased from the Freelancer merchant *shivnagsudhakar* as these were registered over a year ago and thus lay outside the range of data to which we had access.

Service	Rel. Volume	P	R
buuman	0.00%	100.00%	70.67%
smokinbluelady	0.08%	100.00%	98.91%
danyelgallu	0.12%	100.00%	100.00%
twitter007	0.13%	100.00%	97.33%
kathlyn	0.13%	100.00%	93.24%
jackhack08	0.41%	100.00%	100.00%
twitfollowers	0.72%	100.00%	92.50%
denial93	2.18%	100.00%	100.00%
ghetumarian	3.05%	100.00%	85.94%
formefor	4.75%	100.00%	100.00%
shivnagsudhakar	—	—	—
kamalkishover	29.90%	99.60%	92.73%
naveedakhtar	0.24%	100.00%	98.40%
webmentors2009	0.85%	100.00%	99.64%
truepals	1.02%	100.00%	93.08%
alexissmalley	1.68%	100.00%	98.62%
victoryservices	6.33%	99.70%	99.03%
spamvilla.com	0.71%	99.00%	98.70%
getbulkaccounts.com	2.97%	100.00%	100.00%
xlinternetmarketing.com	3.12%	100.00%	95.13%
accs.biz	4.48%	100.00%	97.62%
buytwitteraccounts.biz	6.10%	100.00%	84.27%
quickaccounts.bigcartel	10.91%	100.00%	99.73%
buytwitteraccounts.info	20.45%	99.60%	81.85%
dataentryassistant.com	24.01%	100.00%	96.57%
buyaccountsnow.com	30.75%	99.10%	95.10%
buyaccs.com	58.39%	100.00%	91.66%
Total	100.00%	99.99%	95.08%

Table 7.7: Breakdown of the merchants, the relative volume of all detected accounts in the last year that match their pattern, precision (P) and recall (R).

calculating the fraction of all accounts we purchase that match our classifier. In total, we correctly identify 95% of all purchased accounts; the remaining 5% of missed accounts did not form large enough clusters to be included in a merchant's pattern, and as a result, we incorrectly classified them as legitimate.

Performance Over Time: The performance of our model is directly tied to accurately tracking adaptations in the algorithms used by merchants to register accounts. To under-

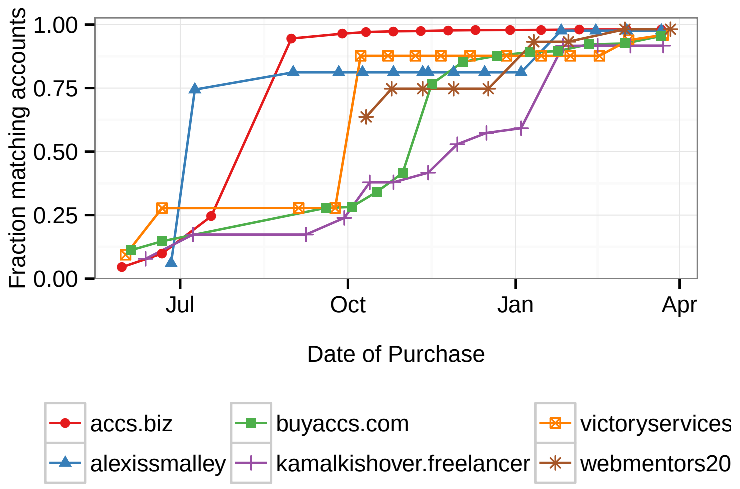


Figure 7.5: Recall of generated merchant patterns for all purchased accounts as a function of training the classifier on data only prior to time t.

stand how frequently these adaptations occur, we evaluate the performance of our classifier as a function of time. Figure 7.5 shows the overall recall of each of our merchant patterns for the sellers we track over the longest period of time. For each merchant, we train a classifier on accounts acquired up to time t and evaluate it on all accounts from the merchant, regardless of when we purchased the account. We find that some sellers such as *alexissmalley* rarely alter their registration algorithm throughout our study, allowing only two purchase to enable accurate detection. In contrast, we see a shift in registration algorithms for a number of merchants around October and January, but otherwise patterns remain stable for long periods. The several million accounts we identify as fraudulent should thus be viewed as a lower bound in the event we missed an adaptation.

Pattern Overlap & Resale: The simultaneous adaptation of merchant patterns in Figure 7.5 around October and other periods leads us to believe that a multitude of merchants are using the same software to register accounts and that an update was distributed. Alternatively, the account marketplace may have multiple levels of resale (or even arbitrage) where accounts from one merchant are resold by another for an increased cost, leading to correlated adaptations. Further evidence of correlated patterns appears in the merchant patterns we construct, where a classifier for one merchant will accurately detect accounts sold to us by a second merchant. For instance, the accounts sold by *kamalkishover* from Freelancer overlap with the patterns of 9 other merchants, the most popular of which is *buyaccountsnow.com*. We find most Fiverr sellers are independent with the exception of *denial93*, *ghetumarian*, and *formefor*, whose patterns overlap with the major account web storefronts. This would explain why these three Fiverr sellers appear to be much larger (from the perspective of Table 7.7) compared to other Fiverr merchants. As a result, our estimates for the number of accounts registered by each merchant may be inflated, though our final total counts only unique matches and is thus globally accurate.

