Detecting Malicious Facebook Applications

Sazzadur Rahman, Ting-Kai Huang, Harsha V. Madhyastha, and Michalis Faloutsos

***Abstract*—With 20 million installs a day [1], third-party apps are a major reason for the popularity and addictiveness of Facebook. Unfortunately, hackers have realized the potential of using apps for spreading malware and spam. The problem is already significant, as we find that at least 13% of apps in our dataset are malicious. So far, the research community has focused on detecting malicious posts and campaigns. In this paper, we ask the question: Given a Facebook application, can we determine if it is malicious? Our key contribution is in developing FRAppE—Facebook's Rigorous Application Evaluator—arguably the first tool focused on detecting malicious apps on Facebook. To develop FRAppE, we use information gathered by observing the posting behavior of 111K Facebook apps seen across 2.2 million users on Facebook. First, 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. Second, leveraging these distinguishing features, we show that FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and a high true positive rate (95.9%). Finally, we explore the ecosystem of malicious Facebook apps and identify mechanisms that these apps use to propagate. Interestingly, we find that many apps collude and support each other; in our dataset, we find 1584 apps enabling the viral propagation of 3723 other apps through their posts. Long term, we see FRAppE as a step toward creating an independent watchdog for app assessment and ranking, so as to warn Facebook users before installing apps.**

***Index Terms*— Facebook apps, malicious, online social networks, spam.**

I. Introduction

ONLINE social networks (OSNs) 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 [2] that facilitates app integration into the Facebook user-experience. There are 500K apps available on Facebook [3], and on average, 20M apps are installed every day [1]. Furthermore, many apps have acquired and maintain a really 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 [4]–[6]. Malicious apps can provide a lucrative business for hackers, given the popularity of OSNs, with Facebook leading the way with 900M active users [7]. There are many ways that hackers can benefit from a malicious app: 1) the app can reach large numbers of users and their friends to spread spam; 2) the app can obtain users' personal information such as e-mail address, home town, and gender; and 3) the app can "reproduce" 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 [8]. In other words, there is motive and opportunity, and as a result, there are many malicious apps spreading on Facebook every day [9].

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 the following: 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 Section III, 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 [10]–[12]. At the same time, in a seemingly backwards step, Facebook has dismantled its app rating functionality recently. A recent work studies how app permissions and community ratings correlate to privacy risks of Facebook apps [13]. Finally, there are some community-based feedback-driven efforts to rank applications, such as WhatApp? [14]; though these could be very powerful in the future, so far they have received little adoption. We discuss previous work in more detail in Section VIII.

In this paper, 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, a security app in Facebook [15] that monitors the Facebook profiles of 2.2 million users. We analyze 111K apps that made 91 million posts over 9 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.

Manuscript received December 05, 2013; revised June 05, 2014 and November 09, 2014; accepted December 11, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Teixeira. This work was supported by NSF SaTC 1314935 and NSF NETS 0721889.

S. Rahman was with the Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA 92507 USA. He is now with Qualcomm Research, San Diego, CA 92126 USA (e-mail: rahmanm@cs.ucr.edu).

T.-K. Huang was with the Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA 92507 USA. He is now with Google, Mountain View, CA 94043 USA (e-mail: huangt@cs.ucr.edu).

H. V. Madhyastha is with the University of Michigan, Ann Arbor, MI 48109 USA (e-mail: harshavm@umich.edu)

M. Faloutsos is with the University of New Mexico, Albuquerque, NM 87131 USA (e-mail: michalis@cs.unm.edu)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2014.2385831

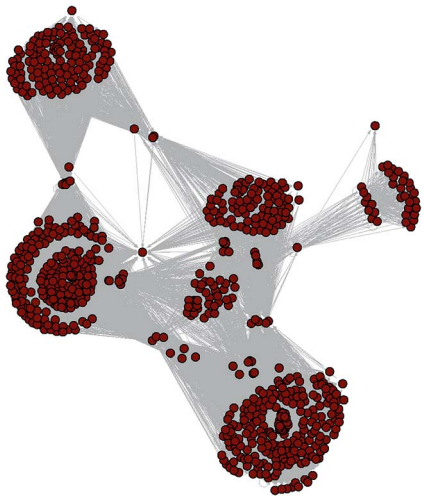


Fig. 1. Emergence of app-nets on Facebook. Real snapshot of 770 highly collaborating apps: An edge between two apps means that one app helped the other propagate. Average degree (number of collaborations) is 195.

- *13% of 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 1000 monthly active users each.
- *Malicious and benign app profiles significantly differ*. 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: 1) 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 2) 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).
- *The emergence of app-nets: 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. We find 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 1584 promoter apps that promote 3723 other apps. Furthermore, these apps form large and highly dense connected components, as shown in Fig. 1.
  Furthermore, hackers use fast-changing indirection: Applications posts have URLs that point to a Web site, and the Web site dynamically redirects to many different apps; we find 103 such URLs that point to 4676 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 app-net, 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.
- *FRAppE can detect malicious apps with 99% accuracy*. We develop FRAppE (Facebook's Rigorous Application Evaluator) to identify malicious apps using 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 high true positives (95.6%). By adding aggregation-based information, FRAppE can detect malicious apps with 99.5% accuracy, with no false positives and higher true positives (95.9%).

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.

1) *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. Note that we only suggested against a special kind of app promotion where the user clicks the app A installation icon, app A redirects the user to the intermediate installation page of app B, and the user cannot see the difference unless she examines the landing URL very carefully where client ID is different. At the end, the user ends up installing app B although she intended to install app A. Moreover, cross promotion among apps is forbidden as per Facebook's platform policy [16].
2) *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.

II. Background

We discuss how applications work on Facebook, and we outline the datasets that we use in this paper.

A. 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, the user grants the application server: 1) permission to access a subset of the information listed on the user's Facebook profile (e.g., the user's e-mail address), and 2) 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 [17] 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

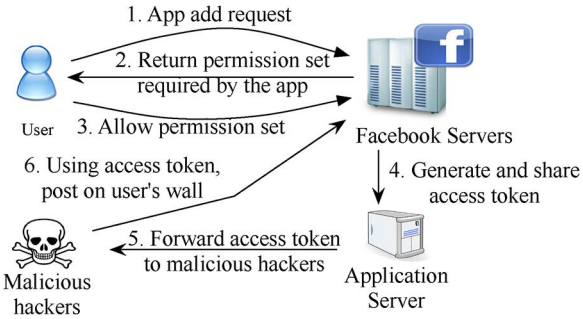


Fig. 2. Steps involved in hackers using malicious applications to get access tokens to post malicious content on victims' walls.

of the user. Fig. 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 [18] to eventually profit the hackers.

