FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Security and Privacy Implications of Third-Party Access to Online Social Networks

PhD THESIS

submitted in partial fulfillment of the requirements of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Markus Huber, M.Sc.
Registration Number 0306665

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl
Second advisor: o.Univ.Prof. Dipl.Ing. Dr. A Min Tjoa

External reviewers:
Assoc. Prof. Dr. Engin Kirda. Northeastern University, USA.
Prof. Dr. Stefan Katzenbeisser. Technische Universität Darmstadt, Germany.

Wien, 27.08.2013

(Signature of Author) (Signature of Advisor)

Declaration of Authorship

Markus Huber, M.Sc.
Burggasse 102/8, AT-1070 Vienna, Austria

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

(Vienna, 27/08/2013)	(Signature of Author)

Acknowledgements

I am grateful to my supervisor Edgar R. Weippl for his excellent mentoring over the course of my postgraduate studies and for giving me the freedom to pursue my own research ideas. I was also fortunate to be surrounded at SBA Research by a group of determined security researchers under Edgar's leadership. I am grateful to all of my collaborators, particularly working with Martin Mulazzani, Sebastian Schrittwieser, and Manuel Leithner was always a pleasure and our discussions resulted in plenty of research ideas. I also like to thank Hannes Werthner and Clarissa Schmid for their efforts to organize and create an interesting curriculum for the Vienna PhD school of informatics. Especially the fundamental research course by Georg Gottlob helped me to plan my PhD student years. I would also like to thank my fellow students from the Vienna PhD school of informatics, who helped to broaden my horizon of computer science and were always fun to socialize with. I am glad to have had the opportunity to visit Alessandro Acquisti's group at Carnegie Mellon University as a research intern and gain valuable experience. I am furthermore grateful to the Information Security Group (ISG) at the Universitat Politècnica de Catalunya for providing me with a place to work throughout my last postgraduate year in Barcelona.

My research depended to a large degree on free software. I am indebted to the entire free software movement, in particular the developers of the GNU, Linux, Ubuntu, LaTeX, and Python software. This thesis would not have been possible without the financial support of the Vienna PhD School of informatics, as well as COMET K1 and project n° 825747 funded by FFG - Austrian Research Promotion Agency.

Above all, I am grateful for support from my parents Ursula and Siegfried Huber, who motivated me to pursue my love for computer security and supported me throughout the years. Finally, I would like to thank Babsy for her endless love and patience during the creation of my thesis.

Abstract

Online Social Networks (OSNs) recently became one of the most popular web services, and these services are used by hundreds of millions of users worldwide. People rely on OSNs to foster social relationships, exchange media files, and to communicate online. OSN users in general provide their real world identity, and social networking providers encourage their users to share a plethora of personal information via their services. Access to the fast amounts of personal information is exclusively governed by social network providers.

In the past decade, research on OSN security and privacy has focused to a large extend on the information-disclosure behavior of OSN users, and the resulting challenges for the privacy management of social networking profiles. However, little research has been conducted on third-party access to social networking data, and third-party applications in particular. Third-party applications, or colloquial "apps", are popular extensions to OSNs and their current modus operandi allows them to transfer personal information out of walled gardens of OSNs. In this thesis, we strive to advance research on the security and privacy implications of OSN third-party access, and hereby tackle three main research problems. The first non-trivial research problem we attempt to solve, is the automated analysis of OSN application ecosystems regarding their security and privacy practices. Second, current methods for digital forensics are rendered useless by the distributed nature of OSNs, and we therefore investigate into novel methods to collect digital evidence from OSNs. Third, we want to estimate the impact of social network's weak communication security practices for advanced session-hijacking attacks.

We develop *AppInspect*, an automated framework to analyze OSN application ecosystems for security and privacy malpractices. Our results helped to mitigate a number of information leaks in popular OSN third-party applications. We furthermore outline a novel approach to collect digital evidence from OSNs, called *Social Snapshots*, which makes an important contribution to the area of digital forensics. Finally, our third main result, *Friend-in-The-Middle (FiTM)* attacks, describes a number of advanced attacks based on hijacking OSN user sessions. Our results stressed the importance of proper OSN communication security, and social networking provider began to acknowledge our findings. We expect that all major social networking providers make encrypted communication protocols their default in the near future.

Kurzfassung

Soziale Online Netzwerke (englische Abkürzung: “OSNs”) zählen heute zu den beliebtesten Internetservices und werden weltweit von hunderten Millionen Menschen benutzt. Soziale Netzwerke werden hierbei verwendet um online: Freundschaften zu pflegen, digitale Inhalte auszutauschen, als auch um zu kommunizieren. Benutzer von sozialen Medien verwenden in der Regel ihre wirkliche Identität. Des Weiteren animieren Betreiber von sozialen Netzwerken ihre Benutzer dazu eine Vielzahl von persönlichen Informationen mittels ihrer Services zu veröffentlichen. Der Zugriff auf die beträchtlichen Datenmengen von persönlichen Informationen wird ausschließlich durch die Betreiber von sozialen Netzwerken geregelt.

Die Forschung im Bereich der Sicherheit und Privatsphäre von sozialen Netzwerken hatte sich in den vergangenen Jahren hauptsächlich auf das Informationsveröffentlichungsverhalten von Benutzern sowie auf die sich daraus ergebenen Herausforderungen für die Verwaltung von Profilinhalten fokussiert. Im Gegensatz dazu, hat sich die Forschung relativ wenig auf den OSN Datenzugriff von Dritten, und insbesondere Drittanbieteranwendungen spezialisiert. Bei Drittanbieteranwendungen, oder umgangssprachlich „apps“, handelt es sich um beliebte Erweiterungen deren aktuelle Funktionsweise den Transfer von persönlichen Informationen außerhalb der geschützten OSN Plattformen ermöglicht. Mit dieser Dissertation streben wir eine Erweiterung des aktuellen Wissensstands zu den Auswirkungen von Drittanbieterzugriffen auf die OSN Benutzer-Sicherheit und –Privatsphäre an. Das erste komplexe Forschungsproblem das wir hierbei behandeln, ist die automatische Analyse von Drittanbieteranwendungen auf Sicherheits- und Privatsphären-Defizite. Die verteilte OSN Infrastruktur verhindert den Einsatz von bestehenden Methoden der digitalen Forensik. Daher fokussieren wir in unserer zweiten Forschungsfrage auf neuartige Methoden, um digitale Beweise aus sozialen Netzwerken zu sammeln. Als dritte Forschungsfrage beschäftigen wir uns mit der Ausnutzung der mangelnden OSN-Verbindungssicherheit für ausgeklügelte Session-Hjiacking Angriffe.

Die erste Forschungsfrage brachte ein automatisiertes System zum Aufspüren von Defiziten bezüglich Sicherheit und Privatsphäre von Drittanbieteranwendungen, mit dem Namen *AppInspect*, hervor. Unsere Ergebnisse führten zur Beseitigung einer Reihe von Informationslecks in populären Anwendungen. Diese Arbeit leistet des Weiteren, mit unserer *Social Snapshot* Software zur OSN Datensammlung, einen wichtigen Beitrag im Bereich der digitalen Forensik. In Folge, stellt unser drittes Hauptergebnis die sogenannten *Friend-in-The-Middle (FiTM)* Angriffe dar. Unsere Ergebnisse heben die Wichtigkeit von OSN-Verbindungssicherheit hervor und OSN Betreiber beginnen diese Erkenntnis zu berücksichtigen. In Folge erwarten wir, dass die führenden Betreiber von sozialen Netzwerken in naher Zukunft alle Verbindungen standardmäßig verschlüsseln.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Online Social Networks (OSNs) are in the epicenter of today's Internet and recently became one of the most popular web services. Consequently, OSNs are used by hundreds of millions of users worldwide, and people rely on OSNs to foster social relationships, exchange media files, and to communicate online. People in general, use their real world identity, and social networking providers encourage them to share a plethora of personal information via OSNs, which includes: dates of birth, email addresses, circles of friends, and personal photos. Access to the fast amounts of personal information is exclusively governed by social network providers.

Social networking providers invite developers to extend their platform functionality with third-party applications. Popular third-party applications such as games ultimately enable application developers to gain access to the walled gardens of personal information collected by OSNs. Research on OSN security and privacy in the past decade has focused, to a large extend, on the information-disclosure behavior of OSN users and the resulting challenges for the privacy management of social networking profiles. Previous research helped to increase user awareness of unwanted information-disclosure, and also led to the adaption of fine grained privacy controls by major OSN services. Little research has however been conducted on third-party access to social networking data, and third-party applications in particular. The aim of this thesis is thus, to advance research regarding security and privacy implications of third-party access to social networking data. In particular, we strive to automatically screen application ecosystems for potential privacy malpractices, and security shortcomings. Automated analyses of application ecosystems help to protect social networking users from privacy violations and the misuse of their data. Furthermore, this thesis analyzes the use of custom applications in the area of computer security. Hereby, we want to develop new methods to support the data extraction of social networking data for forensic investigations, and academic research. Finally, we want to create awareness for the misuse of weak communication security in combination with custom applications before these shortcomings are going to be exploited in the wild.

1.2 Problem Statement

This thesis addresses the implications of third-party access to social networking data for user privacy and security. Hereby, we elaborate three specific research problems in depth. First, third-party applications are popular OSN extensions and their security and privacy issues have not been studied at large scale. Second, people increasingly use OSNs to communicate online and new methods for digital forensics are therefore required to collect digital evidence from social networking providers. Third, social networking providers often rely on insecure communication protocols and the impact of this security shortcoming requires additional research. In the following we describe our three research problems in more detail.

Analysis of Third-Party Application Ecosystems

Third-party applications (apps) are used by hundreds of millions of social network users every day. These apps offer extensions to social networking services, in exchange for personal information from users. In addition, a number of unique privacy and security challenges arise due to the transfer of personal data to application developers. Application developers rely on custom hosting infrastructure to process and store personal data, which might contain security weaknesses. Furthermore, the information transferred to third-party applications might leak to additional advertising and analytics providers. The first research problem is thus concerned with methods to analyze OSN application ecosystems for security and privacy shortcomings on a large scale.

Evidence Collection for Digital Forensics

Digital forensics is concerned with developing methods to recover and investigate digital evidence for criminal investigations. Methods for digital forensics traditionally focus on filesystems, logs, and databases. With the growing popularity of social networking services, the importance of OSNs for criminal investigations is increasing as well. The distributed nature of OSNs renders state-of-the-art methods for digital forensics useless, and novel approaches for evidence collection from OSNs are thus required. Data associated with social networking profiles can gain fast extends and methods to visualize the collected data are therefore another important problem. The second research problem investigates into forensic methods to collect, process, and visualize digital evidence from OSNs.

Impact of Session-Hijacking Attacks

The design and architecture of social networking services has important implications for the privacy and security of its users. One especially important architectural feature is the communication security of data exchanged between users and OSN providers. OSNs currently offer only partial support for exchanging information via secure and encrypted communication channels. It has to be taken into consideration that insufficient protection against session-hijacking attacks, enables attackers to misuse OSN accounts. The third research problem is thus concerned with estimating potential misuse scenarios of hijacked user accounts, and to estimate the impact of sophisticated attacks based on hijacking OSN user accounts.

1.3 Main Results

The main results are organized in accordance to the previously stated research problems. The first main result consists in a framework to automatically analyze OSN third-party application ecosystems, called *AppInspect*, which we further describe in Chapter 4. Our second main result consists in a novel method to extract digital evidence from OSNs via so called *Social Snapshots*, which corresponds to Chapter 5 in this thesis. In this section we furthermore briefly discuss our third main result, called *Friend-in-the-Middle (FiTM)* attacks, which we cover in more detail in Chapter 6.

Methodological Approach

The aim of thesis is not only to present new theoretical methods for social network privacy and security but also to practically evaluate them. Hereby, we develop proof-of-concept implementations, and release our tools under an open-source license where it is applicable. Furthermore, we share our collected data with the research community to advance research in the area of OSN security and privacy. The novel methods presented in this thesis can be applied to the great majority of OSNs, and we primarily tested our approaches on the basis of Facebook, which is currently the most mature and popular social networking service.
The specific methodologies used to tackle our three main research problems, are further discussed in each Chapter separately (see Subsections 4.2, 5.3, and 6.2).

1.3.1 Evaluation of Third-Party Application Ecosystems

First of all, we discuss generic methods to assess the security and privacy of OSN application ecosystems in an automated fashion. The practical implementation of our generic methods for Facebook helped to mitigate a number of privacy and security shortcomings of popular applications. Our main contributions are as follows:

- We present a novel framework for automated privacy and security analysis of social networking apps, called *AppInspect*. Our *AppInspect* framework consists of three main modules to: enumerate available applications for a given OSN, classify applications, and perform in-depth analyses of their hosting infrastructure and network traffic.

- We evaluate the feasibility of our *AppInspect* framework on basis of Facebook and were able to enumerate *434,687* third-party applications. We show that a relatively small number of applications within our subsample ($\sim$10,000 apps), attracts the great majority of users.

- We assess the security of third-party hosting infrastructure and automatically detect security and privacy malpractices of application providers for the most popular Facebook applications.

- We make our enumerated dataset of *434,687* unique Facebook applications available to the research community. Furthermore, we use our *AppInspect* framework to periodically refreshed and extend our dataset.

1.3.2 Digital Forensics with Social Snapshots

We present a novel method to collect data from OSNs, called *Social Snapshots*, which relies on an automated web browser in combination with a customized third-party application. Our novel *social snapshot tool* makes the following contributions in the area of digital forensics:

- We introduce novel techniques to efficiently gather data from online social networks for the purpose of forensic analyses as well as for academic studies. Our tool gathers more data than possible with today's approaches and makes it feasible to link "online evidence" to traditional forensic artifacts found on computers, using state-of-the-art forensic methods.

- We implement a prototype of our social snapshot tool for Facebook, and release it under an open-source license.

- We perform an experimental evaluation involving a real-world test with volunteers and show that our approach is both practicable and feasible.

- We discuss possible methods to visualize collected social networking data and furthermore provide practical examples of these social snapshot visualizations.

1.3.3 Evaluation of Session Hijacking Attacks

In our third main results we present *Friend-in-The-Middle* (FiTM) attacks which describe advanced attacks based on hijacking OSN user sessions. We furthermore discuss the misuse of hijacked user sessions for large scale spam campaigns. Our third main result makes the following contributions:

- We introduce *Friend-in-The-Middle (FiTM) attacks* which enable automated context-aware spam campaigns on a large scale. Hereby, we outline three attacks which are enabled through hijacking social networking user sessions.

- We estimate the potential risk of OSNs session hijacking attacks on the basis of two experiments. First, we show that public WiFi access points are a potential source for session hijacking attacks. Second, we show that Tor exit nodes pose another risk for social networking users and a further potential starting point for FiTM attacks.

- We identify optimal attack strategies for our FiTM attack on the basis of a generated Facebook subgraph.

- We simulate a large-scale spam campaign to estimate the impact of the potentially hijacked OSN users of our two measurement experiments.

- We show that a relatively small number of initially hijacked user sessions might lead to hundreds of thousands users being targeted with context-aware spam.

1.4 Structure of the Work

In Chapter 2 we introduce basic notions and definitions used within this thesis. In addition we discuss third-party applications in Section 2.3, which constitute a fundamental concept for the three main contributions of this thesis. In Chapter 3 we survey the state-of-the-art research in the area of social network security and privacy. This Chapter provides an extensive overview over existing threats and protection mechanisms of OSNs. We outline privacy-related threats in Section 3.1, followed by security-related threats in Section 3.2. Finally, we discuss possible mitigation strategies in Section 3.3. The following three chapters outline the three main contributions of this thesis, whereas:
Chapter 4 describes our large-scale evaluation of OSN application ecosystems. Hereby, Section 4.1 outlines the design and functionality of *AppInspect*, our framework to analyze third-party applications. In Section 4.2, we describe the evaluation of our framework on Facebook and our acquired application sample. Section 4.3 presents the privacy and security shortcomings we detected in Facebook's application ecosystem. We then discuss the implications of our findings in Section 4.4. We outline our second main result: a novel method for collecting data from OSN, called *social snapshots*, in Chapter 5. Hereby, Section 5.1 describes the design of our social snapshot tool and its different modules. The methodology and our proof-of-concept implementation are described in Section 5.3. We discuss different visualizations of social snapshots for forensic purposes in Section 5.2. Our findings are outlined in Section 5.4 and further discussed in Section 5.5. Chapter 6 presents the third main result of this thesis: a sophisticated attack based on hijacking social network sessions, called *Friend-in-The-Middle (FiTM)* attack. Section 6.1 outlines our novel attack and discusses large-scale spam campaigns. We explain the methodology used to evaluate the practicability and impact of our attack in Section 6.2, while we present our findings in Section 6.3. Finally, we discuss our results and possible mitigation strategies in Section 6.4.
Chapter 7 finally summarizes the main results of this thesis and furthermore compares our findings with existing research in Section 7.2, before we discuss future research in Section 7.3.

This thesis is based on the following publications:

- M. Huber, M. Mulazzani, and E. Weippl. Who on earth is mr. cypher? automated friend injection attacks on social networking sites. In *Proceedings of the IFIP International Information Security Conference 2010: Security and Privacy*, 9 2010
- M. Huber, M. Mulazzani, E. Weippl, G. Kitzler, and S. Goluch. Friend-in-the-middle attacks: Exploiting social networking sites for spam. *IEEE Internet Computing*, 15(3):28–34, 2011
- M. Huber, M. Mulazzani, M. Leithner, S. Schrittwieser, G. Wondracek, and E. Weippl. Social snapshots: digital forensics for online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011
- M. Mulazzani, M. Huber, and E. Weippl. Social network forensics: Tapping the data pool of social networks. *Eighth Annual IFIP WG 11.9 International Conference on Digital Forensics*, 1 2012
- M. Huber, M. Mulazzani, S. Schrittwieser, and E. Weippl. Appinspect: Large-scale evaluation of social networking apps. In *Proceedings of ACM conference on Online Social Networks (COSN)*, 2013

CHAPTER 2

Preliminaries

2.1 Online Social Networks (OSNs)

Online Social Networks (OSNs) are a recent phenomenon and account for the most popular web services at the time of writing. One of the first researchers that described the phenomenon of online social networks, were Boyd and Ellison [37] who provided an overview of OSNs in a historical and scholarly context. The authors defined social networking sites (SNSs) as web-based services that allow individuals to: (1) construct a public or semi-public profile within a bounded system, (2) articulate a list of other users with whom they share a connection, and (3) view and traverse their list of connections and those made by others within the system. While we agree with their fundamental definition, we refer to social networking sites as "online social networks" (OSNs) throughout this thesis. OSN describes a more general term, which better accounts for the growing ubiquity of social networking services as web-based services, mobile applications, and social extensions to regular web sites. State-of-the- art online social networks can be divided into two main classes which we describe in the following.

2.1.1 General-purpose OSNs

General-purpose online social networks describe a class of OSNs which Bonneau and Preibusch [33] defined as: anybody is free to join, people commonly present their real-world identity, and the primary use of the site is interacting with others via profile pages. Web services whose primary purpose consists in sharing content (e.g. YouTube, Flickr, or Tumblr), are thus not included in the definition of general-purpose OSNs. General-purpose OSNs affect the area of information security and privacy because users provide their real-world identity and a plethora of personal information to third-party web services. The most popular examples for general-purpose OSNs are currently Facebook and Google+. While Facebook and Google+ have a global user base, a number of general-purpose OSNs cater specific geographic locations. Popular regional online social networks include Qzone and Renren which are used in mainland China, their popularity can be attributed to the inaccessibility of Facebook's and Google's web services in China. Other popular regional ONSs include VKonakte, which is primarily used in Russian-speaking countries, and Tuenti which is a popular online social network in Spain.

2.1.2 Niche OSNs

Except from general-purpose OSNs a number of online social networks fill specialized niches. This class of services either offers a functionality subset of general-purpose OSNs, or these services are used in significantly different ways. *Business-networking* OSNs are a popular niche of online social networking services whereas people focus on exchanging professional information. This business-networking OSNs offer online services to foster business relationships and to hunt for jobs. LinkedIn and XING are two prominent examples for this OSN niche. *Media recommendation* OSNs describe a niche which focuses on sharing of music, movies, and book tastes and provide media recommendation systems. Popular examples of media recommendation OSNs include Last.fm for music, Flixster for movies, and Goodreads for books. *Reunion* OSNs focus on facilitating off-line communication as compared with on-line communication of general-purpose OSNs. Reunion OSNs, such as Classmates.com or MyLife, help former classmates, work colleges, or servicemen to exchange contact details for reunions. *Activity-focused* OSNs enable its users to perform certain activities. Gaia Online, for example, provides online services for social gaming. Another example for this niche is Couchsurfing, which connects travelers and acts as a provider for hospitality services. *Privacy-focused* OSNs focus on user privacy and compose another OSNs niche. Privacy-focused OSNs often rely on decentralized architectures and the currently most widely used privacy-focused OSN is Diaspora. Subsection 3.3.3 provides a further overview of current academic proposals for decentralized OSN architectures.

Name	**Protection**	**Requirements**	**User**	URI
Facebook	General-purpose	Global	1,000,000,000	`http://facebook.com`
Qzone	General-purpose	CN	597,000,000	`http://qzone.qq.com/`
Google+	General-purpose	Global	500,000,000	`https://plus.google.com`
LinkedIn	Business-networking	Global	200,000,000	`http://www.linkedin.com/`
Vkontakte	General-purpose	RU	200,000,000	`http://vk.com`
Renren	General-purpose	CN	160,000,000	`http://renren.com/`
Flixster	Media recommendation	Global	63,000,000	`http://www.flixster.com`
MyLife	Reunion	US, CA	51,000,000	`http://www.mylife.com/`
Classmates.com	Reunion	US	50,000,000	`http://www.classmates.com`
Last.fm	Media recommendation	Global	30,000,000	`http://last.fm`
Gaia Online	Activity-focused	Global	23,523,663	`http://www.gaiaonline.com/`
Tuenti	General-purpose	ES	14,000,000	`https://www.tuenti.com/`
XING	Business-networking	DE, AT, CH	11,100,000	`https://www.xing.com`
Goodreads	Media recommendation	Global	10,000,000	`http://www.goodreads.com`
CourchSurfing	Activity-focused	Global	2,967,421	`https://www.couchsurfing.org`
Diaspora	Privacy-focused	Global	406,960	`http://diasporaproject.org`

Table 2.1: Popular online social networking services as of March 2013

Table 2.1 outlines a number of popular online social networking services. Numbers on active users are based on statistics from March 2013. It should be noted that the numbers of active users represent self-reported statistics by OSN providers.

2.2 OSN graph fundamentals

The underlying structures of online social networks represent graphs. In this section we will therefore briefly outline common notations of graph theory, discuss graph metrics, as well as methods to create graph models.

Social networking graphs represent simple graphs with undirected edges because friendships in OSNs are in general mutual. We consider an OSN to be an undirected social graph $G = (V, E)$ with $v_i \in \mathbb{N}$ and $E \subseteq [V]^2$, where the nodes $v_i \in V$ represent users and the vertices $\{v_i, v_j\}$, denoted as $v_i v_j \in E$, represent connections between two users v_i and v_j.

2.2.1 Social Network Analysis and Graph Metrics

Social Network Analysis (SNA) is concerned with the methodical analysis of social networks. Relationships are hereby analyzed with methods of network theory, which itself is an area of graph theory. Borgatti et al. published an article describing the history of social network analysis [34] and provides a brief introduction to the most import SNA concepts. In this Subsection we briefly discuss two fundamental graph metrics: centrality, and segmentation. We point readers interested in a comprehensive overview on social network analysis to Newman [142].

Centrality metrics

In a graph some vertices are more important and influential than others and centrality metrics are used to describe this property. The most straightforward centrality metric, is *degree centrality* k, which for an undirected graph G represents the number of edges connected to a vertex i. Hence, it follows n for $|V| = n$:

$$k_i = \sum_{j=1}^{n} a_{ij} \tag{2.1}$$

Eigenvector centrality provides a more sophisticated metric of graph centrality, which gives each vertex a score proportional to the sum of scores in its neighbors. The eigenvector centrality x_i of vertex i is proportional to the sum of the centralities x_j of vertex i's neighbors, whereas λ_1 denotes the largest eigenvalue of the adjacency matrix A:

$$x_i = {\lambda_1}^{-1} \sum_j a_{ij} x_j \tag{2.2}$$

Betweenness centrality is another important centrality metric, which measures to which extent a vertex lies on paths between other vertices. The betweenness centrality $C_B(v)$ for a vertex v is defined as:

$$C_B(v) = \sum_{u \neq v} \sum_{w \neq v} \frac{\sigma_{uw}(v)}{\sigma_{uw}} \tag{2.3}$$

Here σ_{uw} denotes the number of shortest paths between vertex u and vertex w, and $\sigma_{uw}(v)$ the number of shortest paths between those vertices that run through v.

