Battling Spam and Sybils on the Social Web

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

Doctor of Philosophy

in

Computer Science

by

Christo Wilson

Committee in Charge:

Professor Ben Y. Zhao, Chair
Professor Christopher Kruegel
Professor Amr El Abbadi

September 2012
The Dissertation of
Christo Wilson is approved:

____________________________________
Professor Christopher Kruegel

____________________________________
Professor Amr El Abbadi

____________________________________
Professor Ben Y. Zhao, Committee Chairperson

August 2012
Battling Spam and Sybils on the Social Web

Copyright © 2012

by

Christo Wilson
To James Broder, for leaving huge footprints
that led me to where I am. You joked that
Teaching me TCP/IP would change my life: it
did.

And to Sally Wilson, for the years of
unconditional support while I corrupted
operating systems, infected computers with
viruses, and set hard drives on fire.
Acknowledgements

The author gratefully acknowledges the tireless work of his many collaborators, without whom none of this research would have been possible. First and foremost, to the members of the Current Lab at UCSB: Alessandra Sala, Krishna Puttaswamy, Gang Wang, Xiaohan Zhao, Troy Steinbauer, Adelbert Chang, and Manish Mohanlal. Second, to the employees at Microsoft Research: Thomas Karagiannis, Hitesh Ballani, Ant Rowstron, Cheng Huang, Sudipta Sengupta, and Jin Li. Third, to Miriam Metzger and Rebekah Pure in the Department of Communications at UCSB. Fourth, to our collaborators at Peking University: Jing Jiang, Zhi Yang, and Yafei Dai. Special thanks to Xiao Wang, for being our advocate at Renren Inc. Finally, to Ben Y. Zhao, who will always be “the boss.”
Curriculum Vitæ

Christo Wilson

Education

2012      Ph.D in Computer Science, University of California, Santa Barbara.
2007      Master of Science in Computer Science, University of California, Santa Barbara.
2006      Bachelor of Science in Creative Studies (Computer Science Emphasis), University of California, Santa Barbara.

Field of Study

Major Field    Computer Science with Professor Ben Y. Zhao.

Professional Employment

2008-2012  Graduate Research Assistant, University of California, Santa Barbara.
Summer 2011 Research Internship, Microsoft Research, Redmond, WA.
Summer 2010 Research Internship, Microsoft Research, Cambridge, UK.
2005-2010  Lead Software Engineer and Principal Investigator, Anacapa Sciences, Inc., Santa Barbara, CA.

Publications


Abstract
Battling Spam and Sybils on the Social Web
Christo Wilson

Since 2004, the social web has become a dominant force on the Internet. As of 2011, 65% of adults in the US used online social networking (OSN) sites, and this number continues to grow, both in the US and around the world. However, as OSNs gradually supplant email and instant messaging as the primary channel for online communication, the incentive for malicious users to attack these systems grows. Social spam and fake Sybil accounts are now the primary tools for online criminals looking to spread malware and steal personal information on OSNs. In this work, we take the first steps towards measuring, understanding, and defending against these threats to the social web.

We begin by conducting detailed studies of two of the largest OSNs in the world: Facebook and Renren. Quantifying the basic graph structural properties of these OSNs gives us a solid foundation of understanding on which to build further research. Our work goes beyond existing studies that are focused on static topologies by accounting for the relative importance of individual edges of the social graph. By analyzing visible and latent interactions between users, we show that all edges in social graphs are not equally important, and develop “interaction graphs” to capture these effects. Through simulations on real social graphs, we show that edge importance has a large effect on
the performance of social applications. This result indicates that ongoing research into social applications and algorithms should take user interactions into account if they hope to obtain realistic and accurate results.

Our baseline OSN measurements allow us to characterize the behavior of normal users in great detail, which opens a window of opportunity for identifying anomalies associated with malicious activity. As a first step towards understanding malicious activity on OSNs, we examine its most prominent outward symptom: spam. We analyze hundreds of millions of wall posts received by millions of Facebook users and develop a novel set of automated techniques to detect social spam. Our results show that a significant portion of the URLs shared on Facebook are spam, the majority of which link to malicious phishing websites. These spam attacks are organized into large, coordinated campaigns by criminals working behind the scenes. Analysis of the behavior of spamming accounts demonstrates that both fake, Sybil accounts and compromised normal accounts are used as tools to attack Facebook users.

Next, we turn our attention to the problem of Sybil accounts. Although our work on spam detection identified Sybils as a major threat to OSNs, at the time no practical solutions to this problem had been developed. To address this challenge, we use ground truth data provided by Renren Inc. to build a measurement based Sybil detector. This system is currently deployed on the Renren OSN, and to date it has caught and banned millions of Sybils. Importantly, our detector operates in real-time, meaning that Sybils
are banned before they get a chance to generate harmful spam. We study the edge creation behavior of Sybils on Renren, and find that contrary to prior conjecture, they do not form tight-knit communities. Instead, they integrate into the social graph just like normal users. This result confirms our hypothesis that existing Sybil community detectors from the literature are unlikely to succeed on today’s OSNs.

In summary, our research makes two fundamental contributions to the study of OSNs. First, our work demonstrates the necessity of measurement driven design of social systems. Repeatedly, our measurements have contradicted assumptions from prior work, and thus revealed new avenues of research. Second, we have discovered, quantified, and developed practical solutions for pressing OSN security problems. However, the social web continues to evolve, and the shape of its attack surface is constantly changing. Attackers will continue to innovate new and unexpected strategies to exploit OSNs and evade security mechanisms. Only by bringing all of our tools to bear: measurement, graph analysis, data mining, machine learning, etc., can computer scientists hope to defend against future threats to the social web.

Professor Ben Y. Zhao
Dissertation Committee Chair
Contents

Acknowledgements v
Curriculum Vitae vi
Abstract xi
List of Figures xviii
List of Tables xxii

1 Introduction 1
  1.1 Social Graphs and User Interactions 7
  1.2 Social Spam 11
  1.3 Social Sybils 12
  1.4 Summary 15

2 Data Collection and Methodology 16
  2.1 The Facebook Social Network 19
  2.1.1 Facebook Fundamentals 19
  2.1.2 Our Relationship with Facebook 21
  2.2 Facebook Dataset and Collection Methodology 25
  2.2.1 Data Collection Process 25
  2.2.2 Description of Collected Data, Limitations, and Privacy 27
  2.3 The Renren Social Network 31
  2.3.1 Renren Fundamentals 31
  2.3.2 Our Relationship with Renren 34
  2.4 Renren Dataset and Collection Methodology 35
  2.4.1 Data Collection Process 35
3 Social Graphs and User Interactions

3.1 Introduction

3.2 Analysis of Social Graphs

3.2.1 Definitions

3.2.2 High Level Statistics

3.2.3 Degree Distributions

3.2.4 Average Path Lengths

3.2.5 Clustering Coefficient

3.2.6 $k_{nn}$ and Assortativity

3.2.7 Network Core Analysis

3.2.8 Network Growth Over Time

3.3 Visible Interactions on Facebook

3.3.1 Visible Interaction Distribution Among Friends

3.3.2 Distribution of Visible Interactions Across Users

3.3.3 Visible Interaction Distribution Across User Lifetimes

3.3.4 Reciprocity

3.4 Latent Interactions on Renren

3.4.1 Popularity and Consumption

3.4.2 Composition of Visitors

3.4.3 Visits to Strangers’ Profiles

3.4.4 Reciprocity

3.4.5 Latent vs. Visible Interactions

3.5 Factors Influencing Latent Interactions

3.5.1 Dataset and Methodology

3.5.2 User Account Characteristics

3.5.3 Diary Entries

3.5.4 Photos, Status Updates, and Comments

3.6 Interaction Graphs

3.6.1 Definition of Interaction Graphs

3.6.2 Visible Interaction Graphs on Facebook

3.6.3 Latent Interaction Graphs on Renren

3.6.4 Comparing Visible and Latent Interaction Graphs Over Time

3.7 Applying Interaction Graphs

3.7.1 RE: Reliable Email

3.7.2 SybilGuard

3.7.3 Influence Maximization

3.8 Facebook Over Time
3.8.1 Description of Collected Data and Methodology .............. 125
3.8.2 Social Graph Analysis .................................. 127
3.8.3 Visible Interaction Analysis .................................. 130
3.9 Summary of Results ........................................... 132

4 Social Spam .................................................. 135
  4.1 Introduction .................................................. 135
  4.2 Dataset and Scope ........................................... 138
    4.2.1 Facebook Dataset ...................................... 138
    4.2.2 Scope of This Work ..................................... 139
  4.3 Spam Campaign Detection .................................... 140
    4.3.1 Overview ................................................ 140
    4.3.2 Analyzing and Grouping Wall Posts ....................... 143
    4.3.3 Identifying Spam Groups ................................ 146
    4.3.4 Detection Results ....................................... 149
  4.4 Experimental Validation .................................... 150
    4.4.1 Validation Methodology .................................. 150
    4.4.2 Validation Results ..................................... 155
    4.4.3 Burst and Distribution Thresholds ....................... 156
    4.4.4 False Negative Rate Estimation ......................... 157
  4.5 Analysis of Spam Activity .................................. 158
    4.5.1 URL Characteristics .................................... 159
    4.5.2 Spam Campaign Analysis ................................ 162
    4.5.3 Malicious Account Analysis .............................. 167
    4.5.4 Temporal Properties of Malicious Activity .............. 173
  4.6 Summary of Results ....................................... 176

5 Social Sybils .................................................. 178
  5.1 Introduction ................................................. 178
  5.2 Detecting Sybils ............................................. 181
    5.2.1 Sybils on Renren ....................................... 181
    5.2.2 Characterizing Sybil Accounts ............................. 183
    5.2.3 Building and Running a Sybil Detector .................... 187
  5.3 Sybil Topology .............................................. 190
    5.3.1 Sybil Edges .............................................. 190
    5.3.2 Sybil Communities ....................................... 192
    5.3.3 Sybil Edge Formation .................................... 194
  5.4 Spam Strategies and Collusion ................................ 198
    5.4.1 Share Spam on Renren .................................... 198
    5.4.2 Case Study: Spam Blogs .................................. 200
List of Figures

2.1 Integrating multiple visitor lists captured by multiple crawls of the same profile into a single history. ................................. 40
2.2 Average daily visit counts of user profiles. ......................... 41
2.3 Number of visits missed when we lower crawler frequency from a high of once every 15 minutes. ................................. 41

3.1 Comparing the degree distribution of Facebook and Renren to Orkut, YouTube and LiveJournal [128]. Both CDF and CCDF distributions are shown. 49
3.2 Clustering coefficient of Facebook users as a function of social degree. 53
3.3 $k_{in}$ of Facebook users as a function of social degree. .............. 53
3.4 Percentage of nodes remaining in the connected component as super-nodes are removed from various social graphs. ...................... 58
3.5 Social degree and profile views for isolated users. ................. 58
3.6 The growth of users in our sample set, starting in September 2004. 59
3.7 Population growth of the PKU network over time. ................. 59
3.8 The distribution of users’ interaction among their friends, for different % of users’ interactions. ........................................ 61
3.9 Normalized Wall post distribution of the users with top total Wall interaction. ...................................................... 62
3.10 Normalized photo comments distribution of the users with top total photo interaction. ............................................. 62
3.11 The contribution of different users to total interactions in Facebook. 63
3.12 Plot of top % of users ordered by social degree and the interaction contributed by them. .............................................. 63
3.13 Average number of interactions per day for old and new Facebook users. 65
3.14 Deviations in pairwise interaction patterns on Facebook. .......... 65
3.15 CDF of user profile popularity defined by visits. ................. 69
3.16 Average number of visits per day per user. ....................... 69
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.17</td>
<td>Overlap between users sorted by popularity vs. sorted by consumption.</td>
</tr>
<tr>
<td>3.18</td>
<td>Ratio of repeat visitors.</td>
</tr>
<tr>
<td>3.19</td>
<td>PDF of interval time between repeat visits.</td>
</tr>
<tr>
<td>3.20</td>
<td>Percentage of strangers in visitors.</td>
</tr>
<tr>
<td>3.21</td>
<td>Breakdown of visitors by owner’s social degree.</td>
</tr>
<tr>
<td>3.22</td>
<td>Average number of visits for friends and strangers.</td>
</tr>
<tr>
<td>3.23</td>
<td>Percentage of visits to strangers’ profiles through social links, as a fraction of total visits to stranger’s profiles.</td>
</tr>
<tr>
<td>3.24</td>
<td>Number of friends browsed before visiting a stranger’s profile.</td>
</tr>
<tr>
<td>3.25</td>
<td>Ratio of reciprocated latent interactions over total latent relationships.</td>
</tr>
<tr>
<td>3.26</td>
<td>Probability of reciprocated profile views over various time windows.</td>
</tr>
<tr>
<td>3.27</td>
<td>Distribution of interactions, with users ordered from most to least interaction.</td>
</tr>
<tr>
<td>3.28</td>
<td>Distribution of interactions, with users ordered by social degree.</td>
</tr>
<tr>
<td>3.29</td>
<td>Distribution of interactions among each user’s friends.</td>
</tr>
<tr>
<td>3.30</td>
<td>Average number of comments per page view for different types of pages.</td>
</tr>
<tr>
<td>3.31</td>
<td>Average number of interactions per profile.</td>
</tr>
<tr>
<td>3.32</td>
<td>Percentage of nodes remaining in interaction graph connected components as $n$ and $t$ vary.</td>
</tr>
<tr>
<td>3.33</td>
<td>Comparison of the Facebook social graph degree and visible interaction graph degree.</td>
</tr>
<tr>
<td>3.34</td>
<td>Comparison of degree distributions for visible interaction graphs and the full social graph. Both CDF and CCDF distributions are shown.</td>
</tr>
<tr>
<td>3.35</td>
<td>Graph measurements for four visible interaction graphs compared to the entire Facebook social network.</td>
</tr>
<tr>
<td>3.36</td>
<td>Clustering coefficient of visible interaction graphs as a function of interaction degree.</td>
</tr>
<tr>
<td>3.37</td>
<td>Percent of edges cut in different visible interaction graphs versus the original social graph.</td>
</tr>
<tr>
<td>3.38</td>
<td>CCDF of node degree for latent interaction graph, visible interaction graph, and social graph.</td>
</tr>
<tr>
<td>3.39</td>
<td>Clustering coefficient distribution for different graph types.</td>
</tr>
<tr>
<td>3.40</td>
<td>$k$-core analysis of different graph types.</td>
</tr>
<tr>
<td>3.41</td>
<td>Interactions per month in the Renren PKU network.</td>
</tr>
<tr>
<td>3.42</td>
<td>Nodes and edges per interaction graph snapshot.</td>
</tr>
<tr>
<td>3.43</td>
<td>Resemblance of evolving visible and latent interaction graphs.</td>
</tr>
<tr>
<td>3.44</td>
<td>Structural properties over time for visible and latent interaction graphs.</td>
</tr>
<tr>
<td>3.45</td>
<td>Friends of friends per user in the Facebook New York social and full interaction graphs.</td>
</tr>
</tbody>
</table>
3.46 The percent of users receiving spam as the number of spammers increases on different social graph topologies. ........................................... 115
3.47 Percentage of path intersections for SybilGuard as random walk length increases. ................................................................. 118
3.48 Influence spread using the MixedGreedyWC algorithm on different graphs. ............................................................. 122
3.49 Comparing social degree in Facebook 2008 to 2009. ............ 128
3.50 Clustering coefficient of Facebook users in 2008 and 2009. .... 128
3.51 $k_{wn}$ of Facebook users in 2008 and 2009. ...................... 129
3.52 The distribution of users’ interaction among their friends, for users in 2008 and 2009. .................................................. 129
3.53 The contribution of the most interactive users to total interactions in 2008 and 2009. .............................................. 131
3.54 The contribution of the most well connected nodes to total interactions in 2008 and 2009. .............................................. 131

4.1 Our spam detection methodology, starting with raw data collection and ending with accurate classification of malicious posts and the users that generate them. .................................................. 140
4.2 Number of true positives and false positives for each tested combination of threshold values. The dotted line represents our assumed best trade-off between the two (1:2). The highlighted point (cross) represents the threshold we chose for our use. ........................................ 156
4.3 The timeline of each campaign. ........................................ 166
4.4 The malicious wall post ratio. ......................................... 170
4.5 The malicious wall post ratio as a function of number of wall posts made by the user. .................................................. 170
4.6 The wall post broadness ratio for malicious users, as a function of their social degree. .................................................. 171
4.7 The post broadness ratio of both malicious and benign accounts. 171
4.8 The active time of malicious and benign accounts. ................ 174
4.9 The hourly distribution of malicious and benign wall posts. .... 174

5.1 Friend invitation frequency over two time scales. ............... 180
5.2 Percent of accepted outgoing friend requests. .................... 180
5.3 Percent of accepted incoming friend requests. ................. 182
5.4 Clustering coefficient for users’ 50 first invitees. ............. 182
5.5 The degree of Sybil accounts. ....................................... 191
5.6 The size of connected Sybil components. ....................... 191
5.7 Scatter plot of Sybil edges vs. attack edges for Sybil components on Renren. ......................................................... 192
5.8 The order of adding Sybil friends for 1,000 Sybils. Each column represents an individual Sybil. ........................................ 192
5.9 Degree distribution of the largest Sybil component. ............ 195
5.10 Degree distribution of normal users (from [92]) and targets of friend requests from Sybils. ........................................ 195
5.11 Sessions per day for Sybil and normal users. .................. 196
5.12 Overall distribution of session durations. ......................... 196
5.13 CDF of shares per Sybil. .............................................. 200
5.14 Distribution of shares across spam blogs. ........................ 200
5.15 Connectivity between users sharing normal and spam blogs. . 203
5.16 The degree distribution for content similarity graphs with varying thresholds. ......................................................... 203
5.17 Sybil connected component sizes under different thresholds. . 207
5.18 Time similarity versus content similarity. ........................ 207
5.19 The external acceptance percentage of normal and Sybil communities. 212
## List of Tables

3.1 Statistics for the ten largest and two smallest regional networks in our Facebook dataset, our full Renren dataset, and four large OSNs from related work.  

3.2 Types of social data on PKU user interactions received directly from Renren in 2010.  

3.3 Number of Renren users in each popularity group.  

3.4 Average value of factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.  

3.5 Diary’s factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.  

3.6 Photo’s factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.  

3.7 Status’s factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.  

3.8 Correlation between diary and photo factors.  

3.9 Correlation between owner’s comments and others’ comments.  

3.10 Topology measurements for latent interaction, visible interaction, and social graphs on Renren.  

3.11 Self-Looping Statistics for SybilGuard.  

3.12 Statistics for 2009 Facebook regional networks compared to 2008. (%) are percent increases from 2008 to 2009. Other values are presented as 2008/2009. The final row (*) shows the values for the entire Facebook social graph in 2011 from [175].  

4.1 Overall statistics of our Facebook dataset.  

4.2 Examples of URL groups and their shared features.
4.3 Validation results. Each row lists the number of confirmed malicious URLs and wall posts in a given validation step. All URLs that have not been validated after all steps are assumed to be benign. 155

4.4 Number of malicious URLs in each format type. 160

4.5 Number of malicious domain names in each group. 161

4.6 A summary of spam campaigns encountered in our study. 163

5.1 Performance of SVM and threshold classifiers. 187

5.2 Statistics for the five largest Sybil components. 192

5.3 Popular Sybil creation and management tools. 196

5.4 Average actions per Sybil on Renren after June 1, 2009. 199

5.5 Simulated Sybil attacks against the PKU graph. 210
Chapter 1

Introduction

Since 2004, the social web has become a dominant force on the Internet. As of 2012, 65% of adults in the US used online social networks (OSNs) [157], and this number continues to grow, both in the US and around the world. Facebook alone accounts for 20% of all web page views [143], and 14% of all online time for web users [144].

This meteoric rise in popularity has made OSNs key channels for communication on the Internet. Social media is supplanting email as the preferred medium for basic online communication, especially among younger Internet users [54]. Twitter and Facebook have become rallying points for political protesters all around the world [151], and major news stories routinely break on Twitter before traditional news media (e.g. the death of Osama bin Laden [161] and the 2009 plane crash in the Hudson River [28]).

The massive numbers of users who have accounts on social networks have made OSNs the de-facto providers of trusted identity services for the web. Through APIs like Facebook Connect, users can sign-up and log-in to third-party websites using their
Facebook account. The simplicity of this process is enticing for users, and third-party websites benefit by gaining access to Facebook’s vast troves of personal information. In 2010, 250 million Facebook users used Connect on over 2 million sites [80]. Facebook is now leveraging their ecosystem to expand into banking and payment processing, in hopes of connecting their user base directly to online retailers [26].

These examples illustrate that OSNs have become systemically important to the web. However, as the importance of OSNs grows so do the incentives for malicious users to attack these systems. Some of the primary vectors for online criminals to profit off of social networks are:

1. Social spam.

2. Identity theft.

3. Spear phishing.

4. Fake, Sybil accounts.

These attacks are interrelated, with Sybils forming the platform on which higher level attacks are launched. We discuss each of these threats in turn, starting with the most obvious form of bad behavior: social spam.

**Social Spam.** The large user base of OSNs makes them a very attractive target for spammers. Botnets now routinely steal credentials for OSN accounts and use them
Chapter 1. Introduction

to forward spam on social networks [10, 11]. Facebook employs a small army of outsourced moderators to manually deal with the flood of spam content on their service, and other large OSNs have resorted to similar measures [42, 49].

Social spam is even more pernicious than traditional email spam because it abuses the trust that users place in their social friends. The trust between users lulls them into a false sense of security, and makes them more likely to engage with spam content. Several studies have shown that links embedded in social spam generate higher click-through rates than email spam, in some cases even approaching the rates usually associated with legitimate web advertising [78, 184]. This misappropriation of trust makes social spam an extremely lucrative target for online criminals.

Identity Theft. Criminals make money from spam by tricking unsuspecting users into clicking on links that lead to unsavory destinations. However, criminals can also make money from OSNs without actively engaging users at all, by stealing their personal information and using it to fuel identity theft. The massive amount of personal information on most OSNs is public by default, meaning anyone can crawl sensitive information from user’s accounts. Names, birth dates, and home addresses can then be used to gain illegitimate access to bank accounts and credit cards, e.g. by bypassing the “secret questions” that gate access to many online services. Theft of personal information from OSNs has helped fuel a 13% surge in identity theft crime in the US in 2011 [186]. Real-world robbers have even begun using geographic information from
status updates, photo-tags, and “check-ins” to plan home burglaries when the occupants are out of town [117].

**Spear Phishing.** The combination of widely available personal information, coupled with the ease of generating social spam, is fueling waves of targeted spear phishing campaigns. In a phishing attack, an attacker sets up a fake website that closely mimics a legitimate bank or credit card website. The attacker then spams links to the fake website to unsuspecting users, in the hope that they will visit the phishing site and divulge their username and password. Spear phishing raises the stakes by leveraging a user’s personal information to make the attack more convincing. Attackers select high-value targets, often corporate executives, and gather personal details about them from the OSN accounts of friends, family, and coworkers. The attacker then sends carefully crafted spam to the target, convincing them to visit malicious websites that are under the attacker’s control. Several high-profile spear phishing attacks have hit major corporations in recent years, resulted in million dollar losses [201].

**Social Sybils.** At the root of malicious activity on OSNs are hordes of fake, Sybil accounts that are created and controlled en-mass by attackers. As we show in Chapter 5, Sybils on OSNs are the stepping stones on which spam campaigns and theft of personal information are based. We observe some Sybils that are so stealthy they are not caught by OSN security systems for years. Facebook has conservatively estimated that there are at least 83 million Sybils on their service, representing almost 9% of the
Chapter 1. Introduction

user base [43]. Social Sybils have even been used to facilitate international espionage: in 2011, spies linked to China created a fake account for a US Navy admiral and used it to friend high ranking NATO officials in order to observe their digital interactions [142].

In addition to being a launch platform for other attacks, social Sybils have become a marketable commodity all by themselves. There now exists a thriving black-market for Sybils on OSNs [184]. Attackers create armies of Sybils and then rent them out to individuals seeking to boost their popularity on OSNs. For example, as of July 2012, Barack Obama has 17.8 million followers on Twitter, making his account one of the most popular on the site. Mitt Romney and Newt Gingrich have both been caught buying Twitter followers to match Barack Obama’s popularity [77, 85]. Rush Limbaugh has also been caught buying “likes” on Facebook [12]. Buying popularity on OSNs has become so commoditized that Twitter followers and Facebook likes can even be purchased cheaply, in bulk on Ebay.

The problem of Sybils distorting popularity and information on OSNs is impacting the social advertising market. An investigation by the BBC revealed that companies are wasting huge sums of money paying for Facebook ads that are getting clicks and likes from Sybils rather than real people [44]. This wasted money is pushing down the return on investment for social ads, to the point where large advertisers like General Motors are completely withdrawing from advertising on Facebook [185]. Since most OSNs depend on advertising as their primary revenue stream, there is an imminent danger
of Sybils undermining the business model that keeps major OSN providers financially solvent.

Towards Improved Security for OSNs. Although there is ample anecdotal evidence of security problems on OSNs, there is little concrete, quantifiable data about these threats. It is unknown how widespread these problems are, what OSN providers are doing to protect users, or how successful these efforts are. OSN providers are loath to release data about their internal operations, since it is not in their best interest to admit to security problems, as this may disenfranchise users and negatively impact profits.

In the absence of action by OSN providers, security researchers from the academic community have attempted to step in and address OSN security threats. However, existing work is hampered by a lack of access to real-world data. These studies base their work on assumptions about the behavior of normal users and attackers, and validate their algorithms through synthetic benchmarks. In some cases, these studies rely on graph attributes that are known to arise in synthetic social graph generated by models, but have not been empirically validated on real-world graphs. Without real-world data, it is impossible to realistically evaluate these security systems, which in turn prevents these technologies from being deployed by OSN providers.

In this dissertation, we take the first steps towards measuring, understanding, and defending against threats to the social web. Our work is characterized by a data-driven approach: measure real-world social graphs, quantify the behavior of normal users
and attackers, then leverage the results to build security solutions. This data-driven approach enables us to overcome the shortcomings of prior work and forge ahead with practical, deployable security systems. It also gives us a unique perspective from which to validate the models and assumptions of prior work. As we will show, the properties of real-world social graphs often diverge from the predictions of models. Similarly, measured data about the actions of OSN users sometimes contradicts “common-sense” assumptions about their behavior.

1.1 Social Graphs and User Interactions

We begin by conducting detailed studies of two of the largest OSNs in the world: Facebook and Renren. The goal of this work is to gain a broad understanding about the topological structure of real-world social networks and the behavior of OSN users. This knowledge is fundamental for determining the normal state of OSNs, and will inform our search for anomalous, malicious activity in later chapters.

To implement our study, we crawled 10 million users from Facebook in 2008 (a time when Facebook’s total population was only 67 million), and another 5 million in 2009. We crawl the complete Renren graph in 2009, gathering in 42 million users. Our Facebook graphs have hundreds of millions of edges, while the Renren graph has
Chapter 1. Introduction

over one billion edges. To date, these graphs represent two of the largest and most comprehensive social graphs ever studied.

These graphs allow us to empirically validate the topological structure of social graphs on large, real-world datasets. Results from theory and small-scale measurements have shown that social graphs exhibit properties like power-law degree scaling, short-average path lengths (small-world), and tight clustering among low-degree nodes. These basic properties underlie the emergent features that make social graphs important, such as their robustness to random failures, and their ability to rapidly disseminate information. Confirming the presence of these features on OSNs is the first step towards validating existing theoretical work, and building a foundation on which further research can be conducted.

Visible Interactions. Our work goes beyond existing studies that are focused on static topologies by accounting for the relative importance of individual edges in the social graph. By analyzing the visible interactions between users on Facebook and Renren, we show that all edges in social graphs are not equally important. In fact, nobody on Facebook interacts with more than 50% of their “friends.” Visible interactions are heavily skewed towards a chatty subset of the population, and the majority of these interactions are reciprocated by the recipient.

One interesting revelation about visible interactions is that there is an apparent ceiling on the number of friends Facebook users interact with. Even among super-users
with thousands of friends, Facebook users tend to only interact with around 150 people. This value corresponds with Dunbar’s Number, which is the theorized upper limit on the number of active social relationships human beings can maintain concurrently [61]. Thus, although users can accumulate thousands of friends on OSNs, their ability to interact with those friends is restricted by fundamental cognitive limits.

**Latent Interactions.** In addition to examining visible interactions, we use profile browsing data from Renren to examine latent interactions. We find that users browse far more profiles than they visibly interact with, and the majority of latent interactions are not reciprocated. Surprisingly, a large portion of latent interactions are generated by non-friend strangers, revealing that many users traverse several hops away from themselves on the social graph to view profiles.

A key finding of our study of latent interactions is that there is no surefire strategy for gaining popularity on OSNs. In theory, it is conceivable that some OSN accounts become popular (*i.e.* they receive many latent interactions) because they frequently post photos, or they author lengthy blog posts. However, correlative analysis between the actions of users and their popularity on Renren indicates that no factors have strong correlation with popularity. This finding suggests that external factors, like real-world celebrity status, are the only way to guarantee popularity on OSN sites.

**Interaction Graphs.** In order to develop a more quantitative approach to understanding interactions on social graphs, we propose “interaction graphs” as a way to
Chapter 1. Introduction

capture these effects. Interactions graphs replace the friendship links in social graphs with edges based on visible and latent interactions between users. We parameterize our interaction graphs with an interaction rate-threshold, which allows the model to be tuned to account for the freshness of interactions.

We compare corresponding social, visible, and latent interaction graphs and demonstrate that they exhibit very different graph topological features. The social graph is by far the densest, has the most clustering, and has the shortest average path lengths. In contrast, the latent interaction graph is much sparser, which reduces clustering and causes path lengths to rise. The visible graph continues these trends in a more extreme way, since visible interactions (and thus edges in the visible interaction graph) are rare. In essence, the key properties that are attributed to social graphs, like a highly clustered fringe and small-world characteristics, are greatly diminished in interaction graphs.

Through simulations on our real social graphs, we show that edge importance has a large effect on the performance of social applications. In many cases, the most appropriate model for social applications is an interaction graph. However, at the time, few researchers took this into account when designing their applications, instead relying on static social graphs to evaluate their work. By reevaluating these applications on interaction graphs, we show that application performance changes significantly, due to the alterations in underlying graph metrics. This demonstrates that researchers developing
new social applications and algorithms should take user interactions into account if they hope to obtain realistic and accurate results.

1.2 Social Spam

Our baseline OSN measurements allow us to characterize the behavior of normal users in great detail. This opens a window of opportunity for identifying anomalies associated with malicious activity. For example, our visible and latent interaction data reveal how frequently normal users talk to and browse their social friends. Sending more messages than average is a red-flag that a user may be spamming. Similarly, browsing significantly more profiles than average may be an indication of crawling, possibly to fuel identity theft scams.

As a first step towards understanding malicious activity on OSNs, we examine its most prominent outward symptom: spam. Spam is the natural starting point for trying to understand malicious activity on OSNs because spreading hyperlinks is the linchpin of many different attacks. The links embedded in spam point to drive-by download sites that spread Trojans (e.g. the Koobface worm [148]), phishing sites that steal passwords, and black-market e-commerce sites selling pharmaceuticals and knockoff goods.

We analyze hundreds of millions of wall posts received by millions of Facebook users and develop a novel set of automated techniques to detect social spam. Our
results provide the first quantifiable evidence that a significant portion (10%) of the URLs shared on Facebook are spam. 70% of this spam links to phishing websites, although drive-by download and underground e-commerce are also present. Analysis of the behavior of spamming accounts demonstrates that both fake, Sybil accounts and compromised normal accounts are used as tools to spread spam on Facebook.

One key finding of our study is that we confirm that spam attacks on Facebook are organized into large, coordinated campaigns. This is the hallmark of profit-driven online criminals, and proves that social media has become a lucrative target. Given that OSN spam is now a thriving business, it is likely that this problem will get worse over time unless significant preventative measures are taken.

A second key finding is that existing countermeasures designed to fight email spam are ineffective against social spam. In particular, blacklists only catch 28% of the malicious URLs present in Facebook spam. This result has also been confirmed against spam on Twitter [78].

1.3 Social Sybils

As detailed in the introduction to this section, Sybil accounts are a key tool leveraged by criminals to launch attacks against OSNs. The danger posed by social Sybils is heightened by the difficulty of defending against them. It is relatively simple for
dedicated attackers to create Sybils en-mass, either using CAPTCHA solvers [86] or by paying for cheap, crowdsourced labor [23, 131, 184]. These Sybils can be made to look very legitimate, including completely filled out profiles and realistic profile images. Sybils managed by crowdsourced labor can even post messages and interact just like normal users, since they are controlled by actual humans.

Unfortunately, as of 2010, there was no empirical understanding of the behavior of Sybils in the wild. Researchers developed many Sybil detectors that are designed to work on social networks, but these techniques all rely on assumptions about Sybil behavior that have not been verified on real-world graphs. Specifically, Sybil community detectors assume that Sybils have difficulty forming friendships with honest users, and that Sybil accounts friend each other in order to make themselves appear more legitimate [181]. The large number of connections between Sybils, coupled with the small number of edges from the Sybils to the honest portion of the graph, combine to give the Sybil community a small quotient-cut. Sybil community detectors identify this feature using specially crafted random walks on the social graph.

To quantify the behavior of Sybils in the wild, we formed a partnership with Renren Inc. Renren provided us with ground truth data about 560,000 Sybils that their security team had caught and manually verified over the course of several years. We used this data to identify four features that accurately identify Sybils, and incorporated them into a novel measurement-based Sybil detector. This system is currently deployed on
Chapter 1. Introduction

Renren, and to date it has caught and banned millions of Sybils. Importantly, our detector operates in real-time, meaning that Sybils are banned before they get a chance to generate harmful spam.

Using the corpus of Sybil data given to use by Renren, as well as 100,000 Sybils caught by our detector, we study the edge creation behavior of Sybils on Renren. Contrary to prior conjecture, we find that Sybils on Renren do not form tight-knit communities. Instead, they integrate into the social graph just like normal users. In the few cases where Sybils do form connected components, temporal data reveals that these Sybil-to-Sybil edges are formed randomly, rather than intentionally by attackers. However, just because Sybils are not explicitly connected does not imply that they are controlled by separate attackers. By examining the similarity of messages sent by Renren Sybils, we demonstrate that these Sybils collude to propagate large scale spam campaigns, indicating that they are controlled by just a few attackers behind-the-scenes.

Our insights about Sybils on Renren demonstrate that existing Sybil community detectors from the literature are unlikely to succeed on today’s OSNs. It is too easy for Sybils to integrate themselves into the social graph for Sybil detectors to rely solely on graph structural features for detection. Instead, our real-time Sybil detector uses a combination of graph structural and behavioral features to isolate Sybils. Our results challenge the status-quo of Sybil detection research on OSNs, and open new avenues for future research of enhanced Sybil detection techniques.
1.4 Summary

In summary, our research makes two high-level, fundamental contributions to the study of OSNs. First, our work demonstrates the necessity of measurement driven design of social systems. Repeatedly, our measurements have contradicted assumptions from prior work, and thus revealed new avenues of research. Our work on quantifying user interactions on OSNs is leading a shift among OSN researchers towards incorporating models of user behavior into the design of graph algorithms and social applications. Similarly, our measurements of Sybils in the wild have opened the door for new and innovative Sybil detection techniques.

Second, we have discovered, quantified, and developed practical solutions for pressing OSN security problems. However, the social web continues to evolve, and the shape of its attack surface is constantly changing. Attackers will continue to innovate new and unexpected strategies to exploit OSNs and evade security mechanisms. Only by bringing all of our tools to bear: measurement, graph analysis, data mining, machine learning, etc., can computer scientists hope to defend against future threats to the social web.
Chapter 2

Data Collection and Methodology

Over the last 60 years, scientists have discovered that complex networks are an integral part of our world. They are pervasive throughout a wide variety of disciplines, including biology (DNA, protein, and neurological structures), civil engineering (road, power, and telecommunications networks), computer science (Internet and Web), and sociology (social networks). Researchers have developed many models, metrics, and algorithms to analyze and quantify complex networks, in order to solve vital real-world problems, such as stopping viral outbreaks, preventing cascading power failures, and making the Internet resilient to attacks.

However, for many years, a lack of large-scale, real-world data prevented these models and techniques from being empirically validated. Existing work is often based on datasets that are small and difficult to generalize (e.g. user studies of a few hundred people). These limitations tie researcher’s hands and hinder the development of models that fully capture the richness of real-world complex networks. The implication is that
our understanding of complex networks is incomplete, and existing algorithms may not function as expected in the real-world.

Online social networks (OSNs) present an exciting opportunity for addressing fundamental questions surrounding complex networks. OSNs are excellent targets for study because they are massive, contain rich meta-data (user profiles, interactions, tagged edges, etc.), and are highly dynamic. Researchers can use this data to finally validate models and theories under real-world, global scale conditions. More importantly, the richness of OSN data opens the door for the discovery of new graph properties and phenomena, along with new models and metrics to quantify them.

In this section, we introduce our efforts to collect data from two of the world’s largest and most prominent OSNs: Facebook and Renren. We target these OSNs in particular because of their massive user bases, their extensive feature sets that encourage interactions between users, and their rapid growth over time. These attributes make them rich troves of data on social graph topology, graph temporal dynamics, and edge-meta data. As we show in Chapters 4 and 5, the popularity of these OSNs also make them targets for malicious attackers, which make them fertile grounds for security research as well.

First, we present general background information about the Facebook and Renren social networks. We describe these websites and their functionality as they existed at the time our measurements were taken, pointing out instances where functionality
Chapter 2. Data Collection and Methodology

has changed over time. Second, we discuss our relationship with each OSN. As we have learned, it is important for researchers to engage with the proprietors of websites that they are studying, in order to avoid later misunderstandings. Our interactions with Facebook’s staff reveal interesting insights about their mentality towards researchers, and data privacy in general, during the early, formative years of Facebook’s existence. Although our relationship with Facebook eventually fell apart, we have managed to build a lasting fruitful collaboration with Renren.

Third, we describe the methodology behind our OSN crawler and present the collected datasets. In total, we crawl information on over 10 million Facebook users (in 2008) and 42 million Renren users (in 2009). We gather the friendships between these users and reconstruct the static social graph topology. From Facebook users, we collect complete histories of wall posts and photo comments, and use this data to analyze visible interactions between users. In addition to visible interactions, Renren has a unique feature: each user profile displays a visible list of “recent visitors” who browse the profile, sorted in order, and updated in real-time. We use repeated crawls to gather 90-day histories of all the visitors to Renren user’s profiles, and use this data to analyze latent interactions between users.

Fourth and finally, we perform experiments to validate our datasets and confirm their completeness. We also briefly examine the limitations of our data and address data privacy concerns.
Chapter 2. Data Collection and Methodology

2.1 The Facebook Social Network

For logical clarity, we separate our discussion by OSN, starting with Facebook. We begin by giving background information about Facebook, and our interactions with their staff. Providing background about the Facebook OSN is vital for giving historical context to our work: Facebook’s website and features change frequently, and it is important to understand how Facebook was structured in 2008 and 2009, when our research was being conducted. Furthermore, Facebook’s attitude towards the research community, and user privacy in general, have also changed over time. Our contacts with Facebook staff provide some insights into how their positions have evolved.

2.1.1 Facebook Fundamentals

Facebook is the largest social network in the world, with over 900 million active users (as of spring 2012). Facebook is also the number one photo sharing site on the Internet [65]. Facebook allows users to set up personal profiles that include basic information such as name, birthday, marital status, and personal interests. Users establish undirected social links by “friending” other users. Each user is limited to a maximum of 5,000 total friends.

Each profile includes a message board called the “Wall” that serves as the primary asynchronous messaging mechanism between friends. Users can upload photos, which
must be grouped into albums, and may “tag” their friends in them. Comments can also
be left on photos. All Wall posts and photo comments are labeled with the name of
the user who performed the action and the date/time of submission. In 2008, Facebook
introduced the Mini-Feed feature to user profiles, which is a detailed log of each user’s
actions on Facebook over time. It allows each user’s friends to see at a glance what he
or she has been doing on Facebook, including activity in applications and interactions
with friends. Other events include new Wall posts, photo uploads and comments, pro-
file updates, and status changes. The Mini-Feed is ordered chronologically, and only
displays (at most) the user’s 100 most recent actions. The Mini-Feed has since been
replaced, first by the News-Feed, and then again by the Timeline.

Originally, Facebook was designed around the concept of “networks” that organized
users into membership-based groups. Initially (i.e. in 2004), networks on Facebook
only represented college campuses. This was soon expanded to include high schools
and companies (called work networks). Facebook authenticated membership in college
and work networks by verifying that users had a valid email address from the associated
educational or corporate domain. Facebook authenticated membership in high school
networks through confirmation by an existing member.

A user’s network membership determined what information they could access and
how their information was accessed by others. By default, a user’s profile, includ-
ing birthday, address, contact information, Mini-Feed, Wall posts, photos, and photo
comments were viewable by anyone in a shared network. Users could modify privacy settings to restrict access to only friends, friends-of-friends, lists of friends, no one, or everyone. Although membership in networks was not required, Facebook’s default privacy settings encouraged membership by making it very difficult for non-members to access information inside a network.

2.1.2 Our Relationship with Facebook

First Steps. Our initial interest in Facebook dates back to early 2008. At the time, Facebook had more than 60 million members, and it was growing at an exponential rate. Although MySpace had a larger user base at the time, it was clear that Facebook was next big thing. Furthermore, Facebook’s tight privacy controls and insistence on real names encouraged people to share themselves online in a truthful, genuine way, meaning that Facebook’s data was very high-fidelity.

However, prior to 2006, tapping into Facebook’s data had been impossible for researchers. Facebook’s default privacy settings, coupled with the strict authentication mechanisms, made it virtually impossible for researchers to crawl data from Facebook. The default privacy settings essentially made each network an isolated island, and the authentication mechanisms made it too onerous to join enough regions to crawl a representative sample of the population.
Chapter 2. Data Collection and Methodology

Things changed in September 2006 when Facebook introduced regional networks. Each region represented a geographic area like a large city, a US state, or a small country. Importantly, access to regional networks was unauthenticated, i.e. anyone could join any region. By spring 2008, Facebook had 67 million users, 66.3% of whom (44.3 million) belonged to a regional network. Users were permitted to belong to one college, high school, or work network, in addition to one regional network (which they could change twice every sixty days). Many existing Facebook users from college and work networks took advantage of this dual membership feature by adding themselves to regional networks as well.