B. Our Datasets

The basis of our study is a dataset obtained from 2.2M Facebook users, who are monitored by MyPageKeeper [15], our security application for Facebook.[1] MyPageKeeper evaluates every URL that it sees on any user's wall or news feed to determine if that URL points to social spam. MyPageKeeper classifies a URL as social spam if it points to a Web page that: 1) spreads malware; 2) attempts to "phish" for personal information; 3) requests the user to carry out tasks (e.g., fill out surveys) that profit the owner of the Web site; 4) promises false rewards; or 5) attempts to entice the user to artificially inflate the reputation of the page (e.g., forcing the user to "Like" the page to access a false reward). MyPageKeeper evaluates each URL using a machine-learning-based classifier that leverages the *social context* associated with the URL. For any particular URL, the features used by the classifier are obtained by combining information from all posts (seen across users) containing that URL. Example features used by MyPageKeeper's classifier include the similarity of text message across posts and the number of comments/Likes on those posts. MyPageKeeper has false positive and false negative rates of 0.005% and 3%. For more details about MyPageKeeper's implementation and accuracy, we refer interested readers to [10].

Our dataset contains 91 million posts from 2.2 million walls monitored by MyPageKeeper over 9 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 I.

[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 [19].

TABLE I
SUMMARY OF THE DATASET COLLECTED BY MYPAGEKEEPER FROM JUNE 2011 TO MARCH 2012

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	6,063	3,227
D-Complete	2,255	487

TABLE II
TOP MALICIOUS APPS IN D-SAMPLE DATASET

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

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. By applying this heuristic, we identified 6350 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 VI-E. To avoid such misclassifications, 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 6273 malicious applications (D-Sample dataset in Table I). Table II shows the top five malicious applications, in terms of number of posts per application. Although we infer the ground truth data about malicious applications from MyPageKeeper, it is possible that MyPageKeeper itself has potential bias classifying malicious app's posts. For example, if a malicious application is very unpopular and therefore does not appear in many users' walls or news feeds, MyPageKeeper may fail to classify it as malicious (since it works on post level). However, as we show here later, our proposed system uses a different set of features than MyPageKeeper and can identify even very unpopular apps with high accuracy and low false positives and false negatives.

Fig. 3 shows the number of new malicious apps seen in every month of the D-Sample dataset. For every malicious app in the D-Sample dataset, we consider the time at which we observed the first post made by this app as the time at which the app was launched. We see that hackers launch new malicious apps every month in Facebook, although September 2011, January 2012, and February 2012 see significantly higher new malicious app activity than other months. Out of the 798 malicious apps launched in September 2011, we find 355 apps all created with the name "The App" and 116 apps created with the name "Profile Viewing." Similarly, of the 3813 malicious apps created in February 2012, 985 and 589 apps have the name "Are You Ready" and "Pr0file Watcher," respectively. Other examples of

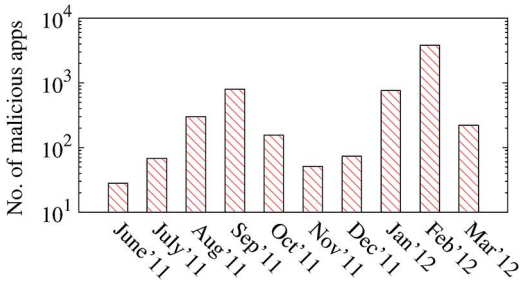


Fig. 3. Malicious apps launched per month in D-Sample dataset.

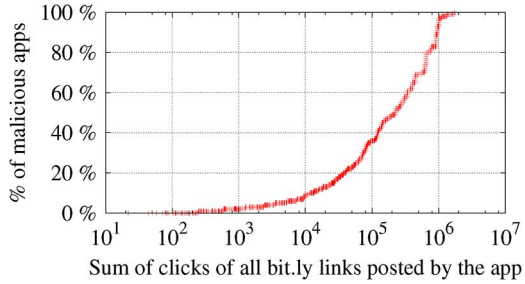


Fig. 4. Clicks received by `bit.ly` links posted by malicious apps.

app names used often are "What does your name mean?," "Fortune Teller," "What is the sexiest thing about you?," and so on.

D-Sample Dataset: Including Benign Applications: To select an equal number of benign apps from the initial D-Total dataset, we use two criteria: 1) none of their posts were identified as malicious by MyPageKeeper, and 2) they are "vetted" by Social Bakers [20], which monitors the "social marketing success" of apps. This process yields 5750 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 6273 benign applications. The D-Sample dataset (Table I) is the union of these 6273 benign applications with the 6273 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 [21]. 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.

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 6067 benign and 2528 malicious apps (D-Summary dataset in Table I). It is easy to understand why malicious apps were more often removed from Facebook.

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 2255 benign applications in our dataset. Automatically crawling the permissions for all apps is not trivial [13], 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.

D-ProfileFeed Dataset: Posts on App Profiles: 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 6063 benign and 3227 malicious apps. We construct the D-Complete dataset by taking the intersection of D-Summary, D-Inst, and D-ProfileFeed datasets.

III. 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 a lower bound on the number of clicks on 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 [22] 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.

Across the posts made by the 6273 malicious apps in the D-Sample dataset, we found that 3805 of these apps had posted 5700 `bit.ly` URLs in total. We queried `bit.ly` for the click count of each URL. Fig. 4 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. Although it would be interesting to find the `bit.ly` click-through rate per user and per post, we do not have data for the number of users who saw these links. We can query `bit.ly`'s API only for the number of clicks received by a link.

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. 5 plots the distribution of Monthly Active Users of the malicious apps in our D-Summary dataset. For each app, the median and

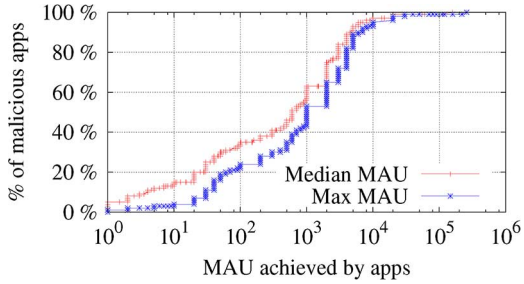


Fig. 5. Median and maximum MAU achieved by malicious apps.

maximum MAU values over the three months are shown. We see that 40% of malicious applications had a median MAU of at least 1000 users, while 60% of malicious applications achieved at least 1000 during the 3-month observation period. The top malicious app here—"Future Teller"—had a maximum MAU of 260 000 and median of 20 000.

IV. Profiling Applications

Given the significant impact that malicious apps have on Facebook, we next seek to develop a tool that can identify malicious applications. Toward 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 II-B, 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.

A. 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.

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. 6 shows the fraction of malicious and benign apps in the D-Summary dataset for which these three fields are nonempty. 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 nonempty 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).