7.6 Impact of the Underground Market

We analyze the several million accounts we flag as registered by merchants operating in the underground market and estimate the fraction that have been sold and used to generate Twitter spam. We find that, during active months, the underground market was responsible for registering 10—20% of all accounts that Twitter later flagged as spam. For their efforts, we estimate that merchants generated a combined revenue between $127,000—$459,000.

7.6.1 Impact on Twitter Spam

From our seed set of 121,027 accounts purchased from 27 merchants, we are able to identify several million fraudulent accounts that were registered by the same merchants. Of these, 73% were sold and actively tweeting or forming relationships at one point in time, while the remaining 37% remained dormant and were yet to be purchased. Only 40% of the accounts identified were suspended by Twitter at the time of our analysis.

In cooperation with Twitter, we analyzed the total fraction of all suspended accounts that appear to originate from the merchants we track, shown in Figure 7.6. At its peak, the underground marketplace was responsible for registering 60% of all accounts that would go on to be suspended for spamming. During more typical periods of activity, the merchants we track contribute 10—20% of all spam accounts. We note that the drop off around April does not indicate a lack of recent activity; rather, as accounts are stockpiled for months at a time, they have yet to be released into the hands of spammers, which would lead to their suspension. The most damaging merchants from our impact analysis operate out of blackhat forums and web storefronts, while Fiverr and Freelancer sellers generate orders of magnitude fewer accounts.[8] We are now actively working with Twitter to disable the accounts we detected as fraudulent and to disrupt the future efforts of these merchants.

7.6.2 Estimating Revenue

We estimate the revenue generated by the underground market based on the total accounts sold and the prices charged during their sale. We distinguish accounts that have been sold from those that lay dormant and await sale based on whether an account has sent tweets or formed relationships. For sold accounts, we identify which merchant created the account and determine the minimum and maximum price the merchant would have charged for

[8]The exception to this is a Freelancer merchant kamalkishover, but based on their merchant pattern overlapping with 9 other merchants, we believe they are simply reselling accounts.

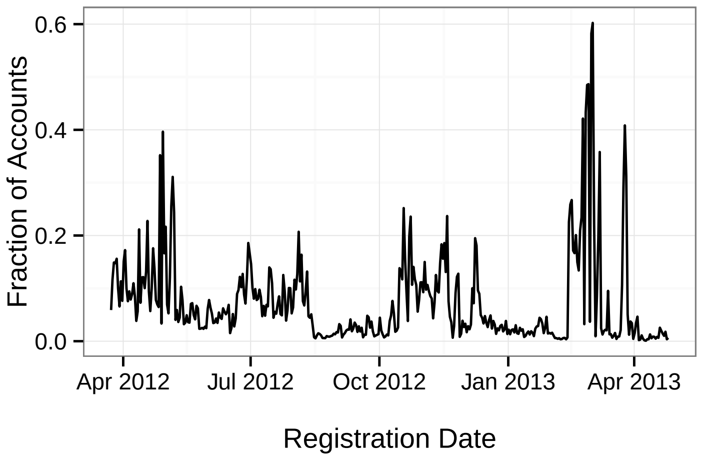


Figure 7.6: Fraction of all suspended accounts over time that originate from the underground market.

that account based on our historical pricing data.[9] In the event multiple merchants could have generated the account (due to overlapping registration patterns), we simply take the minimum and maximum price of the set of matching merchants.

We estimate that the total revenue generated by the underground account market through the sale of Twitter credentials is between the range of \$127,000—\$459,000 over the course of 10 months. We note that many of the merchants we track simultaneously sell accounts for a variety of web services, so this value likely represents only a fraction of their overall revenue. Nevertheless, our estimated income is far less than the revenue generated from actually sending spam [75] or selling fake anti-virus [111], where revenue is estimated in the tens of millions. As such, account merchants are merely stepping stones for larger criminal enterprises, which in turn disseminate scams, phishing, and malware throughout Twitter.

7.7 Disrupting the Underground Market for Fraudulent Accounts

With Twitter's cooperation, we disable 95% of all fraudulent accounts registered by the 27 merchants we track, including those previously sold but not yet suspended for spamming. Throughout this process, we simultaneously monitor the underground market to track fall-out and recovery. While we do not observe an appreciable increase in pricing or delay in merchant's delivering new accounts, we find 90% of all purchased accounts immediately after

[9]Determining the exact time of sale for an account is not possible due to the potential of miscreants stockpiling their purchases; as such, we calculate revenue for both the minimum and maximum possible price.

our actioning are suspended on arrival. While we successfully deplete merchant stockpiles containing fraudulent accounts, we find that within two weeks merchants were able to create fresh accounts and resume selling working credentials.

7.7.1 Suspending Identified Accounts

In order to disrupt the abusive activities of account merchants, we worked with Twitter's Anti-spam, SpamOps, and Trust and Safety teams to manually validate the accuracy of our classifier and tune parameters to set an acceptable bounds on false positives (legitimate users incorrectly identified as fraudulent accounts). Once tuned, we applied the classifier outlined in Section 7.5 to every account registered on Twitter going back to March, 2012, filtering out accounts that were already suspended for abusive behavior.

