AppInspect Large-scale Evaluation of Social Apps

Markus Huber[*†], Martin Mulazzani[†], Sebastian Schrittwieser[†], Edgar R. Weippl[†]
[*]Vienna PhD School of Informatics
[†]SBA Research
{mhuber,mmulazzani,sschrittwieser,eweippl}@sba-research.org

Abstract

Third-party apps for social networking sites have emerged as a popular feature for online social networks, and are used my millions of users every day. In exchange for additional features, users grant third parties access to their personal data. However, these third parties do not necessarily protect the data to the same extent as social network providers. To automatically analyze the unique privacy and security issues of social networking applications on a large scale, we propose a novel framework, called *AppInspect*. Our framework enumerates available social networking apps and collects metrics such as the personal information transferred to third party developers. AppInspect furthermore identifies web trackers, as well as information leaks, and provides insights into the hosting infrastructures of apps. We implemented a prototype of our novel framework to evaluate Facebook's application ecosystem. Our evaluation shows that AppInspect is able to detect malpractices of social networking apps in an automated fashion. During our study we collaborated with Facebook to mitigate shortcomings of popular apps that affected the security and privacy of millions of social networking users.

I. Introduction

Third-party applications, or colloquially "apps", are used by hundreds of millions of social networking users every day. Popular apps include games, horoscopes, and quizzes. To provide additional features, app developers transfer personal information from their users to their application servers. Online social networks typically embed applications as framed websites in their own portal and thus act as proxies between users and third-party applications. The actual application code runs on third-party servers beyond the supervision of social network providers.

The modus operandi of social networking apps gives rise to unique privacy and security challenges. Applications may maliciously harvest a wealth of personal information. One of the key challenges is, therefore, to detect applications that process data in a way that may violate the security or privacy expectation of users, and to identify apps that request more permissions than actually needed for their operation. Furthermore, sensitive user data may be stored on badly maintained third-party servers, making them low-hanging fruits for attackers. Insights into applications' underlying hosting infrastructure would help to get a better understanding of these security risks. Application providers themselves rely on third parties for in-app advertising and analytics. Therefore, social networking apps may leak sensitive information to third parties, both deliberately or by accident. In the worst case, third parties may use leaked personal information to track app users across multiple websites with knowledge of their real identity. As a result, detecting information leakage is another important challenge. Previous research focused on a single challenge, namely analyzing personal information requested by social networking apps [34], [9]. As a result of the deep integration of apps into social networking platforms, users often do not understand that application developers receive and accumulate their personal information [24]. We are left with a dilemma of social networking users' misperception regarding app security and privacy, but also with little insight into third-party application ecosystems.

Manuscript to appear at ACM COSN 2013. http://dx.doi.org/10.1145/2512938.2512942

In this paper we outline a novel framework, called *AppInspect*, to systematically analyze the unique privacy and security challenges of social networking applications. Our proposed framework analyzes both information flows from social networking providers to third-party applications and information flows from social networking applications to third parties. An initial challenge in studies of online social networks lies in obtaining a meaningful sample of applications. Our *AppInspect* framework entails a number of application enumeration strategies to overcome this first obstacle. In the next step, our framework automatically fetches important attributes of enumerated applications, including their popularity and the set of requested permissions. Finally, our framework collects the network traffic of social networking apps to subsequently spot web trackers, poorly maintained application hosts, and leaking of sensitive information to third parties. The motivation of our research is to protect social networking users by automatically detecting security and privacy issues with social networking apps, as well as policy violations. The findings of our *AppInspect* framework assist both social networking providers and application developers in protecting their users. We used our *AppInspect* prototype to carry out a large-scale evaluation of Facebook's application ecosystem, which ultimately helped to detect and report a number of privacy and security shortcomings. The main contributions in this paper are the following:

- We present a novel framework for automated privacy and security analysis of social networking apps.
- We evaluate the feasibility of our *AppInspect* framework with Facebook's application ecosystem.
- We found information leaks and malpractices in popular apps and helped to fix these issues.
- We make our datasets of Facebook applications available to the research community.

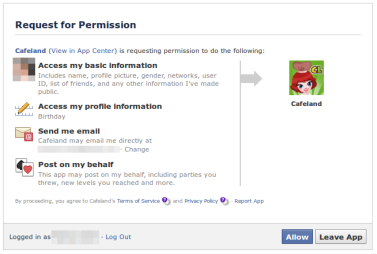
The rest of this paper is structured as follows: Section II provides a brief background on third-party applications in online social networks. Section III outlines the design and functionality of our proposed framework. In Section IV, we describe the evaluation of our framework on Facebook and our acquired application sample. Section V presents our results on Facebook's application ecosystem. We then discuss the implications of our findings in Section VI, explore related work in Section VII, and finally present our conclusions in Section VIII.

II. Social Networking Apps

Third-party applications are a popular feature of today's online social networks (OSNs). These "apps", as they are colloquially referred to, enrich user data to provide additional user experience and functionality. An app might, for example, query a user's birthday to create a personalized horoscope in exchange. Social networking data is hereby provided to third parties through developer application programming interfaces (APIs). At the time of writing, there are two major classes of apps. The first class consists of games, which typically incorporate aspects of social networking into their gameplay. The second class contains general add-ons to social network platforms, ranging from simple horoscope applications to sophisticated job hunting applications. Facebook pioneered third-party applications by introducing the "Facebook Platform" in May 2007 [14]. Facebook's competitors responded with the launch of an open standard for third-party access to social networking data called "OpenSocial" in November 2007 [19]. At the time of writing, Facebook's Connect platform is the most popular and mature framework for social networking apps, which is why we use Facebook as an example of third-party platforms. Ko et al. [25] provides an overview of existing social networking APIs.

A. *Facebook Platform*

The Facebook Platform enables third parties to offer custom applications that extend Facebook's core functionality and integrate deeply into their website. Apps on Facebook are loaded inside Facebook through a "Canvas Page", which represents an *HTML iframe*. Facebook acts as a proxy for displaying the output of apps to its users through iframes, while the actual apps are hosted and executed on third-party servers. Facebook has no control over app servers but legally binds third-party developers to comply with their Platform policies [13]. Before users decide to install a specific app, they need to authorize it. Facebook uses OAuth 2.0 [21] for authentication and authorization of third-party applications. Once users



Request for Permission

Cafeland (View in App Center) is requesting permission to do the following:

Access my basic information
Includes name, profile picture, gender, networks, user ID, list of friends, and any other information I've made public.

Access my profile information
Birthday

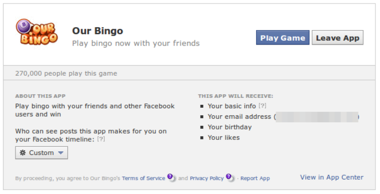
Send me email
Cafeland may email me directly at [redacted] · Change

Post on my behalf
This app may post on my behalf, including parties you threw, new levels you reached and more.