2) Required Permission Set: 97% of malicious apps require only one permission from users. Every Facebook app requires authorization by a user before the user can use it. At the time

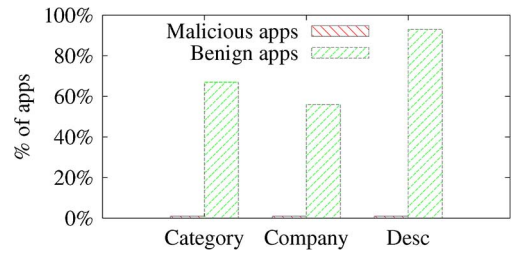


Fig. 6. Comparison of apps based on information in app summary.

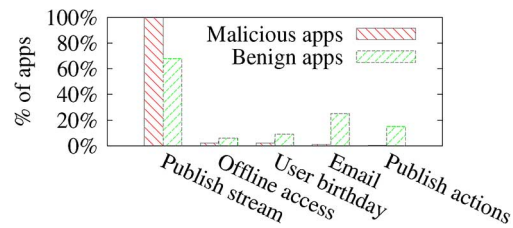


Fig. 7. Top five permissions required by benign and malicious apps.

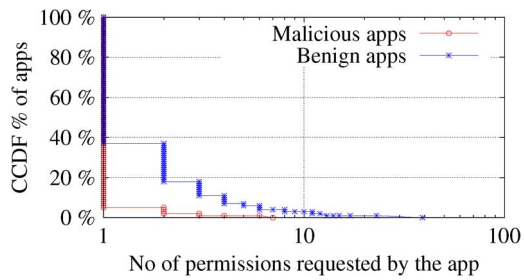


Fig. 8. Distribution of number of permissions requested by apps.

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 predefined by Facebook [23]. Example permissions include access to information in the user's profile (e.g., gender, e-mail, birthday, and friend list), and permission to post on the user's wall.

We see how malicious and benign apps compare based on the permission set that they require from users. Fig. 7 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. 8 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 a 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 [24]. Therefore, to maximize the number of victims, malicious apps seem to follow this hypothesis and require a small set of permissions.

3) Redirect URI: Malicious apps redirect users to domains with poor reputation. In an application'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 [25]. Fig. 9 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

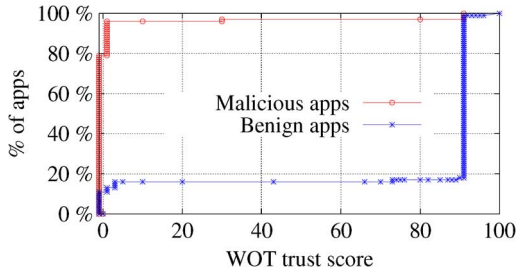


Fig. 9. WOT trust score of the domain that apps redirect to upon installation.

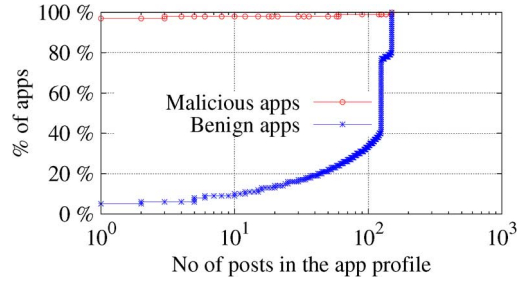


Fig. 10. Number of posts in app profile page.

TABLE III
TOP FIVE DOMAINS HOSTING MALICIOUS APPS IN D-INST DATASET

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

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 a further 8% 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 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 e-mail 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 III shows the top five such domains; these five domains host the content for 83% of the malicious apps in the D-Inst dataset.

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 the app. Facebook then redirects the user to a URL that 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 that she will end up installing is the value of the client ID parameter. Ideally, as described in the Facebook app developer tutorial [24], 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 Section VI, 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.

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 Section II-B, we were able to crawl the app profile pages for 3227 malicious apps and 6063 benign apps.

From Fig. 10, 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.

B. 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 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.

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 [26] between the two

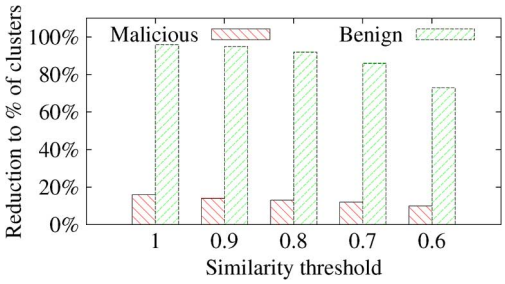


Fig. 11. Clustering of apps based on similarity in names.

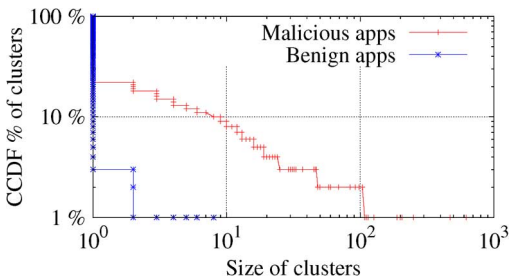


Fig. 12. Size of app clusters with identical names.

names and normalize this distance with the maximum of the lengths 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. 11 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, five malicious apps have the same name. Fig. 12 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. 12, most benign apps have unique names.

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

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

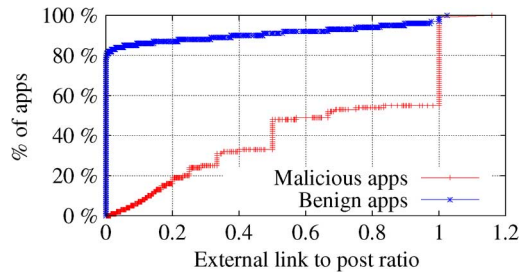


Fig. 13. Distribution of external links to post ratio across apps.

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 9-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 an "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. 13 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.

V. 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.

A. FRAppE Lite

FRAppE Lite is a lightweight version that 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 IV 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) [27] classifier for classifying malicious apps. SVM is widely used for binary classification in security and other disciplines [28], [29]. We use the D-Complete dataset for training and testing the classifier. As shown earlier in Table I, the D-Complete dataset consists of 487 malicious apps and 2255 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 true positive (TP) rate as the three metrics to measure the

TABLE IV
LIST OF FEATURES USED IN FRAPPE LITE

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 V
CROSS VALIDATION WITH FRAPPE LITE

Training Ratio	Accuracy	FP	TP
1:1	98.5%	0.6%	97.5%
4:1	99.0%	0.1%	95.3%
7:1	99.0%	0.1%	95.6%
10:1	99.5%	0.1%	94.5%

TABLE VI
CLASSIFICATION ACCURACY WITH INDIVIDUAL FEATURES

Feature	Accuracy	FP	TP
Category specified?	76.5%	45.8%	98.8%
Company specified?	72.1%	55.0%	99.2%
Description specified?	97.8%	3.3%	99.0%
Posts in profile?	96.9%	4.3%	98.1%
Client ID is same?	88.5%	1.0%	78.0%
WOT trust score	91.9%	13.4%	97.1%
Permission count	73.3%	49.3%	95.9%

TABLE VII
ADDITIONAL FEATURES USED IN FRAPPE

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 VIII
VALIDATION OF APPS FLAGGED BY FRAPPE

Criteria	# of apps validated	Cumulative
Deleted from FB 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 apps	5(0.1%)	7,912(97%)
Manual validation	147 (1.8%)	8051 (98.5%)
Total validated	-	8051(98.5%)
Unknown	-	93 (1.5%)

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 rate is the fraction of benign apps incorrectly classified as malicious, and true positive rate is the fraction of benign and malicious apps correctly classified (i.e., as benign and malicious, respectively).