Unauthenticated regional networks were the perfect opportunity for researchers to gather data on Facebook, and by 2008, enough users were members of regional networks to make data gathering worthwhile. We conducted several small-scale test crawls using accounts located in the largest regional networks, and we were shocked by the results. The majority of users had not modified the default privacy settings, meaning that their profiles were accessible to all users in their region. Furthermore, Facebook only had rudimentary security measures in place against crawling: accounts that generated too much traffic on a given day where banned at midnight. However, during the day there were no roadblocks (e.g. CAPTCHAs) or rate-limits preventing mass crawling.

Contacting Facebook. Once we had determined that Facebook could be crawled, we immediately contacted them to inform them about our research. Prior experience
Chapter 2. Data Collection and Methodology

taught us that this is a necessary precaution when crawling websites for data. In particular, during previous research on Overstock Auctions, we had made the mistake of crawling their website without putting self-imposed rate-limits on our crawler [166]. The crawler ended up denial-of-servicing the Overstock Auctions site, which brought us into contact with their security staff within 24 hours. We managed to resolve this misunderstanding amicably, but it taught us a valuable lesson: proceed with extreme caution when crawling, and contact the target website ahead of time to obtain permission!

Facebook’s staff put us in contact with Chris Kelly, who was Chief Privacy Office at Facebook at the time. Mr. Kelly gave us permission to crawl, and maintained contact with our team over several months. We informed Facebook’s privacy team about the security hole that was enabling our crawls, i.e. the confluence of unauthenticated regional networks and default, permissive privacy settings. Surprisingly, their team had not considered the implications of these factors when they rolled out regional networks.

One thing that immediately became apparent during our conversations with the Facebook privacy team was that they were genuinely confused by our interest in their data. At the time, no other research groups were crawling Facebook, and the immense value of Facebook’s data for scientific research was unknown to Facebook’s staff. By sheer chance, we managed to get approval for our crawls before the furor over Facebook
privacy in the news media made it impossible for them to condone outside academic research.

Thus, in March 2008, we began systematically crawling Facebook’s regional networks. We continued crawling until May 2008, when we halted in order to focus on analyzing the data. We crawled Facebook again in spring 2009 for additional data, and to observe the growth of the social graph over time (see Section 3.8). Facebook stopped being structured around networks in summer of 2009, which prevented us from continuing to crawl. We have not conducted large scale crawls of Facebook since spring 2009.

**Falling Out.** We have not been in contact with Facebook since April 2011, when Facebook’s lead counsel sent a strongly worded letter to the Office of the President of University of California. The letter was written in response to our measurement study on Facebook spam (see Chapter 4). To quote the letter:

> Facebook does not wish to take action against any University of California researchers or students collecting data on their behalf. However, we must insist that University researchers and students abide by all Facebook’s terms, including the Statement of Rights and Responsibilities (available at [http://www.facebook.com/terms.php](http://www.facebook.com/terms.php)), which prohibit the creation of fake accounts, and Facebook’s Automated Data Collection Terms (available at [http://www.facebook.com/robots.txt](http://www.facebook.com/robots.txt)), which prohibit collecting and using data without Facebook’s permission.

Although the spam study was originally published in November 2010, Facebook’s counsel only took notice after several news outlets wrote up stories about our re-
Chapter 2. Data Collection and Methodology

search [159]. Of course, we had permission from Facebook for the data in our possession, as well as the means we used to acquire the data. We forwarded the emails that verify these facts to Facebook’s general counsel. Similarly, the issue of ongoing data collection was moot, since by 2011 we had not crawled Facebook in over a year. Fortunately, the letter made it clear that Facebook did not wish to pursue legal recourse against our group, and the matter was silently dropped.

2.2 Facebook Dataset and Collection Methodology

In this section, we describe our technical methodology for collecting Facebook data. We present experimental validation of the completeness of our graph crawl and detail the types of user data that we collected (which form the basis for our experiments in later chapters). Finally, we address limitations of our methodology, and discuss data privacy issues.

2.2.1 Data Collection Process

Crawling Facebook. While other studies of social networks rely on statistical sampling techniques [128] to approximate graph coverage of large social networks, Facebook’s partitioning of the user population into networks means that subsets of the social graph can be completely crawled in an iterative fashion. Our primary dataset is
composed of profile, Wall and photo data crawled from the 22 largest regional networks on Facebook between March and May of 2008. We list a subset of these networks and their key characteristics in Table 3.1.

To crawl Facebook, we implemented a distributed, multi-threaded crawler using Python with support for remote method invocation (RMI) [34]. In 2008 and 2009, Facebook provided a feature to show 10 randomly selected users from a given regional network; we performed repeated queries to this service to gather 50 user IDs to “seed” our breadth-first searches of social links on each regional network. Two dual-core Xeon servers were generally able to complete each crawl in under 24 hours, while averaging roughly 10 MB/s of download traffic. Our completed dataset is approximately 500 GB in size, and includes full profiles of more than 10 million Facebook users.

At the time of our research (2008-2009), Facebook’s only systematic countermeasure against crawling was that it banned user accounts that generated too much traffic at midnight, every night. This restriction did not impact most of our crawls, since most regions could be crawled in less than 24 hours. However, the larger regional networks (e.g. London) took more than 24 hours to crawl, and thus we engineered our crawlers and our daily schedules to cope with Facebook’s countermeasure. At 11:55pm, all of the crawlers would automatically save their state to disk (including the set of already seen pages and the uncrawled “frontier” queue) and shutdown. Meanwhile, our staff would create fresh Facebook accounts. At 12:01am, the crawlers would restart using
the fresh account credentials, load their previous state off disk, and pick up crawling right where they left off.

On several instances, Facebook banned the IP addresses of our crawling machines, despite the fact that they had given us permission to crawl. These bannings reveal an interesting internal disconnect within Facebook: although the privacy team had given us permission to crawl, the completely distinct security team was unaware of our activities, and hence banned our IPs. After emailing the privacy team, they were able to speak to the security team and get the ban on our IPs lifted.

Crawling all 22 Facebook regions took a considerable number of days to complete. Unfortunately for us, Facebook constantly tweaks the HTML of their pages, which broke our crawlers on several occasions. This forced us to include numerous “sanity checks” in our crawlers that would immediately alert us if a page could not be parsed as expected, and save a copy of the strange HTML. This enabled us to quickly identify the changes in Facebook’s HTML and roll out new versions of the crawler to cope.

2.2.2 Description of Collected Data, Limitations, and Privacy

We collected the full user profile of each user visited during our crawls. In addition to this, we also collected full transcripts of Wall posts and photo comments for each user. Although Facebook profiles do not include a “Date Joined” field, we can estimate this join date by examining each user’s earliest Wall post. The Wall is both ubiquitous
and the most popular application on Facebook, and a user’s first Wall post is generally a welcome message from a Facebook friend. Thus we believe a user’s earliest Wall post corresponds closely with their join date. We also collected photo tags and comments associated with each user’s photo albums, since this is another prevalent form of Facebook interaction, and give us insight into users who share physical proximity as well as online friendships.

While the Wall and photo comments are in no way a complete record of user interactions, they are the oldest and most prevalent publicly viewable Facebook applications. Our datasets from crawls of user Mini-Feeds show that they are also the two most popular of the built-in suite of Facebook applications by a large margin. Most of the other applications are comparatively recent additions to Facebook (in 2008), and cannot shed light on user interactions from Facebook’s earlier history. For example, the Wall was added to Facebook profiles in September 2004, while the Notes application was not introduced until August 2006.

**Dataset Completeness.** Prior research on online social networks indicates that the majority of user accounts in the social graph are part of a single, large, weakly connected component (WCC) [128]. Since social links on Facebook are undirected, breadth first crawling of social links should be able to generate complete coverage of the large connected component, assuming that at least one of the initial seeds of the crawl is linked to the connected component. The only inaccessible user accounts should
be ones that lie outside the regional network of the crawl, have changed their default privacy settings, or are not part of the connected component.

To validate our data collection procedure and ensure that our crawls are reaching every available user in the connected component, we performed five simultaneous crawls of the San Francisco regional network. Each crawl was seeded with a different number of user IDs, starting with 50 and going up to 5000. The difference in the number of users discovered by the most and least revealing crawls was only 242 users out of \(~169K\) total (a difference of only 0.1\%). The 242 variable users display uniformly low node degrees of 2 or less, indicating that they are either new accounts that were added during our crawl, or outliers to the connected component that were only discovered due to the addition of more seeds to the crawl. This experiment verifies that our methodology effectively reaches all nodes in the large connected component in each regional network within a negligibly small margin of error. This testing procedure is the same one used in [128] to verify their crawling methodology.

Limitations. There are two limitations inherent in our data collection process. First, we are restricted by Facebook’s privacy settings. Our dataset only includes users with public profiles, and we only collect public interactions. On Facebook, public interactions include Wall posts and photo comments, while private interactions are direct messages between users and “pokes.” As discussed in Section 3.2.2, our crawl covers
Chapter 2. Data Collection and Methodology

the majority of users in each regional network, and therefore we believe it is representative of the overall Facebook population.

Second, not all of the friend relationships gathered by our crawler provide useful data. Each user has friends that are within their regional network, and we refer to these edges as internal. However, some portion of a user’s friends may be outside the regional network, and we refer to these edges as external. External edges are useless, because they connect to nodes that we are unable to crawl. Thus, external edges, and the external nodes they connect to, cannot be incorporated into our regional graphs.

The presence of external edges complicates our analysis in a few subtle ways. When calculating a user’s social degree, we include their external edges, since this gives us the user’s true degree. However, when calculating other graph metrics, such as clustering coefficient or assortativity, we do not include external edges. These metrics rely on graph information from a given user and their friends. Since we do not have complete information on external friends, we must exclude them from our analysis.

Despite these two limitations, we believe that our Facebook data is still representative of the Facebook graph as a whole. Our results from the crawled Facebook graph are extremely similar to those calculated using the entire Facebook graph (see Section 3.8) [175]. Similarly, other studies of interactions on Facebook that have had access to private interactions have reached similar conclusions to our study [19, 74].
Chapter 2. Data Collection and Methodology

Privacy and Data Anonymization. In order to preserve the privacy of Facebook users, we anonymized all user IDs after our crawls were completed. Users’ photos were not saved by our crawlers. No semantic data (e.g. Wall post content, photo comments) were stored during our 2008 crawls, although we did retain this data in our 2009 crawls (in order to facilitate security research, see Chapter 4). All collected data is stored on an isolated storage system that only our research group has permission to access.

Obviously, we take the privacy of OSN data extremely seriously. This has led us to conduct several research projects aimed at protecting user privacy on OSNs. See Section 6.4 for more information about our privacy protection research.

2.3 The Renren Social Network

We now shift away from Facebook and focus on Renren. In this section, we introduce the Renren OSN and point out some of the unique features that make it an interesting target for research.

2.3.1 Renren Fundamentals

Launched in 2005, Renren is the largest and oldest OSN in China. Renren can be best characterized as Facebook’s Chinese twin, with most or all of Facebook’s features, layout, and a similar user interface. Users maintain personal profiles, upload photos,
write diary entries (blogs), and establish bidirectional social links with their friends. Renren users inform their friends about recent events with 140 character status updates, much like tweets on Twitter. All user-generated updates and comments are tagged with the sender’s name and a time stamp.

User profiles on Renren are very similar to Facebook. Each profile includes a profile picture, personal information (name, age, education background, work experience, hobbies, etc.), and a subset of the user’s friend list (since friend lists are often hundreds of users long). The body of each profile is a chronologically ordered “feed” of the user’s actions: status updates, comments sent and received, photos uploaded and tagged, shared web links, blog entries written, etc.

Like Facebook, Renren evolved from a social network in a university setting. Its predecessor was called Xiaonei, literally meaning “inside school.” In September 2009, Renren merged with 5Q, the second largest OSN in China, and absorbed all of 5Q’s user accounts. In depth analysis of the Xiaonei/5Q merge is available in our study on the graph dynamics of Renren [202].

Renren organizes users into membership-based networks, much like Facebook used to. Networks represent schools, companies, or geographic regions. Membership in school and company networks require authentication. Students must offer an IP address, email address, or student credential from the associated university. Corporate email addresses are needed for users to join corporate networks. Unlike Facebook,
Renren’s default privacy policy makes profiles of users in geographic networks private. This makes them difficult to crawl. Fortunately, profiles of users in authenticated networks are public by default to other members of the same network. This allowed us to access user profiles within the Peking University network, since we could create nearly unlimited authenticated accounts using our own block of IP addresses.

Like Facebook, a Renren user’s homepage includes a number of friend recommendations that encourage formation of new friend relationships. Renren lists 3 users with the most number of mutual friends in the top right corner of the page. In addition, Renren shows a list of 8 “popular users” at the very bottom of the page. These popular users are randomly selected from the 100 users with the most friends in the university network.

**Unique features.** Renren differs from Facebook in several significant ways. First, each Renren user profile includes a box that shows the total number of visitors to the profile, along with names and links to the last 9 visitors ordered from most to least recent. In addition, Renren also maintains a visible counter of visitors (not including the user himself) on each individual photo and diary page. These lists and counters have the same privacy settings as the main profile. These counters have the unique property of making invisible web browsing events visible, and are the basis for our detailed measurements on latent user interactions.
Chapter 2. Data Collection and Methodology

A second crucial feature is that friend lists on Renren were public in 2009, when we collected data for our study. Users had no way to hide them, which allowed us to perform an exhaustive crawl of the largest connected component in Renren (42.1 million users). This contrasts with other OSNs, where full social graph crawls are prevented by user privacy policies that hide friendship links from the public. The exception is Twitter, which behaves more like a public news medium than a traditional social network [105]. Renren has since changed this policy: by default, friend lists are now only viewable by friends.

In addition, comments in Renren are threaded, i.e. each new comment is always in response to one single other event or comment. For example, user A can respond to user B’s comment on user C’s profile, and only B is notified of the new message. Thus we can precisely distinguish the intended target of each comment. One final difference between Renren and Facebook is that each standard user is limited to a maximum of 1,000 friends. Users may pay a subscription fee to increase this limit to 2,000. From our measurements, we saw that very few users (0.3%) took advantage of this feature.

2.3.2 Our Relationship with Renren

Just as with Facebook, we contacted Renren soon after we began crawling their network. We obtained permission to crawl, but unlike Facebook, our relationship with Renren has only gotten stronger over time. The staff at Renren was pleased with our ini-
Chapter 2. Data Collection and Methodology

tial measurement studies of their social graph, and they accepted several of our student collaborators into their internship program. Since then, we have successfully collaborated with Renren on several projects [183, 202], including the research presented in Chapter 5.

2.4 Renren Dataset and Collection Methodology

In this section, we describe our methodology for collecting Renren data. In general, the methodology is similar to the one we used when crawling Facebook. However, gathering latent interaction data from Renren users requires overcoming additional technical challenges.

2.4.1 Data Collection Process

Crawling Renren. We crawled the entire Renren network from April 2009 to June 2009, and again from September to November of 2009. We seeded the crawlers with the 30 most popular users’ profiles, and proceeded to perform a breadth-first traversal of the social graph. During the crawl, we collect unique user IDs, network affiliations, and friendship links to other users. For our study, we use data from our second crawl, which generated an exhaustive snapshot that included 42,115,509 users and 1,657,273,875 friendship links. While this is significantly smaller than the 70 million users advertised
by Renren in September 2009, we believe the discrepancy is due to duplicate accounts that were abandoned after the merge between Renren and 5Q in September 2009.

**Crawling the PKU network.** We performed smaller, more detail oriented crawls of the Peking University (PKU) network between September and November of 2009 (90 days) to collect information about user’s profiles and interaction patterns. This methodology works because the default privacy policy for authenticated networks is to make full profiles accessible to other members of the same network. Since we collected the network memberships of all users during our complete crawl, we were able to isolate the 100,973 members of the PKU network to seed our detailed crawl. Of these users, 61,405 users had the default, permissive privacy policy, enabling us to collect their detailed information. This covers the majority of users (60.8%) in the PKU network, and provides overall network coverage similar to other studies that crawled OSN regional networks [190].

As part of our PKU crawls, we gathered all comments generated by users in message board posts, diary entries, photos, and status updates. This data forms the basis of our experiments involving visible interactions. Our dataset represents the record of public visible interactions between users in the PKU network. In total, 19,782,140 comments were collected, with 1,218,911 of them originating in the September to November 2009 time-frame.
Chapter 2. Data Collection and Methodology

Privacy and data anonymization. Our study focuses on the structure of social graphs and interaction events between users. Since we do not need any actual content of comments, photos, or user profiles, we waited for crawls to complete, then went through our data to anonymize user IDs and strip any private data from our dataset to protect user privacy. In addition, all user IDs were hashed to random IDs, and all timestamps are replaced with relative sequence numbers. We note that our group has visited and held research meetings with technical teams at Renren, and they are aware of our ongoing research.

Limitations. In the following sections, we quantify the graph structural properties of the Renren social graph and show that it is very similar to other large OSNs like Facebook and Orkut. However, in later sections when we analyze latent browsing behavior, it is more difficult to directly compare results to other OSNs. Although latent interactions have been studied before [31, 156], the datasets used in these studies are not publicly available. It is possible that cultural and political issues in China affect the social behavior of Renren users. Therefore, we caution that latent interaction results from Renren may not generalize to all OSNs.
2.4.2 Measuring Latent User Interactions

In addition to visible interactions generated by users in the PKU network, we also recorded the recent visitor records displayed on each user’s profile. This data forms the basis of our study of latent interactions.

Reconstructing Visitor Histories. Crawling Renren for recent visitor records is complicated by two things. First, each user’s profile only lists the last 9 visitors. This means that our crawler must be constantly revisiting users in order to glean representative data, as new visitors will cause older visitors to fall off the list. Clearly we could not crawl every user continuously. Frequent crawls leave the ID of our crawler on the visitor log of profiles, which has generated unhappy feedback from profile owners. In addition, Renren imposes multiple rate limits on crawlers: first, each crawler account is only allowed to visit 1 profile per minute; second, each crawler account must solve a CAPTCHA if it visits 100 profiles in a short time. Otherwise, the crawler account is forbidden from viewing profiles for 2.5 hours. These rate limits slow our crawler significantly despite our large number of crawler accounts. Thus, we designed our crawler to be self-adapting. This means that we track the popularity and level of dynamics in different user profiles, and allocate most of our requests to heavily trafficked user profiles, while guaranteeing a minimum crawl rate (1/day) for low traffic users. The
individual lists from each crawl contain overlapping results, which we integrate into a single history.

The second challenge to crawl recent visitor records is that each visitor is only shown in the list once, even if they visit multiple times. Repeat visits simply cause that user to return to the top of the list, erasing their old position. This makes identifying overlapping sets of visitors from the iterative crawls difficult.

To solve these two challenges, we use a log-integration algorithm to concatenate the individual recent visitor lists observed during each successive crawl. More specifically, some overlapping sets of visitors exist in successive crawl data, and our main task is to find new visitors and remove overlaps. There are two kinds of incoming visitors: new users, who do not appear in the previous list, and repeat users, who appear in the prior list at a different relative position. The first kind of incoming visitor is easily identified, since his record is completely new to the recent visitor list. New visitors provide a useful checkpoint for purposes of log-integration, since other users behind them in the list are also necessarily new incoming visitors. The second type of incoming visitor, repeat users, can be detected by looking for changes in sequence of the recent visitor list. If a user repeatedly visits the same profile in-between two visits of other users, nothing changes in the recent visitor list. Therefore, consecutive repeat visits are ignored by our crawler.
Chapter 2. Data Collection and Methodology

Time 1: ABCDEFGHI
Time 2: CDEFGHIJK
Time 3: DFGHIEJKC
Time 4: GHIJEKLM
Result: ABCDEFGHIELCM

Figure 2.1: Integrating multiple visitor lists captured by multiple crawls of the same profile into a single history.

Figure 2.1 demonstrates our integration algorithm. We observe that visitors ABCDEFGHI viewed a user’s profile at some time before our first crawl. New users view the profile and are added to the recent visitor list by the second crawl at time 2. We re-observe the old sequence CDEFGHI, and identify JK as new visitors, since JK do not exist in the previous visitor list. Next, we compare recent visitor lists at time 2 and 3. We find that E is before K in the recent visitor list crawled at time 2, but this order is changed at time 3. This means that at some time before the third crawl user E revisited the target and changed positions in the list. Thus we identify E as a new visitor. Since C is behind E at time 3, C is also identified as a new visitor. Our integration algorithm also works correctly at time 4. User L has not been observed before, and thus L, plus subsequent visitors C and M, are all classified as new visitors.

Overall, from the 61,405 user profiles we continuously crawled, we obtained a total of 8,034,664 total records of visits to user profiles in the PKU network. After integrat-
Chapter 2. Data Collection and Methodology

![Figure 2.2: Average daily visit counts of user profiles.](image1)

![Figure 2.3: Number of visits missed when we lower crawler frequency from a high of once every 15 minutes.](image2)

After filtering these raw results, we are left with 1,863,168 unique profile visit events. This high reduction (77%) is because most profiles receive few page views, and thus overlaps between successively crawled results are very high. Although Renren does not show individual recent visitors of user diaries and photos, it does display the total number of visits, which we crawled as well.

**Impact of Crawl Frequency.** We are concerned that our crawls might not be frequent enough to capture all visit events to a given profile. To address this concern, we took a closer look at the impact of crawler frequency on missing visits. First, we take all of the profiles we crawled for visit histories, and computed their average daily visit count between September and November 2009. We plot this as a CDF in Figure 2.2. Most users (99.3%) receive $\leq 8$ visits per day on average. Since Renren shows 9 latest visitors, crawling a profile once every day should be sufficient to capture all...
visits. While our crawler adapts to allocate more crawl requests to popular, frequently visited profiles, we guarantee that every profile is crawled at least once every 24 hours.

Next, we select 1,000 random PKU users and crawl their recent visitors every 15 minutes for 2 days. We use the data collected to simulate five frequencies for the crawling process: 15 minutes, 30 minutes, 1 hour, 12 hours, and 1 day. Then we use the log-integration algorithm to concatenate the individual recent visitor lists at different crawling frequencies. For every person, we compute the number of visits missed by the crawler when we reduce the frequency, beginning with visits every 15 minutes. We plot the CDF of these deviations in Figure 2.3. For 86% of users, there are no visits missed when we reduce the crawler rate from once every 15 minutes to once per day. The remaining 14% of users require more than one crawl per day to collect a full history of their visits.

Based on these observations, we engineered our crawler to allocate the bulk of crawl requests to high popularity users. For each PKU user, the crawler determines how many times the user will be visited tomorrow by calculating \( \lfloor v/9 \rfloor + 1 \), where \( v \) is the number of times the user’s profile was visited today. This formula ensures that all users are visited at least once, and users who are visited \( \geq 9 \) times are crawled in proportion to their historical popularity.
Chapter 3
Social Graphs and User Interactions

3.1 Introduction

1 The immense popularity of OSNs is driving the emergence of new classes of socially-enhanced applications that leverage relationships from social networks to improve security and performance of applications. Example applications include spam email mitigation [73], Internet search [126], and defense against Sybil attacks [200]. In each case, meaningful, interactive relationships with friends are critical to improving trust and reliability in the system.

Unfortunately, these applications assume that all online social links denote a uniform level of real-world interpersonal association, an assumption disproven by social science. Specifically, social psychologists have long observed the prevalence of low-interaction social relationships such as Milgram’s “Familiar Stranger” [125]. Recent

1Portions of this chapter originate from our papers “User Interactions in Social Networks and their Implications” [190] and “Understanding Latent Interactions in Online Social Networks” [92].
research on social computing shows that users of social networks often use public display of connections to represent status and identity [59], further supporting the hypothesis that social links often connect acquaintances with no level of mutual trust or shared interests.

This leads to the question: \textit{Are social links valid indicators of real user interaction? If not, then what can we use to form a more accurate model for evaluating socially-enhanced applications?} In this chapter, we address this question through a detailed study of two OSNs: Facebook and Renren.

We make four key contributions throughout this chapter. First, we present a comprehensive examination of the structure of two of the largest and most important OSNs in existence. Our Facebook data represents the largest and most well cited snapshot of the OSN currently in existence. Similarly, our Renren graph is the second largest complete OSN snapshot available in academia, second only to Twitter [105]. We compare the Facebook and Renren social graphs to YouTube, LiveJournal, and Orkut, and reveal areas where the OSNs differ in fundamental graph structure.

Second, we present detailed analysis of user interactions on Facebook and Renren. Using wall posts and photo comments from Facebook, we show that users tend to visibly interact with only a small subset of friends, often having no interactions with over 50% of their Facebook friends. We use latent interactions (\textit{i.e.} profile browsing behavior) between Renren users to examine issues of popularity, visitor composition,
and reciprocity. Our findings demonstrate that although latent interactions are more prevalent than visible interactions, they are rarely reciprocated, and almost 50% are generated by non-friend “strangers.” Altogether, our study of user interactions casts doubt on the practice of extracting meaningful relationships from social graphs, and suggests an alternative model for validating user relationships in social networks.

Third, we propose interaction graphs (Section 3.6) as a model for representing social relationships based on interactions between users. An interaction graph contains all nodes from its social graph counterpart, but only a subset of the links. A social link exists in an interaction graph if and only if its connected users have directly interacted. We construct two types of interaction graphs: visible interaction graphs and latent interaction graphs. We compare their salient properties, such as clustering coefficient and average path lengths, to their social graph counterparts. We observe that interaction graphs demonstrate significantly different properties from those in standard social graphs, including longer average path lengths, lower clustering coefficients, and higher assortativity.

Finally, in Section 3.7 we examine the impact of using different graph models in evaluating socially-enhanced applications. We conduct simulated experiments of Reliable Email [73], SybilGuard [200], and MixedGreedyWC Influence Maximization [50] on social, visible interaction, and latent interaction graphs derived from Facebook and
Chapter 3. Social Graphs and User Interactions

Renren data. Our results demonstrate that differences in the three graph models translate into significantly different application performance results.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>London, UK</td>
<td>1,241K (50.8)</td>
<td>30,743K (26.5)</td>
<td>5.09</td>
<td>0.170</td>
<td>0.25</td>
</tr>
<tr>
<td>Australia</td>
<td>1,215K (61.3)</td>
<td>27,261K (36.0)</td>
<td>5.13</td>
<td>0.175</td>
<td>0.17</td>
</tr>
<tr>
<td>Turkey</td>
<td>1,030K (55.5)</td>
<td>17,739K (35.2)</td>
<td>5.10</td>
<td>0.133</td>
<td>0.06</td>
</tr>
<tr>
<td>France</td>
<td>728K (59.3)</td>
<td>11,227K (34.6)</td>
<td>5.21</td>
<td>0.172</td>
<td>0.12</td>
</tr>
<tr>
<td>Toronto, ON</td>
<td>483K (41.9)</td>
<td>11,829K (21.9)</td>
<td>4.53</td>
<td>0.158</td>
<td>0.23</td>
</tr>
<tr>
<td>Sweden</td>
<td>575K (68.3)</td>
<td>17,290K (44.8)</td>
<td>4.55</td>
<td>0.158</td>
<td>0.19</td>
</tr>
<tr>
<td>New York, NY</td>
<td>378K (45.0)</td>
<td>7,233K (15.7)</td>
<td>4.80</td>
<td>0.146</td>
<td>0.19</td>
</tr>
<tr>
<td>Colombia</td>
<td>565K (71.7)</td>
<td>10,242K (31.7)</td>
<td>4.94</td>
<td>0.136</td>
<td>0.09</td>
</tr>
<tr>
<td>Manchester, UK</td>
<td>395K (55.5)</td>
<td>11,124K (35.2)</td>
<td>4.79</td>
<td>0.195</td>
<td>0.21</td>
</tr>
<tr>
<td>Vancouver, BC</td>
<td>314K (45.1)</td>
<td>8,240K (25.3)</td>
<td>4.71</td>
<td>0.170</td>
<td>0.23</td>
</tr>
<tr>
<td>Egypt</td>
<td>246K (57.8)</td>
<td>3,236K (25.5)</td>
<td>4.88</td>
<td>0.167</td>
<td>0.01</td>
</tr>
<tr>
<td>San Francisco</td>
<td>172K</td>
<td>2,911K</td>
<td>4.8</td>
<td>0.194</td>
<td>0.18</td>
</tr>
<tr>
<td>Total/Avg.:</td>
<td>10,697K (56.3)</td>
<td>240,265K (29.4)</td>
<td>4.8</td>
<td>0.164</td>
<td>0.17</td>
</tr>
<tr>
<td>Renren</td>
<td>42,115K (100.0)</td>
<td>1,657,273K (100.0)</td>
<td>5.38</td>
<td>0.063</td>
<td>0.15</td>
</tr>
<tr>
<td>Orkut [128]</td>
<td>3,072K (11.3)</td>
<td>223,534K</td>
<td>4.25</td>
<td>0.171</td>
<td>0.072</td>
</tr>
<tr>
<td>Flickr [128]</td>
<td>1,846K (26.9)</td>
<td>22,613K</td>
<td>5.67</td>
<td>0.313</td>
<td>0.202</td>
</tr>
<tr>
<td>Cyworld [15]</td>
<td>12,048K (100.0)</td>
<td>190,589K</td>
<td>3.2</td>
<td>0.16</td>
<td>-0.13</td>
</tr>
<tr>
<td>Twitter [91]</td>
<td>88K</td>
<td>829K</td>
<td></td>
<td>0.106</td>
<td>0.59</td>
</tr>
</tbody>
</table>

Table 3.1: Statistics for the ten largest and two smallest regional networks in our Facebook dataset, our full Renren dataset, and four large OSNs from related work.

3.2 Analysis of Social Graphs

In this section, we present high level measurement and analysis results on our Facebook and Renren datasets. First, we define and formalize terminology. Next, we analyze the topological properties of the Facebook and Renren social graphs by focusing
Chapter 3. Social Graphs and User Interactions

on salient graph metrics. We compare the Facebook and Renren social graphs to other
OSNs from the literature, and confirm that they both exhibit properties that are con-
sistent with other representative social networks. Next, we examine the growth of the
Facebook and Renren user populations over time. Finally, we conclude the section by
briefly analyzing fine-grained user activities over time based on Mini-Feed events from
Facebook San Francisco and Renren PKU users.

3.2.1 Definitions

In this chapter, we model each social network as an undirected graph $G = (V, E)$. The set of nodes $V$ corresponds to users on the social network. We use the term node and user interchangeably. The set of edges $E$ corresponds to social links between users on the social graph. On Facebook and Renren, users explicitly create undirected social links by “friending” each other. We say that two users have a relationship if they are connected by an edge.

Much of the analysis in this chapter focuses on user interactions. We identify two different types of interactions: visible interactions and latent interactions. Visible interactions correspond to events like writing on a friend’s wall, or commenting on a friend’s photo. For example, if user $A$ writes on $B$’s wall and comments on one of $B$’s photos, then $A$ has visibly interacted with $B$ twice.
In contrast to visible interactions, latent interactions refer to user’s browsing behavior. For example, if user $A$ visits $B$’s profile, then there exists a single latent interaction from $A$ to $B$.

Formally, we define an interaction as an event $i_{u,v}$ that is generated by one user $u$ and directed at a second user $v$ where $u, v \in V$. The set of interactions $I$ is a multiset, e.g. interactions between pairs of users can occur multiple times. In the case of visible interactions, $I \subseteq E$, meaning each visible interaction $i_{u,v} \in I$ can exist if and only if edge $e_{u,v} \in E$. This constraint arises from the requirement on Facebook and Renren that users be friends before they comment on each other’s content. However, for latent interactions $E \subseteq I$, i.e. latent interactions may occur between users who are not social friends.

### 3.2.2 High Level Statistics

Table 3.1 lists the overall properties of our Facebook and Renren datasets. In total, our crawler was able to collect roughly 10 million users from the 22 largest regional networks on Facebook, which represents 56% of the total user population of those networks. The remaining 44% of users could not be crawled due to restrictive privacy policies or disconnection from the connected component of the graph. Our complete dataset includes about 818 million social links, although table 3.1 only lists statistics on the ten largest and the two smallest Facebook regional networks that we crawled.
Figure 3.1: Comparing the degree distribution of Facebook and Renren to Orkut, YouTube and LiveJournal [128]. Both CDF and CCDF distributions are shown.

Our entire Renren dataset consists of over 42 million users and 1.7 billion edges. This is the second largest, complete social graph ever to be studied by academics, only being surpassed by the Twitter snapshot analyzed in [105].

3.2.3 Degree Distributions

In Figure 3.1, we compare the social degree (i.e. number of friends) of Facebook and Renren users against prior results obtained for three other social networks: Orkut, YouTube and LiveJournal [128]. Connectivity among Facebook and Renren users most closely resembles those of users in Orkut, likely because all three are sites primarily focused on social networking. In contrast, YouTube and LiveJournal are content distribution sites with social components, and exhibit much lower social connectivity. Facebook users are more connected than Orkut users: 37% of Facebook users have more than 100 friends, compared to 20% for Orkut.
Figure 3.1 shows the Cumulative Distribution Function (CDF) and Complementary Cumulative Distribution Function (CCDF) of social degrees for five social networks. Prior work has shown that many measured large graphs, including social networks, exhibit a power-law degree distribution [25]. However, Facebook and Renren do not follow a pure power-law: as shown in the CCDF of Figure 3.1, the degree distribution is not a straight line in a log-log plot. Facebook and Renren’s degree distributions are most similar to Orkut, which is also not pure power-law [128]. Facebook’s distribution drops rapidly as it approaches 5000, which is the maximum friend limit on Facebook. Similarly, Renren’s distribution drops as it approaches 1000, which is the default friend limit. The bump in the tail of the Renren distribution around 2000 is due to a small number of users who pay money to raise their friend limit to 2000.

In order to calculate the power-law exponent of the body of the Facebook and Renren degree distributions, we use the method from [128] (which is a modification of the method from [55]). We calculate that Facebook has an alpha value of 1.25, with fitting error of 0.31. This alpha is significantly lower than the alpha value derived for YouTube (alpha=1.63, fitting error=0.13), a network that does demonstrate a very clear power-law degree distribution. We attempted to calculate the exponent for Renren, but the fitting error was unacceptably high, even if the users with degree >1,000 were filtered out. Thus, neither Facebook nor Renren exhibits strong power-law degree scaling.
In the course of our experiments, we have encountered some difficulties when calculating the power-law exponent of social graphs. Specifically, the software provided by Clauset et al. [55] to calculate power-law fit has a hidden parameter that can lead to erroneous results. By default, the software only calculates power-law coefficients in the range [1.5, 3.5]. In the case of graphs like Facebook, where the best fit is 1.25, the software’s default behavior is to give a result of 1.5. Researchers using the Clauset software must be careful of the hidden range parameter: *any result that is close to the lower or upper bounds of the range should be viewed with suspicion.* We advise researchers to widen the coefficient fitting range to [1.0, 5.0] before conducting any tests, in order to avoid generating any misleading results.

### 3.2.4 Average Path Lengths

Average path length is the average of all-pairs-shortest-paths in the social network. It is simply not tractable to compute shortest path for all node pairs, given the immense size of our social graph. Instead, we choose 1000 random users in the network, perform Dijkstra to build a spanning tree for each user in the social graph, and compute the length of their shortest paths to all other users in the network.

To evaluate graph distances on Facebook, we construct a social graph for each crawled regional network. Some of the social links in our dataset were not crawled, because they point to users that are either not members of the specified regional net-
work, or have modified their default privacy settings. Since we do not have complete
social linkage information on these users, we limit each regional social graph to only in-
clude links for which users at both endpoints were fully visible during our crawls. This
prevents incomplete information on some users from biasing our results. As shown
in Table 3.1, 29% of all social links observed during our crawl remained in our social
graphs after applying this limiting operation.

The average path lengths for each Facebook region, Renren, and three other OSNs
from the literature are shown in table 3.1. The average path length for each of the graphs
used in our study is <6, lending credence to the six-degrees of separation hypothesis
[124].

3.2.5 Clustering Coefficient

Clustering coefficient is a measure to determine whether social graphs conform
to the small-world principle [187]. It is defined on an undirected graph as the ratio
of the number of links that exist between a node’s immediate neighborhood and the
maximum number of links that could exist. For a node with \( N \) neighbors and \( E \) edges
between those neighbors, the clustering coefficient is \( \frac{2E}{N(N-1)} \). Intuitively, a
high clustering coefficient means that nodes tend to form tightly connected, localized
cliques with their immediate neighbors. The clustering coefficient for an entire graph
is the mean of all clustering coefficients for individual nodes.
Chapter 3. Social Graphs and User Interactions

Figure 3.2: Clustering coefficient of Facebook users as a function of social degree.

Figure 3.3: $k_{nn}$ of Facebook users as a function of social degree.

Table 3.1 shows that Facebook social graphs have average clustering coefficients (column label C. Coef) between 0.133 and 0.211, with the average over all 22 regional networks being 0.167. Facebook exhibits levels of clustering roughly equivalent to Orkut and Cyworld. In contrast, Renren exhibits the lowest levels of clustering of any graph in our study. All of the graphs in table 3.1 exhibit higher levels of clustering than either random graphs or random power-law graphs, which indicates a tightly clustered fringe that is characteristic of social networks [128].

Figure 3.2 shows how average clustering coefficient varies with social degree on Facebook and Renren. Users with lower social degrees have high clustering coefficients, again providing evidence for high levels of clustering at the edge of the social graph. This effect is particularly pronounced for Renren: low-degree users cluster more tightly than on Facebook, but high-degree users trend towards almost zero clustering. The tightly clustered fringe evident in Figure 3.2, combined with the low average path
Chapter 3. Social Graphs and User Interactions

lengths seen in table 3.1, is a strong indication that Facebook and Renren are small-world graphs [187].

3.2.6 \( k_{nn} \) and Assortativity

The Joint Degree Distribution (JDD) of a graph describes the likelihood of nodes of different degrees connecting to one another. JDD is approximated on large graphs by the degree correlation function \( k_{nn} \). For undirected graphs, \( k_{nn} \) is defined as the average degree of all nodes connected to nodes of a given degree.

Figure 3.3 shows how average \( k_{nn} \) varies with social degree on Facebook and Renren. The plot reveals very different behaviors for Facebook and Renren users. On Facebook, low-degree users tend to connect to other low-degree users. In contrast, low-degree Renren users tend to connect to high-degree, super-nodes. Users with degree > 100 behave similarly on both OSNs.

Closely related to JDD, the assortativity coefficient, \( r \), of a graph measures the probability for nodes in a graph to link to other nodes of similar degree. It is calculated as the Pearson correlation coefficient of the degrees of node pairs for all edges in a graph, and returns results in the range \(-1 \leq r \leq 1\). Assortativity greater than zero indicates that nodes tend to connect with other nodes of similar degree, while assortativity less than zero indicates that nodes connect to others with dissimilar degrees. The assortativity coefficients for the Facebook regional graphs and Renren, shown in Table 3.1, are
uniformly positive, implying that connections between high-degree nodes in our graphs are numerous. Our assortativity coefficient values are higher than those for Orkut and Cyworld (which is disassortative), but lower than Flickr and Twitter. [128, 137]

Our \( k_{nn} \) and assortativity results both indicate the presence of a well-connected “core” of high-degree nodes in our Facebook graphs. These nodes form the backbone of small-world graphs, enabling the highly clustered nodes at the edge of the graph (see Figure 3.2) to achieve low average path lengths to all other nodes.

3.2.7 Network Core Analysis

Previous studies of large, power-law graphs have shown that the densely connected core of high-degree nodes is necessary to hold the graph together [128]. When these nodes are removed the graph fractures, i.e. the nodes no longer form a single, large, connected component [37].

We analyze the core of our Facebook and Renren graphs by calculating the \( k\)-core of each graph. The \( k\)-core is calculated by sorting the nodes in a graph by degree and removing all nodes with degree \( \leq k \). Note that this removal process is iterative: removing node \( i \) with degree \( k \) may cause node \( j \) with degree \( k + 1 \) to lose an edge, thus causing the degree of \( j \) to drop to \( k \), at which point \( j \) is also cut from the graph. The \( k\)-core of the graph is the size of the connected component that remains after the removal process halts.
Chapter 3. Social Graphs and User Interactions

Figure 3.4 depicts the results of our $k$-core analysis on various large social graphs. Flickr quickly breaks apart as the core is removed. This indicates that the core of the Flickr graph is systemically important: as it gets removed, the outlier nodes that make up the bulk of the Flickr population quickly disconnect from the connected component.

Conversely, Facebook networks are highly resilient to super-node removal. Even when 20% of super-nodes are removed, the graphs still retain $\geq 75\%$ of nodes. The rest of our Facebook regions display characteristics in-between London and Egypt two. These results remain the same even if nodes are ordered using different importance metrics, such as distance-centrality or clustering-coefficient.

Renren displays characteristics in-between Facebook and Flickr. Initially, the size of the Renren connected component drops rapidly. This indicates that there are many users in the graph who only connect to popular, super-nodes. This observation can also be seen in Figure 3.3. However, as $k$ increases past 10, the rate of decline drops, showing that there is a dense, well connected core at the heart of the Renren graph.

Figure 3.4 demonstrates the importance of graph density in social networks. In denser graphs like Facebook and Orkut, the systemic importance of super-nodes decreases. There exist so many connections between disparate, low-to-average degree users that even in the total absence of high-degree nodes, the graph does not partition.

It should be noted that the work by Mislove et al. contains an error with regards to the $k$-core analysis of Orkut [128]. That study states that Flickr, YouTube, LiveJournal,
and Orkut all break apart quickly when subjected to $k$-core analysis. Our analysis of the same exact datasets confirms this result for Flickr, YouTube, and LiveJournal. However, as shown in Figure 3.4, Orkut does not exhibit the same brittle behavior as Flickr.

**Isolated Users.** User’s online friendship links often correspond closely with their offline relationships [106]. Thus, it is natural to assume that all users in a regional or school-based network would form a single connected component, based on their close geographic and cultural ties.

To examine whether this is true, we examine the users in the Renren PKU network. We have two different views of this population: the first comes from our complete crawl of Renren. The second comes from our breadth-first crawl of just the PKU network. The complete crawl is guaranteed to capture all PKU users, while the breadth-first crawl may miss users who technically belong to the PKU network, but are not directly connected to any other PKU users.