By proceeding, you agree to Cafeland's Terms of Service and Privacy Policy · Report App

Logged in as [redacted] · Log Out

(a) Unified Auth Dialog, April 2010



Our Bingo
Play bingo now with your friends

270,000 people play this game

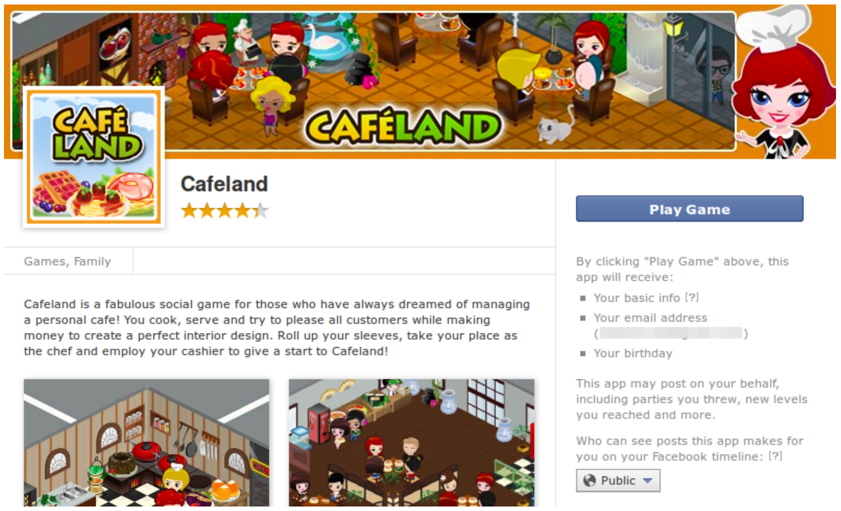
ABOUT THIS APP
Play bingo with your friends and other Facebook users and win

Who can see posts this app makes for you on your Facebook timeline: [?]

THIS APP WILL RECEIVE:
- Your basic info [?]
- Your email address ([redacted])
- Your birthday
- Your likes

By proceeding, you agree to Our Bingo's Terms of Service and Privacy Policy · Report App

(b) Enhanced Auth Dialog, January 2012



Cafeland

Games, Family

Cafeland is a fabulous social game for those who have always dreamed of managing a personal cafe! You cook, serve and try to please all customers while making money to create a perfect interior design. Roll up your sleeves, take your place as the chef and employ your cashier to give a start to Cafeland!

By clicking "Play Game" above, this app will receive:
- Your basic info [?]
- Your email address ([redacted])
- Your birthday

This app may post on your behalf, including parties you threw, new levels you reached and more.

Who can see posts this app makes for you on your Facebook timeline: [?]

(c) App Center Auth Dialog, May 2012

Fig. 1: Adjustments to Facebook's application authorization dialog over time

authorize an application, it is granted access to *basic information* by default. Basic information includes ID, name, picture, gender, locale, and friend connections of a given user. Applications may, however, request additional permissions. At the time of writing, there are four additional permission classes available on the Facebook Platform [12], which applications may request in any combination. In total, there are 67 possible application permissions that app developers may request for their application. Permissions within the *extended permissions* class grant applications access to sensitive information such as exchanged private messages, and allows applications to post content on behalf of its users. Another important permission class is the *user and friend permissions* class. Using permissions in this class, developers may request to gather information on a user's religion, relationship status, birthday, personal email addresses, and virtually any published content. Developers may also request personal profile information from a user's friends, with the exception of private email addresses. Facebook's current default account settings allow applications to access personal information of all of a user's friends. This implies that, even if you have not installed a single application, your data may be transferred to third parties through your friends' applications. Even though users can control which information is provided to their friend's applications, users are often unaware that they share their information with apps per default.

Before users decide to add an application, an authorization dialog with requested permissions is displayed. Hull et al. [22] claim that Facebook's privacy issues are based primarily on design issues, which could be improved by making the flows of information more transparent to users. The example of Facebook's adjustments to their app authorization dialog in Figure 1, suggests that Facebook might currently invest little in making their third-party application system more transparent. In response to complaints from the privacy commissioner of Canada, Facebook introduced a unified permissions dialog in April 2010, of which Figure 1(a) provides an example. The unified dialog has been deprecated and only a small number of applications still use this dialog. In January 2012 Facebook launched a revised permissions dialog called *Enhanced Auth Dialog* (see Figure 1(b)), which replaced the unified permissions dialog. In May 2012, a third permission dialog for all applications listed in Facebook's App Center was introduced. Figure 1(c) shows an instance of this dialog. The standard authentication dialog uses pictograms and verbose descriptions for requested permissions and users may choose between *Allow* and *Leave App*. Furthermore, a directed arrow symbolizes that the requested information is transferred to a third party. Requested permissions faded from the spotlight with the enhanced authentication dialog and permissions are now presented in a bulleted list with little additional information. Facebook also changed the label of the authentication button, which reads *Play Game* instead of *Allow*. Finally, with the App Center authentication dialog, the requested permissions are hardly noticeable and a single prominent button encourages users to play the game.

Previous research indicates that there are a number of common misconceptions regarding how social networking applications work. Besmer and Lipford [5] conducted semi-structured interviews as well as a survey to understand how users perceive privacy risks of social applications. Their findings showed that social interaction drives the spread and use of applications, and influences perceptions about privacy and information disclosure. Furthermore, the participants' privacy concerns were centered around sharing data with other people on the social network, with almost no understanding of the data sharing that occurs with the application developers. The researchers concluded that there is a serious risk of applications maliciously harvesting profile information, and that users are neither aware of nor do they consent to these risks. King et al. [24] conducted an exploratory survey to measure how Facebook app users interact with apps, what they understand about how apps access and exchange their profile information, and how these factors relate to their privacy concerns. They discovered that many users have limited comprehension of the privacy issues posed by apps on Facebook Platform. The authors finally argue that the interleaving of applications into social relationships diverts attention away from the underlying institutional privacy concerns.

B. Application directories and reviews

An important privacy and security challenge with social networking apps is the balance between requested data and app functionality. Horoscope apps may for example harvest a user's personal messages and photos instead of requesting only the date of birth. The Facebook Platform enables third-party developers to make their apps available to other users without requiring prior approval. Between May 2008 and December 2008, Facebook operated a verified apps program, through which it designated certain applications as "verified apps". A verified apps badge was promised to applications that are secure and demonstrated commitment to compliance with the Facebook platform policies. An FTC report, however, found out that Facebook took no extra steps to verify the security of third-party applications and labeled the program as deceptive [18]. Until July 2011, Facebook offered a central application directory, where developers could submit their applications once they considered their software mature. Later, Facebook removed its app directory and applications are now automatically indexed once they reach 10 monthly active users (MAU) [11]. In May 2012 Facebook introduced the App Center[1], which showcases what Facebook describes as high-quality applications. At the time of writing, the App Center contains a few thousand applications. The small number of App center applications is in stark contrast to the overall number of applications. According to Facebook [29, p. 87], as of March 2012, more than nine million apps and websites were connecting to their Platform services.

III. AppInspect

The vast amount of available third-party social networking applications poses a challenge for large-scale security and privacy studies. In order to overcome the naïve solution of manually analyzing security and privacy issues of third-party applications, we propose a novel analysis framework, called *AppInspect*. In this section we outline the design and functionality of our framework.