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 V shows that, irrespective of the ratio of benign to malicious apps, the accuracy is above 98.5%. The higher the ratio 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 rate is below 0.6% and true positive rate is above 94.5% 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, 6273 apps were identified as malicious based on MyPageKeeper's classification of posts, and an additional 8051 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 95.6% true positives in practice.

To understand the contribution of each of FRAppE Lite's features toward its accuracy, we next perform 5-fold cross validation on the D-Complete dataset with only a single feature at a time. Table VI 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 a high true positive rate (99.0%). 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 low true positive rate.

B. FRAppE

Next, we consider FRAppE—a malicious app detector that utilizes our aggregation-based features in addition to the on-demand features. Table VII 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% (true positive rate 95.1% and true negative rate 100%), as compared to 99.0% with FRAppE Lite. Furthermore, the true positive rate increases from 95.6% to 95.9%, and we do not have a single false positive.

C. 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, 8144 apps were flagged as malicious 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 VIII, 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 1464 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 4 months with 10% dating back to over 8 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 a 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." Note that we used "typosquatting" criteria only to validate apps that were already classified as malicious by FRAppE. We did not use this feature as standalone criteria for classifying malicious apps in general. Moreover, it could only validate 0.5% of apps in our experiment as shown in Table VIII.

Manual Verification: For the remaining 232 apps unverified by the above techniques, we 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.

D. Representativeness of Ground Truth for Benign Apps

We demonstrate the representativeness of benign apps used in our ground truth data set in the following ways. First, we selected 6000 apps randomly from 91 000 apps in our dataset and compared the median MAU to that of 6000 benign apps in our ground truth dataset. As shown in Fig. 14, benign apps have median MAUs distributed across a wide range similar to the MAUs of randomly selected apps. Second, we tested FRAppE on two different sets of benign apps (1125 apps in each set), where one set had significantly more popular apps (median MAU 20 000) than the other (median MAU 500). We repeated 5-fold cross validation on each set independently and found that the false positive rate showed only a marginal increase from 0% in the case of popular apps to 0.18% for unpopular apps. Thus, FRAppE's accuracy is not biased by the popularity of apps in our dataset of benign apps.

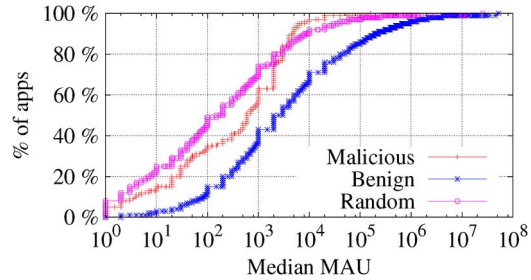


Fig. 14. MAU comparison among malicious, benign, and randomly selected apps.

VI. Malicious Apps Ecosystem

Our analysis in Section III shows that malicious apps are rampant on Facebook and indicates that they do not operate in isolation. Indeed, we find that malicious apps collude at scale—many malicious apps share the same name, several of them redirect to the same domain upon installation, etc. These observed behaviors indicate well-organized crime, with a few prolific hacker groups controlling many malicious apps.

A common way in which malicious apps collude is by having one app post links to the installation page of another malicious app. In this section, we conduct a forensics investigation on the malicious app ecosystem to identify and quantify the techniques used in this cross promotion of malicious apps.

A. Background on App Cross Promotion

Cross promotion among apps, which is forbidden as per Facebook's platform policy [16], 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 any one of a set of apps.

Posting Direct Links to Other Apps: We found evidence that malicious apps often promote each other by making posts that redirect users to the promotee's app page; 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. Promoter apps make such posts on the walls of users who have been tricked into installing these apps. These posts then appear in the news feed of the victim's friends. The post contains an appropriate message to lure users to install the promoted app, thereby enabling the promotee to accumulate more victims. To study such cross promotion, we crawled the URLs posted by all malicious apps in our dataset and identified those where the landing URL corresponds to an app installation page; we extracted the app ID of the promotee app in such cases. In this manner, we find 692 promoter apps in our D-Sample dataset from Section II, which promoted 1806 different apps using direct links.

Indirect App Promotion: Alternatively, hackers use Web sites 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 Web site outside Facebook [30]. This external Web site 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. In the

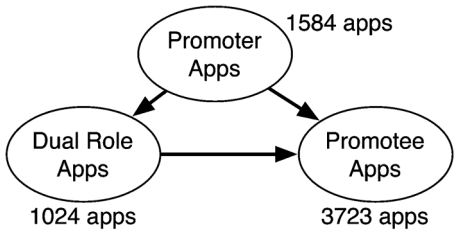


Fig. 15. Relationship between collaborating applications.

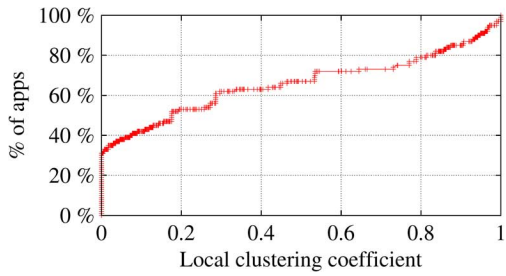


Fig. 16. Connectivity of apps in the collaboration graph.

course of MyPageKeeper's operation, if we find any shortened URL points to an app installation URL (using an instrumented browser), we mark the URL as a potential indirection URL. Then, we crawl such potential indirection URL five times. If it redirects more than one landing URL, we mark it as an indirection URL. In this approach, we identified 103 indirection Web sites in our dataset of colluding apps. Now, to identify all the landing Web sites, for one and a half months from mid-March to the end of April 2012, we followed each indirection Web site 100 times a day using an instrumented Firefox browser. We discover 4676 different malicious apps being promoted via the 103 indirection Web sites.

B. Promotion Graph Characteristics

From the app promotion dataset we collected above, we construct a graph that has an undirected edge between any two apps that promote each other via direct or indirect promotion, i.e., an edge between *app*1 and *app*2 if the former promotes the latter. We refer to this graph as the "Promotion graph."

