UNIVERSITY OF CALIFORNIA
RIVERSIDE

Detecting Social Malware and its Ecosystem in Online Social Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Md Sazzadur Rahman

December 2012

Acknowledgments

I would like to express my gratitude to my advisors Dr. Michalis Faloutsos and Dr. Harsha V. Madhyastha for their guidance, constant support and inspiration throughout my PhD study—I could not have asked for anything better. Special thanks to my committee: Dr. Srikanth Krishnamurthy and Dr. Iulian Neamtiu for their valuable feedback which helped me to improve my thesis. I wish to thank my parents for their support and inspiration. Finally, I would like to thank my wife, Ruhani Afreen, without whom none of this would have been possible.

ABSTRACT OF THE DISSERTATION

Detecting Social Malware and its Ecosystem in Online Social Networks

by

Md Sazzadur Rahman

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2012
Dr. Michalis Faloutsos, Chairperson

Online social networks (OSNs) have become the new vector for cybercrime, and hackers are finding new ways to propagate spam and malware on these platforms, which we refer to as social malware. As we show here, social malware cannot be identified with existing security mechanisms (e.g., URL blacklists), because it exploits different weaknesses and often has different intentions.

In this dissertation, we show that social malware is prevalent in Facebook, the largest OSN to date with more then a billion users and develop an efficient and scalable social malware detection system that takes advantage of the *social context* of posts. We deploy this detection system as a Facebook app called MyPageKeeper to protect Facebook users from social malware. We find that our detection method is both accurate and efficient. Furthermore, we show that, social malware significantly differs from traditional email spam or web-based malware.

One of the major factors for enabling social malware is malicious third-party apps. We show that such malicious apps are also widespread in Facebook. Therefore, to identify malicious apps, we ask the question: given a Facebook application, can we determine if it is malicious? Our key contribution in this part is in developing

FRAppE—Facebook's Rigorous Application Evaluator—arguably the first tool focused on detecting malicious apps on Facebook. We identify a set of features that help us distinguish malicious apps from benign ones. For example, we find that malicious apps often share names with other apps, and they typically request fewer permissions than benign apps. Then, leveraging these distinguishing features, we show that FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and a low false negative rate (4.1%). Finally, we explore the ecosystem of malicious Facebook apps. We identify mechanisms these apps use to propagate and find that many apps collude and support each other.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The new battleground for cybercrime is **Online Social Networks (OSNs)**, which provides a new, fertile, and unexplored environment for the dissemination of malware. Moving beyond spam email, the distribution of malware on OSNs takes the form of postings and communications between *friends*. We use the term **social malware** to describe parasitic or damaging behavior including identity theft, distribution of malicious URLs, spam, and malicious apps that utilizes OSNs. The use of posts from friends adds a powerful element in the propagation of social malware: it comes implicitly with *the endorsement of a friend* who allegedly posts the information. This new social dimension adds to the challenges in fighting web-based crime: (a) the techniques employed by hackers are constantly evolving, and (b) the general public is uninformed, gullible, and easily enticed into visiting suspicious websites or installing apps with the lure of false rewards (e.g., free iPads in memory of Steve Jobs [71]). Beyond this being a nuisance, social malware also enables cyber-crime, with several Facebook scams resulting in loss of real money for users [27, 30] and malicious Facebook apps harvesting users personal information while the level of sophistication of the hackers increases [22]. Therefore, it is becoming increasingly important to understand social malware better and build better defenses to protect users from the crime underlying this social malware.

Detecting social malware requires novel approaches since hackers use significantly different approaches in its distribution compared to email-based spam. For example, reputation-based filtering [69, 68, 113] is insufficient to detect social malware received from friends and, as we show later, the keywords used in email spam significantly differ from those used in social malware. We also find that URL blacklists designed to detect phishing and malware on the web do not suffice, e.g., because a large fraction of social malware (26% in our dataset) points to malicious applications hosted on Facebook. Although such malicious apps are widespread in Facebook, as we show later, currently there is no commercial service, publicly-available information, or research-based tool to advise a user about the risks of an app.

In this dissertation, our goal is to detect such malicious and surreptitious activities on Facebook by identifying a) malicious posts and b) malicious third-party applications. Social malware posts typically include at least one embedded URL link, since without such a link, the posts cannot lure and hurt users or propagate virally. We want to be able to detect such posts on the walls and news feeds of Facebook users, and alert users exposed to social malware so that they do not click through on the URLs included in the posts. Next, we want to detect malicious applications and alert users once they try to install the app in their profiles. Such malicious applications not only steal users valuable information (i.e., email, gender, age group and home town), but also provide a great means to spread social malware on users wall to hackers. For example, once a malicious app is installed by a victim, it periodically posts victim's wall with a URL pointing to itself or other apps for promotion. Finally, we want to investigate the ecosystem supporting and lunching malicious app attacks in Facebook.

1.1 Detecting Social Malware in OSN

In this dissertation, we first present the design and implementation of a Facebook application, MyPageKeeper [57], that we develop specifically for the purpose of protecting Facebook users from social malware. Until October 2011, MyPageKeeper had been installed by more than 12K Facebook users (since its launch in June 2011). By monitoring the news feeds of these users, we also observe posts on the walls of the 2.4 million friends of these users. We evaluate MyPageKeeper using a dataset of over 40 million posts that it inspected during the four month period from June to October 2011.

Our key contributions for detecting social malware can be grouped into three main thrusts.

- **Designing an accurate, efficient, and scalable detection method.** In order to operate MyPageKeeper at scale, but at low cost, the distinguishing characteristic of our approach is our strident focus on efficiency. Prior solutions for detecting spam and malware on OSNs (which we describe in detail later) rely on information obtained either by crawling the URLs included in posts or by performing DNS resolution on these URLs. In contrast, our social malware classifier relies solely on the *social context* associated with each post (e.g., the number of walls and news feeds in which posts with the same embedded URL are observed, and the similarity of text descriptions across these posts). Note that this approach means that we do not even resolve shortened URLs (e.g., using services like bit.ly) into the full URLs that they represent. This approach maximizes the rate at which we can classify posts, thus reducing the cost of resources required to support a given population of users.

  We employ a Machine Learning classification module using Support Vector Machines on a carefully selected set of such features that are readily available from the observed posts. 97% of posts flagged by our classifier are indeed social malware and it incorrectly flags only 0.005% of benign posts. Furthermore, it requires an average of only 46 ms to classify a post.

- **Social malware is a new kind of malware.** We show that social malware is significantly different than traditional email spam or web-based malware. First, URL blacklists cannot detect social malware effectively. These blacklists identify only 3% of the malicious posts that MyPageKeeper flags. The inability of website blacklists to identify social malware is partly due to the fact that a significant fraction of social malware is hosted on popular blogging domains such as `blogspot.com` and on Facebook itself. Specifically, 26% of the flagged posts point to Facebook apps or pages. Moreover, we also observe a low overlap between the keywords associated with email based spam and those we find in social malware.
- **Quantifying social malware: prevalence and intention.** We find that 49% of our users were exposed to at least one social malware post in four months. We also identify a new type of parasitic behavior, which we refer to as "Like-as-a-Service". Its goal is to artificially boost the number of "Likes" of a Facebook page. With the lure of games and rewards, several Facebook apps push users to *Like* the Facebook pages of say a store or a product, thus artificially inflating their reputation on Facebook. This further confirms the difference between social malware and other forms of malware propagation.

1.2 Detecting Malicious Applications in OSN

An important part of identifying social malware is to detect and profile malicious apps, since, as we show here, a significant fraction of social malware are propagated by such malicious Facebook applications. In this dissertation, we take the first step towards profiling and detecting malicious Facebook apps. Our work asks the following question, given a Facebook application, can we determine if it is malicious? Our key contribution in this part is in developing FRAppE—Facebook's Rigorous Application Evaluator—arguably the first tool focused on detecting malicious apps on Facebook. To build FRAppE, we use data from MyPageKeeper that monitors the Facebook profiles of 2.2 million users. We analyze 111,000 apps that made 91 million posts over nine months. This is arguably the first comprehensive study focusing on malicious Facebook apps that focuses on quantifying, profiling, and understanding malicious apps, and synthesizes this information into an effective detection approach.

Our work makes the following key contributions for identifying malicious Facebook apps:

- **Malicious Facebook applications are prevalent; 13% of the observed apps are malicious.** We show that malicious apps are prevalent in Facebook and reach a large number of users. We find that 13% of apps in our dataset of 111K distinct apps are malicious. Also, 60% of malicious apps endanger more than 100K users each by convincing them to follow the links on the posts made by these apps, and 40% of malicious apps have over 1,000 monthly active users each.
- **Malicious and benign app profiles differ significantly.** We systematically profile apps and show that malicious app profiles are significantly different than those of benign apps. A striking observation is the "laziness" of hackers; many malicious apps have the same name, as 8% of unique names of malicious apps are each used by more than 10 different apps (as defined by their app IDs). Overall, we profile apps based on two classes of features: (a) those that can be obtained on-demand given an application's identifier (e.g., the permissions required by the app and the posts in the

application's profile page), and (b) others that require a cross-user view to aggregate information across time and across apps (e.g., the posting behavior of the app and the similarity of its name to other apps).

- **FRAppE can detect malicious apps with 99% accuracy.** We develop FRAppE (Facebook's Rigorous Application Evaluator) to identify malicious apps either using only features that can be obtained on-demand or using both on-demand and aggregation-based app information. FRAppE Lite, which only uses information available on-demand, can identify malicious apps with 99.0% accuracy, with low false positives (0.1%) and false negatives (4.4%). By adding aggregation-based information, FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and lower false negatives (4.1%).

1.3 Understanding Ecosystem of Malicious Applications in OSN

Equipped with an accurate classifier for detecting malicious apps, we next analyze how malicious Facebook apps support each other. Our analysis shows that malicious apps do not operate in isolation—many malicious apps share the same name, several of them redirect to the same domain upon installation, etc. Upon deeper investigation, we identify a worrisome and, at the same time, fascinating trend: malicious apps work collaboratively in promoting each other. Namely, apps make posts that contain links to the installation pages of other apps. We use the term AppNets do describe these colluding groups; we claim that they are for the social world what botnets are for the world of physical devices.

Our key contributions for understanding AppNets ecosystem are as follows:

- **Malicious apps collude at massive scale.** We conduct a forensics investigation on the malicious app ecosystem to identify and quantify the techniques used to promote malicious apps. The most interesting result is that apps collude and collaborate at a massive scale. Apps promote other apps via posts that point to the "promoted" apps. If we describe the collusion relationship of promoting-promoted apps as a graph, we find 1,584 promoter apps that promote 3,723 other apps. These apps form large and highly-dense connected components.
- **Hackers often use sophisticated techniques to promote malicious apps.** Hackers use fast-changing indirection: applications posts have URLs that point to a website, and the website dynamically redirects to many different apps; we find 103 such URLs that point to 4,676 different malicious apps over the course of a month. These observed behaviors indicate well-organized crime: one hacker controls many malicious apps, which we will call an AppNet, since they seem a parallel concept to botnets.
- **Malicious hackers impersonate applications.** We were surprised to find popular good apps, such as 'FarmVille' and 'Facebook for iPhone', posting malicious posts. On further investigation, we found a lax authentication rule in Facebook that enabled hackers to make malicious posts appear as though they came from these apps.

1.4 Summary

In summary, our research makes three high-level contributions to the study of OSN security. First, our work demonstrates the prevalence of social malware in Facebook, the largest OSN to date. We show that certain machine learning techniques equipped with social context based features are very effective in identifying social malware. Since similar social context based features are also available in other OSNs (i.e. Twitter, Pinterest), we speculate that our technique is applicable to those OSNs for identifying social malware. For example, tweets (Twitter posts) include readily available social context based features such as text, embeded URL, mention, hashtag etc. Similarly, Pinterest, another leading OSN, provides number of like, comments, repins etc. in each pin (Pinterest posts).

Second, we show that a significant fraction of social malware are enabled by malicious third-party apps which are also prevalent in Facebook. We demonstrate that benign and malicious Facebook apps differ significantly in terms of number of required permissions or posting behavior. Further, we developed FRAppE to identify malicious Facebook apps.

Finally, we identify the ecosystem of such malicious apps and show that they collude at large scale. Malicious hackers use different sophisticated techniques such as fast-changing indirection to promote malicious apps so that a fraction of such apps can survive in case others are disabled by Facebook. We made several recommendations to Facebook and hope that Facebook will benefit from these recommendations for reducing the menace of hackers on their platform.

Chapter 2

Indentifying Social Malware

As online social networks (OSNs) are becoming the new epicenter of the web, hackers are expanding their territory to these services [20]. Anyone using Facebook or Twitter is likely to be familiar with what we call here **social malware**: fake, annoying, possibly damaging posts from friends of the potential victim. The propagation of social malware takes the form of postings and communications between friends on OSNs. Users are enticed into visiting suspicious websites or installing apps with the lure of false rewards (e.g., free iPads in memory of Steve Jobs [71]), and they unwittingly send the post to their friends, thus enabling a viral spreading. This is exactly where the power of social malware lies: posts come with the implicit endorsement of the sending friend. Beyond this being a nuisance, social malware also enables cyber-crime, with several Facebook scams resulting in loss of real money for users [27, 30].

Defenses against email spam are insufficient for identifying social malware since reputation-based filtering [69, 68, 113] is insufficient to detect social malware received from friends and, as we show later, the keywords used in email spam significantly differ from those used in social malware. We also find that URL blacklists designed to detect phishing and malware on the web do not suffice, e.g., because a large fraction of social malware (26% in our dataset) points to suspicious applications hosted on Facebook. Finally, though Facebook has its own mechanisms for detecting and removing malware [118], they seem to be less aggressive either due to what they define as malware or due to computational limitations.

In this chapter, we present our experience in developing and deploying MyPageKeeper, a Facebook security app with more than 12K installs and 40 million posts collected over roughly four months. After users install it, MyPageKeeper monitors their walls and news feeds, detects and flags malicious posts. MyPageKeeper was officially launched in June 2011 and attracted significant attention from the electronic and printed press.

In MyPageKeeper, we set two requirements for a successfully identifying social malware: (a) accuracy, and (b) low computation. The second requirement makes the solution deployable in practice: ideally, we want to be able to protect all users in a non-intrusive online fashion; we want (near) real-time responses at a massive scale. Prior solutions for detecting spam on OSNs rely on crawling the URLs included in posts or by performing DNS resolutions on these URLs. However, our approach only relies on social context readily available in each posts. Our Machine Learning based classifier takes only 46ms on average to classify a post. The first thought is to reuse all the security services that exist for email anti-spam and URL blacklists, maintained by Google and other organizations. However, as we show later, this is not enough: social malware is different than previous cyber-crime activities and techniques. For example these blacklists identify only 3% of the malicious posts that MyPageKeeper flags. On the contrary, 97% of posts flagged by our classifier are indeed social malware and it incorrectly flags only 0.005% of benign posts.

Through the lens of MyPageKeeper, we found that 49% of our users were exposed to at least one social malware post in four months. We also identify a new type of parasitic behavior, which we refer to as “Like-as-a-Service”. It lures users with fake rewards and pushes users to *Like* the Facebook pages of say a store or a product and thus artificially boost the number of “Likes” of that Facebook page. These applications, which offer Likes as a service, presumably get paid on a Pay-per-Like model by the owners of Facebook pages that make use of their services.

2.1 Social Malware on Facebook

In this section, we provide relevant background about Facebook, and we describe typical characteristics of social malware found on Facebook.

2.1.1 Facebook Terminology

Facebook is the largest online social network today with over 900 million registered users, roughly half of whom visit the site daily. Here, we discuss some standard Facebook terminology relevant to our work.

- *Post*: a post represents the basic unit of information shared on Facebook. Typical posts either contain only text (status updates), a URL with an associated text description, or a photo/album shared by a user. In our work, we focus on posts that contain URLs.
- *Wall*: a Facebook user’s wall is a page where friends of the user can post messages to the user. Such messages are called wall posts. Other than to the user herself, posts on a user’s wall are visible to other users on Facebook determined by the user’s privacy settings. Typically a user’s wall is made visible to the user’s friends, and in some cases to friends of friends.

- *News feed*: a Facebook user's news feed page is a summary of the social activity of the user's friends on Facebook. For example, a user's news feed contains posts that one of the user's friends may have shared with all of her friends. Facebook continually updates the news feed of every user and the content of a user's news feed depends on when it is queried.
- *Like*: like is a Facebook widget that is associated with an object such as a post, a page, or an app. If a user clicks the Like widget associated with an object, the object will appear in the news feed of the user's friends and thus allow information about the object to spread across Facebook. Moreover, the number of Likes (i.e., the number of users who have clicked the Like widget) received by an object also represents the reputation or popularity of the object.
- *Application*: Facebook allows third-party developers to create their own applications that Facebook users can add. Every time a user visits an application's page on Facebook, Facebook dynamically loads the content of the application from a URL, called the canvas URL, pointing to the application server provided by the application's developer. Since content of an application is dynamically loaded every time a user visits the application's page on Facebook, the application developer enjoys great control over content shown in the application page. The Facebook platform uses OAuth 2.0 [10] for user authentication, application authorization and application authentication. Here, application authorization ensures that the users grant precise data (e.g., email address) and capabilities (e.g., ability to post on the user's wall) to the applications they choose to add, and application authentication ensures that a user grants access to her data to the correct application.

2.1.2 Social Malware

We start by defining the meaning of *social malware*. We describe typical characteristics of social malware and elaborate on how social malware distinguishes itself from traditional email spam and web malware.

What is social malware? Our intention is to use the term social malware to encompass all criminal and parasitic behavior in an OSN, including anything that annoys, hurts, or makes money off of the user. In the context of this dissertation, we consider a Facebook post as malicious, if it satisfies one of the following conditions: (1) the post spreads malware and compromises the device of the user, (2) the web page pointed to by the post requires the user to give away personal information, (3) the post promises false rewards (e.g., free products), (4) the post is made on a user's behalf without the user's knowledge (typically by having previously lured the user into providing relevant permissions to a rogue Facebook app), (5) the web page pointed to by the post requires the user to carry out tasks (e.g., fill out surveys) that help profit the owner of that website, or (6) the post causes the user to artificially inflate the reputation of the page (e.g., by forcing the user to 'Like' the page). While the first two criteria are typical malware and phishing, the latter four are distinctive of social malware.

Note that, as with email spam, there can be some ambiguity in the definition of social malware: a post considered as annoying by one user may be considered useful by another user. In practice, our ultimate litmus test is the opinion of MyPageKeeper's users: if most of them report a post as annoying, we will flag it as such.

How does social malware work? Social malware appears in a Facebook user's wall or news feed typically in the form of a post which contains two parts. First, the post contains a URL , usually obfuscated with a URL shortening service, to a webpage that hosts either malicious or spam content. Second, social malware posts typically contain a catchy text message (e.g. *"two free Southwest tickets"*) that entice users to click on the URL included in the post. Optionally, social malware posts also contain a thumbnail screenshot of the landing page of the URL, also used to entice the user to click on the link. For example, a purported image of Osama's corpse is included in a post that claims to point to a video of his death.

The operation of most social malware epidemics can be associated with two distinct mechanisms.

a. Propagation mechanism. Once a user follows the embedded URL to the target website, the post tries to propagate itself through that user. For this, the user is often asked to complete several steps in order to obtain the fake reward (e.g., "Free Facebook T-shirt"). These steps involve "Liking" or "sharing" the post, or posting the social malware on the user's wall. Thus, the cycle continues with the friends of that user, who see the post in their news feed. In contrast, users seldom forward email spam to their friends.

b. Exploitation mechanism. The exploitation often starts after the propagation phase. The hacker attempts either to learn the user's private information via a phishing attack [21], to spread malware to user devices, or to make money by "forcing" a particular user action or response, such as completing a survey for which the hacker gets paid [49].

Where is social malware hosted? Social malware can be broadly classified into two categories based on the infrastructure that hosts them.

a. Social malware hosted outside Facebook: in this category, URLs point to a domain hosted outside Facebook. Since the URL points to a landing page outside

App Name	**Application Message**	**Monthly Active Users**
Free Phone Calls	I'm making a Free Call with the Free Phone Call Facebook App! ... I'll never pay for a phone call again. Make your free call at URL	435,392
The App	Check if a friend has deleted you URL	35,216
The App	Check if a friend has deleted you URL	25,778

Table 2.1: Three rogue Facebook applications identified by MyPageKeeper.

the OSN, hackers can directly launch the different kinds of attacks mentioned above once a user visits the URL in a social malware post. Though several URL blacklists should be able to flag such URLs, the process of updating these blacklists is too slow to keep up with the viral propagation of social malware on OSNs [99].

b. Social malware hosted on Facebook: a significant fraction of social malware is hosted on Facebook itself: the embedded URL points to a Facebook page or application. Naturally, current blacklists and reputation-based schemes fail to flag such URLs. Such URLs typically point to the following types of Facebook objects:

- **Malicious Facebook applications:** rogue applications post catchy messages (e.g., *"Check who deleted you from your profile"*) on the walls of users with a link pointing to the installation page of the application. Table 2.1 lists three such social malware-spreading applications in our data. Users are conned into installing the application to their profile and granting several permissions to it. The application then not only gets access to that user's personal information (such as email address, home town, and high school) but also gains the ability to post on the victim's wall. As before, posts on a user's wall typically appear on the news feeds of the user's friends, and the

Page Name	**Message to persuade 'Like'**	**No. of Likes**
Clif Bar	Hey there! Looking for a clif builder's coupon? Just like us by clicking the button above. thanks!	79919
FarmVille Bonus	You can't claim you you haven't clicked on the like button	94907
Courtesy Chevrolet	Like our page to play and have a change to win!	86287
Greggs The Bakers	Like us to claim your voucher	288039
Mobilink Infinity	Like us for big infinite fun	26105

Table 2.2: Top five pages identified by MyPageKeeper that persuade users to 'Like' them.

propagation cycle repeats. Creating such applications has become easy with ready to use toolkits starting at $25 [48]. We investigate malicious apps more in Chapters 3 and 4.

- **Malicious Facebook events:** sometimes hackers create Facebook events that contain a malicious link. One such event is the *'Get a free Facebook T-Shirt (Sponsored by Reebok)'* scam. This event page states that 500,000 users will get a free T-shirt from Facebook. To be one among those 500,000 users, a user must attend the event, invite her friends to join, and enter her shipping address.
- **Malicious Facebook pages:** another approach taken by hackers to spread social malware is to create a Facebook page and post spam links on the page [65]. We also identified a trend in aggressive marketing by companies that force users to click "Like" on their Facebook pages to spread their pages as well as increase the reputation of the page. Table 2.2 lists the top five such Facebook pages, along with the message on the

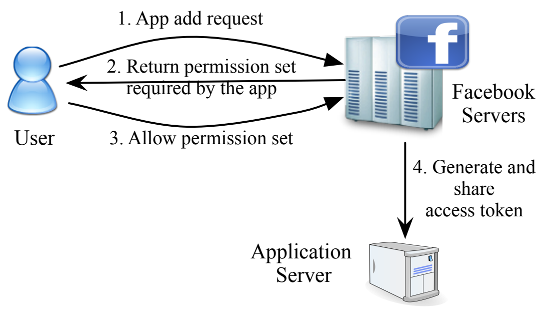


Figure 2.1: Application installation process on Facebook.