From the set of accounts we identified[10], Twitter iteratively suspended accounts in batches of ten thousand and a hundred thousand before finally suspending all the remaining identified accounts. At each step we monitored the rate of users that requested their accounts be unsuspended as a metric for false positives, where unsuspension requests require a valid CAPTCHA solution. Of the accounts we suspended, only 0.08% requested to be unsuspended. However, 93% of these requests were performed by fraudulent accounts abusing the unsuspend process, as determined by manual analysis performed by Twitter. Filtering these requests out, we estimate the final precision of our classifier to be 99.9942%. The tuned classifier has a recall of 95%, the evaluation of which is identical to the method presented in Section 7.5. Assuming our purchases are a random sample of the accounts controlled by the underground market, we estimate that 95% of all fraudulent accounts registered by the 27 merchants we track were disabled by our actioning.

7.7.2 Marketplace Fallout and Recovery

Immediately after Twitter suspended the last of the underground market's accounts, we placed 16 new orders for accounts from the 10 merchants we suspected of controlling the largest stockpiles. Of 14,067 accounts we purchased, 90% were suspended on arrival due to Twitter's previous intervention. When we requested working replacements, one merchant responded with:

> All of the stock got suspended ... Not just mine .. It happened with all of the sellers .. Don't know what twitter has done ...

Similarly, immediately after suspension, *buyaccs.com* put up a notice on their website stating "Временно не продаем аккаунты Twitter.com", translating via Google roughly to "Temporarily not selling Twitter.com accounts".

[10]Due to operational concerns, we cannot specify the exact volume of accounts we detect that were not previously suspended by Twitter's existing defenses.

While Twitter's initial intervention was a success, the market has begun to recover. Of 6,879 accounts we purchased two weeks after Twitter's intervention, only 54% were suspended on arrival. As such, long term disruption of the account marketplace requires both increasing the cost of account registration (as outlined in Section 7.4) and integrating at-signup time abuse classification into the account registration process (similar to the classifier outlined in Section 7.5). We are now working with Twitter to integrate our findings and existing classifier into their abuse detection infrastructure.

7.8 Summary of Results

We have presented a longitudinal investigation of the underground market tied to fraudulent Twitter credentials, monitoring pricing, availability, and fraud perpetrated by 27 account merchants. These merchants specialize in circumventing automated registration barriers by leveraging thousands of compromised hosts, CAPTCHA solvers, and access to fraudulent Hotmail, Yahoo, and mail.ru credentials. We identified which registration barriers positively influenced the price of accounts and distilled our observations into a set of recommendations for how web services can improve existing barriers to bulk signups. Furthermore, we developed a classifier based on at-registration abuse patterns to successfully detect several million fraudulent accounts generated by the underground market. During active months, the 27 merchants we monitor appeared responsible for registering 10—20% of all accounts later flagged by Twitter as spam. For their efforts, these merchants generated an estimated revenue between $127,000—$459,000. With Twitter's help, we successfully suspended 95% of all accounts registered by the 27 merchants we track, depleting the account stockpiles of numerous criminals. We are now working with Twitter to integrate our findings and existing classifier into their abuse detection infrastructure.

Chapter 8

Conclusion

As users increasingly turn to online social networks to share information and fuel political discourse, they are exposed to scams, phishing, and malware which social network operators currently lack adequate defenses to prevent. Such attacks exploit the trust users place in their relationships and the integrity of information found in online social networks.

To address these challenges, we analyzed the range of threats currently targeting online social networks through the lens of Twitter. We mapped out the support infrastructure that is critical to online social network abuse, characterized the tools and techniques used to disseminate malignant content, and evaluated how such attacks ultimately realize a profit for the attackers involved. In the process, we argue that the for-profit infrastructure provided by the underground economy in the form of fake accounts and affiliate programs has become a fundamental weak point of abuse.

To this end, we developed two new strategies for preventing abuse in social networks. Our first defense identifies abusive links in online social networks before they are distributed to recipients. Our second defense identifies fraudulent accounts at the time of their registration, preventing criminals from ever interacting with the legitimate users of online social networks. Combined, these two strategies effectively defend both the ingress points of abuse—fraudulent and compromised account—and the egress points of abuse—spam links that direct victims to spamvertised products, fake software, clickfraud, banking theft, and malware that converts a victim's machine into a commodity for the underground economy.

8.1 Impact

The core value of conducting data-driven security research is exposing flaws in existing assumptions about how to address spam and abuse as well as fueling the adoption of meaningful, long-term solutions. Such solutions must avoid the reactive development cycle typified by industry-lead security and overcome the ease with which criminals adapt to new defenses. We highlight some of our most important findings and the implications they have for the social network research community and industry.

8.1.1 Failure of Existing Defenses

Our results in Chapter 3, Chapter 4, and Chapter 5 demonstrate that existing technologies such as blacklists or abuse-related heuristics are failing to prevent spam and abuse. Blacklist updates are too slow to operate in the real-time environment of social networks, where the majority of victims are exposed to harmful content before existing blacklists even become aware of the threat. Furthermore, if social network operators do not crawl linked content to unravel redirect chains, simple evasive techniques such as URL shortening completely undermine any effectiveness blacklists might have. Similarly, even though abuse-related heuristics are able to identify fraudulent accounts within a day of when criminals engage with victims, the ease by which criminals can generate new accounts leads to a constant stream of abuse.