1) Different Roles in Promotion Graph: Apps act in different roles for promotion. The "Promotion graph" contains 6331 malicious apps that engage in collaborative promotion. Among them, 25% are *promoters*, 58.8% are *promotees*, and the remaining 16.2% play both roles. Fig. 15 shows this relationship between malicious apps.

2) Connectivity: Promotion graph forms large and densely connected groups. We identified 44 connected components among the 6331 malicious apps. The top five connected components have large sizes: 3484, 770, 589, 296, and 247. Upon further analysis of these components, we find the following.

- *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[2] larger than 0.74, as shown in Fig. 16.

[2]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.

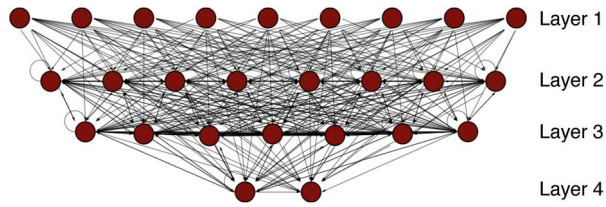


Fig. 17. Example of collusion graph between applications.

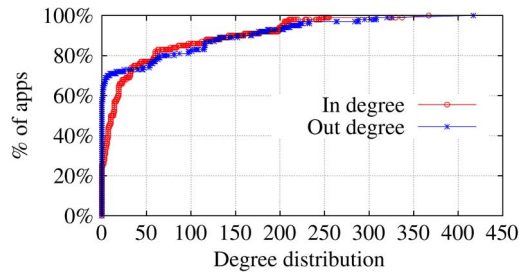


Fig. 18. Degree distribution of apps in promotion graph.

We use the term app-net to refer to each connected component in the Promotion graph. As an example of an app-net, Fig. 17 shows the local neighborhood of the "Death Predictor" app, which has 26 neighbors and has a local clustering coefficient of 0.87. Interestingly, 22 of the node's neighbors share the same name.

3) Degree Distribution: To understand the relationship between promoter and promotee apps, we create a directed graph, where each node is an app, and an edge from *app*1 to *app*2 indicates that *app*1 promotes *app*2. Fig. 18 shows the in-degree and out-degree distribution across the nodes in this graph. We can see that 20% of apps have in-degree or out-degree more than 50, which shows that 20% of the apps have been promoted by at least 50 other apps and another 20% of the apps have each promoted 50 other apps.

4) Longest Chain in Promotion: App-nets often exhibit long chains of promotion. We are interested in finding the longest path of promotion in this directed graph. However, finding a simple path of maximum length in a directed cyclic graph is an NP-complete problem [31]. Therefore, we approximate this with a depth-first-based search that terminates after a threshold runtime. Fig. 19 shows the distribution of the longest path starting from all the apps that are seen only as promoters, i.e., they are never promoted by other apps. We see that the longest path of promotion is 193, and 40% of promoter-only apps have a longest path at least 20. In such paths, at most 17 distinct app names were used, as shown in Fig. 20, 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 promotes "who Are They? v4," which in turn promotes "Secret Lurkers v5.73," and so on.

5) Participating App Names in Promotion Graph: Apps with the same name often are part of the same app-net. The 103 indirection Web sites that we discovered in Section VI-A were used by 1936 promoter apps that had only 206 unique app names. The promotees were 4676 apps with 273 unique app names. Clearly, there is a very high reuse of both names and these indirection Web sites. For example, one indirection Web site distributed in posts by the app "whats my name means" points to the installation page of the apps "What ur name implies!!!," "Name

TABLE IX
GRAPH PROPERTIES

Graph	# Nodes	# of Edges	Avg. clustering co-efficient	# of Connected components	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%

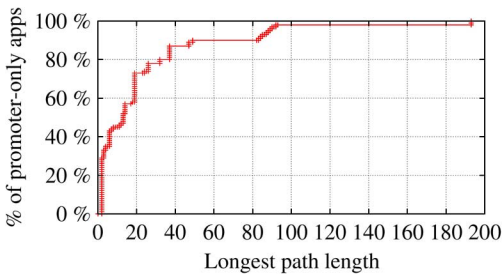


Fig. 19. Longest paths from promoter-only apps in promotion graph.

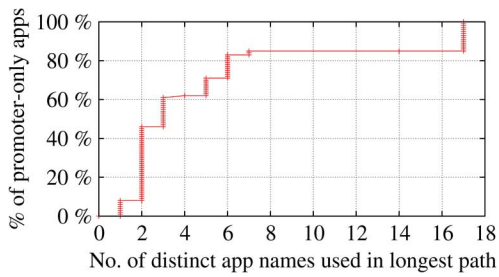


Fig. 20. Number of distinct app names used in the longest path from promoter-only apps in promotion graph.

meaning finder," and "Name meaning." Furthermore, 35% of these Web sites promoted more than 100 different applications each. Following the discussion in Section IV-B.1, it appears that hackers often reuse the same names for their malicious applications. We speculate that the reason for this trend is as follows: Since all apps underlying a campaign have the same name, if any app in the pool gets blacklisted, others can still survive and carry on the campaign without being noticed by users.

C. App Collaboration

Next, we attempt to identify the major hacker groups involved in malicious app collusion. For this, we consider different variants of the "Campaign graph" as follows.

- *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 the same domain once they are installed by a user. We exclude apps that redirect to apps.facebook.com.
- *Promoted URL campaign*: Two apps are part of a campaign if they are promoted by the same indirection URL.

It is important to note that, in all versions of the Campaign graph, the nodes in the same campaign form a clique.

Table IX shows the properties—average clustering coefficient,[3] diameter,[4] and giant connected component (GCC)[5]—of the different variants of the Campaign graph and of the Promotion graph.

TABLE X
TOP FIVE DOMAINS ABUSED FOR FAST FLUX INFRASTRUCTURE

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

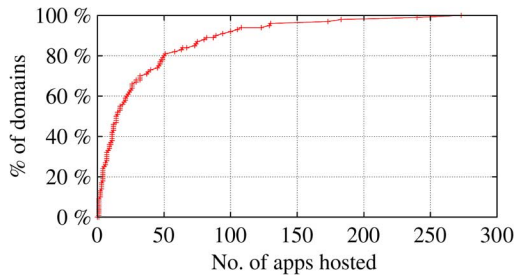


Fig. 21. Characterizations of domains hosting malicious apps.

Finally, we construct the "Collaboration graph" by considering the union of the "Promotion graph" and all variants of the "Campaign graph." We find that the Collaboration graph has 41 connected components, with the GCC containing 56% of nodes in the graph. This potentially indicates that 56% of malicious apps in our corpus are controlled by a single malicious hacker group. The largest five component sizes are 3617, 781, 645, 296, and 247.