Our proposed *AppInspect* framework enables fully automated security and privacy analysis of a target social networking app ecosystem. Figure 2 depicts the four generic processing steps to automatically analyze a given social networking provider with AppInspect. (1) First, the search module enumerates available third-party applications for a given social networking provider. (2) In a second step, the classifier module collects additional information for all enumerated apps. (3) Third, the analysis module adds the applications to experimental accounts and collects the resulting network traffic for further analysis. (4)

[1]FB App Center http://www.facebook.com/appcenter

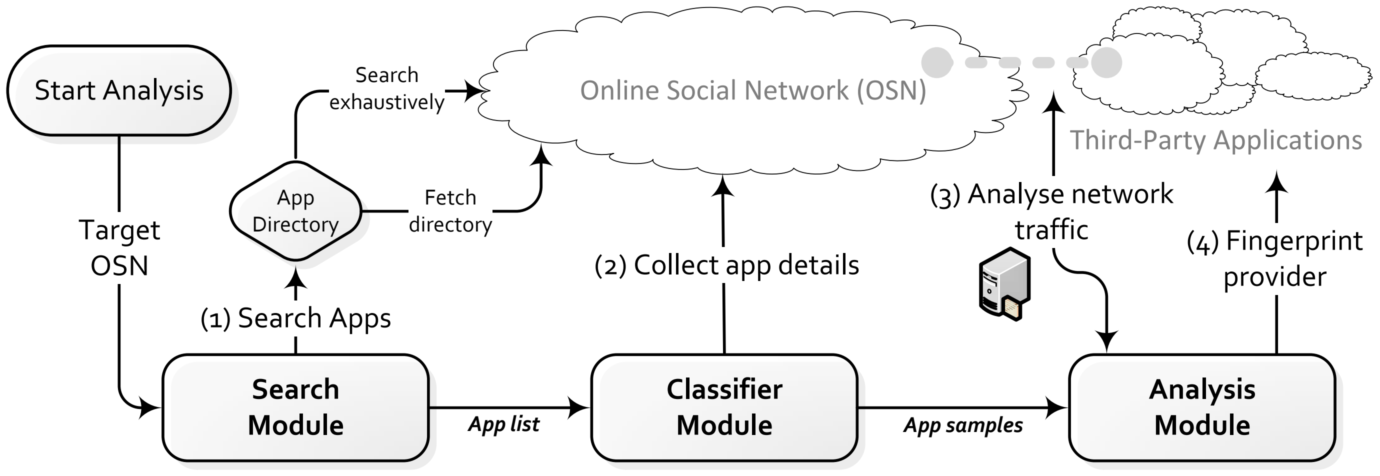


Fig. 2: AppInspect, a framework for automated security and privacy analysis of social network ecosystems

Finally, the analysis module fingerprints the hosting infrastructure of all applications. AppInspect uses a modular software design and its functionality is separated into three main modules. Our design enables a straightforward adaption of features for different security and privacy analyses. The three main modules and their submodules are described in the following.

A. Search module

The initial challenge with social networking providers consists in collecting a preferably complete list of third-party applications for further analysis in case a central app directory is missing. In the best case, the social networking provider offers a complete app directory, which contains all third-party applications. In the non-trivial case, no complete application directories exist and third-party applications have to be enumerated using exhaustive search strategies.

Exhaustive search. In case no central application directory exists, exhaustive search strategies are required. The most straightforward solution is the enumeration of unique application identifiers. This naïve approach works with social network providers with a small range of numerical application identifiers. In the case of LinkedIn, for example, all available applications are easy to enumerate by testing a small range of application identifiers.

HTTP Request: Enumeration of LinkedIn app

```
GET /opensocialInstallation/preview?_applicationId=1000
Host: https://www.linkedin.com
```

With Facebook, the exhaustive search strategy becomes a non-trivial problem because their application identifiers are not easily enumerable. Facebook assigns a unique numerical identifier to every object it stores. Objects include third-party applications but also user profiles, pictures, posts, etc. At the time of writing, Facebook's unique object identifiers are numerical values of length 14, resulting in up to 10^{14} possible combinations. This means that it is not feasible to probe the entire identifier range for third-party applications due to the resulting costs for crawling and the fact that only a subset of these IDs are for apps. However, Facebook indexes all third-party applications that have reached more than 10 monthly active users in their search feature. Hence, an exhaustive search for indexed applications opens a way to enumerate applications on Facebook. Instead of integer ranges, the exhaustive search probes the social network provider for keywords or character n-grams. For example, all trigrams for the English alphabet would result in $26^3 = 17{,}576$ search terms. Castelluccia et al. [8] used this approach for a similar problem, namely the reconstruction of a users' search history. Similar to their work, our module can either use all possible character n-grams or limit the number of search terms by using Castelluccia et al.'s smart tree approach. In addition to character n-grams, lists with common words provide yet another keyword source.

Directory fetch. Some social networks offer a complete application directory. For example, Google+'s game directory[2] consists of a single webpage that contains less than 50 applications in total. In this particular case, the AppInspect framework provides a dedicated submodule to gather the list of all available third-party applications from the social network's application directory.

B. Classifier module

The classifier module collects additional information on applications enumerated with AppInspect's search module. Information is gathered passively from the social network provider without actually running or adding applications to profiles.

Application properties. A number of application properties are available on third-party application description pages. Important properties include the application type, popularity, and rating. This submodule implements functionality to automatically gather a set of predefined properties. The submodule opens the generic application page for every application identifier and, in addition to fetching available information, also observes the URL redirection behavior. Facebook, for example, redirects users to different targets depending on the application type. The following example redirects the user to http://yahoo.com, and the submodule therefore classifies the application as an external website.

```
GET /apps/application.php?id=194699337231859
Host: www.facebook.com
⟹ Redirects to http://yahoo.com
```

The second example of a Facebook application redirects the user to an authentication dialog that is classified as a standard application that requests additional information.

```
GET /apps/application.php?id=102452128776
Host: www.facebook.com
⟹ Redirects to Facebook authentication dialog
```

Permissions. An important classification property of third-party applications is the set of requested permissions. This submodule collects the set of requested permissions using two different techniques: permissions are collected from rendered permission dialogs and based on parameters in permission dialog request URIs.

Language. Third-party applications cater to many different geographical locations. The language submodule detects the language used by third-party applications. An application name together with the set of requested permissions often helps to spot suspicious apps. This submodule therefore translates all non-English application names into English.

C. Analysis module

The analysis module analyses the actual application content. To this end, applications are installed on test accounts by automating a Web browser.

Web tracker identification. Social network application developers themselves rely on third-party components for analytics and advertising. In-app advertising promises revenue, while analytic products provide application developers with additional insights into their applications' users. Third-party analytics and advertising products raise major privacy concerns, because they may track users across multiple websites. The web tracker identification submodule identifies planted web trackers based on network traffic collected with the analysis module.

Information leaks. Personally identifiable information (PII) is information that can be used to uniquely identify a single individual with or without additional data sources. In case of online social networks, a user's unique identifier represents a sensitive PII. This submodule analyses whether social networking apps leak PII to third-party components, such as advertising and analytics providers. In addition to information leaks of personally identifiable information to third parties, application developers may unintentionally

[2]https://plus.google.com/games/directory

leak API authentication tokens through HTTP Referer. Therefore, this submodule traces leaks of tokens and unique user identifiers in the collected traffic. HTTP request (a) provides an example of leakage through an HTTP Referer header where "Super Analytics" receives a user's unique identifier as well as the app's OAuth token through the Referer header of the HTTP request. The analytics provider could then impersonate the application with the leaked access token to access the user's personal information.

(a) Information leakage via HTTP Referer

```
GET /__beacon.gif
Host: www.super-analytics.com
Referer: http://www.fbgameexample.com/flash.php
         ?oauth_token=AAA...&id=111111111&locale=en_US
```

HTTP request (b) provides an example of PII leakage through a URI request. In this example the third-party application transfers unique identifiers directly to a third party.

