University of Crete
Computer Science Department

Antisocial Networks: Turning a Social Network into an Attack Platform

Andreas Makridakis

Master's Thesis

November 2009
Heraklion, Greece

Abstract

World Wide Web has evolved from a collection of static HTML pages to an assortment of "Web 2.0" applications. Examples of "Web 2.0" applications include wikis, blogs, video sharing web sites, social networking web sites, *etc.* Since the establishment of the first online social network *SixDegrees.com*, in 1997, these sites are becoming more popular by the day. Millions of people daily use social networking web sites, such as `facebook.com`, `myspace.com`, `orkut.com`, and `linkedin.com`. As a side-effect of this fast growth, possible exploits can turn them into platforms for antisocial and illegal activities, like DDoS attacks, privacy violations, disk compromising, malware propagation, *etc.*

In this thesis we show that social networking web sites have properties to become attack platforms. We introduce a new term, *Antisocial Networks.* Antisocial Networks are distributed systems based on social networking web sites that can be exploited by attackers, and directed to carry out network attacks. Malicious users are able to take control of the visitors of social web sites by remotely manipulating their browsers through legitimate web control functionality such as image-loading HTML tags, JavaScript instructions, *etc.*

We start by identifying all the properties of Facebook, a real-world online social network, and then study how we can utilize these properties and transform it into an attack platform against any host connected to the Internet. Towards this end, we develop a real-world Facebook application that could perform malicious actions covertly. We experimentally measure its impact by studying how innocent Facebook users can be manipulated into carrying out a Denial-of-Service attack. Finally, we explore other possible misuses of Facebook and how they can be applied to other online social networks.

Supervisor: Professor Evangelos P. Markatos

Περίληψη

Ο Παγκόσμιος Ιστός έχει εξελιχθεί, από μια συλλογή στατικών HTML σελίδων, σε μια συλλογή εφαρμογών τύπου "Web 2.0". Παραδείγματα εφαρμογών τύπου "Web 2.0" είναι τα wikis, τα blogs, οι σελίδες διαμοιρασμού βίντεο, οι σελίδες κοινωνικής δικτύωσης, κ.τ.λ. Από την σύσταση του πρώτου διαδικτυακού κοινωνικού δικτύου, που ονομαζόταν SixDegrees.com, το 1997, οι σελίδες κοινωνικής δικτύωσης μέρα με τη μέρα γίνονται όλο και πιο δημοφιλείς. Καθημερινά εκατομμύρια άνθρωποι χρησιμοποιούν σελίδες κοινωνικής δικτύωσης, όπως τις σελίδες facebook.com, myspace.com, orkut.com και linkedin.com. Μια παρενέργεια της ταχείας ανάπτυξης των σελίδων κοινωνικής δικτύωσης, είναι η πιθανή μετατροπή τους σε πλατφόρμες διεξαγωγής αντικοινωνικών και παράνομων δραστηριοτήτων, όπως κατανεμημένες επιθέσεις άρνησης υπηρεσιών, παραβίαση προσωπικών δεδομένων, έκθεση σκληρών δίσκων σε κίνδυνο, διάδοση κακόβουλου λογισμικού, κ.τ.λ.

Σε αυτή την εργασία αποδεικνύουμε ότι οι σελίδες κοινωνικής δικτύωσης έχουν ιδιότητες ώστε να μετατραπούν σε πλατφόρμες διεξαγωγής διαδικτυακών επιθέσεων. Εισάγουμε έναν νέο όρο, εν΄ ονόματι **Αντικοινωνικά Δίκτυα**. Τα Αντικοινωνικά Δίκτυα είναι κατανεμημένα συστήματα που βασίζονται σε σελίδες κοινωνικής δικτύωσης και είναι δυνατόν να εκμεταλλευτούν από επιτιθέμενους, προκειμένου να διεξαχθούν διαδικτυακές επιθέσεις. Κακόβουλοι χρήστες έχουν την δυνατότητα να ελέγχουν τους επισκέπτες σε σελίδες κοινωνικής δικτύωσης, μέσο της απομακρυσμένης χειραγώγησης των προγραμμάτων που χρησιμοποιούν για την περιήγηση τους στο Διαδίκτυο, χρησιμοποιώντας νόμιμες λειτουργίες για τον δημιουργία σελίδων, όπως HTML ετικέτες για την εισαγωγή εικόνων, εντολές της γλώσσας JavaScript, κ.τ.λ.