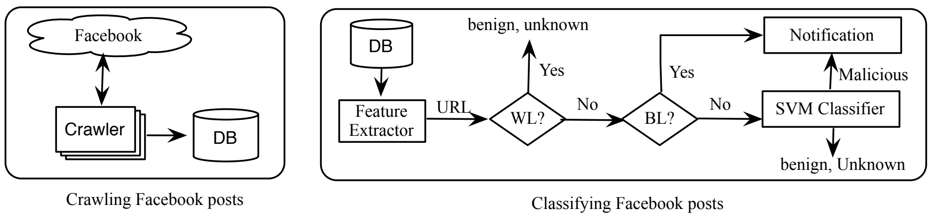


Figure 2.2: Architecture of MyPageKeeper.

page and the number of Likes received by these pages.

2.2 MyPageKeeper Architecture

To identify social malware and protect Facebook users from it, we develop MyPageKeeper. MyPageKeeper is a Facebook application that continually checks the walls and news feeds of subscribed users, identifies social malware posts, and alerts the users. We present our goals in designing MyPageKeeper, and then describe the system architecture and implementation details.

2.2.1 Goals

We design MyPageKeeper with the following three primary goals in mind.

1. Accuracy. Our foremost goal is to ensure accurate identification of social malware. We are faced with the obvious trade-off between missing malware posts (false negatives), and "crying wolf" too often (false positives). Although one could argue that minimizing false negatives is more important, users would abandon overly sensitive detection methods, as recognized by the developers of Facebook's Immune System [118].

2. Scalability. Our end goal is to have MyPageKeeper provide protection from social malware for all users on Facebook, not just for a select few. Therefore, the system must be scalable to easily handle increased load imposed by a growth in the number of subscribed users.

3. Efficiency. Finally, we seek to minimize our costs in operating MyPage-Keeper. The period between when a post first becomes visible to a user and the time it is checked by MyPageKeeper represents the window of vulnerability when the user is exposed to potential social malware. To minimize the resources necessary to keep this window of vulnerability short, MyPageKeeper's techniques for classification of posts must be efficient.

2.2.2 MyPageKeeper Components

MyPageKeeper consists of six functional modules.

a. User authorization module. We obtain a user's authorization to check her wall and news feed through a Facebook application, which we have developed. Once a user installs the MyPageKeeper application, we obtain the necessary credentials to access the posts of that user. For alerting the user, we also request permission to access the user's email address and to post on the user's wall and news feed. Figure 2.1 shows how a Facebook user authorizes an application.

b. Crawling module. MyPageKeeper periodically collects the posts in every user's wall and news feed. As mentioned previously, we currently focus only on posts that contain a URL. Apart from the URL, each post comprises several other pieces of information, such as a text message associated with the post, the user who made the post, number of comments and Likes on the post, and the time when the post was created.

c. Feature extraction module. To classify a post, MyPageKeeper evaluates every embedded URL in the post. Our key novelty lies in considering only the social context (e.g., the text message in the post, and the number of Likes on it) for the classification of the URL and the related post. Furthermore, we use the fact that we are observing more than one user, which can help us detect an epidemic spread. We discuss these features in more detail later in Section 2.2.3.

d. Classification module. The classification module uses a Machine Learning classifier based on Support Vector Machines, but also utilizes several local and external whitelists and blacklists that help speed up the process and increase the overall accuracy. The classification module receives a URL and the related social context features extracted in the previous step. Since the classification is our key contribution, we discuss this in more detail in Section 2.2.3. If a URL is classified as social malware, all posts containing the URL are labeled as such.

e. Notification module. The notification module notifies all users who have social malware posts in their wall or news feed. The user can currently specify the notification mechanism, which can be a combination of emailing the user or posting a comment on the suspect posts. In the future, we will consider allowing our system to remove the malicious post automatically, but this can create liabilities in the case of false positives.

f. User feedback module. Finally, to improve MyPageKeeper's ability to detect social malware, we leverage our user community. We allow users to report suspicious posts through a convenient user-interface. In such a report, the user can optionally describe the reason why she considers the post as social malware.

2.2.3 Identification of Social Malware

The key novelty of MyPageKeeper lies in the classification module (summarized in Figure 2.2). As described earlier, the input to the classification module is a URL and the related social context features extracted from the posts that contain the URL. Our classification algorithm operates in two phases, with the expectation that URLs and related posts that make it through either phase without a match are likely benign and are treated as such.

Using whitelists and blacklists. To improve the efficiency and accuracy of our classifier, we use lists of URLs and domains in the following two steps. First, MyPageKeeper matches every URL against a whitelist of popular reputable domains. We currently use a whitelist comprising the top 70 domains listed by Quantcast, excluding domains that host user-contributed content (e.g., OSNs and blogging sites). Any URL that matches this whitelist is deemed safe, and it is not processed further.

Second, all the URLs that remain are then matched with several URL blacklists that list domains and URLs that have been identified as responsible for spam, phishing, or malware. Again, the need to minimize classification latency forces us to only use blacklists that we can download and match against locally. Such blacklists include those from Google's Safe Browsing API [47], Malware Patrol [56], PhishTank [62], APWG [7], SpamCop [68], joewein [51], and Escrow Fraud [19]. Querying blacklists that are hosted externally, such as SURBL [72], URIBL [78] and WOT [79], will introduce significant latency and increase MyPageKeeper's latency in detecting social malware, thus inflating the window of vulnerability. Any URL that matches any of the blacklists that we use is classified as social malware.

Using machine learning with social context features. All URLs that do not match the whitelist or any of the blacklists are evaluated using a Support Vector Machines (SVM) based classifier. SVM is widely and successfully used for binary classification in security and other disciplines [109, 102] [73]. We train our system with a batch of manually labeled data, that we gathered over several months prior to the launch of MyPageKeeper. For every input URL and post, the classifier outputs a binary decision to indicate whether it is malicious or not. Our SVM classifier uses the following features.

Spam keyword score. Presence of spam keywords in a post provides a strong indication that the post is spam. Some examples of such spam keywords are 'FREE', 'Hurry', 'Deal', and 'Shocked'. To compile a list of such keywords that are distinctive to social malware, our intuition is to identify those keywords that 1) occur frequently in social malware posts, and 2) appear with a greater frequency in social malware as compared to their frequency in benign posts.

We compile such a list of keywords by comparing a dataset of manually identified social malware posts with a dataset of posts that contain URLs that match our whitelist (we discuss how to maintain this list of keywords in Section 2.6). We transform posts in either dataset to a bag of words with their frequency of occurrence. We then compute the likelihood ratio p_1/p_2 for each keyword where $p_1 = p(word|socialmalwarepost)$ and $p_2 = p(word|benignpost)$. The likelihood ratio of a keyword indicates the bias of the keyword appearing more in social malware than in benign posts. In our current implementation of MyPageKeeper, we have found that the use of the 6 keywords with the highest likelihood ratio values among the 100 most frequently occurring keywords in social malware is sufficient to accurately detect social malware.

Thereafter, to classify a URL, MyPageKeeper searches all posts that contain the URL for the presence of these spam keywords and computes a spam keyword score as the ratio of the number of occurrences of spam keywords across these posts to the number of posts.

Message similarity. If a post is part of a spam campaign, it usually contains a text message that is similar to the text in other posts containing the same URL (e.g., because users propagate the post by simply sharing it). On the other hand, when different users share the same popular URL, they are likely to include different text descriptions in their posts. Therefore, greater similarity in the text messages across all posts containing a URL portends a higher probability that the URL leads to spam. To capture this intuition, for each URL, we compute a message similarity score that captures the variance in the text messages across all posts that contain the URL. For each post, MyPageKeeper sums the ASCII values of the characters in the text message in the post, and then computes the standard deviation of this sum across all the posts that contain the URL. If the text descriptions in all posts are similar, the standard deviation will be low.

News feed post and wall post count. The more successful a spam campaign, the greater the number of walls and news feeds in which posts corresponding to the campaign will be seen. Therefore, for each URL, MyPageKeeper computes counts of the number of wall posts and the number of news feed posts which contained the URL.

Like and comment count. Facebook users can 'Like' any post to indicate their interest or approval. Users can also post comments to follow up on the post, again indicating their interest. Users are unlikely to 'Like' posts pointing to social malware or comment on such posts, since they add little value. Therefore, for every URL, MyPageKeeper computes counts of the number of Likes and number of comments seen across all posts that contain the URL.

URL obfuscation. Hackers often try to spread malicious links in an obfuscated form, e.g., by shortening it with a URL shortening service such as *bit.ly* or *goo.gl*. We store a binary feature with every URL that indicates whether the URL has been shortened or not; we maintain a list of URL shorteners.

Note that none of the above features by themselves are conclusive evidence of social malware, and other features could potentially further enhance the classifier (e.g., we can account for spam keywords such as 'free' included in URLs such as `http://nfljerseyfree.com`). However, as we show later in our evaluation, the features that we currently consider yield high classification accuracy in combination.

2.2.4 Implementing MyPageKeeper

We provide some details on MyPageKeeper's implementation.

Facebook application. First, we implement the MyPageKeeper Facebook application using FBML [41]. We implement our application server using Apache (web server), Django (web framework), and Postgres (database). Once a user installs the MyPageKeeper app in her profile, Facebook generates a secret access token and forwards the token to our application server, which we then save in a database. This token is used by the crawler to crawl the walls and news feeds of subscribed users using the Facebook open-graph API. If any user deactivates MyPageKeeper from their profile, Facebook disables this token and notifies our application server, whereupon we stop crawling that user's wall and news feed.

Crawler instances and frequency. We run a set of crawlers in Amazon EC2 instances to periodically crawl the walls and news feeds of MyPageKeeper's users. The set of users are partitioned across the crawlers. In our current instantiation, we run one crawler process for every 1,000 users. Thus, as more users subscribe to MyPageKeeper, we can easily scale the task of crawling their walls and news feeds by instantiating more EC2 instances for the task. Our Python-based crawlers use the open-graph API, incorporating users' secret access tokens, to crawl posts from Facebook. Once the data is received in JSON format, the crawlers parse the data and save it in a local Postgres database.

Currently, as a tradeoff between timeliness of detection and resource costs on EC2, we instantiate MyPageKeeper to crawl every user's wall and news feed once every two hours. Every couple of hours, all of our crawlers start up and each crawler fetches new posts that were previously not seen for the users assigned to it. Once all crawlers complete execution, the data from their local databases is migrated to a central database.

Checker instances. Checker modules are used to classify every post as social malware or benign. Every two hours, the central scheduler forks an appropriate number of checker modules determined by the number of new URLs crawled since the last round of checking. Thus, the identification of social malware is also scalable since each checker module runs on a subset of the pool of URLs. Each checker evaluates the URLs it receives as input—using a combination of whitelists, blacklists, and a classifier—and saves the results in a database. We use the libsvm [91] library for SVM based classification. Once all checker modules complete execution, notifiers are invoked to notify all users who have posts either on their wall or in their news feed that contain URLs that have been flagged as social malware.

Data	Total	# distinct URLs
MyPageKeeper users	12,456	-
Friends of MyPageKeeper users	2,370,272	-
News feed posts	38,764,575	29,522,732
Wall posts	1,760,737	1,532,055
User reports	679	333

Table 2.3: Summary of MyPageKeeper data.

Feature	F-Score
URL obfuscated?	0.300378
Spam keyword score	0.262220
# of news feed posts	0.173836
Message similarity score	0.131733
# of Likes	0.039895
# of wall posts	0.019857
# of comments	0.006367

Table 2.4: Feature scores used by MyPageKeeper's classifier.

2.3 Evaluation

Next, we evaluate MyPageKeeper from three perspectives. First, we evaluate the accuracy with which it classifies social malware. Second, we determine the contribution of MyPageKeeper's social context based classifier in identifying social malware compared to the URL blacklists that it uses. Lastly, we compare MyPageKeeper's efficiency with alternative approaches that would either crawl every URL or at least resolve short URLs in order to identify social malware.

Table 2.3 summarizes the dataset of Facebook posts on which we conduct our evaluation. This data is obtained during MyPageKeeper's operation over a four month period from 20th June to 19th October, 2011. MyPageKeeper had over 12K users during this period, who had around 2.37M friends in union. Our data comprises 38.7 million

Alternative source	**# of posts**
Flagged by blacklist	18,923
Flagged by on.fb.me	2,102
Content deleted by Facebook	3,918
Blacklisted app	1,290
Blacklisted IP	5,827
Domain is deleted	247
Points to app install	4,658
Spamming app	6,547
Manually verified	14,876
True positives	58,388 (97%)
Unknown	1,803 (3%)
Total	60,191

Table 2.5: Validation of social malware flagged by MyPageKeeper classifier.

App name	**Description**	**# of posts**
Sendible	Social Media Management	6,687
iRazoo	Search & win!	1,853
4Loot	4Loot lets you win all sorts of Loot while searching the web	1,891

Table 2.6: Top three spamming applications in our dataset.

and 1.7 million posts that contain URLs from the news feeds and walls of these 12K users. We consider only those posts that contain URLs since MyPageKeeper currently checks only such posts. Overall, these 40 million posts contained around 30 million unique URLs. In addition, we received 679 reports of social malware from 533 distinct MyPageKeeper users during the four month period, with 333 distinct URLs across these reports. Though it is hard to make any general claims with regard to representativeness of our data, we find that several user metrics (e.g., the male-to-female ratio and the distribution of users across age groups) closely match that of the Facebook user base at-large.

2.3.1 Accuracy

As previously mentioned, MyPageKeeper first matches every URL to be checked against a whitelist. If no match is found, it checks the URL with a set of locally queriable URL blacklists. Finally, MyPageKeeper applies its social context based classifier learned using the SVM model. In this process, we assume URL information provided by whitelists and blacklists to be ground truth, i.e., classification provided by them need not be independently validated. Therefore, we focus here on validating the social malware flagged by MyPageKeeper's classifier based on social context features.

We trained MyPageKeeper's classifier using a manually verified dataset of URLs that contain 2,500 positive samples and 5,000 negative samples of social malware posts; we gathered these samples over several months while developing MyPageKeeper. Table 2.4 shows the importance of the various features in the SVM classifier learned. During the course of MyPageKeeper's operation over four months, we applied the classifier to check 753,516 unique URLs; these are URLs that do not match the whitelist or any of the blacklists. Of these URLs, the classifier identified 4,972 URLs, seen across 60,191 posts, as instances of social malware.

It is important to note that when MyPageKeeper sees a URL in multiple posts over time, the values of the features associated with the URL may change every time it appears, e.g., the message similarity score associated with the URL can change. However, once MyPageKeeper classifies a URL as social malware during any of its occurrences, it flags all previously seen posts that contain the URL and notifies the corresponding users. Therefore, in evaluating MyPageKeeper's classifier, URL blacklists, or MyPageKeeper as a whole, we consider here that a technique classified a particular URL as social malware if that URL was flagged by that technique upon any of the URL's occurrences. Correspondingly, we consider a URL to have not been classified as social malware if it was not identified as such during any of its occurrences.

Checking the validity of social malware identified by MyPageKeeper's classifier is not straightforward, since there is no ground truth for what represents social malware and what does not. However, here we attempt to evaluate the positive samples of social malware identified by MyPageKeeper's classifier using a combination of a host of complementary techniques (we later discuss in Section 2.6 the validation of posts that are deemed safe by MyPageKeeper). To do so, we use an instrumented Firefox browser to crawl the 4,972 URLs flagged by MyPageKeeper at the end of the four month period of MyPageKeeper operation. For every URL that we crawl, we record the landing URL, the IP address and other whois information of the landing domain, and contents of the landing page. To verify the reputation of every URL, we then apply several techniques in the order summarized in Table 2.5.

- *Blacklisted URLs:* first, we check if any of the URLs or the corresponding landing URLs are found in any URL blacklists. Note that, though we use blacklists in the operation of MyPageKeeper itself, we use only those that can be stored and queried locally. Therefore, here we use for validation other external blacklists for which we have to issue remote queries. Further, even for blacklists used in MyPageKeeper, they may not identify some instances of social malware when they initially appear because blacklists have been found to lag in keeping up with the viral propagation of spam on OSNs [99]. Hence, we check if a URL identified as social malware by MyPageKeeper's classifier appeared in any of the blacklists used by MyPageKeeper at a later point in time, even though it did not appear in any of those blacklists initially when MyPageKeeper spotted posts containing that URL.

- *Flagged by fb.me URL shortener:* many URLs posted on Facebook are shortened using Facebook's URL shortener `fb.me`. When Facebook determines any link shortened using their service to be unsafe, the corresponding shortened URL thereafter redirects to Facebook's home page—`facebook.com/home.php`—instead of the actual landing page. Of the URLs flagged by MyPageKeeper's classifier, we check if those shortened using Facebook's URL shortening service redirect to Facebook's home page.
- *Content deleted from Facebook:* if Facebook determines any URL hosted under the `facebook.com` domain to be unsafe (e.g., the page for a spamming Facebook application), it thereafter redirects that URL to `facebook.com/4oh4.php`. We use this as another source of information to validate URLs flagged by MyPageKeeper's classifier.
- *Blacklisted apps:* if the URLs in posts made by a Facebook app are flagged due to any of the above reasons, we consider that app to be malicious and declare all other URLs posted by it as unsafe, thus helping validate some of the URLs declared as social malware by MyPageKeeper's classifier.
- *Blacklisted IPs:* for every URL flagged by any of the above techniques, we record the IP address when that URL is crawled and blacklist that IP. Of the URLs flagged by MyPageKeeper's classifier, we then consider those that lead to one of these blacklisted IP addresses as correctly classified.
- *Domain deleted:* malicious domains are often deleted once they are caught serving malicious content. Therefore, we deem MyPageKeeper's positive classification of a URL to be correct if the domain for that URL no longer exists when we attempt to crawl it.
- *Obfuscation of app installation page:* posts made by Facebook applications to attract

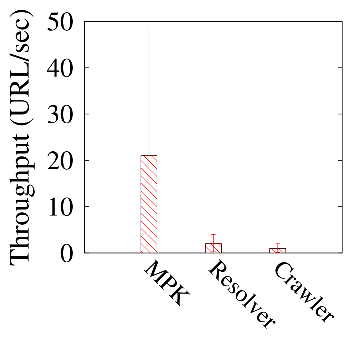
users to install them typically include an un-shortened URL pointing to a Facebook page that contains information about the application. Once a user visits this page, she can read the application's description and then click on a link on this page if she decides to install it. However, posts from some surreptitious applications contained shortened URLs that directly take the user to a page where they request the user to grant permissions (e.g., to post on the user's wall) and install the application. We have found all instances of such applications to be spamming applications. Therefore, if any of the URLs flagged by MyPageKeeper's classifier is a shortened URL that directly points to the installation page for a Facebook app, we declare that classification correct.

- *Spamming app:* from our dataset, we manually identified several Facebook applications that try to spread on Facebook by promising free money to users and make posts that point to the application page. Once installed by a user, such applications periodically post on the user's wall (without requesting the user's authorization for each post) in an attempt to further propagate by attracting that user's friends; Table 2.6 shows some such applications that frequently appear in our dataset. Any URLs classified as social malware by MyPageKeeper's classifier that happen to be posted by one of these manually identified spamming apps are deemed correct.
- *Manual analysis*: finally, over the operation of MyPageKeeper during the four months, we periodically verified a subset of URLs flagged by the classifier. These provide an additional source of validation.

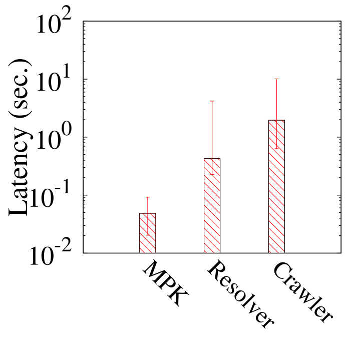
In all, the union of the above techniques validates that 58,388 out of 60,191 posts declared as social malware by the MyPageKeeper classifier are indeed so. Therefore, 97% of the social malware identified by MyPageKeeper's classifier are true positives. On the other hand, the 1,803 posts incorrectly classified as social malware constitute

Source	**# (%) of URLs**	**# (%) of posts**	**Overlap with classifier (# of URLs)**
Google SBA2	221 (6.8%)	378 (0.4%)	0
Phishtank	12 (0.4%)	435 (0.5%)	1
Malware Norm	69 (2.1%)	154 (0.2%)	0
Joewein	240 (7.4%)	652 (0.7%)	11
APWG	56 (1.7%)	569 (0.6%)	0
Spamcop	232 (7.1%)	921 (1.0%)	0
All blacklists	830 (25.6%)	3104 (3.4%)	12
MyPageKeeper classifier	2405 (74.4%)	89389 (96.6%)	

Table 2.7: Comparison of contribution made by blacklists and classifier to MyPageKeeper's identification of social malware during the four month period of operation.



(a) Throughput



(b) Latency

Figure 2.3: Comparison of MyPageKeeper's throughput and latency in classifying URLs with a short URL resolver and a crawler-based approach. The height of the box shows the median, with the whiskers representing 5^{th} and 95^{th} percentiles.

less than 0.005% of the over 40 million posts in our dataset. Note that, though all of the above techniques could be folded into MyPageKeeper itself to help identify social malware, we do not do so because all of these techniques require us to crawl a URL in order to evaluate it; we cannot afford the latency of crawling.

2.3.2 Comparison with Blacklists

Though we see that the identification of social malware by MyPageKeeper's classifier is accurate, the next logical question is: what is the classifier's contribution to MyPageKeeper in comparison with URL blacklists? Table 2.7 provides a breakdown of the URLs and posts classified as social malware by MyPageKeeper during the four month period under consideration. There are two main takeaways from this table. First, we see that the classifier finds 74.4% of social malware URLs and 96.6% of social malware posts identified by MyPageKeeper. Thus, the classifier accounts for a large majority of social malware identified by MyPageKeeper and is thus critical to the system's operation. Second, there is very little overlap between the URLs flagged by blacklists and those flagged by the classifier. The typically low frequency of occurrence of URLs that match blacklists is another reason that the classifier's share of identified social malware posts is significantly greater than its corresponding share of flagged URLs.

2.3.3 Efficiency

Beyond accuracy, it is critical that MyPageKeeper's identification of social malware be efficient, so as to minimize the costs that we need to bear in order to keep the delay in identifying social malware and alerting users low. The matching of a URL against a whitelist or a local set of blacklists incurs minimal computational overhead. In addition, we find that execution of the classifier also imposes minimal delay per URL verified.

To demonstrate the efficiency of MyPageKeeper, we compare the rate at which it classifies URLs with the classification throughput that two other alternative classes of approaches would be able to sustain. Our first point of comparison is an approach that relies only on locally queriable URL whitelists and blacklists but resolves all shortened URLs into the corresponding complete URL. Our second alternative crawls URLs to evaluate them, e.g., using the content on the page or the IP address of the target website. Figure 2.3(a) compares the throughput of classifying URLs with the three approaches, using data from two weeks of MyPageKeeper's execution. We see that the throughput with MyPageKeeper is almost an order of magnitude greater than the alternatives, with all three approaches using the same set of resources on EC2. As we see in Figure 2.3(b), MyPageKeeper's better performance stems from its lower execution latency to check an URL; the median classification latency with MyPageKeeper is 48 ms compared to a median of 426 ms when resolving short URLs and 1.9 seconds when crawling URLs. Thus, we are able to significantly reduce MyPageKeeper's classification latency, compared to approaches that need to resolve short URLs or crawl target web pages, by keeping all of its computation local.