Segmentation

Social graphs divide naturally into groups and communities, e.g. users who share the same co-workers or come from the same home town. Handcock et al. [90] showed that common affiliations implicate clustering within social networks. It can be assumed that OSN graphs are segmented into a number of subgraphs, e.g. V_i' could consist of vertices that share certain properties *nationality, gender* or *affiliations*. The analysis of graph segmentation and algorithms for group detection form important research problems in social network analysis. A basic concept of graph segmentation are *cliques*, which represent subsets of users whereas all members in the subset are connected with each other. The *clustering coefficient* measures the average probability that two randomly selected neighbors of a vertex are connected by an edge. Let k_i be the number of neighbors of a vertex i and e_i the sum of all edges between them. If each pair of neighbors of vertex i were connected by an edge, then there would be $k_i(k_i - 1)/2$ edges in an undirected network. Therefore, the clustering coefficient C_i of a vertex i is:

$$C_i = \frac{2e_i}{k_i(k_i - 1)} \tag{2.4}$$

The clustering coefficient thus shows how close the neighbors of i are to form a clique. Groups and communities often do not form strict cliques and instead form separate dense regions. General approaches for community detection thus identify and remove spanning links within graphs to segment it into a number of community subgraphs. The common Newman-Girvan algorithm [143] for examples performs group detection based on the betweenness centrality scores of graph vertices.

2.2.2 Subgraphs

The complete social graph is only known to OSNs operators and complete graph models are difficult to compute. The total number of vertices in popular OSNs have vast extents, e.g. Facebook has currently a total amount of $|V| = 4 \cdot 10^9$ user vertices (see Table 2.1). In practice, it is thus not feasible to analyze complete OSN graphs. Instead, a partial network is often created by selecting a subgraph of the complete network: if $V' \subseteq V$ and $E' \subseteq E$, then $G' = (V', E')$ is a *subgraph* of G, written $G' \subseteq G$. If $E' = \{v_i v_j \in E \mid v_i, v_j \in V'\}$, then G' is called an *induced subgraph*. A graph is called *simple* when it has no *self-loops* and no *multiple edges* between any pair of nodes. The *degree* $d_G(v)$ of a node v is the number of neighbors which is equal to the number of edges on v, denoted $|E(v)|$. Note that for an induced subgraph G' the degree of a node $v \in V' \subseteq V$ is $d_{G'}(v) \leq d_G(v)$. We say that an *extended induced subgraph* is a simple graph with following enhancements of V' and E': Let $\widetilde{E'}$ be the set of edges with one endpoint being in V' : $\widetilde{E'} := \{v_i v_j \in E \backslash E' | v_i \in V'\}$ then $\widetilde{V'} := \{v_j \in V \backslash V' | v_i v_j \in \widetilde{E'}\}$. $\widetilde{V'}$ contains all vertices which have a direct connection to a vertex within the induced subgraph vertices V'. With this extension we obtain a new graph $\widetilde{G} := (\widetilde{V}, \widetilde{E})$ with $\widetilde{V} = V' \cup \widetilde{V'}$ and $\widetilde{E} = E' \cup \widetilde{E'}$.

We assume that a complete social networking graph, denoted as $SNG = (V_{SNG}, E_{SNG})$, consists of $n \in \mathbb{N}, n < \infty$ extended induced subgraphs $\{G_1, G_2, ..., G_n\} \subseteq SNG$ with a *pairwise almost disjoint* property: $|G_i \cap G_j| < \varepsilon$ with $i, j = 1, ..., n \wedge i \neq j$ and $\varepsilon \in \mathbb{N}$.

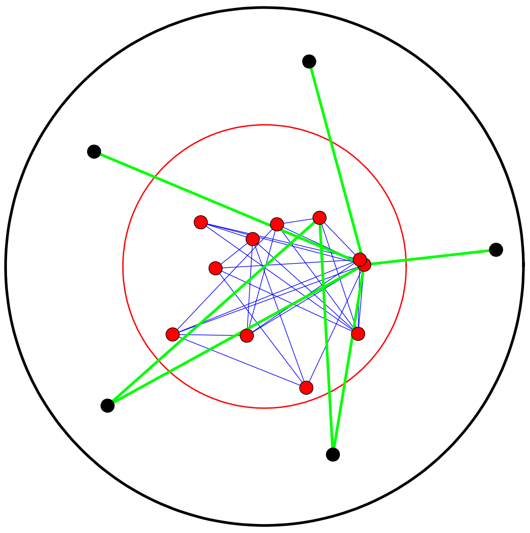


Figure 2.1: Example of an extended induced subgraph $\widetilde{G}$. The blue nodes represent an induced subgraph, the black nodes represent our additions to form an extended induced subgraph.

Concerning the degree of a vertex, following proposition holds:[1]

$$\text{for } v \in V_i' \subseteq V_{SNG} \ : \ d_{SNG}(v) = d_{G_i}(v)$$

Modeling an OSN Subgraph

Since the dawn of the Internet, there has been extensive theoretical work on modeling the structure of social networks [80]. In the following we discuss possible models to generate OSN subgraphs. The various proposed models focus on replicating recurring patterns and structures of social networks observed in real-world. Recent work on social networks within mathematics hereby focused on three distinctive features of network structure: the small average path length (APL) or *small world* effect [135], the high *clustering* effect, and the property of having a *scale free degree distribution* [18]. Within network simulations and modeling there are basically three main classes of paradigms. The first one is the classic random graph or *Poisson* random graph [64, 152], which has been well studied in mathematics [29, 111]. The random graph model is still used in many different fields of research and serves as a benchmark for modeling studies. However, the properties of these classic random graphs are not consistent with the properties observed in real-world online social networks. The second class, motivated by the small world effect, are small-world models. The idea of shortcuts for small-world models was proposed by Watts and Strogatz [189], followed by other researchers in different research areas [8, 144]. These models account for the so called *six degrees of separation* phenomenon. Finally there are scale-free models [17] which are basically networks with a given degree distribution, more

[1] V_i' is the set of vertices from the induced subgraph G_i', whereas G_i is the extended induced subgraph where G_i' is the underlying induced subgraph

precisely with a power law distribution. The discovery of the power-law degree distribution has led to the construction of various scale-free models, which try to provide a universal theory of network evolution and the realization of skewed degree distributions. For an overview of various analytical models, we further refer the interested reader to [6].

2.3 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 [70]. Facebook's competitors responded with the launch of an open standard for third-party access to social networking data called "OpenSocial" in November 2007 [87,92]. 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. provides an overview of existing social networking APIs [119].

2.3.1 Facebook Platform

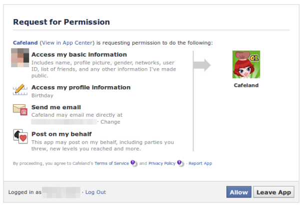
The Facebook Platform enables third parties to offer custom applications that extend Facebook's core functionality and integrate deeply into their website. Apps on Facebook are loaded inside Facebook through a "Canvas Page", which represents an *HTML iframe*. Facebook acts as a proxy for displaying the 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[2]. Before users decide to install a specific app, they need to authorize it. Facebook uses OAuth 2.0 [91] for authentication and authorization of third-party applications. Once users 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 [68], which applications may request in any combination. In total, there are 67 possible application permissions at the time of writing 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

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

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.

Over the course of the last five years, Facebook has also emerged as an identity provider for third-party websites. In addition to traditional canvas apps, websites leverage Facebook as an identity provider and to enhance their social experiences through JavaScript plugins. Finally, Facebook offers third-party access to mobile platforms through dedicated Android and iOS software development kits.

2.3.2 Misconceptions of social applications


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

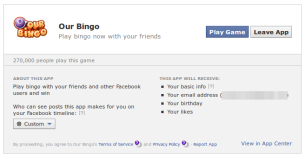
Send me email
Cafeland may email me directly at · Change

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 · Log Out

(a) Unified Auth Dialog, April 2010


Our Bingo
Play bingo now with your friends

270,000 people play this game

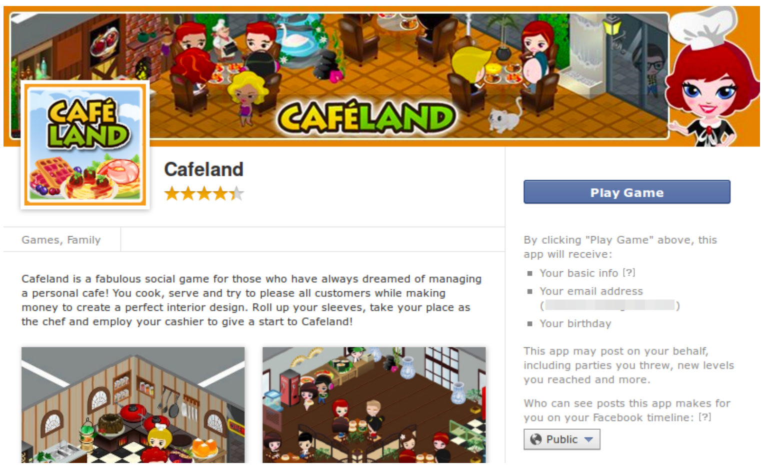
ABOUT THIS APP
Play bingo with your friends and other Facebook users and win

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

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

(c) App Center Auth Dialog, May 2012

Figure 2.2: Adjustments to Facebook's application authorization dialog over time.

Before users decide to add an application, an authorization dialog with requested permissions is displayed. Figure 2.2 shows Facebook's current alternatives to display requested permissions. In response to complaints from the privacy commissioner of Canada, Facebook introduced a unified permissions dialog in April 2010, of which Figure 2.2(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 2.2(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 2.2(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. A TechCrunch article [176] explains further design elements that Facebook uses to gradually conceal the requested permissions of its third-party applications. Hull et al. [106] claims 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 2.2, suggests that Facebook currently invests little in making their third-party application system more transparent. Previous research indicates that there are a number of common misconceptions regarding how social networking applications work. Besmer and Lipford [23] 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. [118] 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.

2.3.3 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 app badge was promised to applications that are secure and demonstrated commitment to compliance with the Facebook platform policies. A 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 [54]. 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) [66]. In May 2012 Facebook finally introduced the App Center[3], which showcases high-quality Facebook applications. At the time of writing a relatively small number of App center applications is in stark contrast to the overall number of applications. According to Facebook [157, p. 87], as of March 2012, more than nine million apps and websites were connecting to their Platform services, in comparison with a few hundred App Center applications.

[3]Facebook App Center `http://www.facebook.com/appcenter`

CHAPTER 3

State of the Art

This chapter provides an overview of research regarding privacy and security issues in online social networks and is divided into three main Sections. First we discuss privacy-related threats in Section 3.1, followed by security-related threats in Section 3.2. Finally, we discuss possible mitigation strategies for OSN privacy and security threats in Section 3.3.

A number of researchers previously surveyed existing research in the area of OSN privacy and security. One of the first attempts to categorize OSN security and privacy threats was published by Hogben et al. [98] in 2007 as a position paper. Their initial work provides an introduction to OSN security and privacy issues and also discusses possible protection strategies. The authors divided OSN security and privacy threats into four broad categories: privacy-threats, OSN variants of traditional threats, identity-, and social-threats. A number of their outlined potential threats have been confirmed by additional research in the meantime. Another overview of existing research was provided Bonneau, who maintained a online bibliography on security and privacy in social networks until late 2010 [30]. A number of publication furthermore attempted to survey this field of research and they are of limited breadth and are outdated as well. For example, Gao et al. [83] provide an overview of security issues in online social networks and Zhang et al. [198] discuss both security and privacy issues. OSNs are a dynamic field of research and this Chapter provides an updated and comprehensive introduction to OSN privacy- and security-related threats. In the following we mainly focus on technical weaknesses of OSNs and also discuss possible mitigation strategies. We point readers interested in general research on OSNs to two online bibliographies. Boyd [36] maintains a bibliography on OSN research, which contains over 500 publications. Wilson et al. [192] conducted a survey on OSN research in the social sciences, the authors also maintain a online bibliography[1].

[1]`http://facebookinthesocialsciences.org`

3.1 Privacy

The great majority of people use their real world identity for their online social networking profiles. OSNs provider furthermore encourage their users to provide and share a number of sensitive personal informations through their services. This pool of personal information has serious implications for the privacy of social networking users.

Acquisti and Gross [2, 88] were amongst the first researchers to raise awareness for the privacy implications of online social networks and the information disclosure behavior of its users. They analyzed the online behavior of more than 4,000 Carnegie Mellon University students and their information disclosure habits on Facebook. Acquisti and Gross found that the great majority of students disclosed personal information, such as their birthday and home address, publicly online. In the following, research by Jones and Soltren [112] showed that large datasets can be harvested from Facebook in an automated fashion. Their study was the first to address the privacy issues of Facebook from both a technical and user awareness standpoint. Stutzman [171] conducted a quantitative analysis of information disclosure in online social networks and found that a disconnect between the value of traditional identity information and online social networks exists. These initial publications created awareness for the impact of online social networks on information privacy and security. All four surveys had been conducted on Facebook, which at this point of time, was limited to college students in the United States. While these early information disclosure surveys all found that the majority of Facebook users made sensitive information available to third-parties, these findings are to some degree explicable by the fact that Facebook used to be exclusively for American college students until late 2006 [69]. A recent publication by Stutzman et al. [172] analyzed the disclosure patterns of Carnegie Mellon students over the course of six years. As outline in Figure 3.1 did the amount of publicly shared personal information decrease after Facebook became available to everyone with a valid email address.

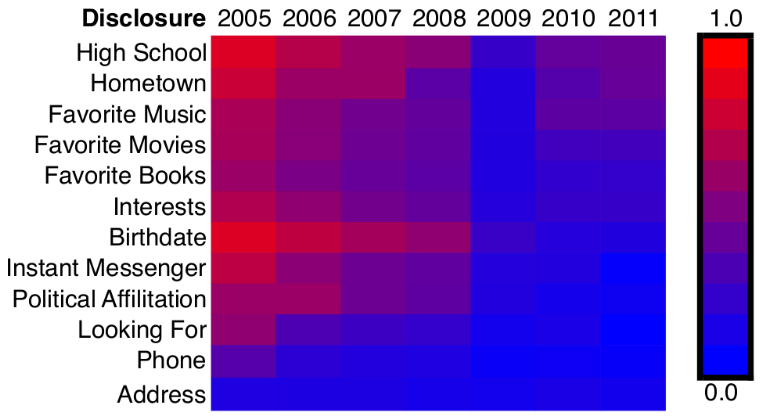


Figure 3.1: Heatmap of disclosure patterns in the CMU Yearly Snapshot Dataset [172].

Research in the social sciences became interested to discover the motives of people to join OSNs and to better understand self-presentation on OSNs. Livingstone [129] conducted research to understand the motives of youth to join online social networks, including OSNs such as Facebook and MySpace, on basis of interviews with teenagers. She found that younger teenagers prefer different OSNs as compared with older teenagers and that the graded perception of friends does not meet the binary notion of friends in online social networks. The initial studies of Acquisti and Gross [2, 88] have been further expanded by Tufekci [184] who conducted a survey based on 704 college students to understand how they protect their personal information online and how they present themselves in OSNs. The author found that online social networking was already a widespread phenomenon amongst college students in 2008 and that students mainly used their real name on Facebook, while relying on nicknames for their MySpace accounts. Strater and Lipford [169] surveyed the strategies used by Facebook users to protect their privacy online. They found that Facebook users often set their privacy settings upon account creation and then adapt their privacy settings once problems occur. Strater and Lipford's early survey on privacy settings on Facebook also argues for improved usability of privacy settings and stricter default settings. Another survey by Schrammel et al. [156] compared four different OSN communities to better understand the disclosure behavior and usage patterns of these sites. The four communities included the most popular professional and personal communities for both English-speaking and German-speaking countries in 2009.

3.1.1 OSNs Platform Design

The actual design and implementation of social networking services has an important impact on user privacy. OSN providers are hereby confronted with the trade-off of enabling convenient information exchange versus protecting the privacy of their users.

Bonneau et al. [32] outlined a number of possible design weaknesses of Facebook which might be misused to extract personal information from OSNs. Users might, might fore example, expose their credentials via cross-site phishing attacks because they are trained to provide their credentials for application authentication. Another weakness the authors found were special FQL queries that expose a number of personal information. The authors simulated the implications of their findings on basis of Stanford's Facebook network. Like other web services, OSNs rely on third-party products for tracking users and for monetizing their services with advertisements. Krishnamurthy and Wills analyzed the use of third-party advertising and tracking products across 11 different online social networks and found that OSNs commonly leak personally identifiable information to third-parties via HTTP parameters, referers, and cookies [123]. Krishnamurthy and Will [124] furthermore analyzed privacy leaks in mobile social networks (mOSNs) and found that in addition to leaking personally identifiable information to third-parties, these mobile applications also leak unique device identifiers and user locations to third parties.

Public information

In order to advertise content within online social networks, OSN providers might decide to make certain content available publicly. Bonneau et al. [31] showed that the social graph and underlying social network metrics can be inferred from public listings of a user's friends in search engines. Zheleva and Geetor [199] furthermore showed that private information can be inferred from public information e.g. based on group membership. The authors studied four different web services, the photo-sharing website Flickr, the social network Facebook, an online social network for dogs called Dogster, and the social bookmarking system BibSonomy. In order to analyze Facebook, they relied upon the data from Lewis et al. [126] and used favorite books, movies, and music as group memberships. Their results showed that it is, amongst other things, feasible to infer the gender of Facebook users based on their friendships. At the time of writing, Facebook limits the amount of personal information accessible to search engines. An initial feature of Facebook were networks, which per default enabled more permissive sharing of personal information for people belonging to the same network. This feature was especially problematic with open networks, such as regional networks, which anyone could join. Krishnamurthy and Wills [122] analyzed the privacy settings of users profiles within regional networks of Facebook. The authors found a negative correlation between permissive privacy settings and regional network size. Another important design feature of OSNs is allowing users to browse through the list of friends of their contacts. Korovola et al. investigated into attacks on link privacy whereas an attacker tries to gather knowledge of a significant fraction of links in the network [120]. The authors concluded that it is feasible to enumerate a significant faction of a ONS's underlying graph by subverting user accounts, and suggested that OSNs use lookahead values of 1 or 2 for their user interfaces. Currently, all major OSNs allow a maximal lookahead of 1 or 2 for their users. Irani et al. [108] finally investigated into publicly available OSN information and how this information can be cross linked to create so called *social footprints*.

3.1.2 De-anonymization and Identity

Because people provide their real-world identity on OSNs, users might be de-anonymized in real life based on their social networking data. Attackers in addition try to steal personal information from OSNs for identity theft.

Wondracek et al. [193] evaluated a practical attack to de-anonymize visitors of websites based on their previously visited social networking profiles. Their attack relied on first crawling publicly accessible social networking groups and the corresponding unique identifiers of all group members. Group membership was then used to optimize the number of queries to exactly identify social networking users via browser history stealing. While their attack was successful against users of XING, current web browsers protect against history stealing attacks and OSNs began to limit the amount of publicly available information. Acquisti et al. [4] showed that off-the-shelve face recognition algorithms are able to link snapshots of strangers with their online social networking identity. In worst case the personal information available on online social networks can lead to identity theft. For example, Acquisti and Gross [3] found that information about an individual's current location and date of birth can be exploited to predict her Social Security number (SSN).

3.1.3 Anonymized Datasets

With the growing importance of OSNs for research, the community started to make collected social networking datasets available online. While researchers usually exclude personal information from these datasets, it has been shown that releasing this kind of data poses non-trivial challenges.

Backstrom et al. [12] outlined a number of passive and active attacks to re-identify social networking users within anonymized datasets. The authors discuss formal methods to ensure user anonymity in datasets and argued that there is a lack of computational methods to automatically anonymize data from online social networks. A first survey on anonymization of social networking data was done by Zhou et al. [200]. The survey discusses social networking specific anonymization techniques on basis of two classes: clustering-based approaches and graph modification approaches. Narayanan and Shmatikov [139] developed a re-identification algorithm targeting anonymized social-network graphs which relies on network structures. The authors then demonstrated that verifiable members of both Flickr and Twitter can be recognized in a completely anonymous Twitter graph. De-anonymization attacks on social networking datasets are successful because in comparison with traditional relational datasets, the structural information of social networking graphs can be used to re-identify individuals without the requirement of personal information. Research acknowledged this problem and e.g. Zou et al. [202] propose the use of k-automorphism to protect against multiple structural attacks and they also developed an algorithm (called KM) that ensures k-automorphism of social networking data. Another early contribution which tackles this research problem was done by Campan and Truta [45] who also relied on the k-anonymity model to design a greedy algorithm to anonymize structural and personal information in social networking datasets.

3.2 Security

In this section we describe OSN security weaknesses and potential attacks based on OSNs. It should however be noted that privacy leaks might also lead to further attacks on the security of OSN users. If an attacker e.g. is able to extract personal information using a privacy leak she could misuse this information for further attacks, such as automated social engineering.

3.2.1 Provider Security

Security of social networking services ultimately depend on the software and service architecture deployed by OSNs providers, which we discuss in the following.

Communication security

OSNs are designed as centralized Internet services and communication security of these services is a necessity. Unencrypted network communication between users and social network providers implies that that the entire communication content (including personal messages, status updates, pictures, etc.) is vulnerable to eavesdropping. Even more important user sessions can be easily hijacked because attackers can extract authentication cookies from unprotected network links and misuse their accounts. Transport Layer Security (TLS) is the state-of-the-art cryptographic protocol to protect Internet services from eavesdropping and man-in-the-middle attacks. Table 3.1 shows the most popular OSNs, at the time of writing, and their support for TLS. Initially none of the major OSNs fully supported TLS with the exception of Google+ which was launched

	Transport Layer Security (TLS)	
OSN	2008	2013
Facebook	Login only	Opt-In/Default
Qzone	No	No
Google+	-	Default
LinkedIn	Login only	Opt-In
Vkontakte	No	Login only

Table 3.1: Top five online social networks and their support for TLS as of April 2013.

in 2011. As reported by Jones and Soltren [112], in 2005 even Facebook login credentials were exchanged in plain text. The fact that OSNs providers initially refrained from offering their services over a secure network channel can be attributed to performance and cost reasons (compare with [94]). It can be argued that the release of tools like Firesheep [43] or Faceniff [149] and our findings (see Chapter 6) sped up the implementation of full TLS support for OSNs. As of March 2013 a number of OSNs support full TLS communication as an optional feature of their platforms. Facebook introduced optional TLS support in 2011 [72], followed by LinkedIn in 2012 [127]. In November 2012, Facebook announced that they will start to roll out full TLS support per default [73] to all their users. The current partial TLS support of OSNs enables more sophisticated attacks on the communication security of user sessions. An example for such sophisticated attacks, is the *sslstrip* attack by Marlinspike [133]. In case whereas OSNs support unencrypted communication for backwards compatibility, *sslstrip* attacks can be used as a man-in-the-middle attack for hijacking user sessions.

Implementation Issues

Like any other web based service, OSNs are prone for implementation bugs that impact the security of their systems. A common implementation error that was misused by attackers of OSNs are cross-site scripting (XSS) bugs. Attackers misused these implementation bugs amongst other things to leverage XSS worms. Examples include *Samy* on MySpace [197], which infected more than 1 million profiles in less than 20 hours, and the *Bom Sabado* XSS worm targeting Google's Orkut in 2010 [195]. Except from XSS bugs, examples for high-impact attacks include the leak of studiVZ user-data through SQL injection attacks in 2007 [95] and the leak of 6.5 million unique LinkedIn passwords in 2012 [177]. A number of security bugs have been discovered on the popular Facebook platform as well, including bugs that allowed to reset passwords of arbitrary user accounts [161]. In June 2011 Facebook launched a bug bounty program to reward security researchers for discovered security bugs [51]. As of April 2013, 185 independent security researchers reported security bugs that eventually got acknowledged and fixed by Facebook[2]. Except from misusing implementation bugs, attackers recently try to trick users into performing certain social networking actions by using *clickjacking* attacks. Attackers hereby

[2]`https://www.facebook.com/whitehat/thanks/`

exploit recommendation buttons for web site developers through *clickjacking* attacks. Hereby, attacker place an invisible iframes over seemingly harmless web links and once users click these links they promote malicious content. In 2010 a number of these attacks targeted the Facebook *Like Button* and these campaigns were therefore dubbed as *likejacking* attacks [153]. Rydstedt et al. [155] found that in 2010 all of the top 500 websites were vulnerable to clickjacking attacks and they proposed possible countermeasures. Balduzzi et al. [14] furthermore developed a new detection technique, called *ClickIDS*, to automatically detect clickjacking attacks.

