FRAppE: Detecting Malicious Facebook Applications

Md Sazzadur Rahman, Ting-Kai Huang, Harsha V. Madhyastha, and Michalis Faloutsos
Dept. of Computer Science, University of California, Riverside
Riverside, CA 92507
rahmanm, huangt, harsha, michalis@cs.ucr.edu

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 low false negative rate (4.1%). 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 1,584 apps enabling the viral propagation of 3,723 other apps through their posts. Long-term, we see FRAppE as a step towards creating an independent watchdog for app assessment and ranking, so as to warn Facebook users before installing apps.

Categories and Subject Descriptors

D.4.6 [**OPERATING SYSTEMS**]: Security and Protection—*Access controls; Verification*

General Terms

Measurement, Security, Verification

CoNEXT'12, December 10–13, 2012, Nice, France.

Keywords

Facebook Apps, Malicious Apps, Profiling Apps, Online Social Networks

1. INTRODUCTION

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 [10] that facilitates app integration into the Facebook user-experience. There are 500K apps available on Facebook [25], and on average, 20M apps are installed every day [1]. 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 [17, 21, 24]. Malicious apps can provide a lucrative business for hackers, given the popularity of OSNs, with Facebook leading the way with 900M active users [12]. There are many ways that hackers can benefit from a malicious app: (a) the app can reach large numbers of users and their friends to spread spam, (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 [13]. In other words, there is motive and opportunity, and as a result, there are many malicious apps spreading on Facebook every day [20].

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, 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 [31, 32, 41]. A recent work studies how app permissions and community ratings correlate to privacy risks of Facebook apps [29]. Finally, there are some community-based feedback-driven efforts to rank applications, such as Whatapp [23]; though these could be very powerful in the future, so far they have received little adoption. We discuss previous work in more detail in Sec. 8.

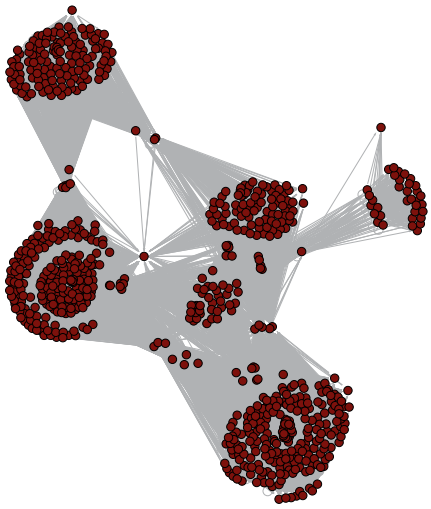

Figure 1: The emergence of AppNets on Facebook. Real snapshot of 770 highly collaborating apps: an edge between two apps means that one app helped the other propagate. Average degree (no. of collaborations) is 195!

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, a security app in Facebook [14] that monitors the Facebook profiles of 2.2 million users. We analyze 111K 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:

- **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 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: (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).
- **The emergence of AppNets: 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. Furthermore, these apps form large and highly-dense connected components, as shown in Fig. 1.

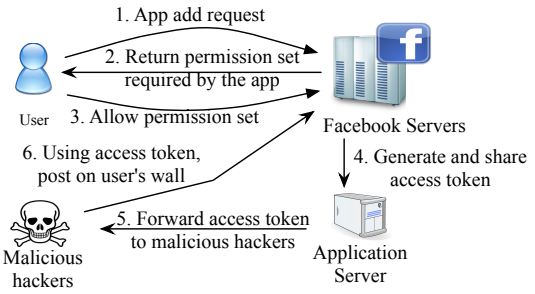

Figure 2: Steps involved in hackers using malicious applications to get access tokens to post malicious content on victims' walls.

Furthermore, 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.
- **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%).

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.

2. BACKGROUND

In this section, we discuss how applications work on Facebook, provide an overview of MyPageKeeper (our primary data source), and outline the datasets that we use in this paper.

2.1 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 applica-

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 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 2: Top malicious apps in D-Sample dataset.

tion 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: (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 [4] 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. 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 [2] to eventually profit the hackers.