Surprising, we find that 23,430 (23.2%) of users in the PKU network have no friends in the PKU campus network, and are therefore disconnected from the PKU connected component. We refer to these as isolated users. To confirm these results, we measured the connected components of 9 other large university networks on Renren and discovered similar numbers of isolated users.

Figure 3.5 shows social degrees and total number of profile visits for these isolated users. 83% of isolated users have social degrees less than 10. In addition, 70% of
Chapter 3. Social Graphs and User Interactions

Figure 3.4: Percentage of nodes remaining in the connected component as super-nodes are removed from various social graphs.

Figure 3.5: Social degree and profile views for isolated users.

isolated users have fewer than 20 total profile visits, meaning their profiles are rarely browsed by others. Although it is not clear why isolated users do not have friends within the network, the vast majority of these users are both unpopular and low-degree. Therefore, they have little impact on our overall results.

3.2.8 Network Growth Over Time

Since Facebook and Renren users typically receive a Wall message shortly after joining Facebook, we use the earliest Wall post from each profile as a conservative estimate of each profile’s creation date. From this data, we plot the historical growth of the user population in our sample sets. The results plotted in Figure 3.6 confirm prior measurements of Facebook growth [168]. Note that Facebook opened its services to the general public in September 2006, which explains the observed exponential growth.
in network size. We see that an overwhelming majority (>80%) of profiles joined Facebook after it opened to the public in 2006.

Figure 3.7 shows the growth of the Renren PKU network over time. We observe a linear increase in the size of the PKU network. This trend makes intuitive sense for an affiliation-based network, i.e. there are a (roughly) constant number of new students admitted to PKU each year, a subset of which create Renren accounts. This contrasts with the Facebook data in Figure [168], which derives from networks that are open to the public.

Our work on quantifying temporal properties of social graphs is ongoing. We refer interested readers towards our study titled “Multi-scale Dynamics in a Massive Online Social Network” [202]. This paper focuses on a large, dynamic social graph based on the first 2 years of Renren’s existence.
3.3 Visible Interactions on Facebook

In the previous section, we characterize the static topological properties of Facebook and Renren. We now turn our attention to the interactions that occur between users on OSNs. In this section we focus on the 24 million Facebook interactions gathered by our crawler. These interactions are all visible, i.e. they represent conversations and comments on photographs. In the next section, we expand our discussion to include both visible and latent interactions.

The goal of our analysis of Facebook user interactions is to understand how many social links are actually indicative of active interactions between the connected users. Delving into this issue raises several specific questions that we will address here. First, is the level of interactions even across the user population, or is it heavily skewed towards a few highly-active users? Second, is the distribution of a user’s interactions across its friends affected by how active the user is? Finally, how does the interaction of users change over their lifetime, and do interactions exhibit any periodic patterns over time?

3.3.1 Visible Interaction Distribution Among Friends

We first examine the difference in size between each user’s entire friend list and the subset they actually interact with. For each Facebook user, we compute a distribution of
Chapter 3. Social Graphs and User Interactions

Figure 3.8: The distribution of users’ interaction among their friends, for different % of users’ interactions.

the user’s interaction events across the user’s social links. We then select several points from each distribution (70%, 90%, 100%) and aggregate across all users the percentage of friends these events involved. The result is a cumulative fraction function plotted in Figure 3.8. This is essentially a CDF showing corresponding points from each user’s CDF. We see that for the vast majority of users (∼ 90%), 20% of their friends account for 70% of all interactions. The 100% fraction line shows that nearly all users can attribute all of their interactions to only 60% of their friends. This proves that for most users, the large majority of interactions occur only across a small subset of their social links. This result allows us to answer our original question: are social links valid indicators of real user interaction? The answer is no, only a subset of social links actually represent interactive relationships.

We also want to understand if user interaction patterns are dependent on specific applications, and how interaction patterns vary between power users and less active users. Figures 3.9 and 3.10 organize users into groups of Top 50%, Top 10% and Top
Chapter 3. Social Graphs and User Interactions

Figure 3.9: Normalized Wall post distribution of the users with top total Wall interaction.

Figure 3.10: Normalized photo comments distribution of the users with top total photo interaction.

1% by their total level of activity. Each figure shows the distribution of incoming Wall posts and photo comments among friends for users within each group. The distribution of Wall posts in Figure 3.9 shows that the same distribution holds across all Wall users regardless of their overall activity level. In contrast, distribution of photo comments in Figure 3.10 varies significantly. The most active users only receive photo comments from a small segment (<15%) of their friends, while the majority of users receive comments from a third as many (∼5%) of their friends.

The low percentage of friends that comment on photos is notable because photo comments generally occur when friends are tagged in the same picture, implying a level of physical proximity in addition to social closeness. In our dataset, 57% of users self-identify with the photo albums they upload by tagging themselves in one or more photos. This fact lends credence to our argument that photo tags accurately capture real life social situations. The photo comment results indicate that users, even
highly social ones, show significant skew towards interacting with, and sharing physical
proximity with a small subset of their friends. Recent studies that focus on location-
based OSNs (e.g. Foursquare, Gowalla, etc.) further examine the correlations between
online friendship and geographic proximity [48, 153].

3.3.2 Distribution of Visible Interactions Across Users

Next, we examine how visible interaction activity is spread out across different
kinds of Facebook users. We plot Figure 3.11 to further understand the contribution of
highly interactive users to the overall interaction in the social network. For both Wall
posts and photo comments, we plot the contribution of different users sorted by each
user’s interaction in that application. We see that the top 1% of the most active Wall post
users account for 20% of all Wall posts and the top 1% of photo comment users account
for nearly 40% of all photo comments. Clearly, the bulk of all visible interactions on
Facebook are generated by a small, highly active subset of users, while the majority of users are significantly less active. This result lends credence to our assertion that not all social links are equally useful when analyzing social networks, since only a small fraction of users are actively engaged with the social network. This also identifies a core set of “power users” of Facebook, who could be identified to leverage their active opinions, ad-clicks, and web usage patterns.

Our next step is to quantify the correlation between users with high social degree and user activity. Figure 3.12 shows that there is a strong correlation between the two: half of all interactions are generated by the 10% most well-connected users. Nearly all interactions can be attributed to only the top 50% of users. This result confirms that a correlation between social degree and interactivity does exist, which is an important first step to validating our formulation of interaction graphs in Section 3.6.

### 3.3.3 Visible Interaction Distribution Across User Lifetimes

We now examine the temporal characteristics of visible interactions on Facebook. We pose the question: does individual user interactivity decline over time, perhaps as the novelty of social networking wears off? This potentially impacts our proposed use of interaction data to augment social graphs: if user activity wanes, then its relevance for assessing social link quality may drop as the information becomes less timely and
Chapter 3. Social Graphs and User Interactions

Figure 3.13: Average number of interactions per day for old and new Facebook users.

Figure 3.14: Deviations in pairwise interaction patterns on Facebook.

relevant. Using our records of user interactions over time, we study the gradual growth or decline in interaction events after users join Facebook.

Figure 3.13 shows users’ average number of interactions at different points in their lifetime. We divide the users in the 22 regional networks into 2 groups: the 10% oldest and the 10% newest users. Both user groups show very high average interaction rates in their first days in Facebook, supporting the hypothesis that users are most active when they first join. For the 10% oldest users (average lifetime of 20 months), we see a net increase in interaction rates over time, which we attribute to the “network effect” caused by more friends joining the social network over time (see Figure 3.6). Newer users (average lifetime of 3 weeks) show a different trend, where interactions drop to nearly nothing as the initial novelty of the site wears off. There are two possible interpretations of this. One view is that the oldest users are the original users who participated in Facebook’s growth, and are therefore a self-selected group of people that are highly
Chapter 3. Social Graphs and User Interactions

interested in social networks (and Facebook in particular). An alternative interpretation is many users who lose interest in Facebook over time close their accounts, leaving only active Facebook users from that time period.

3.3.4 Reciprocity

We now examine whether visible interactions on Facebook are reciprocated, *e.g.* if user $i$ writes on user $j$’s wall, will user $j$ respond in kind? Our intuition about polite social norms in the real world suggests that visible interactions should be reciprocated. However it is unknown whether real world social norms carry over into the digital realm.

In order to examine reciprocity on Facebook, we must correlate each user’s incoming and outgoing interactions. However, evaluating each user’s incoming and outgoing interactions is challenging, because Facebook data only records incoming events for a specific user, *i.e.* the event $i$ writing on $j$’s Wall is only recorded on $i$’s Wall. Crawling $j$ will not reveal this outgoing interaction.

Since we are limited to users within specific regional networks who have not modified their default privacy settings, we do not have access to 100% of the user population. This means we cannot match up all directed interaction events across users. A simple alternative is to examine only users whose friends are also completely contained in
our user population. Unfortunately, the high-degree of social connectivity in Facebook means that only about 400K users (4%) in our dataset fit this strict criterion.

A more reasonable way to study interaction reciprocation on Facebook is to only sample interactions that occur over social links that connect two users in our user population, i.e. ignore interactions with users outside our dataset. Rather than filtering on users as in the previous approach, this performs filtering on individual social links. Assuming that user interactions do not change significantly due to user privacy settings and geolocation, these sampled results should be representative.

After applying this sampling procedure, Figure 3.14 shows the length of the set resulting from the symmetric set difference of each user’s incoming and outgoing interaction partners plotted as a CDF. We refer to this metric as deviation. Intuitively, the deviation for each user counts the number of directed visible interactions that were not reciprocated with a direct reply. For 65% of Facebook users, all interactions are reciprocated, and for 90% of users only 5 interactions go unreciprocated. These results show that majority of visible interactions on Facebook are reciprocated.

3.4 Latent Interactions on Renren

Our analysis of visible interactions on Facebook reveals many important properties about the relative importance of edges in social networks. However, just looking at
visible interactions does not tell the whole story. The browsing behavior of OSN users, known as latent interactions, is also a vital component in understanding fundamental processes on social networks, such as how information disseminates through the graph.

In this section, we use measured data from the PKU regional network on Renren to analyze latent interactions and the role they play in OSNs. We use histories of visits to user profiles to capture latent interactions, and compare them to visible interactions from a variety of perspectives.

### 3.4.1 Popularity and Consumption

We begin by analyzing the distribution of latent interactions across the Renren user base. We define popularity as the number of views a user’s profile receives. Figure 3.15 shows the distribution of user popularity. As expected, popularity is not evenly spread across the population: only 518 people (1%) are popular enough to receive more than 10,000 views. Conversely, the majority of users (57%) exhibit very low popularity with fewer than 100 total profile views.

Figure 3.16 shows the average number of visits users receive on a daily basis. The distribution is fitted to a Zipf distribution of the form $\beta x^{-\alpha}$ where $\alpha = 0.71569687$ and $\beta = 697.4468225$. Popular users receive many more views per day: 141 users (0.2%) are viewed more than 20 times a day on average, with the most popular profile being viewed more than 600 times a day. Most users (85.5%) receive less than one visit per
day on average. This reinforces our finding that latent interactions are highly skewed towards a very popular subset of the population.

Finally, we examine whether the popularity of users corresponds to their profile viewing behavior. We define *consumption* as the number of other profiles a user views. Figure 3.17 plots the overlap between the top users sorted by popularity, and top users sorted by consumption. The graph shows that the top 1% most popular users have 9% overlap with the top 1% biggest consumers. These users represent a hard-core contingent of social network users who are extremely active. For the most part however,
users with high numbers of incoming latent interactions do not overlap with the people generating those interactions, e.g. profiles of celebrities are viewed by many users, but they are inactive in viewing others’ pages. This necessarily means that many (presumably average, low-degree) users actively visit others, but are not visited in return. We examine the reciprocity of latent interactions in more detail in Section 3.4.4.

### 3.4.2 Composition of Visitors

Next, we want to figure out the composition of visitors to user profiles. We pose two questions: first, what portion of profile visitors are repeat visitors? Second, are visitors mostly friends of the profile owner, or are they unrelated strangers?

We begin by addressing the first question. We calculate the percentage of repeated visitors for each profile, and report the distribution in Figure 3.18. Roughly 70% of users have fewer than 50% repeat visitors, meaning that the majority of visitors do not browse the same profile twice. This seems to indicate the long tail of latent interactions is generated by users randomly browsing the social graph.

Next, we take a closer look at repeat profile visits. Figure 3.19 shows the probability density function (PDF) of the interval time between repeat visits. The graph peaks on day 0, meaning that users are most likely to return to a viewed profile on the same day. We will examine the causes for this behavior more closely in Section 3.5. The probability for repeated views decreases as the time delta expands, except for a notice-
able peak at day 7. Interestingly, this shows that many users periodically check on their friends on a weekly basis. We confirmed that this feature is not an artifact introduced by our crawler or the use of RSS feeds by Renren users. Instead, we believe it may be due to the tendency for many users to browse their friends’ profiles over the weekend.

We now move on to our second question: what users are generating latent interactions, friends of the profile owner, or strangers. We define a stranger as any user who is not a direct friend of the target user. Renren’s default privacy settings allow users in the same campus network to browse each other’s profiles.

In order to answer our question, we calculate the percentage of visitors that are strangers and display the results in Figure 3.20. The results are fairly evenly divided: roughly 45% of users receive fewer than 50% of their profile visits from strangers. Or conversely, a slight majority of the population does receive a majority of their profile views from strangers.
We now pose the following questions: how does social degree impact the composition of profile visitors, and how far are they from the profile owner in the social graph? In Figure 3.21, we group the owners of profiles together by their social degree, and compute the average breakdown of their visitors into users who are friends (1-hop), friends-of-friends (2-hop) and other visitors (2+ hops). We see that for users with relatively few (< 100) friends, the large majority of their visitors are complete strangers, with very few friends-of-friends visiting. For well-connected users with 100–1000 friends, the majority of their visitors are direct friends, and also a significant number of friends-of-friends. Finally, for extremely popular users with more than 1000 friends, their notoriety is such that they start to attract more strangers to visit their profiles. These results confirm those from previous work that discovered many Orkut users browse profiles 2 or more hops away on the social graph [31].

Unlike friends, strangers do not build long-term relationships with profile owners. Intuitively, this would seem to indicate that repeat profile viewing behavior should fa-
vor friends over strangers. To investigate this we compute the average number of visits for strangers and friends for each profile and plot the distribution in Figure 3.22. Surprisingly, our results indicate that the repeat profile viewing behavior for friends and strangers is very similar, with friends only edging out strangers by a small margin. This result demonstrates that when considering information dissemination via latent interactions, the significance of non-friend strangers should not be overlooked.

3.4.3 Visits to Strangers’ Profiles

In Section 3.4.2, we observe that strangers account for a significant portion of profile visits. In this section we examine the question: how do people find and view non-friends’ profiles? There are several possible mechanisms that enable this behavior on Renren:

- **Featuring.** Renren automatically recommends the 100 most popular profiles in each network to other users in the same network.

- **Search.** Users may search for specific people within Renren. Personal attributes such as name and university are used by the Renren search engine to locate relevant users.

- **Social Links.** User’s may visit a friends-of-friends’ profile after seeing a link to it while browsing another profile. This is possible because Renren shows 24
random friends on each user’s profile, along with wall posts and comments that also originate from friends. Each user’s full list of friends is also accessible from their profile.

In this section, we focus on visits to strangers’ profiles via social links. This analysis is possible because our crawled data includes each user’s full friend list, as well as the approximate timestamp of all latent interactions. Because our dataset does not include click-through statistics from Renren’s featured links or search functionality, we are unable to directly examine the effects of these features on browsing behavior. However, the fact that visits from strangers are not confined to the top 100 most popular users in the PKU network indicates that profile featuring is not the primary driver behind stranger browsing behavior.

We use a time-based heuristic to infer when a Renren user is likely to have visited a strangers profile via social links. The behavior we are looking for has the following structure:

- A views B’s profile before A views C’s profile.

- B and C are friends.

- A may or may not be friends with B and C.

Intuitively, the inference we draw from this situation is that A visits C’s profile through a link on B’s profile. Although without asking people directly we cannot say for sure
Chapter 3. Social Graphs and User Interactions

that this is what happened, if the time between A’s visits to B and C is small, then social
links are the likely path a browser would have followed, especially if A and C are not
friends.

In order to examine non-friend profile browsing behavior, we create an ordered list
of profiles visited by each member of the PKU network. Profile visits from non-PKU
users are filtered out, since we do not have complete access to those users’ information.
Each user’s list of visits is ordered chronologically. Although Renren does not pro-
vide the exact timestamp for each profile visit, approximate timestamps can be inferred
based on when the visit was first observed by the crawler. New visits to a given profile
recorded by the crawler must have occurred in the time interval since the crawler previ-
ously visited the profile. We use these time intervals as approximate timestamps when
chronologically ordering profile visit events. Returning to our example scenario: if the
earliest possible time of A’s visit to B is smaller than the latest possible time of A’s visit
to C, we assume that A views B’s profile before A views C’s profile. As a sanity check,
we only infer correlation if the time delta between events is 24 hours or less, since this
is the interval between periodic crawls used for most of the Renren population.

Using this approximate ordering methodology, we isolate views of stranger’s pro-
files that immediately follow a visit to a friend of that stranger. Figure 3.23 shows the
number of visits to strangers’ profiles that fit our criteria, as a fraction of all visits to
stranger’s profiles. 62% users do not visit any strangers’ profiles via social links, indi-
Chapter 3. Social Graphs and User Interactions

**Figure 3.23:** Percentage of visits to strangers’ profiles through social links, as a fraction of total visits to stranger’s profiles.

**Figure 3.24:** Number of friends browsed before visiting a stranger’s profile.

cating that they use other Renren functionality (search, featured profiles) when browsing. Conversely, 4% of users only visit strangers’ profiles via social links.

Figure 3.23 also shows the percentage of visits to stranger’s profiles that traverse through a mutual friend. This represents directly browsing from a friend’s profile to a friend-of-a-friend (FoaF). Browsing to FoaF profiles accounts for the majority of visits to stranger’s profiles via social links. This indicates that when Renren users “surf” around profiles they do not stray far from their immediate social circle, *i.e.* users are likely to visit stranger’s profiles if they share a mutual friend.

Figure 3.24 explores the number of friends a user visits before viewing a stranger’s profile. 49% of visits to FoaF strangers occur after visiting only a single friend, indicating that most users browse directly from 1-hop to 2-hop neighbors. However, 10% of stranger browsing occurs after visiting $\geq 5$ mutual friends. This captures cases where a user browses many friends and notices that they all share a mutual friend that the user
herself is not friends with. These cases may indicate locations where edges are missing from the graph, or where the presence of high clustering leads to the creation of new social connections.

### 3.4.4 Reciprocity

As we showed in Section 3.3.4, social norms compel users to reply to one another when contacted via visible interactions. However, is this true of latent interactions? Since Renren users can see the list of recent visitors to their profile, it is possible for people to pay return visits those people. The question is: does visiting other user profiles actually trigger reciprocal visits?

As the first step towards looking at reciprocity of latent interactions, we construct the set of visitors who view each user profile, and the set of people who are visited by each user. Then, we compute the intersection and union of these two sets for every user. Intuitively, intersections include people who view a given user profile and are

---

**Figure 3.25:** Ratio of reciprocated latent interactions over total latent relationships.
Chapter 3. Social Graphs and User Interactions

Figure 3.26: Probability of reciprocated profile views over various time windows.

also visited by that user, i.e. the latent interactions are reciprocated. Unions contain all latent relationships for a given user, i.e. all users who viewed them, or they viewed. We calculate the Jaccard index for each user using their intersection and union set, and then plot the results in Figure 3.25. The ratio represents the number of reciprocated latent interactions divided by the total number of latent relationships. For more than 93% of users, fewer than 10% of latent relationships are reciprocated. This demonstrates that incoming profile views have little influence on user’s profile browsing behavior. This is surprising, especially considering the fact that users know that their visits to a profile are visible to its owner through the visitor history feature.

Next, we examine the time-varying characteristics of reciprocal profile visits for both strangers and friends. We compute the number of reciprocal visits that take place within $t$ days after the initial visit. Figure 3.26 shows the results for threshold $t$ values of 1 and 5 days plus the entire 90 days of our dataset. As we look at increasingly larger
window sizes, we see more profile visits being reciprocated. However, reciprocity remains low overall. Even across the entire measurement period, 73% of users receive no reciprocal page views from strangers, and 45% of users obtain no reciprocal page views from friends. This demonstrates that even with Renren’s visitor history feature, visiting other user profiles is not sufficient to generate reciprocal visits. Compared to strangers, friends have relatively higher probability of reciprocal visits.

We take a further step and quantify the lack of reciprocity for latent interactions. For a dataset of $n$ users, if user $i$ visits user $j$, then $v_{ij} = 1$, otherwise $v_{ij} = 0$. The reciprocity coefficient [53] is defined as

$$\frac{\sum_{i \neq j} (v_{ij} - \bar{v})(v_{ji} - \bar{v})}{\sum_{i \neq j} (v_{ij} - \bar{v})^2}$$

where $\bar{v} = \frac{\sum_{i \neq j} v_{ij}}{n(n-1)}$. The reciprocity coefficient is measured between -1 and 1, where positive values indicate reciprocity, and negative values anti-reciprocity. The reciprocity coefficient of profile visits on Renren is only 0.23. In contrast, reciprocity of visible comments on Renren is 0.49, and the reciprocity of visible interactions on Cyworld [53] is 0.78. Compared to these visible interactions, latent interactions show much less reciprocity.

### 3.4.5 Latent vs. Visible Interactions

We now compare the characteristics of latent and visible interactions on Renren. To understand the level of participation of different users (e.g. highly interactive users vs. more passive users) in both latent and visible interactions, Figure 3.27 plots the contribution of different users to both kinds of interactions. The bulk of all visible in-
Chapter 3. Social Graphs and User Interactions

Figure 3.27: Distribution of interactions, with users ordered from most to least interaction.

interactions can be attributed to a very small, highly interactive portion of the user base: the top 28% of users account for all such interactions. In contrast, latent interactions are more prevalent across the entire population, with more than 93% of all users contributing to latent interaction events. This confirms our original hypothesis that users are more active in viewing profiles than leaving comments. Given its widespread nature, this result also underscores the importance of understanding latent interactions as a way of propagating information across OSNs.

Figure 3.28 shows the distribution of interactions among users when ordered by degree. In contrast to Figure 3.27, visible interactions are not as tightly concentrated amongst high-degree users. Instead, the weak coupling between social degree and interactivity spreads the visible interactions more evenly throughout the population. However, the correlation between degree and visible interactivity is still greater than that
between degree and latent interactivity. Latent interactions have a significantly longer tail than visible interactions on Renren.

Next, we compare latent and visible interactions in coverage of friends. We compute for each user a distribution of their latent and visible interactions across their social links. We then aggregate across all users the percentage of friends involved in these events and plot the results in Figure 3.29. We see that roughly 80% of users only interact visibly with 5% of their friends, and no users interact with more than 40% of their friends. In contrast, about 80% of users view 20% or more of their friends’ profiles, and a small portion of the population views all of their friends’ profiles regularly. Thus, although not all social links are equally active, latent interactions cover a wider range of friends than visible interactions.

To get a sense of how many visible comments are generated by latent interactions, we examine the average number of comments per page view for a variety of pages.
on Renren, including profiles, diary entries, and photos. Figure 3.30 plots the results. Recall that along with visible comments, Renren keeps a visitor counter for each photo and diary entry. For diary entries and photos, the conversion rate is very low: 99% of users have less than 1 comment for every 5 photo views, and 85% people have less than 1 comment for every 5 diary views. This indicates that most users are passive information consumers: they view/read content and then move on without commenting themselves. In contrast, profile views have a higher conversion rate. Interestingly, 13% of users have a view/comment ratio greater than 1. This is because these users use profile comments as a form of instant messaging chat, leaving multiple responses and replies upon each visit.

Finally, we analyze the repeat activity frequency for latent and visible interactions on Renren. In particular, we want to examine the likelihood that users will repeatedly interact with the same page once they have viewed or commented on it once. Figure 3.31 plots the average number of interactions each user has with profile pages. 80% of users view a given profile <2 times. However, 80% of users leave 3.4 comments, almost twice the number of latent interactions. This result makes sense intuitively: for most types of data users only need to view them once to consume the data. However, comments can stimulate flurries of dialog on a given page, resulting in many consecutive interactions.
Chapter 3. Social Graphs and User Interactions

Figure 3.31: Average number of interactions per profile.

3.5 Factors Influencing Latent Interactions

As shown in Section 3.4.1, not all users in Renren are the target of equal numbers of latent interactions. Put another way, not all users have equally popular profiles. Although the popularity of some Renren users can be explained by their real-world celebrity status, this is not true for all popular users on Renren. This leads to the following question: what factors cause certain profiles to receive more latent interactions and become popular?

In this section, we analyze factors that may encourage users to visit Renren profiles. Quantifying how the actions of a profile owner impact the number of views received by that profile is an important step in understanding how OSN accounts become popular. If there are strong correlations between popularity and particular user actions (e.g. posting photos, writing diary entries, etc.), then this provides a roadmap for individuals looking to accrue popularity and promote themselves via social media. On the other hand, if
there are no correlations between user actions and popularity, then this would reveal
that there is no simple formula for a user to gain popularity on social media.

We examine the following factors and correlate them with profile popularity (i.e. number of received latent interactions):

- **Number of Friends.** Does social degree correlate with popularity?

- **Lifetime.** Are long lived accounts more likely to be popular than newer, less active accounts?

- **Shared Links.** Do users attract more visits if they frequently share links to other content?

- **Diary (Blog) Entries.** Are there correlations between diary update frequency or length and user popularity? Do diary entries generated by popular users receive more views and comments than those generated by less popular users?

- **Photos.** Does the popularity of a user’s photos correlate with their popularity?

- **Status Updates.** Does the quantity and length of user’s status updates correlate with their popularity? Are status updates from popular people more likely to receive comments from others?
Chapter 3. Social Graphs and User Interactions

3.5.1 Dataset and Methodology

To assess the impact of each factor on latent interactions, we need more precise data than what was gathered during our crawls of Renren. Thus, in November 2010, we contacted Renren Inc. and were given the anonymized information of 151,672 users in the PKU network. This data includes each user’s popularity score, as well as complete records of diary entries, photos, and status updates. Table 3.2 shows the useful information associated with each piece of user data, including number of unique visitors, comments from the data owner, and comments from other users. Length refers to the number of bytes of text in diary entries and status updates. “Shares” refers to the number of times users have posted links to the data object in friends’ news feeds. This dataset does not include the join-date of PKU users, or timestamps of interactions (for privacy reasons).

In order to examine the correlations between each factors and popularity, we divide users into 4 groups based on their popularity. Table 3.3 shows the popularity score

<table>
<thead>
<tr>
<th></th>
<th>Diary</th>
<th>Photo</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of unique visitors</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Number of “shares”</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Length (in bytes)</td>
<td>√</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Number of comments from owner</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Number of comments from others</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>

Table 3.2: Types of social data on PKU user interactions received directly from Renren in 2010.
Chapter 3. Social Graphs and User Interactions

<table>
<thead>
<tr>
<th>Popularity Groups</th>
<th>Group Sizes for Table 3.4</th>
<th>Group Sizes for Tables 3.5, 3.6, and 3.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-100</td>
<td>35,154</td>
<td>104,254</td>
</tr>
<tr>
<td>100-1000</td>
<td>16,047</td>
<td>27,651</td>
</tr>
<tr>
<td>1000-10000</td>
<td>9,686</td>
<td>18,468</td>
</tr>
<tr>
<td>&gt;10000</td>
<td>518</td>
<td>1,299</td>
</tr>
<tr>
<td>Total:</td>
<td>61,405</td>
<td>151,672</td>
</tr>
</tbody>
</table>

Table 3.3: Number of Renren users in each popularity group.

ranges for each group, as well as the number of Renren users in each group. The analysis in Table 3.4 uses the data from our original crawl of the PKU graph. The number of users grows for the analysis in Tables 3.5, 3.6, and 3.7 because they are based on the more complete dataset that Renren gave us in November 2010.

For each popularity group, we calculate the average value of each factors and display the results in Tables 3.4, 3.5, 3.6, and 3.7. All factors increase along with popularity, i.e. the most popular users also have the most friends, the oldest accounts, and generate the largest amounts of content/visible interactions.

Given the size differences between popularity group, and the average nature of the values in Table 3.4, 3.5, 3.6, 3.7, it is difficult to infer definite correlations between any one factor and popularity. To analyze these correlations more specifically, we leverage a technique from prior work [45] called Spearman’s rank correlation coefficient (Spearman’s $\rho$). Spearman’s $\rho$ is a non-parametric measure of the correlation between two variables that is closely related to Pearson’s correlation coefficient [108]. It is defined as $\rho = 1 - \frac{6 \sum (x_i - y_i)^2}{n(n^2 - 1)}$, where $x_i$ and $y_i$ are the ranks of two different features in
Chapter 3. Social Graphs and User Interactions

### Table 3.4: Average Value of Factors Associated with User Popularity

<table>
<thead>
<tr>
<th>Popularity</th>
<th>Friend</th>
<th>Lifetime</th>
<th>Shared Links</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-100</td>
<td>16 (0.15)</td>
<td>35 (0.55)</td>
<td>1 (0.5)</td>
</tr>
<tr>
<td>100-1000</td>
<td>131 (0.56)</td>
<td>423 (0.41)</td>
<td>43 (0.41)</td>
</tr>
<tr>
<td>1000-10000</td>
<td>401 (0.43)</td>
<td>792 (0.24)</td>
<td>155 (0.23)</td>
</tr>
<tr>
<td>&gt;10000</td>
<td>708 (0.02)</td>
<td>869 (0.02)</td>
<td>273 (-0.05)</td>
</tr>
<tr>
<td>All Users</td>
<td>112 (0.73)</td>
<td>263 (0.75)</td>
<td>39 (0.72)</td>
</tr>
</tbody>
</table>

The table shows the average number of friends, account lifetime, and number of shared links for our four popularity groups. “Shared links” refers to URLs shared by users, with the number of shared links and the Spearman’s ρ value shown in parentheses.

### Limitations

There may be other factors outside the scope of our measurements that contribute to user popularity. One possibility is that real-world celebrity status is the most important determining factor of online popularity. Unfortunately, we cannot quantify these factors at present. Recall that 100 of the most popular users in the university network are recommended to users by Renren. These 100 users account for fewer than 7.7% of the total users in the high popularity group, so the recommendation mechanism has limited impact on the high popularity group results.

### 3.5.2 User Account Characteristics

Table 3.4 shows the average number of friends, account lifetime, and number of shared links for our four popularity groups. “Shared links” refers to URLs shared by users.
Chapter 3. Social Graphs and User Interactions

PKU users, not received from friends. Lifetime is measured as the number of days in between a user joining and leaving Renren. Neither of these pieces of information is provided by Renren, and thus must be estimated. Join date can be approximated by the timestamp of the first comment received by a user, since the comment is likely to be a “welcome message” from a friend greeting the new user [190]. Because abandoned and inactive accounts can still receive comments, the best estimate of departure time is the timestamp of the last comment left by a user.

Although all factors in Table 3.4 exhibit high correlation with the low popularity and “all users” categories, this is an artifact of the tied ranks among the (numerous) low activity users. All of these users exhibit very low interactivity and social degree, thus leading to high levels of correlation. Previous work has observed similar artifacts when analyzing all users in a large OSN dataset [45].

For the two median popularity groups (100-1000 and 1000-10000) in Table 3.4, number of friends has the highest correlation with popularity. Users in these categories can be broadly defined as normal social network users. They are not celebrities; they simply use the OSN for its intended purpose of sharing information with friends. Account lifetime is a less important factor for users in the 1000-10000 popularity range, given the ease with which users can quickly amass hundreds of friends on OSNs.
### Chapter 3. Social Graphs and User Interactions

Table 3.5: Diary’s factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.

<table>
<thead>
<tr>
<th>Popularity</th>
<th>Amount</th>
<th>Visitors</th>
<th>Shared Links</th>
<th>Length</th>
<th>Owner’s Comments</th>
<th>Others’ Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-100</td>
<td>1 (0.53)</td>
<td>10 (0.54)</td>
<td>1 (0.51)</td>
<td>826 (0.53)</td>
<td>1 (0.52)</td>
<td>1 (0.52)</td>
</tr>
<tr>
<td>100-1000</td>
<td>7 (0.43)</td>
<td>220 (0.49)</td>
<td>4 (0.48)</td>
<td>17308 (0.42)</td>
<td>9 (0.51)</td>
<td>19 (0.52)</td>
</tr>
<tr>
<td>1000-10000</td>
<td>49 (0.35)</td>
<td>3759 (0.52)</td>
<td>59 (0.33)</td>
<td>83157 (0.31)</td>
<td>151 (0.44)</td>
<td>292 (0.46)</td>
</tr>
<tr>
<td>&gt;10000</td>
<td>142 (0)</td>
<td>35274 (0.17)</td>
<td>809 (0.08)</td>
<td>273399 (0)</td>
<td>668 (-0.01)</td>
<td>1347 (0.03)</td>
</tr>
<tr>
<td>All Users</td>
<td>9 (0.72)</td>
<td>807 (0.74)</td>
<td>15 (0.64)</td>
<td>16190 (0.72)</td>
<td>26 (0.7)</td>
<td>51 (0.72)</td>
</tr>
</tbody>
</table>

No factor has strong correlation with popularity for users in the high popularity group in Table 3.4. This is an important finding, as it shows popularity is not trivially gained simply by having lots of friends.

#### 3.5.3 Diary Entries

Table 3.5 shows the average value of various metrics associated with users’ diary entries, as well as Spearman’s $\rho$ for each metric. The “amount” column lists the number of diary entries per user, “visitors” is the number of unique visitors to each diary entry, and “length” is the number of characters in each diary entry. “Shared links” lists the number of times users have shared a link to a diary entry with friends. The two comment columns are the number of comments each diary entry receives from the entry’s owner and from other people. Intuitively, many of these metrics are intrinsically linked, e.g. a diary that is shared many times is also likely to receive many visitors, which can also result in many comments.
Chapter 3. Social Graphs and User Interactions

<table>
<thead>
<tr>
<th>Popularity</th>
<th>Amount</th>
<th>Visitors</th>
<th>Owner’s Comments</th>
<th>Others’ Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-100</td>
<td>2 (0.46)</td>
<td>8 (0.67)</td>
<td>1 (0.52)</td>
<td>1 (0.55)</td>
</tr>
<tr>
<td>100-1000</td>
<td>38 (0.41)</td>
<td>822 (0.49)</td>
<td>9 (0.51)</td>
<td>18 (0.49)</td>
</tr>
<tr>
<td>1000-10000</td>
<td>205 (0.33)</td>
<td>12827 (0.51)</td>
<td>125 (0.45)</td>
<td>228 (0.48)</td>
</tr>
<tr>
<td>&gt;10000</td>
<td>668 (0)</td>
<td>178094 (0.19)</td>
<td>575 (0.02)</td>
<td>1185 (0.09)</td>
</tr>
<tr>
<td>All Users</td>
<td>39 (0.73)</td>
<td>3242 (0.85)</td>
<td>22 (0.72)</td>
<td>41 (0.75)</td>
</tr>
</tbody>
</table>

Table 3.6: Photo’s factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.

<table>
<thead>
<tr>
<th>Popularity</th>
<th>Amount</th>
<th>Length</th>
<th>Owner’s Comments</th>
<th>Others’ Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-100</td>
<td>1 (0.53)</td>
<td>9 (0.53)</td>
<td>1 (0.52)</td>
<td>1 (0.52)</td>
</tr>
<tr>
<td>100-1000</td>
<td>24 (0.46)</td>
<td>550 (0.45)</td>
<td>25 (0.49)</td>
<td>39 (0.49)</td>
</tr>
<tr>
<td>1000-10000</td>
<td>164 (0.35)</td>
<td>4078 (0.34)</td>
<td>283 (0.35)</td>
<td>451 (0.36)</td>
</tr>
<tr>
<td>&gt;10000</td>
<td>420 (0)</td>
<td>11796 (0)</td>
<td>870 (-0.02)</td>
<td>1468 (0)</td>
</tr>
<tr>
<td>All Users</td>
<td>28 (0.74)</td>
<td>704 (0.74)</td>
<td>46 (0.7)</td>
<td>75 (0.72)</td>
</tr>
</tbody>
</table>

Table 3.7: Status’s factors associated with user popularity. Spearman’s $\rho$ is shown in parentheses.

For users with less than 10,000 popularity, all factors have high correlation with the popularity. However, for the high popularity group only the counter of visitors has obvious correlation with user popularity. One explanation for this correlation is that when people view a diary, they are also likely to visit the owner’s profile, thus boosting the user’s popularity. For popular users, no correlation exists between popularity and the number of diary entries or their length. Thus, producing copious or expansive diary entries is not enough to attract profile visits.
Table 3.8 shows Spearman’s $\rho$ for users’ photos and status updates. All columns are defined the same as for diary entries. Similar to diary, all factors show strong correlation with popularity in low and median popularity groups. Also similarly, only the visitor counter for photos has significant correlation for high popularity users. Again, this demonstrates that popularity is not simply gained by producing copious amounts of user-generated content (photos or status updates in this case).

Previous work observes that in Flickr, a photo’s visitor counter does not have high correlation with the number of comments or shares associated with that photo. However, the number of comments is strongly correlated with the number of shares [47]. We perform similar cross-correlation on our Renren data for diary entries and photos and show the results in Table 3.8.

In contrast to Flickr [47], the number of visitors has high correlation with the number of shared links and comments (both from the owner and other users) for diary
entries and photos on Renren. This may be due to the more social nature of Renren as compared to Flickr, i.e. all Renren users belong to the social network and the average number of friends is high, versus Flickr where not all users leverage the website’s social capabilities. Similar to Flickr, shares on Renren positively correlate with comments.

Unsurprisingly, the highest correlations occur between comments from the owner and other people, stemming from the use of comment areas to hold bi-directional conversations. As shown in Table 3.9, this trend holds across all Renren features, over all popularity groups.

### 3.6 Interaction Graphs

In Sections 3.3 and 3.4 we demonstrate that not all social links represent active social relationships. Visible interactions on Facebook and Renren are concentrated within a subset of the population, and users rarely engage with >50% of their “friends.” Similarly, not all social edges receive an equal share of latent interactions.
These results imply that social links, and the social graphs they form, are not accurate indicators of social relationships between users. This has profound implications on applications that leverage social graphs. We propose a new model that more accurately represents social relationships between users by taking into account real user interactions. We call this new model an interaction graph.

We begin this section by formally defining two different types of interaction graphs: visible interaction graphs, and latent interaction graphs. Next, we build visible interaction graphs using our Facebook dataset. We demonstrate how drastically the structure of interaction graphs can change when different notions of interaction “freshness” are considered, and compare the resulting visible interaction graphs to the Facebook social graph. Next, we build latent and visible interaction graphs using Renren dataset. We compare the structural properties of latent, visible, and social graphs and examine the time varying properties of latent interaction graphs.

\subsection{Definition of Interaction Graphs}

To better differentiate between users’ active friends and those they merely associate with by name, we introduce the concept of an interaction graph. We define an interaction graph as a graph $G'(n, t) = (V, I)$. A social graph $G = (V, E)$ and interaction graph $G'$ share the same set of vertices $V$. However, $G$ uses edge set $E$ (the social links
between users) while \( G' \) uses edge set \( I \) (the interactions between users). Although \( I \) is a multiset, \( G' \) is not a multigraph: duplicate edges are simply filtered out.

An interaction graph is parameterized by two constants \( n \) and \( t \). These constants filter out edges from the set of interactions \( I \). \( n \) defines a minimum number of interaction events for admitting each edge \( i_{u,v} \), such that \(|i_{u,v}| \geq n\). For example, if \( n = 2 \), then an edge between users \( u \) and \( v \) will only exist in the interaction graph if there are 2 or more interactions occur between \( u \) and \( v \). \( t \) stipulates a window of time during which interactions must have occurred. Taken together, \( n \) and \( t \) delineate an interaction rate threshold. Intuitively, an interaction graph is the subset of the social graph where for each edge, interactivity between the edge’s endpoints is greater than or equal to the rate stipulated by \( n \) and \( t \).

We propose two types of interaction graphs: visible interaction graphs and latent interaction graphs. Each type of graph is constructed using the corresponding type of interactions to denote edges. Visible and latent interaction graphs differ in two important ways. First, in a visible interaction graph, \( I \subseteq E \), i.e. the edges in the interaction graph are a subset of the edges in the corresponding social graph. This property arises because most social networks (including Facebook and Renren) only allow friends to write messages and comments on user’s profiles. This invariant does not hold for latent interactions: as shown in Section 3.4.2, \( \approx 50\% \) of visits to user’s profiles come from strangers.
Chapter 3. Social Graphs and User Interactions

The second difference between visible and latent interaction graphs are that we formulate visible graphs as undirected, while latent graph are directed. As shown in Section 3.3.4, the majority of visible interactions are reciprocated, hence it makes sense to model them as undirected links. A user’s interaction degree is the number of nodes adjacent to \( u \) in \( G' \). The equivalent metric on \( G \) is a node’s degree, or \( \text{deg}(u) \). Conversely, as shown in Section 3.4.4, latent interactions are rarely reciprocated. Thus, we model latent interactions as directed edges. We define latent interaction in-degree of a node as the number of visitors who have visited that user’s profile, while latent interaction out-degree is the number of profiles that user has visited.

Interaction graphs differ from inference graphs because visible and latent interactions between users are explicit. Recall that the interactions we are analyzing come from Wall posts, photo comments, and profile browsing events. These interactions record actual events (with a source, a destination, and a timestamp) generated by users, without any ambiguity. In contrast, inference graphs model edge relationships between nodes using the concept of “similarity,” e.g. nodes that share similar meta-data attributes should be connected via an edge [178].

Our formulation of interaction graphs uses an unweighted graph. It is feasible to re-parameterize the interaction graph such that the interaction thresholds \( n \) and \( t \) no longer cull links, but instead imparts a weight to each edge. We do not attempt to derive
a weight scheme for interaction graphs analyzed in this work, but leave exploration of this facet of interaction graphs to future work.

### 3.6.2 Visible Interaction Graphs on Facebook

We begin our analysis by focusing on visible interaction graphs derived from our Facebook dataset. We begin by examining the interaction rate parameters \( n \) and \( t \), focusing on how the high-level structure of visible interaction graphs changes as these parameters are varied. After choosing some representative values for \( n \) and \( t \) we compare the structural properties of the visible interaction graphs versus the corresponding Facebook social graphs.