D. Hosting Domains

We investigate the hosting domains that enables redirection Web sites. First, we find that most of the links in the posts are shortened URLs, and 80% of them use 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 X shows the top five domains that host these indirection Web sites. Second, we find that 20% of the domains hosting malicious apps each host at least 50 different apps, as shown in Fig. 21. Table XI shows the name of the top five hosting domains and the number of apps they host. This shows that hackers heavily reuse domains for hosting malicious apps.

E. App Piggybacking

From our dataset, we also discover that hackers have found ways to make malicious posts appear as if they had been

[3]Average clustering coefficient is the average of the local clustering coefficient of all nodes.

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

[5]GCC is the fraction of nodes that comprise the largest connected component in the graph.

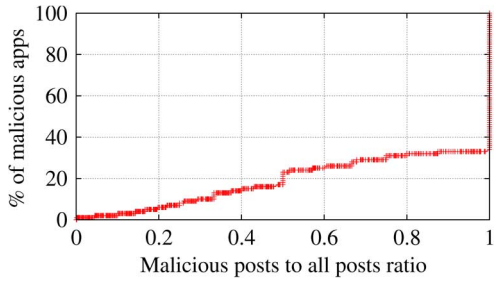


Fig. 22. Distribution of the fraction of an app's posts that are malicious.

TABLE XI
TOP FIVE 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

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 in [32].

We identify instances of app piggybacking in our dataset as follows. For every 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. 22, 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 XII shows the top five most popular apps that we find among this set.

F. Cross Promotion as a Sign of Malicious Intentions

Thus far, we studied cross promotion among malicious apps based on posts marked as malicious by MyPageKeeper. However, MyPageKeeper may have failed to flag the posts of many malicious apps. Therefore, here we study the prevalence of cross promotion simply by observing whether the post made by an app includes a URL that points to another app. This enables us to discover a new set of malicious apps that we have failed to identify so far.

1) Data Collection: Cross promotion via posted apps.facebook.com URL: The simplest way to identify whether a post made by an app is pointing to an app's page is to examine if the URL in the post points to the apps.facebook.com domain. We collect 41M URLs monitored by MyPageKeeper that point to the apps.facebook.com/namespace domain. The posts containing these URLs were posted by 13 698 distinct apps. We then identify the app ID corresponding to each *namespace*, and thus identify cross-promotion relationships between promoter and promotee apps. After ignoring self-promotion where one app promotes itself, we identify 7700 cross-promoting relations involving 4782 distinct apps.

Cross promotion via posted shortened URLs: The above method however does not suffice for identifying all instances of cross promotion since many apps post shortened URLs. To investigate app promotion via shortened URLs, we collect 5.8M shortened links monitored by MyPageKeeper, out of which 65 448 URLs resolve to the apps.facebook.com domain. Applying similar techniques as mentioned above, we identify an additional 1177 cross-promoting relations involving 450 distinct apps.

In total, we found 5077 distinct apps involved in 8069 cross promotions. As per Facebook's platform policy, all of these 5077 apps are violating policy. Intrigued, we investigate them further.

2) Analyzing Cross-Promoting Apps: To identify malicious apps from the 5077 apps, we compare them to our corpus of 14K malicious apps identified by FRAppE in Section V. We consider both apps in a promoter–promotee relationship malicious if either of them appear in our malicious app corpus. This enables us to identify an additional 2052 malicious apps. However, the rest of the 3025 apps are not connected to FRAppE-detected malicious apps.

Table XIII shows the properties of graphs that represent the cross-promotion relationships among malicious apps and the remaining unverified apps. As shown in the table, malicious apps are tightly connected since the largest connected component contains 91% of the apps.

For the cross promotions among unverified apps, the largest five component sizes are 972, 270, 174, 136, and 103. Fig. 23 shows the distribution of sizes of the 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. 24, 90% and 85% of apps have in-degree and out-degree not more than one. However, few apps have very high in-degree or out-degree. For example, the app "Quiz Whiz" belongs to a family of quiz apps, and they often promote each other. Few example apps in this quiz family are: "Which Christmas Character are you?" and "what twilight vampire are you?"

Next, we study the popularity of these apps in terms of MAUs. For this, we crawl their app summary from the Facebook Openi Graph. Out of the 3026 unverified apps, we find that 702 apps have been deleted from Facebook's social graph. Fig. 25 shows the popularity of the remaining 2324 apps in terms of MAU. We see that few apps are very popular since they have MAU of several millions. For example, two different apps with the name "Daily Horoscope" promote each other and have MAUs of 9.7M and 1.4M users. Furthermore, we find 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 their user base.

VII. DISCUSSION

In this section, we discuss potential measures that hackers can take to evade detection by FRAppE. We also present

TABLE XII
TOP FIVE POPULAR APPS BEING ABUSED BY APP PIGGYBACKING

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 XIII
CROSS-PROMOTING APP GRAPH PROPERTIES

Graph	Nodes	Edges	Components	GCC
Malicious	2,052	3,542	22	91%
Unverified	3,026	4,527	370	32%

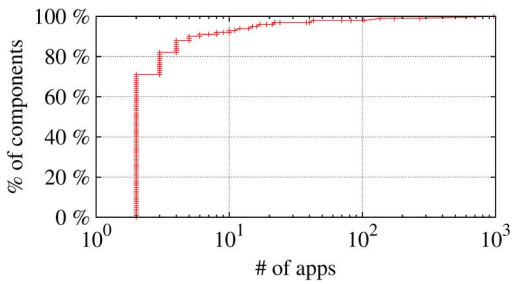


Fig. 23. Distribution of components among unverified cross-promoting apps.

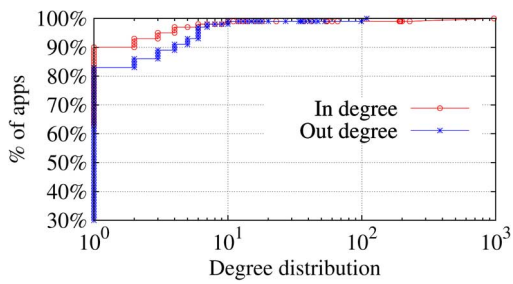


Fig. 24. Degree distribution of unverified cross-promoting apps.

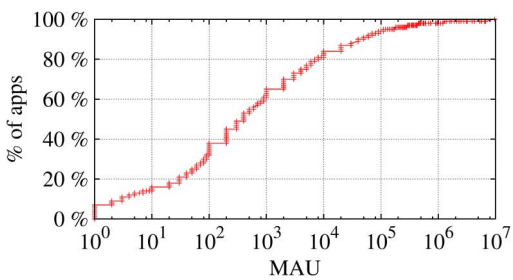


Fig. 25. MAU distribution of unverified cross-promoting apps.

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.