Furthermore, a crawler-based approach will be significantly more expensive than MyPageKeeper. Thomas et al. [123] found that crawler-based classification of 15 million URLs per day using cloud infrastructure results in an expense of \$800/day. Therefore, we estimate that it would cost approximately \$1.5 million/year to handle Facebook's workload; 1 million URLs are shared every 20 minutes on Facebook [80]. Since MyPageKeeper's classification latency is 40 times less than a crawler-based approach, we estimate that the expense incurred with MyPageKeeper would be at least 40 times lower than a system that classifies URLs by crawling them.

2.4 Analysis of Social Malware

Thus far we described how MyPageKeeper detects social malware efficiently at scale. In this section, we analyze the social malware that we have found during MyPageKeeper's

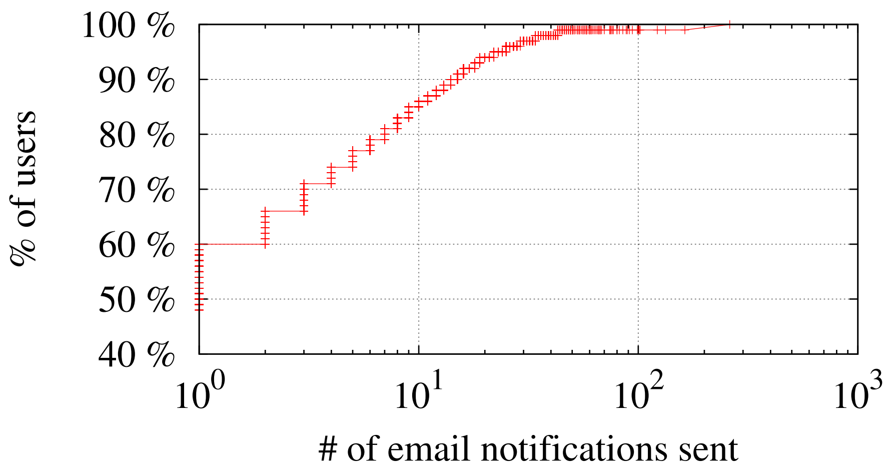


Figure 2.4: 49% of MyPageKeeper's 12,456 users were notified of social malware at least once in four months of MyPageKeeper's operation.

operation to throw light on characteristics of social malware on Facebook.

2.4.1 Prevalence of Social Malware

49% of MyPageKeeper's users were exposed to social malware within four months. First, we analyze the prevalence of social malware on Facebook. To do so, we define that a user was exposed to a particular social malware post if that post appeared in her wall or news feed. As shown in Figure 2.4, 49% of MyPageKeeper's users were exposed to at least one social malware post during the four month period we consider here. Though this already indicates the wide reach of social malware on Facebook, we stress that 49% is only a lower-bound due to a couple of reasons. First, many of MyPageKeeper's users subscribed to our application at some time in the midst of the four month period and therefore, we miss social malware that they were potentially exposed to prior to them subscribing to MyPageKeeper. Second, Facebook itself detects and removes posts that it considers as spam or pointing to malware [84, 86, 118]. All the social malware detected by MyPageKeeper is after such filtering by Facebook.

Given that some users are exposed to more social malware than others, we

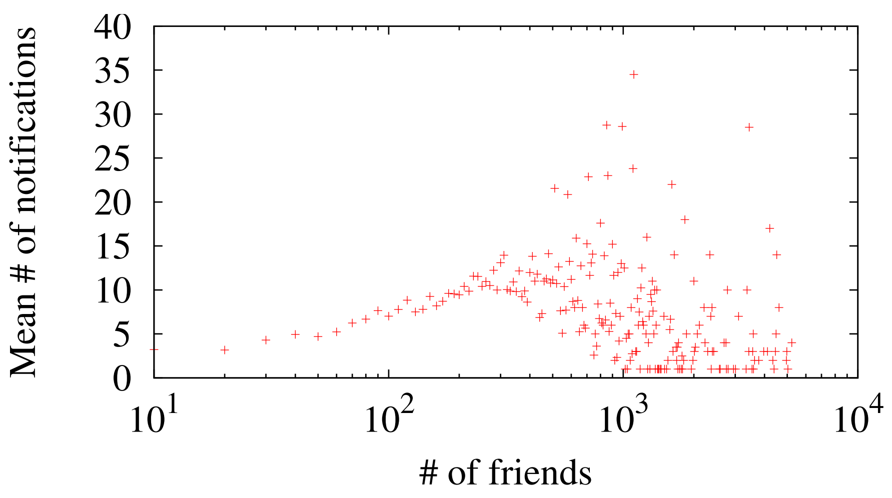


Figure 2.5: Correlation between vulnerability and social degree of exposed users.

analyze if the social degree of a user has any impact on the probability of a user being exposed to social malware. Figure 2.5 shows the number of social malware notifications received by MyPageKeeper users as a function of the number of friends they have on Facebook. We bin users with the number of friends within 10 of each other and plot the average number of notifications per bin; we consider here only those users who were subscribed to MyPageKeeper for at least three months. We see that the probability of users being exposed to social malware is largely independent of their social degree. This indicates that whether a user is more likely to be exposed to social malware is not simply a function of how many friends she has, but likely depends on the susceptibility of those friends to becoming victims of scams and helping propagate them.

We also find that social malware on Facebook is prevalent over time. Figure 2.6 shows the number of social malware notifications sent per day by MyPageKeeper to its users. We see a consistently large number of notifications going out daily, with noticeable spikes on a few days. On 11th July 2011, a scam that conned users to complete surveys with the pretext of fake free products went viral and posts pointing to the scam

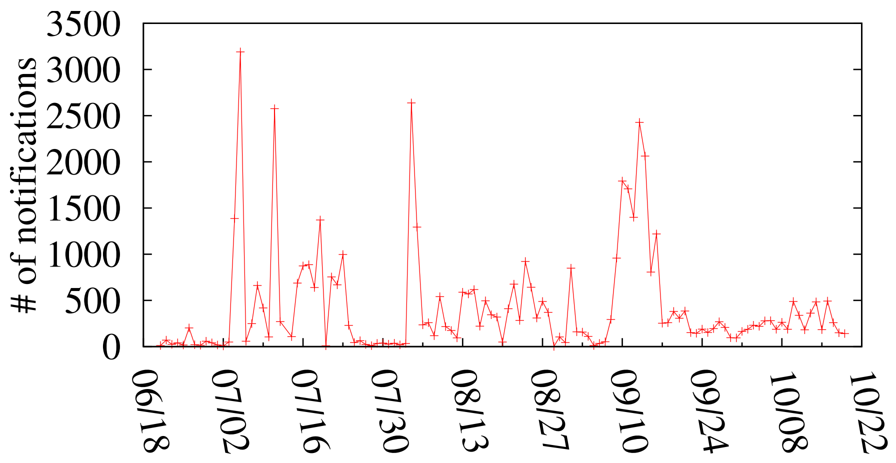


Figure 2.6: No. of social malware notifications per day. On 11th July, 19th Sep, and 3rd Oct, social malware was observed in large scale.

appeared 4,056 times on the walls and news feeds of MyPageKeeper's users. Two other scams, that promised *'Free Facebook shoes'* and conned users to fill out surveys, also caused MyPageKeeper to send out a large number of notifications on that day. On 19^{th} Sep. 2011, different variants of the *'Facebook Free T-Shirt'* scam [21] were spreading on Facebook and was spotted 2,040 times by MyPageKeeper. On 3^{rd} Oct. 2011, a video scam was spreading on Facebook and MyPageKeeper observed it in 1,739 posts.

We next analyze the prevalence and impact of social malware from the perspective of individual social malware links. For each link, we define its "active-time" as the difference between the first and last times of its occurrence in our dataset. Figure 2.7 shows that we did not see 60% of social malware links beyond one day. Subsequent posts containing these links may have been filtered by Facebook once it recognized their spammy or malicious nature, or our dataset may miss those posts due to MyPageKeeper's limited view into Facebook's 850 million users. Further, we do not attempt any clustering of links into campaigns here. However, even with these caveats, 20% of social malware links were seen in multiple posts separated by at least 10 days, suggest-

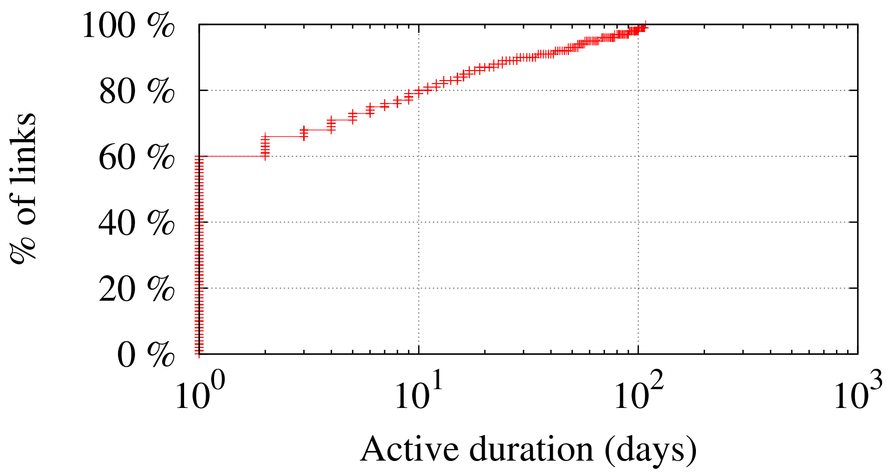


Figure 2.7: Active-time of social malware links. 20% of social malware links were observed more than 10 days apart.

Shortening service	**% of social malware URLs**
bit.ly	21.9%
tinyurl.com	18.8%
goo.gl	5.1%
t.co	3.16%
tiny.cc	1.6%
ow.ly	1.1%
on.fb.me	1.0%
is.gd	0.7%
j.mp	0.4%
0rz.com	0.3%
All shortened URLs	54%

Table 2.8: Top URL shortening services in our social malware dataset.

ing that a significant fraction of social malware eludes Facebook's detection mechanisms and lasts on Facebook for significant durations.

2.4.2 Domain Name Characteristics

20% of social malware links are hosted inside Facebook. In the next section of our analysis, we focus on the domain-level characteristics of social malware links. First, Table 2.8 shows the top ten URL shortening services used in social malware

Domain Name	% of URLs	% of posts
facebook.com	20.7%	26.3%
blogspot.com	6.3%	8.7%
miessass.info	1.9%	3.2%
shurulburul.tk	1.8%	1.2%
tomoday.info	0.8%	0.13%

Table 2.9: Top two-level domains in our social malware dataset.

links observed by MyPageKeeper. In all, shortened URLs account for 54% of social malware links in our dataset. Our design of MyPageKeeper's classifier to rely solely on social context, and to not resolve short URLs, hence makes a significant difference (as previously seen in the comparison of classification latency).

Further, we find it surprising that a large fraction of social malware links (46%) are not shortened, given that shortening of URLs enables spammers to obfuscate them. On further investigation, we find that many Facebook scams such as *'free iPhone'* and *'free NFL jersey'* use domain names that clearly state the message of the scam, e.g., `http://iphonefree5.com/` and `http://nfljerseyfree.com/`. These URLs are more likely to elicit higher click-through rates compared to shortened URLs. On the other hand, most of the shortened URLs were used by malicious or spam applications (e.g., 'The App', 'Profile Stalker') that generate shortened URLs pointing to their application's installation page. We find that 89% of shortened URLs in our dataset of social malware links were posted by Facebook applications.

Next, based on our crawl of the social malware links in our dataset, we inspect the top two-level domains found on the landing pages pointed to by these links. First, as shown in Table 2.9, we find that a large fraction of social malware (over 20% of URLs and 26% of posts) is hosted on Facebook itself. Second, a sizeable fraction of social malware uses sites such as `blogspot.com` and `wordpress.com` that enable the

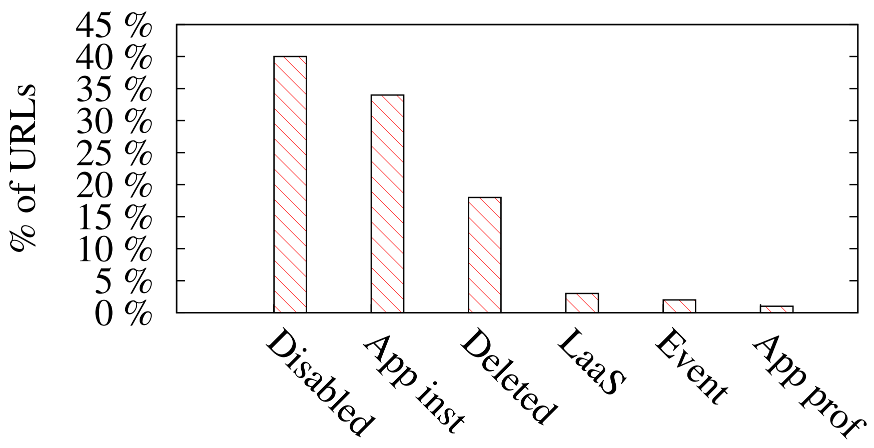


Figure 2.8: Breakdown of social malware links, when crawled in Nov. 2011, that originally point to web pages in the `facebook.com` domain.

spammers to easily create a large number of URLs without going through the hassle of registering new domains. Further, all of these domains are of good repute and are unlikely to be flagged by traditional website blacklists.

2.4.3 Analysis of Social Malware Hosted in Facebook

Hackers use numerous channels in Facebook to spread social malware. Given the large fraction of social malware hosted on Facebook itself, we next analyze this subset of social malware. First, in early November 2011, we crawled every social malware link in our dataset that had pointed to a landing page in the `facebook.com` domain at the time when MyPageKeeper had initially classified that link as social malware. Figure 2.8 presents a breakdown of the results of this crawl. If Facebook disables a URL, it redirects us to `facebook.com/home.php`. Similarly, if crawling a URL points us to `facebook.com/4oh4`, it implies that Facebook has deleted the content at that URL. Therefore, as seen in Figure 2.8, a large fraction of social malware links that were originally pointing to Facebook have now been deactivated. However, we also see that

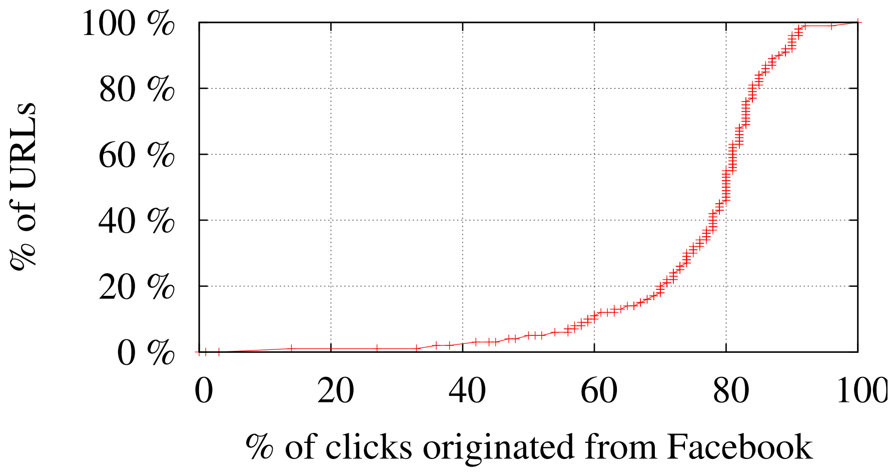


Figure 2.9: For most social malware links shortened with bit.ly or goo.gl, a large majority of the clicks came from Facebook.

a significant fraction of these links—over 40%—were still live. Further, the figure shows that spammers use several different channels, such as applications, events, and pages to propagate their scams on Facebook. In the figure, *'App inst'* and *'App prof'* refer to the installation and profile pages of Facebook applications, and *'LaaS'* refers to campaigns intended to increase the number of Likes on a Facebook page (described in detail in Section 2.5).

In our dataset, we see 257 distinct social malware links shortened with the `bit.ly` and `goo.gl` URL shorteners that point to landing pages in the `facebook.com` domain. Using the APIs [14, 46] offered by these URL shortening services, we computed the number of clicks recorded for these 257 links in two cases—1) where the Referrer was Facebook, and 2) where the Referrer was any other domain. Figure 2.9 shows that Facebook is the dominant platform from which most of these links received most of their clicks; 80% of links received over 70% of their clicks from Facebook. This seems to indicate that most social malware hosted on Facebook is propagated solely on Facebook and tailored for that platform.

Social malware word	**Likelihood ratio**	**Spam email word**	**Likelihood ratio**
free	12.1	money	11.5
< 3	∞	price	26.6
iphone	∞	free	0.08
awesome	31.3	account	9.6
win	24.3	stock	9.7
wow	90.8	address	5.2
hurry	36.8	bank	56.4
omg	332.3	pills	∞
amazing	4.9	viagra	∞
deal	1.9	watch	1.9

Table 2.10: Top keywords from social malware posts and spam emails.

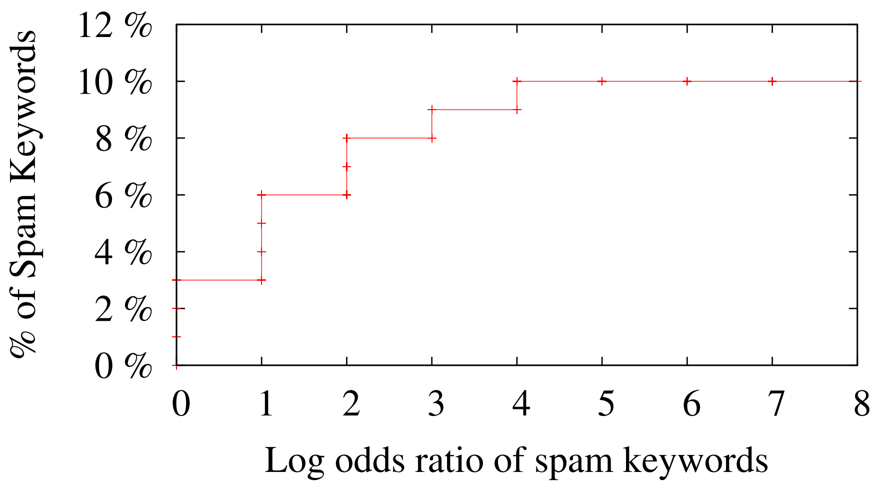


Figure 2.10: Overlap of keywords between email and Facebook.

2.4.4 Comparison of Social Malware to Email Spam

Social malware keywords exhibit little (10%) overlap with spam email keywords. As we saw earlier in Section 2.3, spam keyword score is a key feature in MyPageKeeper's classifier. Therefore, in the final section of our analysis, we investigate the overlap in 'spam keywords' that we observe in social malware on Facebook with those seen in another medium targeted by spammers, specifically email. We investigate whether spammers use similar keywords on Facebook as they use in email spam.

To perform this analysis, we collected over 17,000 spam emails from [111]. For Facebook spam, we use 92,493 social malware posts collected by MyPageKeeper. We transform posts in either dataset to a bag of words with their frequency of occurrence. Similar to [123], we then compute the log odds ratio for each keyword to determine its overlap in Facebook social malware and spam email. Here, the log odds ratio for a keyword is defined by $|log(p_1q_2/p_2q_1)|$ where p_i is the likelihood of that keyword appearing in set i and $q_i = 1 - p_i$. A value of 0 for the log odds ratio indicates that the keyword is equally likely to appear in both datasets, whereas an infinite ratio indicates that the keyword appears in only one of the datasets. In Figure 2.10 (infinite values are omitted), we see only a 10% overlap in spam keywords between email and Facebook. This indicates that Facebook spam significantly differs from traditional email spam.

Further, Table 2.10 shows the likelihood ratio (defined earlier in Section 2.2.3) for the top keywords in either dataset. The higher the likelihood ratio of a social malware keyword, the stronger the bias of the keyword appearing more in Facebook social malware than in email spam; an infinite ratio implies the keyword exclusively appears in Facebook social malware. The word *'omg'* is 332 times more likely to be used in Facebook social malware than in email spam. On the other hand, words such as *'pills'* and *'viagra'* are restricted solely to email spam.

2.5 Like-as-a-Service

Facebook has now become the premier online destination on the Internet. Over 900 million users, half of whom visit the site daily, spend over 4 hours on the site every

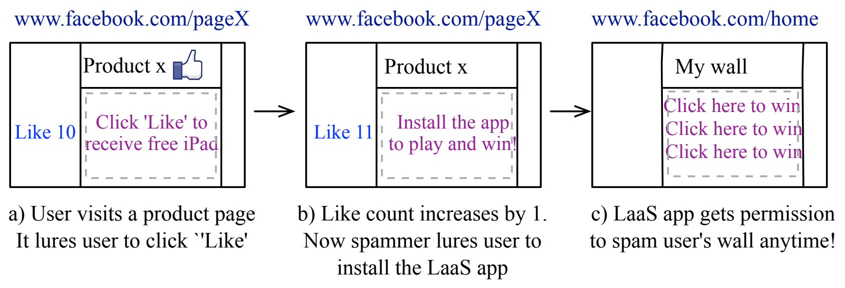

www.facebook.com/pageX
Product x
Like 10
Click 'Like' to receive free iPad
a) User visits a product page It lures user to click ``Like'
www.facebook.com/pageX
Product x
Like 11
Install the app to play and win!
b) Like count increases by 1. Now spammer lures user to install the LaaS app
www.facebook.com/home
My wall
Click here to win
Click here to win
Click here to win
c) LaaS app gets permission to spam user's wall anytime!

Figure 2.11: A representation of how a Like-as-a-service Facebook application collects Likes for its client's page and gains access to the user's wall for spamming. Dotted region of the page is controlled by the spammer.