2.2 MyPageKeeper

MyPageKeeper [14] is a Facebook app designed for detecting malicious posts on Facebook. Once a Facebook user installs MyPageKeeper, it periodically crawls posts from the user's wall and news feed. MyPageKeeper then applies URL blacklists as well as custom classification techniques to identify malicious posts. Our previous work [41] shows that MyPageKeeper detects malicious posts with high accuracy—97% of posts flagged by it indeed point to malicious websites and it incorrectly flags only 0.005% of benign posts.

The key thing to note here is that MyPageKeeper identifies social malware at the granularity of individual posts, without grouping together posts made by any given application. In other words, for every post that it crawls from the wall or news feed of a subscribed user, MyPageKeeper's determination of whether to flag that post does not take into account the application responsible for the post. Indeed, a large fraction of posts (37%) monitored by MyPageKeeper are not posted by any application; many posts are made manually by a user or posted via a social plugin (e.g., by a user clicking 'Like' or 'Share' on an external website). Even among malicious posts identified by MyPageKeeper, 27% do not have an associated application.

MyPageKeeper's classification primarily relies on a Support Vector Machine (SVM) based classifier that evaluates every URL by combining information obtained from all posts containing that URL. Examples of features used in MyPageKeeper's classifier include a) the presence of spam keywords such as 'FREE', 'Deal', and 'Hurry' (malicious posts are more likely to include such keywords than normal posts), b) the similarity of text messages (posts in a spam campaign tend to have similar text messages across posts containing the same URL), and c) the number of 'Like's and comments (malicious posts receive fewer 'Like's and comments). Once a URL is identified as malicious, MyPageKeeper marks all posts containing the URL as malicious.

2.3 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 [14].

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 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 5, we find this to be an effective 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 6.2. 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 1). Table 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 [19], 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 1) is the union of these 6,273 benign applications with the 6,273 malicious applications ob-

[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 [9].

tained 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 [18]. 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 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 [29], 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 sizable fraction (over 20%) of the approximately 500K apps present on Facebook [25]. 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.

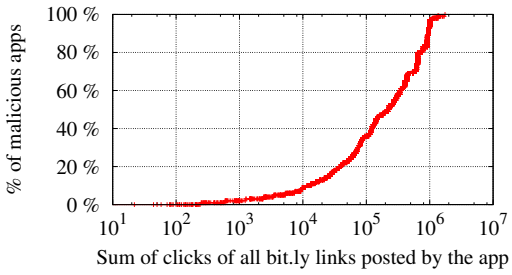

Figure 3: Clicks received by `bit.ly` links posted by malicious apps.

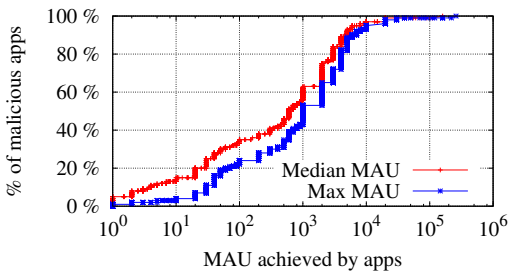

Figure 4: Median and maximum MAU achieved by malicious apps.

3. 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 [6] 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 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 appli-

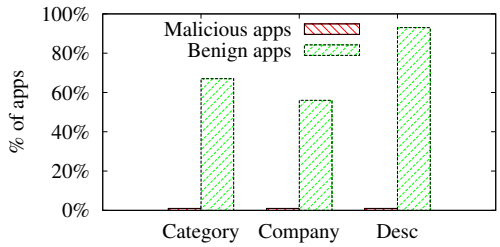

Figure 5: Comparison of apps whether they provide category, company name or description of the app.

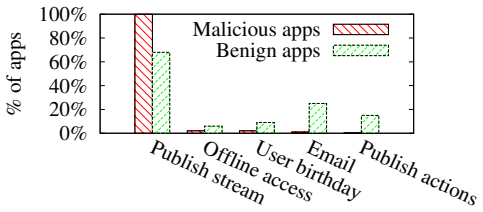

Figure 6: Top 5 permissions required by benign and malicious apps.