Platform Integrity

OSN providers rely on a number of strategies to ensure the integrity of their platforms. Stein et al. [167] describe the Facebook *Immune System* and outline the countermeasures Facebook uses to protect their system. According to the authors do Facebook's protection mechanisms rely on machine learning algorithms to keep pace with the changing nature of malicious campaigns. The authors identified four major threats to Facebook's platform integrity: compromised accounts, fake account, creepers, and spam. With exception of creepers, which are real users creating problems for other users, Facebook's security threats origin from automated bots. OSNs rely in general on two protection strategies to protect against malicious bots. First they make use of CAPTCHAs [187] to tell humans and computers apart, and second they apply per account rate limits to prevent misuse of certain platform features (e.g. number of friend requests per day). While less complex CAPTCHAs can be broken [136], attackers nowadays relay complex challenges to be solved by humans. A number of companies offer remote CAPTCHA solving services for as low as 1 USD per 1,000 solved challenges [137]. Except from paid services, attackers often use malware to solve complex anti-crawling mechanisms. An example includes malware that disguises itself as a striptease-game and requests users to solve CAPTCHAs in exchange for pornographic pictures [183]. Koobface, a malware family targeting OSNs users specifically, deceives users to solve challenges under the pretense of system shutdowns [180]. While, CAPTCHAs and rate limits attempt to contain the impact of malicious bots, OSNs providers require additional mechanisms to prevent the compromise of user accounts. Facebook, for example, used to rely upon IP features and the resulting geodistance, which attackers subverted by using proxies and botnets to fake according geolocations [167]. Facebook pioneered with an additional account verification mechanism, called *social authentication*. Social authentication relies upon social knowledge, such as the correct identification of a user's friends on photos. This novel verification measure promises additional protection and Kim et al. [117] found a number of potential shortcomings in this authentication scheme. The authors argue that insiders might easily subvert Facebook's social authentication schema and that face-recognition algorithms pose a growing risk. Finally, Kim et al. proposed a number of methods to strengthen Facebook's social authentication method. Polakis et al. [148] furthermore demonstrated that showed that the theoretical risks of Kim et al. [117] can be exploited in practice.

3.2.2 Unwanted and Malicious Content

It is believed that the vast majority of emails sent today are unsolicited bulk messages (spam), accounting for more than 70% of all emails sent in 2012 [115]. Empirical studies showed that while the conversion rate of spam is quite low, it is apparently still sufficient for spammers to make money [114]. Phishing is a special type of unsolicited bulk messages, which can be seen as the marriage between social engineering and spam. Attackers try to fool victims into entering their login credentials into malicious websites that mimic a real website (e.g., a bank). As phishing uses the same attack vector (email) and infrastructure as spamming (phishing websites are hosted on fast flux networks [99]), research in this area is closely related to spam and botnet research. Current research focuses on preventing spam delivery and how botnets are used for sending spam (e.g., the Storm [121] or Szizbi [168] botnets) to defeat IP blacklisting from email service providers. In practice, a combination of various techniques is used to minimize spam: sender-based systems such as SPF [194], IP blacklisting such as the Spamhouse blocklist[3], and content matching systems such as SpamAssassin[4].

Online social networks might change the way large-scale spam attacks are carried out. On one hand, are attackers using online social networks to deliver *social spam*, and on the other hand can data from OSNs be misused to increase the authenticity of traditional spam messages (see Subsection 3.2.3). Recent reports by Kaspersky [115] and Cicso [50] show that the global spam volume is declining and suggest that attackers already began to focus on more targeted spam attacks.

Social spam

We define social spam as unwanted content delivered via social networking platforms. Zinman and Donath [201] raised awareness for social spam and proposed to detect possible spam profiles based on sociability and promotion ratings. Hereby, profiles with low sociability and high promotion scores would be more likely to be spam profiles. Heymann et al. [97] surveyed existing anti-spam solutions and outlined how email and website strategies might be applied to OSNs. Webb et al. [190] proposed to use social honeypots to detect and analyze social spam campaigns. The authors created 51 MySpace profiles and collected information on friend requests they received via their honeypot accounts. Within four month, their honeypot profiles received 1,570 friend requests, which either attempted to lure victims to specific websites or to infiltrate their circle of friends. In the following, the authors extended their social honeypot approach and developed classifiers based on machine learning to identify previously unknown spammers [125]. Stringhini et al. [170] set-up 900 honeypot accounts for three major ONSs (Facebook, MySpace, Twitter) to analyze friend requests and received messages. In the following, the authors identified spam campaigns and supported Twitter in taking down a number of spam profiles. Gao et al. [84] analyzed a dataset of over 187 million Facebook wall posts, which they collected in 2009. Their system detected around 200,000 malicious wallposts, which corresponded to 0.1 percent of their sampled wall posts. Their results suggest that more than 70 percent of malicious posts advertised phishing websites. Furthermore, more than 97 percent of spam-sending Facebook accounts were classified as compromised user accounts rather than sybil accounts. Abu-Nimeh et al. [1] finally conducted another survey of malicious spam posts on Facebook and analyzed around half a million posts collected with the help of a Facebook

[3]Spamhaus Project `https://www.spamhaus.org/`

[4]SpamAssassin `http://spamassassin.org/`

application, called *Defensio*. The authors relied on a number of third-party services to classify links in their post sample. Their findings showed that 9 percent of the analyzed posts were social spam and about 0.3 percent of posts linked to malware.

Socware

The term *socware* was introduced by Rahman et al. [150] to describe criminal and parasitic behavior in OSNs, including anything that annoys, hurts, or makes money off of the user. In order to detect socware, Rahman et al. [150] developed a Facebook application called *MyPageKeeper*. Their applications was installed by more than 12,000 Facebook users, and the authors evaluated their approach on million posts collected in the course of four months. They found that 49% of our users were exposed to at least one socware within four months and that their classification approach based on social context significantly outperforms traditional website blacklists. Finally the authors observed and described a new type of parasitic behavior, called *Like-as-a-Service* which describes campaigns to artificially boost the number of likes of specific OSN content. Thomas and Nicol [180] analyzed Koobface a malware targeting OSNs. The Koobface botnet exploits social network users by: leveraging zombies to generate accounts, befriend victims, and to send spam. In order to get insights into the Koobface Botnet they first reverse engineered the Koobface binary and then emulated Koobface zombies. Despite the number of defenses put in place by OSN providers, their results found that domain blacklisting remains ineffective at identifying malicious URLs. Their results showed that blacklisting of malicious links took on average 4 days, while 81% of users visited Koobface URLs within 2 days.

3.2.3 Automated Social Engineering

While social engineering traditionally relies on information collected through dumpster diving or phone calls, OSNs contain a plethora of personal information which could be misused as an initial source for social engineering attacks. Because information harvested from OSNs can be easily processed, we were among the first researchers to argue that OSNs enable automated social engineering (ASE) attacks [100]. Reverse social engineering (RSE) describes a particular social engineering technique whereas an attacker lures targets into initializing the conversion. Irani et al. [107] argue that ONSs enable RSE attacks and describes three potential attack vectors. In the following, the authors evaluated their proposed attack vectors on three different OSNs: recommendation-based RSE on Facebook, demographic-based RSE on Badoo, as well as visitor tracking-based RSE on Friendster. Their results show that RSE attacks are feasible in practice and can be automated by exploiting the features of current online social networks. In comparison with social spam, attackers can send traditional email messages to delivery spam, because users provide their email addresses as part of their profiles. Hence, if spam is delivered via traditional email instead of OSN platforms, these malicious messages cannot be detected by the OSNs providers. Balduzzi et al. [15], for example, showed that OSNs can be misused for automated user profiling. The authors found that OSNs can be misused to validate large sets of email addresses and to collect additional personal information corresponding to these sets.

Social-phishing

Phishing is a common threat on the Internet where an attacker tries to lure victims into entering sensitive information like passwords or credit card numbers into a faked website under the control of the attacker. It has been shown that social phishing [110], which includes some kind of “social” information specific to the victim, can be extremely effective compared to regular phishing. Jagatic et al. [110] found that once phishing mails impersonated a target’s friend the success rate of their phishing campaign increased from 16% to 72%. The social graph is therefore not only of value for the social network operator, but for attackers as well. Especially if it contains additional information like a valid email address or recent communication between the victim and the impersonated friend. With automated data extraction from social networks, a vast amount of further usable data becomes available to spammers. Prior conversations within the social network like private messages, comments or wall posts could be used to deduce the language normally used for message exchange between the victim and the spam target. For example, a phishing target might find it very suspicious if someone sends a message in English if they normally communicate in French.

Context-aware spam

Context-aware spam misuses personal information extracted from OSNs to increase the authenticity of traditional spam messages. Brown et al. [38] identified three context-aware spam attacks which might be misused: relationship-based attacks, unshared-attribute attacks, as well as shared-attribute attacks. Relationship-based attacks solely exploit relationship information, this attack thus represents the spam equivalent of social phishing. The two remaining attacks exploit additional information from social networks, which could either be shared or unshared amongst the spam target and the spoofed friend. An example for an unshared attack are birthday cards that seem to origin from the target’s friend. Finally, attackers might exploit shared attributes for context-aware spam, such as photos where both the spam target and her spoofed friend are tagged.

Fake profiles

At the time of writing the only requirement for the creation of a social networking account is a valid email address, fake accounts can therefore be relatively easy created by attackers. A study by Sophos in 2007 showed, based on randomly chosen Facebook users, that around 41% of social networking users accepted friendship requests from a fake profile [163] they set up. Ryan and Mauch [154] further showed that fake profiles can be misused to infiltrate social networks and gain access to sensitive information in the military and information security community. Bilge et al. [25] outlined two sophisticated fake profile attacks to infiltrate the trusted circles of social networking users. First, with profile cloning attacks, an attacker clones an existing user profile and attempts to “reinvite” her friends. Second, with cross-profile cloning attacks, an attacker creates a cloned profile on a online social network where the target user does not have a profile yet and contacts her target’s friends. For example, if a user has a Facebook account but no LinkedIn account, an attacker clones the Facebook profile of the target to LinkedIn and contacts the target's Facebook friends who are also on LinkedIn. Bilge et al. showed that their attacks can be fully automated and are feasible in practice. If an attacker is able to create fake accounts on a large scale, sybil attacks on ONSs become possible. OSN provider therefore employ a number of protection mechanisms to limit the creation of large amounts of fake accounts (see 3.2.1). Boshmaf et al. [35] found that OSNs can be infiltrated on a large scale and evaluated how vulnerable OSNs are to a large-scale infiltration by socialbots: computer programs that control OSN accounts and mimic real users. The authors created a *Socialbot* Network (SbN): a group of adaptive socialbots that are orchestrated in a command-and-control fashion, on basis of Facebook. Hereby, the authors used 102 fake profiles to sent friendship requests to a random sample of 5,053 Facebook users. Overall friendship requests were accepted by 19.3% of all users. In the following the SbN tried to infiltrate the circle of friends of users who accepted their fake friend requests. Within 8 weeks SbN was able to further infiltrate social networking users and gain access to personal information. A recent survey by Alvisi et al. [7] provides an overview on sybil defenses based for online social networks and proposes community detection algorithms for sybil detection.

3.2.4 Digital Forensics

Digital forensics has received increasing attention in recent years as more and more crimes are committed exclusively or with the involvement of computers. Digital traces help courts and law enforcement agencies to capture valuable evidence for investigations. Existing research as well as applications in the area of digital forensics focus on filesystems [46], volatile memory [48, 93], databases [78], network traffic [53], and logfiles. The emergence of new online services like OSNs replaces the traditional means of digital storage, sharing, and communication [44]. While traditional forensic approaches rely on the seizure of the suspect's hardware (computer, smartphone, storage media) for analysis, the emergence of online services, social networks and novel online communication methods can render this approach useless: A tech-savvy adversary might use a computer without hard disk, communicating securely with the use of encryption and storing files distributed all over the world. This would leave no traces locally for the forensic examiners to work with as soon as the computer is shut down. Another problem is the worldwide distribution of the Internet with its multitude of jurisdictions: while a court might order a company that is located within the same country to reveal information about a suspect, across borders this request may not stand. With hundreds of millions of people sharing and communicating on social networks, they become more and more important for crime scene investigations. Traditional approaches to forensics on cloud computing and social network forensics are insufficient from an organizational as well as a technical point of view [26, 175]: the physical location of server systems is only known to the company, making seizure of hardware for examination in a forensic lab infeasible. Often, the social network operator in question cooperates with law enforcement but in an equal number of cases they do not. Delicts that might happen solely within social networks such as cyber- stalking, mobbing or grooming, in combination with cross-border jurisdictions make it very hard to gather evidence in a forensically sound manner. A number of examples for social network related crimes can be found in [179]. With the increasing number of users we expect the number of social network related investigations to increase heavily in the near future. The Electronic Frontier Foundation (EFF) compiled a report [62] on U.S. law-enforcement agencies' access to social networking data. Most social networking providers have dedicated services to cater for law-enforcement requests. For example, Facebook offers two types of data: basic subscriber information (“Neoselect“) and extended subscriber information (”Neoprint“) as well as an online request form for law enforcement agencies[5].

OSN forensic tools

Currently few digital forensic tools support the collection of evidence from OSNs. The collection techniques hereby focus on two evidence sources, namely: network dumps and forensic images of hard drives. Xplico [196] is an Internet traffic decoder which supports the retrieval of Facebook chats from network dumps. Standard packet analyzer often provide little insights into exchanged content because OSNs rely on their users web browser to render the actual content. Pyflag [53], on the other hand, is a modular network forensic framework built to analyze network dumps. Amongst other features the Pyflag framework can be used to rebuild web pages from packets, allowing examiner to view the web pages the suspect has seen even if it used AJAX or other dynamic techniques for representation. Commercial tools, such as IEF (Internet Evidence Finder)[6] support the reconstruction of social networking fragments from forensic images. Burszstein et al. [42] developed OWADE (Offline Windows Analyzer and Data Extractor) a project dedicated to cloud forensics which supports the recovery of browser history and credentials. Garfinkel [85] finally introduced bulk_extract, an open source tool, for the efficient extraction of forensic fragments. The tool extracts a number of fragments which are relevant for OSN forensics such as browser history and cookies.

3.3 Protection Strategies

In this section we outline protection strategies for OSN users. Hereby, we first discuss security extensions for OSNs, followed by possible usability improvements, and decentralized social networking.

3.3.1 OSN Extensions

Security extensions for OSNs aim to hide personal information from social network providers as well as from third parties without stopping users from sharing information. Guha et al. [89] proposed an OSN extension, called NOYB (None Of Your Business), which substitutes personal profile information with pseudorandom content. Lucas and Borisov [131] introduced another OSN extension called flyByNight, which 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. [132] proposed FaceCloak, whereas OSNs providers receive fake profile information and the real user data is stored encrypted on separate servers. In order

[5] `https://www.facebook.com/records`

[6] `http://www.magnetforensics.com/products/internet-evidence-finder/`

to restore real user information, FaceCloak users require a browser extension and their FaceCloak credentials. FaceCloak's approach is similar to NOYB with the exception of requiring additional servers for storing the actual encrypted data. Baden et al. [13] proposed Persona a privacy extension for OSNs by means of attribute-based encryption. The authors developed a proof-of-concept Facebook implementation bundled with an extension for the Mozilla Firefox web browser. The authors showed that their proposed approach is feasible and evaluated the computational costs of Persona. Tootoonchian et al. [181] proposed Lockr, a system that improves the privacy of centralized and decentralized online content sharing systems. Lockr enables OSNs users to define and enforce fine-grained access control on shared content. Hereby, Lockr implements an attestation verification mechanism, specifically a witness hiding proof of knowledge (WHPOK), which is a variant of zero-knowledge protocols. The authors outlined the feasibility of their approach on basis of an implementation for a centralized OSN as well as for a de-centralized filesharing application. Beato et al. [20] finally proposed "Scramble", a generic method to shield confidential information from social networking providers. Table 3.2 outlines the discussed OSN security extensions, their protection mechanisms, requirements, as well if they are still operational.

Extension	**Mechanism**	**Requirements**	**Operational**
NOYB [89]	Pseudorandom substitution	FF extension, dictionary server	No
flyByNight [131]	Public-key cryptography	OSN application	No
FaceCloak [132]	Fake information, encryption	FF extension, FaceCloak server	No
Persona [13]	Attribute based encryption	FF extension, OSN application	No
Lockr [181]	WHPOK attestation	FF extension, proxy server	No
Scramble [20]	Public-key cryptography	FF extension, tiny link server	Yes

Table 3.2: Overview of proposed OSN security extensions

With the exception of flyByNight all extensions require additional web browser extensions and all researchers initially developed extensions for Mozilla Firefox (FF). In addition to web browser extensions, all proposed OSN extensions require additional servers. Thus, if these extensions were deployed on a large-scale the required additional servers had to scale as well and potentially handle the data of millions of OSN users. Even more important is the maintenance of the proposed OSN extensions. As table 3.2 outlines, did all proof-of-concept implementations except Scramble cease to be operational. Both web browsers and OSN platforms are subject to swift implementation challenges and most authors arguably stopped any further development once their papers got published. The authors of Lockr, for example initially maintained a website for their project (http://lockr.org), at the time of writing the project domain is not owned by the Lockr authors anymore and hosts a compromised personal homepage. Except from practical OSN extensions, a number of researchers proposed generic schemes and architecture to improve the security of centralized OSNs. Anderson et al. [9] proposed an architecture for untrusted centralized services with link protection. Sun et al. [173] furthermore proposed a cryptographic scheme for privacy preserving sharing of information within OSNs. Their scheme attempts to mitigate threats posed by untrusted storage sites, supports dynamic revocation of group members, and enables search over shared data. The authors argue that while their scheme might not completely protect the privacy of traditional OSNs users (e.g. in Facebook users can be identified based on their shared pictures), their approach can enhance the privacy of anonymous OSNs. Feldman et al. [75] finally proposed Frientegrity an OSN extension for untrusted centralized OSNs. Their approach accounts for both confidentiality and integrity, based on a novel method for server equivocation detection, of OSN user data.

API protection

In addition to generic OSN security extensions, a number of researchers proposed extensions to improve the security of OSN third-party APIs. Felt and Evans [76] 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 however impracticable with state-of-the-art applications because the majority of applications require personal information to work. Singh et al. [162] 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. [63] 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 have been adopted by social network application developers.

3.3.2 Usability

Social network providers started to offer a number of fine-grained privacy and security controls, research however showed that users often do not understand existing privacy interfaces [128]. Unusable and complex user interfaces ultimately lead to weak privacy settings of OSN users. Lipford et al. [128] tested new interface prototypes that make use of an audience-oriented view of profile information. Their experiments showed that audience-oriented interfaces significantly improved the understanding of privacy management in ONSs. Danezis [57] proposed the automatic inference of social context to simplify privacy management. Social contexts are hereby inferred by automatically detecting highly cohesive social sub-groups (communities) based on a user's social graph. Fang and LeFevre [74] furthermore proposed the use of wizards to improve the privacy settings of users. Users are hereby requested to label a limited number of their friends and the privacy wizard then infers a privacy-preference model for all remaining friends. Their work found, similar to Danezis [57], that community structures might offer helpful cues for privacy preferences. Jones and O'Neil [113] investigated people's rationales when grouping their contacts for the purpose of controlling their privacy and assessed possible automated approaches. The authors showed that the SCAN algorithm yielded a 66,9% chance of correctly grouping a user's friends for privacy management. Squicciarini et al. [164] proposed collective privacy management based on game theory to tackle the problem of shared social networking content. The authors implemented their approach as a Facebook application and evaluated the performance of their proposed solution. Since late 2011 two of the proposed usability enhancement have been en-cooperated into popular OSNs. Facebook introduced the automatic grouping of friends with their so called "smart-list" feature [65]. Both Google+ and Facebook began to support audience views of user profiles, to enable users to review which profile content is visible to other users.

Application authorization dialogs

Third-party applications receive upon user authorization a number of personal information, the design of these authorization dialogs is therefore an important usability problem. Besmer et al. [22] 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. [188] 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 seem to conceal privacy-relevant information from users (see Figure 2.2).

3.3.3 Decentralized OSNs

Decentralized social networking services attempt to overcome a number of security and privacy issues by not relying on centralized untrusted services.

Buchberger and Datta [39] made a case for peer-to-peer social networking services and discussed the unique challenges and opportunities of such services. Because no centralized service is used to store data in peer-to-peer OSNs, users gain more control over their intellectual property and better protect their privacy. The authors argue however that research into efficient content update mechanisms would be required to realize peer-to-peer OSN services. PeerSoN [40] is a prototypical implementation of a peer-to-peer OSN system, which uses a two-tier system architecture. One tier is used for peer lookups with distributed hash tables (DHTs), while the second tier consists of the actual user peer-to-peer communication. Shakimov et al. [160] discuss three alternatives to centralized OSNs: cloud-based decentralization, desktop-based decentralization, and a hybrid of cloud- and desktop-based decentralization. The authors discuss the cost, and availability tradeoffs of their three different proposed alternatives. In the following Shakimov et al. [159] outlined Vis-à-Vis, a decentralized framework for social networking based on distributed Virtual Individual Server (VIS). They implemented their approach and deployed it in the cloud and their evaluation showed that Vis-à-Vis imposes little performance overhead as compared with centralized OSNs. Cutillo et al. [55, 56] analyzed common threats in OSNs and proposed a privacy-preserving peer-to-peer OSN application called "Safebook". The system architecture of Safebook consists of three tiers: a social networking layer, a P2P substrate implementing the application layer, and the Internet which represents the communication layer. PrPl [158], short for private-public, is another decentralized OSN infrastructure whereas each user relies on a Personal-Cloud butler service. These butler services can be run on home servers but also on paid cloud services and provide secure storage for personal files. Backes et al. [11] proposed a cryptographic framework for distributed social networking which aims to protect both user content as well as their social relationships. The authors formalized their approach and verified it with an automated theorem prover. Except from the growing number of proposed architectures and cryptographic schemes for decentralized OSNs, Diaspora[7] is the first widely deployed distributed social network. Diaspora itself consists of a network of independent, federated servers (pods). Users can run their own server or join an existing one. Bielenberg et al. [24] analyzed the growth of the Diaspora social network and found that most users rely on a few large Diaspora servers to host their data.

[7]`http://diaspora-project.org/`

CHAPTER 4

Study of Application Ecosystems

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. Currently, the user is expected to decide whether the requested permissions are in accordance with the application’s purpose on a technical layer. 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 [49, 188]. 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 [118]. 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.

In this chapter 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 implemented our framework on the basis of Facebook, which currently has the largest and most mature third-party application ecosystem. 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.

4.1 AppInspect Framework

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.

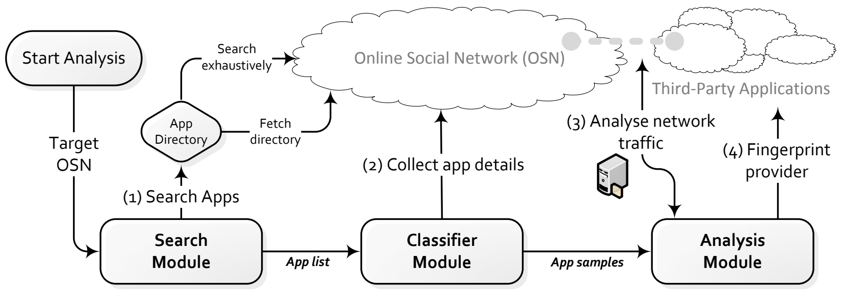


Figure 4.1: AppInspect, a framework for automated security and privacy analysis of social network ecosystems

Our proposed *AppInspect* framework enables fully automated security and privacy analysis of a target social networking app ecosystem. Figure 4.1 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) 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. Each of the three main modules encapsulates additional functionality into submodules. 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.