TABLE XIV
INCREMENTAL CUMULATIVE CLASSIFICATION ACCURACY IN ORDER OF FEATURE ROBUSTNESS

Feature	C.Accuracy	C.FP	C.TP
Client ID is same?	88.2%	1.6%	78.0%
WOT trust score	93.8%	9.5%	97.1%
Permission count	94.3%	6.4%	94.9%
App name similarity	97.2%	0.0%	94.5%
External link to post ratio	98.5%	0.4%	97.3%
Posts in profile?	98.9%	0.6%	98.4%
Description specified?	99.2%	0.4%	98.8%
Company specified?	99.2%	0.2%	98.6%
Category specified?	99.2%	0%	98.4%

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. Table XIV shows incremental cumulative accuracy (C.Accuracy), cumulative false positive (C.FP), and cumulative true positive (C.TP) when we perform 5-fold cross validation on the D-Complete dataset with a 1:1 ratio of benign to malicious apps. As shown in the table, we find that a version of FRAppE that only uses the first three robust features still yields an accuracy of 94.3%, with false positive and true positive rates of 6.4% and 94.9%, respectively.

Recommendations to Facebook: Our investigations of malicious apps on Facebook identified two key loopholes in Facebook's API that hackers take advantage of. First, as discussed in Section IV-A.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. Second, Facebook should restrict users from using arbitrary app IDs in their prompt feed API: http://www.facebook.com/connect/prompt_feed.php?api_key=APPID. As discussed in Section VI-E, hackers use this API to piggyback on popular apps and spread spam without being detected.

VIII. Related Work

Detecting Spam on OSNs: Gao *et al.* [11] analyzed posts on the walls of 3.5 million Facebook users and showed that 10% of links posted on Facebook walls are spam. They also presented techniques to identify compromised accounts and spam campaigns. In other work, Gao *et al.* [12] and Rahman *et al.* [10] develop efficient techniques for online spam filtering on OSNs such as Facebook. While Gao *et al.* [12] rely on having the whole social graph as input, and so is usable only by the OSN provider, Rahman *et al.* [10] develop a third-party application for spam detection on Facebook. Others [33], [34] present mechanisms for detection of spam URLs on Twitter. In contrast to all of these efforts, rather than classifying individual URLs or posts as spam, we focus on identifying malicious applications that are the main source of spam on Facebook.

Detecting Spam Accounts: Yang *et al.* [35] and Benevenuto *et al.* [36] developed techniques to identify accounts of spammers on Twitter. Others have proposed a honey-pot-based approach [37], [38] to detect spam accounts on OSNs. Yardi *et al.* [39] analyzed behavioral patterns among spam accounts in Twitter. Instead of focusing on accounts created by spammers, our work enables detection of malicious apps that propagate spam and malware by luring normal users to install them.

App Permission Exploitation: Chia *et al.* [13] investigate risk signaling on the privacy intrusiveness of Facebook apps and conclude that current forms of community ratings 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. To address privacy risks for using Facebook apps, some studies [40], [41] propose a new application policy and authentication dialog. Makridakis *et al.* [42] use a real application named "Photo of the Day" to demonstrate how malicious apps on Facebook can launch distributed denial-of-service (DDoS) attacks using the Facebook platform. King *et al.* [43] conducted a survey to understand users' interaction with Facebook apps. Similarly, Gjoka *et al.* [44] 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.* [45] 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 [46]. Other Facebook applications [47], [48] that defend users against spam and malware do not provide ratings for apps on Facebook. WhatApp? [14] collects community reviews about apps for security, privacy, and openness. However, it has not attracted many 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.

IX. Conclusion

Applications present convenient means for hackers to spread malicious content on Facebook. However, little is understood about the characteristics of malicious apps and how they operate. In this paper, using a large corpus of malicious Facebook apps observed over a 9-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 app-nets—large groups of tightly connected applications that promote each other. We will continue to dig deeper into this ecosystem of malicious apps on Facebook, and we hope that Facebook will benefit from our recommendations for reducing the menace of hackers on their platform.

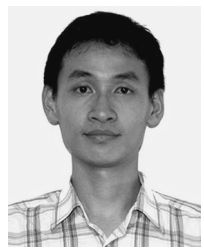
References

[1] C. Pring, "100 social media statistics for 2012," 2012 [Online]. Available: http://thesocialskinny.com/100-social-media-statistics-for-2012/
[2] Facebook, Palo Alto, CA, USA, "Facebook Opengraph API," [Online]. Available: http://developers.facebook.com/docs/reference/api/
[3] "Wiki: Facebook platform," 2014 [Online]. Available: http://en.wikipedia.org/wiki/Facebook_Platform
[4] "Pr0file stalker: Rogue Facebook application," 2012 [Online]. Available: https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_pr0file_viewer_2012_4_4
[5] "Whiich cartoon character are you—Facebook survey scam," 2012 [Online]. Available: https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_whiich_cartoon_character_are_you_2012_03_30
[6] G. Cluley, "The Pink Facebook rogue application and survey scam," 2012 [Online]. Available: http://nakedsecurity.sophos.com/2012/02/27/pink-facebook-survey-scam/
[7] D. Goldman, "Facebook tops 900 million users," 2012 [Online]. Available: http://money.cnn.com/2012/04/23/technology/facebook-q1/index.htm
[8] R. Naraine, "Hackers selling $25 toolkit to create malicious Facebook apps," 2011 [Online]. Available: http://zd.net/g28HxI
[9] HackTrix, "Stay away from malicious Facebook apps," 2013 [Online]. Available: http://bit.ly/b6gWn5
[10] M. S. Rahman, T.-K. Huang, H. V. Madhyastha, and M. Faloutsos, "Efficient and scalable socware detection in online social networks," in *Proc. USENIX Security*, 2012, p. 32.
[11] H. Gao *et al.*, "Detecting and characterizing social spam campaigns," in *Proc. IMC*, 2010, pp. 35–47.
[12] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," in *Proc. NDSS*, 2012.
[13] P. Chia, Y. Yamamoto, and N. Asokan, "Is this app safe? A large scale study on application permissions and risk signals," in *Proc. WWW*, 2012, pp. 311–320.
[14] "WhatApp? (beta)—A Stanford Center for Internet and Society Website with support from the Rose Foundation," [Online]. Available: https://whatapp.org/facebook/
[15] "MyPageKeeper," [Online]. Available: https://www.facebook.com/apps/application.php?id=167087893342260
[16] Facebook, Palo Alto, CA, USA, "Facebook platform policies," [Online]. Available: https://developers.facebook.com/policy/
[17] Facebook, Palo Alto, CA, USA, "Application authentication flow using OAuth 2.0," [Online]. Available: http://developers.facebook.com/docs/authentication/