Αρχικά, προσδιορίζουμε όλες τις ιδιότητες του Facebook, ενός πραγματικού διαδικτυακού κοινωνικού δικτύου, και στην συνέχεια μελετάμε τους τρόπους με τους οποίους μπορούμε να αξιοποιήσουμε αυτές τις ιδιότητες, προκειμένου να το μετατρέψουμε σε μια πλατφόρμα διεξαγωγής επιθέσεων εναντίον οποιουδήποτε μηχανήματος είναι συνδεδεμένο στο Διαδίκτυο. Προς το σκοπό αυτό, αναπτύξαμε μια εφαρμογή στο Facebook, η οποία, συγκαλυμμένα, είναι ικανή να πραγματοποιεί κακόβουλες ενέργειες. Χρησιμοποιώντας μια πειραματική διαδικασία, εξακριβώσαμε τις επιπτώσεις που έχει η εφαρμογή, μελετώντας τον τρόπο με τον οποίο ανυποψίαστοι χρήστες του Facebook, μπορούν να χειραγωγηθούν με σκοπό την διεξαγωγή μιας επίθεσης άρνησης υπηρεσιών. Εν΄ τέλει, διερευνούμε άλλες κακόβουλες χρήσεις του Facebook και τους τρόπους με τους οποίους μπορούν να εφαρμοστούν και σε άλλα διαδικτυακά κοινωνικά δίκτυα.

Επόπτης: Καθηγητής, Ευάγγελος Μαρκάτος

Acknowledgments

I would like to thank my supervisor, Professor Evangelos P. Markatos, for his valuable guidelines in my academic steps in the field of Computer Science. I, also, feel grateful to Dr. Sotiris Ioannidis, for his invaluable help and cooperation over the last two years, and for a real commitment to my technical and professional growth. I am deeply grateful to Elias Athanasopoulos, who gave me the opportunity to work on this subject and whose contribution was a fundamental key for writing this thesis. My best regards to Spyros Antonatos, Demetres Antoniades and Kostas G. Anagnostakis for their joint work on FaceBot.

My best thanks to all former and current members of the Distributed Computing Systems Laboratory, division at ICS/FORTH, Antonis Papadogiannakis, Nikos Nikiforakis, Christos Papachristos, Michalis Polychronakis, Elias Athanasopoulos, Demetres Antoniades, Spyros Antonatos, Giorgos Vasiliades, Iason Polakis, Alexandros Kapravelos, Antonis Krithinakis, Michalis Foukarakis, Manos Athanatos, Eleni Gessiou, Vasilis Pappas, Giannis Velegrakis, Giorgos Kondaxis, Spyros Ligouras, Lazaros Koromilas, Giorgos Chinis, Zaxarias Tzermias, Thanasis Petsas, Apostolis Zarras, Harris Papadakis, Nikos Hatzibodozis, Manolis Stamatogiannakis, Kallia Marakomihelaki, Meltini Christodoulaki and Anna Doxastaki, that contributed for a pleasant and productive environment over the last three years in the lab.

My special thanks to my best friends, Giorgos Stratakis, Kostas Tsikrikas and Diamantis Antoniou, for their support and for sharing with me over the last seven years of my life. They were always by my side, in the joys and sorrows. I feel that they are my brothers.

I want to thank the Computer Science Department's graduate programme secretary, Mrs Rena Kalaitzaki, for her invaluable help all these years. Many thanks to Giorgos Koutras for his help during the last months. I would, also, like to thank Professor Christos Nikolaou, for his presence in the examining committee of this thesis.

I would like to express my deepest gratitude to my parents, Elias and Eleni, and my lovable little sister, Stella, for their support, patience, encouragement and wise advice during my whole life. Truly, I never would have made it through, without their love and understanding.

Finally, I would like to thank all those who believed in me, from the time of my birth until now.

This thesis is based on the paper: **"Antisocial Networks: Turning a Social Network into a Botnet"**, authored by Elias Athanasopoulos, Andreas Makridakis, Spyros Antonatos, Demetres Antoniades, Sotiris Ioannidis, Kostas G. Anagnostakis, and Evangelos P. Markatos. The paper was accepted for the proceedings of the 11th Information Security Conference (ISC) in September 2008 (Taipei, Taiwan).