4.1.1 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 enumerable using a simple query and varying the *applicationId* parameter.

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. [47] 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, e.g., Google+. Google+'s game directory[1] 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.

[1]Google+ Games `https://plus.google.com/games/directory`

4.1.2 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
$\Longrightarrow$ 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
$\Longrightarrow$ 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.

4.1.3 Analysis module

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

Traffic collection. The traffic collection submodule provides a core functionality of the analysis module, namely the collection of all network traffic resulting from social networking apps. In order to collect network traffic, this submodule relies on an HTTP interception proxy, which also acts as a man-in-the-middle HTTPS proxy to include encrypted traffic for subsequent analysis as well.

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 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.trippleclick.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. In order to reveal the app's service provider, the submodule performs a reverse DNS lookup and determines the network hops between the AppInspect server and the app's domain. In case no reverse DNS entry exists, the last network hop might reveal the possible service provider. 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.

Vulnerability search. 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.

4.2 Methodology and Application 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.

4.2.1 Methodology

We chose to implement an instance of our *AppInspect* framework for the Facebook platform. Facebook serves is a good example due to its popularity and the plethora of available third-party applications. We set up a number of experimental accounts with bogus data and to be able to perform automated application evaluations without needing to process personal information. Once we finished our experiments we deactivated all Facebook test accounts. While Facebook offers dedicated whitehat accounts[2] for security researchers, these accounts cannot use any third-party applications. 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[3]. 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.

4.2.2 AppInspect Facebook 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. We include these two application directories to verify the completeness of results enumerated with the exhaustive search submodule.

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.

[2]Facebook whitehat accounts `https://www.facebook.com/whitehat/accounts/`

[3]Ghostery `http://www.ghostery.com/`

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. Furthermore, we ignore leaks to content delivery networks (CDNs) of Facebook (fbcdn.net) and application providers (e.g. zgncdn.com). Leaks to other CDNs such as Akamai or Amazon CloudFront are also less critical because they do not track their users across multiple domains using HTTP cookies. 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*[4] as well as for readily available Metasploit modules[5].

4.2.3 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 exhaustive search resulted in 234,597 applications and took around 5 days to complete. This first run helped us fine-tune our exhaustive search module. In June 2012, we ran the search module again, this time with character trigrams based on the English alphabet and also on integers from 0 - 9. The exhaustive search module enumerated *434,687* unique applications and took two weeks to complete. 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. In addition, we validated our enumerated sample against Socialbakers application statistics[6]. Our validation attempt showed that in addition to including all Socialbaker applications, our approach found a number of high-ranking applications that were missing in their sample.

We observed a great disparity in the monthly average usage (MAU) of the enumerated applications. Figure 4.2 illustrates our observation. While the great majority of applications had an

[4]Offensive Security Exploit DB `http://www.exploit-db.com/`

[5]Metasploit modules `http://www.metasploit.com/modules/`

[6]`http://socialbakers.com/facebook-applications/`

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.

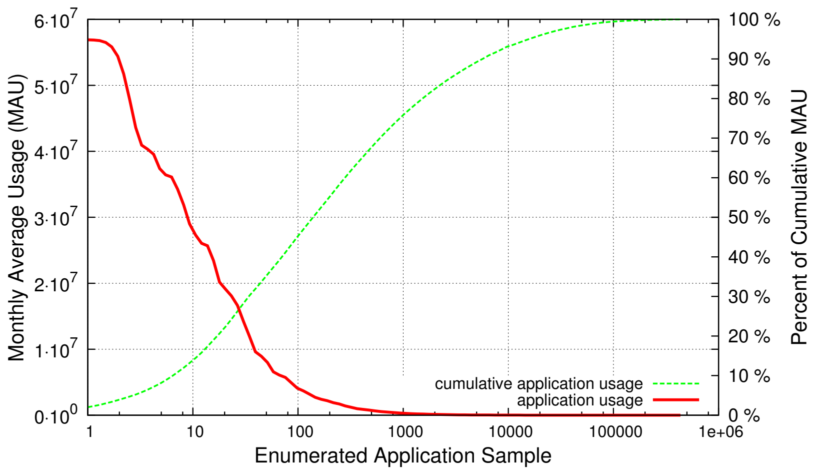


Figure 4.2: Active monthly users in our enumerated 434,687 applications sample.

We discarded all applications with fewer than 10,000 MAU from our subsequent analysis because of their comparably minor impact. In July 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 4.1 show the different application types observed in our sample.

Application Type	**Applications**	**Total %**
Authentication Dialog	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%*

Table 4.1: Classification of subsample with popular applications.

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. In addition to language detection, the API was used to translate non-English application names. The majority of applications were English (64.72%), followed by Spanish and German. In total, we observed 69 different languages.

4.3 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 are widely used and transfer personal information to application developers.

4.3.1 Requested personal information

Table 4.2 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.

	Characteristics		
Name	Type	Users	**URI**
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 4.2: Most common requested permissions by third-party applications (n=4,747)

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.

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 4.3 depicts the number of distinct permissions requested per application provider. The provider samples are sorted by monthly average usage ranging from 1 (most popular) to 1,646 (least popular). 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.

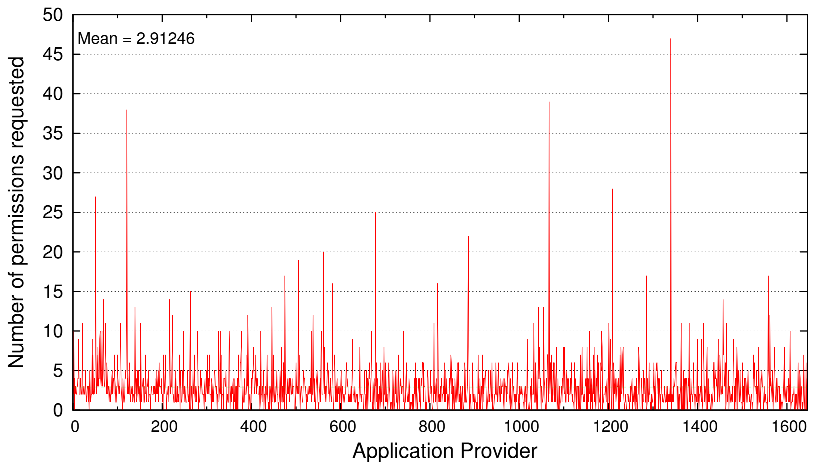


Figure 4.3: Number of requested permissions for 1,646 applications providers.

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, or daily horoscopes in a number of different languages. However, the majority of their applications did not require any personal information at all to function. 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 July 2012).

4.3.2 Hosting environment

Our analysis of reverse DNS queries and network hops showed that developers relied upon 604 distinct Internet hosting services. Table 4.3 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.

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 4.3: Most common Internet hosting services for Facebook third-party applications

Our analysis module probed the 604 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 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 4.4 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 outda-

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 4.4: Most common additional services on application hosts

ted 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 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 processes sensitive personal information such as user email addresses and birthdays.

4.3.3 Web trackers

Our web tracker submodule identified 139 distinct web trackers in the recorded application traffic. Table 4.5 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 that are used to create additional revenue for application developers. All web bugs presented in Table 4.5 potentially track their users across multiple websites based on HTTP cookies.

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 4.5: Common web trackers included in third-party applications

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.

4.3.4 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 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.

4.4 Discussion

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

4.4.1 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

[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 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 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 OAuth 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. As of May 2013, all application providers fixed our detected privacy shortcomings.

4.4.2 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 [16, 41] 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.

4.4.3 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 applications' 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. In addition to valid email addresses, third-party developers also collect information that enables sophisticated email based attacks. For example, social phishing attacks [110] 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 [38] might also misuse user birthdays or photos to increase the authenticity of unsolicited bulk messages. Forbes [77] 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. 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 or OAuth tokens via GET 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 follow the example of LinkedIn or iOS App Store and manually review the balance of app functionality and requested permissions.
- Social network providers should stress application developers to harden their hosting environments.

4.4.4 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 prototype currently does not detect personal information that is scrambled before being sent by third-party products. 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.

4.4.5 Dataset

Our dataset comprises an extensive social application dataset and provides a valuable basis for research on social networking apps. *AppInspect* is designed to steadily provide new insights into third-party application ecosystems, and we will, therefore, periodically refresh and expand our dataset. Our AppInspect dataset is available online at: `http://ai.nysos.net`.

CHAPTER 5

Social Snapshots

In this chapter, we introduce a novel method for data collection from social networks called *social snapshot*. Our approach is based on a hybrid system that uses an automated web browser in combination with an OSN third-party application.

Over the past years, Online Social Networks (OSNs) have become the largest and fastest growing websites on the Internet. OSNs, such as Facebook or LinkedIn, contain sensitive and personal data of hundreds of millions of people, and are integrated into millions of other websites. Research has acknowledged the importance of these websites and recently, a number of publications have focused on security issues that are associated with OSNs. In particular, a number of empirical studies on online social networks [25, 84, 105, 110, 193] highlight challenges to the security and privacy of social network users and their data. We found that these, and similar studies, heavily depend on datasets that are collected from the social networking websites themselves, often involving data that is harvested from user profiles. Furthermore, as social networks continue to replace traditional means of digital storage, sharing, and communication, collecting this type of data is also fundamental to the area of digital forensics. For example, data from OSNs have been used successfully by criminal investigators to find criminals and even confirm alibis in criminal cases [52, 178]. While traditional digital forensics is based on the analysis of file systems, captured network traffic or log files, new approaches for extracting data from social networks or cloud services are needed. Interestingly and contrary to our intuition, we found little academic research that aims at developing and enhancing techniques for collecting this type of data efficiently. Despite the growing importance of data from OSNs for research, current state of the art methods for data extraction seem to be mainly based on custom web crawlers. However, we found this naïve approach to have a number of shortcomings:

- High network traffic: The extraction of profile data via traditional web crawling can be regarded as costly with regard to the required network resources, as it typically incurs a large amount of HTTP traffic and causes a high number of individual network connections. Apart from inherent disadvantages, social networking websites may also choose to block network access for clients that cause high levels of traffic, thus preventing them from harvesting additional data.

- Additional or hidden data: Per definition, web crawlers can only collect data that is accessible on the target website. However, we found that social networks often publish interesting meta-information (e.g. content creation timestamps or numeric identifiers) in other data sources, for example via developer APIs.
- Maintainability: The structure and layout of websites tend to change unpredictably over time. Additionally, the increasing use of dynamic or interpreted content (for example, JavaScript) leads to high maintenance requirements for custom web crawlers.

In the following we show that our system can be used efficiently to gather *social snapshots*, OSN user datasets and are able to overcome the major problems of traditional web crawling techniques.

5.1 Social Snapshot Tool

In this section, we describe the design of our approach as well as the individual components of our digital forensic framework.

Our digital forensics application enables an investigator to snapshot a given online social network account including meta-information, a method we termed *social snapshot*. Meta-information such as exact timestamps are not available to the user via the user interface of the web application. A social snapshot represents the online social networking activity of a specific user such as circle of friends, exchanged messages, posted pictures etc. Due to the diversity of information available via OSNs we propose a twofold approach: an automated web-browser in combination with a custom third-party application. The social snapshot application is initialized with a user's credentials or authentication cookie. In the following, a custom third-party application is temporarily added to the target account. This application fetches the user's data, pictures, friend list, communication, and more. Information that is unavailable through the third-party application is finally gathered using traditional web-crawling techniques. By automating a standard web-browser and avoiding aggressive web-crawling we simulate the behavior of a human user, thus minimizing the risk of being blocked by OSN providers.

Figure 5.1 shows the core framework of our social snapshot application. **(1)** The social snapshot client is initialized by providing the target user's credentials or cookie. Our tool then starts the automated browser with the given authentication mechanism. **(2)** The automated browser adds our social snapshot application to the target user's profile and sends the shared API secret to our application server. **(3)** The social snapshot application responds with the target's contact list. **(4)** The automated web browser requests specific web pages of the user's profile and her contact list. **(5)** The received crawler data is parsed and stored. **(6)** While the automated browser requests specific web pages our social snapshot application gathers personal information via the OSN API. **(7)** Finally the social data collected via the third-party application is stored on the social snapshot application server.

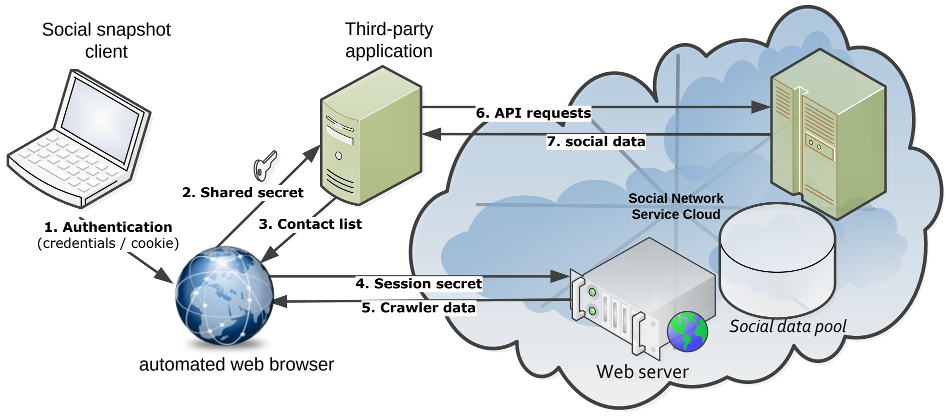


Figure 5.1: Collection of digital evidence through our social snapshot application.

5.1.1 Authentication

In order to get access to the complete content of a target's social network account, social snapshots depend on gathering the initial authentication token. In the following, we outline three digital forensic scenarios that explain how this initial gathering of the authentication token works and that are representative for real-world use cases.

Consent. This naïve approach requires consent from the person whose social networking profiles are analyzed. A person would provide the forensic investigator temporary access to her social networking account in order to create a snapshot. This would also be the preferred method for academic studies to conduct this research in an ethically correct way and to comply with data privacy laws. We used this method for the evaluation of our proposed application as further described in Section 5.4.

Hijack social networking sessions. Our social snapshot application provides a module to hijack established social networking sessions. An investigator would monitor the target's network connection for valid authentication tokens, for example unencrypted WiFi connections or LANs. Once the hijack module finds a valid authentication token, the social snapshot application spawns a separate session to snapshot the target user's account.

Extraction from forensic image. Finally, physical access to the target's personal computer could be used to extract valid authentication cookies from web-browsers. Stored authentication cookies can be automatically found searching a gathered hard drive image or live analysis techniques such as Forenscope [48].

5.1.2 Depth of Information Collection

Starting from a target profile, a number of subsequent elements become available for crawling such as the user's friends, uploaded photos and joined groups. With these elements, again, a number of subsequent elements can be accessed. For example, the single-view page of a photo can contain comments and likes of other users, who do not necessarily have to be direct friends of the owner of the photo. Additionally, users can be tagged in photos. These are all starting

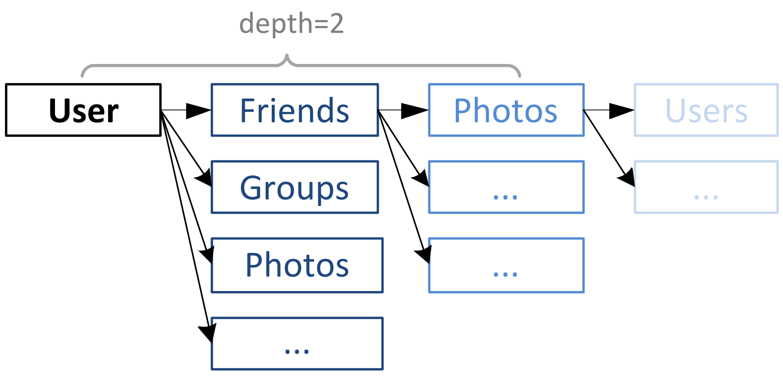


Figure 5.2: Example for elements fetched with social snapshot of depth=2

points for further crawling. The same applies for groups; A group gives access to the profiles of all group members, photos with users tagged, who are potentially not members of the group, and so forth. Consequently, a social snapshot of a single user does not only obtain the user's data and data of her friends, but its depth can reach a high value. Thus, the depth of the social snapshot is an essential configuration option which controls the social snapshot's extent. Figure 5.2 shows an example of a social snapshot with $depth = 2$. For a given user all of her friends are first fetched, followed by the friend's photos. The single path for photos of the friend's user illustrates the magnitude of available paths and thus data. Defining a specific social snapshot depth enables us to limit the amount of fetched data. The amount of data grows exponentially with social snapshot depth.

It is important to note that the relevance of data is not the same for different elements. For example, tagged users in a photo are most likely in a closer relationship to the owner of the photo than two users that joined the same group, just because of similar interests. Therefore, the social snapshot tool prioritizes element types that suggest higher data relevance and uses them as a starting point of each iteration. The prioritization is performed on the basis of predefined priority flags in the third-party application.

5.1.3 Modules

Our social snapshot application consists of a number of modules, which we describe in the following. The core modules are the automated web browser and our custom third-party application as outlined in Figure 5.1.

Social snapshot client. The social snapshot client module initializes the data gathering process with a given user's credentials or cookies. Once started, the client first authenticates itself against the target online social network. In the following, the client automatically adds our custom third-party application with the highest possible permissions to the target's account. Information that cannot be retrieved through our third-party application is crawled and parsed by the client. Once all information has been retrieved, the client removes the third-party application and logs out of the given social networking account. The interaction with the social network as well as web-crawling is performed by the Selenium framework [146], which we describe in the following. We implemented the social snapshot client in Java and the module offers a command line interface.

Automated web browser. The browser module is responsible for the basic interaction with the target online social network. We used the Selenium testing framework [146] to automate the Mozilla Firefox browser. Selenium comes with a command line server that receives Selenium commands. Therefore, we can use the framework to script the behavior of an average user using her Firefox web-browser to surf a social networking website. We had to overcome one initial obstacle though: cookie authentication with Selenium which was not supported out-of-the-box. We finally patched the original Java source code of the command line server to be able to correctly set HTTP cookies for the cookie authentication mode.

Third-party social snapshot application. Our OSN social snapshot application is a third-party application, which sole purpose consists of gathering all possible account data through the target OSN's API. The main design goal of our third-party OSN application is performance, thus multiple program threads are used to gather information as quickly as possible. The third-party application can be configured to prioritize specific account data and to download only a predefined set of account artifacts (social snapshot depth).

Hijack module. The hijack module is a network sniffer module that collects valid OSN HTTP authentication cookies from sources such as LAN or WiFi connections. We built our hijack module on the basis of Mike Perry's modified libpkt library [147], which works out of the box with LAN, unencrypted WiFi, and WEP encrypted WiFi connections. The hijack module offers a command line interface and is implemented in Python.

Digital image forensics. The digital image forensics module matches image files gathered from online social networks with their original source. The goal is to find the pristine image of a compressed picture extracted through our social snapshot application. All images are initially clustered according to their color histograms, rescaled and compressed to the target picture size, and finally matched with pattern recognition techniques. As social networks typically remove meta (EXIF) information of uploaded images this module is helpful in finding the source of collected pictures from OSNs and thus restore information such as the original image creation time, camera model etc.

Analysis module. The analysis module is a parser for the results gathered with the data collection modules of our application. It parses the crawled data as well as the information collected through the OSN's API. Furthermore, the analysis module fetches additional content such as photos that are openly available by knowing the URI from online social networks. Finally, it generates a report on the social snapshot data. The analysis module can be used to generate exact timelines of communication, metadata summaries, e.g. of pictures, a weighted graph from the network of friends, or their online communication.

5.2 Visualization

Our social snapshot tool enables forensic investigators to collect fast amounts of profile information for a given OSN account. Information visualization helps to quickly screen the collected information. In the following we thus describe possible information visualizations.

5.2.1 Basic Visualizations

Social Interconnection Graph. It is trivial to retrieve the list of friends from social networks. In most social networks this is public information, or can be easily collected [31] even without entering the social circle of the account under investigation. However, it is not trivial to cluster these friends, namely to find out who is connected with whom: is a specific contact part of the cluster of work-friends, or are they directly related?

In our approach we use a feature of the Facebook API which allows an application to query if two users are connected. This allows our software to cluster the friends of a user into different groups e.g., people from work, school, family and more, as the members of the groups are much more likely to know each other. The graph can be represented as an undirected graph $G =< V, E >$ where $V = v_1, v_2, ...v_n$ is the set of friends of a user, and $E = (v_x, v_y), ...$ is the set of edges that connects two nodes in case they are friends on the social network. Highly connected nodes have a high degree, representing well connected friends that know most of the suspect's friends as well.

Social Interaction Graph. For many investigations it is of importance to find out who communicated with whom. Various ways of communication are possible among users, like wall posts, direct messages, group communication or following public announcements. Communication can be represented as a directed graph $G =< V, E >$ where the nodes $V = v_1, v_2, ...v_n$ are all the friends while the edges $E = (v_x, v_y), ...$ are directed and the weight of (v_x, v_y) is incremented for every message sent from v_x to v_y. In this thesis, however, we do not distinct between the different forms: direct messages, wall posts, and tags are treated equally as these are the most direct form of communication. An investigator can easily identify the top communication partners on a first sight, and compare them with e.g., obtained phone records.

Complete Timeline. With social network users being online 24/7 by using mobile clients on smartphones, the timeline becomes of increasing importance. Not only activity of the user itself can be extracted, but also the activity of all his or her friends. Often the times of activity can be seen easily, if properly visualized. To make analysis feasible it is necessary to use respectively allow different data layers: activity of the user, the friends, group activities, reactions on events from friends, and so forth. It is also crucial that the timeline is zoomable, to visualize time ranges of importance - a single day can have easily more than 500 events for a given profile and all his or her friends.

Location Visualization. Geotagging and location applications are novel features with increasing usage that needs to be reflected in forensic examinations. With foursquare[1] and Facebook Places, just to name a few, the geolocation information stored in social networks is growing steadily. While our toolset is not yet available to extract geodata, we believe that this will become more and more of an issue. Digital cameras as well as smartphones automatically geotag pictures taken with the exact location. Up till now, most social networks remove metadata during the transformation for picture storage [21], but this might change in the future.

[1]`https://foursquare.com/`

5.2.2 Advanced Visualizations & Information Inference

While the features discussed so far are rather straightforward, we believe that the following list of advanced features and components should become standard tools in digital forensics:

Event tracking. For viral scammers and other malicious applications that use the social network for propagation it might be of interest who or what started such a series of events. Tracking such events is not straightforward, but with a collection of social network footprints of various users these events can be easily dissected. This would allow insight into dissemination characteristics and propagation tactics of scammers, as well as advanced analytical capabilities.

Timeline matching. In highly centralized systems such as online social networks, an investigator has the benefit of consistent timestamps as they are provided by the social network. The operators often run their won NTP infrastructure, and keep the clocks consistent across thousands of servers. This can then be used to match timelines of different profiles, and eventually create an exact timeline for a complete cluster of friends or even bigger. While this has been proposed recently for the NTFS file system [59], we believe that this will be of importance for social networks and cloud computing as well.

Differential Snapshots. Once a forensic image of a user profile is collected, at a later point in time the image might look completely different. Therefore, the forensic framework must provide the functionality to not only visualize the social network data of a user, but also the functionality to visualize differences with previous images of the same user.

5.3 Methodology

In this section, we describe the evaluation of our social snapshot application. Our generic social snapshot approach is applicable to the majority of today's social networking services. The sole requirement for target social networks is the availability of a developer API or the adaption of our automated browser.

For a forensic tool there are some special requirements:

- Ability to *reproduce* results,
- Create a *complete* snapshot of the account.

To make digital evidence sufficiently reliable for court it is helpful if the process of gathering the evidence can be reproduced with identical results. In dynamic Web-based applications this is not possible because data is continuously added (eg. posts by friends) or removed (eg. friends-of-friends deciding to unshare data by modifying their privacy settings). It is, however, possible to have two or more independent investigators make snapshots at a similar time. While not all artifacts will be identical one can easily compare the sets of artifacts retrieved by our tool.

It is important that all artifacts used in the case are contained in both sets and that the sets do not contain too many unique artifacts because this would suggest that the snapshots are not reliable. Similar to information retrieval research we can thus adapt the metrics of precision and recall. n independent investigators gather each a set of artifacts A_i.