**Choosing Interaction Rate Parameters.** The simplest formulation of parameters \( n \) and \( t \) is to consider all interactions over the entire lifetime of Facebook (\( t = 2004 \) to the present, \( n = 1 \)). We will refer to the interaction graph corresponding to this
parameterization as the full interaction graph. We also consider additional interaction graphs that restrict \( t \) and increase \( n \) beyond 1. This allows time and rate thresholds to be applied to generate interaction graphs appropriate for specific applications that have heterogeneous definitions of interactivity.

Figure 3.32 shows the size of the connected components for interaction graphs as \( t \) and \( n \) change. Intuitively, higher \( n \) filters out more edges since user pairs need to have a greater number of pairwise interactions. Similarly, smaller \( t \) also results in fewer edges, since the span of time during which interactions must occur is tighter. This figure is based on data for the year 2007, i.e. 2 months refers to interactions occurring between November 1 and December 31, 2007.

As expected, lower \( n \) and larger \( t \) are less restrictive on links, therefore allowing for more nodes to remain connected. Based on Figure 3.32, we choose several key interaction graphs for further study, including those with \( n \geq 1 \) in the 1 year, 6 months, and 2 months time periods. These three graphs each include connected components that contain a majority of all nodes, and are amenable to graph analysis. For the remainder of this section, we will only consider interaction graphs for which \( n \geq 1 \).

We now take a closer look at interaction graphs and compare them to full social graphs. We look at graph connectivity and examine properties for power-law graphs, small-world clustering, and scale-free graphs.
Chapter 3. Social Graphs and User Interactions

Social vs. Interaction Degree. Figure 3.33 displays the correlation between social degree and interaction degree for the full visible interaction graph. The error bars indicate the standard deviation for each plotted point. Even with this “least-restricted” visible interaction graph, it is clear that interaction degree does not scale equally with social degree. If all Facebook users interacted with each of their friends at least once then this plot would follow a 45 degree line. This is not the case, confirming once again the disparity between friend relationships and active, social relationships.

It is interesting to note that the apparent upper limit on interaction relationships on Facebook corresponds with Dunbar’s Number [61]. Dunbar’s Number is the theorized upper limit on the number of active social relationships the human brain can manage concurrently. The number is usually estimated to be between 100 and 150, which matches up with the asymptotic behavior of the average Facebook user, regardless of the user’s social degree. Thus, although users can accumulate thousands of friends on OSNs, their ability to interact with those friends is restricted by fundamental cognitive limits.

Degree Distribution. Figure 3.34 plots the degree CDFs of the four visible interaction graphs and the Facebook social graph. The visible interaction graphs exhibit a larger percentage of users with zero friends, and reach 100% degree coverage more rapidly than the social graph. This is explained by the uneven distribution of interactions between users’ friends. Referring back to Figure 3.8, we showed that visible
interactions are skewed towards a fraction of each user’s friends. This means many links are removed from the social graph during conversion into a visible interaction graph. As a consequence, many weakly connected users in the social graph have zero interaction degree, while highly connected users in the social graph are significantly less connected in the visible interaction graph.
Chapter 3. Social Graphs and User Interactions

The visible interaction graphs exhibit stronger power-law scaling than the social graph. Figure 3.35 (a) shows the alpha values for the four interaction graphs compared to the social network. The error bars above the histogram are the fitting error of the estimator [55]. The fitting error for the interaction graphs is lower than that for the social graph, indicating that the interaction graphs exhibit more precise power-law scaling. As the link structure of the interaction graphs gets restricted, alpha rises, corresponding to an increased slope in the fitting line. This property is visualized in Figure 3.34 as a lower number of high-degree nodes in the most constrained interaction graphs. These results are further validated by studies on LiveJournal that have uncovered degree distribution and power-law scaling characteristics very similar to those depicted here for Facebook interaction graphs [128].

**Clustering Coefficient.** Figure 3.35 (b) shows that average clustering coefficient drops as visible interaction graphs become more restricted. Figure 3.36 depicts average
clustering coefficients as a function of interaction degree. As with the Facebook social graph, there is more clustering among nodes with lower degrees. However, the overall amount of clustering is reduced by over 50% across all interaction graphs.

To understand why clustering coefficient drops in the visible interaction graphs, we examine which edges are removed from the social graph. Figure 3.37 depicts the percent of edges removed from different visible interaction graphs. Edges are grouped together based on how many triangles they are part of in the original social graph. For example, edges that are not part of any triangles fall into the 0 bucket. In contrast, an edge that forms one side of twenty unique triangles will fall into the 20 bucket. Edges that complete many triangles are more systemically important for increasing average clustering. The “Random Edge Removal” line acts as an experimental control: in this scenario, \( x \) edges are randomly removed from the social graph, where \( x \) is the difference in edges between the social graph and the full interaction graph.

Figure 3.37 reveals that edge importance (in terms of triangles) does correlate with how likely that edge is to be retained in the visible interaction graphs. Low importance edges (e.g. 0 or 1 triangles) are about 20% more likely to be removed than high importance edges. This contrasts with random removal, where edges of all types are equally likely to be removed (all lines are jagged when the number of triangles >60 because such high importance edges are rare). This result indicates that edges which complete many triangles are more likely to correspond to active social relationships.
However, in absolute terms, all edges are >50% likely to be removed in the visible interaction graphs, irrespective of importance. Thus, although visible interaction graphs retain a higher percentage of important edges than random chance predicts, a large number of triangles are still being severed. This problem is particularly acute for edges that complete many triangles, since these edges are more vital for high clustering coefficients, and much rarer.

Taken together, the reduced clustering coefficients and the higher path lengths that characterize Facebook visible interaction graphs indicates that they exhibit significantly less small-world clustering. In order for the visible interaction graphs to cease being small-world, the average clustering coefficient would have to approach levels exhibited by a random graph with an equal number of nodes and edges. This number can be estimated by calculating $K/N$, where $K$ is average node degree and $N$ is the total number of nodes [187]. For the Facebook social graph, $K = 76.54$. We can estimate from this that an equivalent random graph would have an average clustering coefficient of $7.15 \times 10^{-6}$. $K$ is smaller for our visible interaction graphs, therefore the estimated clustering coefficient for equivalent random graphs will be smaller as well. These estimated figures are orders of magnitude smaller than the actual clustering coefficients observed in our social and visible interaction graphs, thus confirming that they both remain small-world. The conclusion that Facebook visible interaction graphs exhibit less small-world behavior than the Facebook social graph has important implications for all
social applications that rely on this property of social networks in order to function, as we will show in Section 3.7.

**Assortativity.** Figure 3.35 (c) shows the relative assortativity coefficients for Facebook social and visible interaction graphs. Assortativity measures the likelihood of nodes to link to other nodes of similar degree. Since visible interaction graphs restrict the number of links high degree nodes have, this causes the degree distribution of visible interaction graphs to become more homogeneous. This is reflected by the assortativity coefficient, which rises commensurately as the visible interaction graphs grow more restricted.

**Average Path Length.** Figure 3.35 (d) shows the average radius, diameter, and path lengths for all of the visible interaction graphs, as well as for the social network. These measures all display the same upward trend as the visible interaction graphs become more restricted. This makes intuitive sense: as the average number of links per node and the number of high-degree “super-nodes” decreases (see Figure 3.34) the overall level of connectivity in the graph drops. This causes average path lengths to rise, affecting all three of the measures presented in Figure 3.35 (d).

### 3.6.3 Latent Interaction Graphs on Renren

We now move on to examining the characteristics of the Renren social, visible, and latent interaction graphs. We use data from the PKU regional network, since we do not
### Table 3.10: Topology measurements for latent interaction, visible interaction, and social graphs on Renren.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Social Graph</td>
<td>753,297</td>
<td>0.18</td>
<td>0.23</td>
<td>3.64</td>
</tr>
<tr>
<td>Visible Interaction Graph</td>
<td>27,347</td>
<td>0.05</td>
<td>0.05</td>
<td>5.43</td>
</tr>
<tr>
<td>Latent Interaction Graph</td>
<td>240,408</td>
<td>0.03</td>
<td>-0.06</td>
<td>4.02</td>
</tr>
</tbody>
</table>

We construct latent interaction graph from our Renren data using profile views as the latent interactions. Similarly, we use user comments as the visible interactions to construct a visible interaction graph for Renren. We only consider interactions that occur between users in the PKU network, as it is possible for interactions to originate from or target users outside the network for whom we have limited information. Also, because non-friend strangers can view user’s profiles, the latent interaction graph contains edges between users who are not friends in the social graph.

In this section, we set $n = 1$ and $t$ to be the 90-day period during which our crawler collected latent interactions from Renren. In the next section we consider alternative interaction threshold ranges.
Figure 3.38: CCDF of node degree for latent interaction graph, visible interaction graph, and social graph.

Degree Distribution. Figure 3.38 plots the CCDFs of node degree for the three types of graphs. Since the latent interaction graph is directed, we plot both in-degree and out-degree. In Section 3.4.5 we show that latent interactions are more prevalent than visible interactions. This is reflected in the relative number of edges in the two interaction graphs, as shown in Table 3.10. This also leads to nodes in the latent graph having a noticeably higher degree distribution in Figure 3.38. However, neither of the interaction graphs have as many edges as the raw social graph, which leads to the social graph having the highest degree distribution. Interestingly, because a small number of Renren users are frequent profile browsers, i.e. they like to visit a large number of profiles (far greater than their circle of friends), the distribution of latent out-degrees flattens out at the tail-end and never approaches 0%.

Clustering Coefficient. Table 3.10 shows that the average clustering coefficient is 0.03 for the latent interaction graph, and 0.05 for the visible interaction graph, which
Chapter 3. Social Graphs and User Interactions

are both much less than that of the social graph. This is because not all social links are accurate indicators of active social relationships, and these links with no interactions are removed in interaction graphs. This produces loose connections between neighbors, and low clustering coefficients in these graphs. A portion of the latent interactions to a profile are from non-friend strangers who randomly browse the network. Thus, links between visitors in the latent interaction graph are less intensive than friends exchanging messages, which further lowers the clustering coefficients in latent interaction graphs.

Figure 3.39 further explores the distribution of clustering coefficients in our three graphs. As noted above, the sparsity of edges in the latent and visible graphs results in much less clustering when compared to the social graph. In fact, the line for the visible graph stops at 87 because that is the maximum node degree in the entire graph. In addition to the overall sparseness of the latent and visible graphs, these graphs also lose the tightly clustered fringe exhibited by the social graph. This indicates that the interaction graphs are comparatively less “small-world” than the social graph.

Assortativity. Table 3.10 shows that the Renren latent interaction graph is slightly disassortative. This makes sense intuitively, as latent interactions are highly skewed towards a small subset of extremely popular users. In contrast, the other two graphs are both assortative, with the social graph being more so. This result contrasts with previous studies in which the interaction graph was more assortative than the social graph [190].
Figure 3.40: $k$-core analysis of different graph types.

**Average Path Length.** The average path length of the latent interaction graph is between that of the visible interaction graph and the social graph. As the average number of links per node and the number of high-degree “super-nodes” decreases, the overall level of connectivity in the graph drops. This causes average path lengths to rise, especially in the visible interaction graph. This further corroborates the weakening of “small-world” properties evinced above with regards to clustering coefficient.

**Network Core Analysis.** Figure 3.40 shows the $k$-core size of social, latent interaction, and visible interaction graphs ($k$-core is defined in Section 3.2.7). In the social graph, the core size remains relatively stable until $k > 60$, at which point the number of nodes drops rapidly. This threshold separates the fringe of the network from the strongly connected core. The Renren social graph exhibits a larger fringe than other large social graphs, such as Cyworld ($k = 38$) [53] and MSN Messenger ($k = 20$) [111].
In contrast to the social graph, the core size for interaction graphs rapidly declines as $k$ increases. Neither interaction graph exhibits a well-defined inflection point between strong and weak core connectivity. The difference between social and interaction graph $k$-core results stems from the lack of high-degree, super-nodes in the interaction graphs.

### 3.6.4 Comparing Visible and Latent Interaction Graphs Over Time

In this section, we explore the evolution of interaction graphs on Renren. In December 2011, we contacted Renren and obtained the anonymized interactions and profile visits for the 61,405 PKU users from September 2009 to August 2010. These interaction records are the most complete dataset in our corpus: they include instant-messages, message board posts, diary entries, photos, and status updates. Each interaction and profile visit includes a sender, a receiver, and a timestamp. In total, 532,326 interactions and 11,875,247 visits were given to us. Figure 3.41 shows the number of latent and visible interactions per month in this dataset. As expected, latent interactions outnumber visible ones by an order of magnitude each month. The data also exhibits a noticeable seasonal trend: during summer break in August, when PKU is not in session, user activity drops significantly.

Using this data, we construct time-varying visible and latent interaction graphs. Recall that we parameterize interaction graphs using two variables: $n$, the minimum number of interactions, and $t$, the time span during which interactions must occur.
our Renren temporal experiments, we set $n = 1$ and establish 6 different time ranges for $t$, each of which contains 2 month’s worth of interactions. As specific in Section 3.6.1, latent graphs are directed, while visible graphs are undirected. Intuitively, these time-based interaction graphs reflect user’s changing communication patterns. In order for a given edge to appear in multiple snapshots, the users in question must communicate at least once every two months. The set of 61,405 PKU users examined in this section remains the same as in Section 3.6.3.

Figure 3.42 shows the number of nodes and edges per snapshot. Solid lines denote number of edges over time, while lines with points denote nodes over time. The number of nodes in each snapshot reveals how many users generated visible and latent interactions during each two month period. Around 17% of PKU users generate at least one latent interaction per month, while only $\approx 4\%$ of PKU users generate visible interactions per month. The number of edges in each graph is around one order of magnitude less than the total number of interactions seen in Figure 3.41, since many interactions occur between duplicate pairs of users. The seasonal fall off in interactions heading into summer break translates directly into fewer nodes and edges in the last two snapshots.

**Resemblance.** One important measure of time-varying graphs is *resemblance*: the fraction of links which remain unchanged from one snapshot to the next. Resemblance is defined as $r_t = \frac{|S_t \cap S_{t+1}|}{|S_t|}$, where $S_t$ is the set of links in the snapshot $t$ [179]. The
value of $r_t$ is between 0 and 1. If $r_t = 1$, all links in snapshot $t$ exist in snapshot $t + 1$. If $r_t = 0$, none of links in the snapshot $t$ persist in snapshot $t + 1$.

We plot the resemblance of the evolving visible and latent Renren interaction graphs in Figure 3.43. In general, latent interaction graphs have more resemblance than the visible graphs, indicating that user’s browsing behaviors are more stable than their communication patterns. However, this greater stability is relative; both types of graphs have resemblance of $<0.5$, meaning that the majority of interacting pairs change every two months. Most of the visible interaction graphs have similar resemblance ($\approx 0.21$), with only the final snapshots having low resemblance (0.1). The resemblance of the
Chapter 3. Social Graphs and User Interactions

Figure 3.44: Structural properties over time for visible and latent interaction graphs.

latent interaction graphs is stable ($\approx 0.3$). This reveals an interesting seasonal trend: PKU students visibly interact with different people when they are away on summer break, possibly friends from home. However, they still use Renren to browse the profiles of their friends from college, presumably to keep up-to-date on their summer time activities.

Structural properties. We now examine how the structural properties of evolving interaction graphs change over time. Figure 3.44 shows the average degree, average clustering coefficient, and average path length for visible and latent interaction graph snapshots over time. The average degree results in Figure 3.44(a) exactly track the seasonal fluctuations in edges seen in Figure 3.42, although this is difficult to see, since Figure 3.42 is in log-scale. Figures 3.44(b) and 3.44(c) also exhibit the same seasonal patterns, with clustering reducing and path lengths rising as the graphs become sparser.
Chapter 3. Social Graphs and User Interactions

Our results from Renren indicate that seasonal factors can drastically affect the properties of interaction graphs. These results contrast with the findings from prior work, which showed that the properties of visible interaction graphs on Facebook were stable over time [179]. Viswanath et al.'s results differ from ours because they focus on a regional network, while we are focused on an academic network. Students exhibit strong trends that are tied to the school year, while an entire American city is not as strongly synchronized seasonally.

3.7 Applying Interaction Graphs

When social graphs are used to drive simulations of socially-enhanced applications, changes in user connectivity patterns can produce significantly different results for the evaluated application. Given the lack of publicly available social network topological datasets, many current proposals either use statistical models of social networks based on prior measurement studies [120, 187, 200], or bootstrap social networks using traces of emails [73].

However, the hypothesis of this section is that the validation of socially-enhanced applications requires a model that explicitly takes user interactions into account. To validate how much impact the choice of graph model can make on socially enhanced applications, we implement simulations of three well-known socially-enhanced sys-
Chapter 3. *Social Graphs and User Interactions*

tems [50, 73, 200], and compare the effectiveness of each system on real social graphs, and real interaction graphs derived from our Facebook and Renren measurements.

Note that we are not advocating for social network providers to begin revealing visible and latent interactions to third-parties. Obviously, user’s interactions and browsing behavior are privacy sensitive information. Our purpose in this section is simply to point out that many current social applications implicitly rely on user interactions, and that evaluating these applications on models derived from static graph topologies may lead to results that do not conform to reality.

3.7.1 RE: Reliable Email

“RE” [73] is a white-listing system for email based on social links that allows emails between friends and Friends-of-Friends (FoFs) to bypass standard spam filters. Socially-connected users provide secure attestations for each other’s email messages while keeping users’ contacts private. The key advantage of RE is that it works automatically based on social connectivity data: users do not have to take the time to manually create white-lists of authorized senders.

**Expected Impact.** The presence of small-world clustering and scale-free behavior in social graphs translate directly into short average path lengths between nodes. For RE, this means that the set of friends and FoFs that will be white-listed for each given
user is very large. In this situation, a single user who sends out spam email is likely to be able to successfully target a very large group of recipients via the social network.

In contrast, RE that leverages interaction graphs should not experience as high a proliferation of spam, given an equal number of spammers. As an example, Figure 3.45 shows the number of friends of friends per user in the Facebook New York social graph and full visible interaction graph. The size of each user’s friend of friend set is reduced by about an order of magnitude in the visible interaction graph. Similar results are present in the other Facebook regional graphs. The reduced size of the FoF set should limit the dissemination potential for spam, while still maintaining the key advantage of RE, i.e. users do not need to manually enumerate white-lists of senders.

Results. For each social graph and interaction graph, we randomly choose a percentage of nodes to act as spammers. In the RE system, all friends and FoFs of the spammer will automatically receive the spam due to white-listing. All experiments were repeated ten times and the results averaged.
Chapter 3. Social Graphs and User Interactions

Figure 3.46: The percent of users receiving spam as the number of spammers increases on different social graph topologies.

This experiment leads to Figure 3.46, which plots the percentage of users in each graph receiving spam versus the percentage of users who are spamming. On the Facebook social graph spam penetration quickly reaches 90% of the users in the connected component. A similar trend is observed on Renren: spam penetration quickly reaches >50% of the user base, when only 1% of users are spamming.

In contrast, spam penetration is reduced by up to 40% when RE is run on the Facebook visible interaction instead of the social graph. The gains on Renren are even larger: spam penetration is reduced by almost 50% when RE uses the latent interaction graph, and 66% when using the visible interaction graph.

The secondary benefit of using RE on visible interaction graphs is that spammers are naturally excluded from the graph. Intuitively, honest users are unlikely to exchange messages with spammers. Hence the visible interaction graph should have few edges connection spammers and honest users, which prevents spammers from be-
ing whitelisted by RE. We empirically confirmed this phenomenon during prior work when we examined a social e-commerce marketplace: scammers were almost totally excluded from the visible interaction graph [166].

Although the performance of RE improves when the algorithm is run on interaction graphs, this does come at a cost: by shrinking the size of each user’s whitelist, more email will need to be evaluated by traditional spam filters. Fortunately, our evaluation results provide a roadmap for managing the trade-off between resiliency to spam from compromised accounts, versus higher load on the spam filters. By choosing a different underlying graph model, administrators looking to deploy RE can choose a point in the spectrum that suits their individual needs.

Unfortunately, the Achilles’ heel of RE is that friend’s accounts can be compromised by attackers. Spam sent from compromised accounts will successfully disseminate due to RE’s white-listing of friendly accounts. This shortcoming is equally damaging when using RE on social and interaction graphs.

### 3.7.2 SybilGuard

A Sybil attack [60] occurs when a single attacker creates a large number of online identities that can collude together and grant the attacker significant advantage in a distributed system. Sybil identities can work together to distort reputation values, out-vote legitimate nodes in consensus systems, or corrupt data in distributed storage systems.
Chapter 3. Social Graphs and User Interactions

SybilGuard [200], SybilLimit [199], SybilInfer [57], and SumUp [174] are all algorithms for performing decentralized detection of Sybil nodes on social graphs. At their core, all of these algorithms are based on two assumptions of Sybil and normal user behavior:

1. Attackers can create unlimited Sybils and edges between them. Edges between Sybils are beneficial since they make Sybils appear more legitimate to normal users.

2. The number of edges between Sybils and normal users will be limited, since normal users are unlikely to accept friend requests from unknown strangers.

Under these assumptions, Sybils tend to form tight knit clusters, since the number of edges between Sybils is greater than the number of edges connecting to normal users. We refer to edges between Sybils as *Sybil edges*, while edges connecting Sybils and normal users are called *attack edges*.

Sybil detection algorithms identify Sybil clusters by locating the small number of edge cuts that separate the Sybil region from the social graph. SybilGuard, SybilLimit, and SybilInfer all leverage specially engineered random walks for this purpose, while SumUp uses a max-flow approach. Although all of these algorithms are implemented differently, it has been shown that they all generalize to the problem of detecting communities of Sybil nodes [181].
In this section we focus on SybilGuard, as it is representative of this class of Sybil detection algorithms. In SybilGuard, each node in the social network creates a persistent routing table that maps each incoming edge to an outgoing edge in a unique one-to-one mapping. To determine whether to accept a “suspect” node \( v \) as a real user, a “verifier” node \( u \) initiates \( n \) random walks of length \( w \) on the social graph. Node \( v \) also initiates \( n \) random walks of length \( w \). \( u \) accepts \( v \) if some predefined percentage of the random walks intersect. \( w \) is the most important parameter in SybilGuard: as \( w \) grows, the number of Sybils that will be erroneously accepted grows. Thus, it is beneficial for \( w \) to be small.

**Expected Impact.** The success of SybilGuard relies on the premise that Sybil identities cannot easily establish trusted social relationships with legitimate users, and hence have few “attack edges” in the social network. In particular, SybilGuard requires connected users to exchange encryption keys. We believe that typical social connections in social graphs do not represent this level of trust. Given our results that demonstrate...
most Facebook friend pairs do not even interact, it seems unreasonable to assume that most friend pairs have the requisite level of trust to exchange secure keys.

Instead, we expect that the visible interaction graph is a closer approximation to the representation of trusted links that SybilGuard would observe in reality. Unfortunately, under these conditions, we expect the effectiveness of SybilGuard to decrease. SybilGuard’s functionality is dependent on the fast mixing behavior of graphs. [129] provide an overview of the mixing behavior of “trusted” social networks (e.g. physics co-authorship, Enron emails), “untrusted” social networks (e.g. Facebook), and interaction graphs. Their results confirm that the mixing properties of interaction graphs closely resemble trusted social networks, and that both are slow mixing.

**Results.** For our experiments, we implement the SybilGuard algorithm on our Facebook social and visible interaction graphs and measure the percentage of random walks that successfully intersect as $w$ increases. For each graph and each value of $w$ we chose 25000 random pairs of nodes to perform intersection tests on.

The reduction in highly connected super-nodes in the visible interaction graph means that random walks are less likely to connect. Figure 3.47 shows that for the Facebook social graph, the probability for all paths to intersect approaches 100% at $w = 1200$. For Facebook visible interaction graphs, the percentage of intersecting paths never reaches 100% since a large fraction of random walks never intersect. SybilGuard, as a
result, is less effective on a graph that models user trust (visible interaction graph) than on a normal social graph.

<table>
<thead>
<tr>
<th>Facebook Graph</th>
<th>Total Loops (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social</td>
<td>951 (3.8)</td>
</tr>
<tr>
<td>Full Interaction</td>
<td>3196 (12.8)</td>
</tr>
<tr>
<td>1 Year I.Graph</td>
<td>4726 (18.9)</td>
</tr>
<tr>
<td>6 Month I.Graph</td>
<td>4953 (19.8)</td>
</tr>
<tr>
<td>2 Month I.Graph</td>
<td>5782 (23.1)</td>
</tr>
</tbody>
</table>

**Table 3.11:** Self-Looping Statistics for SybilGuard.

A major factor affecting the performance of the SybilGuard algorithm is the prevalence of self-loops in the random walks. Any walk that returns to the origin point before going \( w \) steps is useless for the purposes of performing intersection tests. Table 3.11 shows the total number of self-loops encountered during all experimental runs on each graph. The drop in efficacy observed in Figure 3.47 is directly correlated to the increase in self-looping from 3.8% on the social graph to an upwards of 20% on visible interactions graphs.

### 3.7.3 Influence Maximization

The capability to model and predict the spread of information through social networks has many real-world applications. These range from combating the spread of disease, to generating effective word-of-mouth marketing campaigns. An important
problem in this area is influence maximization: locating the most influential users who will maximize the spread of information through the social network.

Previous works have designed algorithms that use statistical methods to model information dissemination over social links [50, 98]. One such model is the weighted cascade model. In this model, each user \( u \) that is “activated” by receiving or producing some new information has a chance to activate their friend \( v \) with probability \( 1/\text{deg}(v) \). This process is repeated for all of \( u \)'s friends. The MixedGreedyWC algorithm implements the weighted cascade model, and calculates for a given social network topology, the most influential users (called “seeds”) and the set of nodes influenced by them [50].

**Expected Impact.** The weighted cascade model assumes that, for a given node, the probability of activating each neighbor is proportional to their degree. However, as we have demonstrated in this work, not all social links are equally important. A node is more likely to be influenced by users it interacts with, as opposed to familiar strangers. Thus, we propose running the weighted cascade model on interaction graphs, as the interaction graph prunes out edges that are unlikely to ever be activated in reality. The reduction in average node degrees will increase the activation probability of the remaining links in the graph. However, the overall reach of each node will be reduced, thus constraining the spread of information as compared to the full social graph.

We evaluate the weighted cascade model on both visible and latent interaction graphs, since both types of interactions are necessary for information to spread on
social networks. Latent interactions capture one-hop information dissemination, \textit{i.e.} reading new information that a friend has posted to their profile. However, in order for information to move two-hops (or farther), visible interactions are required. Users must retweet or “like” the original information in order to copy it to their own profile, so their friends can see it. Thus, information dissemination requires a combination of interactions. We leave as future work a combined graph model that incorporates visible and latent interactions together.

**Results.** Figure 3.48 shows the results of running the MixedGreedyWC algorithm on our Facebook and Renren social and interaction graphs. In Figure 3.48(a), each line is an average over 12 of our Facebook regional graphs, since the largest regions are too big to be processed by the software. Figure 3.48(b) is computed over the Renren PKU network. For all graph types, the total number of influenced users grows as the number of highly influential seed nodes is increased. However, the relative number of
users reached is lower for interaction graphs, as compared to the full social graph. On Facebook, the reduction is drastic: an order of magnitude fewer users are influence on interaction graphs. On Renren, the reductions are more modest: 15% fewer users are influence on the latent graph, and 50% fewer are influenced on the visible graph.

The reduction in information dissemination is due to a combination of factors. First, there are fewer total nodes in the interaction graphs, since nodes that do not interact at all have zero links. This shrinks the pool of potential targets. More importantly, the interaction graphs also have drastically fewer edges than the full social graph. For example, average node degree drops from $\sim 77$ on the Facebook social graph to 1 for the time-constrained visible interaction graphs. The same logic applies to Renren: the Renren latent interaction graph results in greater dissemination than the Renren visible interaction graph because average node degrees are higher in the former. The reduction in node degrees has the effect of limiting the potential reach of seed nodes.

To get a finer grained understanding of the output of the MixedGreedyWC algorithm, we compared the 50 seed nodes chosen by the algorithm on the 12 Facebook social graphs versus the corresponding full (i.e. non-time constrained) visible interaction graphs. Only 11% of the seeds from the social graphs are also selected on the full interaction graphs. This demonstrates that the optimal seed selection changes depending on the type of graph being examined. This result agrees with Figure 3.12, which
Chapter 3. Social Graphs and User Interactions

shows that the highest degree nodes on the social graph are not necessarily the most visibly interactive.

In terms of practical impact, these results indicate that researchers examining information dissemination and influence maximization should take care when performing experiments. Modeling information dissemination using the full social graph is an optimistic upper-bound on how information would spread in reality, since not all social links correspond to active relationships. Assuming uniform information spread along all social links can lead to overestimation of information dissemination, as well as leading to the selection of influential seeds that may not be optimal on more constrained graph topologies.

3.8 Facebook Over Time

Since our initial data collection in 2008, Facebook has continued to grow and mature. The user base has grown exponentially, surpassing the 900 million user milestone. The site itself has gone through significant architectural changes, such as the shift towards a Twitter-like, News-Feed centric interface. All of these changes beget the question: do the social graph and visible interaction characteristics observed in Facebook 2008 continue to hold true?
Chapter 3. Social Graphs and User Interactions

In this section, we address this question by performing a comparative analysis between data gathered from Facebook in 2008 and 2009. We also compare our results to those from a study of the Facebook social graph conducted in 2011 by Facebook employees [175]. Our results show that while the rapid growth of Facebook’s user population has added weight to the long-tail of the social graph, the overall trends of interactions on Facebook remain the same.

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes (%)</th>
<th>Links (%)</th>
<th>Rad.</th>
<th>Diam.</th>
<th>Avg. Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>London</td>
<td>1,690K (36%)</td>
<td>46,169K (50%)</td>
<td>11 / 10</td>
<td>15/15</td>
<td>5.09 / 4.99</td>
</tr>
<tr>
<td>New York</td>
<td>905K (139%)</td>
<td>21,230K (194%)</td>
<td>11 / 11</td>
<td>14/15</td>
<td>4.80 / 4.77</td>
</tr>
<tr>
<td>Sweden</td>
<td>651K (13%)</td>
<td>23,213K (34%)</td>
<td>8 / 8</td>
<td>11 / 12</td>
<td>4.55 / 4.36</td>
</tr>
<tr>
<td>Los Angeles</td>
<td>603K (119%)</td>
<td>15,352K (263%)</td>
<td>12 / 10</td>
<td>16/15</td>
<td>5.14 / 4.54</td>
</tr>
<tr>
<td>Mexico</td>
<td>598K (90%)</td>
<td>9,104K (39%)</td>
<td>9 / 9</td>
<td>13 / 15</td>
<td>4.89 / 5.22</td>
</tr>
<tr>
<td>Egypt</td>
<td>298K (21%)</td>
<td>5,047K (56%)</td>
<td>9 / 8</td>
<td>12 / 13</td>
<td>4.88 / 4.58</td>
</tr>
<tr>
<td>Total</td>
<td>4,745K (57%)</td>
<td>120,115K (73%)</td>
<td>10 / 9</td>
<td>13 / 13</td>
<td>4.8 / 4.74</td>
</tr>
<tr>
<td>Facebook 2011*</td>
<td>721M</td>
<td>68.7B</td>
<td>N/A</td>
<td>≥11</td>
<td>4.7</td>
</tr>
</tbody>
</table>

Table 3.12: Statistics for 2009 Facebook regional networks compared to 2008. (%) are percent increases from 2008 to 2009. Other values are presented as 2008/2009. The final row (*) shows the values for the entire Facebook social graph in 2011 from [175].

3.8.1 Description of Collected Data and Methodology

In order to validate our conclusions drawn from Facebook 2008 data, we crawled additional data from Facebook in 2009. We crawled 6 regional networks between April and June of 2009, just over one year after our original crawls. This resulted in data on 4.7 million users with 120 million friend links (see Table 3.12), at a time when
Facebook’s total population was \( \sim 200 \) million [210]. Our crawl methodology remained the same as for the 2008 crawls: 50 random users were chosen to seed the crawlers BFS of each region. These crawls were conducted before Facebook deprecated the networks feature in summer of 2009.

Between 2008 and 2009, the Facebook site went through significant architectural and usability changes, the most significant of which was the move to a News-Feed centric profile layout. These changes impact the type and amount of per-user information accessible to our crawlers. In 2008, each user’s profile page was composed of different applications such as photos, Wall, and events, each of which inhabited a separate area of the page. The data contained in each application domain was completely separate from the data in other applications. This partitioning made it straightforward to completely crawl application specific data. In 2009, Facebook began moving towards its current architecture, which is centered on the News-Feed. Each user’s News-Feed aggregates all of their status updates, as well as all incoming interactions from friends.

We gathered interaction data from each crawled user by downloading their News-Feed histories going back to January 1st, 2008. This gives us a complete 1.5 year record of incoming interactions and status updates for each user. Limitations stemming from Facebook’s back-end architecture made it impractical to crawl older Feed data. However, because Facebook’s population more than doubled between 2008 and 2009,
Chapter 3. Social Graphs and User Interactions

this 1.5 year history encompasses the full lifetime of the majority of Facebook accounts.

In total, we gathered 244 million interactions between Facebook users.

Each Feed item is characterized by a sender and receiver, a time-stamp, an application descriptor, and an application specific data payload. The application descriptor either refers to one of the built-in Facebook applications, such as Wall, photos, or events, or to a third-party application (referenced by a unique application ID). Although Facebook supported “likes” and comments on Feed items during the time of our crawls, security measures prevented us from reliably gathering these interactions. Thus, our 2009 interaction data should be viewed as a lower bound on the total number of interactions on Facebook at the time.

To maintain compatibility with our 2008 study, our 2009 interaction analysis focuses on Feed items from the “wall” application. Facebook lumps all text comments into this umbrella category, meaning the “wall” application includes users’ traditional Wall posts, as well as comments left on photos, events, videos, notes, etc. Items from the “wall” application account for over 85% of all interactions in our 2009 dataset.

3.8.2 Social Graph Analysis

We begin our comparison between Facebook 2008 and 2009 by focusing on the overall social graph. Table 3.12 shows the number of nodes and edges in each of our 2009 regional networks, as well as the percent increase in size of each compared to
Chapter 3. Social Graphs and User Interactions

Figure 3.49: Comparing social degree in Facebook 2008 to 2009.

Figure 3.50: Clustering coefficient of Facebook users in 2008 and 2009.

2008. Overall, the user base of the regions grew by over 57%. More significantly, the number of edges grew by 73%, outstripping the growth of the user population. This results in higher average social degrees for users in 2009, which in turn causes the radius and average path lengths for 2009 social graphs to decrease slightly. Between 2009 and 2011 Facebook’s user population grew by an additional order of magnitude to 721 million, but the average path length and diameter of the graph stayed relatively constant. This indicates that the Facebook graph may have reached an equilibrium point by 2009.

Figure 3.49 depicts the social degree CDF for Facebook 2008 and 2009. The 2009 graph shifts to the right of 2008, reflecting the increase in average social degree during this time period. The two lines re-converge around the 900 friend mark, indicating that the additional links fueling this growth are not concentrated among super-nodes. On the contrary, Facebook’s hard limit of 5000 friends ensures that additional edges are formed between lower degree users. The power law coefficient for Facebook 2009 is
Chapter 3. Social Graphs and User Interactions

Figure 3.51: $k_{nn}$ of Facebook users in 2008 and 2009.

Figure 3.52: The distribution of users’ interaction among their friends, for users in 2008 and 2009.

1.21 with a fitting error of 0.34157, which is slightly lower than the alpha value of 1.25 observed for 2008.

As a node’s degree increases, its clustering coefficient usually drops commensurately, since the likelihood of forming complete three-person friend cliques is reduced. However, as shown in Figure 3.50, even users of the same degree have lower clustering coefficients in 2009 than users in 2008. Although the overall trend remains the same, i.e., lower degree nodes demonstrate more local clustering, the drop in 2009 reflects changing dynamics in Facebook. As the overall population of Facebook grows, user’s friend bases are diversifying such that the likelihood of sharing mutual acquaintances with your friends is reduced. The average clustering coefficient for users with degree = 100 is 0.14 in Facebook 2011 [175], which is in-between the values for 2007 and 2009.

As Figure 3.51 demonstrates, $k_{nn}$ values in Facebook 2009 are generally higher than in 2008, with the peak values for super-nodes being almost twice as high. In this case,
the higher average node degrees for 2009 shown in Figure 3.49 translate to increased $k_{nn}$ values: higher node degrees on average cause the average degree of all friends for each given node to also increase commensurately. The large increases for high-degree nodes indicate increasing homogeneity for this class of users, \textit{i.e.} super-nodes in 2009 are more likely to be friends with other super-nodes than 2008.

The assortativity for Facebook in 2007, 2009, and 2011 is 0.17, 0.21, and 0.22, respectively [175]. Once again, it appears that Facebook reached an equilibrium point in the structure of the social graph around 2009.

### 3.8.3 Visible Interaction Analysis

At this point we have examined how the social graph of Facebook changed in the one year period between spring of 2008 and 2009. The next sets of comparative tests examine how visible interactions between users have changed during this period.

The first question we revisit is how interactions are distributed among each user’s friends. Figure 3.52 depicts the results when considering 70\% and 100\% of total interactions. The trends for each range are similar for 2008 and 2009, and thus the high level conclusion of this figure remains the same as our original discussion in Section 3.2: users do not interact with the majority of their friends. The 100\% lines are slightly divergent, indicating that users in 2009 interact with fewer of their friends than in 2008. This result is consistent with our observation of increased average node degrees: even
though users continue to accumulate friends, the amount of time they have to dedicate towards social interactions remains limited and fixed.

Figure 3.53 demonstrates that the top interactive users on Facebook contribute less towards total interactions in 2009 than in 2008. This can be attributed to the exponential growth in the Facebook user population during this time, which results in many more users interacting overall. Even though these casual users may seldom interact, taken in aggregate their numbers are large enough to dwarf the output of the most interactive users. This observation also holds true in Figure 3.54, which plots the contributions of the highest degree nodes to total interactions.

In summary, we observe that the high-level conclusions we have drawn about visible interactions on Facebook in 2008 also hold true in 2009. Specifically, we observe that: 1) interactions are confined to a subset of each user’s friends (Figure 3.52), 2) interactions are skewed towards a highly-active subset of the population (Figure 3.53),
and 3) super-nodes are not necessarily the most interactive users on Facebook (Figure 3.54). However, the massive population growth on Facebook between 2008 and 2009 does have an effect on interaction patterns. The overall increase in average node degrees means users end up interacting with even fewer of their “friends.” Similarly, the increase in normal users relative to super-nodes reduces the impact of high-degree and high-interaction nodes on total interactions. These results are to be expected, given the changes in the social graph over the measurement period.

3.9 Summary of Results

In this chapter, we answer the question: *Are social links valid indicators of real user interaction?* To do this, we gathered extensive data from crawls of Facebook (in 2008 and 2009) and Renren (in 2009). We confirm that the graph structures of Facebook and Renren exhibit the expected properties of large OSNs, and make some key observations about the nature of visible interactions between users, including:

- *No social network users interact with more than half of their “friends,”* and most users interact with significantly less.

- Visible interactions are heavily skewed towards a small subset of the OSN population who are extremely active.
Most visible interactions are reciprocated, which agrees with the expected social norms for polite conversations.

Although Facebook’s user base grew exponentially between 2008 and 2009, its fundamental graph properties and user interaction patterns remained relatively constant. This suggests that OSNs quickly reach an equilibrium point in their evolution.

We also uncover important properties about latent interactions between users, including:

- Users’ profile popularity varies significantly across the population, and closely follows a Zipf distribution.

- Profile visits have extremely low reciprocity, despite the fact that Renren users have full access to the list of recent visitors to their profile.

- Compared to visible interactions, latent profile browsing is far more prevalent and more evenly distributed across a user’s friends. Profile visits are less likely to be repeated than visible interactions, but are more likely to generate visible comments than other content such as photos and diary entries.

- Users receive a significant portion of visits from strangers that are more than two hops removed from them on the social graph.
Chapter 3. Social Graphs and User Interactions

- For all users, regardless of their number of friends, profile popularity is not strongly correlated with frequency of new profile content.

To capture the observed differences in edge importance caused by user interactions, we introduce the interaction graph as a more accurate representation of meaningful connectivity on social networks. We calculated that the graph properties of interaction graphs vary significantly from their social graph counterparts. To understand how these differences effect social applications, we evaluate three social applications from the literature on our graphs. The results of these experiments reveal that the behavior of social applications varies dramatically when user interactions are taken into account. These results strongly suggest that social applications should be designed with interactions graphs in mind, so that they reflect real user activity rather than social linkage alone.

To support the efforts of the social network research community, we make available a selection of anonymized social and interaction graphs from our Facebook dataset. Details on the available graphs, as well as instructions for requesting access to the data, are available on our lab website.²

²http://current.cs.ucsb.edu/facebook/
Chapter 4

Social Spam

4.1 Introduction

As communities built out of friends, family, and acquaintances, OSNs strive to project the image of being secure environments for online communication, free from the threats prevalent on the rest of the Internet. In fact, a study we conducted on a social auction site demonstrated that the social network could indeed provide a protective environment with significantly lower levels of fraud [166].

Unfortunately, this image of security on large OSNs is only a thin façade. Popular OSNs are the target of phishing attacks launched from large botnets [10, 11], and OSN account credentials are being sold online in underground forums [177]. Using compromised or fake accounts, attackers can turn the trusted OSN environment against

\footnote{Portions of this chapter originate from our paper “Detecting and Characterizing Social Spam Campaigns” [71].}
its users by disguising phishing messages as communications from friends and family members.

In this chapter, we analyze the primary outward symptom of security problems on OSNs: spam. We use the 1.5 year long wall post histories from our Facebook 2009 dataset (introduced in Section 3.8) as the basis for this study. Wall posts were the primary form of communication on Facebook in 2009, and each post is persistent unless explicitly removed by the owner. As such, wall messages are the intuitive place to look for attempts to spread spam on Facebook since the messages are persistent and semi-public, i.e. likely to be viewed by the target user and potentially the target’s friends. In total, our dataset contains over 187 million wall posts received by 3.5 million users.