Οι επαναστάτες προσδοκούν σε ένα καλύτερο αύριο.

Όχι μόνο γι' αυτούς, αλλά για την κοινωνία γενικότερα.

Τους χαρακτηρίζει η αισιοδοξία και όχι η ηττοπάθεια.

Έχουν μάθει να πολεμούν και να προσπαθούν, υπό αντίξοες συνθήκες.

Προσπάθεια, μια λέξη μαγική, πολυδιάστατη, πολύχρωμη.

Οδηγός και συνοδοιπόρος μιας ολόκληρης ζωής.

Οφείλω τούτη την εργασία σε έναν αληθινό επαναστάτη, ο οποίος

με έμαθε να προσπαθώ, γιατί όπως μου έλεγε πάντα:

"Η προσπάθεια καταξιώνει τη ζωή σου".

Ανδρέας Η. Μακριδάκης

Στον πατέρα μου

Contents

List of Figures

List of Tables

1
Introduction

In the last years, the static and non-interactive web sites have moved towards rich Web applications, like online collaborative encyclopedias, personal and corporate blogs, mashup services, social networking web sites, *etc.* We ofter refer to these as the "Web 2.0" [67]. In recent years, the popularity of online social networks (OSNs) [51] is increasing rapidly every day. The online communities created by OSNs compose a fast growing phenomenon on the Web, by introducing new modes of social interaction among people from all around the world. Social networking web sites have attracted millions of users, many of whom have integrated the browsing of these sites into their daily activities. OSNs are useful for keeping in touch with friends, making new contacts, research collaboration, information sharing, political campaigns [61, 69], *etc.* Some OSNs are used for professional contacts, *e.g.* LinkedIn [19] and XING [33], where a user can discover business connections when he is looking for a new job, while others, such as Facebook [3], MySpace [20] and Orkut [24], are friendship-focused and are primarily used for communication, photo sharing, video sharing and entertainment.

The structure of an OSN is quite simple. Users register to the site and create their virtual persona in the form of an online profile. They describe their interests, activities, favorite music, *etc.* Also, they can provide personal information, like political and religious views, their current relationship status and details about their education and work. Finally, they can add friends/contacts to their profile. Adding a friend involves a confirmation step from the other party. Through this possibility, a user can find old classmates and long-lost friends, his coworkers and relatives, strangers with the same interests, his circle of acquaintances, *etc* and share photos & videos, exchange private messages, chat online, play games, and so on. Perhaps, the above capabilities would be impossible in the absence of OSNs. The view of a user's profile is usually limited to the friends of that user, unless the user wants the profile to be public. In that case, all users who have joined the same network with that user, can view it. Users can join one or more networks, each based around a workplace, region, high school or college. Social networking web sites also support the creation of groups by anybody, where users can find content related to their favorite musicians, movies, tv-shows, *etc.* As of the most popular social networking web sites are Facebook and MySpace. Alexa.com [1] ranks them among the top ten visited web sites on the Internet. Figure 1.1 presents the percent of global Internet users who daily visited `www.facebook.com` and `www.myspace.com` during the last 6 months, as computed by Alexa.

MySpace [20] was launched in January 2004. As reported in [21], it has nearly 125 million monthly active users around the globe and nearly

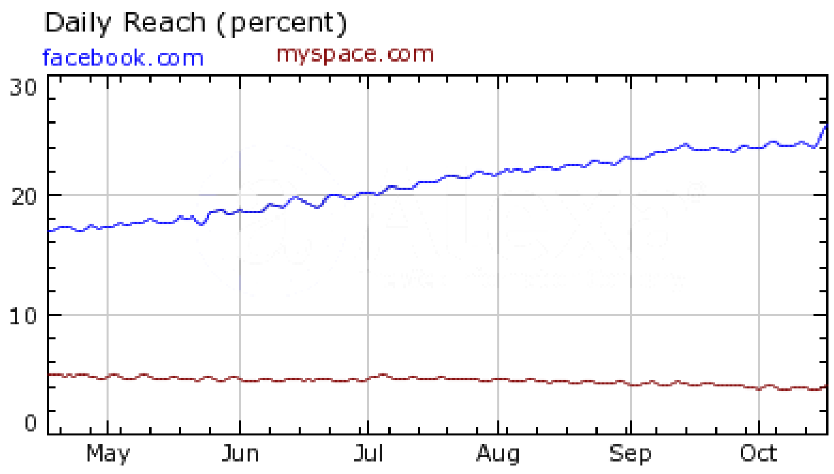

FIGURE 1.1: `Facebook.com` and `MySpace.com` daily reach during the last 6 months.