Contrary to research that advocates identifying fraudulent accounts via their social graph, we demonstrate that spammers rarely form relationships with legitimate users. Instead, they rely on features provided by social networks to reach a global audience such as mentions and hashtags. Even so, targeting these individual features with abuse-related heuristics is a poor solution. Spammers are constantly adapting how they engage with victims. As such, the signals that heuristics rely upon are easily evaded by criminals. At the same time, the perpetual development cycle of new social networking features means that defenders must enact new spam heuristics for each product change, leaving defenders in a purely reactive position. The components we believe are fundamental to abuse (and thus more difficult to evade) are the ingress points of spam—compromised and fraudulent account—and the egress points of spam—spam links that direct victims off of social networks to a web page that generates a profit. For this reason, we advocate that defenders should concentrate on these two points rather than fighting the intermediate, multifaceted abuse conducted through multiple engagement vectors.

8.1.2 Adopting New Defenses

In Chapter 6 and Chapter 7, we advocate two new strategies for preventing abuse in social networks. Our first defense identifies abusive links in online social networks (or any web service) before they are distributed to recipients. Our second defense relies on directly engaging with the underground economy that fuels online social network abuse to understand how millions of fake accounts are registered in an automated fashion. We leverage this understanding to detect abusive accounts at the time of their registration, preventing criminals from ever interacting with the legitimate users of online social networks.

With respect to industry adoption, we know that *bit.ly* is using a similar technique to the one we presented in Chapter 6 and that Twitter is also using URL-based features for detecting abuse. Furthermore, we are actively working with Twitter, Google, and other major social networking companies to adapt our strategy for detecting fraudulent accounts (presented in Chapter 7) to their networks.

8.2 Parting Words

The challenge of securing social networks never ends. So long as legal and technical solutions lag in preventing criminals from generating a profit from abuse, a threat will always exist to users engaging in online social networks. When researchers develop strategies to address this challenge, they should move beyond simple metrics such as the accuracy of a classifier or false positives to measure success. Instead, more meaningful metrics that treat abuse as an *economic problem* can be adopted. These include reaction time to new threats; the user perception of spam throughout a social network and its impact on engagement; or whether defenses increase the cost incurred by criminals. Such metrics capture the economic value derived by criminals or the value lost to companies targeted with spam, both of which ultimately fuel the abuse landscape. Finally, solutions to online social network abuse should be data-driven, combining domain expertise, machine learning, and distributed computation to identify what threats are most pressing. This dissertation is an example of the value derived from empirical security research.

Bibliography

[1] Advanced Network Technology Center. *Univeristy of Oregon Route Views Project.* `http://www.routeviews.org/`. 2010.

[2] Alexa. *Alexa Top 500 Global Sites.* `http://www.alexa.com/topsites`. 2012.

[3] Amazon Web Services. *Amazon EC2 Instance Types.* `http://aws.amazon.com/ec2/instance-types/`. 2009.

[4] D.S. Anderson, C. Fleizach, S. Savage, and G.M. Voelker. "Spamscatter: Characterizing internet scam hosting infrastructure". In: *USENIX Security.* 2007.

[5] E. Bakshy, B. Karrer, and L. Adamic. "Social influence and the diffusion of user-created content". In: *Proceedings of the 10th ACM conference on Electronic commerce.* ACM. 2009, pp. 325–334.

[6] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. "The role of social networks in information diffusion". In: *Proceedings of the 21st international conference on World Wide Web.* ACM. 2012, pp. 519–528.

[7] M. Barreno, B. Nelson, R. Sears, A.D. Joseph, and J. Tygar. "Can machine learning be secure?" In: *Proceedings of the ACM Symposium on Information, Computer and Communications Security.* 2006.

[8] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. "Detecting Spammers on Twitter". In: *Proceedings of the Conference on Email and Anti-Spam (CEAS).* 2010.

[9] bit.ly. *Spam and Malware Protection.* `http://blog.bit.ly/post/138381844/spam-and-malware-protection`. 2009.

[10] C. Buenviaje. *Fake Facebook Toolbar Makes Rounds.* `http://blog.trendmicro.com/trendlabs-security-intelligence/fake-facebook-toolbar-makes-rounds/`. 2013.

[11] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. "Measuring pay-per-install: The commoditization of malware distribution". In: *USENIX Security Symposium.* 2011.

[12] CBL. *Composite Blocking List.* `http://cbl.abuseat.org/`. 2012.

[13] K. Chellapilla and A. Maykov. "A taxonomy of JavaScript redirection spam". In: *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web.* 2007.

[14] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru. “Phi.sh/$oCiaL: The Phishing Landscape Through Short URLs”. In: *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*. 2011.

[15] C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. “Insights from the Inside: A View of Botnet Management from Infiltration”. In: *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats*. 2010.

[16] A. Chowdhury. *State of Twitter spam*. `http://blog.twitter.com/2010/03/state-of-twitter-spam.html`. Mar. 2010.

[17] G. Cluley. *This you????: Phishing attack hits Twitter users*. `http://www.sophos.com/blogs/gc/g/2010/02/24/phishing-attack-hits-twitter-users/`. 2010.

[18] D. Coldewey. *Romney Twitter account gets upsurge in fake followers, but from where?* `http://www.nbcnews.com/technology/romney-twitter-account-gets-upsurge-fake-followers-where-928605`. 2012.

[19] M. Cova, C. Kruegel, and G. Vigna. “Detection and analysis of drive-by-download attacks and malicious JavaScript code”. In: *Proceedings of the 19th International Conference on World Wide Web*. 2010.