Our study of Facebook wall posts contains three key phases. First, we analyze all wall messages and use a number of complementary techniques to identify attempts to spread suspicious content (Section 4.3). We focus our analysis on messages that contain URLs or web addresses in text form. From these messages, we produce correlated subsets of wall posts. We model each post as a node, and create edges connecting any two nodes referring to the same URL, or any two nodes sharing similar text content as defined by an approximate textual fingerprint. This process creates a number of connected subgraphs that partition all wall posts into mutually exclusive subsets, where posts in a set are related.
Chapter 4. Social Spam

Our second phase is to separate groups of benign wall posts from the malicious spam. In Chapter 3, we characterized the overall behavior of Facebook and Renren users. Our results paint a very clear picture of the normal, expected interaction patterns between friends on Facebook. This knowledge provides us with a solid foundation for looking for anomalous interaction patterns that are indicative of spam. Using the dual behavioral hints of bursty activity and distributed communication, we can identify groups of wall posts that deviate significantly from expected communication patterns. We use several complementary mechanisms to validate the effectiveness of our detection methodology, and show that our approach is highly effective at detecting the spread of social spam (Section 4.4).

In our third and final phase, we analyze the characteristics of the malicious wall posts we have identified (Section 4.5). Our results provide several interesting observations on the spread of malicious content in OSNs, and the behavior of users that spread it. We find that phishing is by far the most popular attack on Facebook. We also find that users who spread malicious content communicate using very different patterns compared to the average user, and that malicious users stand out by the bursty nature of their wall posts. By studying the time-duration of malicious messages and the lifetimes of users that send them, we conclude that the majority of spam messages are sent through compromised accounts, rather than fake accounts specifically created.
for spam delivery. Finally, we study the largest observed spam campaigns, and make observations about their attack goals and sales pitches.

4.2 Dataset and Scope

In this section, we introduce the dataset used in this chapter, and clarify the scope of our spam detection work before we begin analysis of the data.

4.2.1 Facebook Dataset

We leverage our Facebook 2009 dataset as the basis for our spam detection work. The collection methodology for this data is given in Section 3.8.1, while the specifics for each Facebook regional network are available in Table 3.12. From the entire Facebook 2009 dataset, we focus on the 3.5 million users that received about 187 million wall posts. Table 4.1 summarizes the dataset characteristics. Unfortunately, there was a one year gap between our data collection and our spam detection work, which creates some additional challenges in our analysis. We leverage a stringent set of heuristic tests (Section 4.4) to overcome this difficulty.

In this chapter we are interested in studying wall posts that could potentially be spam. This means isolating the set of wall posts that are:  

i) generated by users, not third-party applications, and

ii) include embedded URLs to external websites. Note
that we do not limit ourselves to URLs in the form of hypertext links. We also handle wall posts with “hidden” URLs, i.e. URLs in plain text or even obfuscated form.

### 4.2.2 Scope of This Work

A wide range of attacks exists in today’s OSNs. We do not attempt to address all of them in this chapter. Our focus is solely on detecting and measuring large scale spam campaigns transmitted via Facebook users’ wall posts. Although spam traditionally refers to massive, unsolicited campaigns to sell goods over email, we do not restrict ourselves to this behavior alone. Rather, we identify and measure multiple types of attacks that are executed via spam wall posts, including but not restricted to: 1) product advertisements, 2) phishing attacks and 3) drive-by-download attacks. Although the purpose of each attack varies, they share one common feature: attackers create or compromise a large number of Facebook accounts and use them to spam wall posts to an even larger set of users. The wall posts left by attackers each contain a (potentially obfuscated) URL and text to convince the recipient to visit the URL. If a recipient is deceived and visits the URL, she will be led to a malicious website associated with the spam campaign. Throughout this chapter, we refer to a wall post as “malicious” if it belongs to a spam campaign. In addition, we refer to an account as “malicious” if it has made at least one malicious wall post.
Chapter 4. Social Spam

4.3 Spam Campaign Detection

To identify malicious wall posts from our large dataset, we use semantic similarity metrics to identify mutually exclusive groups of wall posts. We then use behavioral cues to classify wall post groups as benign or potentially malicious. In this section, we describe the multi-step process through which we organize, group, and identify potentially malicious posts. We begin with an overview of the entire process, followed by a detailed description of each step.

4.3.1 Overview

Our spam detection methodology is guided by intuition about techniques used in spam campaigns to maximize reach and avoid detection. Spammers generate profit by selling products or services, performing phishing attacks, or installing malware onto
victim’s machines. The common thread that runs through all of these attacks is that the spammer must trick users into visiting a URL that leads to a malicious website. Therefore, we assume each spam campaign is focused on convincing the maximum number of users to visit some particular URL(s).

To make a spam campaign effective, spammers are likely to a) customize individual messages towards the targeted user, and b) attempt to avoid detection by hiding the destination URL through obfuscation. Thus, it is possible for messages within the same campaign to look significantly different. This “diversity” makes detection of spam campaigns challenging, since neither the textual description nor the destination URL, or even their combination, can be used as an effective signature to detect a single campaign.

Our intuition is to use complementary techniques to overcome these hurdles, with the goal of aggregating messages from the same spam campaign together into groups. For our purposes, we use the term “group” to refer to a connected component of nodes. We avoid the term “cluster” since our groups are defined by connectivity alone, not by the optimization of a function like conductance or modularity. Note that we do not aim to completely aggregate spam wall posts from one campaign into one single group, as this level of precision is unnecessary.

First, to overcome user-customization techniques, we refer to prior work that shows spamming botnets use templates to generate customized email spam messages [100].
Chapter 4. Social Spam

Indeed, during manual inspection of wall posts in our dataset we observed a large number of suspicious posts that look similar to each other. From this we hypothesize that spam wall posts are also generated using templates, and posts generated from the same template should contain only small differences. Thus, we propose to group wall posts with “similar” textual description together, where similarity is captured by a probabilistic fingerprint. This probabilistic fingerprint must be more efficient to compute than edit distance, yet accurate enough to reflect similarity between text samples despite attempts by attackers to obfuscate or customize the text.

Second, we reason that all attempts to direct OSN users towards a single destination URL must come from the same spam campaign. Thus, we group together all wall posts that include URLs to the same destination, including those that have been hidden through textual obfuscation (e.g. www dot hack dot com) and chains of HTTP redirects.

In summary, our detection approach focuses on two techniques that group together wall posts that share either the same (possibly obfuscated) destination URL, or strong textual similarity. We model all wall posts as nodes in a large graph, and build edges when two posts are connected by one of the above techniques. Intuitively, the resulting connected subgraphs represent messages within the same spam campaign. The pairwise comparison among wall posts results in $O(n^2)$ time complexity where $n$ is the number of wall posts. Although this time complexity can be significant for large values of $n$, this approach actually performs reasonably fast in practice. Processing our entire dataset
of wall posts dating back to January 1, 2008 took a total of 2 days of computation time on a commodity server. While the worst case space complexity is also $O(n^2)$, the graph is very sparse in practice, i.e. most wall posts are non-malicious, and thus distinct. A modestly configured server has sufficient memory ($\approx 4$GB) to handle such computations.

After constructing a graph of wall posts, we leverage two additional assumptions about spam campaigns to separate suspicious, spam groups from the large majority of benign wall posts. These assumptions are: a) any single account is limited in the number of wall posts it can post, thus spammers must leverage a significant number of user accounts for large campaigns, and b) spam campaigns must maximize time efficiency of compromised or fake accounts before detection, thus messages in a single campaign are relatively bursty in time. We apply threshold filters based on the number of distinct user accounts sending wall posts and time correlation within each connected component to distinguish potentially malicious groups from benign ones. An overview of the methodology workflow is shown in Figure 4.1.

### 4.3.2 Analyzing and Grouping Wall Posts

We model each wall post as a $<\text{description, URL}>$ pair. The description is the text in the wall post surrounding the URL. The URL may be encoded in a legitimate format or obfuscated. The description and URL are the only information we use during the
Chapter 4. Social Spam

detection phase of our work. Additional information, like the sender’s and receiver’s
unique Facebook IDs, are involved in later phases of our work. It is possible for the
description to be an empty string, in which case the wall post contains only an URL.
Clearly, while most benign wall posts do not contain any URLs, not all wall posts with
URLs are malicious.

Since any wall post without a URL cannot achieve a spammer’s goals, we begin
by first excluding all wall posts without embedded URLs from further analysis. Next,
we use our two assumptions (described above) to connect any two wall posts together
if they: point to the same destination URL, or share an approximately similar text
description. Thus, we define two wall posts as similar if they share the same URL
and/or similar description. Accordingly, the wall post similarity graph is an undirected
graph $G = \langle V, E \rangle$, where each node $v \in V$ represents one wall post, and two nodes
$u$ and $v$ are connected with an edge $e_{uv}$ if and only if they are similar.

Building the Wall Post Similarity Graph. Before linking wall posts into a graph,
we first identify wall posts containing URLs and extract their URLs. Locating and
extracting well formed, properly marked up hyperlinks from wall posts is a simple
process. However, it is non-trivial to recover obfuscated URLs that have been hidden
inside plaintext. To detect obfuscated URLs, we first use keyword searches, e.g. “click
here,” to detect the starting location of a potential URL. Next, we scan along the text
and remove any characters that are not legal in HTTP URLs (e.g. whitespace, etc.). We
also reverse common techniques used by spammers to obfuscate URLs, e.g. replacing “dot” with “.”, during the scan. This reconstruction process continues until we either successfully rebuild the URL, or determine that this chunk of text cannot be a legitimate URL. Locating and recovering obfuscated URLs is only done once in the preprocessing stage and is not repeated when building the wall post similarity graph. Following reconstruction, two URLs are considered the same if they match, ignoring HTTP URL parameters.

To find approximate similarity matches between text descriptions, we compute a fingerprint for each block of the description text. We treat the description as a single string, and compute 128-bit MD5 hashes for each 10-byte long substring. We then sort these hash values and choose the 20 numerically smallest values as our approximate fingerprint. Two descriptions are similar if and only if at least 19 of their fingerprints match. This approach matches descriptions even after some substrings have been modified, and has been shown to be successful against adversarial spammers in our prior work [208]. We experimented with our dataset to determine that 19/20 was an appropriate threshold that yielded the best trade-off between robustness against obfuscation while avoiding false positives. Additionally, a relatively high absolute threshold implicitly filters out strings shorter than 19 characters, which is useful since it is common for very short text descriptions to have matching content.
Chapter 4. Social Spam

Now the problem of identifying spam campaigns reduces to a problem of identifying connected subgraphs inside the similarity graph. Each connected subgraph is equivalent to one component of a potential spam campaign. Identifying connected subgraphs is easily solved by iteratively choosing arbitrary nodes and identifying their transitive closures. This can be accomplished using a simple breadth-first search algorithm.

As an optimization, we could first group together wall posts that share the same URL, then do a pair-wise comparison between the description of wall posts in different groups. If two wall posts share similar description, their corresponding groups are merged and the comparison between the remaining wall posts within these two groups are skipped. This optimization eliminates redundant comparisons between wall posts that are already within the same group, although the asymptotic time complexity remains $O(n^2)$.

4.3.3 Identifying Spam Groups

Now that we have organized all wall posts into connected components that could represent coordinated spam campaigns, the next step is to identify which groups are likely to represent actual spam campaigns.

To detect spam groups, we use two widely acknowledged distinguishing features of spam campaigns: their “distributed” coverage and “bursty” nature. The “distributed” property is quantified using the number of distinct users that send wall posts in the
group. In email spam campaigns, the “distributed” property is usually quantified using the number of IP addresses or ASes of the senders [195]. The analogous identifier in OSNs is each user’s unique ID. The “bursty” property is based on the intuition that most spam campaigns involve coordinated action by many accounts within short periods of time [195]. We characterize each group of wall posts by measuring the absolute time interval between consecutive wall posts (using timestamps associated with each wall post), and extracting the median value of all such intervals. The median interval characterizes how fast the wall posts were generated, and is robust to outliers.

While we use both assumptions of spam’s distributed and bursty nature to detect spam campaigns, it is possible that social cascades in the social network might produce similar wall post groups. For example, a URL to buy tickets to a highly anticipated concert might be widely propagated throughout Facebook by friends. However, studies of social cascades have shown that cascades take significant time to propagate [46], suggesting that false positives from social cascades would be filtered out by our temporal burstiness filter.

Now that we are using the “distributed” and “bursty” properties to identify malicious groups, we face the problem of identifying the best cutoff threshold values. We can maximize the number of malicious groups we identify by relaxing the thresholds, but that generally results in the methodology producing more false positives on benign
Chapter 4. Social Spam

groups. In contrast, using more restrictive thresholds can reduce the number of false positives, but may fail to identify malicious groups (false negatives).

To solve this dilemma, we first ask ourselves, how many false positives are we willing to tolerate in order to gain one more true positive? Since our methodology is designed for postmortem analysis, and not detection in real time, we choose this value to be 2. Our goal is to locate as many spam campaigns as possible, and we do not need to guarantee zero false positives, since they can be filtered out using a follow-up validation phase. Instead, we can tolerate a moderate number of false positives at the benefit of reducing false negatives, i.e., the malicious groups that are missed by the detection process. For example, we can set our thresholds to (4, 6hr), where 4 is the lower bound of the “distributed” property, and 6 hours is the upper bound of the “bursty” property. If the spammer is using fewer than 4 accounts to send spam with an interval greater than 6 hours, he is likely to generate negligible impact on benign OSN users.

In practice, we test different possible threshold values to determine values that maximize our utility as we defined earlier. We present our detailed experimental results in Section 4.4.3. We determined that the best threshold values are (5, 1.5hr), but also that a number of possible threshold combinations can work well.
Table 4.1: Overall statistics of our Facebook dataset.

### 4.3.4 Detection Results

Before we describe our efforts to validate our methodology, we first briefly summarize the outcome of our grouping and classification approach. Table 4.1 summarizes statistics of our initial wall posts dataset, and lists the total number of wall posts, distinct users that send posts, and distinct users who received wall posts. The first row corresponds to the entire crawled dataset. The second row is restricted to the subset of wall posts that include embedded URLs, which form the basis of our study.

Applying our grouping approach to the corpus of 2.08 million wall posts produces 1,402,028 groups. As expected, there is a heavy tail in the size of groups, i.e. a small number of very large groups and a large number of very small groups. When we apply our chosen detection threshold, which uses 5 as the minimum number of users involved in each groups and 5400 seconds (1.5 hours) as the maximum median interval between the timestamp of two consecutive wall posts, we produce 297 groups that are classified as potentially malicious spam campaigns. There are a total of 212,863 wall posts contained in these 297 groups.
4.4 Experimental Validation

We have described our methodology for identifying suspicious wall posts and users from the general Facebook population. In this section, we delve more deeply into the results of our detection techniques, in an effort to verify that the posts we identified were indeed malicious. We accomplish this using a combination of techniques and tools from both the research community and the commercial sector. These techniques provide independent, third-party verification of our results.

We apply a stringent set of heuristic tests to each URL that has been embedded in one or more wall posts in a potentially malicious group. Whether the URL is malicious determines whether the wall posts containing it are malicious. However, none of these techniques are foolproof, since there are no guaranteed foolproof methods to identify malicious content online. In the absence of such tools, we take a best-effort approach to validating our results.

4.4.1 Validation Methodology

Our validation methodology includes a series of steps, each of which encapsulates a different heuristic or tool and aims to concretely verify some portion of the suspicious wall posts as definitively malicious. Our goal is to investigate the false positive rate of our proposed methodology by reexamining the 212,863 malicious wall posts identified
Chapter 4. Social Spam

<table>
<thead>
<tr>
<th># of URLs</th>
<th>Features</th>
</tr>
</thead>
</table>
| 1,895     | URL signature: www.facebook.com/profile.id.*
|           | Wall post content: Invitation to visit a fake Facebook profile |
| 407       | URL signature: A domain followed by single folder with obfuscated video as name, *e.g.* www.nemr.net/publicshow/ |
|           | Wall post content: Either “wow video” or “cool video” |
| 511       | URL signature: */imageshack.us/img[0-9]{2-3}/[0-9]{3-4}/mcr[a-z][2-2][0-9].swf |
|           | Wall post content: Invitation to find out about a secret admirer or read a disparaging remark written by a personal enemy |
| 296       | URL signature: *ring*.blogspot.com |
|           | Wall post content: Facebook is giving out ring tones for the user’s cell phone via the provided URL |
| 317       | URL signature: *mcy|sz|*[0-9]{3-11}.com or (multi-ring)tn|sz|*[0-9]{2-6}.com |
|           | Wall post content: Invitation to win a free Playstation 3 |

Table 4.2: Examples of URL groups and their shared features.

in Section 4.3.4. Later in Section 4.4.4, we estimate the false negative rate amongst our entire dataset of 2.08 million wall posts.

Step 1: URL De-obfuscation. A common technique among spammers is to obfuscate malicious URLs by adding white spaces and Unicode characters into them. This allows the offending message to bypass filters that look for blacklisted URLs by simple string matching. We observe that a significant number of wall posts on Facebook included obfuscated links of this nature. Since there is no incentive for benign users to obfuscate links, we mark any wall posts that include such URLs as malicious. We de-obfuscate URLs by reversing the obfuscation process, including removing whitespace padding and canonicalizing URL encoded characters (*e.g.*, “%65%76%69%6C%2E%63%6F%6D” becomes “evil.com”).

151
Chapter 4. Social Spam

Step 2: Redirection Analysis. Another common technique used by spammers to evade detection is to hide malicious sites behind chains of redirects [188]. This serves to obfuscate the true destination of a URL (which may be blacklisted) from users and automated filters. Before proceeding with our validation for a suspicious URL, we use a scripted web browser to detect and follow the chain of redirects, if they exist. More specifically, we use the JSSH extension for Mozilla, which allows us to establish a JavaScript shell connection to a running Mozilla process via TCP/IP [93]. This allowed us to control running Mozilla processes while they traverse through chains of HTTP redirects, eventually retrieving the final destination URL.

Step 3: Third-party tools. We leverage multiple third-party tools from the research community and the private sector to assess the malice of URLs in our dataset. We leverage a number of the most popular URL blacklisting services to determine if URLs are malicious, including: McAfee SiteAdvisor [3], Google’s Safe Browsing API [1], SURBL [6], URIBL [7], Spamhaus [4], and SquidGuard [5]. We submit each unique URL from our dataset to each service in order to account for discrepancies between the coverage of different blacklists. If any blacklist returns a positive for a given URL, it is classified as malicious.

In addition to URL blacklists, we leverage the Wepawet [8] tool from UC Santa Barbara. Wepawet is a specialized tool that uses machine learning to identify web pages with characteristics associated with drive-by download attacks [145]. URLs receiving
Chapter 4. Social Spam

a “malicious” rating from Wepawet are immediately classified as malicious for our purposes as well.

The primary challenge when using automated tools for validation is that our Facebook data was collected during the first half of 2009, roughly one year before we started hunting for spam. Given the ephemeral nature of malicious websites, this means that many of the URLs in our dataset point to stale destinations that no longer exist. Blacklists periodically purge old records from their databases, which further complicates matters. Whenever possible, we query the Internet archive service [2] for the content of URLs based on the timestamp of the associated wall post.

Step 4: Wall Post Keyword Search. Certain keywords often appear in spam messages sent over email. The spammed wall posts in our dataset are no exception: many of them attempt to sell the usual assortment of shady merchandise. To capture these wall messages, we built a set of well-known keywords that are indicative of spam, such as “viagra,” “enlargement pill,” and “legal bud.” We then performed full-text searches on the suspicious wall posts from the detection result for these strings, and classify the resulting posts as malicious.

Step 5: URL Grouping. At this point in our validation, we have verified that some portions of the wall posts in our dataset are malicious. However, shortcomings will likely have prevented us from performing full validation. For example, some URLs are too old and stale to be verified by blacklists. Similarly, we may have missed variants of
spam keywords (e.g. “v14gr4” versus “viagra”) in our full-text search. To expand the coverage of the positive results from the previous validation steps, we use a grouping strategy to detect relationships between posts.

We manually construct groups of URLs that exhibit highly uniform features, which is a strong indicator that the whole group is under the control of a single attacker and is typical of many well-organized spam campaigns. Benign URLs are more random in nature, and highly unlikely to exhibit regular patterns or templates. The features we leverage include signatures that characterize the URL group [195] and similarity of textual content of the associated wall posts. For each group, if any of its constituent URLs has been identified as malicious in a previous validation step, then all URLs in the group are classified as malicious. We identify 8 such groups in total and list examples with their features in Table 4.2.

**Step 6: Manual Analysis.** Even after all five previous validation steps, a small portion of suspicious wall posts still remain unclassified. Since widespread spam campaigns are likely to be reported and discussed by people on the web, we can manually validate these URLs by searching for them on Google. Because this task is highly time-intensive, we only use this approach on URLs that each appear in at least four hundred wall posts.
4.4.2 Validation Results

We use the multi-step validation process outlined above to confirm the malice of each wall post identified in our detection process. We assume that all wall posts whose malice cannot be confirmed are false positives. Table 4.3 summarizes the number of URLs and wall posts confirmed as malicious following each validation step, as well as the number of false positives. We see that our heuristic verification mechanisms are surprisingly successful at confirming the large majority of our detected spam messages. Only a very small portion of URLs (roughly 3.9%) remain unconfirmed.

The bulk of true positive URLs are either blacklisted, redirect to a blacklisted site, or grouped together with other blacklisted URLs/wall posts. In contrast, when viewed in terms of total wall posts, Table 4.3 shows that obfuscated and manually verified URLs account for a significant proportion of wall posts. This demonstrates that some URLs
Figure 4.2: Number of true positives and false positives for each tested combination of threshold values. The dotted line represents our assumed best trade-off between the two (1:2). The highlighted point (cross) represents the threshold we chose for our use.

are spammed a disproportionally large number of times, while other spam campaigns spread their traffic across a long tail of unique URLs.

4.4.3 Burst and Distribution Thresholds

Our detection mechanism relies heavily on the bursty and distributed nature of spam messages. In order to choose a good cutoff threshold between normal and aberrant behavior, we define a desired trade-off between false positives and true positives: we are willing to tolerate two false positives if we can gain at least one true positive. We then choose the threshold value with the best utility (Section 4.3.3). In order to find the optimal threshold, we vary our filtering threshold for the “distributed” property of wall posts from 4 to 10, and for the “bursty” property from 0.5 to 6 hours in increments of 0.5 hour. Every combination of these two properties is tested to examine the resulting false positives and true positives, with the results plotted in Figure 4.2. The resulting
points roughly form a convex curve. We choose a straight line with slope of 0.5, which represents the desired trade-off, to approach the curve from above. The straight line and the curve intersect at the point representing the threshold (5, 1.5hr). Thus we choose (5, 1.5hr) as the threshold for our detector. Note that, as Figure 4.2 shows, a number of alternative threshold value combinations would also have yielded good results.

4.4.4 False Negative Rate Estimation

In addition to estimating the rate of false positives resulting from our detection methodology, it is also desirable to characterize the amount of false negatives it can produce, i.e. the number of malicious wall posts that go undetected. Unfortunately, computing the false negative rate is not feasible in practice for a number of reasons. Given the sheer size of our dataset (2.08 million wall posts with URLs), none of our validation techniques, third-party or manual, will scale. While we cannot compute a real false negative rate, we can offer some estimates of the effectiveness of some of our mechanisms as rough indicators of how many potential malicious posts could be missed.

Obfuscated URLs. As mentioned in Section 4.4.1, we believe that all obfuscated URLs are indicative of malicious activity. Hence, any obfuscated URL that is not included in the detection results should be viewed as a false negative. We searched our entire dataset for obfuscated URLs and identified 1,012 total URLs, of which 1,003
Chapter 4. Social Spam

were found in our detection results. These 9 missed URLs are only used in 141 wall posts. In contrast, the 1,003 correctly identified obfuscated URLs appear in 45,655 wall posts. Thus, using the pool of obfuscated URLs as ground truth, our detection mechanisms are more than 99% effective.

**Blogger Pages.** We notice that Blogger pages are frequently used by attackers in our dataset to host malicious content. We took all the blogspot.com pages from our full dataset, and applied our full validation process to them. This generated a total of 8,162 Blogger pages identified as malicious. Using this set as ground truth, only 406 of these URLs were not detected by our detection mechanism, for a false negative rate of 5%.

### 4.5 Analysis of Spam Activity

After going through the detection (Section 4.3) and the validation (Section 4.4) steps, we arrive at a set of true positive malicious wall posts. In this section, we analyze the detected malicious users and wall posts to understand their characteristics and their impact on the OSN. Wherever possible, we compare and contrast the behavior of malicious accounts using corresponding characteristics of benign users.

Because of the way default privacy settings are configured on Facebook, we have more complete information on users within the crawled regional networks than outside users. Thus we prepare two datasets. The first contains the full set of detected malicious
posts and users. We refer to it as the *full set*. The second excludes all malicious users outside the crawled regional networks, and all wall posts they generated. We refer to it as the *local set*. The full set contains 199,782 posts from 56,922 users, while the local set includes 37,924 posts from 6,285 users. In the following analyses, we present results from the more appropriate dataset, depending on whether complete OSN meta-data is necessary for the metric.

### 4.5.1 URL Characteristics

We categorize malicious URLs by URL format and by domain name, and measure the prevalence of each category. We use the *full set* for this analysis.

**Categorized by Format.** Throughout our analysis, we identified three different formats used to embed URLs in wall posts: *hyperlinks*, *plain text* and *obfuscated text*. A hyperlink, *e.g.* `<a href="http://2url.org/?67592">http://2url.org/?67592</a>`, is a standard link embedded in a wall post. It is the easiest format for victims to visit, but is also easily recognized by automated detection techniques. A plain text URL, *e.g.* `mynewcrush.com`, is not as easy to use. The victim must copy and paste the URL to the browser’s address bar to visit the site. An obfuscated URL, *e.g.* `nevasubevu\t. blogs pot\t.\tco\t\tm (take out spaces)`, is the most sophisticated. It describes the actual URL to the victim in a human decipherable way. The victim must comprehend and reconstruct the URL by
Chapter 4. Social Spam

Table 4.4: Number of malicious URLs in each format type.

<table>
<thead>
<tr>
<th>Type</th>
<th># of URLs</th>
<th># of Wall Posts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obfuscated</td>
<td>1,003</td>
<td>50,459</td>
</tr>
<tr>
<td>Plain text</td>
<td>583</td>
<td>13,361</td>
</tr>
<tr>
<td>Hypertext</td>
<td>13,898</td>
<td>135,962</td>
</tr>
</tbody>
</table>

Table 4.4 shows the number of distinct URLs and the number of malicious wall posts that contain URLs of each format. Normal hyperlinks are the dominant format when counting either the distinct URLs or total wall posts. They make up 89.8% of all distinct destinations and 68.1% of all malicious posts. However, the other two formats still account for a non-negligible fraction of the overall results. 25.2% of malicious posts contain plain text URLs, while 6.7% contain obfuscated text URLs.

Interestingly, non-hyperlink format URLs are repeated in more total wall posts. On average, each hyperlink URL appears in 9.8 malicious posts. In contrast, an average plain text URL is used in 22.9 malicious posts, while each obfuscated URL is observed in 50.3 wall posts on average.

These results convey two takeaways. First, attackers are willing to embed URLs in a way that is relatively difficult for the target user to visit, presumably for the sole purpose of evading detection. Second, obfuscated URLs are much more likely to be used repeatedly in many wall posts. This observation may stem from differing attacker...
Table 4.5: Number of malicious domain names in each group.

<table>
<thead>
<tr>
<th>Type</th>
<th># of Domains</th>
<th># of Wall Posts</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Shortening</td>
<td>110</td>
<td>10,041</td>
</tr>
<tr>
<td>Blogs</td>
<td>8,609</td>
<td>31,488</td>
</tr>
<tr>
<td>Content Sharing</td>
<td>440</td>
<td>9,506</td>
</tr>
<tr>
<td>Other</td>
<td>6,325</td>
<td>148,747</td>
</tr>
</tbody>
</table>

strategies. If the spammer has many unique URLs at their disposal, then they can safely spam the URLs as hyperlinks, knowing that only a fraction will be blacklisted. Conversely, if the spammer only controls a few URLs, blacklisting could cripple their operation. Thus, URL obfuscation may be a necessary precaution to hinder automated link detection and blacklisting.

Categorized by Domain. We extract the domain names of malicious URLs and categorize them into four general types: content sharing services, URL shortening services, blogs, and others. The first three types demonstrate attackers misusing legitimate services. For content sharing domains, the attacker puts the malicious content into a file and uploads it to the content sharing service, e.g. imageshack. For URL shortening domains, the attacker uses the URL shortening service, e.g. tinyurl, to hide the real (malicious) destination website. For blog domains, the attacker registers an account on the blog service, e.g. blogspot, and uses it to host malicious content. Finally, the other category contains domain names that do not appear to have any systematic or contextual similarities.
Chapter 4. Social Spam

Table 4.5 shows the number of distinct domains in each category, as well as the number of malicious wall posts that contain domains in each category. The blog category dominates in terms of distinct domains. We conjecture that the ease of registering new accounts is the main reason why blogs are popular. However, since all major blog services are administered centrally, it is also easy to identify and remove malicious users and their pages. This potentially explains the high turnover in blog domains: on average, each individual blog is used in only 3.7 malicious wall posts, while domains in other categories are observed in 20 malicious wall posts on average.

4.5.2 Spam Campaign Analysis

A spam “campaign” refers to a set of malicious wall posts that all strive towards a common goal, e.g. selling online pharmaceuticals. We use the description of each wall post to distinguish between campaigns, without considering the destination URL.

Campaign Identification. To isolate individual spam campaigns out of the full set of malicious posts, we iteratively classify wall posts by identifying characteristic strings associated with each campaign. Human input is used to aid this process. For example, any wall post containing words like “viagra” or “enlarge pill” are classified into the “Pharmaceuticals” campaign. If the wall post contains words like “crush on you” or “someone admires you,” we classify it into the “Secret Admirer” campaign. Overall, we identified 19 unique campaigns. They represent 19 common ways that spammers
Table 4.6: A summary of spam campaigns encountered in our study.

At a high level, three major types of campaigns appear most popular amongst OSN spammers. First, spammers may promise free gifts, \textit{e.g.}, free iPhones, free ringtones, \textit{etc.} Second, spammers may trigger the target’s curiosity by saying that someone likes them, is disparaging them, or has written about them on another website. Finally, spam-
mbers may simply describe a product for sale (usually drugs). These three types of campaigns account for roughly 88.2% of all malicious posts.

We associate each campaign with the groups produced by the detection mechanism. The “Fake Video” campaign appears in an exceptionally large number of groups. The reason is that the description in these wall posts is very short, thus the detection mechanism does not merge the groups.

For most campaigns, the corresponding groups form mutually exclusive connected components. There are only two instances where one group is shared by multiple campaigns. The “Secret Admirer” campaign shares one group with the “Love Calculator” campaign and the “Free PS3” campaign. This overlap occurs because both the “Secret Admirer” and “Love Calculator” campaigns use a single common URL in the same period of time. The “Secret Admirer” campaign also shares 400 URLs with the “Free PS3” campaign. The wall posts containing these shared URLs are naturally grouped together. This suggests that there is a single attacker controlling all three of these campaigns.

**Phishing and Malware.** We now turn our focus to spam campaigns that attempt to lure victims to phishing and drive-by download sites. To determine which malicious URLs in our dataset link to sites of these types, we rely on McAfee SiteAdvisor’s [3] user review feature, which allows users to collaboratively mark websites as malicious and categorize their misbehavior. We discard reported sites for which the number of
benign reports exceeds the number of malicious reports. If multiple malicious behaviors are reported, \textit{i.e.} phishing and malware propagation, we count all of them.

Our results demonstrate that phishing is an extremely common practice among OSN spammers. Approximately 70.3\% of malicious wall posts direct the victim to a phishing site. We encountered two different types of phishing attacks during our investigation. In the first case, spammers target the OSN credentials (username and password) of the victim. In one example, the spammer posts wall messages saying Facebook is giving out free ringtones to its users. When the victim visits the provided URL, he is led to a page identical to the Facebook login page and prompted to “log in” again.

In the second case, the spammer is after monetary gain. For example, the spammer leaves wall posts asking victims to take a love compatibility test. Clicking the malicious link directs the victim to a site that asks them to provide their cellphone number and agree to a “terms of service” before they can see the results of the love compatibility test. If the victim proceeds, she is automatically signed up for a service and charged a monthly fee.

Malware propagation is the second most common attack associated with malicious URLs in our dataset. About 35.1\% of malicious wall posts direct victims to sites laced with malware.

The high percentage of phishing and malware attacks likely stems from the social context of OSNs. It has been shown in prior work \cite{90} that adding a bit of personal
information to spam greatly increases the effectiveness of phishing attacks. Intuitively, OSN spam messages appear to come directly from trustworthy friends, and are thus more likely to successfully trick their victims. The same argument holds for malware propagating messages, which effectively turn legitimate users into “bots” controlled by the attacker. These bots are then used to send out more wall posts that propagate phishing and malware.

**Temporal Behavior.** We study the temporal characteristics of the identified spam campaigns, and plot the result in Figure 4.3. The x-axis represents the time period between January 1, 2008 and September 1, 2009, which corresponds to the time frame of wall posts from our dataset. Each spam campaign is represented by a different horizontal strip. Each single point within the strip corresponds to one malicious wall post.
within the campaign. A dark band in the strip reflects a burst of messages in the campaign.

Figure 4.3 shows the bursty nature of all the campaigns. The majority of malicious posts within each campaign are densely packed into a small number of short time bursts, while the entire campaign may span a much longer time period. As stated earlier, URL overlap seems to indicate that the “Secret Admirer”, “Love Calculator” and “Free PS3” campaigns are correlated. The active time periods for these three campaigns overlap, thus corroborating our inference that one attacker may have initiated all three campaigns.

4.5.3 Malicious Account Analysis

We now examine the characteristics of accounts that generate malicious wall posts. We use our data to study the possible origins of these accounts, their impact on their social circles, temporal characteristics of their wall post activity, and whether malicious activity dominates these accounts.

Are Malicious Accounts Compromised? Spammers can obtain their malicious accounts in one of two ways: compromising existing accounts, or creating fake, Sybil accounts [60]. In the first case, the spammer takes control of a legitimate account following a successful phishing or password-cracking attack. This method is attractive, because legitimate accounts already have a significant number of friends (a.k.a. poten-
Chapter 4. Social Spam

tial targets). Plus, since these accounts are trusted or at least known to their friends, spam messages sent from these accounts are be more likely to succeed. However, the downside of this approach is that it is a non-trivial task for the spammer to compromise such accounts. Additionally, they may lose control over these accounts at any time, i.e. the original owners may detect the account compromise and change the password. Alternatively, spammers may create brand new Sybil accounts. These Sybil accounts are “fake” in the sense that they do not represent a real person. Despite the use of mechanisms like CAPTCHAs, account registration is still relatively easy to automate, and attackers can potentially create a large number of accounts [23]. Two studies have even shown that the creation of OSN Sybils has become a lucrative, industrial scale operation powered by low wage, crowdsourced labor [131, 184]. However, Sybils still have to establish friendship with potential victims before they can post spam messages to the victims’ walls.

To distinguish between the two types of malicious accounts, we analyze the overall behavior of malicious accounts to determine if they have ever exhibited characteristics of a benign user. In particular, we study each account’s application usage and number of received, benign wall posts. Application usage includes uploading photos and video, as well as using third-party social applications. All of these activities are indicative of “normal” account behavior, since it is highly unlikely that attackers would expend time and effort to perform these activities on Sybil accounts. We find that 33.9% of
malicious accounts exhibit application usage, while 84.5% of accounts received benign wall posts from their friends. Only 11% of malicious accounts do not use applications or receive benign wall posts.

Simply receiving benign wall posts may not be enough to prove that a malicious account is a compromised legitimate account. First, the received posts may be complaints from other accounts. Second, a sophisticated attacker may create Sybil accounts to post messages on each other’s walls, in order to make them look like normal user accounts. Unfortunately, it is very difficult to use automated tools to identify these cases. Instead, we chose uniform random sample of 200 malicious accounts (approximately 5% of the local malicious accounts) and manually inspect the wall posts they received. The key properties that we look for are conversation topic diversity and the context of offline events. For example, if the friends are talking about the party last week, the college they go to, etc., it strongly suggests that this conversation involved two real human users. For 5 accounts in the sampled set, users conversed in a foreign language we were unable to reliably translate. For the remaining 195 accounts, we only suspect one account to be a fake Sybil account created by attackers. This user only received complaints and wall posts asking who he is. In addition, the corresponding user account has been suspended. The other 194 (97%) accounts exhibited normal communication patterns with their friends, despite their posting of spam-like content. This sampled result, while not representative, suggests that compromised accounts were the prevail-
Chapter 4. Social Spam

Malicious Versus Benign Posts. We now study the ratio of malicious wall posts to the total number of wall posts made by the detected malicious accounts. This ratio indicates how dominant the malicious posting activity is. A compromised account is expected to exhibit mixed behaviors, \textit{i.e.} they post benign wall messages before the account compromise and (potentially) after the compromise is detected and the account re-secured.

Figure 4.4 plots the CDF of this ratio for all malicious accounts. Fewer than 20\% of the malicious accounts only post malicious wall messages. The remaining 80\% of accounts post a mixture of malicious and benign wall posts. Among these accounts, the ratio of malicious wall posts distributes quite evenly. If we make the assumption that
Figure 4.6: The wall post broadness ratio for malicious users, as a function of their social degree.

Figure 4.7: The post broadness ratio of both malicious and benign accounts.

fake Sybil accounts would not post non-malicious messages to friends’ walls, then this result means that at most 20% of all malicious accounts are Sybils.

Figure 4.5 considers the malicious post ratio relative to the total number of wall posts. Most malicious users have fewer than 100 total wall posts. Within this range, the malicious post ratio is distributed relatively evenly. For users with larger number of wall posts, the malicious post ratio decreases. This suggests that attackers are intentionally throttling the number of malicious wall posts they generate, possibly to avoid detection.

The Impact of Malicious Accounts. We now measure the extent of influence of malicious accounts over their friends, i.e. how many of their friends receive spam messages? We quantify this using the “broadness ratio,” defined as the fraction of friends that receive wall posts from a user. Since this measurement requires us to know a user’s social degree (total number friends), we use the local dataset and extract the social degree from the measured social graph.
Figure 4.6 plots the broadness ratio of malicious accounts as a function of the user’s social degree. Surprisingly, malicious accounts are not posting malicious messages on all of their friends’ walls. Instead, broadness ratio values are most concentrated around the 20% range. As social degree increases, the trend is that the broadness ratio decreases. The result is consistent with Figure 4.5. Note that there are striped, round patterns towards the lower end of the x-axis in both Figures 4.6 and 4.5. This is due to the fact that we are dividing two integers to compute the percentage value. For example, the bottom left curve is formed when the numerator is one while the denominator increases. This is a numerical artifact, and represents no underlying trends in the dataset.

Figure 4.7 shows the CDF of the broadness ratio of both malicious and benign accounts. For malicious accounts, we plot two broadness ratio curves. One curve represents the broadness or coverage of malicious wall posts sent by the account, and the other represents the broadness of all wall posts (malicious and benign) sent by the account. The broadness of benign accounts reflects how broadly each user interacts with its friends on Facebook, and the results match those presented in Section 3.8.3. About 60% of benign accounts have broadness less than 0.1, meaning they interact with fewer than 10% of their friends. In addition, more than 30% of users have not posted any wall messages.
Overall, malicious accounts tend to interact with a broader portion of their friends than benign users. However, if we only consider malicious wall posts, the results are similar: spammers tend to spam the same distribution of friends as normal users post to their friends. If we consider both malicious and benign posts for malicious accounts, however, the broadness value becomes considerably larger, indicating that the friends that the accounts normally interact with are disparate from the friends receiving the malicious wall posts. Note that the our data only reveals the lower bound of the broadness of malicious accounts, since recipients of spam wall posts will often delete them from their wall, making it impossible for us to find those posts through measurement.

4.5.4 Temporal Properties of Malicious Activity

Finally, we analyze the temporal activity patterns of malicious accounts. Specifically, we examine the length of time that accounts are actively under the control of malicious attackers, and daily activity patterns of malicious accounts.

Active Time of Malicious Accounts. We define a malicious account’s “active time” to be the time span between the posting time of its first and last malicious wall posts. We ignore benign wall posts made by the malicious account when computing its active time. Since most malicious accounts are actually compromised accounts, the active time is also a conservative estimate of the length of time it took for the account owner
Chapter 4. Social Spam

**Figure 4.8:** The active time of malicious and benign accounts.

**Figure 4.9:** The hourly distribution of malicious and benign wall posts.

to detect the account compromise and fix the problem. Similarly, we define a benign account’s “active time” to be the time span between its first and last wall posts.

Figure 4.8 plots the CDF of the active time of both malicious accounts and benign accounts. Note that we have again plotted two curves for malicious accounts. One shows active time as defined above, the other captures the time between the first and last wall posts (benign and malicious) of a malicious account. In all cases, only the accounts that have made at least 2 wall posts are included. To show greater detail on the curve, we plot the x-axis using log-scale, and mark ticks for 1 minute, 1 hour, 1 day, 1 week, 1 month, and 1 year.

Most malicious accounts are under the attacker’s control for only a short period of time. Roughly 30% of all malicious accounts are active for only a single moment in time, when they posts a single burst of spam to their friends, and never behave maliciously again. Roughly 80% of the malicious accounts are active for less than 1 hour, and only about 10% of them are active for longer than 1 day. The reason for this
Chapter 4. Social Spam

phenomenon may be that when an account starts to post malicious wall messages, its friends will immediately complain, and the account owner will quickly realize that their account has been compromised. We observe a considerable number of such complaints in our wall post dataset. Once account owners recognize the attack, they can quickly reclaim control by contacting Facebook and changing the password. Nevertheless, a small portion of malicious accounts continues to post malicious wall messages for a longer period of time (1 month or more), perhaps exploiting accounts of users who rarely log in to Facebook.

In contrast, the benign accounts exhibit a drastically different pattern. More than 80% of benign accounts are active for more than 1 month, and about 35% of them remain active for more than 1 year. However, if we count all wall posts made by the malicious accounts, i.e. including benign messages, the curve for malicious accounts shifts and becomes similar to the curve for benign users. This is further support for the belief that most malicious accounts are actually compromised legitimate accounts, rather than those created by the attackers.