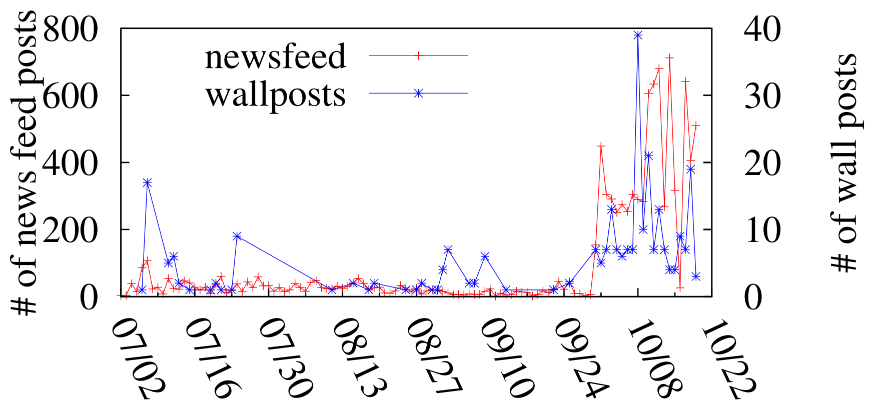


Figure 2.12: Timeline of posts made by the Games LaaS Facebook application seen on users' walls and news feeds.

month [24]. To leverage user activity on Facebook, an increasingly large number of businesses have Facebook pages associated with their products. However, attracting users to their page is a challenge for any business. One way of doing so is to make users who visit a Facebook page click the 'Like' button on the page. A large number of Likes has two significant implications. First, the number of Likes associated with a page has begun to represent the reputation associated with a page, e.g., a higher number of Likes improves the page's rank in Bing [12]. Second, a link to the product page appears in the news feed of the friends of the user who clicked Like on the page, thus enabling the

Page Name	**Application Message**	**No. of Likes**
Raging Bid	Just got a better score on Raging Bid's Bouncing Balls contest and I am now in 12297th place. I am getting closer to winning a Sony Bravia 3D HDTV. Who thinks they can beat my score? Click here to try: URL	168,815
www.WalkerToyota.com	DAILY CONTEST UPDATE: I am currently in 7573rd place in Walker Toyota's Tetris contest. There is still plenty of time to try and win a 16GB iPad2. Who thinks they can get a better score than me? Click here to try: URL	136,212
Chip Banks Chevrolet Buick	DAILY CONTEST UPDATE: I am currently in 310th place in Chip Banks Chevrolet Buick's Gem Swap II contest. There is still plenty of time to try and win a 16GB iPad2. Who thinks they can get a better score than me? Click here to try: URL	2,190
Casey Jamerson	DAILY CONTEST UPDATE: I am currently in 6234th place in Casey Jamerson Music's Gem Swap II contest. There is still plenty of time to try and win a 16GB iPad2. Who thinks they can get a better score than me? Click here to try: URL	47,496
Tara Gray	DAILY CONTEST UPDATE: I am currently in 10213th place in Tara Gray's Gem Swap II contest. There is still plenty of time to try and win a Burma Ruby Ring. Who thinks they can get a better score than me? Click here to try: URL	231,035

Table 2.11: Five example Facebook pages integrated with the Games LaaS application to spam users' walls for propagation.

link to the page to spread on Facebook.

Based on our view of Facebook social malware through the MyPageKeeper lens, we see an emerging Like-as-a-Service [1] market to help businesses attract users to their pages. We identify several Facebook apps (e.g., 'Games' [45], 'FanOffer' [39], and 'Latest Promotions' [53]) which are hired by the owners of Facebook pages to help increase the number of Likes on their pages. These applications, which offer Likes as

[1] Note that 'Like-as-a-Service' differs from 'Likejacking' [54], where users are tricked into clicking the Like button without them realizing they are doing so, e.g., by enticing the user to click on a Flash video, within which the Like button is hidden.

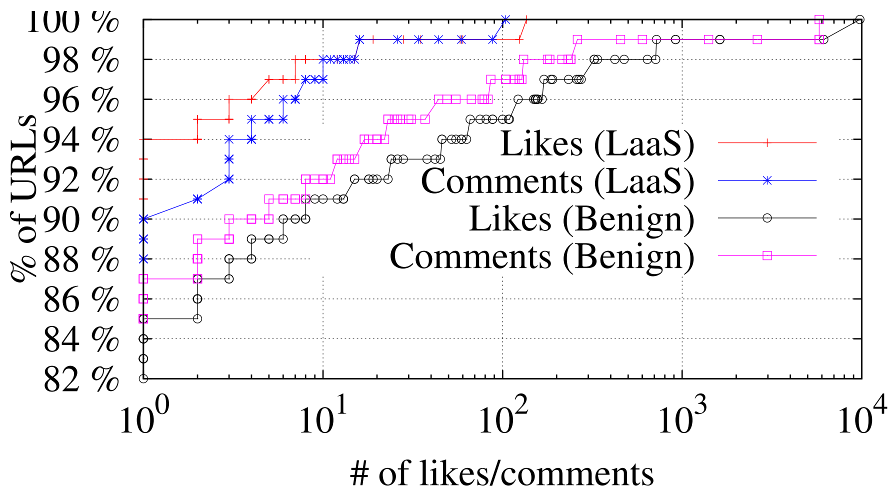


Figure 2.13: # of Likes and comments associated with URLs posted by the Games Facebook app.

a service, presumably get paid on a 'Pay-per-Like' model by the owners of Facebook pages that make use of their services.

Figure 2.11 shows how a Like-as-a-Service (LaaS) application typically works. First, a customer of the LaaS application integrates the application into their Facebook page. When users visit the page, the LaaS application entices the user to click Like on page. Typically, the reward promised to the user in return for his Like is that the user can play some games on the page or have a chance of winning free products. However, once the user clicks Like on the page to access the promised reward, the LaaS application then demands that the user add the application to his profile in order to proceed further. In the process of getting the user to add the LaaS application, the application requests the user to grant permission for it to post on the user's wall. Once the application obtains such permissions, it periodically spams the user's wall with posts that contain links to the Facebook page of the customer who enrolled the LaaS application for its services. These posts will appear in the news feeds of the unsuspecting user's friends, who in turn may visit the Facebook page and go through the same cycle again. The

LaaS application thus enables the Facebook pages of its customers to accumulate Likes and increase their reputation, even though users are clicking Like on these pages with the promise of false rewards rather than because they like the products advertised on the page.

Here, we analyze the activity of one such LaaS application—Games [45]. Figure 2.12 shows that posts made by this application appear regularly in the walls and news feeds of MyPageKeeper's users. Even with our small sample of roughly 12K users from Facebook's total population of over 850 million users, we see that 40 users have posts made by Games on their walls, which implies that these users have installed the application and granted it permission to make posts on their wall at any time. We also see that the number of users who installed Games significantly rose around mid-September 2011. Further, from the news feeds of MyPageKeeper users, we see that Games posted links to as many as 700 Facebook pages on a single day; each link points to the Facebook page of a different customer of this LaaS application. Table 2.11 shows the posts made by Games for some of its customers, the variation in text messages across these posts, and the large number of Likes garnered by the Facebook pages of these customers.

We next analyze the Likes and comments received by 721 URLs posted by the Games app. As shown in Figure 2.13, we see that over 95% of these URLs have less than 100 Likes and less than 100 comments; this fraction is significantly lesser on a dataset of randomly chosen 721 URLs from benign posts. However, over 20% of the URLs posted by the Games app do receive Likes and comments, thus enabling them to propagate on Facebook. Real users may be unknowingly helping to spreading spam in these cases; such users have been previously referred to as creepers [118].

2.6 Discussion

Client-based solution. An alternative to MyPageKeeper's server-side detection of social malware would be to identify social malware on client machines. In such an approach, a client-side tool can classify a post at the instant when the user accesses the post. However, we choose not to use such an approach for multiple reasons. First, a server-side solution is more amenable to adoption; it is easier to convince users to add an app to their Facebook profile than to convince them to download and install an application or browser extension on their machines. Second, users can access Facebook from a range of browsers and even from different device types (e.g., mobile phones). Developing and maintaining client-side tools for all of these platforms is onerous. Finally, and most importantly, many of the features used by our social malware classifier (e.g., message similarity score) fundamentally depend on aggregating information across users. Therefore, a view of Facebook from the perspective of a single client may be insufficient to identify social malware accurately.

Estimating false negatives. While we evaluated the accuracy of social malware identified by MyPageKeeper by cross-validating with other techniques, evaluating the accuracy of MyPageKeeper's classifier in cases where it declares a URL safe is much harder. Not only do we lack ground truth, but since the highly common case is that a Facebook post is benign, manual verification of a randomly chosen subset of the classifier's negative outputs is insufficient.

We therefore evaluate whether MyPageKeeper's classifier misses any social malware by using data from user-reported samples of social malware. As shown in Table 2.3, 533 distinct MyPageKeeper users have submitted 679 such reports and we have received 333 unique URLs across these reports. Based on manual verification, we find that 296 of these 333 URLs indeed point to spam or malware. The remaining 37 URLs point to sites like `surveymonkey.com` (fill out surveys) and `clixsense.com` (get paid to view advertisements), which though abused by spammers have legitimate uses as well. We suspect that our users did come across social malware, but reported the URL of the landing page, rather than the URL that they originally found in a social malware post.

Of the 296 instances of true social malware reported by users, MyPageKeeper's classifier flagged all but 17 of them, independently of users reporting them to us. This translates into a false negative rate of 5% for the classifier. However, 16 of these 17 URLs had been found to match against one of the URL blacklists used by MyPageKeeper. Thus, the false negative rate for the whole MyPageKeeper system, which combines blacklists and the classifier to detect social malware, is 0.3%.

Arms race with spammers. Though our current techniques seem to suffice to accurately identify social malware on Facebook, we speculate here on how spammers may evolve social malware, given the knowledge of how MyPageKeeper works. One option for spammers to evade MyPageKeeper is to use different shortened URLs for a single malicious landing URL. In such cases, MyPageKeeper would consider every posted shortened URL seperately even though they are all part of the same campaign. Thus, if any of these shortened URLs does not appear on the walls/news feeds of several users, MyPageKeeper may fail to flag it. Another option for social malware to evade MyPageKeeper is for spammers to slow down its rate of propagation; as we found in Section 2.3.2, MyPageKeeper sometimes misses social malware which is observed only a few times in our dataset. However, slowing down a social malware epidemic makes it likely that it will be flagged by other techniques, such as URL blacklists. Moreover, spammers may often be unable to control how fast a social malware epidemic spreads. In the case where an epidemic spreads by luring users into installing a Facebook app, the spammer can control how often the app posts spam on the user's wall. However, in cases where users are asked to 'Like' or 'Share' a post to access a fake reward, the social malware is self-propagating and its viral spread cannot be controlled by spammers.

Another option is for spammers to change the keywords that they use in social malware posts, thus affecting the spam keyword score used by MyPageKeeper's classifier. Though spammers are constrained in their choice of keywords by the need to attract users, some of the keywords may evolve over time as popular colloquial expressions (e.g., 'OMG') change. To evaluate MyPageKeeper's ability to cope with such change, we identified the top keywords (those with high likelihood ratio compared to benign posts among frequently occurring keywords) distinctive to user-reported social malware posts. We find that the spam keywords that we use in MyPageKeeper's classifier (identified from manually identified samples of social malware) match those computed here. Though this captures data only across four months, MyPageKeeper can similarly recompute the set of spam keywords over time.

2.7 Summary of Results

In this chapter, we present MyPageKeeper, an efficient and scalable approach to detect social malware in Facebook. Based on 12K Facebook users and their 2.2M Facebook friends and 40M posts over four months, we show the following key results in this chapter:

- We show that MyPageKeeper can accurately and efficiently identify social malware at scale. It's true positive rate is 97% and false positive rate is 0.005%. Further, it requires only 46 ms on average to classify a post.

- We found that the reach of social malware is widespread —49% of MyPageKeeper's 12K users were exposed at least once to social malware within four months, and that a significant fraction of social malware is hosted on Facebook itself.

- We also showed that existing defenses, such as URL blacklists, are ill-suited for identifying social malware, and that social malware significantly differs from email spam.

- We identified a new trend in aggressive marketing of Facebook pages using "Like-as-a-Service" applications that spam users to make money based on a "Pay-per-Like" model.

Chapter 3

Indentifying Malicious Applications

Online social networks (OSN) enable and encourage third party applications (apps) to enhance the user experience on these platforms. Such enhancements include interesting or entertaining ways of communicating among online friends, and diverse activities such as playing games or listening to songs. For example, Facebook provides developers an API [26] that facilitates app integration into the Facebook user-experience. There are 500K apps available on Facebook [83], and on average, 20M apps are installed every day [3]. Furthermore, many apps have acquired and maintain a large user-base. For instance, FarmVille and CityVille apps have 26.5M and 42.8M users to date.

Recently, hackers have started taking advantage of the popularity of this third-party apps platform and deploying malicious applications [63, 82, 77]. Malicious apps can provide a lucrative business for hackers, given the popularity of OSNs, with Facebook leading the way with 900M active users [35]. There are many ways that hackers can benefit from a malicious app: (a) the app can reach large numbers of users and their

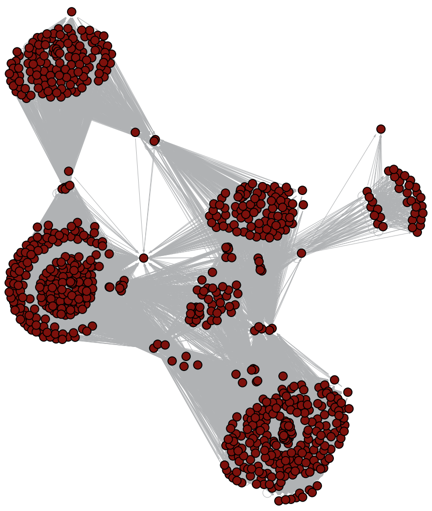


Figure 3.1: The emergence of AppNets on Facebook. Real snapshot of 770 highly collaborating apps: an edge between two apps means that one app helped the other propagate. Average degree (no. of collaborations) is 195!

friends to spread social malware, (b) the app can obtain users' personal information such as email address, home town, and gender, and (c) the app can "re-produce" by making other malicious apps popular. To make matters worse, the deployment of malicious apps is simplified by ready-to-use toolkits starting at $25 [48]. In other words, there is motive and opportunity, and as a result, there are many malicious apps spreading on Facebook every day [70].

Despite the above worrisome trends, today, a user has very limited information at the time of installing an app on Facebook. In other words, the problem is: given an app's identity number (the unique identifier assigned to the app by Facebook), can we detect if the app is malicious? Currently, there is no commercial service, publicly-available information, or research-based tool to advise a user about the risks of an app. As we show in Sec. 3.2, malicious apps are widespread and they easily spread, as an infected user jeopardizes the safety of all its friends.

So far, the research community has paid little attention to OSN apps specifically. Most research related to spam and malware on Facebook has focused on detecting malicious posts and social spam campaigns [115, 97, 96]. A recent work studies how app permissions and community ratings correlate to privacy risks of Facebook apps [92]. Finally, there are some community-based feedback-driven efforts to rank applications, such as Whatapp [81]; though these could be very powerful in the future, so far they have received little adoption. We discuss previous work in more detail in Chapter 5.2.

In this work, we develop FRAppE, a suite of efficient classification techniques for identifying whether an app is malicious or not. To build FRAppE, we use data from MyPageKeeper discussed in Chapter 2. We analyze 111K apps that made 91 million posts over nine months. This is arguably the first comprehensive study focusing on quantifying, profiling, and understanding malicious apps, and synthesizes this information into an effective detection approach.

First, we show that malicious apps are rampant in Facebook since we identified 13% of apps in our dataset of 111K distinct apps are malicious. These malicious apps reach a large number of users. For example, 60% of malicious apps endanger more than 100K users each by convincing them to follow the links on the posts made by these apps, and 40% of malicious apps have over 1,000 monthly active users each.

Next, we systematically profile both malicious and benign apps and show that the corresponding profiles differ significantly. For example, malicious apps require fewer permissions during installation than benign apps and they heavily reuse app names (8% of unique names of malicious apps are each used by more than 10 different apps) which is rare for benign apps. We profile apps into two different classes of features: (a)on-demand features which can be obtained on-demand given an app ID (i.e. app permission requirement) and (b) aggregation-based features which can be obtained across apps collected over time (i.e. posting behavior of apps).

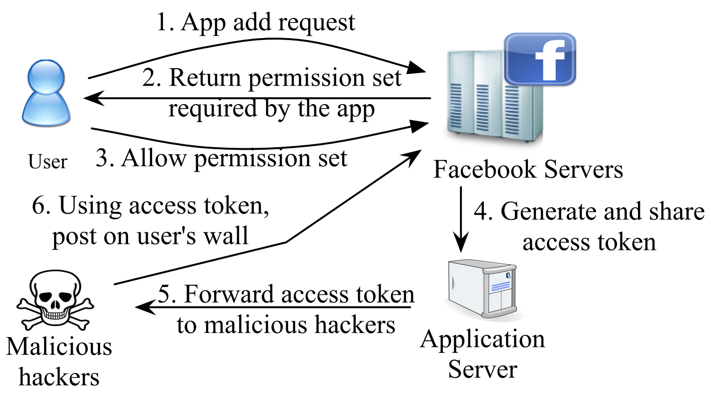

1. App add request
2. Return permission set required by the app
User
3. Allow permission set
Facebook Servers
6. Using access token, post on user's wall
4. Generate and share access token
5. Forward access token to malicious hackers
Malicious hackers
Application Server

Figure 3.2: Steps involved in hackers using malicious applications to get access tokens to post malicious content on victims' walls.

Based on our insight in app profiles, we develop FRAppE (Facebook's Rigorous Application Evaluator) to identify malicious apps. We show that, FRAppE Lite, which only uses information available on-demand, can identify malicious apps with 99.0% accuracy, with low false positives (0.1%) and false negatives (4.4%). However, by adding aggregation-based information, FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and lower false negatives (4.1%).

3.1 Background

In this section, we discuss how applications work on Facebook and outline the datasets that we use in this chapter. Chapter 2 provides detailed overview of MyPageKeeper, our primary data source.

3.1.1 Facebook Apps

Facebook enables third-party developers to offer services to its users by means of Facebook applications. Unlike typical desktop and smartphone applications, installation of a Facebook application by a user does not involve the user downloading and executing an application binary. Instead, when a user adds a Facebook application to her profile,

Dataset Name	**# of apps**	
	Benign	Malicious
D-Total	111,167	
D-Sample	6,273	6,273
D-Summary	6,067	2,528
D-Inst	2,257	491
D-ProfileFeed	3,227	6,063
D-Complete	2,255	487

Table 3.1: Summary of the dataset collected by MyPageKeeper from June 2011 to March 2012.

App ID	**App name**	**Post count**
235597333185870	What Does Your Name Mean?	1006
159474410806928	Free Phone Calls	793
233344430035859	The App	564
296128667112382	WhosStalking?	434
142293182524011	FarmVile	210

Table 3.2: Top malicious apps in D-Sample dataset.

the user grants the application server: (a) permission to access a subset of the information listed on the user's Facebook profile (e.g., the user's email address), and (b) permission to perform certain actions on behalf of the user (e.g., the ability to post on the user's wall). Facebook grants these permissions to any application by handing an OAuth 2.0 [10] token to the application server for each user who installs the application. Thereafter, the application can access the data and perform the explicitly-permitted actions on behalf of the user. Fig. 3.2 depicts the steps involved in the installation and operation of a Facebook application.

Operation of malicious applications. Malicious Facebook applications typically operate as follows.

- Step 1: hackers convince users to install the app, usually with some fake promise (e.g., free iPads).

- Step 2: once a user installs the app, it redirects the user to a web page where the user is requested to perform tasks, such as completing a survey, again with the lure of fake rewards.
- Step 3: the app thereafter accesses personal information (e.g., birth date) from the user's profile, which the hackers can potentially use to profit.
- Step 4: the app makes malicious posts on behalf of the user to lure the user's friends to install the same app (or some other malicious app, as we will see later).

This way the cycle continues with the app or colluding apps reaching more and more users. Personal information or surveys can be "sold" to third parties [4] to eventually profit the hackers.

3.1.2 Our Datasets

In the absence of a central directory of Facebook apps [1], the basis of our study is a dataset obtained from 2.2M Facebook users, who are monitored by MyPageKeeper [57].

Our dataset contains 91 million posts from 2.2 million walls monitored by MyPageKeeper over nine months from June 2011 to March 2012. These 91 million posts were made by 111K apps, which forms our initial dataset D-Total, as shown in Table 3.1. Note that, out of the 144M posts monitored by MyPageKeeper during this period, here we consider only those posts that included a non-empty "application" field in the metadata that Facebook associates with every post.

The D-Sample dataset: finding malicious applications. To identify malicious Facebook applications in our dataset, we start with a simple heuristic: if any post made by an application was flagged as malicious by MyPageKeeper, we mark the application as malicious; as we explain later in Section 3.4, we find this to be an effective

[1]Note that Facebook has deprecated the app directory in 2011, therefore there is no central directory available for the entire list of Facebook apps [25].

technique for identifying malicious apps. By applying this heuristic, we identified 6,350 malicious apps. Interestingly, we find that several popular applications such as 'Facebook for Android' were also marked as malicious in this process. This is in fact the result of hackers exploiting Facebook weaknesses as we describe later in Section 4.3. To avoid such mis-classifications, we verify applications using a whitelist that is created by considering the most popular apps and significant manual effort. After whitelisting, we are left with 6,273 malicious applications (D-Sample dataset in Table 3.1). Table 3.2 shows the top five malicious applications, in terms of number of posts per application.

The D-Sample dataset: including benign applications. To select an equal number of benign apps from the initial D-Total dataset, we use two criteria: (a) none of their posts were identified as malicious by MyPageKeeper, and (b) they are "vetted" by Social Bakers [67], which monitors the "social marketing success" of apps. This process yields 5,750 applications, 90% of which have a user rating of at least 3 out of 5 on Social Bakers. To match the number of malicious apps, we add the top 523 applications in D-Total (in terms of number of posts) and obtain a set of 6,273 benign applications. The D-Sample dataset (Table 3.1) is the union of these 6,273 benign applications with the 6,273 malicious applications obtained earlier. The most popular benign apps are FarmVille, Facebook for iPhone, Mobile, Facebook for Android, and Zoo World.

For profiling apps, we collect the information for apps that is readily available through Facebook. We use a crawler based on the Firefox browser instrumented with Selenium [66]. From March to May 2012, we crawl information for every application in our D-Sample dataset once every week. We collected app summaries and their permissions, which requires two different crawls as discussed below.

The D-Summary dataset: apps with app summary. We collect app summaries through the Facebook Open graph API, which is made available by Facebook at a URL of the form `https://graph.facebook.com/App_ID`; Facebook has a unique identifier for each application. An app summary includes several pieces of information such as *application name*, *description*, *company name*, *profile link*, and *monthly active users*. If any application has been removed from Facebook, the query results in an error. We were able to gather the summary for 6,067 benign and 2,528 malicious apps (D-Summary dataset in Table 3.1). It is easy to understand why malicious apps were more often removed from Facebook.

The D-Inst dataset: app permissions. We also want to study the permissions that apps request at the time of installation. For every application *App_ID*, we crawl `https://www.facebook.com/apps/application.php?id=App_ID`, which usually redirects to the application's installation URL. We were able to get the permission set for 487 malicious and 2,255 benign applications in our dataset. Automatically crawling the permissions for all apps is not trivial [92], as different apps have different redirection processes, which are intended for humans and not for crawlers. As expected, the queries for apps that are removed from Facebook fail here as well.

The D-ProfileFeed: posts on the app profile. Users can make posts on the profile page of an app, which we can call *the profile feed* of the app. We collect these posts using the Open graph API from Facebook. The API returns posts appearing on the application's page, with several attributes for each post, such as *message*, *link*, and *create time*. Of the apps in the D-Sample dataset, we were able to get the posts for 6,063 benign and 3,227 malicious apps. We construct the D-Complete dataset by taking the intersection of D-Summary, D-Inst, and D-ProfileFeed datasets.