(b) Information leakage via URI request

```
GET /api/v1/ip=...&uid=111111111&data=%7B%7D
Host: api.tripppleclick.net
```

Network fingerprint. The network fingerprint submodule provides network metrics of a given social networking app. The submodule first performs an analysis on the collected network traffic to determine the application's domain. Subsequently, a number of metrics are collected on the application domain. The network fingerprint submodule furthermore performs a non-intrusive service discovery scan against the third-party systems by enumerating a list of TCP ports accepting packets and their corresponding service banners on the application host. Finally, the vulnerability search submodule determines whether a third-party host uses outdated software that might eventually compromise the security of their systems. This submodule matches discovered service types and version numbers against publicly available vulnerability databases.

IV. Evaluation and App Sample

In this section we briefly discuss our research methodology and outline the prototype implementation of our *AppInspect* framework for Facebook. In the following we describe our enumerated third-party application sample.

A. Methodology

We chose to implement an instance of our *AppInspect* framework for the Facebook platform. Facebook serves as a good example due to its popularity and the plethora of available third-party applications. Facebook offers dedicated whitehat accounts[3] for security researchers, these accounts however cannot interact with third-party applications. We therefore set up a number of experimental accounts with bogus data, in order to perform automated application evaluations without processing actual personal information. Once we finished our experiments we deactivated all Facebook test accounts. In order to detect third-party products, we used traffic patterns from the Ghostery database, which contains more than 1,200 ad networks and trackers[4]. We complemented Ghostery's traffic patterns with additional trackers we identified during our traffic analysis. In order to find potential vulnerabilities of application hosts, we fingerprinted their publicly available web services in a non-intrusive way. We strictly refrained from interfering with application web services and instead based our analysis on detected service banners.

[3]https://www.facebook.com/whitehat/accounts/

[4]Ghostery http://www.ghostery.com/

B. AppInspect prototype

Our prototype uses a twofold crawling strategy. The enumeration and initial classification of social networking apps is performed with a lightweight Python module. In addition, we use a full-fledged Mozilla Firefox web browser that collects data for the application security and privacy analysis. We found that many gaming applications required Adobe Flash, and an up-to-date version of Adobe Flash was therefore included in the web browser to get realistic network traffic samples. Automating a full-fledged web browser allows us to execute inline JavaScript code and Adobe Flash content exactly as they would be processed by a normal application user. Our twofold approach thus allows efficient headless crawling for enumerating and classifying apps, but also a thorough analysis of executed application content with a state-of-the-art web browser.

Search Module. Facebook offers two application directories that contain a tiny subset of their third-party applications. The majority of third-party applications, however, is only retrievable with Facebook's global search feature. Therefore, we implemented three submodules to enumerate third-party applications: an exhaustive search submodule that generates application search terms and feeds them into Facebook's search, and two submodules to collect all applications from Facebook's *Timeline* and *Application Center* directories.

Classification Module. The classification module for Facebook implements the collection of application type, permissions, rating, and language. A submodule of our AppInspect framework collects application properties based on application info pages. The application type is determined based on both harvested information and the application target URI. The permission submodule detects application authentication dialogs and collects the set of requested permissions. Finally, we implemented a generic language detection module that relies on the Google Translate API to detect and translate non-English application names.

Analysis Module. The traffic analysis submodule automates the installation of a given Facebook application on a test account and collects all traffic with a transparent HTTP(S) proxy. Moreover, our prototype implements our proposed analysis submodules. We implemented the information leaks submodule, which probes the session recordings for sensitive information. To detect information leaks we verify whether our test account's unique identifier or OAuth tokens are transmitted to detected third-party products. We inspect the collected traffic dumps for plaintext and Base64-encoded unique identifiers and authentication tokens. In order to reduce false positives of HTTP Referer leaks, the information leaks submodule verifies whether information is leaked to third parties other than application providers themselves. We also did not consider leaks to content delivery networks critical, because they usually do not track web users. The network fingerprint submodule parses web session recordings and determines the application host based on the application's OAuth session initialization. The submodule, furthermore, uses the `tracepath` and `dig` utilities to collect network metrics. In addition to collecting network metrics, the networking fingerprint submodule also provides port scanning functionality. Finally, our prototype implements a vulnerability submodule that searches for outdated software. A number of vulnerability databases exist and we focus on databases with readily accessible exploits. The vulnerability submodule thus searches within the *Exploit Database*[5] as well as for readily available Metasploit modules[6].

C. Enumerated application sample

We performed an initial enumeration of applications in April 2012 with search terms based on bigrams of the English alphabet. The search module was configured to harvest information non-aggressively with a limit of 2,500 queries per day. Facebook imposes rate limits on standard accounts on a daily per-account basis. Therefore, we relied on a pool of accounts set up for the experiment, which we rotated during app analysis. Our first exhaustive search resulted in 234,597 applications. This first run helped us fine-tune our exhaustive search module. In Mid 2012, we reran the search module, this time with character trigrams

[5]Exploit DB http://www.exploit-db.com/

[6]Metasploit http://www.metasploit.com/modules/

based on the English alphabet and also on integers from 0 - 9. The exhaustive search module enumerated *434,687* unique applications. Our application directory submodules found 129 Timeline applications and 108 applications in Facebook's App Center. Our search module successfully verified that all Timeline and App Center applications were included in our enumerated application sample. We observed a great disparity in the monthly active users (MAU) of the enumerated applications. Figure 3 illustrates our observation. While the great majority of applications had a MAU lower than 10,000, a small number of applications attracted a wider audience (red graph). Relative to our sample's cumulative application usage, the top 10,000 apps covered *93.16* percent (green graph) of all MAUs.