Precision$_j = \frac{|\bigcup_{i=0}^{n} A_i \bigcap A_j|}{|A_j|}$ Recall$_j = \frac{(|\bigcup_{i=0}^{n} A_i) \bigcap A_j|}{|\bigcup_{i=0}^{n} A_i|}$. Both can be combined to the F score.

$$F = 2 \cdot \frac{\frac{|\bigcup_{i=0}^{n} A_i \bigcap A_j|}{|A_j|} \cdot \frac{(|\bigcup_{i=0}^{n} A_i) \bigcap A_j|}{|\bigcup_{i=0}^{n} A_i|}}{\frac{|\bigcup_{i=0}^{n} A_i \bigcap A_j|}{|A_j|} + \frac{(|\bigcup_{i=0}^{n} A_i) \bigcap A_j|}{|\bigcup_{i=0}^{n} A_i|}} \tag{5.1}$$

5.3.1 Social Snapshots on Facebook

At the time of writing Facebook is the most popular online social network with a claimed user base of over one billion users. Furthermore, Facebook supports third-party applications and user profiles contain a plethora of information. We thus decided to evaluate our social snapshot tool on Facebook. Third-party applications on Facebook have access to account data via the Graph API [67]. Almost the entire account data of Facebook users and their contacts are made available through their API. Facebook solely makes sensitive contact information such as phone numbers and e-mail addresses inaccessible to third-party applications. Hence our social snapshot client crawls the contact information of Facebook profiles, while all remaining social data is fetched through a custom third-party application. In October 2010, Facebook introduced a download option [71] that enables users to export their account data.

Element	**Download**	**social snapshot**
Contact details	–	✓Crawler
News feed	–	✓Graph API
Checkins	–	✓Graph API
Photo Tags	–	✓Graph API
Video Tags	–	✓Graph API
Friends	name only[a]	✓Graph API
Likes	name only[a]	✓Graph API
Movies	name only[a]	✓Graph API
Music	name only[a]	✓Graph API
Books	name only[a]	✓Graph API
Groups	name only[a]	✓Graph API
Profile feed (Wall)	limited[b]	✓Graph API
Photo Albums	limited[b]	✓Graph API
Video Uploads	limited[b]	✓Graph API
Messages	limited[b]	✓Graph API

[a] No additional information available.
[b] Missing meta-information such as UIDs.

Table 5.1: Account information available through social snapshots compared with Facebook's download functionality.

Table 5.1 outlines the different profile content elements gathered through our social snapshot application as compared with Facebook's download functionality. As shown in Table 5.1, the download functionality only offers a very limited representation of a user's online activity. For example, for a given user's friends, only their ambiguous names are made available and no information on the activity of a given user's friends is included.

5.3.2 Hardware and Software Setup

To test the functionality of our social snapshot application, we developed a third-party application for Facebook based on their PHP Graph SDK. One of the main modifications we performed on their original library was the support for multi-threaded API requests. Our third-party social snapshot application for Facebook is thus able to handle a number of predefined API requests simultaneously. The single requests are hereby pushed on a request queue with a specific priority. Hence our third-party application can be configured to, for example, fetch private messages before user comments of a Facebook group. The extent/depth of social snapshots can be further configured as a parameter for our third-party application. We deployed it on a Linux server in our university network.

Our third-party application fetches Facebook elements of a given account and stores them as separate JSON files. The separate JSON files correspond to specific requests, whereas the files are named as follows. The first part of the JSON file name is the ID of an API object while the second part specifies the requested connection detail. For instance, "123456789∼friends.request" contains all friends of the object with ID 123456789 formatted as a JSON object. In order to improve the performance of our application, we configured it not to download any videos or photos through the Graph API directly. As the third-party application collects direct links to photos, the digital image forensics module was configured to download photos during the analysis phase. Once the third-party application is finished fetching account data, it creates a tarball containing the social snapshot data.

The social snapshot client was adapted to fetch contact details of given user profiles and automatically add our third-party application to a target account. One particular challenge we had to overcome was to reliably obtain the list of friends of a given target account. Obstacles we had to cope with were the changing layout of the friend lists as well as Facebook only displaying a random subset of friends at a given time. We overcame the obstacles of creating the list of friends to be crawled, by fetching it through our third-party application and sending the profile links back to the client. Our client generates requests for every friend of the target user and sends them to the Selenium server that automates a Mozilla Firefox browser. The responses from the automated web browser module are parsed by the client and the contact information is extracted with a set of XPath queries. The client finally creates a CSV-file containing the contact information of all users. We deployed our client application in a virtual machine with a standard Ubuntu Desktop that runs our patched Selenium server. Our social snapshot analysis module implements both a parser for the fetched JSON Graph API requests as well as for fetched CSV contact details. The analysis module merges the results from the social snapshot client and the third-party application into a single database. We implemented the analysis module in Java.

We furthermore extended our digital image forensics module to automatically search a social snapshot for photo links, which it automatically downloads from the Facebook content distribution network. The hijack module did not require any Facebook specific modifications as it simply strips cookies of a given domain from a monitored network connection.

5.3.3 Visualization

To evaluate our proposed basic visualizations we relied on Gephi [19], an open source graph visualization tool. Gephi is an interactive tool that features a number of clustering and data analysis metrics for graph structures, as well as layout algorithm for graph visualizations. We used Gephi in a semi-automated fashion, that is we relied upon the Gephi GUI to produce graph visualizations. Gephi is also available as a standard Java library[2] and our semi-automated visualizations could therefore be fully automated. Our timeline visualization, see Figure 5.7, is currently a mock-up and we did not implement any automated timeline visualizations.

5.3.4 Test Subjects and Setting

We recruited human volunteers via e-mail, describing our experiment setting. The e-mail contained the experiment instructions and a briefing on how their personal information is going to be stored and analyzed. Furthermore, we briefed volunteers on the ethics of our experiment: no Facebook account data is modified, the social snapshots are stored in an encrypted filecontainer, no personal information is given to third-parties nor published. The invitation to support this first social snapshot evaluation was sent to researchers and students in computer science. Finally 25 people gave their consent to temporarily provide us access to their Facebook accounts. Volunteers temporarily reset their Facebook account credentials, which we used to create a social snapshot of their accounts. Once a social snapshot had been created, we informed our test group to reset their account password.

We configured our third-party social snapshot application for fetching an extensive account snapshot. We found that 350 simultaneous API requests lead to the best performance results in a series of indicative experiments we conducted beforehand. Our third-party application was configured to fetch the following elements recursively:

- Highest priority ($priority = 3$)
  inbox, outbox, friends, home, feed, photos, albums, statuses

- Medium priority ($priority = 2$)
  tagged, notes, posts, links, groups, videos, events

- Lowest priority ($priority = 1$)
  activities, interests, music, books, movies, television, likes

Our priority settings ensure that important information is fetched first. Account elements with highest and medium priority are fetched with $depth = 2$ while elements with the lowest priority are gathered with $depth = 1$. Thus a social snapshot of a given user includes for example, her friend's groups, tagged pictures, links etc. but no pictures, comments, etc. are downloaded from her favorite television series. These social snapshot settings imply that not only the target's account is completely fetched but also social data on the targets' friends is collected.

[2]`https://gephi.org/toolkit/`

5.4 Results

In this section we discuss the evaluation of our social snapshot prototype as well our first results on the visualization of social snapshots. Finally we discuss a possible usage of our novel forensic method on basis of a practical example.

5.4.1 Social Snapshot Performance and Completeness

Figure 5.3 illustrates the time required by our third-party social snapshot application to snapshot the test accounts through the Graph API. Our third-party application required on average 12.79 minutes. Account elements of our test accounts were on average fetched with $93.1kB$ per second.

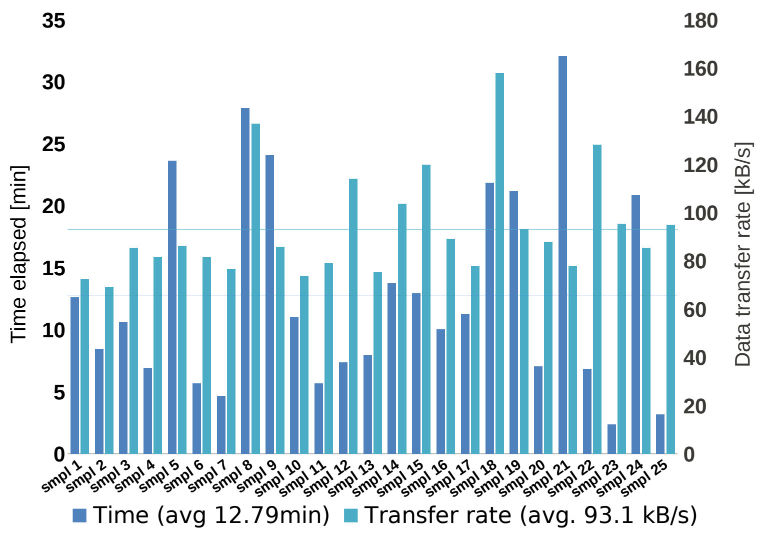


Figure 5.3: Time required by our social snapshot third-party application

Listing 5.1 shows an anonymized example from the fetched Facebook account elements. The example represents the basic information fetched of the user "John Doe" formatted as a JSON object. This example request also highlights that account data fetched through the Graph API provides a richer information set for further investigations. The standard web interface does not provide information if a user's account is verified nor an update time that is accurate to the nearest second with information on the used time zone.

Listing 5.1: Example of collected JSON user object

```
{
    "id": "12345678",
    "first_name": "John",
    "last_name": "Doe",
    "email": "johndoe@example.com",
    "birthday": "04/01/1975",
        "gender": "male",
        "hometown": {
            "id": "",
            "name": null
        },
        "username": "johndoe",
        "link": "http://www.facebook.com/johndoe",
        "locale": "en_US",
        "name": "John Doe",
        "quotes": "social snapshot your account!.\n",
        "timezone": 2,
        "updated_time": "2011-05-15T13:05:19+0000",
        "verified": true
}
```

Compared to data collected via the standard web interface, our social snapshot contains a number of additional information tokens. Most notably for forensic investigation is the availability of exact creation timestamps through the Graph API. We used our image forensic module to download all unique photos in the highest available resolution from the gathered social snapshots.

The time required for crawling contact details with our automated web browser is outlined in Figure 5.4. Test accounts have been crawled within 14 minutes on average. The average elapsed time per account corresponds to 3.4 seconds per user profile page.

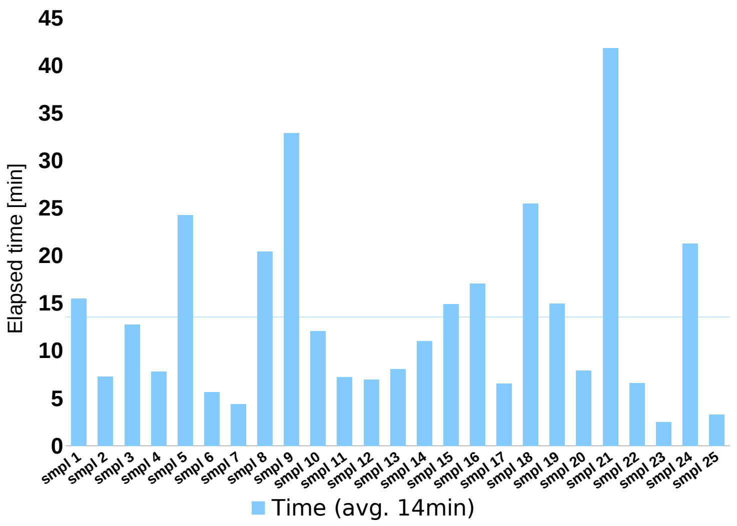


Figure 5.4: Time required for crawling contact details with social snapshot client and automated web browser.

As illustrated in Figure 5.5, our third-party application found and fetched on average 9,802 Facebook account elements per test subject. The storage size of the fetched JSON files accounted to $72.29MB$ on average. The downloaded photos corresponded to $3,250$ files or $225.28MB$ on average per test account.

Figure 5.6 shows the additional contact details crawled with our social snapshot client. On average, our social snapshot client had to crawl 238 profile sites per test account. For all crawled profile pages our crawler found 22 phone numbers, 65 instant messaging accounts, as well as 162 e-mail addresses on average. We noticed that after a number of subsequent requests to user profiles of a given account, Facebook replaces textual e-mail addresses with images. This behavior was noticeable with our social snapshot client, whereas on average we fetched 85 e-mail addresses in image form (OCR in Figure 5.6). Due to the fact that Facebook uses e-mail addresses in image form as a web crawler protection method, we could not directly parse the fetched images.

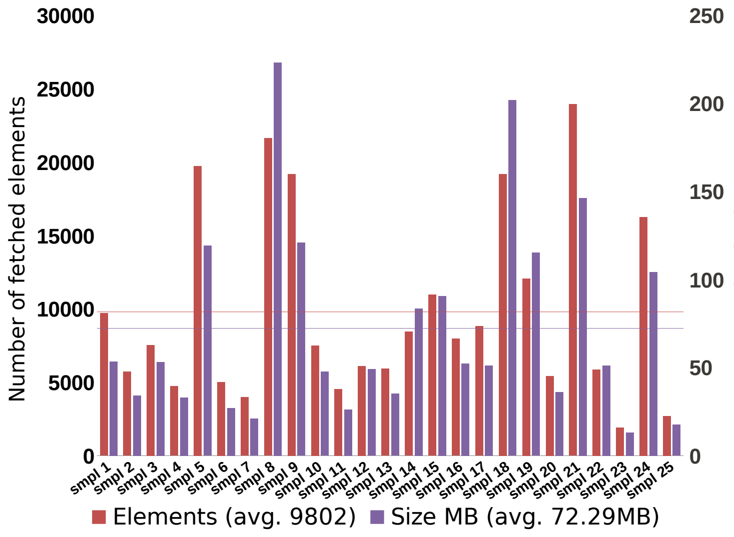


Figure 5.5: Account elements fetched through social snapshot third-party application.

Finally we used our analysis module to verify the integrity of the collected snapshots. We successfully verified that every entry in our fetched contact details CSV files had correspondent entries within the retrieve JSON files, as well as that no invalid responses where received through the Graph API. We furthermore implemented a mechanism for the analysis module to overcome the obstacle of parsing image e-mail addresses. By providing Facebook's e-mail image creation script the maximum possible font size of 35 instead the default of 8.7, we fetched higher resolution versions of the e-mail address pictures. We could thus rely on GNU Ocrad [82] to resolve these high resolution images into their textual representation. The idea of replacing the default font size with a larger one was first described in [151] and we could successfully verify that the described method still applies.

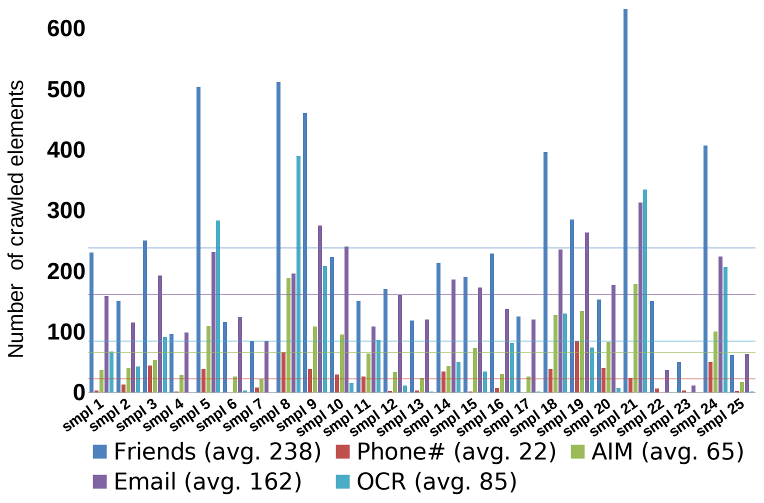


Figure 5.6: Contact details crawled with social snapshot client and automated web browser.

5.4.2 Forensic Analysis of Social Snapshots

Collected social snapshots enable the forensic analysis of social network activity of specific users and their online peers. Since the entire content of a users' social networking account with exact timestamps is collected, timelines can be easily generated. Moreover, social snapshots offer a valuable source for further investigations. The collected e-mail addresses could for example be used to identify users on other online platforms such as photo and file storage services, while collected media data could be matched with evidence collected through traditional forensic images. Figure 5.7 shows an example of a generated timeline for a fictitious forensic investigation on the

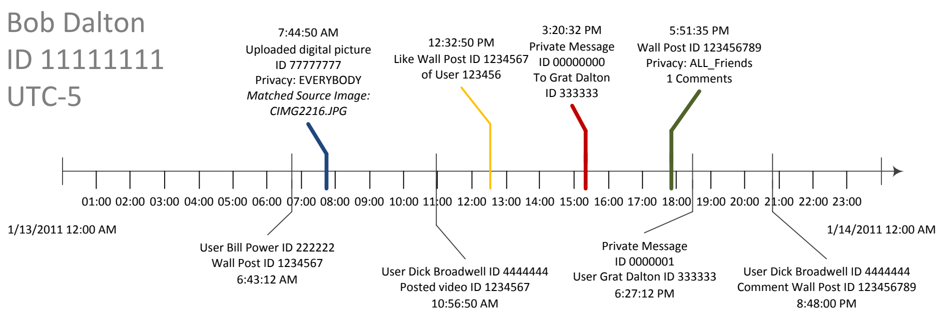


Figure 5.7: Example timeline created from collected social snapshot.

"Dalton Gang". The Dalton gang is suspected of having committed an aggravated bank robbery between 8:00am and 8:30am on the 13th of January 2011. All four gang members have an alibi for the specific time and said they were all on a joint getaway together with their families. Bob Dalton, the head of the gang, presents a group photo he posted on Facebook that very day. In order to validate the posting, five close friends of Bob give their consent to *social snapshot* their social networking accounts. While the posted group photo correctly shows up with the specified date in all five social snapshots, an interesting posting from Bob Dalton's wife is collected in two of the social snapshots. The posting dated one week before the robbery, timestamped with 01/06/2011 07:32:12 AM reads "Off to the beach, for our family group picture. Hehe". The investigators at this point start to suspect that the alibi picture had been taken a week beforehand to fabricate an alibi. Unaware to Bob's brother Grat Dalton, investigators social snapshot his account using the *hijack module* during his daily Internet browsing, exploiting a coffeeshop's insecure WiFi connection. Analyzing Grat's social snapshot the investigator noticed that Grat exchanged private messages with his brother Bob on the day of the robbery. The first messages with ID 00000000 sent at 3:20:32 PM reads "Grat, That was almost too easy today ... we should start thinking on how to spend all the Benjamins:-). greetings Bob". In the second message Grat replied to Bob at 6:27:12 PM: "Yeah almost too easy:-) Great idea with the group picture at the beach btw, that will cause them some serious teeth gnashing." With this further evidence on a possible false alibi, the investigators perform a house search on Bob Dalton's home. While the search does not reveal any of the stolen money, the personal computer of Bob Dalton is seized during the house search. Amongst digital documents and images the investigators find a valid Facebook *authentication cookie* on Bob's forensic image. The investigator creates a social snapshot of Bob's social networking account using the extracted authentication cookie. Comparing Bob's and Grat's social networking activity on the day of the robbery they find that the social snapshots accurately correlate with a F1-score of 0.84, and both accounts hold the treasonous private messages. The timeline generated from the social snapshot and outlined in Figure 5.7 shows Bob's online activity on the day of the bank robbery. Curious as to whether the pristine digital image of Bob's posting can be recovered the investigator runs the *digital image forensic module* to match digital images from the forensic image with the image collected through the seven independent social snapshots. The digital image forensic module reports a positive match on a digital image named "CIMG2216.JPG". The original EXIF information of image "CIMG2216.JPG" reveals that their alibi group picture had indeed been taken a week before the robbery.

5.4.3 Visualization Results

In the following we describe our visualization results, we replaced the actual personal information of social snapshots with a random subset of the list of computer scientists on Wikipedia[3]. As the replaced set is random, it obviously does not show real connections between the computer scientists.

For the social interconnection graph we relied on a feature from the Facebook API that allows an application to query if two specific users are connected. We iteratively tested if the first friend is in connection with the $n-1$ friends of the tested profile, then tested for the second the remaining $n-2$, and so forth. We then extracted the social interconnection graph and plotted it with Gephi using the Fruchterman-Reingold algorithm [81] to automatically layout the graph. An example of a social interconnection graph from one of the authors can be seen in Figure 5.8.

[3] `http://en.wikipedia.org/wiki/List_of_computer_scientists`

The size of single nodes depends on its degree. The nodes are furthermore clustered and colored on basis of communities. Hereby, we used Gephi's modularity feature, which implements the community detection algorithm by Blondel et al. [28]. The automated visual representation of communities further supports quick analyses by forensic investigators.

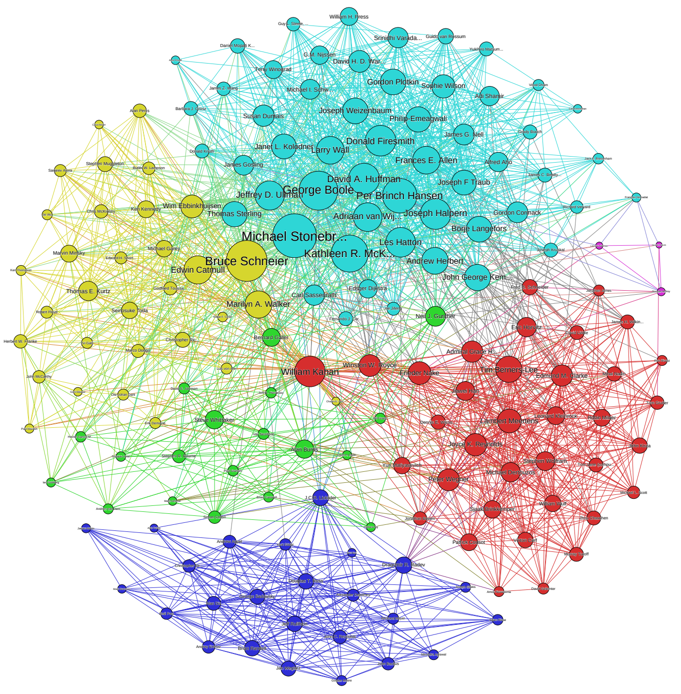


Figure 5.8: Anonymized Social Interconnection Graph

With the data from Facebook it becomes possible to create different social interaction graphs. In our implementation we created different graphs for different forms of interaction, while they could be easily integrated to a complete social interaction graph. An example for a social interaction graph based on tags in pictures on Facebook can be seen in Figure 5.9. The graph is created using the following steps: (1) starting from an account under investigation, all pictures from all friends are collected, and searched for tagged people. (2) People that are tagged in pictures, and not in the list of friends, are ignored. (3) If the tagged person is in the friend list as well, an edge is added between the two nodes pointing from the profile that uploaded the picture to the profile that is tagged, or the weight increases by one if the edge already exists. The edges are directed and weighted. An investigator can find with this graph persons that have a tight social connection.

Yet another form of social interaction graph can be created using direct messages: instead of using picture tags, an edge is added between two nodes if the profile under investigation exchanged messages with the other profile. Intuitively, an edge pointing to a node represents a message sent to that profile. Again the edges are weighted, for the number of messages exchanged. An example for a social interaction graph using direct message communication can be seen in Figure 5.10.