Coverage: while the focus of our study is to highlight the differences between malicious and benign apps and to develop a sound methodology to detect malicious apps, we cannot aim to detect all malicious apps present on Facebook. This is because MyPageKeeper has a limited view of Facebook data—the view provided by its subscribed users—and therefore it cannot see all the malicious apps present on Facebook. However, during the nine month period considered in our study, MyPageKeeper observed posts from 111K apps, which constitutes a sizeable fraction (over 20%) of the approximately 500K apps present on Facebook [83]. Moreover, since MyPageKeeper monitors posts from 2.4 million walls on Facebook, any malicious app that affected a large fraction of Facebook users is likely to be present in our dataset. Therefore, we speculate that malicious apps missing from our dataset are likely to be those that affected only a small fraction of users.

Data privacy: our primary source of data in this work is our MyPageKeeper Facebook application, which has been approved by UCR's IRB process. In keeping with Facebook's policy and IRB requirements, data collected by MyPageKeeper is kept private, since it crawls posts from the walls and news feeds of users who have explicitly given it permission to do so at the time of MyPageKeeper installation. In addition, we also use data obtained via Facebook's open graph API, which is publicly accessible to anyone.

3.2 Prevalence of Malicious Apps

The driving motivation for detecting malicious apps stems from the suspicion that a significant fraction of malicious posts on Facebook are posted by apps. We find that 53% of malicious posts flagged by MyPageKeeper were posted by malicious apps. We further quantify the prevalence of malicious apps in two different ways.

60% of malicious apps get at least a hundred thousand clicks on the URLs they post. We quantify the reach of malicious apps by determining the number of clicks on the the links included in malicious posts. For each malicious app in our D-Sample dataset, we identify all `bit.ly` URLs in posts made by that application. We focus on `bit.ly` URLs since `bit.ly` offers an API [14] for querying the number of clicks received by every `bit.ly` link; thus our estimate of the number of clicks received by every application is strictly a lower bound. On the other hand, each `bit.ly` link that we consider here could potentially also have received clicks from other sources on web (i.e., outside Facebook); thus, for every `bit.ly` URL, the total number of clicks it received is an upper bound on the number clicks received via Facebook.

Across the posts made by the 6,273 malicious apps in the D-Sample dataset, we found that 3,805 of these apps had posted 5,700 `bit.ly` URLs in total. We queried `bit.ly` for the click count of each URL. Fig. 3.3 shows the distribution across malicious apps of the total number of clicks received by `bit.ly` links that they had posted. We see that 60% of malicious apps were able to accumulate over 100K clicks each, with 20% receiving more than 1M clicks each. The application with the highest number of `bit.ly` clicks in this experiment—the 'What is the sexiest thing about you?' app—received 1,742,359 clicks.

40% of malicious apps have a median of at least 1000 monthly active users. We examine the reach of malicious apps by inspecting the number of users that these applications had. To study this, we use the Monthly Active Users (MAU) metric provided by Facebook for every application. The number of Monthly Active Users is a measure of how many unique users are engaged with the application over the last 30 days in activities such as installing, posting, and liking the app. Fig. 3.4 plots the distribution of Monthly Active Users of the malicious apps in our D-Summary dataset. For each app, the median and maximum MAU values over the three months are shown. We see that

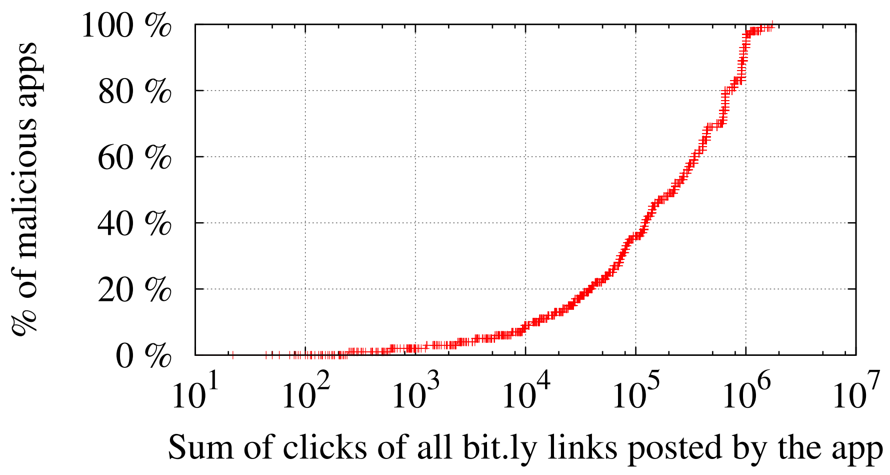


Figure 3.3: Clicks received by `bit.ly` links posted by malicious apps.

40% of malicious applications had a median MAU of at least 1000 users, while 60% of malicious applications achieved at least 1000 during the three month observation period. The top malicious app here—'Future Teller'—had a maximum MAU of 260,000 and median of 20,000.

3.3 Profiling Applications

Given the significant impact that malicious apps have on Facebook, we next seek to develop a tool that can identify malicious applications. Towards developing an understanding of how to build such a tool, in this section, we compare malicious and benign apps with respect to various features.

As discussed previously in Section 3.1.2, we crawled Facebook and obtained several features for every application in our dataset. We divide these features into two subsets: on-demand features and aggregation-based features. We find that malicious applications significantly differ from benign applications with respect to both classes of features.

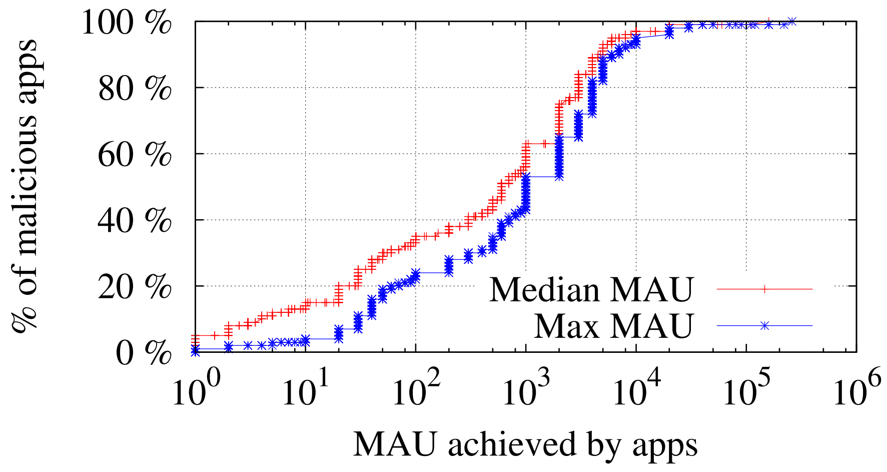


Figure 3.4: Median and maximum MAU achieved by malicious apps.

3.3.1 On-demand Features

The on-demand features associated with an application refer to the features that one can obtain on-demand given the application's ID. Such metrics include app name, description, category, company, and required permission set.

3.3.1.1 Application Summary

Malicious apps typically have incomplete application summaries. First, we compare malicious and benign apps with respect to attributes present in the application's summary—*app description*, *company name*, and *category*. Description and company are free-text attributes, either of which can be at most 140 characters. On the other hand, category can be selected from a predefined (by Facebook) list such as 'Games', 'News', etc. that matches the app functionality best. Application developers can also specify the company name at the time of app creation. For example, the 'Mafia Wars' app is configured with description as 'Mafia Wars: Leave a legacy behind', company as 'Zynga', and category as 'Games'. Fig. 3.5 shows the fraction of malicious and benign

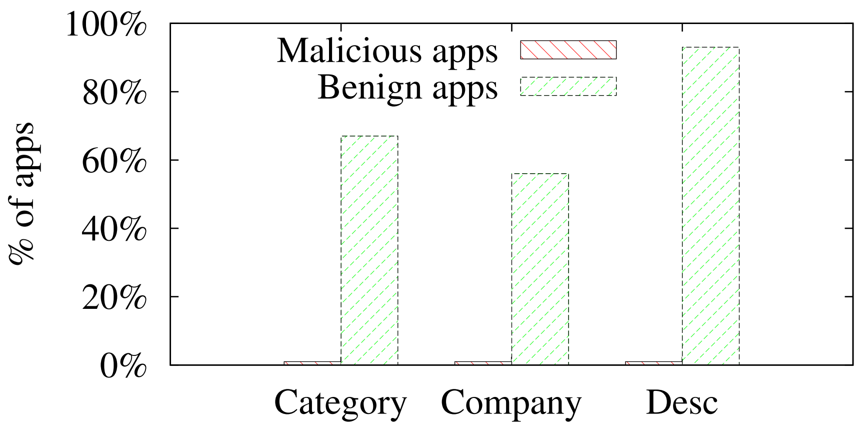


Figure 3.5: Comparison of apps whether they provide category, company name or description of the app.

apps in the D-Summary dataset for which these three fields are non-empty. We see that, while most benign apps specify such information, very rarely malicious apps do so. For example, only 1.4% of malicious apps have a non-empty description, whereas 93% of benign apps configure their summary with a description. We find that the benign apps that do not configure the description parameter are typically less popular (as seen from their monthly active users).

3.3.1.2 Required Permission Set

97% of malicious apps require only one permission from users. Every Facebook application requires authorization by a user before the user can use the app. At the time of installation, every app requests the user to grant it a set of permissions that it requires. These permissions are chosen from a pool of 64 permissions pre-defined by Facebook [61]. Example permissions include access to information in the user's profile such as gender, email, birthday, and friend list, and permission to post on the user's wall.

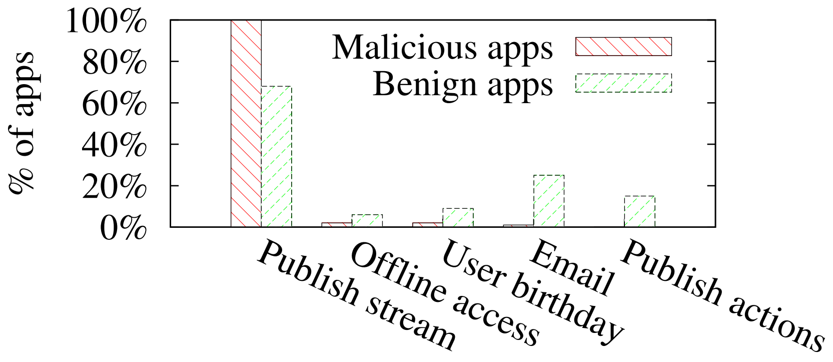


Figure 3.6: Top 5 permissions required by benign and malicious apps.

We see how malicious and benign apps compare based on the permission set that they require from users. Fig. 3.6 shows the top five permissions required by both benign and malicious apps. Most malicious apps in our D-Inst dataset require only the 'publish stream' permission (ability to post on the user's wall). This permission is sufficient for making spam posts on behalf of users. In addition, Fig. 3.7 shows that 97% of malicious apps require only one permission, whereas the same fraction for benign apps is 62%. We believe that this is because users tend not to install apps that require larger set of permissions; Facebook suggests that application developers do not ask for more permissions than necessary since there is a strong correlation between the number of permissions required by an app and the number of users who install it [23]. Therefore, to maximize the number of victims, malicious apps seem to follow this hypothesis and require a small set of permissions.

3.3.1.3 Redirect URI

Malicious apps redirect users to domains with poor reputation. In an applica-

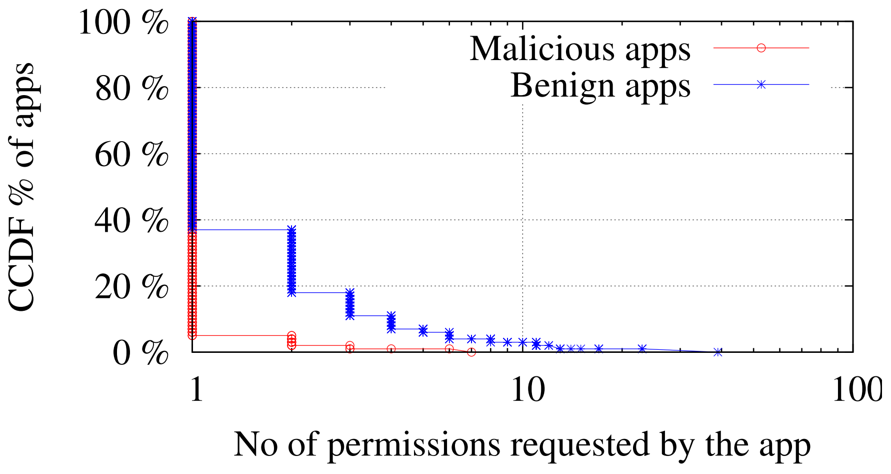


Figure 3.7: Number of permissions requested by every app.

Domains	**Hosting # of malicious apps**
thenamemeans3.com	34
fastfreeupdates.com	53
wikiworldmedia.com	82
technicalyard.com	96
thenamemeans2.com	138

Table 3.3: Top five domains hosting malicious apps in D-Inst dataset.

tion's installation URL, the 'redirect URI' parameter refers to the URL where the user is redirected to once she installs the app. We extracted the redirect URI parameter from the installation URL for apps in the D-Inst dataset and queried the trust reputation scores for these URIs from WOT [79]. Fig. 3.8 shows the corresponding score for both benign and malicious apps. WOT assigns a score between 0 and 100 for every URI, and we assign a score of -1 to the domains for which the WOT score is not available. We see that 80% of malicious apps point to domains for which WOT does not have any reputation score, and in addition, 95% of malicious apps have a score less than 5. In contrast, we find that 80% of benign apps have redirect URIs pointing to the `apps.facebook.com` domain and therefore have higher WOT scores. We speculate that

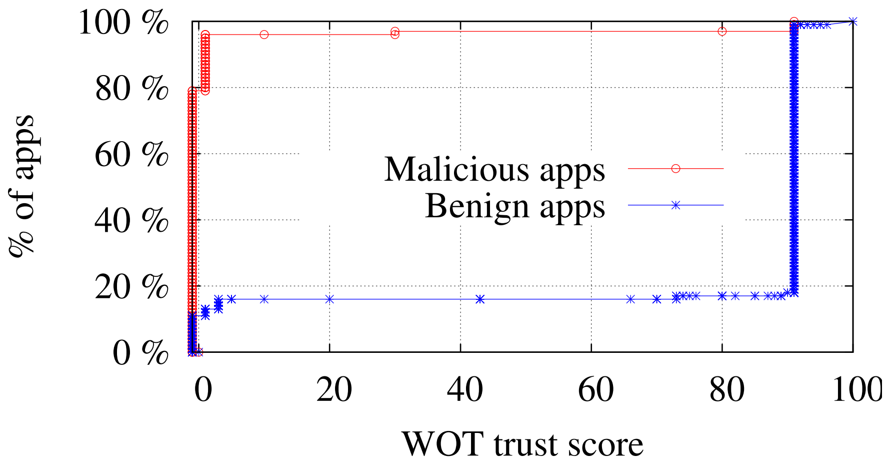


Figure 3.8: WOT trust score of the domain that apps redirect to upon installation.

malicious apps redirect users to web pages hosted outside of Facebook so that the same spam/malicious content, e.g., survey scams, can also be propagated by other means such as email and Twitter spam.

Furthermore, we found several instances where a single domain hosts the URLs to which multiple malicious apps redirect upon installation. For example, `thenamemeans2.com` hosts the redirect URI for 138 different malicious apps in our D-Inst dataset. Table 3.3 shows the top five such domains; these five domains host the content for 83% of the 491 malicious apps in the D-Inst dataset.

3.3.1.4 Client ID in App Installation URL

78% of malicious apps trick users into installing other apps by using a different client ID in their app installation URL. For a Facebook application with ID A, the application installation URL is `https://www.facebook.com/apps/application.php?id=A`. When any user visits this URL, Facebook queries the application server registered for app A to fetch several parameters, such as the set of permissions required by

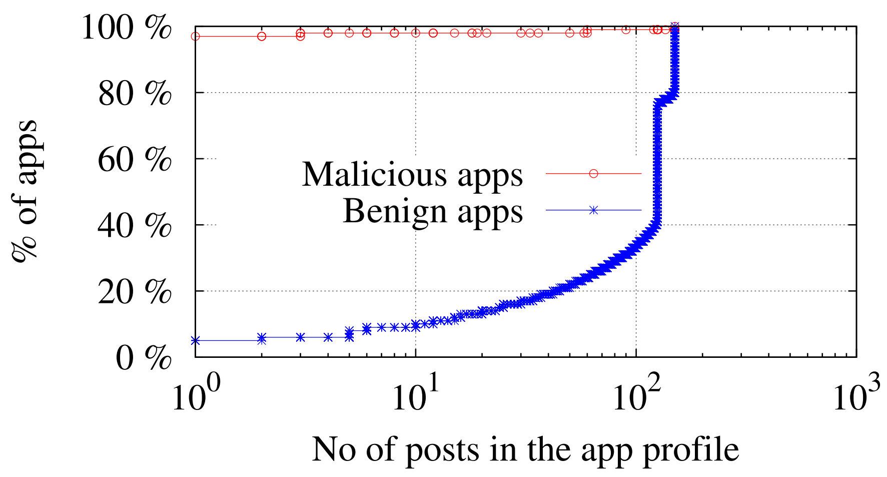


Figure 3.9: Number of posts in app profile page.

the app. Facebook then redirects the user to a URL which encodes these parameters in the URL. One of the parameters in this URL is the 'client ID' parameter. If the user accepts to install the application, the ID of the application which she will end up installing is the value of the client ID parameter. Ideally, as described in the Facebook app developer tutorial [23], this client ID should be identical to the app ID A, whose installation URL the user originally visited. However, in our D-Inst dataset, we find that 78% of malicious apps use a client ID that differs from the ID of the original app, whereas only 1% of benign apps do so. A possible reason for this is to increase the survivability of apps. As we show later in Chapter. 4, hackers create large sets of malicious apps with similar names, and when a user visits the installation URL for one of these apps, the user is randomly redirected to install any one of these apps. This ensures that, even if one app from the set gets blacklisted, others can still survive and propagate on Facebook.

3.3.1.5 Posts in App Profile

97% of malicious apps do not have posts in their profiles. An application's profile page presents a forum for users to communicate with the app's developers (e.g., to post comments or questions about the app) or vice-versa (e.g., for the app's developers to post updates about the application). Typically, an app's profile page thus accumulates posts over time. We examine the number of such posts on the profile pages of applications in our dataset. As discussed earlier in Sec. 3.1.2, we were able to crawl the app profile pages for 3,227 malicious apps and 6,063 benign apps.

From Fig. 3.9, which shows the distribution of the number of posts found in the profile pages for benign and malicious apps, we find that 97% of malicious apps do not have any posts in their profiles. For the remaining 3%, we see that their profile pages include posts that advertise URLs pointing to phishing scams or other malicious apps. For example, one of the malicious apps has 150 posts in its profile page and all of those posts publish URLs pointing to different phishing pages with URLs such as `http://2000forfree.blogspot.com` and `http://free-offers-sites.blogspot.com/`. Thus, the profile pages of malicious apps either have no posts or are used to advertise malicious URLs, to which any visitors of the page are exposed.

3.3.2 Aggregation-based Features

Next, we analyze applications with respect to aggregation-based features. Unlike the features we considered so far, aggregation-based features for an app cannot be obtained on-demand. Instead, we envision that aggregation-based features are gathered by entities that monitor the posting behavior of several applications across users and across time. Entities that can do so include Facebook security applications installed by a large population of users, such as MyPageKeeper, or Facebook itself. Here, we consider

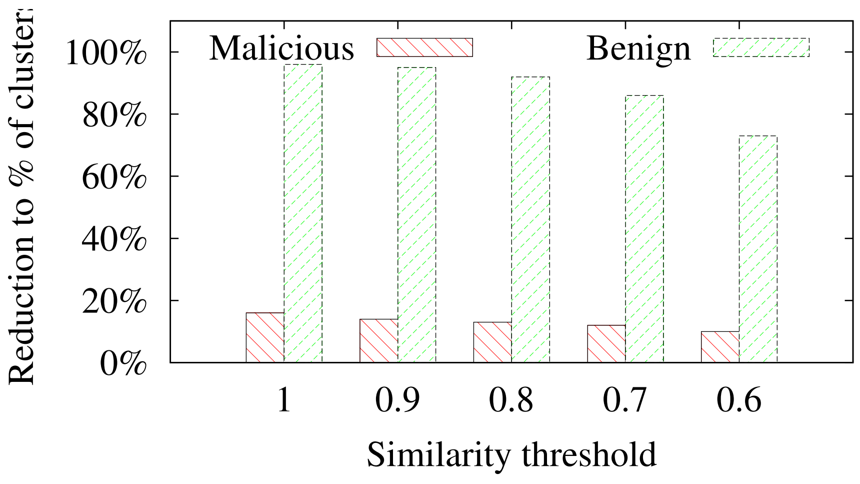


Figure 3.10: Clustering of apps based on similarity in names.

two aggregation-based features: similarity of app names, and the URLs posted by an application over time. We compare these features across malicious and benign apps.

3.3.2.1 App Name

87% of malicious apps have an app name identical to that of at least one other malicious app. An application's name is configured by the app's developer at the time of the app's creation on Facebook. Since the app ID is the unique identifier for every application on Facebook, Facebook does not impose any restrictions on app names. Therefore, although Facebook does warn app developers not to violate the trademark or other rights of third-parties during app configuration, it is possible to create multiple apps with the same app name.

We examine the similarity of names across applications. To measure the similarity between two app names, we compute the Damerau-Levenshtein edit distance [94] between the two names and normalize this distance with the maximum of the lengths

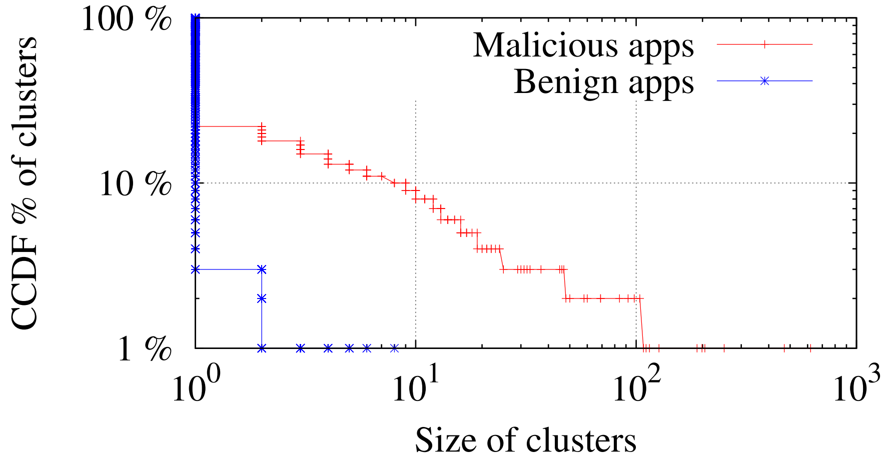


Figure 3.11: Size of app clusters with identical names.

of the two names. We then apply different thresholds on the similarity scores to cluster apps in the D-Sample dataset based on their name; we perform this clustering separately among malicious and benign apps.