[20] N. Dalvi, P. Domingos, S.S. Mausam, and D. Verma. “Adversarial classification”. In: *Proceedings of the International Conference on Knowledge Discovery and Data Mining*. 2004.

[21] G. Danezis and P. Mittal. “Sybilinfer: Detecting sybil nodes using social networks”. In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2009.

[22] J. Duchi and Y. Singer. “Efficient Online and Batch Learning Using Forward Backward Splitting”. In: *Journal of Machine Learning Research* 10 (Dec. 2009), pp. 2899–2934.

[23] M. Duggan and J. Brenner. *The Demographics of Social Media Users*. `http://pewinternet.org/Commentary/2012/March/Pew-Internet-Social-Networking-full-detail.aspx`. 2013.

[24] P. Eckersley. “How Unique Is Your Web Browser?” In: *Privacy Enhancing Technologies (PET)*. 2010.

[25] S. Egelman, L.F. Cranor, and J. Hong. “You’ve been warned: an empirical study of the effectiveness of web browser phishing warnings”. In: *Proceeding of the Conference on Human Factors in Computing Systems*. 2008.

[26] W. Englund and K. Lally. *In Protests, Two Russias Face Off*. `http://wapo.st/wiVnV8`. 2011.

[27] J. Epstein. *President Obama Google+ Chat Gets Personal*. `http://politi.co/zTvgQO`. 2012.

[28] Facebook. *Explaining Facebook's Spam Prevention Systems.* `http://blog.facebook.com/blog.php?post=403200567130`. 2010.

[29] Facebook. *Newsroom.* `https://newsroom.fb.com/Key-Facts`. 2013.

[30] M. Felegyhazi, C. Kreibich, and V. Paxson. "On the potential of proactive domain blacklisting". In: *Proceedings of the USENIX Conference on Large-scale Exploits and Emergent Threats.* Apr. 2010.

[31] R. Flores. *The real face of Koobface.* `http://blog.trendmicro.com/the-real-face-of-koobface/`. Aug. 2009.

[32] J. Franklin, V. Paxson, A. Perrig, and S. Savage. "An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants". In: *Proceedings of ACM Conference on Computer and Communications Security.* Oct. 2007, pp. 375–388.

[33] F–Secure. *Twitter now filtering malicious URLs.* `http://www.f-secure.com/weblog/archives/00001745.html`. 2009.

[34] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B.Y. Zhao. "Detecting and characterizing social spam campaigns". In: *Proceedings of the Internet Measurement Conference (IMC).* 2010.

[35] S. Garera, N. Provos, M. Chew, and A.D. Rubin. "A framework for detection and measurement of phishing attacks". In: *Proceedings of the 2007 ACM Workshop on Recurring Malcode.* 2007.

[36] E. Gilbert and K. Karahalios. "Predicting tie strength with social media". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM. 2009, pp. 211–220.

[37] D. Goodin. "Scammers skirt spam shields with help from Adobe Flash". In: *The Register* (2010). `http://www.theregister.co.uk/2008/09/04/spammers_using_adobe_flash/`.

[38] J. D. Goodman. *In Mexico, Social Media Become a Battleground in the Drug War.* `http://nyti.ms/wgWUZb`. 2011.

[39] Google. *Google Safebrowsing API.* `http://code.google.com/apis/safebrowsing/`. 2010.

[40] M. Granovetter. "The strength of weak ties". In: *American journal of sociology* (1973), pp. 1360–1380.

[41] C. Grier, K. Thomas, V. Paxson, and M. Zhang. "@spam: the underground on 140 characters or less". In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS).* 2010.

[42] C. Grier, L. Ballard, J. Caballero, N. Chachra, C.J. Dietrich, K. Levchenko, P. Mavrommatis, D. McCoy, A. Nappa, A. Pitsillidis, et al. "Manufacturing Compromise: The Emergence of Exploit-as-a-Service". In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS).* 2012.

[43] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. “Information diffusion through blogspace”. In: *Proceedings of the 13th international conference on World Wide Web*. ACM. 2004, pp. 491–501.

[44] Hadoop. *Hadoop Distributed File system*. `http://hadoop.apache.org/hdfs/`. 2010.

[45] J. Halliday. *David Cameron Considers Banning Suspected Rioters from Social Media*. `http://bit.ly/xI8MJs`. 2011.

[46] D. Harvey. *Trust and Safety*. `http://blog.twitter.com/2010/03/trust-and-safety.html`. Mar. 2010.

[47] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer, 2009.

[48] C. Herley. “So long, and no thanks for the externalities: The rational rejection of security advice by users”. In: *Proceedings of the 2009 Workshop on New Security Paradigms Workshop*. 2009.

[49] T. Holz, C. Gorecki, F. Freiling, and K. Rieck. “Detection and mitigation of fast-flux service networks”. In: *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS)*. 2008.

[50] C.Y. Hong, F. Yu, and Y. Xie. “Populated IP Addresses—Classification and Applications”. In: *Proceedings of the 19th ACM Conference on Computer and Communications Security*. 2012.

[51] HootSuite. *Kapow! HootSuite Fights the Evils of Phishing, Malware, and Spam*. `http://blog.hootsuite.com/hootsuite-fights-malware-phishing/`. 2010.