Malicious Activity per Day. Finally, we study the daily activity patterns of malicious and benign users. We start by extracting the Unix timestamps attached to malicious and benign wall posts. Since timestamps correspond to the local time where our crawling machines reside, (see Section 3.8.1), we adjust them to the local time zone based on the location of the regional network. For example, to compute local times-
Chapter 4. Social Spam

tamps of events in the New York City regional network, we add an offset of 3 hours to timestamps to account for the 3-hour time difference between New York City and U. C. Santa Barbara (California). Next, we group the wall posts by the hour of day, e.g., all wall posts that are made between 8am and 9am are grouped together, and plot the results in Figure 4.9. For readability, we normalize the y-value based on the total number of wall posts.

The curve for the benign wall posts clearly reflects a typical diurnal pattern. Few posts are made late at night, between 1am and 7am. Once people wake up and become active, more wall posts are made. Wall activity increases steadily over the day, peaking at 9PM and dropping quickly afterwards. However, the hourly distribution of malicious wall posts shows a much different pattern. Malicious wall posts peak at 3am, when most users are asleep and away from their accounts. This is one possible reason for attackers to pick this time to post messages, so that they avoid immediate detection by account owners. This suggests that mechanisms for detecting spam activity can prove the most useful in the early morning of each time zone.

4.6 Summary of Results

In this chapter, we present the most comprehensive look as spam on Facebook ever published. We develop automated techniques to isolate spam from our dataset of 187
million wall posts received by 3.5 million Facebook users. A rigorous validation process demonstrates that our spam detection methodology captures 94% of spam wall posts, with low false positive and negative rates. Analysis of the captured wall posts leads to several insights about the nature of social spam, including:

- The spam ecosystem on Facebook mirrors its email counterpart: Facebook spam is organized into large campaigns that leverage tens of thousands of accounts, and disseminate messages very rapidly.

- 10% of wall posts with URLs in 2009 were spam.

- 70% of spam on Facebook directs users to phishing sites, making it the most popular form of attack.

- Attackers leverage compromised accounts, as well as fake, Sybil accounts to spread spam. Differentiating between the two types of accounts is often quite challenging.

- Existing anti-spam techniques are ineffective against social spam. URL blacklists in particular only catch 28% of Facebook spam.
Chapter 5

Social Sybils

5.1 Introduction

As shown in Chapter 4, online social networks are under attack from fake, Sybil accounts. Other studies have observed Sybils sending spam on Twitter [78, 170], as well as infiltrating social games [136]. OSNs are ill-equipped to defend against Sybil attacks, since determining a tight mapping between real people and online identities is an open problem. Looking forward, Sybil attacks on OSNs are poised to become increasingly widespread and dangerous as more people come to rely on OSNs for basic online communication [109, 132] and as replacements for news outlets [105].

To address the problem of Sybils on OSNs, researchers have developed algorithms such as SybilGuard [200], SybilLimit [199], SybilInfer [57], and SumUp [174] to perform decentralized detection of Sybils on social graphs. These systems detect Sybils by

---

1Portions of this chapter originate from our paper “Uncovering Social Network Sybils in the Wild” [197].
identifying tightly connected communities of Sybil nodes [181]. We give more details of these algorithms in Section 3.7.2.

Recent work showed that one of the key assumptions of community-based Sybil detectors, fast mixing time, does not hold on social graphs where edges correspond to strong real-world trust (e.g. DBLP, Physics co-authorship, Epinions, etc.) [129]. Thus, community-based Sybil detectors do not perform well on “trusted” social graphs. To date, however, no large scale studies have been performed to validate the assumptions of community-based Sybil detectors on untrusted social networks such as Facebook and Twitter.

In this chapter, we describe our efforts to detect and understand Sybil account activity in Renren. In Section 5.2, we use ground truth data on Sybils provided by Renren Inc. to characterize Sybil behavior. We identify several behavioral attributes that are unique to Sybils, and leverage them to build a measurement based, real-time Sybil detector. Our detector is currently deployed on Renren’s production systems, and between August 2010 and February 2011 it led to the banning of over 100,000 Sybil accounts.

In Section 5.3 we analyze the graph structural properties of Sybils on Renren, based on the 100,000 Sybils identified by our detector, as well as 560,000 more identified by Renren using prior techniques. Most interestingly, we find that contrary to prior conjecture, Sybil accounts in Renren do not form tight-knit communities: >70% of Sybils do not have any edges to other Sybils at all. Instead, attackers use biased random
Chapter 5. Social Sybils

![Graph 1](image1.png) ![Graph 2](image2.png)

**Figure 5.1:** Friend invitation frequency over two time scales.

**Figure 5.2:** Percent of accepted outgoing friend requests.

sampling to identify and send friend requests to popular users, since these users are more likely to accept requests from strangers. This strategy allows Sybil accounts to integrate seamlessly into the social graph.

We analyze the remaining 30% of Sybils that are friends with other Sybils, and discover that 69% (65,000 accounts) form a single connected component. By analyzing the creation timestamps of these edges, we determine that this component formed accidentally, not due to coordinated efforts by attackers. We briefly survey several popular Sybil management tools, and show that large Sybil components can form naturally due to the biased way these tools target friend requests.

In Section 5.4, we examine collusion between spamming Sybils on Renren. We show that although Sybils do not have explicit friendship links with each other, Sybils collude behind the scenes to promote spam blogs, revealing that they are under the control of a single attacker. We use content similarity and temporal correlation to quantify the inferred links between Sybils, and quantify the extent of Sybil collusion on Renren.
Finally, in Section 5.5, we investigate enhanced techniques to identify stealthy Sybils. We show that by combining community detection with friend request statistics, we can detect Sybils that collude to evade our currently deployed Sybil detector. These results demonstrate that although Sybil behavior may evolve in the future, we will still be able to accurately detect and ban these malicious accounts.

5.2 Detecting Sybils

In this section, we set the backdrop for our data analysis. First, we briefly describe the role of Sybil accounts in Renren. Second, we describe experiments characterizing Sybil accounts on a verified ground-truth dataset provided by Renren. Finally, we describe and build a real-time Sybil account detector deployed on Renren, and show how it led to the large Sybil dataset we analyze in the remainder of the chapter.

5.2.1 Sybils on Renren

As its user population has grown, Renren has become an attractive venue for companies to disseminate information about their products. This has created opportunities for Sybil accounts to spam advertisements for companies, a growing trend observed by the analytics team at Renren. The increased prevalence of spam on Renren mirrors our findings for spam on Facebook (in Chapter 4) and other’s findings for Twitter [78, 170].
To effectively attract friends and disseminate advertisements, most Sybil accounts on Renren blend in extremely well with normal users. They tend to have completely filled-out user profiles with realistic background information, coupled with attractive profile photos of young women or men, making their detection quite challenging.

Prior to this project, Renren had already deployed a suite of orthogonal techniques to detect Sybil accounts, including: using thresholds to detect spamming, scanning content for suspect keywords and blacklisted URLs, and providing Renren users with the ability to flag accounts and content as abusive. However, these techniques are generally ad-hoc, require significant human effort, and are effective only after spam content has been posted. To improve security for their users, Renren began collaborating with our research team in December 2010 to augment their detection systems with a systematic, real-time solution. To support the project, Renren provided full access to user data and operational logs on their servers, as well as allowing us to test and deploy research prototypes of Sybil detectors on their operational network.
Defining Sybils. In this chapter, as in prior work [57, 174, 199, 200], we are interested in detecting and deterring the use of mass Sybil identities by malicious users. We broadly define Sybils as fake accounts created for the purpose of performing spam or privacy attacks against normal users. We observe that the main goal of Sybils is to increase the power of the attacker by amassing friend links to normal users, thus integrating themselves into the social graph. Attackers create many Sybils to increase their coverage and penetration of the social graph, as well as to combat attrition from Sybils getting banned. Although penetrating the social graph is a precursor for other malicious activity (e.g. spamming), our work is agnostic to these secondary goals, as well as the specific methods and tools used to create and manage the Sybils.

Our definition of Sybils does not include fake accounts generated by users for benign purposes, such as preserving privacy and anonymity, acting on behalf of young children, separating work and personal identities, etc. These “benign Sybils” act just like normal accounts, and therefore do not fall under our definition of malicious Sybils. As discussed in Section 5.2.3, benign Sybils are unlikely to be flagged by the techniques proposed in this work.

5.2.2 Characterizing Sybil Accounts

Our approach to building a real-time Sybil detector begins by first identifying features that distinguish Sybil accounts from normal users. To help, Renren provided us
with two sets of user accounts, containing 1000 Sybil accounts and 1000 non-Sybil accounts, respectively. The Sybil accounts were previously identified using existing mechanisms. A volunteer team carefully scrutinized all accounts in both sets to confirm they were correctly classified by looking over detailed profile data, including uploaded photos, messages sent and received, email addresses, and shared content (blogs and web links).

Using this dataset as our ground truth, we searched for behavioral attributes that serve to identify Sybil accounts. After examining a wide range of attributes, we found four potential identifiers. We describe them each in turn, and illustrate how they characterize Sybils in our dataset.

**Invitation Frequency.** Invitation frequency is the number of friend requests a user has sent within a fixed time period (*e.g.* an hour). Figure 5.1 shows the friend invitation frequency of our dataset, averaged over long term (400 hour) and short term (1 hour) time scales. Since adding friends is a goal for all Sybil accounts, they are much more aggressive in sending requests than normal users. There is a clear separation: accounts sending more than 20 invites per time interval are Sybils. This result holds true at both long and short time scales, meaning that invitation frequency can be used to detect Sybils without significant delays. For example, a threshold of 40 requests/hour can identify $\approx 70\%$ of Sybils with no false positives. Prior to our work, Renren deployed
a CAPTCHA that users must solve if they send $\geq 50$ requests in a day, which explains the apparent upper limit on friend requests.

**Outgoing Requests Accepted.** A second distinguishing feature is the fraction of outgoing friend requests confirmed by recipients. The CDF shown in Figure 5.2 shows a distinct difference between Sybils and normal users. In general, non-Sybil users generally have high accepted percentages with an average of 79%. However, on average only 26% of friend requests sent by Sybil accounts are accepted. This is unsurprising, since normal users typically send invites to people with whom they have prior relationships, whereas Sybils target strangers.

Despite prior studies that show users accept requests indiscriminately [9, 164], our results show that most users can still effectively identify and decline invitations from Sybils. The fact that some users still accept requests from Sybils is explained by two factors. First, most Sybils target members of the opposite sex by using photos of attractive young men and women in their profiles. While women make up 46.5% of the overall Renren user population, they make up 77.3% of the Sybils in our dataset. Second, Sybils typically target popular, high-degree users who are more likely to be careless about accepting friend requests from strangers. We further explore this point in Section 5.3.3.

**Incoming Requests Accepted.** Figure 5.3 plots a CDF of users by the fraction of incoming friend requests they accept. The incoming requests accepted by normal users
are spread across the board. In contrast, Sybil accounts are nearly uniform in that they accept all incoming friend requests, \(e.g.\) 80\% of Sybils accepted all friend requests. In fact, many of the Sybils with \(<100\%\) accept rate also likely fall into this category because Renren banned them before they could respond to all outstanding requests. However, since Sybil accounts receive few friend requests, this detection mechanism can incur significant delay.

**Clustering Coefficient.** Clustering coefficient is a graph metric that measures the mutual connectivity of a user’s friends. Since normal users tend to have a small number of well-connected social cliques, we expect them to have much higher clustering coefficient values than Sybil accounts, which are likely to befriend users with no mutual friendships. Figure 5.4 plots the CDF of clustering coefficient values for each user’s first 50 friends (sorted by time). As expected, non-Sybil users have clustering coefficient values orders of magnitude larger than Sybil users (average clustering coefficient values of 0.0386 and 0.0006 respectively). Since clustering coefficient can be computed based on invitations only (\(i.e.\) user responses are not required) it performs well as a real-time Sybil detection metric.
Chapter 5. Social Sybils

<table>
<thead>
<tr>
<th></th>
<th>SVM Predicted</th>
<th>Threshold Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sybil</td>
<td>Non-Sybil</td>
</tr>
<tr>
<td>True Sybil</td>
<td>98.99%</td>
<td>1.01%</td>
</tr>
<tr>
<td>Non-Sybil</td>
<td>0.66%</td>
<td>99.34%</td>
</tr>
</tbody>
</table>

Table 5.1: Performance of SVM and threshold classifiers.

5.2.3 Building and Running a Sybil Detector

Our analysis results indicate that a threshold based scheme can effectively detect most Sybil accounts. Our next step is to verify this assertion by comparing the efficacy of a simple threshold detector against a more complex learning algorithm.

We apply a support vector machine (SVM) classifier to our ground truth dataset of 1000 normal users and 1000 Sybils. We randomly partition the original sample into 5 sub-samples, 4 of which are used for training the classifier, and the last used to test the classifier. The results in Table 5.1 show that the classifier is very accurate, correctly identifying 99% of both Sybil and non-Sybil accounts. We compare these results to those of a threshold-based detector: outgoing requests accepted % < 0.5 ∧ frequency > 20 ∧ clustering coefficient < 0.01. Our results show that a properly tuned threshold-based detector can achieve performance similar to the computationally expensive SVM.

Real-time Sybil Detection. Our analytical results using the ground-truth dataset led to the design of an adaptive, threshold-based Sybil detector that identifies Sybil accounts in near real-time. The detector monitors all accounts using a combination of
friend-request frequency, outgoing request acceptance rates, and clustering coefficient. It uses an adaptive feedback scheme to dynamically tune threshold parameters on the fly\(^2\). Tuning the thresholds minimizes the likelihood of false positive classifications of normal accounts as Sybils. Because our system works by detecting abnormal behavior in friend or content dissemination, it is unlikely to detect benign Sybils that behave like normal users.

Our detector incorporates real-time changes in friendship links when calculating acceptance percentages. In some cases, normal users accept friend requests from Sybils, only to later revoke the friendship. This causes the accept percentage for the Sybil to drop. Similarly, when Renren bans Sybils all of their edges are destroyed. This causes the acceptance percentages for other Sybils they are linked with to drop. In both cases, the decrease in acceptance percentage helps our detector to more accurately detect Sybils.

After offline testing, Renren deployed our Sybil detection mechanism in late August 2010, and it has been in continuous operation ever since. From August 2010 to February 2011, Renren administrators used our mechanism to detect and subsequently ban \(\approx\)100,000 Sybil accounts. In addition to these accounts, Renren provided us with data on \(\approx\)560,000 accounts that were detected and banned using prior techniques from

\(^2\)We omit details of the adaptive scheme for Renren’s security and confidentiality.
Chapter 5. Social Sybils

2008 to February 2011. For the remainder of this chapter, we will use all of these Sybil accounts (660,000 in all) to study the behavior of Sybil accounts.

**Sybil Account Behavior.** We confirmed that the Sybil accounts identified by our detector are actually malicious by analyzing the content generated by these accounts. Offline analysis confirmed that 67% of content generated by Sybils trips Renren’s spam detectors (*e.g.* suspicious keyword filter, blacklisted URLs, *etc.*). Of the remaining accounts, the vast majority were banned before they had a chance to generate any content. Section 5.4 delves into the details of Sybil behavior and spam content on Renren.

**False Positives.** To assess false positives, we examine feedback to Renren’s customer support department. Renren operates a telephone number and email address where customers can attempt to get banned accounts reinstated. Complaints are evaluated by a human operator, who determines if the account was banned erroneously.

We use the complaint rate, measured as the number of complaints per-day divided by the number of accounts banned per-day, as an upper-bound on false positives. During the two week period between December 13 and 26, 2010, Renren received \( \approx 50 \) complaints per-day, with the complaint rate being \( \approx 0.015 \), which is extremely low. Of these complaints, manual inspection confirms that 48% of the accounts are Sybils, meaning that attackers attempted to recover Sybils by abusing the account recovery process. The majority of the remaining complaints can be attributed to compromised accounts. Thus, the true false positive rate is even less than the daily complaint rate.
Chapter 5. Social Sybils

5.3 Sybil Topology

In this section we analyze the graph topological characteristics of Sybils on Renren. In particular, we are interested in analyzing whether Sybils in the wild are vulnerable to identification using the community-based Sybil detectors that have been proposed by researchers. Refer to Section 3.7.2 for detailed descriptions of these algorithms and how they function. Although Sybil community detectors have been shown to work on synthetic graphs (i.e. real social graphs with Sybil communities artificially injected), to date no studies have demonstrated their efficacy at detecting Sybils in the wild.

Our results show that Sybils on Renren do not conform to the assumptions of existing work. Analysis of the degree distribution of Sybil accounts demonstrates that, contrary to expectations, the vast majority of Sybils do not form social links with other Sybils. Furthermore, temporal analysis of social links between Sybils indicates that these connections are often formed randomly by accident, rather than intentionally by attacker. Thus, Sybils on Renren are not amenable to identification by community-based Sybil detectors.

5.3.1 Sybil Edges

We begin our analysis of Sybil topology by examining the degree distribution of Sybils on Renren. Our goal is to test the most basic assumption of community-based
Chapter 5. Social Sybils

Sybil detectors: *do Sybils in the wild form tight-knit communities?* In order for Sybils to cluster, they must have at least one edge to another Sybil, otherwise they will be disconnected.

Figure 5.5 shows the degree distribution of all 667,723 Sybil accounts compared to the degree distribution of the complete, 42 million node crawl of the Renren presented in Section 3.2.3. When all edges attached to Sybils are considered, the degree distribution is unremarkable. Sybils tend to have fewer friends than normal users, but both Sybils and the overall population follow the same general trend.

However, when we restrict the distribution to only edges between Sybils, we discover an unexpected result: only 20% of Sybils are friends with one or more other Sybils. This indicates that the vast majority of Sybils do not cluster with other Sybils. Instead, most Sybils only form attack edges, and thus totally integrate into the normal social graph.

---

**Figure 5.5:** The degree of Sybil accounts.

**Figure 5.6:** The size of connected Sybil components.
Chapter 5. Social Sybils

<table>
<thead>
<tr>
<th>Sybils</th>
<th>Sybil Edges</th>
<th>Attack Edges</th>
<th>Audience</th>
</tr>
</thead>
<tbody>
<tr>
<td>63,541</td>
<td>134,941</td>
<td>9,848,881</td>
<td>6,497,179</td>
</tr>
<tr>
<td>631</td>
<td>1153</td>
<td>104,074</td>
<td>21,014</td>
</tr>
<tr>
<td>68</td>
<td>67</td>
<td>7,761</td>
<td>7,702</td>
</tr>
<tr>
<td>51</td>
<td>50</td>
<td>15,349</td>
<td>15,179</td>
</tr>
<tr>
<td>37</td>
<td>40</td>
<td>14,431</td>
<td>13,886</td>
</tr>
</tbody>
</table>

Table 5.2: Statistics for the five largest Sybil components.

Figure 5.7: Scatter plot of Sybil edges vs. attack edges for Sybil components on Renren.

Figure 5.8: The order of adding Sybil friends for 1,000 Sybils. Each column represents an individual Sybil.

5.3.2 Sybil Communities

We now shift our focus to the minority of Sybils that do connect to other Sybils. Although we can conclude from Figure 5.5 that most Sybils in the wild do not obey the key assumption of community-based Sybil detectors, it is still possible that the connected minority are vulnerable to community detection. Thus, we now seek to answer the following questions: what are the characteristics of Sybil communities on Renren, and would community-based Sybil detectors be able to identify them?
To bootstrap our analysis, we construct a graph consisting solely of Sybils with at least one edge to another Sybil. The resulting graph is highly fragmented: it consists of 7,094 separate connected components. Figure 5.6 shows the size distribution of these Sybil components. The distribution is heavy tailed: although 98% of Sybil components have fewer than 10 members, the vast majority of Sybil accounts belong to a single, large connected component. Table 5.2 lists the details for the five largest Sybil components.

In order for Sybil communities to be identifiable by existing algorithms, they must form tight knit communities. Put another way, the number of Sybil edges inside the community must be greater than the number of attack edges that connect to honest nodes. However, as shown in Table 5.2, this assumption does not hold for the largest Sybil components on Renren.

Figure 5.7 shows a scatter plot comparing the number of Sybil edges and attack edges in each Sybil component on Renren. All components are above the 45° line, meaning that they have more attack edges than Sybil edges. Thus, no components meet the requirements for detection using existing community-based Sybil identification algorithms.
5.3.3 Sybil Edge Formation

We now examine the processes driving the formation of Sybil edges on Renren. In particular, we seek to determine if edges between Sybils are intentionally created by attackers. If so, then this means community detection may still be a viable approach to detecting Sybils on OSNs. However, if Sybil edges are created unintentionally, then this raises a new question: what process drives the accidental creation of Sybil edges?

**Temporal Characteristics.** One simple litmus test for identifying intentional Sybil edge creation is examining the order in which edges were established. If Sybil edges are formed intentionally by attackers, then we would expect to see them created sequentially, before friend requests are sent out to normal users. This behavior maximizes the utility of Sybil edges by giving Sybils the appearance of “normal” friend relations, thus (potentially) deceiving normal users into accepting friend requests from Sybils.

Figure 5.8 shows the order in which edges were created for 1,000 random Sybils drawn from the largest Sybil component on Renren (containing 63,541 Sybils). For each Sybil $i$ with $n$ edges, we construct the sequence of edges $\langle f_1, f_2, \ldots, f_n \rangle$, sorting the edges chronologically by creation time. Each column of the figure shows the sequence of edge creations for a particular Sybil, with black dots representing Sybil edges.
Chapter 5. Social Sybils

Figure 5.9: Degree distribution of the largest Sybil component.

Figure 5.10: Degree distribution of normal users (from [92]) and targets of friend requests from Sybils.

As shown in Figure 5.8, the order of Sybil edge creation is almost uniformly random. This indicates that the vast majority of Sybil edges in the large component were formed accidentally: attackers had no intention to link Sybils together and form a connected component. Intentionally created connections between Sybils appear as solid vertical lines in the figure. We highlight two examples in Figure 5.8. It is unclear why a tiny minority of Sybils exhibit correlated behavior.

Sybil Degree. In order to reinforce the idea that the vast majority of Sybil edges in the large component are not intentionally created, we plot the degree distribution of the large component in Figure 5.9. 34.5% of Sybils only connect to 1 other Sybil, and 93.7% connect to \( \leq 10 \). It is unlikely that an attacker would expend the effort to link Sybils in such a loose way, since these edge counts are not high enough to make Sybils appear legitimate to normal users.
Chapter 5. Social Sybils

<table>
<thead>
<tr>
<th>Tool Name &amp; URL</th>
<th>Platform</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Renren Marketing Assistant V1.0</td>
<td>Windows</td>
<td>$37</td>
</tr>
<tr>
<td><a href="http://www.duote.com/soft/30348.html">http://www.duote.com/soft/30348.html</a></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Renren Super Node Collector V1.0</td>
<td>Windows</td>
<td>Contact</td>
</tr>
<tr>
<td><a href="http://www.snstools.com/snstool/86.html">http://www.snstools.com/snstool/86.html</a></td>
<td></td>
<td>Author</td>
</tr>
<tr>
<td>Renren Almighty Assistant V5.8</td>
<td>Windows</td>
<td>Contact</td>
</tr>
<tr>
<td><a href="http://www.sns78.com/">http://www.sns78.com/</a></td>
<td></td>
<td>Author</td>
</tr>
</tbody>
</table>

Table 5.3: Popular Sybil creation and management tools.

Figure 5.11: Sessions per day for Sybil and normal users.

Figure 5.12: Overall distribution of session durations.

Biased Random Sampling. At this point we have established that attackers do not create the vast majority of Sybil edges intentionally; instead, they appear to occur randomly by accident. To understand how this happens, we conducted a brief survey of three software tools used to manage Sybil accounts on Renren. The details for each tool are given in Table 5.3. The purpose of these tools is to automate the process of creating Renren accounts, forming edges between the Sybils and other users, and posting content en-mass.
Chapter 5. Social Sybils

The documentation for the tools in Table 5.3 state that they select targets for friend-
ing by performing biased random sampling on the social graph to locate popular users.
In practice, the tools simplify this process by leveraging Renren’s friend recommend-
dation features, which are designed to highlight popular, well connected users. Thus,
attackers are abusing Renren’s features to locate a biased random sample of targets for
friending.

Although we cannot be certain whether the Sybils in our dataset were created using
the tools in Table 5.3, we can show that Sybils on Renren do bias friend requests toward
high-degree nodes. Figure 5.10 shows the degree distribution for all users that received
friend requests from Sybils, and illustrates that it is skewed toward high-degrees when
compared to the actual degree distribution of the Renren population.

Based on the advertised functionality of these tools, and the results in Figure 5.10,
we can surmise that Sybil edges are created accidentally due to two factors. First, the
goal of Sybils is to accrue many friends by sending out numerous friend requests. If a
Sybil is successful, it becomes popular by virtue of its large social degree. Second, the
biased random sampling performed by Sybil management tools is intentionally geared
toward locating popular users. Thus, it is likely that these tools will, unbeknownst
to the attacker, occasionally select Sybil nodes to send friend requests to. As shown
in Figure 5.3, Sybils almost always accept incoming friend requests, hence when this
situation arises a Sybil edge is likely to be created.
5.4 Spam Strategies and Collusion

In Section 5.3 we revealed that the vast majority of Sybils on Renren do not form explicit friend connections with each other. However, this does not imply that each Sybil is independently controlled by a different attacker. In this section, we uncover collusion between Sybils by leveraging spam content similarity and temporal correlation to locate inferred relationships.

We begin by presenting an overview of the strategies used by Sybils to disseminate spam on Renren. The most common strategy is “sharing” links to spam blogs, and we present a detailed case study on this phenomenon. The results of the case study indicate that Sybils share links to the same content at about the same time. These observations motivate us to cluster Sybils based on content similarity and temporal correlation (the same methodology used in Section 4.3). Our results reveal that even under strict correlation thresholds, Sybils form large connected components. This indicates that Sybils are being controlled collectively by colluding attackers behind-the-scenes.

5.4.1 Share Spam on Renren

We begin our analysis by characterizing the overall spamming behavior of Sybils on Renren. On Renren, the dominant method for Sybils to disseminate spam is to “share” links to spam content. As shown in Table 5.4, shares per Sybil far outnumber status
Chapter 5. Social Sybils

<table>
<thead>
<tr>
<th>Sybil Action</th>
<th>Avg. per Sybil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write a blog post</td>
<td>0.11</td>
</tr>
<tr>
<td>Update status</td>
<td>2.13</td>
</tr>
<tr>
<td>Upload photo</td>
<td>1.81</td>
</tr>
<tr>
<td>Post to a friend’s wall</td>
<td>0.73</td>
</tr>
<tr>
<td>Share a link</td>
<td>6.88</td>
</tr>
</tbody>
</table>

Table 5.4: Average actions per Sybil on Renren after June 1, 2009.

updates and wall posts. Sharing is an advantageous strategy because the attacker only needs to generate one unique piece of spam, i.e. the original piece of content. Sybils simply forward links to this content to all their friends. Users on all OSNs engage in link sharing (e.g. “liking” on Facebook), so this behavior is not overtly suspicious. However, as we show later, there are quantifiable differences between Sybil and normal sharing.

Out of our complete dataset of 660K Sybils, about 64% have not shared any content due to being banned by Renren before beginning to spam. Hence, in this section we focus on the remaining 237,205 Sybils that have shared content. In total, these Sybils shared 3,491,988 links. Figure 5.13 shows the number of shares per Sybils. 25% of Sybils only share a single piece of content before they are caught and banned. <1% of Sybils go uncaught long enough to share ≥100 links.

To investigate the purpose of spam on Renren, we manually examined the shares of a random sample of 1,000 Sybils. We found that Sybils on Renren only share two types of links:
Chapter 5. Social Sybils

Figure 5.13: CDF of shares per Sybil.  
Figure 5.14: Distribution of shares across spam blogs.

1. **Blogs**: 62.5% of shares from Sybils link to spam blog posts. These blogs engage in typical spam activities, such as promoting shady online merchants and selling pharmaceuticals. Many of these blogs attempt to obfuscate themselves by copying content from popular, legitimate blogs, and then slightly modifying the content to include spam text and links.

2. **Videos**: The remaining Sybil shares link to bogus online videos. These videos use provocative titles and thumbnail images to entice users to click on them. Users are then redirected to sites that include spam content.

### 5.4.2 Case Study: Spam Blogs

We now turn our attention to spam blogs, and the Sybils that promote them. We focus on shares of spam blogs because they are the most prolific type of spam on Ren-
Chapter 5. Social Sybils

ren. We begin by addressing the fundamental question: *are Sybils colluding to promote spam blogs, or is each Sybil operating independently?*

To answer this question, we calculated the amount of duplication among the spam blogs promoted by Sybils. Among the 3 million individual spam shares in our dataset, only 302,333 unique spam blogs are promoted. Figure 5.14 shows the number of shares for each spam blog, sorted from most to least promoted. The top 30 spam blogs are highly promoted, each garnering \( >10,000 \) shares. 25\% of spam blogs receive \( \geq 2 \) shares from Sybils, which is the bare minimum evidence for collusion. However, this result is a lower bound, since we only compared the links shared by Sybils on Renren. It is possible that these links eventually redirect to the same external websites, which would indicate additional levels of collusion among attackers.

**Information Dissemination.** One reason that Sybils on Renren collude to promote spam blogs has to do with the trending content section on the Renren homepage. Each day, the 100 most popular blog posts, images, and videos on Renren get featured on the sites homepage, which receives millions of hits per day. Attackers can use Sybils to inflate the popularity of spam blogs and try to make them artificially trend. Other researchers have observed similar attacks against Digg [174] and Twitter [78]. Currently, Renren relies on manual inspection by humans to filter spam out of the trending content section.
This raises our next question: \textit{does Sybil collusion create quantifiable differences in the dissemination pattern of spam content?} Intuitively, content on OSNs should spread organically along social links. A user posts a link, then friends and friends-of-friends share, “like,” and “retweet” it to an ever-expanding audience. Researchers have identified information dissemination trees matching this pattern on many OSNs [105]. However, as we showed in Section 5.3, Sybils are not friends with each other. Thus, each Sybil that shares a spam link creates an independent information dissemination tree.

To test whether this intuition is true, we took a snapshot of the 100 most popular blogs on Renren on February 2, 2011. We manually inspected each blog and determined that 26 were spam, \textit{i.e.} they included overt spam content, and/or links to malicious websites. Figure 5.15 shows the number of edges that connect users who share links to the blogs in our sample. There is a clear distinction between normal blogs and spam blogs. Legitimate blogs exhibit an order of magnitude more shared edges, indicating that their popularity has grown organically along friendship links.

In contrast, spam blogs promoted by Sybils have far fewer shared edges. This result makes sense, given that we have shown Sybils have few edges connecting to each other. Normal users are unlikely to re-share links to spam blogs, so their popularity growth is almost entirely driven by disconnected Sybils. Only two spam blogs from our sample approach the cluster of normal blogs in Figure 5.15.
Chapter 5. Social Sybils

![Figure 5.15: Connectivity between users sharing normal and spam blogs.]

![Figure 5.16: The degree distribution for content similarity graphs with varying thresholds.]

5.4.3 Content-based Sybil Components

At this point we have demonstrated that Sybils on Renren collude to disseminate spam content, despite having few social links between them. We now examine whether content similarity can be used to group Sybils into connected components. Intuitively, strongly connected components are likely to be under the control of a single attacker.

We model collusion between Sybils as a content similarity graph. In a content similarity graph, Sybils are nodes and two Sybils are connected if they share similar content. We consider the following similarity criteria: let $s_i$ and $s_j$ be the sets of content that two Sybils $i, j$ share. We define the content similarity between them as $s_{ij} = s_i \cap s_j / s_i \cup s_j$. Content similarity can range from 0 to 1, where 0 means there is no duplication of content and 1 means the two Sybils share exactly same content. We say that two Sybils $i, j$ share similar content if $s_{ij}$ is larger than some threshold $T_s$ (or equal to $T_s$ in the special case of $T_s = 1$).
To understand collusion between Sybils, we built content similarity graphs using three values for $T_s$: 0, 0.5, and 1. $T_s = 0$ is the most lax threshold: pairs of Sybils sharing $\geq 1$ identical piece of content will be connected. This is the same similarity threshold that we use in Section 4.3. Studies of OSN spam from other researchers also use this threshold [78]. $T_s = 1$ is the strictest threshold: all content shared by two Sybils must be identical in order for them to be connected.

Figure 5.16 shows the degree distribution of our three content similarity graphs. Compared to the original social graph (see Figure 5.5), Sybils have many more edges connecting to each other in the content similarity graphs. Even under the tightest threshold $T_s = 1$, over 50% of Sybils have at least one Sybil partner forwarding exactly the same content. These Sybils pairs are obviously under the control of a single attacker.

In the case of moderate and low thresholds, the number of edges significantly increases. Under the most permissive threshold $T_s = 0$, 97% of Sybils have edges to other Sybils. Large connected components that emerge under more permissive thresholds may represent Sybils that are “rented” by spammers. Studies have shown that black-market crowd-sourcing services allow spammers to cheaply hire workers to spam on OSNs for them [131, 184]. These workers (and the Sybils they control) end up being involved in multiple spam campaigns, which helps explain why different groups of Sybils exhibit varying amounts of overlapping content.
Chapter 5. Social Sybils

Next, we study the component structure of the content similarity graph. Figure 5.17 shows the quantity and sizes of connected components under different thresholds, ordered from largest to smallest. As expected, tighter thresholds fragment the graph into a larger number of distinct components, and reduce the size of the largest (giant) component. Under the lax $T_s = 0$ threshold the 237K Sybils form 4.9K connected components. The giant component contains over 219K (90%) Sybils. In contrast, there are 76K connected components under the moderate $T_s = 0.5$ threshold. The giant component only contains 84K (35%) Sybils, but the next 100 largest components contain an additional 60% of the Sybils. Finally, under the tight $T_s = 1$ threshold there are 114K components, with the largest only containing 3700 Sybils. In this scenario, the first 1000 components only contain 40% of the overall Sybils.

The results in Figure 5.17 confirm our intuition that content-similarity can be used to identify groups of colluding Sybils. Tight thresholds pinpoint Sybils that were created and managed by a single attacker. The maximum size of these components demonstrates the upper bound on the number of Sybil accounts a single attacker can reasonably generate. More relaxed thresholds identify Sybils that belong to several overlapping spam campaigns, possibly under the control of different attackers at different times. These components can grow to huge sizes, as attrition forces attackers to gradually replace old Sybil accounts with new ones.
5.4.4 Temporal Correlation Between Sybils

We now investigate whether there are temporal correlations between Sybils that exhibit content similarity. We suspect that Sybils under the control of a single attacker will be active at similar times engaging in coordinated spam campaigns.

Let $t_i$ and $t_j$ be the sets of links that two Sybils $i, j$ share during time interval $S$. We define the temporal similarity between the Sybils as $t_{ij} = t_i \cap t_j / t_i \cup t_j$. Like content similarity, temporal similarity can range from 0 to 1, with 0 meaning no overlap, and 1 meaning exact overlap. The size of time interval $S$ can be varied to control the granularity of comparisons. In our experiments, we evaluate time similarity over two time intervals: one hour and one day.

To test our hypothesis, Figure 5.18 shows content similarity versus time similarity for Sybils in our dataset under two time intervals. Each line plots the average time similarity for discreet sets of Sybil pairs with close content similarity. For example, the first point of the hour-scale line represents the average time similarity for all pairs of Sybils with content similarity in the range 0 to 0.1. The error bars show the standard deviation for each data point.

Figure 5.18 reveals that time similarity is roughly proportional to content similarity. As expected, Sybils that share similar content tend to do so at similar times, reinforcing our conclusion that Sybils are colluding. Setting the time interval to one day significantly increases the amount of collusion we can identify. Sybils that share near-
Chapter 5. Social Sybils

Figure 5.17: Sybil connected component sizes under different thresholds.

identical content (content similarity $\approx 0.95$) also exhibit $\approx 0.92$ time similarity under the one day threshold. This shows that attackers move rapidly to complete social spam campaigns using many Sybils within short time spans.

5.5 Moving Sybil Defense Forward

Currently, our detector relies on specific features of Sybil behavior in order to identify these accounts. However, in the future, attackers may try to avoid our detector by modifying the behavior of Sybils. One feature used by our detector is the friend request acceptance percentage. An attacker could inflate the acceptance percentage of their Sybils by sending friend requests to other Sybils, who are guaranteed to accept.

In this section, we examine this alternative attack model. First, we formally define the parameters of the new attack model. Next, we conduct simulated attacks against a real social graph: the Peking University regional network of Renren. We observe
that Sybils colluding to inflate their acceptance percentage inevitably form tight-knit communities. However, our results show that the false positive rates of Sybil community detectors are unacceptably high. Thus, we propose a two-step solution that uses community detection and the external acceptance percentage of communities to detect Sybils utilizing the new attack model.

### 5.5.1 Attack Model and Experimental Setup

Consider an attacker that controls $N$ total Sybils. In order to avoid detection, each Sybil must maintain a friend request acceptance percentage of at least $\beta$. Let $\alpha$ be the probability that normal users accept friend requests from Sybils. In order to keep a high accept percentage, let each Sybil send friend requests to other Sybils with probability $p$, and to normal users with probability $1 - p$. Thus, in order to avoid detection, each Sybil must send friend requests that obey the following inequality: $\beta \leq p + \alpha(1 - p)$.

If each Sybil sends $n$ total friend requests, then each Sybil will create $n \times p$ Sybil edges and $n \times \alpha(1 - p)$ attack edges.

**Experimental Setup.** We investigate the graph structure of Sybils that follow this attack model by simulating attacks against a real social graph. We use the Peking University (PKU) regional network on Renren as our target social graph, since its size is reasonable (170,000 nodes) and its properties have been thoroughly studied in Chapter 3. In our simulation we create $N$ Sybils, each of which sends $n$ friend requests.
Chapter 5. Social Sybils

divided between normal users and Sybils according to the above inequality. We fix \( \alpha = 0.26 \), which is the median acceptance percentage for Sybil friend requests in our dataset (see Figure 5.2). Similarly, we set \( \beta = 0.5 \), which is the detection threshold used by our deployed detector.

We experiment with many combinations of attack parameters in order to observe their effects on Sybil graph structure. As shown in Table 5.5, we test small and large groups of Sybils (column \( N \)). Given \( \alpha = 0.26 \) and \( \beta = 0.5 \), \( p \) can be calculated using the above inequality (\( p = 0.33 \) in our simulations). For each value of \( N \), we vary \( n \), the number of friend requests sent per Sybil. Table 5.5 shows the number of Sybil edges and attack edges per Sybil as \( n \) varies.

In order to avoid creating Sybil clusters that obviously deviate from normal graph structure, the attacker can target Sybil edges in such a way as to create “natural” looking clusters. In our simulations, we use two common models to direct the creation of Sybil edges: (i) Erdős-Rényi, where the attacker links randomly chosen Sybils, and (ii) Preferential Attachment, where the destination of each Sybil edge is chosen proportionally to the destination Sybil’s degree. Normal users are selected uniformly at random as targets for attack edges.
Chapter 5. Social Sybils

5.5.2 Simulation Results

Sybil Community Detection. We now seek to answer the question: do Sybils form strong communities when attackers follow the new attack model? To answer this question, we use the community detection algorithm developed by Blondel et al. [33] to locate communities in the graphs listed in Table 5.5. In each case, we locate the community with the largest number of Sybils, and evaluate its precision (the ratio of Sybils to total number of nodes within the community) and recall (the ratio of Sybils within the community to total number of Sybils). Intuitively, these metrics tell us the false positive (precision) and false negative (recall) rates of the community detector when we use it to isolate Sybils.

The results shown in Table 5.5 are mixed. On one hand, when \( n \leq 300 \) the community detector is able to identify Sybils with high precision. However, as \( n \) grows, the

---

<table>
<thead>
<tr>
<th>( N )</th>
<th>( n )</th>
<th>Edges per Sybil Attack</th>
<th>Erdős-Rényi Precision</th>
<th>Recall</th>
<th>Pref. Attach. Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>100</td>
<td>33 17</td>
<td>99.4% 1</td>
<td></td>
<td>99.8% 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>99 52</td>
<td>98.3% 1</td>
<td></td>
<td>97.8% 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>165 87</td>
<td>89.4% 1</td>
<td></td>
<td>91.3% 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>330 174</td>
<td>79.6% 1</td>
<td></td>
<td>79.5% 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>660 348</td>
<td>64.3% 1</td>
<td></td>
<td>63.4% 1</td>
<td></td>
</tr>
<tr>
<td>10K</td>
<td>100</td>
<td>33 17</td>
<td>98.9% 1</td>
<td></td>
<td>98.5% 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>330 174</td>
<td>82.9% 1</td>
<td></td>
<td>82.7% 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>660 348</td>
<td>72.9% 1</td>
<td></td>
<td>72.5% 1</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Simulated Sybil attacks against the PKU graph.
Chapter 5. Social Sybils

so does the false positive rate. Recall is consistently 1 (\textit{i.e.} all Sybils are caught), but many normal users are mistakenly grouped into the Sybil community as well.

The results in Table 5.5 indicate that community detectors alone are not practical for locating Sybils using the new attack model. A stealthy attacker could create many Sybils, add Sybil and attack edges over a period of many days until \( n \) is large, then begin using the Sybils to send spam. Thus, by the time the Sybils become actively malicious, a community detector will generate many false positives when attempting to isolate the Sybil community.

\textbf{External Acceptance Percentage.} In order to identify Sybil communities without incurring false positives, we introduce a new feature: the \textit{external acceptance percentage}. The external acceptance percentage is the fraction of friend requests sent by members of a community to users outside the community that are accepted. Intuitively, this feature is useful because accepted Sybil friend requests tend to be from other Sybils \textit{inside} the local community. Conversely, rejections are from normal users \textit{outside} the local community.

Figure 5.19 shows the external acceptance percentage for normal communities from the PKU network, as well as Sybil communities for our simulated attacks. There is a clear distinction between normal communities and Sybil communities. This distinction holds even as attackers vary \( N \) and \( n \). The Sybils in Figure 5.19 follow the Erdős-Rényi attachment model, but the results are the same for the Preferential Attachment.
Figure 5.19: The external acceptance percentage of normal and Sybil communities.