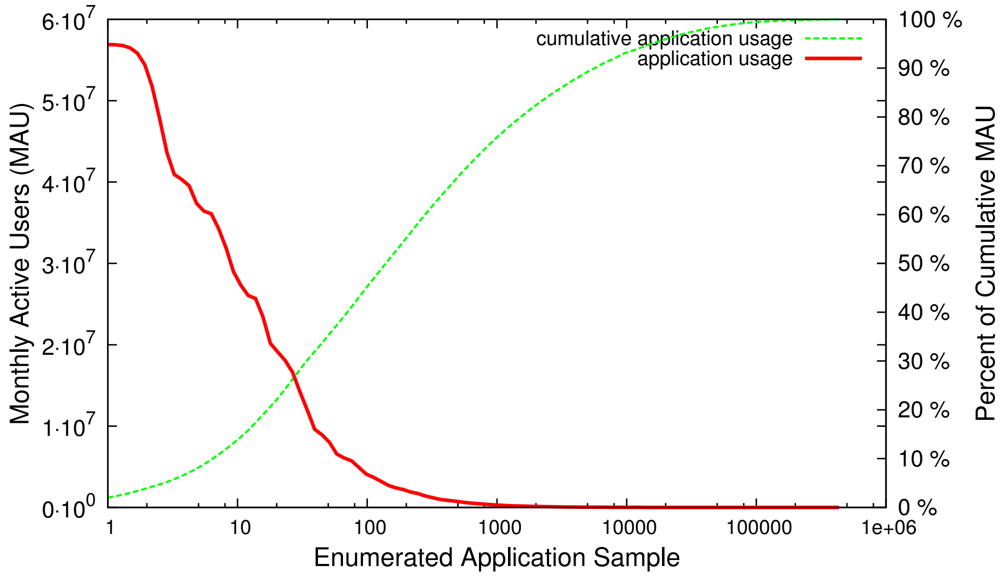


Fig. 3: Active monthly users in our sample of 434,687 enumerated applications

We discarded all applications with fewer than 10,000 MAU from our subsequent analysis because of their comparably minor impact. In Autumn 2012, we performed an analysis of the 10,624 most popular apps with AppInspect's classifier module. Our selected subsample covered 94.07% of all applications relative to our samples' cumulative application usage. The results in Table Ia show the different application types observed in our sample. 44.68% or 4,747 applications belonged to the *Authentication Dialog* class. The *Authentication Dialog* class represents canvas applications that request personal information from their users. The *Canvas* class represents applications that load external content into the Facebook canvas but do not require any personal information from their users to work. *Connect* applications are external websites that leverage the Facebook API. *Connect* applications often use Facebook as an identity provider and to import Facebook content into their own portals. A number of apps responded with an error or were canceled (*Defect*). *Page Add-ons* are applications that provide add-ons to Facebook pages; these apps have access to the content of Facebook pages. The *Mobile* class, finally, represents applications that target mobile platforms such as Android or iOS.

The relatively high number of defective applications (8.14%) can be attributed to two independent observations. First, developers of less popular applications had trouble maintaining reliable applications. As a result, a number of applications responded with error codes or did not respond at all. Second, Facebook's application ecosystem is volatile and some applications are available only for a limited timespan. In the final step, the classification module leveraged the Google Translate API to detect languages used. The majority of applications were English (64.72%), followed by Spanish and German. In total, we observed 69 different languages.

Application Type	**Applications**	**Total %**
Authentication	4,747	44.68%
Canvas	2,365	22.26%
Connect	2,260	21.27%
Defect	865	8.14%
Page Add-ons	280	2.64%
Mobile	107	1.01%
Total	*10,624*	*100.00%*

(a) Application Type

ISO	**Language**	**Total %**
en	English	64.72%
es	Spanish	3.71%
de	German	3.07%
tr	Turkish	2.41%
hu	Hungarian	2.10%
fr	French	2.08%
-	Other	21.91%

(b) Application Language

TABLE I: Classification of most popular apps (n=10,624)

V. Results

This section describes the results of our extensive security and privacy analysis of third-party applications. The in-depth analysis was performed on the most popular Facebook canvas applications that requested additional information. These 4,747 apps represent a significant subsample of our enumerated applications because they impact most users.

A. *Requested personal information*

Table II shows the most frequently requested permissions to access personal information out of the *4,747* most popular third-party applications. The most requested permission for games was “publish posts to stream”, which allows an app to post to a user’s profile. In total, 51.32% of these third-party applications asked permission to publish to a user’s stream. The table also shows that access to a user’s personal email address was most commonly requested for generic apps and by 46.07% of all third-party applications that requested personal information. It is also interesting to observe that access to users’ birth dates and photos are often requested as well.

	App Category		
Permission	game	app	**Total %**
Publish posts to stream	*1,617*	819	51.32%
Personal email address	1,055	*1,132*	46.07%
Publish action	435	857	27.22%
Access user’s birthday	582	428	21.28%
Access user’s photos	721	99	17.27%
Access data offline	517	120	13.42%
Access user likes	438	153	12.45%
Access user location	350	143	10.39%
Read stream	409	80	10.3%
Access friends’ photos	319	17	7.08%

TABLE II: Most common requested permissions by third-party applications (n=4,747)

We clustered applications based on their hosting domains to identify application providers. Our results showed that the 4,747 applications belonged to *1,646* distinct domains or providers. Furthermore, 73.42% or 3,485 apps belonged to a third party with more than one application. Third parties that offer multiple applications can request different personal information with different applications. Once a user installs more than one of their applications, providers can simply aggregate all collected user information. Therefore, we argue that requested permissions need to be analyzed not only based on individual apps, but also based on application providers. In this analysis, we found that the most requested permission per application provider was access to personal email addresses, which 60.24% of all providers requested. Figure 4 depicts the number of distinct permissions requested per application provider. The provider samples are hereby sorted by their monthly active users. On average, providers requested close to three permissions. As our figure illustrates, there are a number of application providers that represent outliers because they request a vast amount of different permissions from their users.

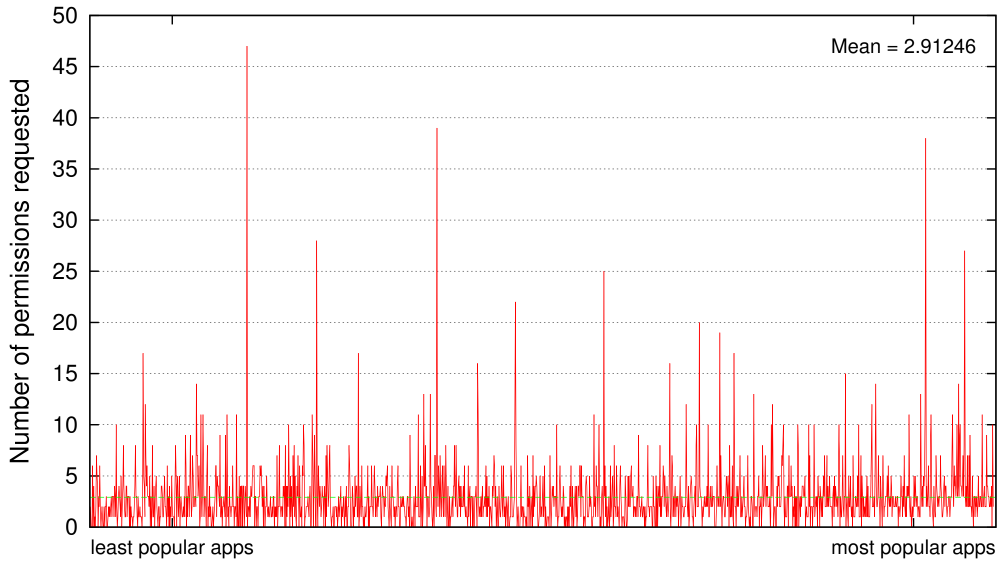


Fig. 4: Number of requested permissions for 1,646 application providers

Our results show that 40 providers (2.43% of all application providers in our sample) requested more than 10 permissions. For these 40 providers, we manually verified whether the requested permissions were in required for application functionality. Our findings suggest that a number of applications genuinely required a large amount of permissions to function at all. These legitimate applications often transferred large amounts of personal information to create their own specialized social networks. Examples of such applications include dating and job seeking applications. Dating applications, for example, gathered personal information to offer matchmaking features. The majority of providers that requested more than 10 permissions, did, however, request more permissions than were required to function. Especially one application provider set a striking example of application misuse. This provider offered a total of 140 applications for proverbs, quotes, and daily horoscopes in a number of different languages. However, the majority of their applications did not require any personal information at all to function. For example, while access to a user's birthday is required to generate horoscopes, the provider requested 27 different permissions in total. The provider's most popular application was a daily horoscope in Portuguese with 2.5 million monthly users (as of November 2012).