[18] "11 million bulk email addresses for sale—Sale price $90," [Online]. Available: http://www.allhomebased.com/BulkEmailAddresses.htm
[19] E. Protalinski, "Facebook kills app directory, wants users to search for apps," 2011 [Online]. Available: http://zd.net/MkBY9k
[20] SocialBakers, "SocialBakers: The recipe for social marketing success," [Online]. Available: http://www.socialbakers.com/
[21] "Selenium—Web browser automation," [Online]. Available: http://seleniumhq.org/
[22] "bit.ly API," 2012 [Online]. Available: http://code.google.com/p/bitly-api/wiki/ApiDocumentation
[23] Facebook, Palo Alto, CA, USA, "Permissions reference," [Online]. Available: https://developers.facebook.com/docs/authentication/permissions/
[24] Facebook, Palo Alto, CA, USA, "Facebook developers," [Online]. Available: https://developers.facebook.com/docs/appsonfacebook/tutorial/
[25] "Web-of-Trust," [Online]. Available: http://www.mywot.com/
[26] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Commun. ACM*, vol. 7, no. 3, pp. 171–176, Mar. 1964.
[27] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, Art. no. 27.
[28] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs," in *Proc. KDD*, 2009, pp. 1245–1254.
[29] A. Le, A. Markopoulou, and M. Faloutsos, "PhishDef: URL names say it all," in *Proc. IEEE INFOCOM*, 2011, pp. 191–195.
[30] C. Wueest, "Fast-flux Facebook application scams," 2014 [Online]. Available: http://www.symantec.com/connect/blogs/fast-flux-facebook-application-scams
[31] "Longest path problem," 2014 [Online]. Available: http://en.wikipedia.org/wiki/Longest_path_problem
[32] "App piggybacking example," [Online]. Available: https://apps.facebook.com/mypagekeeper/?status=scam_report_fb_survey_scam_Converse_shoes_2012_05_17_boQ
[33] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time URL spam filtering service," in *Proc. IEEE Symp. Security Privacy*, 2011, pp. 447–462.
[34] S. Lee and J. Kim, "WarningBird: Detecting suspicious URLs in Twitter stream," in *Proc. NDSS*, 2012.
[35] C. Yang, R. Harkreader, and G. Gu, "Die free or live hard? Empirical evaluation and new design for fighting evolving Twitter spammers," in *Proc. RAID*, 2011, pp. 318–337.
[36] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on Twitter," in *Proc. CEAS*, 2010, pp. 1–9.
[37] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting spammers on social networks," in *Proc. ACSAC*, 2010, pp. 1–9.
[38] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers: Social honeypots + machine learning," in *Proc. SIGIR*, 2010, pp. 435–442.
[39] S. Yardi, D. Romero, G. Schoenebeck, and D. Boyd, "Detecting spam in a twitter network," *First Monday,* vol. 15, no. 1, 2010 [Online]. Available: http://firstmonday.org/ojs/index.php/fm/article/view/2793/2431
[40] A. Besmer, H. R. Lipford, M. Shehab, and G. Cheek, "Social applications: Exploring a more secure framework," in *Proc. SOUPS*, 2009, Art. no. 2.
[41] N. Wang, H. Xu, and J. Grossklags, "Third-party apps on Facebook: Privacy and the illusion of control," in *Proc. CHIMIT*, 2011, Art. no. 4.
[42] A. Makridakis *et al.*, "Understanding the behavior of malicious applications in social networks," *IEEE Netw.*, vol. 24, no. 5, pp. 14–19, Sep.–Oct. 2010.
[43] J. King, A. Lampinen, and A. Smolen, "Privacy: Is there an app for that?," in *Proc. SOUPS*, 2011, Art. no. 12.
[44] M. Gjoka, M. Sirivianos, A. Markopoulou, and X. Yang, "Poking Facebook: Characterization of OSN applications," in *Proc. 1st WOSN*, 2008, pp. 31–36.
[45] T. Stein, E. Chen, and K. Mangla, "Facebook immune system," in *Proc. 4th Workshop Social Netw. Syst.*, 2011, Art. no. 8.
[46] L. Parfeni, "Facebook softens its app spam controls, introduces better tools for developers," 2011 [Online]. Available: http://bit.ly/LLmZpM
[47] "Norton Safe Web," [Online]. Available: http://www.facebook.com/apps/application.php?id=310877173418
[48] "Bitdefender Safego," [Online]. Available: http://www.facebook.com/bitdefender.safego

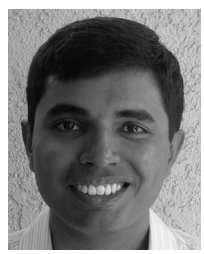
Sazzadur Rahman received the bachelor's degree from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2004, the M.S. degree from the University of Oklahoma, Norman, OK, USA, in 2009, and the Ph.D. degree from the University of California, Riverside, CA, USA, in 2012, all in computer science and engineering.

He is a Senior Software Engineer with Qualcomm Research, San Diego, CA, USA. His research interests include computer vision, systems, and security.



Ting-Kai Huang received the B.Sc. and M.S. degrees in computer science from National Tsing-Hua University, Taiwan, in 2003 and 2005, respectively, and the Ph.D. degree in computer science and engineering from the University of California, Riverside, CA, USA, in 2013.

He joined Google, Mountain View, CA, USA, in 2013. His research interests include networking and security.



Harsha V. Madhyastha received the B.Tech. degree from the Indian Institute of Technology Madras, Chennai, India, in 2003, and the M.S. and Ph.D. and degrees from the University of Washington, Seattle, WA, USA, in 2006 and 2008, respectively, all in computer science and engineering.

He is an Assistant Professor with the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor, MI, USA. His research interests include distributed systems, networking, and security.

Dr. Madhyastha's work has resulted in award papers at the USENIX NSDI, ACM SIGCOMM IMC, and IEEE CNS conferences.

Michalis Faloutsos received the bachelor's degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1993, and the M.Sc and Ph.D. degrees in computer science from the University of Toronto, Toronto, ON, Canada, in 1995 and 1999, respectively.

He is a faculty member with the Computer Science Department, University of New Mexico, Albuquerque, NM, USA. In 2014, he co-founded programize.com, which provides product development as a service. He has been authoring the popular column "You must be joking. . ." in the *ACM SIGCOMM Computer Communication Review*, which reached 19 000 downloads. His interests include Internet protocols and measurements, network security, and routing in ad hoc networks.

Dr. Faloutsos coauthored the paper "On power-law relationships of the Internet topology" (SIGCOMM 1999) with his two brothers, which received the "Test of Time" award from ACM SIGCOMM. His work has been supported by several NSF and DAPRA grants, including the prestigious NSF CAREER Award with a cumulative of more than $10M in grants. He is the co-founder of stopthehacker.com, a Web-security start-up, which received two awards from the National Science Foundation and got acquired in 2013.