Fig. 3.10 shows the ratio of the number of clusters to the number of apps, for various thresholds of similarity; a similarity threshold of 1 clusters applications that have identical app names. We see that malicious apps tend to cluster to a significantly larger extent than benign apps. For example, even when only clustering apps with identical names (similarity threshold = 1), the number of clusters for malicious apps is less than one-fifth that of the number of malicious apps, i.e., on average, 5 malicious apps have the same name. Fig. 3.11 shows that close to 10% of clusters based on identical names have over 10 malicious apps in each cluster. For example, 627 different malicious apps have the same name ‘The App’. On the contrary, even with a similarity threshold of 0.7, the number of clusters for benign apps is only 20% lesser than the number of apps. As a result, as seen in Fig. 3.11, most benign apps have unique names.

Moreover, while most of the clustering of app names for malicious apps occurs

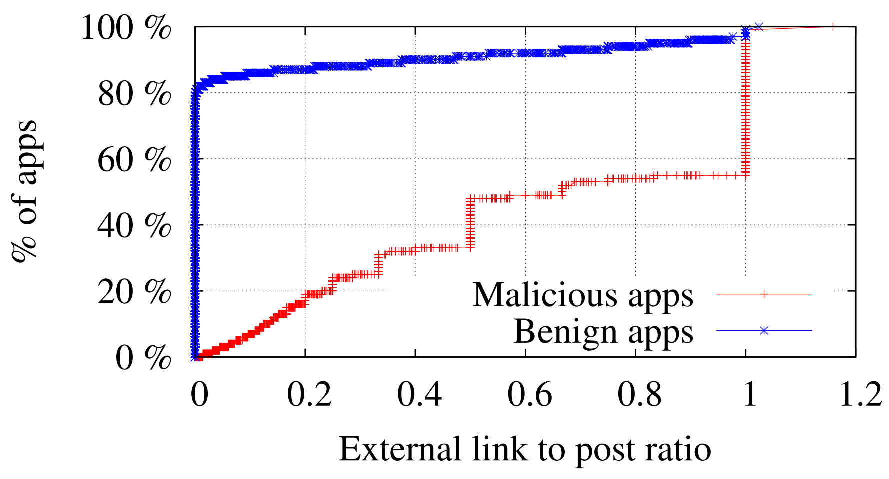


Figure 3.12: Distribution of external links to post ratio across apps.

even with a similarity threshold of 1, there is some reduction in the number of clusters with lower thresholds. This is due to hackers attempting to "typo-squat" on the names of popular benign applications. For example, the malicious application 'FarmVile' attempts to take advantage of the popular 'FarmVille' app name, whereas the 'Fortune Cookie' malicious application exactly copies the popular 'Fortune Cookie' app name. However, we find that a large majority of malicious apps in our D-Sample dataset show very little similarity with the 100 most popular benign apps in our dataset. Our data therefore seems to indicate that hackers creating several apps with the same name to conduct a campaign is more common than malicious apps typo-squatting on the names of popular apps.

3.3.2.2 External Link to Post Ratio

Malicious apps often post links pointing to domains outside Facebook, whereas benign apps rarely do so. Any post on Facebook can optionally include an URL. Here, we analyze the URLs included in posts made by malicious and benign apps. For every app in our D-Sample dataset, we aggregate the posts seen by MyPageKeeper over our nine month data gathering period and the URLs seen across these posts. We consider every URL pointing to a domain outside of `facebook.com` as an external link. We then define a 'external link to post ratio' measure for every app as the ratio of the number of external links posted by the app to the total number of posts made by it.

Fig. 3.12 shows that the external link to post ratios for malicious apps are significantly higher than those for benign apps. We see that 80% of benign apps do not post any external links, whereas 40% of malicious apps have one external link on average per post. This shows that malicious apps often attempt to lead users to web pages hosted outside Facebook, whereas the links posted by benign apps are almost always restricted to URLs in the `facebook.com` domain.

Note that malicious apps could post shortened URLs that point back to Facebook, thus potentially making our external link counts over-estimates. However, we find that malicious apps rarely do so. In our D-Sample dataset, we find 5700 *bit.ly* URLs (which constitute 92% of all shortened URLs) were posted by malicious apps. *bit.ly*'s API allowed us to determine the full URL corresponding to 5197 of these 5700 URLs, and only 386 of these URLs ($< 10\%$) pointed back to Facebook.

3.4 Detecting Malicious Apps

Having analyzed the differentiating characteristics of malicious and benign apps, we next use these features to develop efficient classification techniques to identify malicious Facebook applications. We present two variants of our malicious app classifier—FRAppE Lite and FRAppE. It is important to note that MyPageKeeper, our source of "ground truth" data, cannot detect malicious apps; it only detects malicious posts on Facebook. Though malicious apps are the dominant source of malicious posts, MyPageKeeper is

Features	**Source**
Is category specified?	`http://graph.facebook.com/appID`
Is company name specified?	`http://graph.facebook.com/appID`
Is description specified?	`http://graph.facebook.com/appID`
Any posts in app profile page?	`https://graph.facebook.com/AppID/feed?access_token=`
Number of permissions required	`https://www.facebook.com/apps/application.php?id=AppID`
Is client ID different from app ID?	`https://www.facebook.com/apps/application.php?id=AppID`
Domain reputation of redirect URI	`https://www.facebook.com/apps/application.php?id=AppID` and WOT

Table 3.4: List of features used in FRAppE Lite.

agnostic about the source of the posts that it classifies. In contrast, FRAppE Lite and FRAppE are designed to detect malicious apps. Therefore, given an app ID, MyPageKeeper cannot say whether it is malicious or not, whereas FRAppE Lite and FRAppE can do so.

3.4.1 FRAppE Lite

FRAppE Lite is a lightweight version which makes use of only the application features available on-demand. Given a specific app ID, FRAppE Lite crawls the on-demand features for that application and evaluates the application based on these features in real-time. We envision that FRAppE Lite can be incorporated, for example, into a browser extension that can evaluate any Facebook application at the time when a user is considering installing it to her profile.

Table 3.4 lists the features used as input to FRAppE Lite and the source of each feature. All of these features can be collected on-demand at the time of classification and do not require prior knowledge about the app being evaluated.

We use the Support Vector Machine (SVM) [91] classifier for classifying ma-

Training Ratio	Accuracy	FP	FN
1:1	98.5%	0.6%	2.5%
4:1	99.0%	0.1%	4.7%
7:1	99.0%	0.1%	4.4%
10:1	99.5%	0.1%	5.5%

Table 3.5: Cross validation with FRAppE Lite.

licious apps. SVM is widely used for binary classification in security and other disciplines [109, 102]. The effectiveness of SVM depends on the selection of kernel, the kernel's parameters, and soft margin parameter C. We used the default parameter values in libsvm [91] such as radial basis function as kernel with degree 3, $coef_0 = 0$ and $C = 1$ [91]. We use the D-Complete dataset for training and testing the classifier. As shown earlier in Table 3.1, the D-Complete dataset consists of 487 malicious apps and 2,255 benign apps.

We use 5-fold cross validation on the D-Complete dataset for training and testing FRAppE Lite's classifier. In 5-fold cross validation, the dataset is randomly divided into five segments, and we test on each segment independently using the other four segments for training. We use accuracy, false positive (FP) rate, and false negative (FN) rate as the three metrics to measure the classifier's performance. Accuracy is defined as the ratio of correctly identified apps (i.e., a benign/malicious app is appropriately identified as benign/malicious) to the total number of apps. False positive (negative) rate is the fraction of benign (malicious) apps incorrectly classified as malicious (benign).

We conduct four separate experiments with the ratio of benign to malicious apps varied as 1:1, 4:1, 7:1, and 10:1. In each case, we sample apps at random from the D-Complete dataset and run a 5-fold cross validation. Table 3.5 shows that, irrespective of the ratio of benign to malicious apps, the accuracy is above 98.5%. The higher the ratio

Feature	**Accuracy**	**FP**	**FN**
Category specified?	76.5%	45.8%	1.2%
Company specified?	72.1%	55.0%	0.8%
Description specified?	97.8%	3.3%	1.0%
Posts in profile?	96.9%	4.3%	1.9%
Client ID is same?	88.5%	1.0%	22.0%
WOT trust score	91.9%	13.4%	2.9%
Permission count	73.3%	49.3%	4.1%

Table 3.6: Classification accuracy with individual features.

of benign to malicious apps, the classifier gets trained to minimize false positives, rather than false negatives, in order to maximize accuracy. However, we note that the false positive and negative rates are below 0.6% and 5.5% in all cases. The ratio of benign to malicious apps in our dataset is equal to 7:1; of the 111K apps seen in MyPageKeeper's data, 6,273 apps were identified as malicious based on MyPageKeeper's classification of posts and an additional 8,051 apps are found to be malicious, as we show later. Therefore, we can expect FRAppE Lite to offer roughly 99.0% accuracy with 0.1% false positives and 4.4% false negatives in practice.

To understand the contribution of each of FRAppE Lite's features towards its accuracy, we next perform 5-fold cross validation on the D-Complete dataset with only a single feature at a time. Table 3.6 shows that each of the features by themselves too result in reasonably high accuracy. The 'Description' feature yields the highest accuracy (97.8%) with low false positives (3.3%) and false negatives (1.0%). On the flip side, classification based solely on any one of the 'Category', 'Company', or 'Permission count' features results in a large number of false positives, whereas relying solely on client IDs yields a high false negative rate.

Feature	Description
App name similarity	Is app's name identical to a known malicious app?
External link to post ratio	Fraction of app's posts that contain links to domains outside Facebook

Table 3.7: Additional features used in FRAppE.

3.4.2 FRAppE

Next, we consider FRAppE—a malicious app detector that utilizes our aggregation-based features in addition to the on-demand features. Table 3.7 shows the two features that FRAppE uses in addition to those used in FRAppE Lite. Since the aggregation-based features for an app require a cross-user and cross-app view over time, in contrast to FRAppE Lite, we envision that FRAppE can be used by Facebook or by third-party security applications that protect a large population of users.

Here, we again conduct a 5-fold cross validation with the D-Complete dataset for various ratios of benign to malicious apps. In this case, we find that, with a ratio of 7:1 in benign to malicious apps, FRAppE's additional features improve the accuracy to 99.5%, as compared to 99.0% with FRAppE Lite. Furthermore, the false negative rate decreases from 4.4% to 4.1%, and we do not have a single false positive.

3.4.3 Identifying New Malicious Apps

We next train FRAppE's classifier on the entire D-Sample dataset (for which we have all the features and the ground truth classification) and use this classifier to identify new malicious apps. To do so, we apply FRAppE to all the apps in our D-Total dataset that are not in the D-Sample dataset; for these apps, we lack information as to whether they are malicious or benign. Of the 98,609 apps that we test in this experiment, 8,144 apps were flagged as malicious by FRAppE.

Criteria	**# of apps validated**	**Cumulative**
Deleted from Facebook graph	6,591(81%)	6,591 (81%)
App name similarity	6,055(74%)	7,869 (97%)
Post similarity	1,664 (20%)	7,907(97%)
Typosquatting of popular apps	5(0.1%)	7,912(97%)
Manual validation	147 (1.8%)	8051 (98.5%)
Total validated	-	8051(98.5%)
Unknown	-	93 (1.5%)

Table 3.8: Validation of apps flagged by FRAppE.

Validation. Since we lack ground truth information for these apps flagged as malicious, we apply a host of complementary techniques to validate FRAppE's classification. We next describe these validation techniques; as shown in Table 3.8, we were able to validate 98.5% of the apps flagged by FRAppE.

Deleted from Facebook graph: Facebook itself monitors its platform for malicious activities, and it disables and deletes from the Facebook graph malicious apps that it identifies. If the Facebook API (`https://graph.facebook.com/appID`) returns false for a particular app ID, this indicates that the app no longer exists on Facebook; we consider this to be indicative of blacklisting by Facebook. This technique validates 81% of the malicious apps identified by FRAppE. Note that Facebook's measures for detecting malicious apps are however not sufficient; of the 1,464 malicious apps identified by FRAppE (that were validated by other techniques below) but are still active on Facebook, 35% have been active on Facebook since over four months with 10% dating back to over eight months.

App name similarity: if an application's name exactly matches that of multiple malicious apps in the D-Sample dataset, that app too is likely to be part of the same campaign and therefore malicious. On the other hand, we found several malicious apps using version numbers in their name (e.g., 'Profile Watchers v4.32', 'How long have you spent logged in? v8'). Therefore, in addition, if an app name contains a version number at the end and the rest of its name is identical to multiple known malicious apps that similarly use version numbers, this too is indicative of the app likely being malicious.

Posted link similarity: if an URL posted by an app matches the URL posted by a previously known malicious app, then these apps are likely part of the same spam campaign, thus validating the former as malicious.

Typosquatting of popular app: if an app's name is "typosquatting" that of a popular app, we consider it malicious. For example, we found five apps named 'FarmVile', which are seeking to leverage the popularity of 'FarmVille'.

Manual verification: lastly, for the remaining 232 apps un-verified by the above techniques, we first cluster them based on name similarity among themselves and verify one app from each cluster with cluster size greater than 4. For example, we find 83 apps named 'Past Life'. This enabled us to validate an additional 147 apps marked as malicious by FRAppE.

Validation of ground truth. Note that some of the above-mentioned techniques also enable us to validate the heuristic we used to identify malicious apps in all of our datasets: if any post made by an application was flagged as malicious by MyPageKeeper, we marked the application as malicious. As of October 2012, we find that, out of the 6273 malicious apps in our D-Sample dataset, 5390 apps have been deleted from the Facebook graph. An additional 667 apps have an identical name to one of the 5390 deleted apps. Therefore, we believe that the false positive rate in the data that we use to train FRAppE Lite and FRAppE is at most 2.6%.

3.5 Discussion

In this section, we discuss potential measures that hackers can take to evade detection by FRAppE. We also present recommendations to Facebook about changes that they can make to their API to reduce abuse by hackers.

Robustness of features. Among the various features that we use in our classification, some can easily be obfuscated by malicious hackers to evade FRAppE in the future. For example, we showed that, currently, malicious apps often do not include a category, company, or description in their app summary. However, hackers can easily fill in this information into the summary of applications that they create from here on. Similarly, FRAppE leveraged the fact that profile pages of malicious apps typically have no posts. Hackers can begin making dummy posts in the profile pages of their applications to obfuscate this feature and avoid detection. Therefore, some of FRAppE's features may no longer prove to be useful in the future while others may require tweaking, e.g., FRAppE may need to analyze the posts seen in an application's profile page to test their validity. In any case, the fear of detection by FRAppE will increase the onus on hackers while creating and maintaining malicious applications.

On the other hand, we argue that several features used by FRAppE, such as the reputation of redirect URIs, the number of required permissions, and the use of different client IDs in app installation URLs, are robust to the evolution of hackers. For example, to evade detection, if malicious app developers were to increase the number of permissions required, they risk losing potential victims; the number of users that install an app has been observed to be inversely proportional to the number of permissions required by the app. Similarly, not using different client IDs in app installation URLs would limit the ability of hackers to instrument their applications to propagate each other. We find that a version of FRAppE that only uses such robust features still yields an accuracy of 98.2%, with false positive and false negative rates of 0.4% and 3.2% on a 5-fold cross validation.

Recommendations to Facebook. Our investigations of malicious apps on Facebook identified two key loopholes in Facebook's API which hackers take advantage of. First, as discussed in Sec. 3.3.1.4, malicious apps use a different client ID value in the app installation URL, thus enabling the propagation and promotion of other malicious apps. Therefore, we believe that Facebook must enforce that when the installation URL for an app is accessed, the client ID field in the URL to which the user is redirected must be identical to the app ID of the original app. We are not aware of any valid uses of having the client ID differ from the original app ID.

3.6 Summary of Results

In this chapter, we take a first look into malicious Facebook application characteristics in terms of number of permissions required, app name, number of URLs posted by an app etc. The key results we presented in this chapter are as follows:

- We show that, a significant number of apps (13%) are deployed by malicious hackers that affect a large number of Facebook users in terms of higher clickthrough or monthly active users.
- We show that malicious apps differ significantly from benign apps with respect to different features. For example, malicious apps are much more likely to share names with other apps, and they typically request fewer permissions than benign apps.
- Leveraging our observations, we developed FRAppE, an accurate classifier for

detecting malicious Facebook applications.

Chapter 4

Understanding Malicious Applications' Ecosystem

Our analysis in Chapter 3 shows that malicious apps are rampant in Facebook and indicates that they do not operate in isolation. Indeed, we find in this chapter that malicious apps collude at large scale—many malicious apps share the same name, several of them redirect to the same domain upon installation, etc. To establish such collaboration, malicious apps make post containing links to the installation page of other malicious apps. We use the term AppNets to capture such phenomenon of collaboration.

In this chapter, we conduct a forensics investigation on the malicious app ecosystem to identify and quantify the techniques used to promote malicious apps. We find that, malicious hackers often use highly sophisticated mechanism such as fast-changing indirection for promoting malicious applications. In this approach, malicious applications post URLs that point to a website, and the website dynamically redirects to many different app installation pages. We find 103 such URLs that point to 4,676 different malicious apps over the course of a month. Overall, we find 1,584 promoter apps that promote 3,723 other apps.

If we describe such collusion relationship of promoting-promoted apps as a graph, which we call AppNet, we find that apps collude and collaborate at a massive scale. These apps form large and highly-dense connected components in AppNet. For example, 25% apps in AppNet have a local clustering larger than 0.74 and 70% of the apps collude with more than 10 other apps.

We were also surprised to find popular good apps, such as 'FarmVille' and 'Facebook for iPhone', posting malicious posts. On further investigation, we found a lax authentication rule in Facebook that enabled hackers to make malicious posts appear as though they came from these apps.

These observed behaviors indicate well-organized crime: few prolific hacker groups control many malicious apps. For example, 56% of apps in AppNet are created by a single hacker group.

Our recommendations to Facebook. The most important message of the work is that there seems to be a parasitic eco-system of malicious apps within Facebook that needs to be understood and stopped. However, even this initial work leads to the following recommendations for Facebook that could potentially also be useful to other social platforms:

a. Breaking the cycle of app propagation. We recommend that apps should not be allowed to promote other apps. This is the reason that malicious apps seem to gain strength by self-propagation.

b. Enforcing stricter app authentication before posting. We recommend a stronger authentication of the identity of an app before a post by that app is accepted. As we saw, hackers fake the true identify of an app in order to evade detection and appear more credible to the end user.

4.1 Background on App Cross-promotion

App promotion happens in two different ways. The promoting app can post a link that points directly to another app, or it can post a link that points to a *redirection URL*, which points dynamically to multiple different apps.

a. Posting direct links to other apps. We found evidence that malicious apps often promote each other by posting victim's wall that redirect users to the promotee's app page. Such post appears to the victim's friend's news feed and lures them to install the promoted app and thereby promotee accumulates more victims. To investigate the evidence further, We crawled URLs posted by malicious apps and examined if the landing URL if it corresponds to an app installation page and extracted the app ID. In this way, we constructed a promoter-promotee relationship We find 692 promoter apps in our D-Sample dataset from Sec. 3.1 which promoted 1,806 different apps using direct links.

b. Indirect app promotion. Alternatively, hackers use websites outside Facebook to have more control and protection in promoting apps. In fact, the operation here is more sophisticated and it obfuscates information at multiple places. Specifically, a post made by a malicious app includes a shortened URL and that URL, once resolved, points to a website outside Facebook [40] This external website forwards users to several different app installation pages over time.

The use of the indirection mechanism is quite widespread, as it provides a layer of protection to the apps involved. We identify 103 indirection websites in our dataset of colluding apps. To identify all the landing websites, for one and a half months from mid-March to end of April 2012, we follow each indirection website 100 times a day using an instrumented Firefox browser.

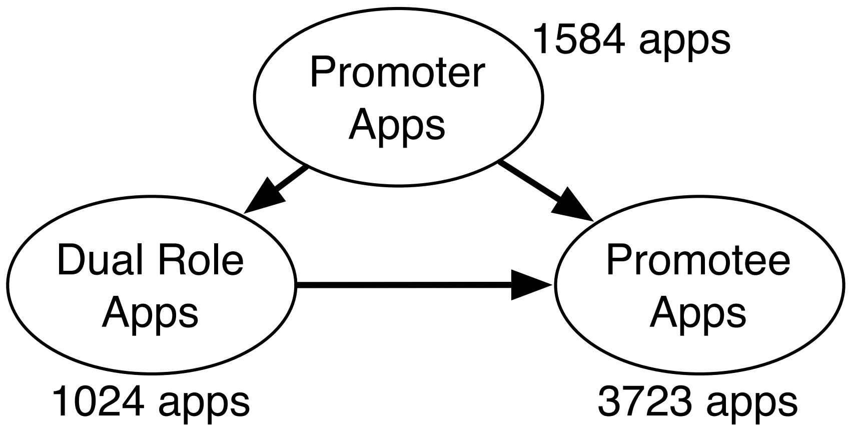


Figure 4.1: Relationship between collaborating applications

4.2 Analysis

In Sec. 4.1, we show different ways malicious apps promote each-other. In this section, we analyze app promotion in general to throw light on characteristics on AppNet.

4.2.1 Promotion Graph Characteristics

From the app promotion dataset we collected in Sec. 4.1, we construct a graph having an undirected edge between any two apps that promote each other via direct or indirect promotion, i.e., an edge from *app*1 to *app*2 if the former promotes the latter. We refer such graph as 'Promotion graph'.

4.2.1.1 Different Roles in Promotion Graph

Apps act in different roles for promotion. In 'Promotion graph', there are 6,331 malicious apps that engage in collaborative promotion. Among them, 25% are *promoters*, 58.8% are *promotees*, and the remaining 16.2% play both roles. Here, when *app*1 posts a link pointing to *app*2, we refer to *app*1 as the promoter and *app*2 as the promotee.

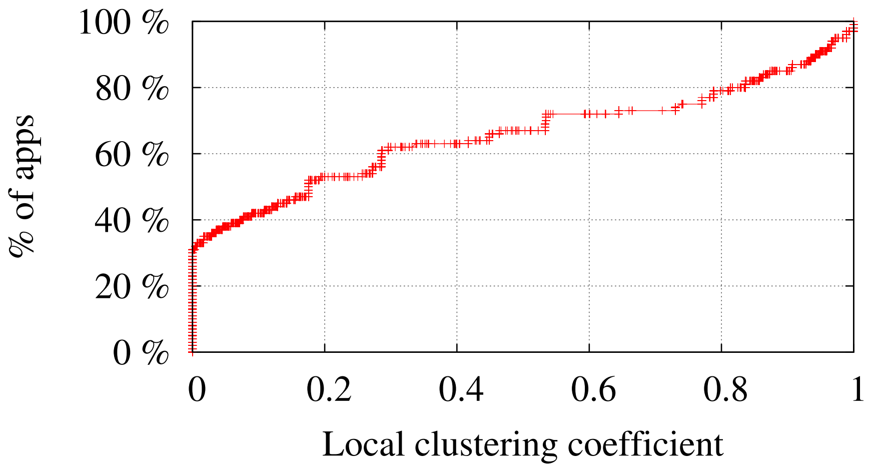


Figure 4.2: Local clustering coefficient of apps in the Collaboration graph.

Fig. 4.1 shows this relationship between malicious apps.