B. Hosting environments

Our analysis of reverse DNS queries and network hops showed that developers relied upon 604 distinct Internet hosting services. Table III outlines the most commonly used hosting services and their geographical location. Amazon's elastic cloud service hosted 18.72% of all third-party applications in our sample. Amazon EC2 was especially popular with developers of applications that attracted a large number of active users. We also observed that the hosting services were geographically spread across 64 different countries, although our analysis showed that the majority of applications (55%) were hosted in the United States.

Our analysis module probed all 1,646 application hosts for open TCP ports and corresponding software products. All applications were accessible via HTTP but 11.5% of all applications did not offer access via HTTPS, which made those applications inaccessible via a secure connection. Our detailed results show that 55% of web servers were powered by Apache httpd, followed by nginx (15.63%) and Microsoft

Provider	Location	Total %
Amazon EC2	US (755), IE (82), SG (52)	18.72%
SoftLayer	US (505)	10.65%
Peak Hosting	US (244)	5.14%
Rackspace	US (147), GB (11), HK (4)	3.41%
GoDaddy	SG (51), US (29), NL (6)	1.82%
Linode	US (72), GB (6), JP (2)	1.69%
OVH	FR (42), PL (7), ES (2)	1.04%
Hetzner	DE (47)	0.99%
Internap	US (35)	0.73%

TABLE III: Most commonly used Internet hosting services for Facebook third-party applications

IIS (9.4%). An accessible web server that handles both HTTP and HTTPS is the only requirement for a working Facebook third-party application. However, we found that third-party developers exposed a number of additional services on their application hosts. Table IV outlines the most common publicly exposed services. It shows that 40.24% of application hosting environments allowed access via SSH and 38.91% access via FTP. The used products, together with their specific software versions, were used by the analysis module to identify hosts with potential software vulnerabilities. Our security analysis showed that HTTP and FTP services posed the highest risks. Two hosts ran an outdated nginx version that is prone to a source code disclosure vulnerability (CVE-2010-2263). Furthermore, we found two outdated versions of ProFTPD that possibly allow an attacker to execute arbitrary code on application hosts (CVE-2006-5815, CVE-2010-4221). Eight hosts were susceptible to these buffer overflow attacks via their FTP service. The most popular of the eight application hosts gathered information from an average of 1.2 million users per month. This vulnerable application provider furthermore processes sensitive personal information such as user email addresses and dates of birth.

TCP Port	Service	Hosts	% Total
22	ssh	662	40.22%
21	ftp	640	38.88%
25	smtp	572	34.75%
110	pop3	439	26.67%
143	imap	417	25.33%

TABLE IV: Most common additional services on application hosts

C. Web trackers

Our web tracker submodule identified 139 distinct web trackers used by social apps. Table V outlines the most common web trackers detected in our application sample. Google dominated both with their web analytics product and their online advertising products DoubleClick and AdWords. Our analysis showed that web trackers were mostly planted by online advertising products, which are used to create additional revenue for application developers. All web bugs presented in Table V potentially track their users across multiple websites based on HTTP cookies.

The ranking of web trackers changes slightly when the popularity of the different applications is factored in. Based on the total number of application users exposed to web trackers, LifeStreet Media becomes the most popular advertising product. Furthermore, analytics based on BlueKai and advertising by Rubicon move to the top ten products when ranking is based on cumulative monthly active users instead of the cumulative occurrences.

Web bug	Type	Apps	% Total
Google Analytics	analytics	3,378	71.16%
DoubleClick	advertising	529	11.14%
Google Adsense	advertising	361	7.61%
AdMeld	advertising	276	5.81%
Cubics	advertising	153	3.22%
LifeStreet Media	advertising	94	1.98%
Google AdWords	advertising	91	1.92%
OpenX	advertising	82	1.73%
Quantcast	analytics	49	1.03%
ScoreCard Beacon	analytics	48	1.01%

TABLE V: Common web trackers in third-party apps

D. *Information leaks*

Our findings suggest that ten advertising and analytics products directly received users' unique identifiers from social networking applications via URI requests. One advertising provider even received our test user's birthday and gender in addition to the unique identifier. The following request shows an instance of the detected application provider; however, we have replaced the actual host with a fictional name.

Information leakage via URI request

```
GET /1111111111/landingbirthday=5%2F2%2F1978&gender=male
Host: notdisclosed.com
```

We observed that 315 social networking applications in our sample directly transferred personally identifiable information to at least one additional third-party product via HTTP parameters. Three out of these 10 products were also previously classified as web trackers. This implies that these three third parties can track users across multiple websites with additional knowledge of their unique Facebook identifier and, thus, their real name[7]. Two advertising products that received unique user identifiers via URI requests were also approved by Facebook as valid advertising products.

Our analysis showed that 51 applications leaked unique user identifiers to third parties via the HTTP Referer header. In addition to user identifiers, 14 out of these 51 applications also leaked their API authorization tokens via HTTP Referer. Third parties could misuse leaked OAuth tokens to impersonate the leaking apps and harvest additional personal information. Referers were in both cases mainly leaked to Google Analytics and DoubleClick. It became clear that a popular game was affected by this issue, leaking on average 4.7 million OAuth tokens and user identifiers per month to third-party analytics and advertising companies.

VI. Discussion and Limitations

In this section we discuss the implications of our findings and possible limitations of our approach.

A. *Detected malpractices*

The evaluation of our novel AppInspect framework on the basis of Facebook shows that automated security and privacy analysis of social networking apps is feasible. Our framework detected 14 application providers that requested a disproportionate amount of personal information. These application providers offer hundreds of applications and collect sensitive personal information from millions of social networking users. Our automated analysis showed that application providers make use of 139 different web tracking and advertising products. It furthermore showed that application developers transmit personally identifiable

[7]The Facebook Graph API allows to query certain information without the requirement of prior authentication. Given a user's unique identifier, one can simply perform a query like: http://graph.facebook.com/4; where "4" in this example is the unique user identifier of Mark Zuckerberg.

information to third parties. 315 applications directly transferred user identifiers to third-party products via URI parameters. In addition to receiving personally identifiable information, two out of ten products set tracking cookies. Hence, a single social networking app might lead to users being tracked across multiple websites with their real name. Web tracking in combination with personal information from social networks represents a serious privacy violation that is also not transparent to social networking users. Finally, our AppInspect framework detected that a number of applications leaked personal information and authentication tokens to third-party products via HTTP Referer headers. 51 applications leaked user information and out of these, 14 applications also leaked authentication tokens. We found a popular game that suffered from this implementation bug and leaked 4.7 million authentication tokens per month on average. After we ran our automated analysis, we manually verified all detected malpractices and implementation errors. We reported our findings to Facebook in November 2012. Facebook confirmed our findings and reached out to application developers to provide implementation fixes. In May 2013, Facebook confirmed that all of our detected malpractices have been fixed by application developers.

B. Application hosting infrastructure

The hosting infrastructure of social networking apps is beyond the control of social networking providers and users ultimately have to trust third-party developers with protecting their personal data appropriately. Our findings show that application developers rely on a wide range of custom systems to provide social networking applications. Over one third of all application hosts maintained publicly accessible FTP and SSH services. While these services offer proper administration tools, they also increase the attack surface of application hosts. For example, both FTP and SSH are well known to be popular targets of brute-force password guessing attacks. Moreover, a number of application hosts used outdated software versions that are susceptible to remote exploits. Our findings include an application host with more than 1 million monthly active users that was susceptible to a remote buffer overflow via their FTP service. Our analysis also showed that Amazon EC2 is a popular choice for application developers. Insecure Amazon EC2 community images may pose another security risk for third-party hosting infrastructures. Two recent publications [7], [2] came to the conclusion that Amazon's community images contain a number of serious vulnerabilities. Our findings furthermore showed that application servers were geographically spread over 64 different countries. This geographical distribution, finally, results in non-technical challenges because a great number of different data protection laws apply.