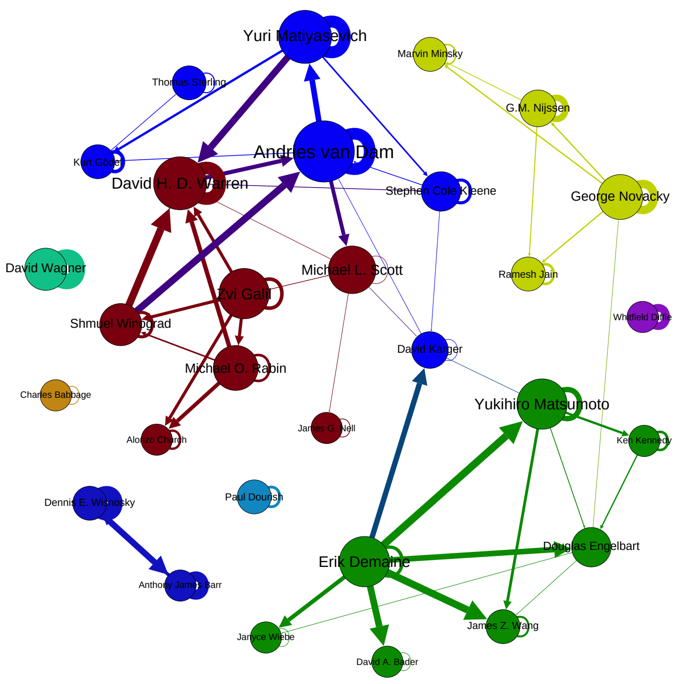


Figure 5.9: Anonymized Social Interaction Graph using Picture Tags

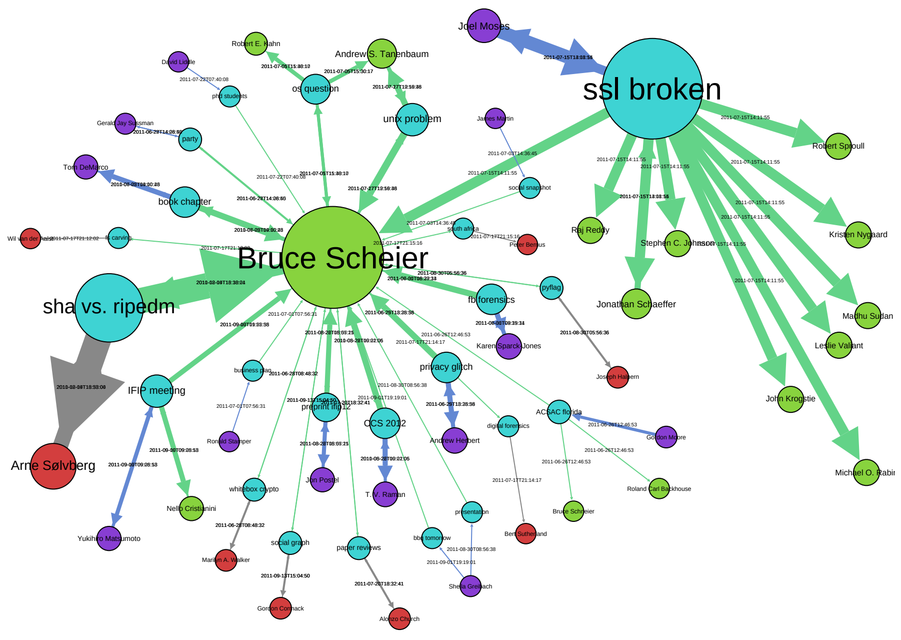


Figure 5.10: Social interaction graph using direct messages

5.4.4 Social Snapshot Open-Source Release

We release the social snapshot core framework for Facebook under a GPL v3 open source license[4]. The source code contains the social snapshot client, our third-party application, as well as the patched Selenium server. Not included in the open source release are the analysis and photo forensics modules. We furthermore decided not to release the hijack module, which could be potentially misused for malicious attacks.

5.5 Discussion

Our evaluation required on average $9,802$ API and 238 HTTP requests to successfully snapshot an entire social networking account in less than 15 minutes. In order to collect forensic evidence with traditional web-crawling more than $10,000$ HTTP requests are necessary to snapshot a single test account. The generated network traffic of traditional web-crawling would have been likely detected and blocked by social networking providers. Moreover, our evaluated approach retrieved the great majority of social networking account data without the requirement of additional parsing and with exact timestamps. During the implementation of our social snapshot techniques, Facebook's web-site layout changed a number of times. Since only contact details were crawled, we could promptly adapt the parser of our client, while our third-party application did not require any changes at all. As Facebook has no mandatory review process for third-party applications we could also make our third-party application available straightforward. Third-party applications on Facebook do not even have to appear in their application directory in order to be usable.

Apart from digital forensics, social snapshots could also be used to raise user awareness. Users would run our social snapshot tool and get a report on their account data. Thus, social networking users could sight the magnitude of information that is stored with their social networking providers. We hope that this would help the average social networking user to make better informed decisions on which information they post.

Unencrypted social networking sessions enable the gathering of social snapshots for digital forensics but also pose a serious security threat. Since TLS is not enabled by default on the majority of today's social networking services, user sessions can easily be hijacked. Two proof-of-concept tools have been released that make session hijacking of social networking sessions available to the average user. *Firesheep* [43] has been released in October 2010 as a browser extension and at the time of writing is not functioning anymore. *Faceniff* [149] offers a point-to-click interface and supports a number of wireless network protocols. It is an Android application for hijacking social networking sessions released in June 2011. Both hijacking applications were released in order to create awareness for the problem of insecure social networking sessions. It is trivial however to couple such simple hijacking applications with our social snapshot tool. Thus, attackers could harvest complete account snapshots in an automated fashion. In chapter 6 we further show that the large amount of sensitive data stored in social networks could be misused for large-scale spam attacks via session hijacking and malicious social snapshots.

[4]`https://github.com/markushuber/social-snapshot-tool`

5.5.1 Threats to Validity

Whilst our method is novel and can be easily used in addition to already existing and deployed social network analysis methods (i.e., subpoena requests to the social network operator) they introduce new challenges at the same time as they solve others. One of the most obvious drawbacks is that the data collection is hardly reproducible: the timelines and graphs generated will look differently for multiple runs as the social network is very dynamic in nature, and the amount of data rather big. Within a single day a user could change his social interconnection graph to large extend, and could try to hide his or her communication in covert traffic. Some other data, like login IP addresses as provided by the Facebook NeoPrint [62], are only available to the operator of the network and not accessible with our method, neither with an automated webbrowser or an API. Furthermore it is not easily possible to guarantee completeness: once data is deleted by the user it can only be undeleted from the social network operator, and is thus not available for our analysis. We believe, however, that most of the data items are rather static in nature, and that our methods are applicable and auxiliary for forensic investigations.

CHAPTER 6

Friend-in-The-Middle (FiTM) Attacks

Criminals, as well as direct marketers, continue to clog mailboxes with unsolicited bulk e-mails in the hope of financial gain. So far, their strategy is straightforward, namely to send out a vast numbers of unsolicited e-mails in order to maximize profit on the tiny fraction that falls for their scams. Their pool of target e-mail addresses is normally based upon data harvested with web crawlers or trojans, sometimes even including plain dictionary-based guessing of valid targets. Previous research indicates that online social networks might change the playing field of spam attacks in the near future (see Subsection 3.2.2). OSNs contain a pool of sensitive information which can be misused for spam messages, namely contact information (email addresses, instant messaging accounts, etc.) and personal information which can be used to improve the believability of spam messages. A successful extraction of sensitive information from OSNs would result in spam attacks that are based upon a pool of verified e-mail addresses. Thus messages may have higher conversion rates, increasing the success rate of spam.

Gaining access to the pool of personal information stored in OSNs and impersonating a social network user poses a non-trivial challenge. Gross and Acquisti [88] as well as Jones and Soltren [112] were among the first researchers to raise awareness for information extraction vulnerabilities of OSNs. While their techniques were rather straightforward (automated scripts which retrieve web pages), their results eventually led to security improvements of OSNs. Existing attempts to extract information from OSNs focus on the web application layer and can thus be mitigated by adapting a specific social network's web application logic. The leakage of personal information from these platforms creates a remarkable dilemma as this information forms the ideal base for further attacks. Jagatic et al. [110] showed that they could increase the success rate of phishing attacks from 16 to 72 % by posing as a target's friend. In social engineering, additional available information on targets could lead to automated social engineering attacks [100]. The main obstacle for large-scale spam attacks on basis of OSNs are the various access protection measures providers offer to keep sensitive information private or at least limit access to a closed circle of friends. Our friend-in-the-middle (FiTM) attack overcomes this obstacle by hijacking HTTP sessions on the network layer, which the majority of OSNs providers fail to secure appropriately.

6.1 Friend-in-The-Middle (FiTM) Attacks

We define friend-in-the-middle attacks as active eavesdropping attacks against social networking sites (see Subsection 3.2.1). Our FiTM attack is based on the missing protection of the communication link between users and social networking providers. By hijacking user HTTP sessions, it becomes possible to impersonate victims and interact with the social network without proper authorization. While at first glance the risk of hijacking social networking seems like yet another threat to privacy, we claim that FiTM attacks enable large-scale spam attacks. Within this section, we first explain various attack scenarios on basis of session hijacking and describe how FiTM attacks could be misused for large-scale spam campaigns on basis of Facebook.

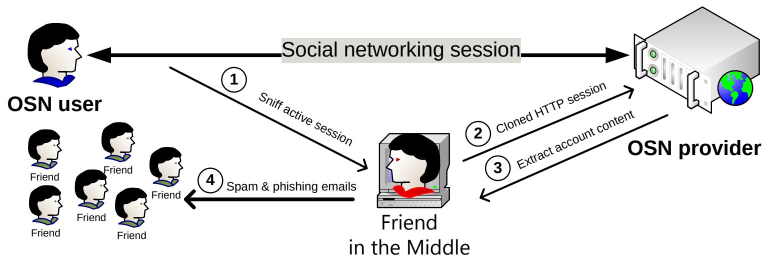


Figure 6.1: An attacker becomes the "friend-in-the-middle" by hijacking an OSN user session and targeting the user's friends.

6.1.1 HTTP Session Hijacking Attacks on OSNs

As a precondition the attacker needs to have access to the communication channel between the OSNs and the user. This can be achieved either passively (e.g., by monitoring unencrypted wireless networks) or actively (e.g., by installing malicious software on the victim's computer). The adversary then simply clones the HTTP header containing the authentication cookies and can interact with the social network, unbeknownst to the OSN operator or user. The victim is unable to detect or prevent such attacks and the attacker is able to use the social network to its full extent from the victim's point of view. As with all HTTP session hijacking attacks, it becomes possible to both retrieve information (*data acquisition from the social network*) as well as to insert malicious requests on the behalf of a user (*data publication into the social network*). However in the case of our FiTM attack, further scenarios become available to attackers, which are specific to social networking sites:

- *Friend injection* to infiltrate a closed network
- *Application injection* to extract profile content
- *Social engineering* to exploit collected information

The rudimentary security and privacy protection measures of OSNs available to users are based on the notion of "friendship", which means that sensitive information is made available only to a limited set of accounts (friends) specified by the OSN user. Once an attacker is able to hijack a social networking session, she/he is able to add herself/himself as a friend on behalf of the victim and thus infiltrate the target's closed network. The *injected friend* could then be misused to access profile information or to post messages within the infiltrated network of friends. By *injecting* a custom third-party *application*, written and under the control by the attacker, it is possible to access the data in an automated fashion (see Chapter 5). Among other things, an application has access to sensitive information (birthday, email address, demographic information, pictures, interests) and with most OSNs even access to information of friends of the application user. Third-party applications such as online games have become popular features of OSNs, and hiding a malicious application without any activity visible to the user is possible. Thus, the application is likely to remain undetected within a pool of installed third-party applications. This ultimately enables an attacker to extract profile content in a stealthy way as this retrieval method does not cause as much noise as a burst of separate HTTP requests. Even worse, the attacker might install the application, take all the data needed in an automated fashion and remove the application afterwards. This would be completely undetectable to the user and most likely to the OSNs providers as well. Whereas *social engineers* traditionally relied upon context-information gathered through dumpster diving or quizzing people over the phone, with FiTM attacks the context-information harvesting process becomes automated. We thus claim that FiTM attacks allow sophisticated social engineering attacks. In the following we outline how context-aware spam and social phishing (see 3.2.3) could be misused for automated social engineering.

6.1.2 Large-scale spam campaigns through FiTM attacks

Figure 6.2 illustrates a spam campaign exploiting our novel FiTM attack.

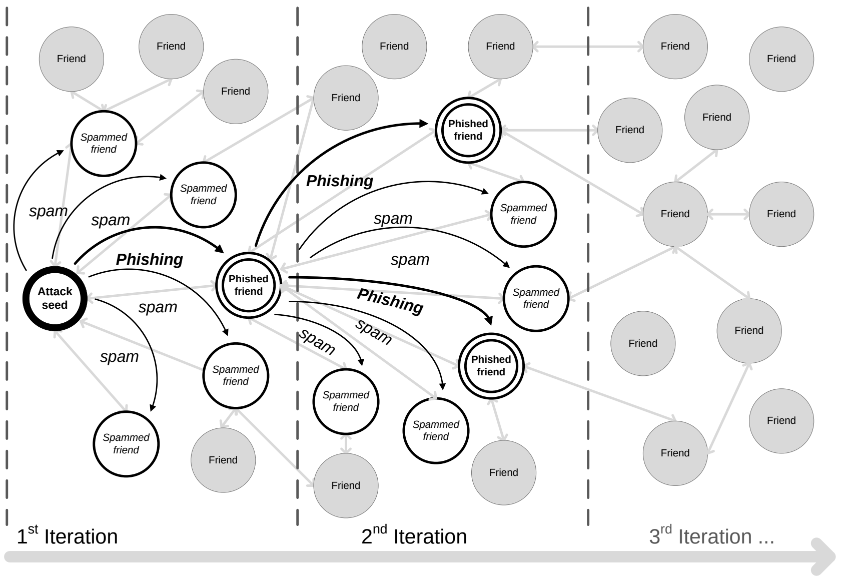


Figure 6.2: Outline of a large-scale spam campaign via the friend-in-the-middle attack: A social networking session is hijacked to fetch personal information from a victim's profile. The extracted information is then used for spam and phishing emails targeted at the victim's friends.

(1) In the first step, a network connection is monitored. Once the FiTM application detects an active social networking session, it clones the complete HTTP header including the session cookie. **(2)** The cloned HTTP header serves then as a valid authentication token for the OSN provider and is used to temporarily hijack the OSN user's session. **(3)** In order to extract the profile content as well as information on the target's friends, a custom third-party application is added to the target's profile. Once all information has been extracted the application is removed from the profile. Additional queries are used to fetch the email addresses of the target's friends in case they cannot be retrieved through the third-party application. In this step we thus basically create a malicious social snapshot of a users profile (see Chapter 5). **(4)** The extracted email addresses and account content are used to generate tailored spam and phishing emails. While the spam messages contain the actual payload of the attack, the phishing emails are used to steal credentials of the target's friends for further propagation (the FiTM attack starts again from (3) with the phished OSN account credentials).

6.2 Methodology

We decided to evaluate the impact of a large-scale spam campaign on basis of Facebook. FiTM attacks based on Facebook serve in our opinion as a good example because it is the biggest OSN at the time of writing, HTTPS is per default only used to protect login credentials, and Facebook supports custom applications. Furthermore, injections of third-party applications into Facebook profiles promise access to a plethora of personal information. Within the Facebook application framework, third-party applications can for example access the following information:

- *Basic context information*: Full name, geographical location, birthday, affiliations, education, etc.
- *Likes and interests*: Favorite books, movies, tv-series, music, quotations, etc.
- *Private content*: Sent and received messages, photos, videos, etc.

In addition, third-party applications within Facebook are allowed to access the information of a user's friends as well. Thus an application injection in Facebook enables the extraction of a pool of valuable context information from the targeted user as well of his/her friends. Email addresses of users are not accessible through third-party applications and the addresses can be collected by using the hijacked user session. The different permission classes available for social networking apps, are further outlined in Section 2.3.

In order to make assertions on the effectiveness of our FiTM attack, an experiment which mimics a real large-scale attack would provide valuable insights on effectiveness, but also raises serious ethical concerns. Hence, we applied the following twofold approach: we performed an empirical evaluation on the number of possible sessions that could have been hijacked, without collecting any data or injecting any malicious requests. We then simulated the impact of our FiTM attack on basis of established results from similar work on a graph model. In comparison to survey-based methods we avoid problems such as selection bias, refusal rates, telescoping, forgetting and exaggeration [96]. The methodology applied within our two experiments is explained in this section.

6.2.1 Finding attack seeds

To conduct the FiTM attack, numerous attack vectors could be used: DNS poisoning, cross-site request forgery (*CSRF*), wireless networks without or with only weak encryption, malicious software like a trojan or a rootkit running on the victims computer, deep packet inspection from an ISP or other malicious entity that has access to the traffic between the client and the OSN, as well as modified software running on a users residential or company gateway.

In order to perform first indicative experiments we decided to measure the number of active social networking sessions on unencrypted WiFi access points. We chose the library of a big Austrian university as our experiment location. The university's library was selected because of two reasons: OSNs are very popular amongst students, and secondly this particular library offers both secure and insecure (unencrypted) Internet access via wireless LAN. The university's wireless LAN is operated on three channels: *1*, *6*, and *11*. To capture all three channels we equipped a laptop with three WiFi USB dongles with an analysis script listening to each of the three channels. In a first experiment we performed the analysis over the period of one and a half hours and in a second experiment we performed the evaluation for seven hours.

As a second source of possible FiTM attack seeds we analyzed social networking sessions passing through a Tor exit node. We selected this second source of possible FiTM attack seeds based on results of previous experiments, where we discovered that unencrypted social networking sessions are popular within the Tor network [103]. The Tor network [60] is a widely deployed anonymization network which hides the user's IP address on the Internet. It is expected to be used by hundreds of thousands of users every day and is believed to be the most heavily used open anonymization network today [182]. The Tor infrastructure relies on servers run by volunteers, hence anyone can support the Tor project by setting up a dedicated Tor server. For our experiment, we set up a Tor exit node on a minimal GNU/Linux Debian server with a relay bandwidth rate of 5 Mbit. The server was furthermore configured to only allow HTTP traffic (TCP port 80) from the Tor network to the Internet. We then counted the number of Facebook sessions together with the Facebook locales that were observable by our Tor exit node and could have be used for our FiTM attack. Note that we only counted and saved the number of unique social network sessions, not the number of users that used our Tor node.

During our experiments we ensured that the privacy of user data was not put at risk. We did not collect or store any personal data of observed user sessions. To prevent counting the same users multiple times, we matched the hash value of unique user identifiers of detected sessions against a list of previously hashed user identifiers.

6.2.2 FiTM attack simulation.

We estimate the impact a large-scale FiTM attack on basis of the following simulations.

Finding an optimal FiTM attack strategy

We implemented a configuration model to generate an extended induced subgraph (see Subsection 2.2.2) of Facebook. Hereby, we constructed a simple graph out of a degree distribution p_k, such that p_k is the fraction of vertices in the graph having degree k. Then choosing a degree sequence $\mathbf{d} = \{d_i \mid i = 1, ..., n\}$, WLOG we assume $d_i \geq 1$, which are the degrees of the n vertices $\{v_1, v_2, ..., v_n\}$. Since it is not always possible to construct a simple graph with a given degree sequence [186], we first constructed a *multigraph*. Obtaining a simple graph out of such a multigraph is easily achieved by erasing all loops and combing all multiple edges into one. The obtained simple graph has asymptotically the same degree distribution. It has been shown [140] that the chance of finding a loop goes as n^{-1}, therefore the probability is humble for large n. For an applicable degree distribution we considered the much studied *power-law* distribution [141], more distributions are discussed in Subsection 2.2.2. Distributions of the form $p(x) = Cx^{-\alpha}$ are said to follow a power law, where $\alpha > 0$ is called the *exponent* and C functions as a *normalizing constant*. C is given by the following normalization requirement:

$$1 = C \int_{x_{min}}^{\infty} x^{-\alpha} dx = \frac{C}{1-\alpha} \left[x^{-\alpha+1} \right]_{x_{min}}^{\infty} \tag{6.1}$$

Formula 6.1 shows that: 1) $\alpha > 1$ and 2) for a given $\alpha > 1$ and known limit x_{min} it is easy to compute the normalization constant C. Gjoka et al. [86] presented a possible degree distribution for Facebook which does not follow a single power-law but instead they found two regimes $1 \leq k < 300$ and $300 \leq k \leq 5000$, each following a power law with exponent $\alpha_{k<300} = 1.32$ and $\alpha_{k \geq 300} = 3.38$. With this specific information it was possible to generate a accurate power law degree sequence for the two intervals $[1; 300[$ and $[300; 5000]$. We generated a model with a total amount of $1 \cdot 10^4$ nodes and computed C for each of the two intervals with formula 6.1.

Attack Cycle. For an attacker the properties of the entire graph is unknown. An attacker knows the degree of a hijacked user vertex but other useful knowledge such as centrality, betweenness, or clustering coefficient etc. is unknown to the attacker. The attack process thus behaves as follows: We choose a random vertex v_i, in the following called **user**, the user has a predetermined degree ($d(v_i) = d_i = k$) which is the amount of friends. We *spam* a fixed percentage p of the users friends and *propagate*[1] the remaining ones. This cycle then repeats itself for a given amount of iterations, e.g. 5 times ($it = m = 5$). With these iteration steps we probably get more vertices to further spread our attack compared with a single vertex. In order for the attack to be less noticeable, we assume that a user can either be *spammed* or *propagated* and no more than 1 time. We performed two consecutive evaluations to find an optimal attack based on our graph model. **Optimization 1:** Find an optimal spam to phishing ration p, hereby we randomly choose a single attack seed and perform $it = 1, ..., m$ attack iterations for different values of p. **Optimization 2:** Find an optimal number of attack iterations with our optimized value of p and multiple attack seeds $as = 1, ..., 35$.

Simulating a large-scale attack

In a second step we simulated the impact of an optimized FiTM attack on a basis of an additional graph model. We used the anonymous regional network collected by Wilson et al. [191], which contains 3×10^6 Facebook nodes, as basis of our attack simulation. The goal of our simulation was to estimate how many Facebook nodes would be affected by a large-scale FiTM spam attack given a specific amount of initial attack seeds and our optimized attack strategy.

[1]Our evaluated propagation strategy is social-phishing, therefore we assume that propagations have a success rate of 72% based on Jagatic et al.'s findings [110].

6.3 Results

In this section we present the results of finding possible attack seeds as well as the outcome of our attack simulations.

6.3.1 Finding Attack Seeds

Indicative WiFi experiments

We performed two indicative experiments on possibly hijackable social networking sessions in February 2010. Within our first WiFi experiment we counted social networking sessions for a period of 1.5 hours during the peak time of the university's library, where we expected most of the students would use the library's wireless access points. Figure 6.3 shows our findings, on average our analysis tool detected a unique social networking session every 1.8 minutes.

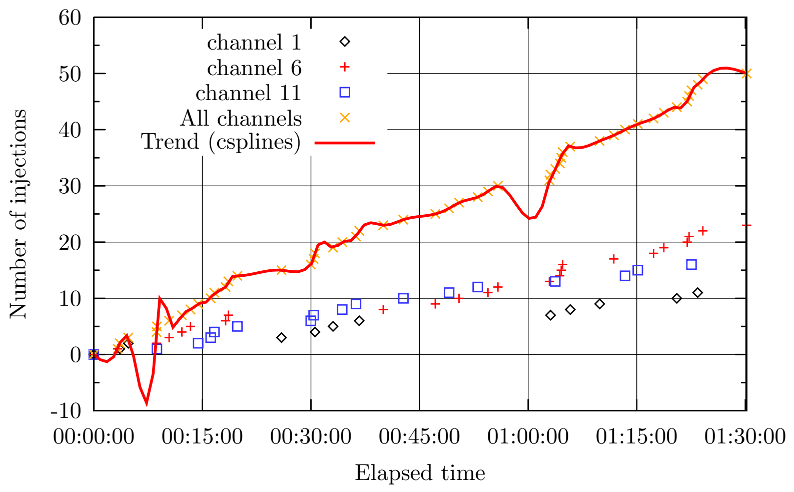


Figure 6.3: Observed social networking sessions measured during peak time of Internet usage.

The second WiFi experiment was carried out during an average day in the university's library were less students were using the Internet. During a period of seven hours we were able to detect 60 social networking sessions which corresponds to one unique session every seven minutes.

Tor exit node

During a period of 14 days, approximately 6.1 x 10^6 HTTP requests passed through our Tor exit node. Facebook was the most requested domain and was responsible for 7.68 % of the overall traffic. The second most frequent social networking site was Orkut which caused 0.49 % off all HTTP requests. We observed 4,267 unique Facebook sessions throughout our experiment which could have been hijacked for friend-in-the-middle attacks. Furthermore our cookie analysis suggests that the majority (92.81 %) of observed unique Facebook sessions were persistent sessions. Figure 6.4 outlines our findings, whereas the red graph represents the unique attack seeds during a period of 14 days.