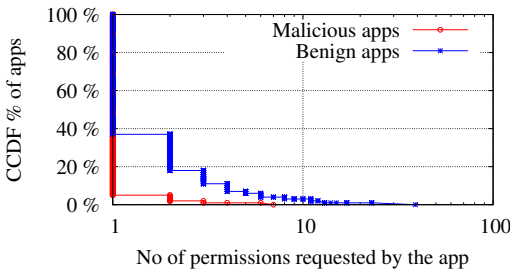

Figure 7: Number of permissions requested by every app.

cation over the last 30 days in activities such as installing, posting, and liking the app. Fig. 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 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.

4. 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 2.3, 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.

4.1 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.

4.1.1 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. 5 shows the fraction of malicious and benign 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).

4.1.2 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 [16]. 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.

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

4.1.3 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 [22]. Fig. 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

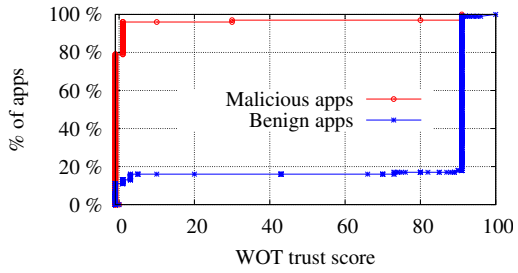

Figure 8: WOT trust score of the domain that apps redirect to upon installation.

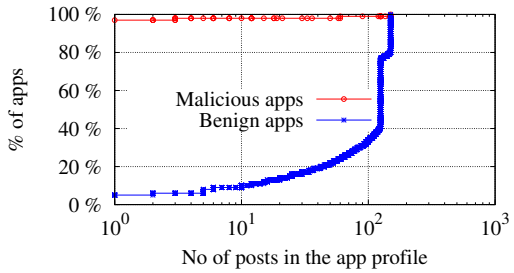

Figure 9: Number of posts in app profile page.

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

Table 3: Top five domains hosting malicious apps in D-Inst dataset.

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 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 shows the top five such domains; these five domains host the content for 83% of the 491 malicious apps in the D-Inst dataset.

4.1.4 *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 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 [8], 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 Sec. 6, 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.

4.1.5 *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. 2.3, we were able to crawl the app profile pages for 3,227 malicious apps and 6,063 benign apps.

From Fig. 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.

4.2 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 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.

4.2.1 *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 [30] between the two names

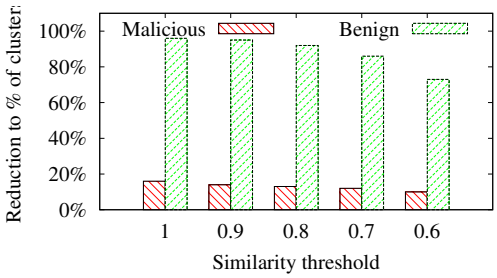

Figure 10: Clustering of apps based on similarity in names.

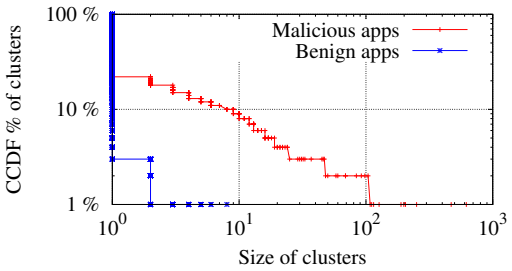

Figure 11: Size of app clusters with identical names.

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. 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. 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. 11, 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.

4.2.2 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.

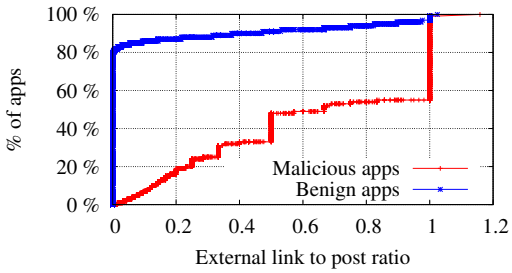

Figure 12: Distribution of external links to post ratio across apps.