[52] D. Ionescu. *Twitter Warns of New Phishing Scam*. `http://www.pcworld.com/article/174660/twitter_warns_of_new_phishing_scam.html`. Oct. 2009.

[53] N. Jindal and B. Liu. “Opinion Spam and Analysis”. In: *Proceedings of the International Conference on Web Search and Web Data Mining*. 2008.

[54] Joewein. *Joewein.de LLC – fighting spam and scams on the Internet*. `http://www.joewein.net/`.

[55] John E. Dunn. “Zlob Malware Hijacks YouTube”. In: (2007). `http://www.pcworld.com/article/133232/zlob_malware_hijacks_youtube.html`.

[56] J.P. John, A. Moshchuk, S.D. Gribble, and A. Krishnamurthy. “Studying spamming botnets using Botlab”. In: *Usenix Symposium on Networked Systems Design and Implementation (NSDI)*. 2009.

[57] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G.M. Voelker, V. Paxson, and S. Savage. “Spamalytics: An empirical analysis of spam marketing conversion”. In: *Proceedings of the 15th ACM Conference on Computer and Communications Security*. 2008.

[58] H. Kelley. *83 million Facebook accounts are fakes and dupes*. `http://www.cnn.com/2012/08/02/tech/social-media/facebook-fake-accounts/index.html`. 2012.

[59] Kim Zetter. "Trick or Tweet? Malware Abundant in Twitter URLs". In: *Wired* (2009).

[60] F. Klien and M. Strohmaier. "Short Links Under Attack: Geographical Analysis of Spam in a URL Shortener Network". In: *Proceedings of the Conference on Hypertext and Social Media*. 2012.

[61] B. Krebs. *Battling the Zombie Web Site Armies*. `https://krebsonsecurity.com/2011/01/battling-the-zombie-web-site-armies`. 2011.

[62] B. Krebs. *Spam Volumes: Past & Present, Global & Local*. `http://krebsonsecurity.com/2013/01/spam-volumes-past-present-global-local/`. 2012.

[63] B. Krebs. *Twitter Bots Drown Out Anti-Kremlin Tweets*. `http://bit.ly/w9Gnaz`. 2011.

[64] B. Krebs. *Twitter Bots Target Tibetan Protests*. `http://krebsonsecurity.com/2012/03/twitter-bots-target-tibetan-protests/`. 2012.

[65] C. Kreibich, C. Kanich, K. Levchenko, B. Enright, G.M. Voelker, V. Paxson, and S. Savage. "Spamcraft: An inside look at spam campaign orchestration". In: *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*. 2009.

[66] H. Kwak, C. Lee, H. Park, and S. Moon. "What is Twitter, a Social Network or a News Media?" In: *Proceedings of the International World Wide Web Conference*. 2010.

[67] K. Lee, J. Caverlee, and S. Webb. "Uncovering social spammers: social honeypots+ machine learning". In: *Proceeding of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2010.

[68] S. Lee and J. Kim. "Warningbird: Detecting Suspicious URLs in Twitter Stream". In: *Symposium on Network and Distributed System Security (NDSS)*. 2012.

[69] K. Levchenko, A. Pitsillidis, N. Chachra, Br, o. Enright, M. Felegyhazi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and S. Savage. "Click Trajectories: End-to-End Analysis of the Spam Value Chain". In: *Proceedings of the IEEE Symposium on Security and Privacy*. May 2011.

[70] D. Lowd and C. Meek. "Adversarial learning". In: *Proceedings of the International Conference on Knowledge Discovery in Data Mining*. 2005.

[71] C. Ludl, S. McAllister, E. Kirda, and C. Kruegel. "On the effectiveness of techniques to detect phishing sites". In: *Detection of Intrusions and Malware, and Vulnerability Assessment* (2007).

[72] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker. "Beyond blacklists: learning to detect malicious web sites from suspicious urls". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2009.

[73] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker. “Identifying suspicious URLs: an application of large-scale online learning”. In: *Proceedings of the 26th Annual International Conference on Machine Learning.* 2009.

[74] MaxMind. *Resources for Developers.* `http://www.maxmind.com/app/api`. 2010.

[75] D. McCoy, A. Pitsillidis, G. Jordan, N. Weaver, C. Kreibich, B. Krebs, G.M. Voelker, S. Savage, and K. Levchenko. “Pharmaleaks: Understanding the business of online pharmaceutical affiliate programs”. In: *Proceedings of the 21st USENIX conference on Security symposium.* USENIX Association. 2012.

[76] R. McDonald, K. Hall, and G. Mann. “Distributed Training Strategies for the Structured Perceptron”. In: *Proceedings of the North American Association for Computing Linguistics (NAACL).* Los Angeles, CA, June 2010.

[77] D.K. McGrath and M. Gupta. “Behind phishing: an examination of phisher modi operandi”. In: *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats.* 2008.

[78] R. McMillan. *Stolen Twitter accounts can fetch $1,000.* `http://www.computerworld.com/s/article/9150001/Stolen_Twitter_accounts_can_fetch_1_000`. 2010.

[79] M. McPherson, L. Smith-Lovin, and J. Cook. “Birds of a feather: Homophily in social networks”. In: *Annual review of sociology* (2001), pp. 415–444.

[80] B. Meeder, J. Tam, P. G. Kelley, and L. F. Cranor. “RT @IWantPrivacy: Widespread Violation of Privacy Settings in the Twitter Social Network”. In: *Web 2.0 Security and Privacy.* 2010.