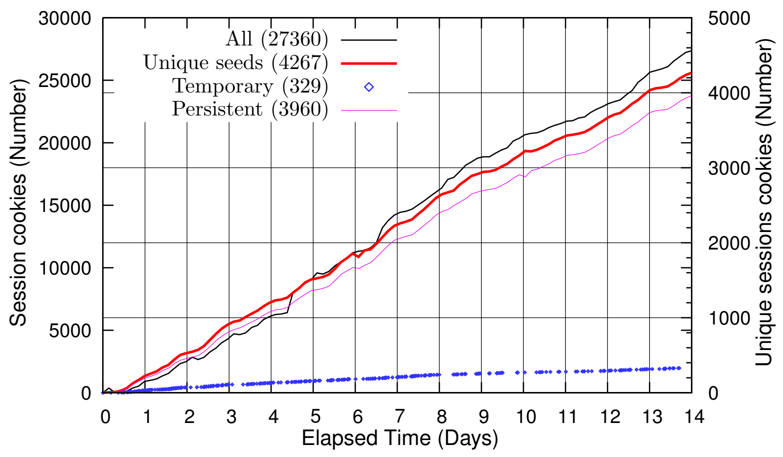


Figure 6.4: Number of sessions found through our Tor exit node server within 14 days.

Table 6.1 shows the distribution of the Facebook sessions in regard of the used language. 71.77 % of all users used English, followed by 6.19 % used an Italian, and 5.55 % of users used Spanish. We furthermore observed that in total 3.45 % of users might have originated from China and 1.28 from Iran, where in both countries Facebook is blocked by governmental authorities. The information of the different locales used could be exploited by attackers to adapt the language of their spam messages.

Facebook locale		
Language	ISO	%
English (US)	en_US	60.46
English (UK)	en_GB	11.31
Italian	it_IT	6.19
Spanish	es_LA	5.55
Indonesian	id_ID	3.21
Simplified Chinese (China)	zh_CN	2.41
French (France)	fr_FR	2.01
Persian	fa_IR	1.28
Traditional Chinese (Taiwan)	zh_TW	1.04
Others	-	6.54

Table 6.1: Facebook locales of analysed user sessions

6.3.2 Simulation Results

Optimization 1

We ran our attack model for *optimization 1* for each number of iterations, $it = 1, ..., 35$, 1000 times. For each iteration step we averaged over the 1000 results and got the mean value for the potential spam targets. Figure 6.5 outlines our results for attack optimization 1. For example, for $it = 5$, the mean number of spam targets is $sp = 1178$ corresponding to the red curve. The five colored curves correspond to different percentages p of spam messages in each iteration step. The black curve describes the number of spam targets when only 1 user is phished in each iteration step. One can observe that with the initial attack iterations $(it = 1, ..., 10)$ the number of spammed nodes increases almost linear with a high slope. After that $(it = 11, ..., 35)$ the curve slowly levels to $sp \approx 643$. The colored curves behave similarly, but their linear growing area is reduced to 5 attack iterations. The slope is significantly higher for values of p between 50 and 80 percent. All curves nearly level to a final value within the first 5 iteration steps. Based on this results we tested an optimal ratio between spam targets and propagating percentage. Our results show that propagating a fixed percentage $100-p$ yields to better results as compared with only phishing 1 user in each attack iteration. One can also observe that a too small percentage of spamming targets ($p \leq 60\%$ in our simulation) yields to a decrease of spam targets. Based on our simulation, we found the value $p = 70\%$ to be the best choice for p. The leveling of all six curves yields to the assumption that in a highly clustered structure it is not possible to elude the entire cluster. We expected this behavior based on the hypothesis that OSN graphs represent an assembly of *extended induced subgraphs*. In the figure we limited the number of iterations to 35 because the slope for larger iterations converges to 0.

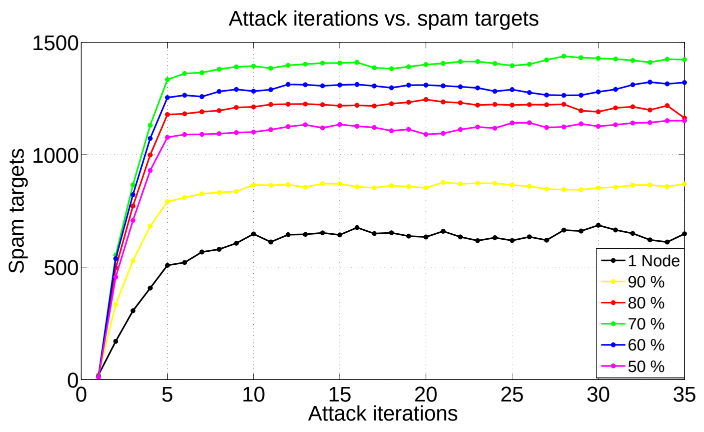


Figure 6.5: Simulation results for FiTM attack optimization 1

Optimization 2

Our second attack optimization can be seen as an extended algorithm of *Optimization 1*, whereas we used the optimized value $p = 70\%$ and varied the number of attack seeds $as = 1, ..., 35$.

As with Optimization 1 we averaged over 1000 results and got the mean value of spammed user vertices. Figure 6.6 outlines our simulation results for Optimization 2. We found that for one iteration $(it = 1)$ the slope is almost linear (red curve). This can be explained by the fact that in a large network we get almost the same number of *spammed users* for each attack seed. Furthermore, one can observe that there is significant difference in the number of *spammed* nodes between $it = 1$ and $it = 2$. This difference however decreases with a growing number iteration steps. While there is still a considerable difference between $it = 2$ and $it = 3$ and $it = 10$, there is almost none between $it = 20$ and $it = 35$. As in strategy 1, initially our simulation shows a rapid increase of spammed users $(as = 1, ..., 10)$ and after $as = 10$ the slope decreases. Our simulations shows that all curves converge to a limit of 3000 spammed nodes. This is a result of the fact that even with 35 iterations and 35 attack seeds, the chance of finding *unattacked* graph segments is negligible small. It is interesting to observe, that even with a moderate rate of iterations, say $it = 10$ per attack seed, we get almost the same amount of spammed nodes when executing $it = 35$ iterations per attack seed. Hence, performing an attack with multiple attack seeds and few iterations yields a higher impact than an attack with a single attack seed and a high number of iterations. Furthermore, attacks with multiple attack seeds are more inconspicuously because the attack spreads over multiple clusters in the social graph.

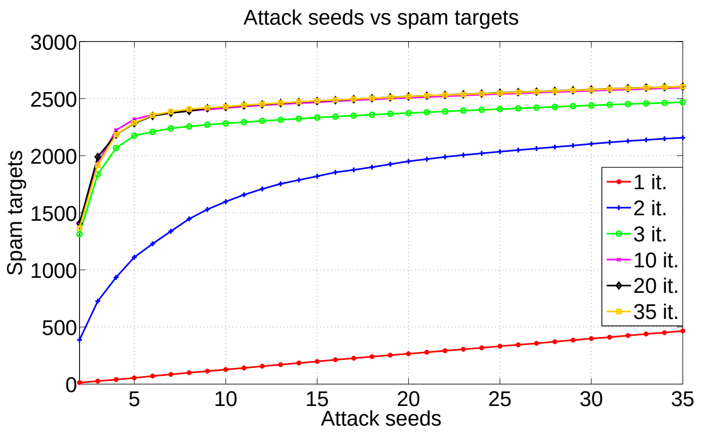


Figure 6.6: Simulation results for FiTM attack optimization 2

6.3.3 Impact Evaluation

We performed the impact evaluation on basis of our optimized FiTM attack cycle, with a spam ratio of $p = 70\%$ and a fixed number of attack iterations $it = 3$.

Table 6.2 shows our simulation results for 4,000 initial attack seeds with a propagation success probability of 72%. We chose 4,000 as a maximum amount of attack seeds because this value corresponds to the number of user sessions we observed with our Tor exit node (see Subsection 6.3.1). Our results suggest that with 250 seeds it is possible to spam 1.94×10^5 potential targets (6.28 % of the total vertices in our graph model). When comparing the first and the last line of the table, one finds a factor 16 between the number of corresponding seeds

attack seeds as	as [%]	spammed users su	su [%]
250	0.01	$1.94 \cdot 10^5$	6.28
450	0.01	$2.12 \cdot 10^5$	6.86
750	0.02	$2.27 \cdot 10^5$	7.35
1000	0.03	$2.40 \cdot 10^5$	7.77
1500	0.05	$2.58 \cdot 10^5$	8.35
2000	0.06	$2.70 \cdot 10^5$	8.74
3000	0.10	$2.90 \cdot 10^5$	9.39
4000	0.13	$3.03 \cdot 10^5$	9.80

Table 6.2: Number of spammed users depending on initial attack seeds.

$(250 \times 16 = 4000)$. This does not hold when comparing the possible spam targets: for an amount of 250 attack seeds we found 1.94×10^5 targets, with 4,000 seeds we found 3.03×10^5 targets, resulting in a growth factor 1.56. Our FiTM attack simulation on the regional network by Wilson et al. [191] with 4,000 attack seeds thus finds fewer user vertices that have not been already spammed. Our result of 3.03×10^5 overall spam targets represents however a conservative estimation as seeds collected trough a Tor server most likely belong to a more disperse set of Facebook clusters.

Because we did not actually perform phishing attacks, we were interested on the effect of phishing success rates $< 72\%$. Figure 6.7 shows the impact of the phishing success rate on number of spam targets. As the figure illustrates, even with a 10%-success rate, it is possible to spam 3% of the regional network. For higher success probabilities the difference between the curves decreases. Thus, even if social-phishing would succeed with a probability of 30% instead of 72% the overall effect would not be significant.

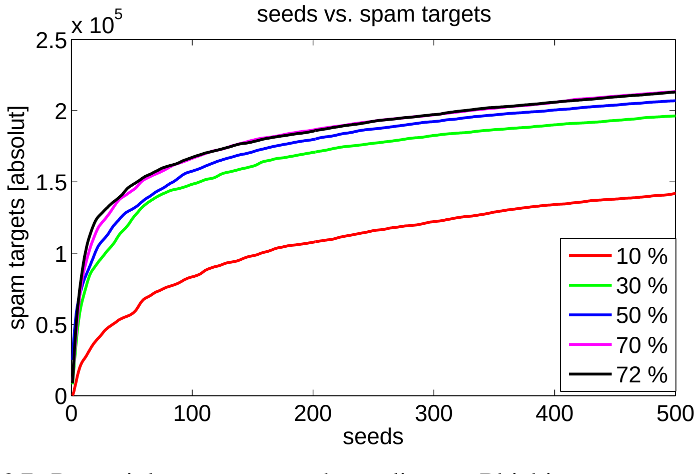


Figure 6.7: Potential spam targets depending on Phishing success probability

6.4 Discussion

Our results suggest that unprotected OSN user session can be observed and potentially hijacked when snooping on traffic of WiFi access points or Tor exit nodes. Our simulation results suggest that our *FiTM attack* would result in malicious spam campaigns on a large scale. In a relatively short amount of time, an attacker would be able to collect information from thousands of users in an automated fashion, resulting in hundreds of thousands possible victims for context-aware spam. For example 250 initially hijacked user sessions (attack seeds), would result in more than $1.94 \cdot 10^5$ users being targeted with context aware spam. With regard to our Tor experiment, these 250 initial attack seeds would have been gathered within 15 hours. Because our attack enables the extraction of e-mail addresses from social networking accounts, context aware could be delivered via email and thus evade detection from social networking providers. In February 2010, a user's email address was even included in Facebook session cookies in plaintext.

6.4.1 Mitigation strategies

In recent years numerous privacy protection schemes have been published with the intention to protect sensitive social networking data from unwanted disclosure. One possible protection strategy are security extensions for OSNs, which we discussed in Subsection 3.3.1. These extensions enforce access control to sensitive information with encryption. While OSN security extensions conceal sensitive information from OSN providers as well as from attackers, they have a number of shortcomings. First, the protection mechanisms in general focus to protect messages and profile information while e.g. pictures and graph information remains vulnerable to disclosure. Second, none of the proposed security extensions are actively maintained and the majority of proof-of-concept implementations are not operational anymore (see Table 3.2).

The second and more promising protection strategy against our FiTM attack is protection against session-hijacking attacks by social networking providers. In order to effectively mitigate FiTM attacks, OSNs providers have to ultimately ensure that all communication between their users and their platform is performed via TLS. At the time of writing Google$+$ is the only major OSN that supports TLS per default, see Table 3.1. Browser extensions such as *ForceHTTPS* [109] or *EFF HTTPS Everywhere* [61] offer a transitional mitigation strategy to the average user by attempting to force HTTPS for requests that would have been normally transferred over HTTP. The Tor project furthermore announced [27] that they will include a special version of NoScript [145] within the Tor browser bundle which enforces HTTPS for a number of websites including OSNs. OSNs currently refrain from enabling TLS per default for cost and performance reasons and they might also deploy mechanisms that specifically protect against session hijacking attacks. Examples for such a protection approach is *SessionLock* which was proposed by Adida [5]. We furthermore developed *SHPF* a framework to detect session-hijacking attacks [185].

CHAPTER 7

Conclusion

7.1 Summary

Online Social Networks (OSNs) are used by millions of people on a daily basis, to share a plethora of sensitive information online. The personal user information is hereby stored in the walled gardens of social networking providers. In this thesis, we first outlined existing research on OSN security, and privacy, as well as on protection strategies. Previous research focused to a large extend on the information-disclosure behavior of social networking users and privacy management of their profile's content. Hence, the main research problem of this thesis focused on advancing research on security and privacy implications of OSN third-party access, and third-party applications in particular. Third-party applications, or colloquial "apps", are popular extensions to OSNs and their current modus operandi allows them to transfer personal user information out of the walled gardens of OSNs. The first non-trivial research problem we solved, is the automated analysis of OSN application ecosystems, regarding their security and privacy practices. We furthermore showed that custom third-party applications have important implications for the area of information security. We then presented a novel approach to collect digital evidence from OSNs with the help of a customized third-party application. While our results suggest that OSN third-party access enables crucial methods for digital forensics, we also showed that custom third-party applications could be misused for spam campaigns. In particular we demonstrated that session hijacking attacks can be used to inject custom applications into user profiles. Our results suggest that a relatively small number of hijacked user sessions potentially lead to a large number of users being spammed.

Our results helped to mitigate information leaks of popular OSN third-party applications, and thus ultimately to protect the security of users. Furthermore, our novel approach for collecting digital evidence from OSNs made an important contribution to the area of digital forensics. Finally, the findings of this thesis stressed the importance of proper OSN communication security, and social networking providers recently acknowledged our findings. We expect that all major social networking providers make encrypted communication protocols their default in the near future. In the following we are going to summarize our three main results in more detail.

7.1.1 AppInspect

In Chapter 4, we proposed a novel framework called *AppInspect* to automatically analyze security and privacy issues of social network third-party applications. Our analysis software first enumerates available applications for a given social network provider. In the following, our software collects application metrics from the social network provider. In a last step, *AppInspect* installs third-party applications to test accounts and analyzes their network traffic. Hereby, our framework analyses 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. Our approach automatically enumerated *434,687* unique Facebook applications and analyzed the most popular applications in detail. Our results showed that *AppInspect* offers a practical approach to detect common malpractices of third-party applications. Our findings helped to automatically spot information leaks of popular OSN applications and revealed the underlying hosting infrastructure of application developers.

7.1.2 Social Snapshots

In Chapter 5 we introduced *Social Snapshots* and explored novel techniques for automated collection of digital evidence from online social networks. An evaluation of our approach showed that it is a practical and effective method to collect the complete information of a given social networking account. In addition, our approach did not get detected by social networking providers. We believe that our techniques can be used in cases where no legal cooperation with social networking providers exists. In order to provide a digital evidence collection tool for modern forensic investigations of social networking activities, we released our core social snapshot framework as open source software.

7.1.3 Friend-in-The-Middle (FiTM) attacks

In Chapter 6, we introduced *Friend-in-The-Middle (FiTM)* attacks which exploit the partial communication security support of OSNs. It becomes possible to impersonate OSN users and to use the social network on their behalf, by eavesdropping on unprotected social networking communication. We showed that OSNs enable a number of unique attacks based on session hijacking: (1) Friend injection, (2) Application injection, and (3) Social engineering. In the following, we outlined a large-scale spam campaign on the basis of our *FiTM* attacks, whereas custom applications are injected into hijacked user profiles. A number of sources could be used to eavesdrop on social networking sessions and we performed measurements on public WiFi access points and an experimental Tor exit node. Our results suggested that finding possible *FiTM* attack seeds for spam campaigns is cheap regarding time and hardware resources. We furthermore showed on the basis of a simulation that a large-scale spam campaign via *FiTM* attacks would have a severe impact. We discussed the limited protection strategies available against our attack and emphasized that social networking providers have to ultimately protect their users against *FiTM* attacks by securing all user communication with TLS per default.

7.2 Comparison with Related Work

This section discusses our major findings with regard to the state-of-the-art.

7.2.1 Security and Privacy of Application Ecosystems

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. [188] conducted the first measurement study regarding to 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 [10]. 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. [49] 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. [79] 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 [123]. Their observation was confirmed by investigative journalism of the Wall Street Journal, which found that both advertising and tracking products received social network user identifiers [165, 166]. In May 2011, Symantec also found that third-party applications leaked OAuth tokens to third parties [174] 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.

7.2.2 Digital Forensics

Numerous forensic frameworks have been proposed in recent years. However, none of them were designed specifically to extract information from social networks. To the best of our knowledge, no other publication has examined the impact of a hybrid API and crawler based approach to digital forensics in social networks. Datar and Mislan discussed OSN cyber crime but did not propose any methods to analyze these crimes [58]. Even though social networks are not per-se part of the cloud computing paradigm, the area of cloud forensics poses some related challenges as these service operators rely on private clouds. Specifically, the unknown location of data centers [175], and the difficulty to obtain access to forensic data sources without trusting a third party [26], as well as data provenance [130]. To the best of our knowledge there do not exist any tools for extracting complete OSN account information. Pyflag [53] is a modular network forensic framework built to analyze network dumps. Among other features it is able to rebuild HTML pages from packets, allowing the examiner to view the webpages, even if they used AJAX or other dynamic techniques for representation. Xplico [196] is an Internet traffic decoder which can retrieve Facebook chat conversations from network dumps. In relation to our digital image forensics module a recent approach is PhotoDNA [134], which is a program to detect known and explicitly illegal pictures based on calculated signatures. However, it is only available to law enforcement agencies. Similar to signature-based antivirus software, a trusted party calculates the signatures for illicit pictures which in turn is then compared with the signatures of pictures on webpages, data archives, or pictures from forensic hard drive examinations. In [116] characteristics of embedded thumbnails are used to authenticate the source of a picture. While both approaches work similar to our module, they have not been designed or employed to compare digital images from social networks with pictures from a suspect's hard drive.

7.2.3 Impact of Session-Hijacking Attacks

The missing support for communication security was first brought up by Jones and Soltren [112], who discovered that all communication with the early version of Facebook was performed in plain text. We were amongst the first researchers to stress that, due to the popularity of OSNs, unencrypted user session can be observed at virtually any Internet access point [103, 104]. In addition to our publications, the risk of OSN session-hijacking gained public attention with the development and availability of easy-to-use attack tools, such as Firesheep [43], and Faceniff [149]. Our FiTM attack uses session-hijacking to exploit two previously published attacks, namely social phishing [110] and context-aware spam [38]. Jagatic et al. [110] relied on simple webcrawling to gather initial user information. In comparison to their outlined approach, our attack gathers more detailed context information, through injecting third-party applications.

7.3 Future Work and Open Issues

We are currently improving the stability and functionality of our *AppInspect* framework. Our current in-depth application analysis is limited to a small subsample and we therefore plan to expand our analyses, and provide periodically refreshed data sets online[1]. We also plan to release an updated version of our open-source *social snapshot tool*[2], which aims to improve the user-friendliness of our proof of concept implementation. Furthermore, we are working on a storage backend to properly archive social snapshots, and to aid their forensic analysis. Furthermore, we are developing a custom Gephi plugin for visualizing social snapshots.

The current security and privacy shortcomings of popular OSNs result in a number of open research questions. Hence, we plan to conduct further research on secure alternatives to centralized OSNs.

[1]AppInspect datasets `http://ai.nysos.net`

[2]Social Snapshot Tool `https://github.com/markushuber/social-snapshot-tool`

Bibliography

[1] S. Abu-Nimeh, T. Chen, and O. Alzubi. Malicious and spam posts in online social networks. *Computer*, 44(9):23–28, 2011.

[2] A. Acquisti and R. Gross. Imagined Communities: Awareness, Information Sharing, and Privacy on the Facebook. *Lecture Notes in Computer Science*, 4258:36, 2006.

[3] A. Acquisti and R. Gross. Predicting social security numbers from public data. *Proceedings of the National academy of sciences*, 106(27):10975–10980, 2009.

[4] A. Acquisti, R. Gross, and F. Stutzman. Faces of facebook: Privacy in the age of augmented reality. *BlackHat USA*, 2011.

[5] B. Adida. Sessionlock: securing web sessions against eavesdropping. In *Proceedings of the 17th international conference on World Wide Web*, pages 517–524. ACM, 2008.

[6] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Review of Modern Physics*, 74:47–97, 2002.

[7] L. Alvisi, A. Clement, A. Epasto, S. Lattanzi, and A. Panconesi. Sok: The evolution of sybil defense via social networks. *IEEE Symposium on Security and Privacy*, 2013.

[8] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. In *Proceedings Of The National Academy Of Sciences Of The United States Of America*, volume 97, pages 11149–11152. National Acad Sciences, 2000.

[9] J. Anderson, C. Diaz, J. Bonneau, and F. Stajano. Privacy-enabling social networking over untrusted networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 1–6. ACM, 2009.

[10] J. Angwin and J. Singer-Vine. Selling you on facebook. *Wall Street Journal*, 2012. Available online: `http://on.wsj.com/Ii13iA`, last access: 07/05/2013.

[11] M. Backes, M. Maffei, and K. Pecina. A security api for distributed social networks. In *Proc. NDSS*, 2011.

[12] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, pages 181–190. ACM, 2007.

[13] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 135–146. ACM, 2009.

[14] M. Balduzzi, M. Egele, E. Kirda, D. Balzarotti, and C. Kruegel. A solution for the automated detection of clickjacking attacks. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 135–144. ACM, 2010.

[15] M. Balduzzi, C. Platzer, T. Holz, E. Kirda, D. Balzarotti, and C. Kruegel. Abusing social networks for automated user profiling. In *Recent Advances in Intrusion Detection*, pages 422–441. Springer, 2010.

[16] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro. A security analysis of amazon's elastic compute cloud service. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 1427–1434, New York, NY, USA, 2012. ACM.

[17] A. Barabási and E. Bonabeau. Scale-free networks. *Scientific American*, 288:50–59, 2003.

[18] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[19] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, pages 361–362, 2009.

[20] F. Beato, M. Kohlweiss, and K. Wouters. Scramble! your social network data. In *Privacy Enhancing Technologies*, volume 6794, pages 211–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[21] D. Beaver, S. Kumar, H. Li, J. Sobel, and P. Vajgel. Finding a needle in haystack: Facebooks photo storage. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, pages 1–8. USENIX Association, 2010.

[22] A. Besmer, H. Lipford, M. Shehab, and G. Cheek. Social applications: exploring a more secure framework. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, page 2. ACM, 2009.

[23] A. Besmer and H. R. Lipford. Users' (mis)conceptions of social applications. In *Proceedings of Graphics Interface 2010*, GI '10, pages 63–70, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.

[24] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang. The growth of diaspora-a decentralized online social network in the wild. In *Computer Communications Workshops*, pages 13–18. IEEE, 2012.

[25] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the 18th international conference on World wide web*, pages 551–560. ACM New York, NY, USA, 2009.

[26] D. Birk and C. Wegener. Technical issues of forensic investigatinos in cloud computing environments. In *Systematic Approaches to Digital Forensic Engineering, 2011. SADFE 2011. Sixth International Workshop on*. IEEE, 2011.

[27] T. T. Blog. Tor Browser Bundle for GNU/Linux, 2010. Available online: `https://blog.torproject.org/blog/tor-browser-bundle-gnulinux`, last access: 27/05/2013.

[28] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[29] B. Bollobás. *Random Graphs*. Cambridge University Press, 2nd edition, 2001.

[30] J. Bonneau. Security and privacy in social networks bibliography, 2010. Archived website available online: `http://web.archive.org/web/20101125092945/http://www.cl.cam.ac.uk/~jcb82/sns_bib/main.html`.