Fig. 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.

5. 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 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.

5.1 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.

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 4: List of features used in FRAppE Lite.

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 5: Cross validation with FRAppE Lite.

Table 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) [28] classifier for classifying malicious apps. SVM is widely used for binary classification in security and other disciplines [35, 39]. 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 [28] such as radial basis function as kernel with degree 3, $\text{coef}_0 = 0$ and $C = 1$ [28]. We use the D-Complete dataset for training and testing the classifier. As shown earlier in Table 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 5 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 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 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	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 6: Classification accuracy with individual features.

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 7: Additional features used in FRAppE.

5.2 FRAppE

Next, we consider FRAppE—a malicious app detector that utilizes our aggregation-based features in addition to the on-demand features. Table 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.

5.3 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.

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 8, we were able to validate 98.5% of the apps flagged 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 8: Validation of 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 unverified 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, 5440 apps have been deleted from the Facebook graph. An additional 667 apps have an identical name to one of the 5440 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%.

6. THE MALICIOUS APPS ECOSYSTEM

Equipped with an accurate classifier for detecting malicious apps, we next analyze how malicious Facebook apps support each other. Some of our analysis in Sec. 4 is already indicative of the fact 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.

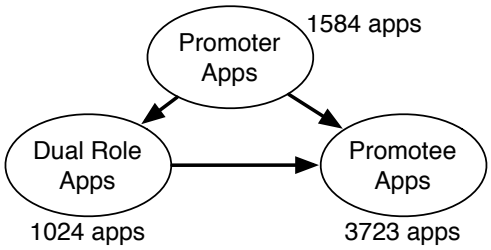

Figure 13: Relationship between collaborating applications

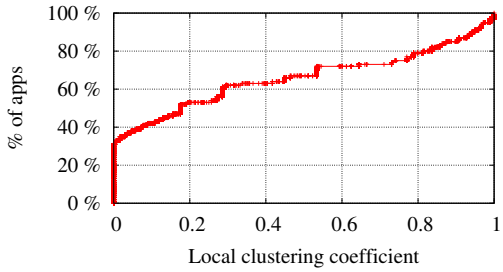

Figure 14: Local clustering coefficient of apps in the Collaboration graph.

6.1 The emergence of AppNets

We identify 6,331 malicious apps in our dataset 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. Fig. 13 shows this relationship between malicious apps. Intrigued, we study this group of applications further.

AppNets form large and densely connected groups. Let us consider the graph that is created by having an edge between any two apps that collude, i.e., an edge from *app*1 to *app*2 if the former promotes the latter. We call this graph the *Collaboration graph*. In this graph, we identify 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 [2] larger than 0.74 as shown in Fig. 14.

[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.

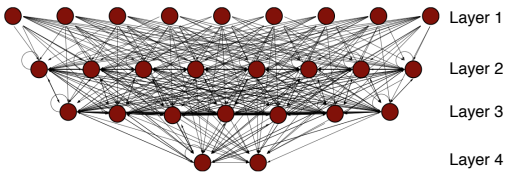

Figure 15: Example of collusion graph between applications.

As an example, in Fig. 15, we show 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.

App collusion 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 find 692 promoter apps in our D-Sample dataset which promoted 1,806 different apps using direct links. This activity was intense: 15% of the promoters promoted at least 5 promotee apps. For example, 'The App' was promoting 24 other apps with names 'The App' or 'La App'.

b. Indirect app promotion. Alternatively, hackers use websites outside Facebook to have more control and protection in promoting apps. Specifically, a post made by a malicious app includes a shortened URL and that URL, once resolved, points to a website outside Facebook. 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.

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 reuse 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 applications each. Following the discussion in Sec. 4.2.1, 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.

Amazon hosts a third of these indirection websites. 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`.