[81] M. Memmott. *AP Twitter Account Hacked, Tweet About Obama Shakes Market.* `http://www.npr.org/blogs/thetwo-way/2013/04/23/178620410/ap-twitter-account-hacked-tweet-about-obama-shakes-market`. 2013.

[82] A. Metwally and M. Paduano. “Estimating the number of users behind ip addresses for combating abusive traffic”. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM. 2011.

[83] B. Miller, P. Pearce, C. Grier, C. Kreibich, and V. Paxson. “What’s Clicking What? Techniques and Innovations of Today’s Clickbots”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment* (2011), pp. 164–183.

[84] E. Mills. *Facebook hit by phishing attacks for a second day.* `http://news.cnet.com/8301-1009_3-10230980-83.html`. 2009.

[85] A. Monroy-Hernández, E. Kiciman, M. De Choudhury, S. Counts, et al. “The new war correspondents: the rise of civic media curation in urban warfare”. In: *Proceedings of the 2013 conference on Computer supported cooperative work.* ACM. 2013, pp. 1443–1452.

[86] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G.M. Voelker, and S. Savage. "Re: Captchas–understanding captcha-solving services in an economic context". In: *USENIX Security Symposium*. Vol. 10. 2010.

[87] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G.M. Voelker. "An analysis of underground forums". In: *Proceedings of the Internet Measurement Conference (IMC)*. 2011.

[88] M. Motoyama, D. McCoy, K. Levchenko, S. Savage, and G.M. Voelker. "Dirty Jobs: The Role of Freelance Labor in Web Service Abuse". In: *Proceedings of the 20th USENIX Security Symposium*. 2011.

[89] M. Motoyama, D. McCoy, K. Levchenko, Geoffrey M. Voelker, and S. Savage. "Dirty Jobs: The Role of Freelance Labor in Web Service Abuse". In: *Proceedings of the USENIX Security Symposium*. San Francisco, CA, Aug. 2011.

[90] Mozilla. *API & Language References*. `https://addons.mozilla.org/en-US/developers/docs/reference`. 2010.

[91] Mozilla. *Netscape Plugin API*. `http://www.mozilla.org/projects/plugins/`. 2004.

[92] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. "Detecting spam web pages through content analysis". In: *Proceedings of the 15th International Conference on World Wide Web*. 2006.

[93] Office of the Press Secretary. *White House to Host Twitter @TOWNHALL*. `http://1.usa.gov/zplVBV`. 2011.

[94] Pew Research Center. *Social Networking Popular Across Globe*. `http://www.pewglobal.org/2012/12/12/social-networking-popular-across-globe/`. 2012.

[95] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G.M. Voelker, V. Paxson, N. Weaver, and S. Savage. "Botnet Judo: Fighting Spam with Itself". In: (2010).

[96] A. Pitsillidis, K. Levchenko, C. Kreibich, C. Kanich, G.M. Voelker, V. Paxson, N. Weaver, and S. Savage. "Botnet Judo: Fighting spam with itself". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2010.

[97] P. Prasse, C. Sawade, L, N. wehr, and T. Scheffer. "Learning to Identify Regular Expressions that Describe Email Campaigns". In: *International Conference on Machine Learning*. 2012.

[98] J. Preston. *What Does 40 Mean to You?* `http://nyti.ms/zfMuQ2`. 2011.

[99] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. "All Your iFRAMEs Point to Us". In: *Proceedings of the 17th Usenix Security Symposium*. July 2008, pp. 1–15.

[100] M.A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao. "The Nocebo Effect on the Web: An Analysis of Fake Anti-Virus Distribution". In: *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*. 2010.

[101] Ramach, A. ran, N. Feamster, and S. Vempala. "Filtering spam with behavioral blacklisting". In: *Proceedings of the 14th ACM Conference on Computer and Communications Security.* 2007.

[102] L. Rao. *Twitter Seeing 90 Million Tweets Per Day, 25 Percent Contain Links.* `http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day/`. Sept. 2010.

[103] M. Richtel. *Egypt Cuts Off Most Internet and Cell Service.* `http://nyti.ms/z44cWc`. 2011.

[104] W.E. Ricker. *Computation and interpretation of biological statistics of fish populations.* Vol. 191. Department of the Environment, Fisheries and Marine Service Ottawa, 1975.

[105] E. Schonfeld. *When it comes to URL Shoteners, bit.ly is now the biggest.* `http://techcrunch.com/2009/05/07/when-it-comes-to-url-shorteners-bitly-is-now-the-biggest/`. May 2009.

[106] S. Sinha, M. Bailey, and F. Jahanian. "Improving spam blacklisting through dynamic thresholding and speculative aggregation". In: *Proceedings of the 17th Annual Network & Distributed System Security Symposium.* 2010.

[107] S. Sinha, M. Bailey, and F. Jahanian. "Shades of grey: On the effectiveness of reputation-based blacklists". In: *3rd International Conference on Malicious and Unwanted Software.* 2008.

[108] socialcapital. *Twitter, Facebook and YouTube's Role in Arab Spring.* `http://bit.ly/xxBNmo`. 2011.

[109] J. Song, S. Lee, and J. Kim. "Spam Filtering in Twitter using Sender-Receiver Relationship". In: *Proceedings of International Symposium on Recent Advances in Intrusion Detection (RAID).* 2011.