4.2.1.2 Connectivity

Promotion graph forms large and densely connected groups. We identified 44 connected components among the 6,331 malicious apps. The top 5 connected components have large sizes: 3484, 770, 589, 296, and 247. Upon further analysis of these components, we find:

- *High connectivity:* 70% of the apps collude with more than 10 other apps. The maximum number of collusions that an app is involved in is 417.
- *High local density:* 25% of the apps have a local clustering coefficient [1] larger than 0.74 as shown in Fig. 4.2.

As an example, in Fig. 4.3, we show the local neighborhood of the "Death Predictor" app, which has 26 neighbors and has a local clustering coefficient of 0.87.

[1]Local clustering coefficient for a node is the number of edges among the neighbors of a node over the maximum possible number of edges among those nodes. Thus, a clique neighborhood has a coefficient of value 1, while a disconnected neighborhood (the neighbors of the center of a star graph) has a value of 0.

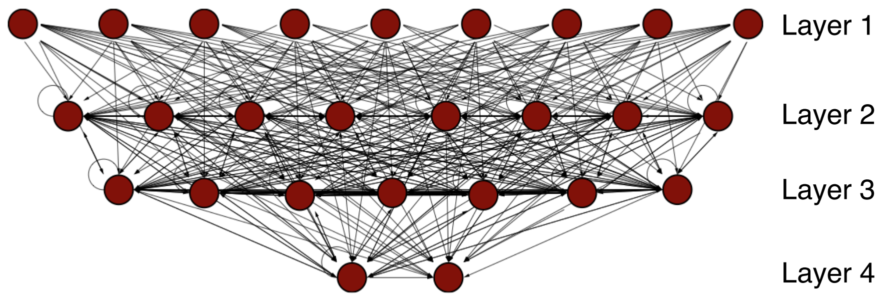


Figure 4.3: Example of collusion graph between applications.

Interestingly, 22 of the node's neighbors share the same name.

4.2.1.3 Degree Distribution

In order to understand the relationship between promoter and promotee apps, we create a directed graph, where each nodes is an app, and an edge from App1 to App2 means that App1 promotes App2. Fig. 4.4 shows the in-degree and out-degree distribution of the graph. We can see that 20% of apps have in-degree or out-degree more than 50 which means these 20% apps have been promoted by at least 50 other apps and each of them also promotes 50 other apps.

4.2.1.4 Longest Chain in Promotion

AppNet often exhibits large chain of promotion. We are interested in finding the longest path of promotion in this directed graph. This promotion graph is a directed cyclic graph and finding a simple path of maximum length in directed cyclic graph is NP-complete problem [55]. Therefore, we approximate this algorithm by using a threshold where the algorithm returns after reaching the threshold time. Fig. 4.5 shows the distribution of longest path starting from different pure promoter apps. We see that the longest path of promotion is 193 and 40% of pure promoters have longest path at

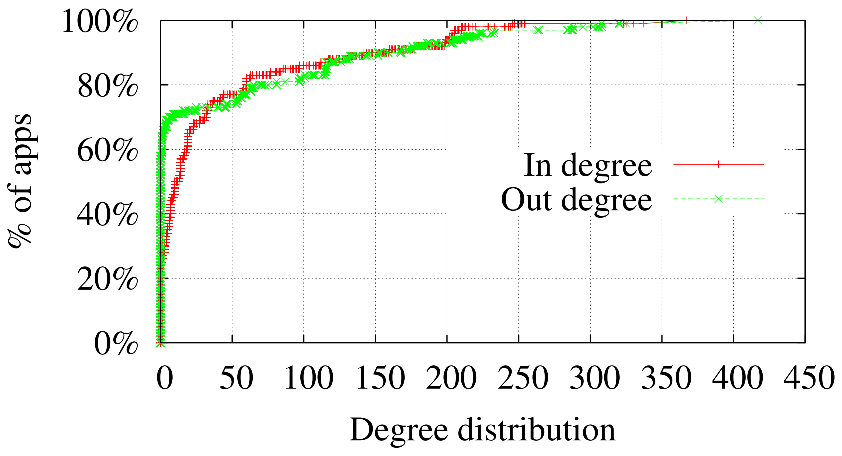


Figure 4.4: Degree distribution of apps in promotion graph.

least 20. In such paths, at most 17 distinct app names were used as shown in Fig.4.6 and 40% of the longest paths use at least four different app names. For example, 'Top Viewers v5' promotes 'Secret Lookers v6' which in turn promotes 'Top Lookers v6'. 'Top Lookers v6' then which promotes 'who Are They? v4' which in turn promotes 'Secret Lurkers v5.73' and so on.

4.2.1.5 Participating App Names in Promotion Graph

Apps with the same name often are part of the same AppNet. These 103 indirection website were used by 1,936 promoter apps which had only 206 unique app names. The promotees were 4,676 apps with 273 unique app names. Clearly, there is a very high re-use of both names and these indirection websites. For example, one indirection website distributed in posts by the app '*whats my name means*' points to the installation page of the apps '*What ur name implies!!!*', '*Name meaning finder*', and '*Name meaning*'. Furthermore, 35% of these websites promoted more than 100 different

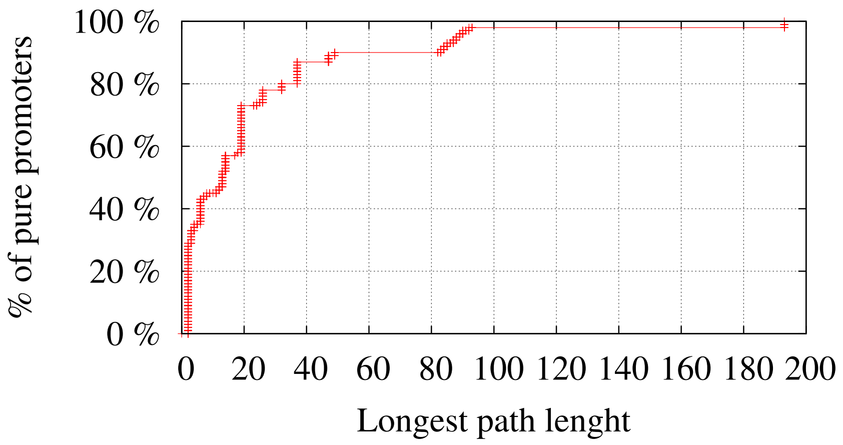


Figure 4.5: Longest path length from pure promoters in promotion graph.

applications each. Following the discussion in Sec. 3.3.2.1, it appears that every hacker reuses the same names for his applications. Since all apps underlying a campaign have the same name, if any app in the pool gets black listed, others can still survive and carry on the campaign without being noticed by users.

4.2.2 App Collaboration

In this part, we want to identify the major hacker groups involved for malicious app collusion. Therefore, we construct 'Collaboration graph' which comprises of both 'Promotion graph' and 'Campaign graph'. We defined 'Promotion graph' in Sec. 4.2.1. We define different variations of 'Campaign graph' below.

- Posted URL campaign: two apps are part of a campaign if they post a common URL.

- Hosted domain campaign: two apps are part of a campaign if they redirect to

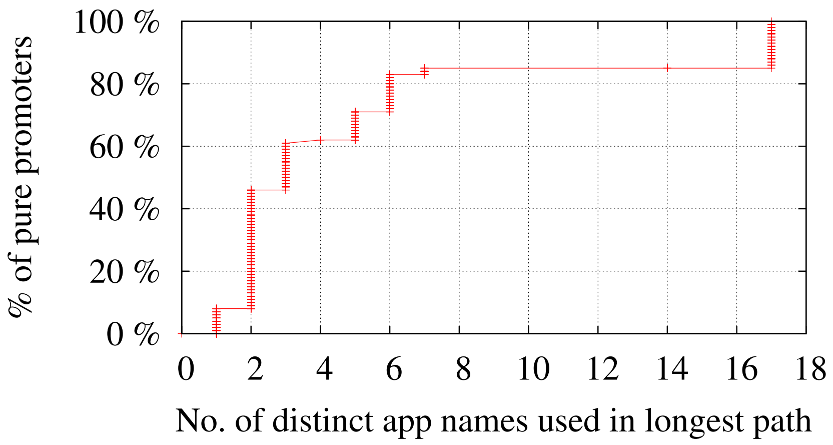


Figure 4.6: No. of distinct app names used in longest path from pure promoters in promotion graph.

same domain once they are installed by victim. We exclude `apps.facebook.com` as a redirect domain for such campaigns.

- Promoted URL campaign: two apps are part of a campaign if they are promoted by same indirection URL.

Table 4.1 shows properties (Avg. clustering coefficient [2], diameter [3] and GCC [4]) of such campaign graphs along with the promotion graph we discussed in Sec. 4.2.1 It is important to note that the nodes in a campaign form a clique. Next, we merge all graphs into a single graph to identify the largest connected component. We found that the largest GCC is 56% with 41 connected component which means 56% of malicious apps in our corpus are controlled by a single malicious hacker group. The

Graph	**# Nodes**	**# of Edges**	**Avg clustering coef**	**# of Connected comp.**	**Diameter**	**GCC**
Promotion graph	6331	206,983	.3	44	14	55%
Promoted URL campaign	4538	491,528	.92	21	10	66%
Hosted domain campaign	3970	193,061	.99	116	3	10%
Posted URL campaign	866	9,052	0.59	69	10	70%

Table 4.1: Graph properties.

Domain Name	**# of fast flux URLs hosted**
s3.amazonaws.com	27
chinappameu.co.in	23
t.co	7
dlaasta.org.in	4
www.facebook.com	4

Table 4.2: Top five domains abused for fast flux infrastructure.

largest five component sizes are 3617, 781, 645, 296 and 247.

4.2.3 Hosting Infrastructure

We investigate the hosting infrastructure that enables these redirection websites. First, we find that most of the links in the posts were shortened URLs and 80% of them were using the `bit.ly` shortening service. We consider all the `bit.ly` URLs among our dataset of indirection links (84 out of 103) and resolve them to the full URL. We find that one-third of these URLs are hosted on `amazonaws.com`. Table 4.2 shows the top five domains that host these indirection websites. We see that malicious hackers use `amazonaws.com` significantly.

[2]Average clustering coef is calculated by taking average of the local clustering coefficient of all nodes

[3]Diameter is the longest shortest path between two nodes in the graph

[4]Giant connected component (GCC) is the fraction of nodes comprise the largest connected component in the graph

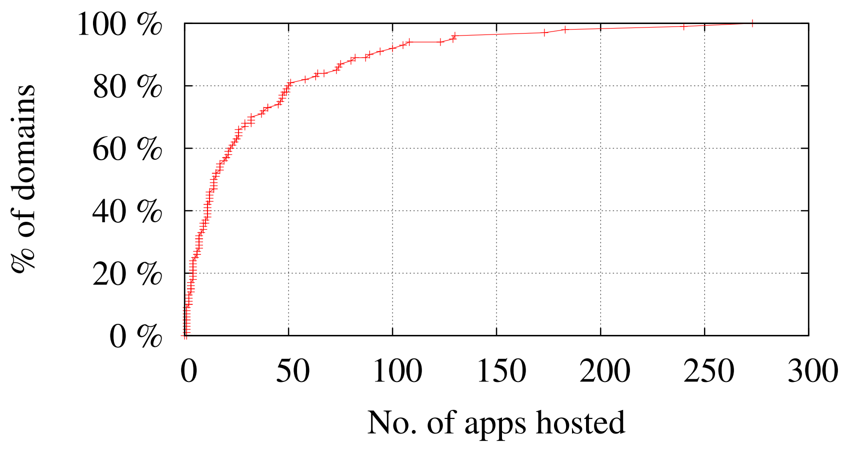


Figure 4.7: Domain hosting malicious apps.

Domain	**# of app hosted**
atisiokalisana.com	129
tipsyard.com	130
bchuck.info	173
birthdinnmans.co.cc	183
super-dox.co.cc	240

Table 4.3: Top five domain hosting malicious apps.

Next, we analyze the hosting domains of the malicious apps promoted by indirect promotion. We found the activity intense, 20% of domains host at least 50 different apps, as shown in Fig. 4.7. Table 4.3 shows the name of the top five hosting domains and the number of apps they host. This shows that, malicious hackers heavily reuse domains for hosting malicious apps.

App name	# of posts	Post msg	Link in post
FarmVille	9,621,909	WOW I just got 5000 Facebook Credits for Free	`http://offers5000credit.blogspot.com`
Links	7,650,858	Get your FREE 450 FACEBOOK CREDITS	`http://free450offer.blogspot.com/`
Facebook for iPhone	5,551,422	NFL Playoffs Are Coming! Show Your Team Support!	`http://SportsJerseyFever.com/NFL`
Mobile	4,208,703	WOW! I Just Got a Recharge of Rs 500.	`http://ffreerechargeindia.blogspot.com/`
Facebook for Android	3,912,955	Get Your Free Facebook Sim Card	`http://j.mp/oRzBNU`

Table 4.4: Top five popular apps being abused by app piggybacking.

4.3 App Piggybacking

From our dataset, we also discover that hackers have found ways to make malicious posts appear as if they had been posted by popular apps. To do so, they exploit weaknesses in Facebook's API. We call this phenomenon **app piggybacking**. One of the ways in which hackers achieve this is by luring users to 'Share' a malicious post to get promised gifts. When the victim tries to share the malicious post, hackers invoke the Facebook API call `http://www.facebook.com/connect/prompt_feed.php?api_key=POP_APPID`, which results in the shared post being made on behalf of the popular app POP_APPID. The vulnerability here is that any one can perform this API call, and Facebook does not authenticate that the post is indeed being made by the application whose ID is included in the request. We illustrate the app piggybacking mechanism with a real example here: [8].

We find instances of app piggybacking in our dataset as follows. For every

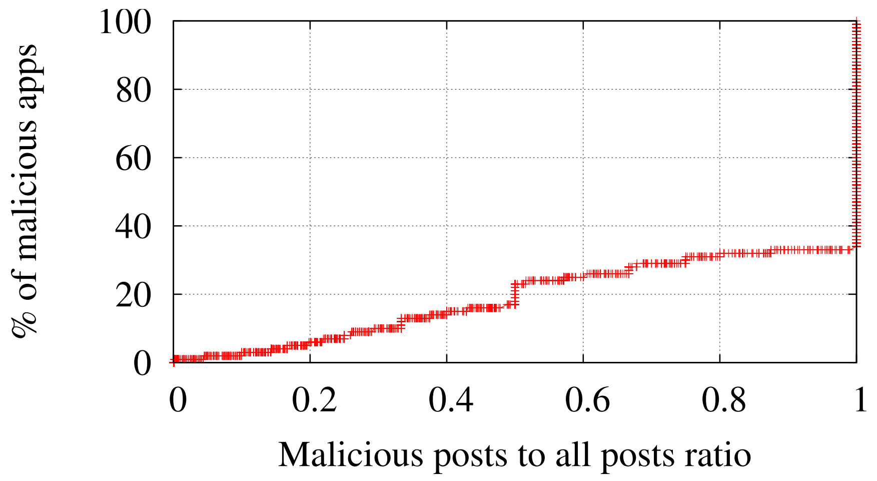


Figure 4.8: Distribution across apps of the fraction of an app's posts that are malicious.

app that had at least one post marked as malicious by MyPageKeeper, we compute the fraction of that app's posts that were flagged by MyPageKeeper. We look for apps where this ratio is low. In Fig. 4.8, we see that 5% of apps have a *malicious posts to all posts ratio* of less than 0.2. For these apps, we manually examine the malicious posts flagged by MyPageKeeper. Table 4.4 shows the top five most popular apps that we find among this set.

4.4 Cross-promotion as a Sign of Malicious Intentions

According to Facebook platform policy, cross-promotion of other apps is forbidden [28]. In this chapter, we show that cross-promotion is prevalent for malicious apps. However, we identified these cross-promoting malicious apps by investigating malicious URLs flagged by MyPageKeeper. Therefore, it is possible that cross-promotion may happen with URLs not flagged by MyPageKeeper due to lack of social context. In this section, we analyze cross-promotion made via posted `apps.facebook.com` URL and posted shortened URL nor flagged by MyPageKeeper.

Graph	**# Nodes**	**# of Edges**	**# of Connected comp.**	**GCC**
Malicious cross-promoting apps	2,052	3,542	22	91%
Unverified cross-promoting apps	3,026	4,527	370	32%

Table 4.5: Cross-promoting app graph properties.

4.4.1 Data Collection

Cross-promotion via posted `apps.facebook.com` URL: to investigate app promotion via posted `apps.facebook.com` links, we collect 41M URLs monitored by MyPageKeeper that points to `apps.facebook.com/namespace` domain and posted by $13,698$ apps. Then we identify corresponding app ID of a *namespace* from our dataset containing 80K namespace and app ID mapping. In this way, we find the cross-promotion relationship between promoter and promotee apps. We ignored self promotion where one app promotes itself. We identified 7,700 cross promoting relations involving 4,782 distinct apps in this way.

Cross-promotion via posted shortened URLs: to investigate app promotion via posted shortened URLs, we collect 5.8M shortened links monitored by MyPageKeeper, out of which 65,448 resolves to `apps.facebook.com` domain. Applying similar technique as mentioned above, we identified 1,177 cross promoting relations involving 450 distinct apps via shortened URLs.

In total, we found 5,077 distinct apps involved in 8,069 cross promotion via posted `apps.facebook.com` or shortened URLs. As per Facebook platform policy, these 5,077 apps are violating policy. Intrigued, we investigate them further.

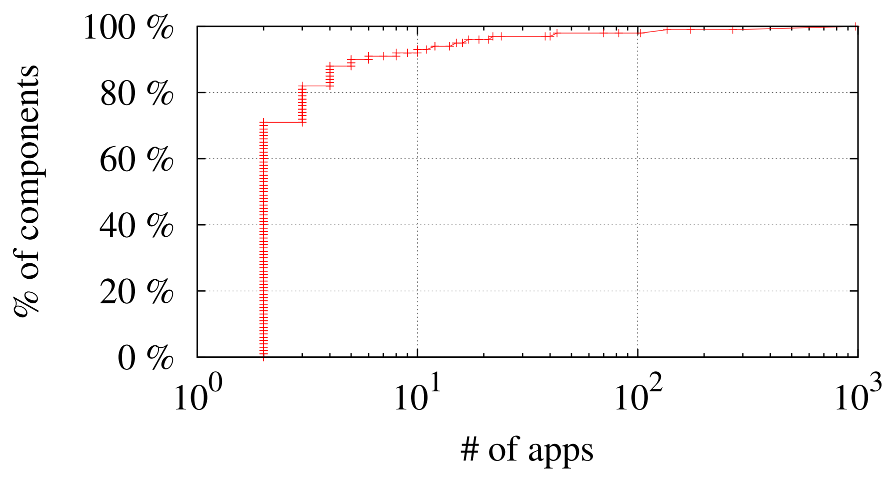


Figure 4.9: App distribution of components for unverified cross-promoting apps.

4.4.2 Analyzing Cross-promoting Apps

To identify malicious apps from the 5,077 apps, we compare them with our 14K malicious apps corpus identified by FRAppE in Chapter 3. We consider both apps in a promoter-promotee relationship malicious if either of them appear in our malicious app corpus. In this way, we could identify 2,052 distinct apps as malicious. However, the rest of the 3,025 apps are not connected to FRAppE detected malicious app corpus. Table 4.5 shows the properties of graphs created from the malicious cross-promoting apps and unverified cross-promoting apps. As shown in the table, malicious apps are tightly connected since the largest connected component owns 91% of the apps.

For unverified cross-promoting app graph, the largest five component sizes are: 972, 270, 174, 136 and 103. Fig. 4.9 shows the app distribution for connected components found in unverified cross-promoting apps. We see that 8% of components have at least 10 apps promoting each other. As shown in Fig. 4.10, 90% and 85%of app has in degree and out degree not more than one respectively. However, few apps have very high in

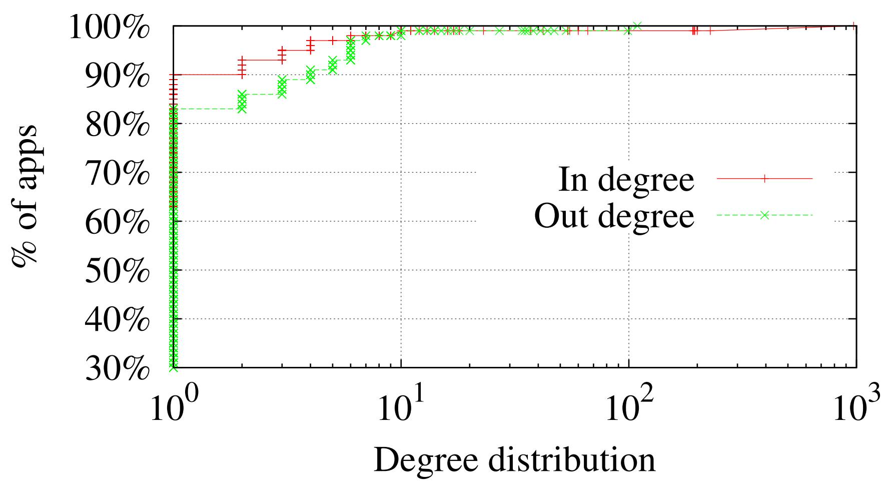


Figure 4.10: Degree distribution of unverified cross-promoting apps.

degree or out degree. For example, the app 'Quiz Whiz' belongs to a family of quiz apps and they often promotes each other. Few example apps in this quiz family are: 'Which Christmas Character are you?' and 'what twilight vampire are you?'.

Next, to find the popularity of these apps in terms of monthly active users (MAU), we crawl their app summary from Facebook opengraph. We found that, 702 out of 3,026 apps are already deleted from Facebook social graph. Fig. 4.11 shows the popularity of rest of the 2324 apps in terms of MAU. We found that few apps are very popular since they have MAU of several millions. For example, two different app ID with name 'Daily Horoscope' promote each other who have 9.7M and 1.4M MAU. Further, we found that popular games sometimes cross-promote each other. For example, 'Social Empires' game was promoted by 'Trial Madness' and 'Social Wars'. We speculate that they belong to the same company and they often promote each other for increasing user base.

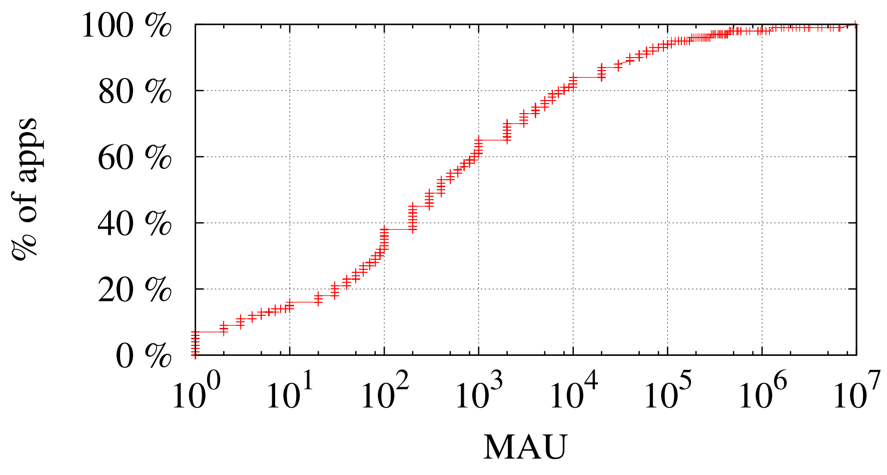


Figure 4.11: MAU distribution of unverified cross-promoting apps.