C. Implications and protection strategies

Since January 2010, application developers on Facebook can request users' personal email addresses instead of proxied email addresses. It is interesting to observe that 60.24% of all providers in our third-party application sample made use of this feature and requested the personal email addresses of their application users. Both social networking providers and application developers host a pool of sensitive personal information. Large social networking providers possess the necessary resources to maintain and improve the security of their services. In contrast, our findings suggest that a considerable number of third-party developer leak personal information to third parties and fail to harden their systems. While application developers collect email addresses to contact users directly, valid email addresses are also in demand with spammers and phishers. Forbes [16] reported that 1.1 million email addresses of social networking users are sold for as little as 5 US$. According to the seller, the information was collected via a Facebook third-party application. In addition to valid email addresses, third-party developers also collect information that enables sophisticated email based attacks. For example, social phishing attacks [23] leverage the success rate of traditional phishing messages based on knowledge of a user's friend. In the case of Facebook, all applications can access friendship information by default. Context-aware spam attacks [6] might also misuse user birthdays or photos to increase the authenticity of unsolicited bulk messages. Based on our findings, we propose the following protection strategies:

- Developers need to sanitize the landing page of their application and ensure that they do not pass on unique identifiers and authorization tokens via HTTP parameters.
- Developers need to provide third-party products that require a unique user identifier with random identifiers and maintain internal mapping between these random identifiers and the real Facebook user identifiers.
- Social network providers should stress that application developers should harden their hosting environments.

Finally, social network providers should follow the example of LinkedIn or iOS App Store and manually review apps before they make them available.

D. Limitations

Our AppInspect prototype is currently limited to Facebook applications, but a number of modules can be reused when extending our prototype to other social networking providers. Our in-depth analysis was limited to the most popular Facebook applications due to Facebook's anti-crawling mechanisms. Finally, due to the non-intrusiveness of our performed security tests, our results indicate the vulnerability of application hosts and may contain false positives and negatives.

E. Dataset

We make our social networking dataset available to the research community[8]. Furthermore, our collected dataset offers an important snapshot on application popularity, as Facebook stopped to make exact usage metrics publicly online in January 2013. *AppInspect* is designed to steadily provide insights into third-party application ecosystems, and we will, therefore, periodically refresh and expand our dataset.

VII. Related Work

This section surveys related research regarding social networking app security and privacy.

Social application studies

To the best of our knowledge, there has been no study on security and privacy issues of social networking apps of a scope comparable to our work. Wang et al. [34] conducted the first measurement study regarding the data collection practices of third-party apps. Their study analyzed the 200 most popular applications from nine different categories of Facebook's discontinued application directory. Based on their collected dataset, their study showed the most commonly requested permissions of 1,305 Facebook applications in December 2010. The Wall Street Journal conducted an investigation into information gathered by the 100 most popular Facebook applications in May 2012 [1]. Their manual review of popular applications found that applications often seek permission to access sensitive information. Two recent studies provide additional insights into permission systems of third-party applications: Chia et al. [9] studied the effectiveness of user-consent permission systems through a data collection of Facebook apps, Chrome extensions and Android apps. They constructed a Facebook dataset with 27,029 apps by web scraping a social media analytics platform's list of Facebook applications. Chia et al. then collected the requested permissions, popularity, and ratings of apps in their dataset. The authors found that popularity and ratings are not reliable indicators of potential privacy risks associated with third-party applications. Frank et al. [17] relied on Chia et al.'s dataset and used unsupervised learning to detect permission request patterns. Their results showed that permission patterns of low-reputation apps differed significantly from high-reputation apps. Krishnamurthy and Wills discovered that online social networks commonly leak personally identifiable information [26]. Their observation was confirmed by investigative journalism of the Wall Street Journal, which found that both advertising and tracking products received social network

[8]AppInspect Datasets: http://ai.sba-research.org

user identifiers [32], [31]. In May 2011, Symantec also found that third-party applications leaked OAuth tokens to third parties [33] due to a now deprecated authentication scheme of Facebook.

Our AppInspect performs a fully automated analysis of requested permissions, information leaks, as well as the application hosting infrastructures. Previous research either focused exclusively on requested permissions or was limited to manually verifying a small number of applications.

User protection

related work on user protection focuses on three main research areas: security extensions to online social networks, privacy preserving third-party data access, and improved application authorization dialogs.

generic security extensions aim to hide personal information from social network providers as well as from third parties without stopping users from sharing information. guha et al. [20] proposed nyob, a method to substitute personal profile information with pseudorandom content. lucas and borisov [27] introduced flybynight, a tool that relies on public-key cryptography and a third-party application to exchange confidential messages via facebook. their concept only applies to messages; the remaining personal information is still exposed to social network providers and third-party developers. luo et al. [28] proposed facecloak, where social network providers receive fake profile information and real user data is stored encrypted on a separate server. users require passwords and a facecloak browser extension to restore the real information. facecloak's approach is similar to nyob with the exception of requiring additional servers. beato et al. [3] finally proposed "scramble", a generic method to shield confidential information from social networking providers. most of these apps no longer work, as either the backend servers have been shut down or the apps were just intended as proof of concept and have not been updated to the changed apis of facebook, e.g., requiring https communication to the third-party servers.

felt and evans [15] conducted a survey of the 150 most popular facebook applications in october 2007. based on their analysis, they proposed a privacy protection method for social networking apis. their method suggests providing third-party developers with no personal information at all but with a limited interface that only provides access to an anonymized social graph. developers would use placeholders for user data, which the social network providers would replace with actual user data. felt and evans' design is impracticable with state-of-the-art applications because the majority of applications require personal information to work. singh et al. [30] proposed the "xbook" framework for building privacy-preserving social networking applications. their xbook framework is based on information flow models to control what an application provider can do with the personal information they receive. while their approach mitigates privacy and security issues of apps, it would require all third-party developers to host their applications on the xbook platform. egele et al. [10] proposed fine-grained access control over application data requests. their suggested solution, called "pox", relies on a browser plugin that mediates application data access and a modified facebook api library for application developers. at the time of writing none of these privacy-enhancing frameworks has been adopted by social network application developers.

besmer et al. [4] evaluated a user interface prototype that would help users to choose which information they want to share with third-party applications. they found that privacy-conscious users would benefit from their new user interface, while careless users would continue to expose their personal information to third-party developers. wang et al. [34] evaluated two alternative application permission dialogs to help users understand better how third-party applications function. the authors then proposed interface design cues based on their user interface evaluation. in contrast to these previous findings, current authentication dialogs conceal privacy-relevant information from users (see figure 1).

VIII. Conclusions

Social networking applications have become a popular feature of online social networks and are used by millions of users every day. In exchange for additional features, users grant social networking apps permission to transfer their personal data to third-party services.