[110] B. Stone. "Facebook Joins With McAfee to Clean Spam From Site". In: *New York Times* (2010).

[111] B. Stone-Gross, R. Abman, R. Kemmerer, C. Kruegel, D. Steigerwald, and G. Vigna. "The Underground Economy of Fake Antivirus Software". In: *Proceedings of the Workshop on Economics of Information Security (WEIS).* 2011.

[112] 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: *Proceedings of the 16th ACM conference on Computer and communications security.* 2009, pp. 635–647.

[113] B. Stone-Gross, R. Stevens, A. Zarras, R. Kemmerer, C. Kruegel, and G. Vigna. "Understanding Fraudulent Activities in Online Ad Exchanges". In: *Proceedings of the Internet Measurement Conference (IMC).* 2011.

[114] G. Stringhini, C. Kruegel, and G. Vigna. “Detecting Spammers on Social Networks”. In: *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. 2010.

[115] K. Thomas, C. Grier, and V. Paxson. “Adapting social spam infrastructure for political censorship”. In: *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats*. USENIX Association. 2012.

[116] K. Thomas, C. Grier, V. Paxson, and D. Song. “Suspended Accounts In Retrospect: An Analysis of Twitter Spam”. In: *Proceedings of the Internet Measurement Conference*. Nov. 2011.

[117] K. Thomas and D. M. Nicol. “The Koobface Botnet and the Rise of Social Malware”. In: *Proceedings of The 5th International Conference on Malicious and Unwanted Software (Malware 2010)*. 2010.

[118] TrendMicro. *The Dark Side of Social Media*. `http://http://bit.ly/zn217U`. 2011.

[119] Twitter. *Building on Open Source*. `http://blog.twitter.com/2009/01/building-on-open-source.html`. 2010.

[120] Twitter. *Numbers*. `http://blog.twitter.com/2011/03/numbers.html`. Mar. 2011.

[121] Twitter. *Terms of service*. `http://twitter.com/tos`. May 2011.

[122] Twitter. *The Twitter Rules*. `http://help.twitter.com/forums/26257/entries/18311`. 2009.

[123] Twitter. *The Twitter Rules*. `http://support.twitter.com/entries/18311-the-twitter-rules`. 2010.

[124] Twitter. *Twitter API Wiki*. `http://dev.twitter.com/doc`. 2010.

[125] Twitter Engineering. *The Engineering Behind Twitter’s New Search Experience*. `http://bit.ly/iuRwp8`. 2011.

[126] URIBL. *URIBL.COM – Realtime URI blacklist*. `http://uribl.com/`. 2010.

[127] S. Venkataraman, S. Sen, O. Spatscheck, P. Haffner, and D. Song. “Exploiting network structure for proactive spam mitigation”. In: *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. 2007.

[128] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y. Zhao. “Serf and Turf: Crowdturfing for Fun and Profit”. In: *Proceedings of the International World Wide Web Conference*. 2011.

[129] Y. 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: *Proceedings of the 2006 Network and Distributed System Security Symposium (NDSS)*. Feb. 2006.

[130] R. Wauters. *China Blocks Access To Twitter, Facebook After Riots*. `http://tcrn.ch/yaxKjP`. 2009.

[131] K.Q. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. "Feature Hashing for Large Scale Multitask Learning". In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2009.

[132] C. Whittaker, B. Ryner, and M. Nazif. "Large-Scale Automatic Classification of Phishing Pages". In: *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2010.

[133] K. Wickre. *Celebrating Twitter7*. `https://blog.twitter.com/2013/celebrating-twitter7`. 2013.

[134] C. Wisniewski. *Twitter hack demonstrates the power of weak passwords*. `http://www.sophos.com/blogs/chetw/g/2010/03/07/twitter-hack-demonstrates-power-weak-passwords/`. Mar. 2010.

[135] M. Wu, R.C. Miller, and S.L. Garfinkel. "Do security toolbars actually prevent phishing attacks?" In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006.

[136] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. "Spamming botnets: Signatures and characteristics". In: *Proceedings of ACM SIGCOMM* (2008).

[137] S. Yadav, A.K.K. Reddy, A.L.N. Reddy, and S. Ranjan. "Detecting Algorithmically Generated Malicious Domain Names". In: *Proceedings of the Internet Measurement Conference (IMC)*. 2010.

[138] C. Yang, 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: *Proceedings of the 21st International Conference on World Wide Web*. 2012.

[139] H. Yu, M. Kaminsky, P.B. Gibbons, and A. Flaxman. "Sybilguard: defending against sybil attacks via social networks". In: *ACM SIGCOMM Computer Communication Review* (2006).

[140] B. Zadrozny, J. Langford, and N. Abe. "Cost-Sensitive Learning by Cost-Proportionate Example Weighting". In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2003.

[141] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica. "Spark: Cluster Computing with Working Sets". In: *Proceedings of the 2nd USENIX Conference on Hot topics in Cloud Computing*. 2010.

[142] Y. Zhang, J.I. Hong, and L.F. Cranor. "Cantina: a content-based approach to detecting phishing web sites". In: *Proceedings of the 16th international conference on World Wide Web*. 2007.

[143] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. "Botgraph: Large scale spamming botnet detection". In: *Proceedings of the 6th Symposium on Network System Design and Implementation (NSDI)*. 2009.