Thus, OSN operators like Renren can successfully defend against Sybils using the new attack model using a combination of community detection and external acceptance percentage. In practice, the OSN operator would need to run offline community detection on their social graph once per day. Any community with an external acceptance percentage below a certain threshold (e.g. 0.6, as seen in Figure 5.19), would be flagged as suspicious. By running community detection every day, the OSN operator would catch Sybil communities while the number of edges per Sybil is low. Most OSNs limit the number of friend requests each account can send per day (on Renren, the limit is 50). This slows the growth of $n$ in Sybil communities, and gives the OSN operator a window of several days to catch the Sybil community before the precision of the community detector falls below 99% (see Table 5.5).
5.6 Summary of Results and Ongoing Work

Our results in this chapter make several key contributions to the understanding of Sybils on large scale OSNs. Some key observations include:

- **80% of Sybils on Renren do not connect to** any other Sybils. Instead, they only connect with normal users and blend completely into the social graph.

- The Sybils that are connected form loose, rather than tight-knit clusters. *The majority of links between these Sybils form accidentally*, rather than intentionally by attackers.

- Despite not having explicit social ties, Sybils collude to promote large spam campaigns. **50% of Sybils have at least one partner that shares identical content with them.**

We isolate several unique features of Sybil behavior and leverage them to create a real-time Sybil detection system. We show that a computationally efficient, threshold-based classifier is sufficient to *catch 99% of Sybils*, with low false positive and negative rates. Our system is *actively deployed on Renren*, and scales up to handle the load of their production environment.

Using edge creation information for over 660,000 Sybil accounts on Renren, we show that Sybils on Renren *do not obey behavioral assumptions that underlie previous work* on decentralized Sybil detectors. Although we cannot be sure that our results
generalize to all OSNs, our findings for a traditional, i.e. untrusted, OSN, coupled with results from prior work on trusted OSNs [129], suggest that we should explore new approaches to perform decentralized detection of Sybil accounts on OSNs.

**Ongoing Work.** Our work on quantifying and detecting social Sybils is ongoing. We direct interested readers to look at some of our follow-up work that builds on the results presented in this chapter.

Our paper titled “Serf and Turf: Crowdturfing for Fun and Profit” examines the problem of black-market crowdsourcing websites [184]. We perform extensive measurements of two of the largest crowdsourcing sites in China and reveal that millions of dollars are being spent each month paying workers to create Sybils and spam on social websites. We infiltrated these “crowdturfing” websites as customers in order to perform the first ever end-to-end measurements of social spam: from originating workers, to Sybils on the social web, and finally out to unsuspecting users who click on the spam links. Our results highlight the extreme difficulty of using automated detection systems to locate Sybils that are controlled by real people, as opposed to software bots.

In the paper “Social Turing Tests: Crowdsourcing Sybil Detection” we investigate the feasibility of using crowdsourced labor to detect social Sybils [183]. In essence, our goal is to fight fire with fire: attackers are using crowdsourced labor to create Sybils, so perhaps we can use crowdsourced labor to identify these stealthy Sybils. To examine the feasibility of this approach, we collect a large, ground-truth sample of Sybils from
Chapter 5. Social Sybils

Renren and Facebook. We then conduct a carefully controlled user study, asking both experts from academia and workers from crowdsourcing sites (Amazon’s Mechanical Turk in the US, Zhubajie in China) to classify each of our sample accounts as real or fake. The results of this user study are quite positive: experts are very accurate at identifying Sybils, and some crowdworkers also very accurate. Using simulations driven by the results of our user study, we develop a practical system for cheaply and reliably locating Sybils using just crowdsourced labor.
Chapter 6

Related Work

Research is not conducted in a vacuum: our work is part of a continuum of discovery that is still ongoing. In this chapter, we survey related work in three areas of OSN research: measurement studies, security, and privacy. Although these areas do not cover the full breadth of OSN related research, they are the areas that the author has personally contributed to and is most familiar with. Within each area we divide our discussion into more fine-grained topics.

The goals of this chapter are twofold. First, we aim to capture the current state-of-the-art research in each topic area by pointing out seminal papers, key concepts, and important results. Second, by presenting a high-level overview of each area, we identify research areas that have become crowded and complete, versus areas that still hold unanswered questions.
Chapter 6. Related Work

6.1 OSN Measurements

The single most fundamental aspect of OSN research is measuring and quantifying real-world OSN data. This provides the foundation upon which all other OSN research is based. Here, we briefly survey the literature related to OSN measurement, focusing on four specific areas: static social graphs, user interactions (both visible and latent), and temporal graph dynamics.

6.1.1 Static Social Graphs

Measurements of static social graph topologies are the natural starting point for researchers interested in the burgeoning area of OSNs. As such, static OSN topologies are very well studied: results on all of the major OSNs have been published, and their fundamental properties have been extensively quantified. Understanding the work in this area is a prerequisite for conducting research on OSNs in general. However, publishing new results measuring static graph snapshots is very difficult, and somewhat unnecessary, given that the fundamental discoveries in this area have already been made.

One of the first studies to empirically analyze an online social network (as opposed to a real-life social network) uses data from the Club Nexus website of Stanford University [13]. Since then, Ahn et al. analyze the topological characteristics of Cyworld,
Chapter 6. Related Work

MySpace and Orkut [15]. Mislove et al. measure the structure of Flickr, YouTube, LiveJournal, and Orkut [128]. Fu et al. analyze Xiaonei, the early predecessor to Renren that was only open to college students in China [70].

These studies are important for two reasons: first, they confirm that OSNs exhibit many of the properties that have been observed in other complex networks, including power-law scaling characteristics [25], and high clustering coefficients. This firmly establishes OSNs within the pantheon of small-world graphs [16], allowing years of theoretical work on complex networks to be brought to bear in this domain.

The second reason these studies are seminal is that they establish the high-level measurement methodology that all subsequent OSN studies (including ours) use. This includes how and what to measure when collecting data, what metrics to calculate during graph analysis, and even how to present the data graphically when publishing.

**Twitter.** Twitter deserves a special mention because it is an aberration amongst OSNs. Java et al. published the first measurement study of Twitter, which gives a useful perspective on the OSN early on its history [91]. However, Twitter has changed drastically since 2007, and the measurement results from this study are no longer representative. Kwak et al. published the definitive measurement study on Twitter in 2010, using a 100% complete snapshot of the social graph (with 42 million users and 1.5 billion edges) [105]. This study reveals that the graph structure of Twitter is unlike any other known OSN: it is dominated by celebrities, who are surrounded by millions
of low-degree, low interaction followers. For this reason, Kwak et al. conclude that Twitter behaves more like a news-media than a social network.

6.1.2 Interactions

Visible Interactions. Once the static structures of OSNs were well characterized, researchers shifted focus to analyzing interactions between users. As we have demonstrated throughout this dissertation, understanding user interactions on OSNs is of vital importance, as they add a whole new dimension of information on top of the social graph. More than ten papers have been published on visible interactions alone, and it is safe to say that this specific topic has been completely explored.

Prior to our work on Facebook, there were two studies on visible interactions on OSNs. Leskovec et al. analyzed the MSN instant messaging network, constructing a social graph based on user’s messages to each other [111]. This work is impressive because of the size of the analyzed graph (180 million users), as well as the thorough characterization of how different types of interactions are related to tie-strength. Chun et al. analyzed visible interactions on Cyworld, the largest OSN in South Korea at the time [53]. Their work was the first to model interactions as a graph (the activity network, in their terminology), and explicitly compare its properties to the social graph.

Four papers analyze the intersection between user interactions and popularity on OSNs. Huang et al. measure user prestige and visible interaction preference in Ren-
Chapter 6. Related Work

ren [87], and reach similar conclusions to our work in Chapter 3. Valafar et al. examine how users tag favorite photos on Flickr and discover that interactions are highly skewed towards a small set of the population [176]. 50% of these chatty users are also producers of popular photos, revealing that a small group of core users lie at the heart of the OSN. Cha et al. expand on the analysis of Flickr by using temporal data to trace the spread of popular photos over time [47]. Contrary to expectations, they find that social links are of little utility for disseminating popular photos, and that photos tend to accrue popularity slowly over years rather than quickly through viral propagation. Finally, Cha et al. examine the correlations between popularity and influence among Twitter users [45]. Again, contrary to expectations, they find that the most popular users (in terms of number of followers) are not the best at spreading information, leading to the title phrase “the million follower fallacy.” Yang et al. expand on this work by building models to predict the speed and scale of information cascades on Twitter using the features identified by Cha et al. [196].

Viswanath et al. analyzed the temporal interaction patterns of Facebook users in order to determine how they affect interaction graphs [179]. They find that only 30% of interacting pairs remain stable month to month, i.e. the friends you talk to this month are unlikely to be the people you talk to next month. However, despite the changing nature of the edges in the interaction graph, the overall structural properties of the interaction graph remain stable over time. This is an important result, as it demonstrates that even
a single snapshot of an interaction graph is likely to have representative properties, despite the ephemeral nature of user interactions.

Burke et al. investigate the role of Facebook interactions on emotional factors like loneliness and social bonding [41]. This work is notable because they leverage the definition of interaction graphs defined in Chapter 3 of this work and apply it to studying sociological issues, rather than graph theoretic issues.

The general consensus from the body of research on visible interactions on OSNs reinforces our findings from Chapter 3: only a fraction of all friendship links represent active connections between users, interactions are not spread evenly over users’ friends, and the relative interactivity of different links varies with time.

Two papers have leveraged the observation that visible interactions are heterogeneous across social links to use them as predictors of tie-strength. Choudhury et al. focus on email graphs and use interaction frequency to predict the strength of ties between users [52]. Xiang et al. take things a step further by correlating visible interactions with similarity of profile elements (e.g. shared hometowns, interests, etc.) [194].

**Latent Interactions.** The study of visible interactions on OSNs has naturally led to work focused on latent user interactions. Including our work, there are only four papers on latent interactions on OSNs. Benevenuto et al. collected click-stream data from a Brazilian social network aggregator, and measured silent activities like browsing [31]. Using 12 day’s worth of clicks from 37,000 users, they conclude that: 1) latent browsing
behavior is far more prevalent than visible interactions, 2) users latently interact with an order of magnitude more friends that they visibly interact with, and 3) 22% of profile visits come from strangers, rather than friends.

A subsequent studies by Schneider et al. extracted click-streams from passively monitored network traffic in order to observe how 8,500 users browse OSNs [156]. This study focuses on examining the session level characteristics of users, what OSN features they use, and constructing models that describe user navigation patterns. Although the study identifies a significant amount of profile browsing, no attempt is made to correlate this with the structure of the social graph.

Finally, visible and latent interactions on Facebook have also been characterized [19]. However, the results from this study are suspicious, because they only focus on 16 million Facebook users who visited the site on at least 80% of the days in 2009 and 2010. Thus, it is difficult to draw general conclusions based on measurements of this highly active, self-selected population.

Latent interactions are one area where additional measurements studies would be very useful. Existing studies are constrained by small sample sizes and complicated, error prone methodologies (e.g. reconstructing latent interactions from click-streams is non-trivial). Although latent data is very hard to acquire, a large scale, long term study on latent interactions would be a huge boon, and could potentially validate (or invalidate) all of the results from existing, limited studies.
Chapter 6. Related Work

**Location-based OSNs and Check-ins.** Since the advent of location-based OSNs like Foursquare and Gowalla, numerous large OSNs have incorporated location “check-in” features into their service (e.g. Facebook Places). Cecilia Mascolo’s group at the University of Cambridge has mined Gowalla and Foursquare to extensively analyze this confluence of social and spatio-temporal data. The efforts of this team have single handedly addressed most of the interesting research questions associated with location-based OSNs. Their earliest works focus on the links between social friendship and geographic distance, leading to the discovery that users tend to form social connections with people that are geographically close [153, 154]. Since then, the group has leveraged geographic data from OSNs to improve the performance of systems and algorithms, including using socio-spatial data to improve content delivery networks (CDNs) [152], and improve link prediction algorithms [155].

**Information Dissemination.** The study of user interactions on OSNs naturally leads to a broader discussion of how information disseminates on social graphs. Although a complete discussion of the information dissemination literature is out of scope for this dissertation, Agrawal et al. have published a thorough tutorial that introduces many of the key concepts and research papers in this area [14]. This team has also done original research on classifying the structure of information dissemination cascades on OSNs [40] as well as algorithms for combating the spread of misinformation on OSNs [39].
6.1.3 Temporal Dynamics

The next step beyond static snapshots of graphs and user data is measuring how they evolve over time. This space has become increasingly crowded as more OSNs mature, and datasets about their evolution get quantified. However, the measured data has revealed inconsistencies between real-world OSN growth and the popular preferential attachment (PA) model of network growth. More work needs to be done creating accurate models of social graph temporal evolution. Similarly, there is a dearth of graph metrics for distilling temporal behavior. As it stands, the state-of-the-art is simply computing the deviations between time-varying distributions of existing static graph metrics.

An early study by Leskovec et al. examined the temporal dynamics of citation and patent co-authorship networks [112]. To their surprise, they discovered that the graph diameters shrink over time. They propose the Forest Fire model to capture this property as well as the increasing graph densities present in their data. Kang et al. also examine the temporal dynamics of graph diameters [96].

**Preferential Attachment.** A follow up study by Leskovec et al. measured the temporal dynamics of four OSNs: Flickr, Delicious, Yahoo! Answers, LinkedIn [110]. This is the first study to use real-world OSN data to examine whether the PA model can explain the observed growth patterns on OSNs. The authors find that PA is not a good
model of OSN growth, and propose a new model based on local edge attachment and triangle closures.

Two other studies have also examined the efficacy of PA, but reach opposite conclusions from the previous work. Mislove et al. measure temporal dynamics on Flickr and conclude that PA is a good fit for the observed data [127]. Garg et al. measure dynamics from the OSN aggregator FriendFeed and also conclude that PA fits the data, although they note that the goodness of the fit changes depending on the age of the node, with younger nodes having the worst fit [72]. These results complicate the issue of whether PA is the right model for OSN temporal dynamics, and suggest that more measurements, and definitely more modeling, are needed to resolve these inconsistent results.

**Structural Metrics Over Time.** Two studies focus solely on examining the temporal dynamics of fundamental OSN metrics. Studies of Cyworld [15] and Google+ [75] focus on how the degree distribution, clustering coefficient, assortativity, $k_{nn}$, and network path lengths change over time. Although these metrics often fluctuate due to OSN specific event (e.g. an advertising campaign to sign-up new users), the overall trend is for these metrics to quickly stabilize over time. These results indicate that even though OSNs often have spurts of exponential growth, the overall characteristics of the graphs quickly reach equilibrium.
Chapter 6. Related Work

Component Evolution. Three studies focus on the evolution of the component structure of complex networks. Kumar et al. examine the component structure of Flickr and Yahoo! 360 [104]. They observe that the relative proportions of nodes in the giant connected component (GCC), small disconnected components, and singletons are stable over time. However, the actual nodes in disconnected components and singletons change rapidly over time, as they form new edges and get sucked into the giant component. This result is important because it suggests that graph crawls that only collect the giant connected component of the graph are likely to be largely complete and representative.

McGlohon et al. expand on these findings by measuring the temporal dynamics of components in 13 graphs, mostly representing blog, academic, and patent citation networks [122]. They observe that early on, complex networks are fractured into many small components and the effective diameter rises until a “gelling point” is reached. After the gelling event, the GCC forms and typical complex network properties emerge, such as shrinking diameter and densification. The most important contribution of this work is their graph generator, which is capable of producing synthetic graphs that have disconnected components and follow the observed gelling pattern.

Lastly, Kang et al. develop a model, called CommunityConnection, to describe if and when new nodes in complex networks will join the GCC [95]. Their model is based on the observation that the probability of new nodes joining the GCC exponentially
decreases as the nodes degree increases. However, it is unknown if this observation and
the associated model hold for social networks, since the researchers base their findings
on citation networks and a large web graph.

Group Formation. Three studies examine OSN dynamics through the lens of group
formation. Backstrom et al. analyze the correlations between social graph structure
and user’s propensity to form and join communities [21]. Using data from LiveJournal
(which has explicit user defined communities) and DBLP (academic conferences
are used as communities), the authors find that the number of friends a user has in a
community is a relatively weak predictor of whether that user will join. Instead, users
are much more likely to join a community if their friends are members, and those
friends are well connected to each other. However, there are diminishing returns to this
phenomenon: as a community becomes denser and more “clique-ish,” its growth rate
slows.

Zheleva et al. propose the first generative model that produces social networks as
well as affiliation groups [206]. Using data from LiveJournal, Flickr, and YouTube,
they measure the correlations between node degree and community membership. Their
model produces graphs with power-law degree distribution, densification, and shrinking
diameter, while also labeling nodes with community affiliations that match observed
distributions.
Chapter 6. Related Work

Kairam et al. use data from the Ning social network platform to measure whether groups grow through diffusion (i.e. users go where their friends go) or via other mechanisms [94]. They build a model to predict the growth of groups over short and long timescales, and demonstrate that it has 79% accuracy at predicting the growth and death of groups over time.

6.2 OSN Security

As we show in Chapters 4 and 5, OSNs are now under constant attack by well organized, for-profit online criminals. Boshmaf et al. demonstrate just how easy it is for attackers to infiltrate large OSNs like Facebook with swarms of fake accounts that are under their control [36]. This has led to many efforts from the research community to measure, understand, and combat these emerging threats. Here, we survey the available literature on OSN security related subjects.

6.2.1 Social Spam

There is a large body of prior work on measuring email spam [17, 97, 100, 195]. Since the emergence of OSNs, researchers have begun measuring social spam as well. Two studies have focused on measuring spam on Twitter. Grier et al. all find that 8% of URLs on Twitter are spam, and that the click-through rate of these URLs is
Chapter 6. Related Work

0.13%, which is significantly higher matriculation that email spam [78]. Thomas et al. analyze 1.1 million accounts that Twitter has banned and find that they generated 80 million spam tweets [170]. They identify widespread abuse of legitimate web services (such as URL shorteners) by attackers, and observe a burgeoning ecosystem of support services for Twitter spammers, including affiliate marketing programs. Both studies find extensive evidence of large-scale, well-orchestrated spam campaigns, proving that profit-driven, online criminals are responsible for the torrent of spam on Twitter.

Honeypots. Various techniques borrowed from email spam detection have been applied to OSN spam. Two studies leverage honeypot accounts on MySpace and Twitter, respectively, to trap spammers that attempt to friend them [107, 188]. Although these methods are somewhat successful, they rely on attackers actively friending the honeypot accounts to be effective. Unfortunately, friending and following are not necessarily prerequisites for spamming (e.g. spam can be directly targeted on Twitter using “@” messages). Thus, attackers can easily evade honeypot traps. Furthermore, our results from Renren indicate that unless social honeypots are engineered to appear popular, they will only be targeted by a small, naive subset of spammers.

Machine Learning. Four studies use Bayesian filters and SVMs to identify spammers on Twitter [29, 182, 198] and Facebook [164]. All these studies share common methodology and techniques: collect a large sample of spam from an OSN, and then train a classifier to identify it based on features like follow/following ratio, number of
retweets, and number of mentions. These techniques are fairly successful, producing detection rates upwards of 85%. However, as with honeypots, attackers can adapt to avoid these machine-learning based detectors, *e.g.* by using direct messages (“@”) and hashtags (“#”) to target users and trending topics, rather than following users.

Benevenuto *et al.* propose a supervised learning approach for detecting spammers in the user feedbacks of YouTube videos [30]. However, this study is constrained by the small sample size of only 829 users. Lastly, Markines *et al.* use machine learning to detect spam on the social bookmarking site BibSonomy [119]. This study leverages features that are unique to social bookmarking sites to isolate spam, such as tag frequency and popularity.

Rahman *et al.* built and deployed a Facebook application that identifies spam on user’s walls using a SVM [147]. Their MyPageKeeper application was installed by 12,000 Facebook users, of which 49% received spam over the four month measurement period. This application is noteworthy because of its large user base, and its focus on detecting wall spam in real-time. Speed is achieved by only using features that are embedded inside wall posts, rather than reaching out to crawl suspect URLs for additional data. However, the application’s primary detection mechanism relies on keywords to identify spam, which make it vulnerable to obfuscation (*e.g.* “v14gr4” instead of “viagra”).

230
Chapter 6. Related Work

Facebook’s own internal security system (known as the Facebook Immune System) is also based on machine learning technologies [162]. This paper provides an interesting glimpse into Facebook’s security architecture. However, it provides no concrete details about the characteristics of the threats they face, the scope of attacks, or the efficacy of their technology (e.g. false positive and negative rates).

In general, despite the large number of studies that have focused on detecting social spam, this area is still in its infancy. Existing social spam detection systems have exhausted the limits of machine learning to fight social spam, and yet OSNs are still rife with spam. Next generation tools are needed that leverage deeper, more nuanced data, in order to bolster robustness against evolving attacker strategies.

6.2.2 Social Sybils

Sybil attacks [60] are one of the most prevalent and practical attacks against distributed systems. In this attack, a user creates multiple fake identities, known as Sybils, to unfairly increase their power and influence within a target system or community. Distributed systems are ill-equipped to defend against this attack, since determining a tight mapping between real users and online identities is an open problem. To date, researchers have demonstrated the efficacy of Sybil attacks against P2P systems [113], anonymous communication networks [27], and sensor networks [138].
As we show in Chapters 4 and 5, OSNs are also under attack from Sybils. In addition to Facebook and Renren, Sybils have been identified on Twitter [78, 170], as well as within social games [136]. These studies highlight the difficulty of distinguishing real users from Sybils. For example, some Sybils on Twitter remix and rebroadcast the content of legitimate users, creating the illusion that the Sybil accounts are real.

**Sybil Community Detectors.** To address the problem of Sybils on OSNs, researchers have developed graph structure-based algorithms for detecting Sybils. The common assumption of all these algorithms is that Sybils have difficulty forming edges to honest nodes. The rarity of these *attack edges* causes the Sybil nodes to be outliers to the graph, *i.e.* there is a small quotient-cut separating the Sybil nodes from the honest region of the graph. SybilGuard [200], SybilLimit [199], and SybilInfer [57] all rely on specially constructed random walks to identify these Sybils. SybilLimit is the most advanced system of the three, incorrectly admitting the fewest Sybils per attack edge.

The three algorithms mentioned above are *decentralized*, meaning that any node in the social graph can run the algorithm to determine if any other node is a Sybil. SybilRank [42] uses similar techniques, but is designed to be run in a *centralized* manner, *e.g.* by the OSN provider using MapReduce. Despite their differences, it has been shown that all four these systems can be generalized using a ranked community detection approach [181]. SybilRank is notable because it is the only Sybil community detector
Chapter 6. Related Work

that has been successfully evaluated and deployed on a real social graph against real attackers (the other studies all base their results on simulated attacks).

Credit Networks. The previously mentioned algorithms are designed to detect Sybils on OSNs, presumably so they can be banned. Other algorithms are designed to make social networks resistant to Sybil attacks (or, equivalently, tolerant to Sybil attacks). SumUp [174], GateKeeper [173], and Bazaar [140] all use max-flow techniques to limit the influence of Sybils attempting to attack social graph-based applications. SumUp focuses on collecting votes from honest users in a polling application, while Bazaar is designed to exclude fraudulent accounts from online marketplaces. SumUp assumes that all graph edges have a weight of one (e.g. one user, one vote), but GateKeeper and Bazaar expand upon this by assigning custom weights to edges (using tickets in the former case, and positive transaction feedback in the latter case). Viswanath et al. demonstrate that all these systems can be modeled using credit networks, and build the Canal system to implement this concept [180].

In summary, there are seven papers that all focus on detecting social Sybils using graph structural metrics. This particular avenue of Sybil detection has been thoroughly explored, and the success of these schemes is debatable. SybilRank has been shown to work in practice; however, as we show in Chapter 5, Sybils can easily adapt to avoid these schemes. We urge researchers interested in working on social Sybil detection to explore alternative strategies, possibly leveraging behavioral and temporal data, in
Chapter 6. Related Work

order to push the boundaries in this area. We have conducted novel research on using
crowdsourced labor to detect social Sybils, the details of which are in Section 5.6.

6.2.3 Malicious Crowdsourcing

Crowdsourcing is the process of outsourcing work to a large group of unrelated people. Crowdsourcing work online has been popularized by websites like Wikipedia and Amazon’s Mechanical Turk. On Mechanical Turk (as well as on other similar sites), users submit small jobs that they need completed, such as transcribing audio, tagging images, or completing surveys, along with a small monetary reward. Anyone may sign up for the service, complete jobs, and earn the rewards. Crowdsourcing is advantageous because it leverages human beings to solve problems that computers are ill-suited for, in a manner that is highly scalable and elastic.

Although crowdsourcing is being applied to many useful tasks, malicious users are also leveraging crowdsourcing to attack major websites. For example, P. G. Ipeirotis discovered that 40.92% of jobs on Mechanical Turk were related to spamming and search engine optimization (SEO) in December 2010 [88]. This trend is very troubling, because attacks driven by real people are able to circumvent security systems designed to stop automated attacks (e.g. CAPTCHAs).

The study of malicious crowdsourcing is of vital importance, as these marketplaces are the direst threats currently facing OSNs. However, this research area is still in its
infancy: only three studies have quantified malicious crowdsourcing marketplaces to date. Motoyama et al. discovered that over 30% of the jobs listed on Freelancer are related to SEO, registering fake accounts, CAPTCHA solving, writing fake reviews, astroturfing, liking/following OSN accounts, and spamming [131]. The authors purchased fake email accounts from sellers on Freelancer, and determined that these sellers are using blacklisted IPs to create stockpiles of fake accounts, suggesting that they are involved with traditional botnet and spam schemes as well. Finally, the study notes that the demand for OSN related services, like fake accounts and likes, is currently less than the demand for SEO and fake email accounts, but growing more rapidly. This suggests that the problem of crowdworkers targeting OSNs will get worse as time goes on.

We conducted the second study on malicious crowdsourcing, the details of which are in Section 6.4.8. The third study by Stringhini et al. focuses on websites selling fake Twitter followers [163]. Buying services on these marketplaces is cheap: $20 per 1,000 followers (although high-quality followers often command a premium), and $10 for 1,000 tweets/retweets. In contrast to the pay-for-work model observed on crowdsourcing sites, the Twitter account market functions by duping normal users. In exchange for a small number of followers, users agree to hand over their Twitter account credentials to the marketplace. This pool of accounts that are given to the marketplace are then sold to paying customers who want to buy “premium” services, e.g. large numbers of followers and tweets.
Other studies are beginning to emerge that are observing the effects of malicious crowdsourcing. Thomas et al. observe that purchased, fake accounts are being used to stifle political speech on Twitter [169]. Their study is based on massive spam attacks that were directed at political protesters in Russia. They estimate that close to 1 million Twitter Sybils belong to this group of attackers alone.

6.3 OSN Privacy

The rapid rise in OSN popularity has led to commensurate levels of alarm about user privacy. Several studies have surveyed the overall privacy footprint of OSN users [64, 79, 101], while others have pinpointed specific privacy leaks in web- and mobile-based OSNs [102, 103]. Liu et al. conducted a user study and discovered that the majority of Facebook users believe that their privacy settings adequately protect their personal photos, when in fact they do not [115].

All of these studies highlight the fact that OSN providers do not protect users’ privacy by default (in many cases, this would be against their primary business model). Thus, researchers have stepped in to provide tools and technologies for users looking to enhance their privacy on OSNs.
Chapter 6. Related Work

6.3.1 Privacy-Enhancing Tools

**Browser Extensions.** Researchers have designed a variety of tools that are meant to enhance users’ privacy on existing OSNs. NOYB is a browser extension that scrambles a user’s Facebook profile by randomly swapping fields (e.g. email address, phone number, etc.) with other NOYB users [81]. The index revealing the true location of a user’s profile fields is kept secret from Facebook, and shared with friends out-of-band.

FaceCloak provides a similar level of protection by storing fake profile information on a user’s Facebook profile, while storing the true information in an encrypted format on a third-party website [116]. A browser extension is used to substitute fake information for real information on demand, while users are browsing Facebook.

Safebutton is a browser extension that prevents OSNs from tracking users’ browsing histories via social plug-ins (e.g. Facebook’s “like” button) [99]. The extension works by caching friends’ likes on the user’s hard drive. When the user visits a page with a like button, the local cache is queried to determine if their friends like that page (which is then displayed next to the like button). This obviates the need for the user’s browser to contact the OSN on each page visit, thus protecting the privacy of their browsing history.

Although all of these browser extensions are clever hacks, none have seen widespread adoption. Furthermore, in a broad sense, none address the fundamental privacy prob-
Chapter 6. Related Work

lems associated with monolithic OSNs. As long as centralized providers control the social graph and can observe all user interactions, there is little that can be done to safeguard users’ privacy from the whims of these providers.

Social Applications. Two studies focus on protecting users’ privacy against social applications built by third-parties, rather than against the OSN provider itself. Singh et al. propose an application sandbox called xBook that restricts what personal information applications may access, and what third-party websites the application may contact [160]. Applications are broken down into modules, each of which has its own set of access policies. This design enables applications to segregate modules that manipulate sensitive information, with the sandbox insuring that information only flows into that module, and never out of it.

Felt et al. propose a scheme called privacy-by-proxy, where third-party applications are not allowed to access sensitive information at all [68]. Instead, the application returns snippets of markup embedded with special tags that are replaced by the OSN. For example, rather than inserting a friend’s birthday directly into the output HTML, an application would embed a \(<\ uvalid = \"[Sid]\"\ field = \"birthday\"/ >\), where \$id is the anonymized user ID of the friend. Although Facebook already supports many tags like this, the authors introduce new tags and methods in order to fully decouple social applications from personal information.
There are three drawbacks to these social application security systems. First, they require OSN providers to modify their development platform and APIs, which they may be unwilling or unable to do. Second, they necessitate that third-party developers rewrite their applications, which they may resist. Finally, enhancing user privacy comes at the expense of giving personal information to third-parties, which may hinder their ability to monetize and fund the development and hosting of their social applications.

Privacy Wizards. Rather than attempting to obfuscate information from OSN providers and applications, Fang et al. propose the use of machine-learning to help users more accurately implement their desired OSN privacy settings [66]. Their Facebook application acts as a privacy wizard, using a small amount of explicit user input to extrapolate the user’s overall privacy preferences via machine learning. The authors conduct a user study and show that if users label 25 out 200 friends, the wizard can infer overall privacy settings with 90% accuracy. However, the user study only included 45 participants, so the generality of these performance claims is questionable.

Stopping Malicious Crawlers. Many researchers, including us, have crawled OSNs for data. However, the ease with which this can be done is disturbing. If our team can gather 10 million Facebook profiles in a matter of weeks, who knows how much personal information a malicious attacker could crawl.
Chapter 6. Related Work

Three papers specifically address the issue of stopping malicious OSN crawlers, the first of which was conducted by us. The details of our SpikeStrip system can be found in Section 6.4.4.

Genie uses graph distance-based rate limiting to stop malicious crawlers [130]. The intuition behind this approach is that users tend to browse profiles in their local area of the graph (i.e. where their friends are), and that browsing far away from oneself is suspicious. Thus, each account is given a browsing budget that exhausts faster the farther the user gets from their own profile. Although this technique is very innovative, the computational overhead of calculating graph distances after each page hit make this technique unlikely to succeed in practical settings.

PubCrawl relies on heuristics metrics and time series analysis to identify anomalous browsing patterns [89]. The key insight of this system is that browsing by normal users is bursty in time, while crawlers exhibit more uniform traffic levels. Using this approach, distributed crawlers can be detected by looking for correlated browsing patterns from different source IPs. This work is notable because it has been actively deployed by a large European OSN.

6.3.2 Distributed OSNs

Numerous researchers have addressed the OSN privacy problem by replacing monolithic, centralized OSN providers with more secure, distributed alternatives. This space
has become extremely crowded: there are at least two papers that build distributed, peer-to-peer (P2P) OSNs, and another seven that advocate using cloud-based storage to federate social content. However, none of these technologies has been adopted by users.

The unfortunate truth about distributed OSNs is that solid research and innovative technology are simply not enough to drive real-world adoption. The vast majority of users are apathetic or simply unaware about the privacy risks of OSNs. This is especially true when gaining additional privacy comes at the cost of running complicated desktop software or paying for cloud storage. Simply put, the factor holding back privacy preserving OSNs is not technological but cultural, and no amount of research is going to fix this.

**P2P OSNs.** Two papers propose building P2P OSNs based on distributed hash tables (DHTs). PeerSoN uses the OpenDHT infrastructure to store encrypted personal data, wall messages, and photos [38]. However, this paper mostly describes practical experiences building software on OpenDHT, rather than focusing on security and privacy issues. SafeBook is a more in-depth attempt to build a feasible P2P OSN [56]. SafeBook is organized into two layers: a DHT, which is used for message routing, and matryoshkas that surround each user. Matryoshkas are constructed from a user’s trusted personal friends, and provide persistent storage and presence for users when they are
Chapter 6. Related Work

offline. Group-based encryption is used to securely store and distribute private content among groups of friends.

Cloud-based OSNs. There are seven papers that propose to use cloud-based storage to create decentralized OSNs. Cloud storage is an attractive alternative to P2P designs because it sidesteps the data availability and churn issues that plague DHTs. Instead, data in the cloud is persistent and highly available, simplifying the design of higher-level services.

Vis-a-Vis uses cloud-hosted virtual machines to store and serve social data, with a focus on secure sharing of location data [158]. Each virtual individual server (VIS) stores a user’s personal data, and the VISs self-organize into hierarchical groups based on the physical location of users (i.e. countries, cities, blocks, etc.). Although Vis-a-Vis assumes that the cloud hosting provider is trustworthy (and is therefore able to view user data stored in virtual machines), the overall architecture prevents information disclosure to non-friend, third-parties.

In contrast to Vis-a-Vis, Persona assumes that the cloud storage provider is untrustworthy [22]. In Persona, users tag their friends with attributes (e.g. co-worker, neighbor, family) in order to form groups. Attribute based encryption (ABE) is then used to securely distribution content to different groups. Although data is stored in the cloud for retrieval by friends, the cloud provider does not have access to the ABE keys, and thus cannot view user’s data.
Chapter 6. Related Work

Anderson et al. propose a simple client/server model for building OSNs that uses encryption to protect users’ social data and friendship linkages [18]. This work also assumes that the server is untrustworthy, and focuses on hiding the user’s data on disk using random padding.

Frientegrity also uses encryption to secure social data stored on third-party providers, but goes a step further to prevent equivocation by those providers [67]. In a system like Vis-a-Vis, the cloud provider knows what data belongs to what user, even if they cannot read the data. This creates an opportunity for mischief: a provider can present different views of the system to different users, i.e. when a content author checks, all their data appears available, but when friends look, data is missing. Frientegrity enforces fork* consistency, which is a property where updates to objects (e.g. a news feed) are stored in a strictly linearizable order. Users collaborate to ensure that cloud providers are meeting this requirement.

In contrast to Vis-a-Vis and Frientegrity, which are concerned with protecting social content, Lockr proposes a cryptographic protocol that is designed to protect the privacy of social relationships [172]. In Lockr, users are free to store their data with any provider (centralized or decentralized) in encrypted or open formats. Access to this data is controlled by social access control lists that enumerate the friend groups (e.g. co-worker, neighbor, family) that are permitted to view the data. Users issue social attestations to their friends certifying what groups they belong to, thus enabling them
to access data. Thus, service providers are not informed about individual social links, only friend groups. The authors build a prototype implementation of Lockr that use Facebook for identity and friend management, and Flickr and BitTorrent for data storage. Although this demonstration is convoluted, it does show that Lockr’s techniques can be applied to many different systems.

Contrail is a mobile OSN that uses client side filters and a store-and-forward service in the cloud [165]. Although Contrail encrypts data in the cloud to protect user’s privacy, the main focus of this work is on making the system feasible for power and connectivity strained mobile devices. Contrail uses a publish/subscribe system to forward status updates between users. However, to save power, users push filters to their friends’ devices. These filters evaluate each status update to determine if the user is interested in the content. If so, the message is forwarded to the user via the cloud. If not, then the update is never sent, thus saving power on both users mobile devices.

The final cloud-based OSN was produced by our research group. Details on our Polaris system are available in Section 6.4.5.

6.3.3 De-anonymization and Privacy Preserving Models

In the prior sections, we survey the literature on privacy issues associated with live, deployed OSNs. However, static datasets, such as those used by OSN researchers, are also tainted by privacy concerns. OSN providers are reluctant to share data with re-
searchers even if it has been anonymized because numerous studies have shown that graph datasets can be *de-anonymized* [20, 134, 135, 193, 205]. The threat of de-anonymization also prevents researchers from sharing social graph data with each other, hindering the progress and openness of science.

Developing strong techniques for preserving the anonymity of graph data are of vital importance to researchers working on complex networks. Without these techniques, researchers may forever be blocked from using and sharing commercial datasets. Although researchers have begun to address this need, existing techniques are fundamentally inadequate: they either ruin the statistical fidelity of graphs, or are vulnerable to novel de-anonymization strategies. Significantly more work needs to be done in this area before these anonymization techniques will be practically useful.

**k-anonymity.** Numerous researchers have leveraged the *k*-anonymity model [69, 167] to anonymize graph datasets. In a *k*-anonymous graph, no individual can be identified with probability greater than $1/k$. This is accomplished by altering the edges of the graph to ensure that each user is surrounded by a group of nodes that are isomorphic to at least $k-1$ other groups of nodes. Unfortunately, optimal *k*-anonymity is NP-hard, meaning that heuristics must be used in order to practically implement the technique [123].

Zhou *et al.* published the first work applying *k*-anonymity to the problem of anonymizing social graph datasets [207]. The focus of their work is on efficiently calculating
Chapter 6. Related Work

$k$-anonymous groups using a depth-first encoding technique. This is necessary because the alternative is to use graph isomorphism, which is computationally complex. Liu et al. also examine $k$-anonymous graphs with a focus on securing the privacy of the graph’s degree distribution [114]. However, these approaches are only robust against attackers with 1-hop neighborhood information. Attackers with more information can still de-anonymize the graph.

Hay et al. examine the threats to privacy posed by attackers with different levels of knowledge about graph structure [83]. The three attack models are complete knowledge, partial knowledge, and hub knowledge, i.e. the local neighborhoods around super-nodes. The authors find that denser graphs are more resistant to attack, and that hubs, although themselves distinctive, are not useful for de-anonymizing their neighbors. The authors propose a graph anonymization algorithm that groups nodes together into clusters connected by super-edges. This technique is robust against attackers with a great deal of knowledge about the graph structure. Unfortunately, the aggregated, anonymized graph loses all local subgraph structure.

Bhagat et al. propose a similar a technique similar to Hay et al. but focused on anonymizing interaction graphs [32]. The authors divide interactions into classes (clusters, essentially) and only release graphs of classes, rather than individual interactions.

Zou et al. propose a technique called $k$-automorphism that is similar to $k$-anonymity but is also robust against knowledgeable attackers [209]. The benefit of this method is
Chapter 6. Related Work

that it preserves local subgraph structure. Unfortunately, \( k \)-automorphism still adds significant noise to graphs, harming their fidelity.

Puttaswamy et al. consider a social intersection attack, where attackers in a social graph collude to identify the source of shared data [146]. In this context, users share data via links in the social graph, but the source of the data is not explicitly stated. However, if two attackers with only one mutual friend both observe the same shared data, then they can infer that it originated from the mutual friend. The authors develop the StarClique system to add latent edges to the graph that grant \( k \)-anonymity to users and prevent intersection attacks.

Das et al. use a linear programming-based approach to \( k \)-anonymize the weights of social graphs [58]. In their attack model, graph structure is publicly available, but edge weights are private (e.g. trust values between friends). The authors use linear programming to anonymize edge weights because many important graph metrics can be expressed as linear properties (e.g. shortest paths, max flow, etc.). Benchmarks demonstrate that Anónimos can anonymize graphs with millions of edges in only a few minutes using an open source LP solver.

In general, \( k \)-anonymity techniques have two major drawbacks: first, they tend to be “attack-specific,” i.e. they are designed to prevent de-anonymization by known methods. However, novel methods may be able to break the provided anonymity. Second, these techniques introduce significant changes to the graph, altering it in statistically
significant ways. Thus, results gleaned from the anonymized graphs may not hold on the original graphs.

**Differential Privacy.** Other researchers have investigated aspects of differentially private graphs. Differential privacy is a technique designed to provide provable anonymity guarantees for statistical databases [62, 63]. In this model, users are allowed to execute a query against a database $D$ and receive a result $Q$. By carefully adding noise to $Q$, the user is unable to tell if the result came from $D$ or $D'$, which is a database that differs from $D$ by exactly one element. Intuitively, this process prevents a malicious user from repeatedly modifying $D$ and re-querying in order to use the differences in query output to violate the privacy of individual users.

Two studies focus on generating differentially private degree distributions, both written by Hay et al. The first study focuses on developing a fast algorithm for generating differentially private degree distributions that is also low-noise [82]. Both of these goals are important: the algorithm needs to be efficient so as not to overburden the database server. Similarly, the query results should include as little noise as possible (without compromising anonymity) in order to preserve the fidelity of data. The second paper expands on the initial work by adding support for histogram queries that are consistent with overall summation queries [84].
Chapter 6. Related Work

Unfortunately, the techniques presented by Hay et al. are not sufficient to produce differentially private graphs in their entirety. We have conducted research that aims for this goal, the details of which are in Section 6.4.7.

Proserpio et al. have also developed a workflow for synthesizing differentially private graphs [141]. In their work, a secret, private graph is queried for key distributions, such as degree distribution, JDD, and clustering coefficient. Each query result is made differentially private using non-uniform noise, in order to maximize fidelity. Finally, a random graph is fitted to the differentially private query results using a Markov-Chain Monte Carlo search. This process ultimately produces a graph that closely mimics the characteristics of the original, while preserving differential privacy.

Although differential privacy has been shown to work well in many contexts, applying it to graphs is still an active research area with a great deal of promise. Researchers have only begun to scratch the surface of generating differentially private graphs. There is still a great deal of room for investigating new query functions with low sensitivity that can hopefully be used to improve the fidelity of differentially private graphs.

6.4 Other Works by the Author

In this section, we briefly summarize other published works from the author that have not been mentioned elsewhere in the manuscript.
Chapter 6. Related Work

6.4.1 Social Marketplaces

One of the most useful aspects of OSNs is the trust that exists between social friends. Numerous proposals have been made that leverage this trust to improve existing systems. In this study, we examine Overstock Auctions, a online auction site (similar to Ebay) that includes a social layer [166]. In particular, we are interested in researching whether transactions between social friends receive higher satisfaction ratings than transactions between strangers. This would indicate that social networks can be used to prevent fraud in online auction systems.