In this paper, we proposed a novel framework called *AppInspect* to automatically analyze security and privacy issues of social network third-party applications. Our *AppInspect* framework first enumerates applications available for a given social network provider. Next, *AppInspect* collects application metrics from the social network provider. In a last step, *AppInspect* installs third-party applications on test accounts and analyzes their network traffic. *AppInspect* analyzes the collected network traffic for existing tracking software, information leakage to third parties, and application hosting infrastructure. We have implemented our *AppInspect* framework and used it to evaluate Facebook's application ecosystem. *AppInspect* automatically enumerated *434,687* unique Facebook applications and analyzed the most popular applications in detail. Our findings helped improve the security and privacy of social networking users, and finally, our results showed that *AppInspect* is a practicable framework for detecting common malpractices of third-party applications on a large scale. We will make our dataset available to the research community and continue to expand and update our comprehensive dataset.

Acknowledgements

This research was funded by COMET K1, FFG - Austrian Research Promotion Agency and would not have been possible without the financial support of the Vienna PhD School of informatics. The authors would also like to thank Manuel Leithner, Maciej Piec, and Sebastian Neuner for their source code contributions to the AppInspect framework.

REFERENCES

[1] ANGWIN, J., AND SINGER-VINE, J. Selling you on facebook. *WSJ* (2012). http://on.wsj.com/HR3pYb.

[2] BALDUZZI, M., ZADDACH, J., BALZAROTTI, D., KIRDA, E., AND LOUREIRO, S. A security analysis of amazon's elastic compute cloud service. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (2012), SAC '12, ACM.

[3] BEATO, F., KOHLWEISS, M., AND WOUTERS, K. Scramble! your social network data. In *Privacy Enhancing Technologies*, vol. 6794. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 211–225.

[4] BESMER, A., LIPFORD, H., SHEHAB, M., AND CHEEK, G. Social applications: exploring a more secure framework. In *Proceedings of the 5th Symposium on Usable Privacy and Security* (2009), ACM, p. 2.

[5] BESMER, A., AND LIPFORD, H. R. Users' (mis)conceptions of social applications. In *Proceedings of Graphics Interface 2010* (Toronto, Ont., Canada, Canada, 2010), GI '10, Canadian Information Processing Society, pp. 63–70.

[6] BROWN, G., HOWE, T., IHBE, M., PRAKASH, A., AND BORDERS, K. Social networks and context-aware spam. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work* (2008), ACM, pp. 403–412.

[7] BUGIEL, S., NÜRNBERGER, S., PPPELMANN, T., SADEGHI, A.-R., AND SCHNEIDER, T. AmazonIA: when elasticity snaps back. In *Proceedings of the 18th ACM conference on Computer and communications security* (2011), CCS '11, ACM.

[8] CASTELLUCCIA, C., DE CRISTOFARO, E., AND PERITO, D. Private information disclosure from web searches. In *Privacy Enhancing Technologies* (2010), Springer, pp. 38–55.

[9] CHIA, P. H., YAMAMOTO, Y., AND ASOKAN, N. Is this app safe?: a large scale study on application permissions and risk signals. In *Proceedings of the 21st international conference on World Wide Web* (2012), WWW '12, ACM, pp. 311–320.

[10] EGELE, M., MOSER, A., KRUEGEL, C., AND KIRDA, E. Pox: Protecting users from malicious facebook applications. *Computer Communications* (2012).

[11] FACEBOOK. Getting your apps into facebook search faster. last accessed 04/27/2012 https://developers.facebook.com/blog/post/2011/07/12/getting-your-apps-into-facebook-search-faster/.

[12] FACEBOOK. Permissions reference. last accessed: 08/10/2012 https://developers.facebook.com/docs/authentication/permissions/.

[13] FACEBOOK. Platform policies. last accessed 04/20/2013 https://developers.facebook.com/policy/.

[14] FACEBOOK. Facebook platform launches, May 2007. last accessed 08/15/2012 https://developers.facebook.com/blog/archive.

[15] FELT, A., AND EVANS, D. Privacy protection for social networking APIs. In *W2SP '08* (2008).

[16] FORBES. Facebook investigating how bulgarian man bought 1.1 million users' email addresses for five dollars. last accessed 11/03/2012 http://www.forbes.com/sites/andygreenberg/2012/10/25/facebook-investigating-how-bulgarian-man-bought-1-1-million-users-email-addresses-for-five-dollars/.

[17] FRANK, M., DONG, B., FELT, A. P., AND SONG, D. Mining permission request patterns from android and facebook applications. *To appear: IEEE International Conference on Data Mining (ICDM) 2012* (2012).

[18] FTC. In the matter of facebook, inc., a corporation, Aug 2012. last accessed 06/05/2013 http://www.ftc.gov/os/caselist/0923184/120810facebookcmpt.pdf.

[19] GOOGLE. Google launches opensocial to spread social applications across the web, Nov 2007. last accessed 04/05/2012.

[20] GUHA, S., TANG, K., AND FRANCIS, P. Noyb: Privacy in online social networks. In *Proceedings of the first workshop on Online social networks* (2008), vol. 1, ACM, pp. 49–54.

[21] HARDT, D. The OAuth 2.0 authorization framework. last accessed 02/05/2012 http://tools.ietf.org/html/draft-ietf-oauth-v2-31.

[22] HULL, G., LIPFORD, H., AND LATULIPE, C. Contextual gaps: privacy issues on facebook. *Ethics and information technology 13*, 4 (2011), 289–302.

[23] JAGATIC, T., JOHNSON, N., JAKOBSSON, M., AND MENCZER, F. Social phishing. *Communications of the ACM 50*, 10 (2007), 94–100.

[24] King, J., Lampinen, A., and Smolen, A. Privacy: is there an app for that? In *Proceedings of the Seventh Symposium on Usable Privacy and Security* (2011), ACM, p. 12.

[25] Ko, M. N., Cheek, G., Shehab, M., and Sandhu, R. Social-networks connect services. *Computer 43*, 8 (2010), 37 –43.

[26] Krishnamurthy, B., and Wills, C. E. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks* (2009), ACM, pp. 7–12.

[27] Lucas, M., and Borisov, N. Flybynight: mitigating the privacy risks of social networking. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society* (2008), ACM, pp. 1–8.

[28] Luo, W., Xie, Q., and Hengartner, U. Facecloak: An architecture for user privacy on social networking sites. In *Computational Science and Engineering, 2009. CSE'09. International Conference on* (2009), vol. 3, IEEE, pp. 26–33.

[29] SEC. Amendment no. 4 to form s-1, facebook, inc., Apr 2012. last accessed 08/01/2012 https://www.sec.gov/Archives/edgar/data/1326801/000119312512175673/d287954ds1a.htm.

[30] Singh, K., Bhola, S., and Lee, W. xBook: redesigning privacy control in social networking platforms. In *Proceedings of the 18th conference on USENIX security symposium* (Berkeley, CA, USA, 2009), SSYM'09, USENIX Association, p. 249266.

[31] Steel, E., and Fowler, G. a. Facebook in privacy breach. *Wall Street Journal* (2010).

[32] Steel, E., and Vascellaro, J. E. Facebook, MySpace confront privacy loophole. *Wall Street Journal* (2010).

[33] Symantec. Facebook applications accidentally leaking access to third parties. last accessed 06/20/2012 http://www.symantec.com/connect/blogs/facebook-applications-accidentally-leaking-access-third-parties-updated.

[34] Wang, N., Xu, H., and Grossklags, J. Third-party apps on facebook: privacy and the illusion of control. In *Proceedings of the 5th ACM Symposium on Computer Human Interaction for Management of Information Technology* (2011), ACM, p. 4.