4.5 Summary of Results

In this chapter, we take a first look into malicious Facebook apps ecosystem—how they support each other. The key results we present in this chapter are as follows:

- We show different promotion strategies such as direct and indirect promotions malicious apps employ to support each other. Indirect promotion is a very sophisticated technique where malicious hackers redirect users to fast changing URL which randomly takes users to different malicious app installation page.
- We analyze such collusion using graph-based technique and show that such graph exhibits high density and high local clustering coefficient. The apps in AppNet often shows promotion path chain as long as 193.
- We demonstrate how malicious hackers abuse Facebook API to piggyback on benign and popular apps to spread malicious posts in Facebook.

Chapter 5

Related Work

5.1 Indentifying Social Malware

Motivated by the increasing presence of spam and malware on OSNs, there have been several recent related efforts. Here, we contrast our work with these prior efforts.

Studies of spam on OSNs. Gao et al. [97] analyzed posts on the walls of 3.5 million Facebook users and showed that 10% of links posted on Facebook walls are spam, with a large majority pointing to phishing sites. They also presented techniques to identify compromised accounts and spam campaigns. In a similar study on Twitter, Grier et al. [99] showed that at least 8% of links posted on Twitter are spam while 86% of the involved accounts are compromised. In contrast to this study, Thomas et al. [122] show that the majority of suspended accounts in Twitter are created by spammers as opposed to compromised users. All of these efforts however focus on post-mortem analysis of historical OSN data and are not applicable to MyPageKeeper's goal of identifying social malware soon after it appears on a user's wall or news feed.

Detecting spam accounts. Benevenuto et al. [87] and Yang et al. [129] developed techniques to identify accounts of spammers on Twitter. Others have proposed a honey-pot based approach [119, 103] to detect spam accounts on OSNs. Yardi et al. [130] analyzed behavioral patterns among spam accounts in Twitter. Instead of focusing on accounts created by spammers, MyPageKeeper enables social malware detection on the walls and news feeds of legitimate Facebook users.

Real-time spam detection in OSNs. Thomas et al. [123] developed Monarch, a real-time system that crawls URLs submitted from services such as Twitter to determine whether a URL directs to spam. Monarch relies on the network and domain level properties of URLs as well as the content of the web pages obtained when URLs are crawled. Interestingly, Monarch's classification accuracy is shown to be independent of the social context on Twitter. MyPageKeeper distinguishes itself from Monarch in several ways—1) we study social malware on Facebook, which we see significantly differs in its characteristics from traditional spam messages, 2) to make MyPageKeeper efficient, our social malware classifier operates without crawling of links found in posts, and 3) we find that the use of social context based features is crucial to efficient detection of social malware. In another study, Gao et al. [96] perform online spam filtering on OSNs using incremental clustering. Their technique however relies on having the whole social graph as input, and so, is usable only by the OSN provider. MyPageKeeper instead relies only on the view of the OSN as seen by MyPageKeeper's users. Lee et al. [104] built Warningbird, a system to detect suspicious URLs in Twitter; their system however relies on following the HTTP redirection chains of URLs, thus making their approach less efficient than MyPageKeeper.

Wang et al. [126] propose a unified spam detection framework that works across all OSNs, but they do not have an implementation of such a system in practice. Stein et al. [118] describe Facebook's Immune System (FIS), a scalable real-time adversarial learning system deployed in Facebook to protect users from malicious activities. However, Stein et al. provide only a high-level overview about threats to the Facebook graph and do not provide any analysis of the system. Similarly, other Facebook applications [16, 58, 13] that defend users against spam and malware are proprietary with no details available about how they work. Abu-Nimeh et al. [85] analyze the URLs flagged by one of these applications, Defenseio, but they do not discuss Defenseio's classification techniques and their analysis is restricted to that of the hosting infrastructure (country and ASN) underlying Facebook spam. To the best of our knowledge, we are the first to provide classification of social malware on Facebook that relies solely on social context based features, thus enabling MyPageKeeper to efficiently detect social malware at scale.

Social context based email spam. Jagatic et al. [100] discuss how email phishing attacks can be launched by using publicly available personal information (e.g., birthday) from social networks, and Brown et al. [90] analyzed such email spam seen in practice. However, due to revisions in Facebook's privacy policy over the last couple of years, only a user's friends have access to such information from the user's profile, thus making such email spam no longer possible. Further, MyPageKeeper focuses on spam propagated on Facebook rather than via email.

5.2 Indentifying Malicious Applications

App permission exploitation. Chia et al. [92] investigated the privacy intrusiveness of Facebook apps and concluded that currently available signals such as community ratings, popularity, and external ratings such as Web of Trust (WOT) as well as signals from app developers are not reliable indicators of the privacy risks associated with an app. Also, in keeping with our observation, they found that popular Facebook apps tend to request more permissions. They also found that 'Lookalike' applications that have names similar to popular applications request more permissions than is typical. Based on a measurement study across 200 Facebook users, Liu et al. [106] showed that privacy settings in Facebook rarely match users' expectations.

To address the privacy risks associated with the use of Facebook apps, some studies [88, 127] propose a new application policy and authentication dialog. Makridakis et al. [110] use a real application named 'Photo of the Day' to demonstrate how malicious apps on Facebook can launch DDoS attacks using the Facebook platform. King et al. [101] conducted a survey to understand users' interaction with Facebook apps. Similarly, Gjoka et al. [98] study the user reach of popular Facebook applications. On the contrary, we quantify the prevalence of malicious apps, and develop tools to identify malicious apps that use several features beyond the required permission set.

App rating efforts. Stein et al. [118] describe Facebook's Immune System (FIS), a scalable real-time adversarial learning system deployed in Facebook to protect users from malicious activities. However, Stein et al. provide only a high-level overview about threats to the Facebook graph and do not provide any analysis of the system. Furthermore, in an attempt to balance accuracy of detection with low false positives, it appears that Facebook has recently softened their controls for handling spam apps [32]. Other Facebook applications [16, 58, 13] that defend users against spam and malware do not provide ratings for apps on Facebook. Whatapp [81] collects community reviews about apps for security, privacy and openness. However, it has not attracted much reviews (47 reviews available) to date. To the best of our knowledge, we are the first to provide a classification of Facebook apps into malicious and benign categories.

Chapter 6

Conclusion

The emergence of Online Social Networks (OSNs) has opened up new possibilities for the dissemination of malware. As Facebook is becoming the new web, hackers are expanding their territory to Online Social Networks (OSNs) and spread social malware. Social malware is a new kind of cyber-threat, which requires novel security approaches. Cyber-fraud is an immediate and expensive problem that affects people and business through identity theft, the spread of viruses, and the creation of botnets, all of which are interconnected manifestations of Internet threats.

In this dissertation, we presented the design and implementation of MyPageKeeper, a Facebook application that can accurately and efficiently identify social malware at scale. Using data from over 12K Facebook users, we found that the reach of social malware is widespread —49% of MyPageKeeper's users were exposed at least once to social malware within four months. We also showed that existing defenses, such as URL blacklists, are ill-suited for identifying social malware, and that social malware significantly differs from email spam.

Interestingly, we found that a significant fraction of social malware is hosted on Facebook itself in the form of malicious Facebook apps and so on. However, little is understood about the characteristics of malicious apps and how they operate. Therefore, using a large corpus of malicious Facebook apps observed over a nine month period, we showed that malicious apps differ significantly from benign apps with respect to several features. For example, malicious apps are much more likely to share names with other apps, and they typically request fewer permissions than benign apps. Leveraging our observations, we developed FRAppE, an accurate classifier for detecting malicious Facebook applications. Most interestingly, we highlighted the emergence of AppNets—large groups of tightly connected applications that promote each other. We also made few recommendations to Facebook and hope that Facebook will benefit from these recommendations for reducing the menace of hackers on their platform.

Bibliography

[1] Active Facebook Scams to Avoid - July 4th, 2011. http://facecrooks.com/Safety-Center/Scam-Watch/Active-Facebook-Scams-to-Avoid-July-4th-2011.html.

[2] An overview of video scams on Facebook . http://www.helium.com/items/2212509-an-overview-of-video-scams-on-facebook.

[3] 100 social media statistics for 2012. http://thesocialskinny.com/100-social-media-statistics-for-2012/.

[4] 11 Million Bulk email addresses for sale - Sale Price $90. http://www.allhomebased.com/BulkEmailAddresses.htm.

[5] $500 free H&M gift card Facebook survey scam. https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_HnM_2012_03_8.

[6] Ads to blame for malware in Facebook's Farm Town? http://news.cnet.com/8301-27080_3-20002267-245.html.

[7] Anti-phishing working group. http://www.antiphishing.org/.

[8] App piggybacking example. https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_Converse_shoes_2012_05_17_boQ.

[9] AppData: Independent, Accurate Application Metrics and Trends from Inside Network. http://www.appdata.com.

[10] Application authentication flow using oauth 2.0. http://developers.facebook.com/docs/authentication/.

[11] BBC News: Identity 'at risk' on Facebook, 2008. http://news.bbc.co.uk/2/hi/programmes/click_online/7375772.stm.

[12] Bing gets friendlier with Facebook. http://www.technologyreview.com/web/37585/.

[13] Bitdefender Safego. http://www.facebook.com/bitdefender.safego.

[14] bit.ly API. http://code.google.com/p/bitly-api/wiki/ApiDocumentation.

[15] Cyber Attacks Jump 81 Percent, Target Mobile, Social Networks. http://www.pcmag.com/article2/0,2817,2403763,00.asp.

[16] Defensio Social Web Security. http://www.facebook.com/apps/application.php?id=177000755670.

[17] Discussion about selling facebook fan page. http://www.warriorforum.com/main-internet-marketing-discussion-forum/206355-can-you-sell-facebook-fan-pages.html.

[18] Drive-by Download Attack on Facebook Used Malicious Ads. http://www.pcworld.com/businesscenter/article/241164/driveby_download_attack_on_facebook_used_malicious_ads.html.

[19] Escrow-fraud. http://escrow-fraud.com/.

[20] Experts: Facebook crime is on the rise. http://www.zdnet.com/blog/facebook/experts-facebook-crime-is-on-the-rise/2632.

[21] Facebook birthday T-shirt scam steals secret mobile email addresses. http://nakedsecurity.sophos.com/2011/09/08/facebook-birthday-t-shirt-scam-steals-secret-mobile-email-addresses/.

[22] Facebook Cybercrimes Getting More Sophisticated. http://www.securitynewsdaily.com/facebook-cybercrimes-getting-more-sophisticated-1048/.

[23] Facebook developers. https://developers.facebook.com/docs/appsonfacebook/tutorial/.

[24] Facebook is the web's ultimate timesink. http://mashable.com/2010/02/16/facebook-nielsen-stats/.

[25] Facebook kills App Directory, wants users to search for apps. http://zd.net/MkBY9k.

[26] Facebook Opengraph API. http://developers.facebook.com/docs/reference/api/.

[27] Facebook Phishing Scam Costs Victims Thousands of Dollars. http://www.hyphenet.com/blog/2011/10/04/facebook-phishing-scam-costs-victims-thousands-of-dollars/.

[28] Facebook Platform Policies. https://developers.facebook.com/policy/.

[29] Facebook says 600,000 accounts compromised per day. http://mashable.com/2011/10/28/facebook-600000-accounts-compromised/.

[30] Facebook scam involves money transfers to the Philippines. http://profitscam.com/facebook-scam-involves-money-transfers-to-the-philippines-post/.

[31] Facebook Scam: Southwest Free Flights Will Comment Spam Your Friends. http://mashable.com/2011/02/22/facebook-scam-southwest/.

[32] Facebook softens its app spam controls, introduces better tools for developers. http://bit.ly/LLmZpM.

[33] Facebook stat 2011. http://www.kenburbary.com/2011/03/facebook-demographics-revisited-2011-statistics-2/.

[34] Facebook to Remove Reviews Page Tab App, Says New Tools Are Coming. http://www.insidefacebook.com/2011/10/07/remove-reviews-app/.

[35] Facebook tops 900 million users. http://money.cnn.com/2012/04/23/technology/facebook-q1/index.htm.

[36] Facebook users warned over 'free iPhone' scam. http://www.macworld.com/article/1154607/free_iphone_scam.html.

[37] Facebook Walmart Gift Card Scam. http://www.walmartstores.com/PrivacySecurity/10747.aspx?p=9620.

[38] Facebook Warnings About Lady Gaga Scam Go Viral. http://www.allfacebook.com/facebook-warnings-about-lady-gaga-scam-go-viral-2011-08.

[39] Fan Offer. https://www.facebook.com/apps/application.php?id=107611949261673.

[40] Fast-Flux Facebook Application Scams. http://www.symantec.com/connect/blogs/fast-flux-facebook-application-scams.

[41] FBML- Facebook Markup Language. https://developers.facebook.com/docs/reference/fbml/.

[42] FBML has been deprecated. https://developers.facebook.com/docs/reference/fbml/.

[43] Firefox JSSH extension. http://croczilla.com/bits_and_pieces/jssh/.

[44] FireWatir. http://code.google.com/p/firewatir/.

[45] Games. https://www.facebook.com/apps/application.php?id=121297667915814.

[46] goo.gl API. http://code.google.com/apis/urlshortener/v1/getting_started.html.

[47] Google Safe Browsing API. http://code.google.com/apis/safebrowsing/.

[48] Hackers selling $25 toolkit to create malicious Facebook apps. http://zd.net/g28HxI.

[49] How to spot a Facebook Survey Scam. http://facecrooks.com/Safety-Center/Scam-Watch/How-to-spot-a-Facebook-Survey-Scam.html.

[50] Jaccard index. http://en.wikipedia.org/wiki/Jaccard_index.

[51] Joewein: Fighting spam and scams on the Internet. http://www.joewein.net/.

[52] JSON: JavaScript Object Notation. http://www.json.org/.

[53] Latest Promotions. https://www.facebook.com/apps/application.php?id=174789949246851.

[54] Likejacking takes off on Facebook. http://www.readwriteweb.com/archives/likejacking_takes_off_on_facebook.php.

[55] Longest path problem. http://en.wikipedia.org/wiki/Longest_path_problem.

[56] MalwarePatrol- Malware is everywhere! . http://www.malware.com.br/.

[57] MyPageKeeper. https://www.facebook.com/apps/application.php?id=167087893342260.

[58] Norton Safe Web. http://www.facebook.com/apps/application.php?id=310877173418.

[59] Number of malicious Android apps grows by 2200% year over year. http://www.androidauthority.com/number-of-malicious-android-apps-grows-by-2200-year-over-year-86116/.

[60] One in 100 tweets and one in 60 Facebook posts are malicious. http://www.informationweek.com/news/security/client/231901895.

[61] Permissions Reference. https://developers.facebook.com/docs/authentication/permissions/.

[62] Phishtank. http://www.phishtank.com/.

[63] Pr0file stalker: rogue Facebook application. https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_pr0file_viewer_2012_4_4.

[64] REPORT: 15% Of Videos Posted On Facebook Are Actually Likejacking Attacks. http://www.scribbal.com/2011/09/report-15-of-videos-posted-on-facebook-are-actually-likejacking-attacks/.

[65] Rihanna video scam. "http://www.virteacon.com/2011/11/sick-i-just-hate-rihanna-after-watching.html".

[66] Selenium - Web Browser Automation. http://seleniumhq.org/.

[67] SocialBakers: The receipe for social marketing success. http://www.socialbakers.com/.

[68] Spamcop. http://www.spamcop.net/.

[69] Spamhaus. http://www.spamhaus.org/sbl/index.lasso.

[70] Stay Away From Malicious Facebook Apps. http://bit.ly/b6gWn5.

[71] Steve Jobs death scams are just the greedy exploiting the gullible. http://redandblack.com/2011/10/12/steve-jobs-death-scams-are-just-the-greedy-exploiting-the-gullible/.

[72] SURBL. http://www.surbl.org/.

[73] SVM Tutorials. http://svms.org/tutorials/.

[74] Symantec: Cybercrime costs $114 billion a year. http://news.yahoo.com/symantec-cybercrime-costs-114-billion-184901713.html.

[75] Team-Cymru. http://www.team-cymru.org/.

[76] The Impact of Facebook Scams and How Scammers Make Money. http://techie-buzz.com/social-networking/how-facebook-scams-spread-and-make-money.html?tb_spage=true.

[77] The Pink Facebook - rogue application and survey scam. http://nakedsecurity.sophos.com/2012/02/27/pink-facebook-survey-scam/.

[78] URIBL. http://www.uribl.com/.

[79] Web-of-trust. http://www.mywot.com/.

[80] What 20 Minutes On Facebook Looks Like. http://tcrn.ch/KytqzB.

[81] Whatapp (beta) - A Stanford Center for Internet and Society website with support from the Rose Foundation. https://whatapp.org/facebook/.

[82] Which cartoon character are you - rogue Facebook application. https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_whiich_cartoon_character_are_you_2012_03_30.

[83] Wiki: Facebook Platform. http://en.wikipedia.org/wiki/Facebook_Platform.

[84] Facebook becomes partner with Web of Trust (WOT). https://www.facebook.com/notes/facebook-security/keeping-you-safe-from-scams-and-spam/10150174826745766, May 2011.

[85] Saeed Abu-Nimeh, Thomas M. Chen, and Omar Alzubi. Malicious and spam posts in online social networks. In *IEEE Computer Society*, 2011.

[86] Facebook becomes partner with WebSense. http://www.thetechherald.com/article.php/201139/7675/Facebook-implements-malicious-link-scanning-service, Oct 2011.

[87] Fabricio Benevenuto, Gabriel Magno, Tiago Rodrigues, and Virgilio Almeida. Detecting spammers on Twitter. In *CEAS*, 2010.

[88] Andrew Besmer, Heather Richter Lipford, Mohamed Shehab, and Gorrell Cheek. Social applications: exploring a more secure framework. In *SOUPS*, 2009.

[89] Yazan Boshmaf, Ildar Muslukhov, Konstantin Beznosov, and Matei Ripeanu. The socialbot network: When bots socialize for fame and money. In *Proc. of the Annual Computer Security Applications Conference 2011*, 2011.

[90] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *ACM CSCW*, 2008.

[91] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.

[92] P. Chia, Y. Yamamoto, and N. Asokan. Is this app safe? a large scale study on application permissions and risk signals. In *WWW*, 2012.

[93] CVE-2007-0071. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-0071.

[94] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3), March 1964.

[95] Flare. http://www.nowrap.de/flare.html.

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

[97] Hongyu Gao, Jun Hu, Christo Wilson, Zhichun Li, Yan Chen, and Ben Y. Zhao. Detecting and characterizing social spam campaigns. In *IMC*, 2010.

[98] Minas Gjoka, Michael Sirivianos, Athina Markopoulou, and Xiaowei Yang. Poking facebook: characterization of osn applications. In *Proceedings of the first workshop on Online social networks*, WOSN, 2008.

[99] Chris Grier, Kurt Thomas, Vern Paxson, and Michael Zhang. @spam: The underground on 140 characters or less. In *CCS*, 2010.

[100] Tom N. Jagatic, Nathaniel A. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Commun. ACM*, 2007.

[101] Jennifer King, Airi Lampinen, and Alex Smolen. Privacy: Is there an app for that? In *SOUPS*, 2011.

[102] Anh Le, Athina Markopoulou, and Michalis Faloutsos. Phishdef: Url names say it all. In *Infocom*, 2010.

[103] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: social honeypots + machine learning. In *SIGIR*, 2010.

[104] S. Lee and J. Kim. Warningbird: Detecting suspicious urls in twitter stream. In *NDSS*, 2012.

[105] Kirill Levchenko, Andreas Pitsillidis, Neha Chachra, Brandon Enright, Mark Felegyhazi, Chris Grier, Tristan Halvorson, Chris Kanich, Christian Kreibich, He Liu, Damon McCoy, Nicholas Weaver, Vern Paxson, Geoffrey M. Voelker, and Stefan Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

[106] Yabing Liu, Krishna P. Gummadi, Balachander Krishnamurthy, and Alan Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *IMC*, 2011.

[107] Samuel Loirat. http://www.adopstools.com/.

[108] S. Jha M. Christodorescu and C. Kruegel. Mining specifications of malicious behavior. In *6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE*, pages 5–14, 2007.

[109] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *KDD*, 2009.

[110] Andreas Makridakis, Elias Athanasopoulos, Spiros Antonatos, Demetres Antoniades, Sotiris Ioannidis, and Evangelos P. Markatos. Understanding the behavior of malicious applications in social networks. *Netwrk. Mag. of Global Internetwkg.*, 2010.

[111] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes – which naive bayes? In *CEAS*, 2006.

[112] N. Provos, D. McNamee, P. Mavrommatis, K. Wang and N. Modadugu. The ghost in the browser analysis of web-based malware. In *HotBots*, 2007.

[113] Zhiyun Qian, Zhuoqing Morley Mao, Yinglian Xie, and Fang Yu. On network-level clusters for spam detection. In *NDSS*, 2010.

[114] R. M. H. Ting and J. Bailey. Mining Minimal Contrast Subgraph Patterns. In *6th SIAM International Conference on Data Mining*, pages 638–642, 2006.

[115] Md Sazzadur Rahman, Ting-Kai Huang, Harsha V. Madhyastha, and Michalis Faloutsos. Efficient and Scalable Socware Detection in Online Social Networks. In *USENIX Security*, 2012.

[116] S. Ford, M. Cova, C. Kruegel and G. Vigna. Analyzing and Detecting Malicious Flash Advertisements. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 1–10, Honolulu, HI, USA, Dec 7-11, 2009.

[117] Hispasec Sistemas. http://www.virustotal.com/.

[118] Tao Stein, Erdong Chen, and Karan Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, 2011.

[119] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. Detecting spammers on social networks. In *ACSAC*, 2010.

[120] SWFIntruder. http://www.owasp.org/index.php/Category:SWFIntruder.

[121] SWFScan. https://h30406.www3.hp.com/campaigns/2009/wwcampaign/1-5TUVE/index.php?key=swf&jumpid=go/swfscan.

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

[123] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

[124] Kurt Thomas and David Nicol. The koobface botnet and the rise of social malware. In *Malware*, 2010.

[125] Alex Wang. Detecting Spam Bots in Online Social Networking Sites: A Machine Learning Approach. In *Data and Applications Security and Privacy XXIV*. 2010.

[126] De Wang, Danesh Irani, and Calton Pu. A social-spam detection framework. In *CEAS*, 2011.

[127] Na Wang, Heng Xu, and Jens Grossklags. Third-party apps on facebook: privacy and the illusion of control. In *CHIMIT*, 2011.

[128] Wepawet. http://wepawet.cs.ucsb.edu/.

[129] Chao Yang, Robert Harkreader, and Guofei Gu. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *RAID*, 2011.

[130] S. Yardi, D. Romero, G. Schoenebeck, et al. Detecting spam in a twitter network. *First Monday*, 2009.

[131] Sarita Yardi, Daniel Romero, Grant Schoenebeck, and D Boyd. Detecting spam in a twitter network. 2010.

[132] Yajin Zhou and Xuxian Jiang. Dissecting Android Malware: Characterization and Evolution. In *IEEE Security and Privacy*, 2012.