6.2 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

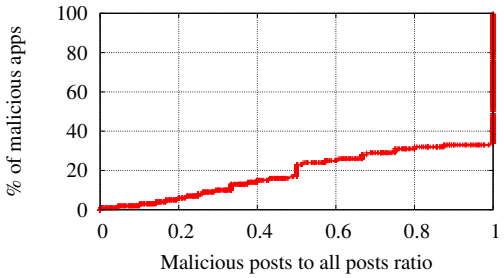

Figure 16: Distribution across apps of the fraction of an app's posts that are malicious.

'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: [3].

We find 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. 16, 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 9 shows the top five most popular apps that we find among this set.

7. 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

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 9: Top five popular apps being abused by app piggybacking.

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. 4.1.4, 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. 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 Sec. 6.2, hackers use this API to piggyback on popular apps and spread spam without being detected.

8. RELATED WORK

Detecting spam on OSNs. Gao et al. [32] 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. [31] and Rahman et al. [41] develop efficient techniques for online spam filtering on OSNs such as Facebook. While Gao et al. [31] rely on having the whole social graph as input, and so, is usable only by the OSN provider, Rahman et al. [41] develop a third-party application for spam detection on Facebook. Others [37,44] 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. [46] and Benevenuto et al. [26] developed techniques to identify accounts of spammers on Twitter. Others have proposed a honey-pot based approach [36,43] to detect spam accounts on OSNs. Yardi et al. [47] 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. [29] 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. [38] showed that privacy settings in Facebook rarely match users' expectations.

To address the privacy risks associated with the use of Facebook apps, some studies [27, 45] propose a new application policy and authentication dialog. Makridakis et al. [40] 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. [34] conducted a survey to understand users' interaction with Facebook apps. Similarly, Gjoka et al. [33] 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. [42] 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 [11]. Other Facebook applications [5,7,15] that defend users against spam and malware do not provide ratings for apps on Facebook. Whatapp [23] 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.

9. CONCLUSIONS AND FUTURE WORK

Applications present a 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 work, 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 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.

10. REFERENCES

[1] 100 social media statistics for 2012. http://thesocialskinny.com/100-social-media-statistics-for-2012/.

[2] 11 Million Bulk email addresses for sale - Sale Price $90. http://www.allhomebased.com/BulkEmailAddresses.htm.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[24] 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.

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

[26] F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida. Detecting spammers on Twitter. In *CEAS*, 2010.

[27] A. Besmer, H. R. Lipford, M. Shehab, and G. Cheek. Social applications: exploring a more secure framework. In *SOUPS*, 2009.

[28] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 2011.

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

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

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

[32] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao. Detecting and characterizing social spam campaigns. In *IMC*, 2010.

[33] M. Gjoka, M. Sirivianos, A. Markopoulou, and X. Yang. Poking facebook: characterization of osn applications. In *Proceedings of the first workshop on Online social networks*, WOSN, 2008.

[34] J. King, A. Lampinen, and A. Smolen. Privacy: Is there an app for that? In *SOUPS*, 2011.

[35] A. Le, A. Markopoulou, and M. Faloutsos. Phishdef: Url names say it all. In *Infocom*, 2010.

[36] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In *SIGIR*, 2010.

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

[38] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing facebook privacy settings: user expectations vs. reality. In *IMC*, 2011.

[39] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious urls. In *KDD*, 2009.

[40] A. Makridakis, E. Athanasopoulos, S. Antonatos, D. Antoniades, S. Ioannidis, and E. P. Markatos. Understanding the behavior of malicious applications in social networks. *Netwrk. Mag. of Global Internetwkg.*, 2010.

[41] M. S. Rahman, T.-K. Huang, H. V. Madhyastha, and M. Faloutsos. Efficient and Scalable Socware Detection in Online Social Networks. In *USENIX Security*, 2012.

[42] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, 2011.

[43] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *ACSAC*, 2010.

[44] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and Evaluation of a Real-Time URL Spam Filtering Service. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

[45] N. Wang, H. Xu, and J. Grossklags. Third-party apps on facebook: privacy and the illusion of control. In *CHIMIT*, 2011.

[46] C. Yang, R. Harkreader, and G. Gu. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers. In *RAID*, 2011.

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