Overstock Auctions maintains two separate but parallel networks: one connects all buyers and sellers, the other connects all social friends. Overstock was kind enough to give us anonymized copies of both graphs. The business network includes 400,000 users, while the social network includes 85,000 users (with 52,000 users appearing in both networks). For users that appear in both networks, we observe that their business connections and social friends show little overlap, indicating that most users keep their business and personal contacts separate.

Next, we examine the correlation between path length and transaction success. Overstock Auctions users are encouraged to rate transactions after they are completed on a -2 to +2 scale, with +2 being the best. We observe that almost all transactions on

---

1This work was published as “Do Social Networks Improve e-Commerce: a Study on Social Marketplaces” at WOSN 2008 [166].
the social graph are successful, even if the buyer and seller are up-to 6 hops away from each other. In contrast, transaction success falls precipitously on the business graph when buyers and sellers are >3 hops apart. This is the most important finding of our study: because users tend to form social connections with people they know and trust, fraudulent users are almost entirely excluded from the social graph. Thus, the transitive trust between social users enables transactions to be highly successful, even when many hops separate the buyer and seller. In contrast, the business network is easily infiltrated by malicious users because there is no transitive trust.

6.4.2 Measurement-calibrated Graph Models

De-anonymization attacks currently prevent private companies and researchers from sharing sensitive social graph datasets. Although researchers have proposed techniques to anonymize social graphs, these techniques suffer for a variety of reasons. Some techniques are only robust against known attacks, while others offer strong privacy but destroy statistical fidelity in the process. Other techniques are only designed to protect specific graph metrics, rather than entire graphs.

In this work, we propose using measurement-calibrated graph models as a novel form of graph privacy protection [149]. The high level idea is that instead of releasing real (i.e. sensitive) graph data, companies can instead release a graph model and a

\[\text{measurement-calibrated graph models}\]

\[\text{WWW 2010}\] \[\text{[149]}\]
set of parameters. The given model and parameters combine to generate graphs that are statistically similar to the original graph, giving researchers the ability to synthesize an infinite number of test graphs confident in the knowledge that they are highly representative.

Achieving this vision requires overcoming several challenges. First, we have to develop a methodology for fitting graph models from the literature to real data. This challenge immediately raises a second challenge: how do we compare graphs in a scalable way, i.e. without resorting to isomorphism. We address these challenges using several tools. First, we use maximum likelihood estimation to compare the distributions for two graphs, in order to assess their similarity. We use the $dK$-series [118] of each graph as the distribution for comparison. Using these tools, we can conduct a course search of the parameter space for a given graph model to see what parameter values produce synthetic graphs that are similar to a target real graph. We follow the course search with a fine grained search in order to zero in on the best parameter values.

We evaluate the ability of six well known graph models from the literature to fit real world graphs taken from Facebook. These models are Barabasi-Albert, Forest Fire, Random Walk, Nearest Neighbor, Kronecker Graphs, and $dK$-graphs. Using our fitting methodology, we use each model to produce synthetic graphs that are like our Facebook graphs, and then compare key graph metrics to see if the synthetic graphs are high fidelity. We find that $dK$ and Nearest Neighbor are both able to produce synthetic
graphs that are statistically similar to real graphs. Thus, either one of these models would be an excellent choice for producing privacy preserving, synthetic graphs.

In addition to developing a novel methodology for producing privacy preserving, synthetic graphs, our work is also useful as an empirical evaluation of existing graph models. To our knowledge, this is the first and only study that quantifies the ability of existing models to produce representative models of today’s social graphs.

6.4.3 Orion: Fast Shortest Path Estimation on Large Graphs

OSNs are often limited in the services they can provide to users due to the computational overhead of running complex queries on massive social graphs. For example, many potentially useful features (e.g. proximity-based social search) rely on being able to compute shortest paths between nodes. However, it is known that graph distance algorithms are computationally challenging, especially on graphs with hundreds of millions of nodes.

To mitigate the computational overhead associated with graph distance queries, we developed a system called Orion that is able to compute highly accurate graph distance estimates in only a few microseconds [203]. The key insight behind Orion is that social graphs can be embedded into a multi-dimensional, geometric space. This reduces the

\[^3\text{This work was published as “Orion: Shortest Path Estimation for Large Social Graphs” at WOSN 2010 [203].}\]
problem of computing graph distance (which is slow, $O(n \log n + m)$ for Dijkstra) to computing geometric distance (which is near instantaneous, $O(1)$).

The primary challenge to successful graph embedding is selecting landmark nodes to bootstrap the process. The landmarks determine the relative coordinates of all other nodes, and thus must be chosen carefully to maximize accuracy. Similarly, all-pairs shortest paths must be computed between the landmarks, which can be slow if too many landmarks are chosen. Fortunately, experimental results from Orion reveal that simply selecting 30 high-degree nodes as landmarks provides excellent performance with low error rates.

Comparisons between Orion path length estimates and true path lengths on Facebook graphs reveal that Orion produces very accurate results. The relative error for 90% of nodes is less than 0.4, meaning that the vast majority of estimated path lengths are within 1-hop of the true value. Subsequent research by our team has further improved the accuracy of the system by embedding graphs into hyperbolic space [204]. Zynga, Renren, and Google have all deployed versions of our hyperbolic graph system (called Rigel).
OSNs have become huge targets for malicious users looking to harvest massive amounts of personal information from unsuspecting users. As researchers have repeatedly demonstrated, it is simple for automated crawlers to collect millions of users worth of information in a relatively short amount of time.

In order to combat the threat posed by malicious OSN crawlers, we propose the SpikeStrip system [191]. SpikeStrip is an Apache web server module that adds two features to a website. First, hyperlinks on served pages are encrypted using the current user’s session key and a server-side secret key. Second, incoming requests are rate limited using the user’s session key.

These mechanisms enable SpikeStrip to perform very accurate, fine grained rate limiting. By using session keys to track requests instead of IP addresses, SpikeStrip can differentiate between users hidden behind proxies and NATs. Additionally, by using session keys to encrypt hyperlinks, SpikeStrip uniquely binds every served page to specific user. This prevents crawlers from avoiding the rate limits. For example, if a crawler tries to avoid the limit by changing their session key, their entire queue of URLs is invalidated, since the URLs will no longer decrypt. This technique also

---

4This work was published as “Don’t Tread on Me: Moderating Access to OSN Data with SpikeStrip” at WOSN 2010 [191].
Chapter 6. Related Work

hinders distributed crawlers, since machines can no longer trade URLs (again, because of the unique binding).

We evaluate our SpikeStrip prototype on a demo website powered by real Facebook data and show that it is both highly effective at rate limiting crawlers, while also being low overhead on the web server. SpikeStrip uses counting bloom filters to minimize its footprint in RAM while still maintaining fast, accurate hit counts.

6.4.5 The Polaris Mobile Social Network

Researchers have developed many designs for privacy preserving, decentralized OSNs. However, none of these proposals have been embraced by users. This problem stems from the extreme point in the design space that current decentralized OSNs inhabit. One end of the design space is occupied by simple, free, centralized OSNs that offer no privacy protection (e.g. Facebook). On the other extreme end are current research proposal, which offer very tight privacy guarantees, but place all costs onto users. Unfortunately, users seem unwilling or unable to shoulder these costs, be they monetary (i.e. paying to host data in the cloud) or time-based (i.e. maintaining a home server to run a P2P OSN).

We developed the Polaris system as a practical middle-ground between centralized OSNs and secure but costly distributed OSNs [192]. In Polaris, user’s sensitive personal

\footnote{This work was published as “Privacy, Availability and Economics in the Polaris Mobile Social Network” at HotMobile 2011 [192].}
information (e.g. phone number, email address, etc.) is stored and served from their mobile device (e.g. a smartphone). However, content that is large or frequently accessed (e.g. status updates, photos, and video clips) are stored by cloud hosting providers. Data stored in the cloud is not encrypted so that providers can mine the data and profit from targeted advertising. This enables cloud providers to offer high availability, persistent storage to users for free, which is the model that users seem to prefer. However, sensitive information is only stored on the user’s personal device, and is shared directly with friends. Thus, service providers are never privy to sensitive information.

Polaris enables each user to choose whichever service providers they wish. This fosters an ecosystem of competing providers with different business models. For example, many users may prefer providers that offer free service but serve ads. More privacy conscious users may prefer providers that offer encrypted storage for a fee. Polaris’ social access controls enable all of these competing providers to seamlessly coexist. Each user can access their friend’s data even if those friends have chosen different providers than the user.

In Polaris, users are encouraged to use different providers for different types of data (e.g. status updates, photos, videos), as this further enhances the user’s privacy. Polaris also includes APIs that enable seamless data migration between providers. This encourages providers to be well behaved, since they disaffected users easily migrate to competing services.
6.4.6 Datacenter Oriented Network Transport Protocols

6 Many applications that run on datacenter networks are deadline driven. For example, the distributed applications that power Internet search services include hard deadlines that limit how long the application can take to generate search results and return them to clients. These deadlines are based on research showing that long page load times alienate users, who then migrate to competing services that are more responsive.

Unfortunately, datacenter networks are not designed to facilitate deadline driven applications. Consider a server $A$ that needs to deliver 4 packets of data to server $B$ before a deadline expires. If one packet from $A$ is dropped, or arrives past the deadline, then the entire response is useless to $B$. Even worse, in high utilization scenarios, the network resources devoted to the three useless packets could potentially have been used by a different flow to meet its own deadline.

Currently, engineers designing deadline driven applications have to go to great lengths in order to mitigate network level issues that impede meeting deadline. For example, Facebook’s datacenters run a totally custom transport protocol instead of TCP. Similarly, some applications incorporate jitter into their packet sending code to try and avoid TCP-incast congestion. Finally, Google’s search architecture limits responses

\footnote{This work was published as “Better Never than Late: Meeting Deadlines in Datacenter Networks” at SIGCOMM 2011 [189].}
from distributed search servers to a single, highly compressed packet, so as to avoid network congestion issues.

In our work, we propose a novel transport protocol, called $D^3$, that uses application deadline information to more efficiently schedule packets in the network [189]. Our protocol is a clean-slate design that uses router-augmented explicit congestion notifications in order to maintain high network utilization while minimizing packet drops. Clients with deadline driven flows calculate the send rate they need to meet their deadline, and include the desired rate in their packet headers. $D^3$ routers use this information to make bandwidth reservations for clients using a low overhead, fast, soft-state approach. Soft-state in the routers is critical, because bandwidth reservations for deadline flows needs to change very quickly; there is not enough time to use heavyweight reservation protocols like DiffServ and IntServ. Experimental results from a test bed implementation of $D^3$ demonstrate that the protocol is able to double the peak load that the network is able to support without missing any deadlines (compared to TCP).

### 6.4.7 Differential Privacy for Graphs

Several researchers have attempted to use the concept of differential privacy [62, 63] to protect sensitive social graph data. However, these existing works have only applied differential privacy to statistical distributions derived from a graph, *e.g.* the

---

7 This work was published as “Sharing Graphs using Differentially Private Graph Models” at IMC 2011 [150].
degree distribution. We propose a new system, called Pygmalion, that is the first to produce differentially private graphs in their entirety [150].

Differential privacy works by adding carefully calibrated noise to the output of a query function. Pygmalion uses the $dK$-2 series as the query function: starting from a real graph, produce the $dK$-2 series of that graph, add noise to the series, then use the $dK$ generator to generate a differentially private graph. However, analysis of the $dK$-2 series indicates that it is a very sensitive query function. This means that noise injection must be carefully controlled, otherwise the statistical fidelity of the generated graph is ruined.

One of the primary contributions of this work is the idea of using non-uniform noise to guarantee differential privacy while minimizing the overall amount of needed noise. In our context, the key insight is that the noise level is correlated with the maximum degree of nodes in the graph. Pygmalion sorts the $dK$ series by degree and clusters groups of nodes with similar degree together. Noise is then injected into each cluster, with low-degree clusters requiring less noise than high-degree clusters. We prove that this process provides the same level of differential privacy as uniform noise injection, while requiring significantly less noise overall (since low-degree clusters are far more numerous than high-degree clusters).
Chapter 6. Related Work

We evaluate Pygmalion on real graphs from Facebook, the web, and the AS graph, and demonstrate that Pygmalion can generate differentially private graphs that retain most of the statistical fidelity of the original graphs.

6.4.8 Crowdturfing

Although crowdsourcing websites like Amazon’s Mechanical Turk have proven to be a boon for legitimate businesses, there is increasing evidence that malicious individuals are leveraging crowdsourcing to attack social websites. We have dubbed this new phenomena crowdturfing, a portmanteau of astroturfing and crowdsourcing. We have scoured the web and located crowdturfing sites in the US and India. However, in this study we focus on measuring the two largest crowdturfing sites in China: Zhubajie (ZBJ) and Sandaha (SDH) [184].

Both ZBJ and SDH follow the Mechanical Turk model of operation. Customers post mini-jobs on each site and offer a small reward for their completion. In the case of crowdturfing sites, these job include things like signing up for fake accounts on OSNs, sending social spam, writing fake reviews, and posting shill questions and answers on social Q&A sites. Workers complete the jobs using legions of Sybil accounts that they control, and post links to their handiwork back to SDH and ZBJ. The customer can then verify that the work was completed by following the link. Workers that generate low

---

8 This work was published as “Serf and Turf: Crowdturfing for Fun and Profit” at WWW 2012 [184].
Chapter 6. Related Work

quality output do not get paid, so this verification process forces workers to generate high quality spam output.

All of the job information and responses from workers on ZBJ and SDH are public. We crawl all of this data, including historical records of jobs and payments that stretch back several years. Analysis of this data reveals that the number of jobs, and the amount of money being spent on them, has been growing exponentially for years on ZBJ and SDH. ZBJ is the older and larger of the two sites: overall $3 million has been spent on the site, with $600,000 being collected by the site itself as commission. In August 2011 alone, there were $1 million in jobs just on ZBJ, with 169,000 workers competing for the rewards.

One disturbing trend on ZBJ and SDH is that attacks against microblogging sites are the fastest growing job category. The increase in microblogging jobs parallels the massive growth of sites like Sina Weibo (a Twitter clone) in China. By tracing the hyperlinks embedded in reports from workers, we trace the origins of spam from crowd-workers out to honest users on Weibo. We observe that prolific crowdturfers can each control hundreds of Weibo accounts, and that campaigns that buy retweets tend to reach large audiences of normal users than campaigns that ask crowdturfers to generate original tweets.

Finally, we infiltrated ZBJ and bought several small-scale spam campaigns in order to measure the immediate response rates of workers, as well as clicks from users duped
by social spam. Our results show that crowdturfers work extremely quickly to spread spam and collect rewards, indicating that there is spare capacity amongst the worker population. We also find that campaigns targeting Weibo and QQ (an instant messaging service) are the most effective at acquiring clicks from real users. The cost-per-click of our spam campaigns is only slightly worse than traditional web display advertising, meaning that it is a cost effective way to advertise illegitimate content.

We are actively investigating the possibility of using crowdsourced labor to detect stealthy Sybils generated by crowdturfers. See Section 5.6 for more details.

6.4.9 Temporal Dynamics on Renren

9 One of the most under-studied areas of OSNs is graph temporal dynamics. This is due to the difficulty of acquiring social graph data with labeled edge creation times. In this study, we address this gap in the literature using data from the first 2 years of Renren’s existence, when it grew from a single user to 19 million users [202].

First, we examine the overall growth of the social graph over time, and how key metrics like average path length and assortativity change. We find that these metrics stabilize within 100 days of the OSN’s creation, even though the user base was growing exponentially at the time. Second, we examine what factors drive the creation of new edges over time. As expected, new users tend to create the vast majority of their new

---

9This work was published as “Multi-scale Dynamics in a Massive Online Social Network” at IMC 2012 [202].
Chapter 6. Related Work

does soon after they join the network. However, over time the overall fraction of new
dges created per day by new users drops, with edges between older, more mature users
dominating. This result is important, because most generative graph models assume
that new edge creation is driven by new nodes, when our results indicate that this is
clearly not the case for more mature graphs. We bolster this argument by demonstrating
that the strength of the popular preferential attachment (PA) model weakens over time
on our Renren data.

Next, we quantify the dynamics of communities on Renren over time. We use
the Louvain community detection algorithm, because it allows communities to be con-
structed in an iterative fashion. This enables us to isolate communities in early graph
snapshots, then iterate on them in later snapshots to see how they have changed. We
observe that most communities are short-lived, and that we can reliably predict when
they will merge into a larger community based on structural features. Interestingly, we
also note that users in strong communities tend to be more active in creating new edges,
especially with other users in their same community.

Finally, we analyze a network-level event that is unique to our dataset. In 2006, the
OSNs Xiaonei and 5Q merged to form Renren, giving us the opportunity to observe
how the 600,000 users in each OSN gradually connect with friends in the opposing
OSN. Immediately following the merge, close to 25% of user accounts on 5Q and 10%
on Xiaonei immediately become inactive, indicating that a significant number of users
Chapter 6. Related Work

had duplicate accounts that they abandoned. Initially, users in both networks preferred to create edges towards users on the other network, rather than to new users that had just joined Renren. However, as the number of new users in the system grows, they quickly come to dominate new edge creation. Despite the relatively low number of edges that “bridge the gap” between the two networks, we calculate the distance (in hops) from one network to the other rapidly drops: within just 50 days, the two networks can no longer be seen as separate, and are effectively one giant component.
Chapter 7

Conclusion

In this chapter, we conclude by discussing some of the high-level wisdom that the authors have gained while researching social networks. The aim is to comment on social network and security research in general, and not to focus on specific aspects of the previous chapters. First, we comment on the necessity of data-driven design for social systems, which is a running theme of our work (as exemplified in Sections 3.7 and 5.3). We touch on several issues inherent in contemporary OSN research, including the generality and reproducibility of results based on social graph datasets. Next, we examine challenges related to security research on social networks, and give high-level guidance for researchers working in this space. Finally, we conclude with a discussion of future research directions.
Chapter 7. Conclusion

7.1 Data-driven Design of Graph Algorithms and Applications

One of the running themes in this dissertation is demonstrating areas where real-world graph data challenges the assumptions of prior work. This is exemplified in Sections 3.7 and 5.3. In each case, researchers proposed a new graph algorithm or social application and evaluated it on a small number of real social graphs, or in some cases, synthetic graphs generated by models. However, as we demonstrate, the performance of these algorithms and applications is highly graph-dependent. By only using a select few graphs when performing evaluations, prior work often arrives at overly optimistic conclusions that are not representative of real-world performance.

Social Graph Properties Are Not Homogeneous. Part of the problem with existing evaluation methodologies is that there is a tendency among academics to generalize the concept of social graphs. In this mentality, all social graphs conform to a list of expected “buzzword” properties: power-law degree scaling, small world clustering, tightly clustered fringe, low average path lengths, etc. The assumption is that all real-world social graphs obey these properties, and are essentially homogeneous. The sizes and sources of real-world graphs may vary, but as long as they are “social graphs” then the differences between them are minor and can be safely ignored.
Chapter 7. Conclusion

Unfortunately, this folk wisdom ignores the fact that social graphs exhibit a diverse set of properties. Consider the five graphs examined in Section 3.2. Although all five are considered social graphs, they exhibit drastically different properties. For example, Flickr is dependent on a small core for its connectivity, but the other four a resilient to $k$-core analysis. On the other hand, Renren is the only graph that is disassortative.

Distributed Sybil detectors are one example of research that falls prey to the assumption that all social graphs exhibit homogeneous properties [57, 174, 199, 200]. In particular, all of these systems assume that social graphs are fast-mixing. However, this assumption is not universally true. Mohaisen et al. provide a very comprehensive look at how mixing time can vary wildly across two dozen social graphs [129]. Thus, the performance of these algorithms is highly graph dependent and does not generalize, as we highlight in Section 3.7.

Early on in our research, we also succumbed to the mentality that all social graphs are created equal. At one point, we endeavored to build a distributed database tailored to storing social graphs. The key challenge in building such a database is partitioning the graph across the machines in a way that minimizes edge cuts, since each edge cut is an inter-machine dependency that necessitates costly network traffic. Our insight to solve this challenge (or so we thought), was that if you remove the $k$-core of a social graph the remaining nodes partition into disjoint subgraphs. We based this assumption on results presented in [128]. Our plan was to leverage this property to distribute social graphs...
Chapter 7. Conclusion

in our database: place each disjoint subgraph on a separate machine, and replicate the small number of $k$-core nodes on each machine.

Unfortunately, we soon realized that not all social graphs partition nicely under $k$-core analysis. We clearly demonstrate this fact Section 3.2.7. The lesson we learned from this experience is to never assume that social graph properties generalize.

**Social Graph Models Do Not Generalize.** An over-reliance on social graph models also compounds the problem of evaluating graph algorithms and applications. In the absence of real social graph data, researchers often turn to synthetic graphs generated by these models to evaluate their work. However, models are only designed to capture specific graph properties like power-law degree scaling, short average path lengths, etc. No single model captures all of the important properties of social graphs, nor can it replicate the diverse breadth of values these properties can exhibit. Thus, designing and evaluating graph algorithms and applications based on synthetic graphs leads to unrealistic results.

One example of a model that has become over generalized is the forest fire model [112]. The forest fire model is designed to produce graphs with power-law degree scaling, high levels of density, and small graph diameters. The model achieves these aims, and is shown to generate graphs that approximate the properties of citation co-authorship, email, and Autonomous System (AS) graphs. The authors conclude the paper by stat-
Chapter 7. Conclusion

ing that forest fire can forecast the properties of future social networks and be used as a social graph generator.

Unfortunately, it turns out that this is not the case. In our paper “Measurement-calibrated Graph Models for Social Network Experiments,” we show that forest fire does a poor job of capturing the properties of large OSN graphs like Facebook and Renren [149]. This failure is not the fault of forest fire’s authors: the model was published in 2005, well before Facebook and Renren rose to prominence. However, forest fire is extremely well cited, and researchers continue to use graphs generated by the model as the basis for evaluating graph algorithms and applications targeted at contemporary OSNs. Simply put, using synthetic graphs as the sole benchmark for evaluating graph algorithms and applications does not lead to realistic, representative results.

Social Graph Taxonomies to the Rescue? The process of evaluating graph algorithms and applications would be much simpler if there existed a classification system, or taxonomy, for social graphs. In theory, researchers could choose a representative, benchmark set of graphs for evaluation that covered the entire space of observed social graph characteristics. Unfortunately, this goal is impossible to achieve: there is no all-encompassing super-metric to quantify graphs. Instead, each dimension in a taxonomy would correspond to a different graph metric, and new metrics are proposed in the literature constantly. Similarly, many metrics are not independent, and the dependencies
between them are unknown or unexplored. In light of these hurdles, the feasibility of a social graph taxonomy quickly becomes nil.

**No Simple Solution for Researchers.** In the absence of a social graph taxonomy or rigorous analytical methods, the only option left for researchers proposing new graph algorithms and social applications is to evaluate their work on as many graphs as possible. This is the only reasonable way for researchers to quantify the true, real-world performance of new algorithms. Although the additional testing does necessitate more of a time investment, this evaluation approach is methodologically straightforward. The growing number of large scale, publicly available graph datasets also helps to facilitate this additional testing.

Integrating more rigorous evaluation procedures into graph algorithm work is going to require some adjustments in the mentality of the social graph community at large. On one hand, reviewers tasked with appraising new work on social graph algorithms and applications need to start holding work to a higher level of standards. We know definitively that all social graphs are not created equal, and any work claiming that their algorithms are universally applicable and performant is likely to be overstating its claims.

On the other hand, we as a community must begin tempering our expectations about the performance and generality of new algorithms and applications. Not every algorithm is going to have spectacular performance on every social graph: a paper that
Chapter 7. Conclusion

presents poor results on certain graphs is not admitting failure, it is simply being thorough and honest. Negative results on some graphs can no longer be viewed as a mark of failure, and grounds for rejection. Instead, authors with the scientific fortitude to present complete and well-rounded evaluations should be applauded for their efforts.

7.2 Challenges in Security Research on Social Networks

One of the great promises of modern social networks is increased security for users. The transitive trust relationships between social users are supposed to exclude attackers and provide a buffer against spam and malware. Similarly, the centralized architecture of large, monolithic OSNs give these services unfettered capabilities to monitor all aspects of the system. This complete view is the ideal vantage point to implement stringent security systems.

Unfortunately, as exhibited in Chapters 4 and 5, OSNs are just as rife with security threats as the rest of the Internet. Users are not careful about who they friend, and trust-undermining account compromises are common. OSN providers, like other profit seeking companies, view security as an afterthought and a cost-center, not as a feature to be placed front-and-center.
Chapter 7. Conclusion

Although the proliferation of security threats on OSNs represents a fertile new area for security researchers, working in this area requires overcoming many significant challenges, including:

- **How to Gather Data**: Data is carefully guarded by OSN providers, who are often wary of security researchers and the potentially embarrassing security issues they may discover.

- **How to Ascertaining Ground-Truth**: Once data is gathered, separating malicious users and content from benign data is non-trivial.

- **Designing for Scalability**: OSN security systems must scale to handle traffic from hundreds of millions of users in real-time.

- **Coping with OSN and Attacker Evolution**: OSN feature sets are constantly changing, which alters the attack surfaces of the OSN and forces attack strategies evolve.

- **Defending Against Crowdsourced Security Threats**: Attackers are now paying real humans from crowdsourcing services to attack OSNs. These attacks are significantly more difficult to detect than traditional threats.

In this section we will address each of these challenges, and describe techniques for researchers to overcome them.
Gathering Data. Gathering data for OSN security research is extremely difficult. Many large OSNs forbid data crawling in their terms of service, and sometimes use technological means to stifle crawlers \( (e.g. \) rate limiters). Moreover, OSNs are sometimes openly hostile to security research, as they fear negative publicity from having security problems revealed publicly. We know from firsthand experience that Facebook, in particular, is willing to threaten legal action in order to halt security research on their platform.

One way for researchers to acquire quality data while avoiding technological and legal backlash is to focus on OSNs with open data policies. The best example of this is Twitter: (almost) all data on Twitter is public, and they offer simple APIs for gaining programmatic access to data. Although Twitter charges money for access to high-volume datastreams \( (e.g. \) the “Fire Hose”), lower volume streams are free for researchers \( (e.g. \) the “Garden Hose”). Even these low volume datastreams are often sufficient for basic research purposes. Twitter’s open access data policies are one of the reasons why the service is so popular among security researchers \[29, 78, 107, 164, 170, 182\].

In the cases where an OSN’s data access policies are unclear, it never hurts to email their staff and ask (very politely) if it is okay to crawl their data. If the data on the OSN you are interested in is public \( (i.e. \) you don’t need an account to view the content), then the OSN likely will not have a problem with being crawled. Of course, it is always
Chapter 7. Conclusion

best to ask first, just in case. Furthermore, make sure to be courteous when crawling by not directing too many requests at the target site. Angering the OSN’s IT staff with overzealous crawling is a good way to get your IP range permanently blocked. Finally, be aware that OSNs change their data access and privacy policies all the time; just because they said it was okay to crawl six months ago does not mean you will be able to crawl indefinitely.

Contacting the staff at OSNs (even private, membership only OSNs like MySpace, Renren, and LinkedIn) is a good way to open the door to collaborative partnerships. This is the best possible outcome for researchers, as it offers direct access to datasets that are too large to crawl, or are not crawlable at all (e.g. HTTP access logs). In our experience, the best way to approach these relationships is to not ask for data up-front. Instead, conduct some preliminary research using public or semi-public data from the OSN. Contact the OSN, inform them about your research, and offer to show them the work pre-publication. Publish the work as quickly as possible, to demonstrate your team’s competency and efficiency.

After your initial work gets published, approach the OSN about moving forward on a collaborative project. The key is to find common ground with your contacts at the OSN: find a project that is both promising research, but also of practical use to the OSN. Oftentimes, security research is an excellent target for collaboration, since the
security teams at OSNs are overworked, and they welcome the additional (free) labor researchers provide.

Make sure to be as accommodating as possible when it comes time to share sensitive data. One way to assuage concerns about data leaking is to leave the data on a machine that is within the OSN’s firewalls and has been provisioned specifically for your team. In this way, the OSN’s IT staff can monitor and make sure no data escapes the corporate firewall, while your team is free to access and experiment on the data remotely.

**Ascertaining Ground-Truth.** Once security researchers have gathered data, they immediately run into a second challenge: separating the malicious users and content from benign users and content. This process is fraught with difficulty: if there was a foolproof technique to distinguish between malicious and benign content, then OSNs would already be using that technique to prevent all attacks against their systems.

The simplest way to tackle this problem is to obtain definitive ground-truth directly from the OSN provider, *e.g.* ask them what users and content they have flagged as malicious. Obtaining data directly from the OSN mitigates the problem of false positives in the dataset. This is the strategy we use in Section 5.2.

However, not all OSN providers are forthcoming with their proprietary information. Thomas *et al.* present an innovative technique for indirectly obtaining ground-truth from an OSN: crawl the OSN for data, then periodically re-crawl and note which accounts and content get removed over time [170]. Although users may delete their
Chapter 7. Conclusion

own OSN accounts and content, these events are usually rare. Instead, any user accounts or content that have been removed are likely to have been culled by the OSNs security systems. Thus, it can be inferred that deleted materials are malicious, according to the OSN. In the event that there is a false positive (e.g. the OSN erroneously bans a real user), the user contacts the OSN and has the error corrected. This means that researchers utilizing this method are unlikely to have many false positives in their dataset. We have successfully used this technique to obtain ground-truth information about Sybil accounts on Facebook [183].

Of course, obtaining ground-truth from an OSN (either directly or indirectly) does not mean that there are no false negatives in the dataset. Ground-truth from an OSN is only as comprehensive as the OSNs in-house security systems, i.e. any malicious accounts or content that the OSN’s security systems miss will not be in the ground-truth data. Researchers should be aware of this limitation and mention it in their manuscripts.

In situations where ground-truth cannot be obtained, researchers must fall back on heuristic methods to separate malicious and benign content. For example, in Section 4.3, we apply features that are common indicators of email spam (widespread content similarity and bursty message sending) to the detection of social spam. Rahman et al. use similar methodology in their study of Facebook spam [147]. While heuristics can be successful at detecting malicious behavior and content, they are not a stand-in for ground-truth: all results derived from heuristics must be thoroughly validated.
Chapter 7. Conclusion

for false positives and negatives. There are several examples of heuristic validation of social spam in the literature [78, 147], as well as in Section 4.4 of this dissertation.

**Scalability.** Using measured data to drive the development of OSN security systems is a sound way to address real-world attacks against OSNs. However, researchers must also keep in mind that OSN security systems need to operate in real-time and at tremendous scale to be practically deployable. These scalability requirements limit the types of algorithms and techniques that can be leveraged by OSN security systems. For example, there are a wealth of very rich graph metrics that can be applied to security problems, *e.g.* random walks to detect Sybils. However, the computational cost of these techniques makes them impractical for deployment. Running complicated OSN security algorithms offline, as batch jobs, is not a solution to this problem: well-orchestrated attacks on OSNs can reach hundreds of thousands of victims in a few minutes, and 90% of the clicks (i.e. most of the practical damage to honest users) occur shortly thereafter [78].

In order to operate in real-time and at massive scale, OSN security systems have to be lightweight. Heavyweight techniques like graph clustering, random walks, and web page crawling/analysis are invaluable for ascertaining the ground-truth of unclassified datasets, and for discovering the key distinguishing features of malicious content. However, these techniques are not suitable for large scale deployment since they are too slow and cost intensive for detecting real-time threats (unless you have an un-
Chapter 7. Conclusion

limited budget for computing resources [162]). Instead, after the initial, heavyweight discovery phase, security systems should be implemented using lightweight techniques like Support Vector Machines (SVMs) [147], threshold counters (see Section 5.2), and bloom filters (see our SpikeStrip work for an example [191]). Heavyweight jobs can be run periodically in the background to re-examine traffic logs and adjust the parameters of the lightweight system.

OSN and Attacker Evolution. In addition to scalability issues, the development of security systems for OSNs is complicated by OSN's rapid pace of change. Sometimes, these changes are driven by the expanding feature set of the OSN. For example, in 2008 spam on Facebook was confined to wall posts and group discussion boards. In 2009, with the introduction of the Facebook Developer Platform, miscreants began using Facebook's APIs to build malicious applications. As of 2010, the “like” button and associated fan pages have been targeted by attackers looking to promote spam and sell popularity. These examples highlight the difficulty of maintaining up-to-date security countermeasures for a single OSN. Considering that there are many OSNs (with new ones appearing every day), all of which are constantly changing, the challenge for researchers only becomes more daunting.

In addition to OSNs evolving over time, attackers too are constantly upgrading their tactics. For example, the initial strategy used by spammers on Twitter was to follow thousands of people (follow spam) and hope those users would reciprocate by following
Chapter 7. Conclusion

back. After Twitter placed a hard upper bound on the follow/following ratio of users, spammers switched strategies to using hashtags (#) and directed messages (@) to spam users directly. As Twitter has grown in popularity, the underground has also expanded beyond simple spamming: there is now a burgeoning market for buying fake followers and paying them to retweet.

Adaptability is the key for building OSN security systems that are flexible enough to handle evolving OSN feature sets and threat vectors. Building an expert system to try and cover each and every attack surface as it gets deployed is simply not a tenable long term security strategy. Instead, systems need to be flexible enough to incorporate new features and metrics over time. For example, SVMs can be retrained offline to incorporate newly released OSN features. Similarly, our Sybil detector can be easily updated to incorporate new metrics against new attacks (see Section 5.5).

Crowdsourced Security Threats. Many existing security systems assume that online attacks are powered by automated scripts, and in some contexts this remains true. The majority of email spam is generated by botnets [76, 100, 195]. Worms that exploit common web vulnerabilities (e.g. SQL injection) periodically sweep the web [51].

However, the assumption that attacks against OSNs are powered by software automation is quickly eroding. Two studies (one of which was conducted by us) have uncovered black-market crowdsourcing websites where malicious users can cheaply hire real people to generate social spam and Sybils [131, 184]. Crowdsourcing blurs the line
Chapter 7. Conclusion

between what is real and what is fake, and challenges many assumptions that automated security systems rely on to identify malicious content. For example, crowdworkers can easily bypass CAPTCHAs to register accounts en-mass. Similarly, Sybils controlled by people send hand-written spam at human speeds, in contrast to the copy/pasted, bursty waves of messages generated by traditional spam bots.

Researchers should consider incorporating human intelligence into OSN security systems in order to combat crowdsourced threats. Many OSNs already include mechanisms for users to “flag” or “report” suspicious or abusive content. These existing systems are a step in the right direction, but more concerted efforts are needed if the threat of black-market crowdsourcing is to be countered. In our research, we have demonstrated the feasibility of using crowdsourcing to identify high-quality, difficult to detect Sybils on OSNs [183]. The existing OSN user base is also a vast, largely untapped resource for detecting security threats. For example, detecting malicious content could be “gameified” to incentive participation, such as by rewarding OSN users with badges or credits in exchange for helping to moderate content.

7.3 Future Work

The work presented in this dissertation has advanced the state-of-the-art in social networks analysis and security. However, our understanding of social networks is still
incomplete, and OSNs continue to evolve and grow at a blistering pace. Here, we briefly outline some future projects that enhance our existing work, as well as strike out in promising new researcher directions.

### 7.3.1 Identifying Malicious OSN Users Through Real-time Clickstream Analysis

Combining behavioral data with structural information from the social graphs results in very strong security systems. For example, the Sybil detector we develop and deploy in Section 5.2 leverages friend request acceptance statistics (behavioral data) as well as clustering coefficients (graph structural data). This approach is superior to earlier approaches that just leverage graph structural information, since the behavioral data raises the bar against stealthy attackers. However, despite the success of our detector, as shown in Section 5.5, it is still vulnerable to changing attack strategies.

The natural extension of our existing work is to incorporate more real-world behavioral data into our detection system. Richer behavioral data can be used to make our detector more resistant to evolving attacker strategies by covering a wider range of suspicious behaviors. Similarly, incorporating more features makes the system more resilient against false positives, since the discriminatory power of the system is increased.

The goal of this project is to incorporate real-time clickstream analysis into our Renren security system. Clickstreams are the ultimate form of behavioral data, since
they capture all user actions, as well as the ordering and temporal relations between actions. We have conducted a preliminary examination of Sybil clickstreams on Renren, and uncovered many characteristics that differentiate them from normal user’s clickstreams. Spamming and crawling behavior both stand out as specific, repetitive patterns. Similarly, Sybils do not engage in behavior that are popular with real people, such as browsing photos, which manifests as a noticeable absence in their clickstream patterns.

Although it is easy to analyze clickstreams offline using distributed analysis platforms like MapReduce, the challenge of this work is to conduct clickstream analysis in real-time, at OSN scales. Building such a system requires balancing several competing trade-offs. First, the web servers that report user clicks cannot be overburdened with statistics tracking or bookkeeping requests to back-end services. Thus, overhead on web servers must be balanced against the timeliness of information updates. Second, recording detailed clickstreams for all OSN users in-memory is simply not feasible; probabilistic data structures must be used to store the data. However, the level of data compression must be balanced against detection accuracy. We plan to use trace-driven experiments to quantify and evaluate these trade-offs before incorporating the results into a production ready security system.
7.3.2 Detecting Personal Information Leaks from OSNs

It is an understood fact that social network users give up some privacy in exchange for free, reliable service. This is the prevailing business model amongst OSN providers: monetize the user base by leveraging their personal information to target ads. Although numerous researchers have proposed tools to enhance privacy on OSNs [66, 81], or even replace centralized OSNs with more secure, distributed systems [18, 22, 22, 38, 56, 158, 165, 172, 192], none of these techniques have caught on with social network users. It appears that the vast majority of OSN users are unconcerned about the privacy implications of OSN providers having unfettered access to vast troves of personal data.

However, the prevailing privacy status quo is based on the assumption that personal data is only stored and used by OSN providers (or social applications that the user has opted-in to). There is now evidence that OSN providers are leveraging user’s personal data to power targeting systems on unrelated websites. One example of this is the Google+ OSN. Google recently changed their terms of service to allow them to share personal data between their different services [171]. This allows Google to use information from a user’s Google+ profile to personalize Google Search results. This practice is concerning, because most users have no idea that content is being personalized and targeted to them, which may promote social problems like the filter bubble [139].

Once OSN providers begin sharing data with outside partners, many anti-consumer forms of personalization become possible. One prominent example is using sensitive
data from OSNs to personalize user’s online shopping experiences. Consider the example of Orbitz, an online travel booking site, which was recently caught changing their product listings to highlight more expensive options for Mac OSX users [121]. In this case, the personalization was based on HTTP User-Agent headers. However, it is obvious how OSN data could also be used to alter e-commerce sites, or even conduct price discrimination. If an OSN provider formed a partnership with an e-commerce site, they could easily share data behind the scenes, and users would be totally unaware that their shopping experience was being altered.

The goal of this work is to monitor and understand how personal data from user’s OSN accounts is being used to personalize content for them around the web. The high-level methodology is straightforward: sign up for many user accounts on various OSNs and vary their personal details (e.g. gender, age, etc.) and browsing histories. Once then accounts are set up, crawlers can be used to browse and query popular websites while maintaining active cookies for the OSN accounts. By comparing the crawl results for OSN cookie-enabled pages with non-cookied, control crawls, instances of personalization can be clearly identified.

The implications for this work are potentially far reaching. Our preliminary results confirm that Google is personalizing search results based on Google+ profile information, and we expect to find many more instances of OSN-based personalization. The long term goal is to create an infrastructure for continuous monitoring of website per-
sonalization, and make the real-time data available as a website. This website will serve as a portal for consumer and privacy advocates who are concerned with the proliferation of personal information from OSNs, and the impact of personalization on Internet users.

7.3.3 End-to-End Models of Information Consumption on OSNs

Information dissemination on OSNs has emerged as a very hot topic both inside and outside the research community. Marketers are interested in understanding how information flows through OSNs so that they can maximize the effectiveness of viral advertising campaigns. Sociologists are looking to social networks to understand how trends, ideas, and memes spread around the world. Even economists are attempting to leverage sentiment gleaned from OSNs to predict the movements of the stock market [35].

Existing work on information dissemination in OSNs has focused on measuring and modeling how users share information across the social graph using visible interactions. Bakshy et al. perform controlled experiments on Facebook to observe how likely users are to share information that they are exposed (or not exposed) to by their friends [24]. Similarly, Myers et al. look at how information gets retweeted across Twitter and develop models based on the data [133].
Chapter 7. Conclusion

Unfortunately, existing work presents a fundamentally incomplete picture of how information is actually consumed on OSNs. Existing models only consider two aspects of the social network: static topology (e.g. friendship links), and visible interactions (e.g. sharing and resharing, tweeting and retweeting, etc.). These two features are important for information dissemination, but they are not the whole story. Latent interactions (e.g. News-Feed browsing) determine how many people actually see the hyperlinks to shared information. As we show in Chapter 3, visible and latent interactions have drastically different dynamics. By modeling information dissemination purely based on who is sharing, existing work underestimates how many people may see that information and actually consume it.

Similarly, the number people who click on each link is also not considered by existing information dissemination models. The prevailing assumption is that if a user shares a link to some content, they must have at least read it, i.e. diffusion is assumed to equal consumption. However, there are numerous reasons why a user would share or reshare a link without actually clicking on it. A user may retweet content from a well-known figure based solely on the originator’s reputation. Or, a user may retweet content in order to ingratiate themselves with the originator, irrespective of the actual content. The point is that visible interactions are not synonymous with information consumption. The only way to truly know if users are consuming content is to measure and quantify whether they are actually clicking on shared links.
Chapter 7. Conclusion

Our goal is to measure and quantify all four aspects of OSN information dissemination and consumption: graph topology, visible interactions, latent interactions, and clicks. One way to implement this as with a user study: have volunteers install browser plug-ins that record their activities while on OSNs. The data can be fully anonymized to protect user’s privacy, and we do not need to record any actual content in order to measure trends and develop models. A second way to get data is to partner with a large OSN. We have existing relationships with several OSNs that we can leverage to enable this study. Once we have acquired and analyzed our dataset, we will be able to create empirical models that capture the observed dynamics.
Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