[31] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano. Eight friends are enough: social graph approximation via public listings. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 13–18. ACM, 2009.

[32] J. Bonneau, J. Anderson, and G. Danezis. Prying data out of a social network. In *Social Network Analysis and Mining, 2009. ASONAM'09. International Conference on Advances in*, pages 249–254. IEEE, 2009.

[33] J. Bonneau and S. Preibusch. The privacy jungle: On the market for data protection in social networks. *Economics of information security and privacy*, pages 121–167, 2010.

[34] S. Borgatti, A. Mehra, D. Brass, and G. Labianca. Network analysis in the social sciences. *science*, 323(5916):892–895, 2009.

[35] Y. Boshmaf, I. Muslukhov, K. Beznosov, and M. Ripeanu. The socialbot network: when bots socialize for fame and money. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 93–102. ACM, 2011.

[36] D. Boyd. Bibliography of research on social network sites. Available online: `http://www.danah.org/researchBibs/sns.php`, last access: 17/02/2013.

[37] D. m. Boyd and N. B. Ellison. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication*, 13(1):210–230, 2007.

[38] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social networks and context-aware spam. In *Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 403–412. ACM New York, NY, USA, 2008.

[39] S. Buchegger and A. Datta. A case for p2p infrastructure for social networks-opportunities & challenges. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pages 161–168. IEEE, 2009.

[40] S. Buchegger, D. Schiöberg, L. Vu, and A. Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52. ACM, 2009.

[41] S. Bugiel, S. Nürnberger, T. Pöppelmann, A.-R. Sadeghi, and T. Schneider. AmazonIA: when elasticity snaps back. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 389–400, New York, NY, USA, 2011. ACM.

[42] E. Bursztein, I. F. Cassidian, and M. Martin. Doing forensics in the cloud age owade: beyond files recovery forensic. *BlackHat USA*, 2011.

[43] E. Butler. Firesheep, 2010. Available online: `http://codebutler.com/firesheep/`, last access: 27/05/2013.

[44] M. Caloyannides, N. Memon, and W. Venema. Digital forensics. *Security & Privacy, IEEE*, 7(2):16–17, 2009.

[45] A. Campan and T. Truta. Data and structural k-anonymity in social networks. *Privacy, Security, and Trust in KDD*, pages 33–54, 2009.

[46] B. Carrier. *File system forensic analysis*. Addison-Wesley Professional, 2005.

[47] C. Castelluccia, E. De Cristofaro, and D. Perito. Private information disclosure from web searches. In *Privacy Enhancing Technologies*, pages 38–55. Springer, 2010.

[48] E. Chan, S. Venkataraman, F. David, A. Chaugule, and R. Campbell. Forenscope: A framework for live forensics. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 307–316. ACM, 2010.

[49] P. H. Chia, Y. Yamamoto, and N. Asokan. 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*, WWW '12, pages 311–320, New York, NY, USA, 2012. ACM.

[50] Cisco. Annual security report, 2012. Available online: `http://www.cisco.com/en/US/prod/collateral/vpndevc/security_annual_report_2011.pdf`, last access 28/04/2013.

[51] cnet. Facebook launches bug bounty program, 2011. Available online: `http://news.cnet.com/8301-27080_3-20085163-245/facebook-launches-bug-bounty-program/`, last access 05/27/2013.

[52] CNN. Facebook status update provides alibi, 2009. Available online: `http://cnn.com/2009/CRIME/11/12/facebook.alibi/index.html`, last access: 05/27/2013.

[53] M. Cohen. PyFlag-An advanced network forensic framework. *Digital Investigation: The International Journal of Digital Forensics & Incident Response*, 5:S112–S120, 2008.

[54] F. T. Commision. In the matter of facebook, inc., a corporation, Aug 2012. Available online: `http://www.ftc.gov/os/caselist/0923184/120810facebookcmpt.pdf`, last access 10/07/2012.

[55] L. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pages 145–152. IEEE, 2009.

[56] L. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12):94–101, 2009.

[57] G. Danezis. Inferring privacy policies for social networking services. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, pages 5–10. ACM, 2009.

[58] T. D. Datar and R. Mislan. Social networking: a boon to criminals. *Journal of Network Forensics*, page 14, 2010.

[59] X. Ding and H. Zou. Time based data forensic and cross-reference analysis. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 185–190. ACM, 2011.

[60] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, San Diego, CA, USA, August 2004.

[61] EFF. HTTPS Everywhere. Available online: `https://www.eff.org/https-everywhere`, last accessed: 05/27/2013.

[62] EFF. Social Media and Law Enforcement: Who Gets What Data and When? Online at https://www.eff.org/deeplinks/2011/01/social-media-and-law-enforcement-who-gets-what.

[63] M. Egele, A. Moser, C. Kruegel, and E. Kirda. Pox: Protecting users from malicious facebook applications. *Computer Communications*, 2012.

[64] P. Erdos and A. Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[65] eWeek. Facebook smart lists one-up google circles, 2011. Available online: `http://www.eweek.com/c/a/Messaging-and-Collaboration/Facebook-Smart-Lists-One-Ups-Google-Circles-596726/`, last access: 05/11/2013.

[66] Facebook. Getting your apps into facebook search faster. Available online: `https://developers.facebook.com/blog/post/2011/07/12/getting-your-apps-into-facebook-search-faster/`, last access 04/27/2012.

[67] Facebook. Graph API. Available online: `https://developers.facebook.com/docs/reference/api/`, last access 11/21/2012.

[68] Facebook. Permissions reference. Available online: `https://developers.facebook.com/docs/authentication/permissions/`, last access 08/10/2012.

[69] Facebook. Welcome to Facebook, everyone, 2006. Available online: `https://blog.facebook.com/blog.php?post=2210227130`, last access 05/25/2013.

[70] Facebook. Facebook platform launches, May 2007. Available online: `https://developers.facebook.com/blog/archive#2007`, last access 08/15/2012.

[71] Facebook. The Facebook Blog: Giving You More Control. Available online: `https://blog.facebook.com/blog.php?post=434691727130`, last access 11/21/2012, oct 2010.

[72] Facebook. A continued commitment to security, jan 2011. Available online: `http://blog.facebook.com/blog.php?post=486790652130`, last access 05/07/2013.

[73] Facebook. Platform updates: Operation developer love, nov 2012. Available online: `https://developers.facebook.com/blog/post/2012/11/14/platform-updates--operation-developer-love/`, last access 05/07/2013.

[74] L. Fang and K. LeFevre. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World wide web*, pages 351–360. ACM, 2010.

[75] A. J. Feldman, A. Blankstein, M. J. Freedman, and E. W. Felten. Social networking with frientegrity: privacy and integrity with an untrusted provider. In *Proceedings of the 21st USENIX conference on Security symposium*, pages 31–31. USENIX Association, 2012.

[76] A. Felt and D. Evans. Privacy protection for social networking APIs. In *W2SP '08*, 2008.

[77] Forbes. Facebook investigating how bulgarian man bought 1.1 million users' email addresses for five dollars. Available online: `http://www.forbes.com/sites/andygreenberg/2012/10/25/facebook-investigating-how-bulgarian-man-bought-1-1-million-users-email-addresses-for-five-dollars/`, last access 11/03/2012.

[78] K. Fowler. *SQL Server forensic analysis*. Addison-Wesley Professional, 2008.

[79] M. Frank, B. Dong, A. P. Felt, and D. Song. Mining permission request patterns from android and facebook applications. *To appear: IEEE International Conference on Data Mining (ICDM) 2012*, 2012.

[80] L. C. Freeman. *The Development of Social Network Analysis: A Study in the Sociology of Science*. Empirical Press, 2004.

[81] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[82] FSF. Ocrad - The GNU OCR. Online at `http://www.gnu.org/software/ocrad/`.

[83] H. Gao, J. Hu, T. Huang, J. Wang, and Y. Chen. Security issues in online social networks. *Internet Computing, IEEE*, 15(4):56–63, 2011.

[84] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao. Detecting and characterizing social spam campaigns. In *Proceedings of the 10th annual conference on Internet measurement*, pages 35–47. ACM, 2010.

[85] S. L. Garfinkel. Digital media triage with bulk data analysis and bulk_extractor. *Computers & Security*, 2012.

[86] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[87] Google. Google launches opensocial to spread social applications across the web, Nov 2007. Available online: `http://googlepress.blogspot.com/2007/11/google-launches-opensocial-to-spread_01.html`, last access 04/05/2012.

[88] R. Gross and A. Acquisti. Information revelation and privacy in online social networks (the Facebook case). In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[89] S. Guha, K. Tang, and P. Francis. Noyb: Privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, volume 1, pages 49–54. ACM, 2008.

[90] M. S. Handcock, A. R. Raftery, and J. M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society*, 170:301–354, 2007.

[91] D. Hardt. The OAuth 2.0 authorization framework. Available online: `http://tools.ietf.org/html/draft-ietf-oauth-v2-31`, last access 02/05/2012.

[92] M. Häsel. Opensocial: an enabler for social applications on the web. *Communications of the ACM*, 54(1):139–144, 2011.

[93] B. Hay, K. Nance, and M. Bishop. Live analysis: Progress and challenges. *Security & Privacy, IEEE*, 7(2):30–37, 2009.

[94] X. He. A Performance Analysis of Secure HTTP Protocol. *STAR Lab Technical Report, Department of Electrical and Computer Engineering, Tennessee Tech University*, 2003.

[95] heise. StudiVZ-Nutzerdaten ausgespaeht, 2007. Available online: `http://www.heise.de/newsticker/meldung/StudiVZ-Nutzerdaten-ausgespaeht-150950.html`, last access 05/20/2013.

[96] C. Herley and D. Florêncio. A profitless endeavor: phishing as tragedy of the commons. In *NSPW '08: Proceedings of the 2008 workshop on New security paradigms*, pages 59–70, New York, NY, USA, 2008. ACM.

[97] P. Heymann, G. Koutrika, and H. Garcia-Molina. Fighting spam on social web sites: A survey of approaches and future challenges. *Internet Computing, IEEE*, 11(6):36–45, 2007.

[98] G. Hogben. Security Issues and Recommendations for Online Social Networks. *Position Paper. ENISA, European Network and Information Security Agency*, 2007.

[99] T. Holz, C. Gorecki, K. Rieck, and F. Freiling. Measuring and detecting fast-flux service networks. In *Symposium on Network and Distributed System Security*. Citeseer, 2008.

[100] M. Huber, S. Kowalski, M. Nohlberg, and S. Tjoa. Towards automating social engineering using social networking sites. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 117–124. IEEE, 2009.

[101] M. Huber, M. Mulazzani, M. Leithner, S. Schrittwieser, G. Wondracek, and E. Weippl. Social snapshots: digital forensics for online social networks. In *Proceedings of the 27th Annual Computer Security Applications Conference*, 2011.

[102] M. Huber, M. Mulazzani, S. Schrittwieser, and E. Weippl. Appinspect: Large-scale evaluation of social networking apps. In *Proceedings of ACM conference on Online Social Networks (COSN)*, 2013.

[103] M. Huber, M. Mulazzani, and E. Weippl. Tor http usage and information leakage. In *Communications and Multimedia Security*, pages 245–255. Springer, 2010.

[104] M. Huber, M. Mulazzani, and E. Weippl. Who on earth is mr. cypher? automated friend injection attacks on social networking sites. In *Proceedings of the IFIP International Information Security Conference 2010: Security and Privacy*, 9 2010.

[105] M. Huber, M. Mulazzani, E. Weippl, G. Kitzler, and S. Goluch. Friend-in-the-middle attacks: Exploiting social networking sites for spam. *IEEE Internet Computing*, 15(3):28–34, 2011.

[106] G. Hull, H. Lipford, and C. Latulipe. Contextual gaps: privacy issues on facebook. *Ethics and information technology*, 13(4):289–302, 2011.

[107] D. Irani, M. Balduzzi, D. Balzarotti, E. Kirda, and C. Pu. Reverse social engineering attacks in online social networks. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 55–74, 2011.

[108] D. Irani, S. Webb, K. Li, and C. Pu. Large online social footprints–an emerging threat. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 271–276. IEEE, 2009.

[109] C. Jackson and A. Barth. ForceHTTPS: Protecting high-security web sites from network attacks. 2008.

[110] T. Jagatic, N. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[111] S. Jansen, T. Luczak, and A. Rucinski. *Random Graphs*. John Wiley & sons, Inc., 2000.

[112] H. Jones and J. Soltren. Facebook: Threats to Privacy. *Project MAC: MIT Project on Mathematics and Computing*, 2005.

[113] S. Jones and E. O'Neill. Feasibility of structural network clustering for group-based privacy control in social networks. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, SOUPS '10, pages 9:1–9:13, New York, NY, USA, 2010. ACM.

[114] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. Voelker, V. Paxson, and S. Savage. Spamalytics: An empirical analysis of spam marketing conversion. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 3–14. ACM New York, NY, USA, 2008.

[115] Kaspersky. Spam evolution 2012, 2013. Available online: `https://www.securelist.com/en/analysis/204792276/Kaspersky_Security_Bulletin_Spam_Evolution_2012`, last access 04/25/2013.

[116] E. Kee and H. Farid. Digital image authentication from thumbnails. *Proceedings of the SPIE, Electronic Imaging, Media Forensics and Security XII*, 2010.

[117] H. Kim, J. Tang, and R. Anderson. Social authentication: Harder than it looks. *Financial Cryptography and Data Security*, pages 1–15, 2012.

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

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

[120] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu. Link privacy in social networks. In *Proceeding of the 17th ACM conference on Information and knowledge management*, pages 289–298. ACM New York, NY, USA, 2008.

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

[122] B. Krishnamurthy and C. Wills. Characterizing privacy in online social networks. In *Proceedings of the first workshop on Online social networks*, pages 37–42. ACM, 2008.

[123] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 7–12, New York, NY, USA, 2009. ACM.

[124] B. Krishnamurthy and C. E. Wills. Privacy leakage in mobile online social networks. In *Proceedings of the 3rd conference on Online social networks*, pages 4–4. USENIX Association, 2010.

[125] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots+ machine learning. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM, 2010.

[126] K. Lewis, J. Kaufman, M. Gonzalez, A. Wimmer, and N. Christakis. Tastes, ties, and time: A new social network dataset using facebook. com. *Social Networks*, 30(4):330–342, 2008.

[127] LinkedIn. A more secure LinkedIn browsing experience, feb 2012. Available online: `http://blog.linkedin.com/2012/02/07/linkedin-ssl/`, last access 05/07/2013.

[128] H. Lipford, A. Besmer, and J. Watson. Understanding privacy settings in facebook with an audience view. In *Proceedings of the 1st Conference on Usability, Psychology, and Security*, pages 1–8. USENIX Association Berkeley, CA, USA, 2008.

[129] S. Livingstone. Taking risky opportunities in youthful content creation: teenagers' use of social networking sites for intimacy, privacy and self-expression. *New Media & Society*, 10(3):393–411, 2008.

[130] R. Lu, X. Lin, X. Liang, and X. Shen. Secure provenance: the essential of bread and butter of data forensics in cloud computing. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 282–292. ACM, 2010.

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

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

[133] M. Marlinspike. New tricks for defeating ssl in practice. *BlackHat DC, February*, 2009.

[134] Microsoft. PhotoDNA. Online at `http://www.microsoftphotodna.com/`.

[135] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[136] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–134. IEEE, 2003.

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

[138] M. Mulazzani, M. Huber, and E. Weippl. Social network forensics: Tapping the data pool of social networks. *Eighth Annual IFIP WG 11.9 International Conference on Digital Forensics*, 1 2012.

[139] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE, 2009.

[140] M. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[141] M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351, September 2005.

[142] M. Newman. *Networks: an introduction*. OUP Oxford, 2009.

[143] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[144] M. E. J. Newman. Models of the small world. *Journal of Statistical Physics*, 101(3-4):819–841, November 2000.

[145] Noscript browser extension. `http://noscript.net/`.

[146] OpenQA. Selenium wep application testing system. Online at `http://seleniumhq.org/`.

[147] M. Perry. CookieMonster: Cookie Hijacking. Online at `http://fscked.org/projects/cookiemonster`, aug 2008.

[148] I. Polakis, M. Lancini, G. Kontaxis, F. Maggi, S. Ioannidis, A. D. Keromytis, and S. Zanero. All your face are belong to us: breaking facebook's social authentication. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 399–408. ACM, 2012.

[149] B. Ponurkiewicz. Faceniff. Online at `http://faceniff.ponury.net/`, jun 2011.

[150] M. S. Rahman, T.-K. Huang, H. V. Madhyastha, and M. Faloutsos. Efficient and scalable socware detection in online social networks. In *Proceedings of the 21st USENIX conference on Security symposium*, pages 32–32. USENIX Association, 2012.

[151] N. A. Rahman. Scraping facebook email addresses. Online at http://www.kudanai.com/2008/10/scraping-facebook-email-addresses.html, aug 2008.

[152] A. Rapoport. Contribution to the theory of random and biased nets. *Bulletin of Mathematical Biophysics*, 19:257–277, 1957.

[153] ReadWrite. Likejacking takes off on facebook. http://readwrite.com/2010/06/01/likejacking_takes_off_on_facebook, 2010.

[154] T. Ryan and G. Mauch. Getting in bed with robin sage. In *Black Hat Conference*, 2010.

[155] G. Rydstedt, E. Bursztein, D. Boneh, and C. Jackson. Busting frame busting: a study of clickjacking vulnerabilities at popular sites. *IEEE Oakland Web*, 2, 2010.

[156] J. Schrammel, C. Köffel, and M. Tscheligi. How much do you tell?: information disclosure behaviour indifferent types of online communities. In *Proceedings of the fourth international conference on Communities and technologies*, pages 275–284. ACM, 2009.

[157] 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.

[158] S.-W. Seong, J. Seo, M. Nasielski, D. Sengupta, S. Hangal, S. K. Teh, R. Chu, B. Dodson, and M. S. Lam. Prpl: a decentralized social networking infrastructure. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 8. ACM, 2010.

[159] A. Shakimov, H. Lim, R. Cáceres, L. P. Cox, K. Li, D. Liu, and A. Varshavsky. Vis-a-vis: Privacy-preserving online social networking via virtual individual servers. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–10. IEEE, 2011.

[160] A. Shakimov, A. Varshavsky, L. Cox, and R. Cáceres. Privacy, cost, and availability tradeoffs in decentralized osns. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 13–18. ACM, 2009.

[161] S. C. Shiong. Facebook bug 4: Password reset vulnerability found in www.facebook.com. http://chingshiong.blogspot.co.uk/2013/01/facebook-bug-4-password-reset.html, 2013.

[162] K. Singh, S. Bhola, and W. Lee. xBook: redesigning privacy control in social networking platforms. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, page 249, Berkeley, CA, USA, 2009. USENIX Association.

[163] Sophos. Sophos facebook id probe shows 41% of users happy to reveal all to potential identity thieves, 2007. http://www.sophos.com/en-us/press-office/press-releases/2007/08/facebook.aspx.

[164] A. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *Proceedings of the 18th international conference on World wide web*, pages 521–530. ACM, 2009.

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

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

[167] T. Stein, E. Chen, and K. Mangla. Facebook immune system. In *Proceedings of the 4th Workshop on Social Network Systems*, SNS '11, pages 8:1–8:8, New York, NY, USA, 2011. ACM.

[168] H. Stern. A survey of modern spam tools. In *Proceedings of the 5th Conference on Email and Anti-Spam*. Citeseer, 2008.

[169] K. Strater and H. Lipford. Strategies and struggles with privacy in an online social networking community. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction-Volume 1*, pages 111–119. British Computer Society, 2008.

[170] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 1–9. ACM, 2010.

[171] F. Stutzman. An evaluation of identity-sharing behavior in social network communities. *International Digital and Media Arts Journal*, 3(1):10–18, 2006.

[172] F. Stutzman, R. Gross, and A. Acquisti. Silent listeners: The evolution of privacy and disclosure on facebook. *Journal of Privacy and Confidentiality*, 4(2):2, 2013.

[173] J. Sun, X. Zhu, and Y. Fang. A privacy-preserving scheme for online social networks with efficient revocation. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[174] Symantec. Facebook applications accidentally leaking access to third parties. last accessed 06/20/2012.

[175] M. Taylor, J. Haggerty, D. Gresty, and D. Lamb. Forensic investigation of cloud computing systems. *Network Security*, 2011(3):4–10, 2011.

[176] TechCrunch. 5 design tricks facebook uses to affect your privacy decisions. Available online: `http://techcrunch.com/2012/08/25/5-design-tricks-facebook-uses-to-affect-your-privacy-decisions/`, last access 09/21/2012.

[177] TechCrunch. LinkedIn confirms hack and leak of some user passwords. Available online: `http://techcrunch.com/2012/06/06/linkedin-speaks-some-of-those-compromised-passwords-are-from-linkedin-accounts/`, last access: 07/20/2013.

[178] The New York Criminal Law Blog. Criminal found via Facebook. Online at `http://newyorkcriminallawyersblog.com/2010/03/assault-criminal-who-was-found-via-facebook-is-back-in-ny.html`, mar 2009.

[179] The Washington Post. Facebook: a place to meet, gossip, share photos of stolen goods. Online at `http://www.washingtonpost.com/wp-dyn/content/article/2010/12/14/AR2010121407423_pf.html`, dec 2010.

[180] K. Thomas and D. M. Nicol. The koobface botnet and the rise of social malware. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 63–70. IEEE, 2010.

[181] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: better privacy for social networks. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 169–180. ACM, 2009.

[182] Tor: anonymity online. `https://www.torproject.org/`.

[183] Trend Micro. Captcha wish your girlfriend was hot like me, oct 2007. `http://blog.trendmicro.com/trendlabs-security-intelligence/captcha-wish-your-girlfriend-was-hot-like-me/`.

[184] Z. Tufekci. Can you see me now? audience and disclosure regulation in online social network sites. *Bulletin of Science, Technology & Society*, 28(1):20–36, 2008.

[185] T. Unger, M. Mulazzani, D. Frühwirt, M. Huber, S. Schrittwieser, and E. Weippl. Shpf: Enhancing http(s) session security with browser fingerprinting. In *Eigth International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2013.

[186] R. van der Hofstad. *Random Graphs and Complex Networks*. Lecture Notes, Departement of Mathematics and Computer Science, Eindhofen University of Technology, 2009.

[187] L. Von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. *Lecture notes in computer science*, pages 294–311, 2003.

[188] N. Wang, H. Xu, and J. Grossklags. 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*, page 4. ACM, 2011.

[189] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[190] S. Webb, J. Caverlee, and C. Pu. Social honeypots: Making friends with a spammer near you. In *Proceedings of the Fifth Conference on Email and Anti-Spam (CEAS 2008), Mountain View, CA*, 2008.

[191] C. Wilson, B. Boe, A. Sala, K. Puttaswamy, and B. Y. Zhoa. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer system*, pages 205–218. ACM New York, NY, USA, 2009.

[192] R. Wilson, S. Gosling, and L. Graham. A review of facebook research in the social sciences. *Perspectives on Psychological Science*, 7(3):203–220, 2012.

[193] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A Practical Attack to De-Anonymize Social Network Users. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2010.

[194] M. Wong and W. Schlitt. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail,. RFC 4408, Apr. 2006.

[195] N. World. Google's orkut battered by 'Bom sabado' worm, 2010. Available online: `https://www.networkworld.com/news/2010/092710-googles-orkut-battered-by-bom.html`, last access 05/27/2013.

[196] Xplico. Xplico - Network Forensic Analysis Tool. Online at `http://www.xplico.org/`.

[197] ZDNet. Samy opens new front in worm war, 2005. Available online: `http://www.zdnet.com/news/samy-opens-new-front-in-worm-war/145114`, last access 05/27/2013.

[198] C. Zhang, J. Sun, X. Zhu, and Y. Fang. Privacy and security for online social networks: challenges and opportunities. *Network, IEEE*, 24(4):13–18, 2010.

[199] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540. ACM, 2009.

[200] B. Zhou, J. Pei, and W. Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM SIGKDD Explorations Newsletter*, 10(2):12–22, 2008.

[201] A. Zinman and J. Donath. Is britney spears spam. In *Fourth Conference on Email and Anti-Spam, Mountain View, CA*, 2007.

[202] L. Zou, L. Chen, and M. Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.

List of Figures

List of Tables