65 million total unique users in the United States. One of the reasons that MySpace has a massive response, is the ability for users to customize their profile page. Users can determine the fonts and colors of their profile and provide their favorite song, via MySpace Music, that starts playing when a user visits their online profile. MySpace helps the music industry, by allowing artists to upload their entire discographies, consisting of MP3 songs. Through this service many singers have gained fame through MySpace, as their songs can be accessed from millions of people.

Facebook [3] started in February 2004, as a project of a student in Harvard University to keep track of schoolmates. The web site's membership was initially limited to Harvard students, but was expanded to other colleges in the Boston area, the Ivy League, and Stanford University. Facebook has now grown up to serve more than 300 million people from around the world, as well as more than 8 billion minutes are spent worldwide on browsing the site each day [15]. Also, more than 65 million users are currently accessing Facebook through their mobile devices, via Facebook Mobile. It forms a large online database of photos, as more than 2 billion photos are uploaded to the site each month. Additionally, more than 14 million videos are uploaded each month and more than 2 billion pieces of content (*e.g.* web links, news stories, blog posts, notes, photos, *etc.*) are shared each week. Moreover, there are more than 70 translations available on the site. Concerning its technology, it is the second most-visited PHP site in the world, and one of the largest MySQL installations anywhere, running thousands of databases. Facebook has a very interesting and powerful feature, the Facebook applications. Facebook builders have implemented a platform on top of which developers can build complete applications. In the *Facebook Platform* any developer with a good idea and basic programming skills can create one from scratch. Over one million developers and entrepreneurs from more than 180 countries have done so, as reported by Facebook Press Room [15]. Users can add these applications to their profile and invite their friends[1] to add them too. Typical applications involve solving a quiz, filling questionnaires, playing games and many more. Up to date, the number of Facebook applications has surpassed 350,000. Additionally, more than 250 applications have more than one million monthly active users. In a nutshell, Facebook applications can be considered as XHTML snippets that inherit all properties of web applications. For more details, about Facebook applications architecture, please refer to Chapter 3.

The massive adoption of online social networks by Internet users, as described above, provides us with a unique opportunity to study possible exploits that will turn them into platforms for antisocial and illegal activities, like DDoS attacks, malware propagation, spamming, privacy violations, disk compromising, *etc.* Online social networks have by nature some intrinsic properties that make them ideal to be exploited by an adversary. The most important of these properties are: (*i*) a very large and highly distributed user-base, (*ii*) clusters of users sharing the same social interests, developing

[1]An average user has 130 friends on the site [15].

trust with each other, and seeking access to the same resources, and (*iii*) platform openness for deploying fraud resources and applications that lure users to install them. All these characteristics give adversaries the opportunity to manipulate massive crowds of Internet users and enable them to commit antisocial acts against the rest of the Internet, without their knowledge. Apart from controlling social network users and drive them to launch attacks against third parties, an adversary can harm the social networks' users themselves. For example, a malicious user may seek to harvest the personal information that a social user enters in his online profile page.

In this thesis we explore the above properties, develop a proof of concept exploit, and analyze its impact. By experimentally measure its firepower, we can coin a new term, *Antisocial Networks.* We define ***Antisocial Networks*** as a *social network, deviously manipulated for launching activities connected with fraud and cyber-crime.*

1.1 Contributions

The work done in this thesis has unleashed in public the skeleton of a proof of concept application, which is able to transform a social utility with open architecture, *i.e.* `facebook.com`, to an attack platform. In this way, we believe that we assist crucially in the creation of safer social network platforms.

More specifically, the main contribution of this thesis is a first investigation into the potential misuse of an online social network for launching DDoS attacks on third parties. We have built an actual Facebook application, that may turn its users into a *FaceBot.* We used our application to carry out a complete evaluation of our proof of concept attack via real-world experiments. Extrapolating from these measurements along with popularity metrics of current Facebook applications, we show that owners of popular Facebook applications have a highly distributed platform with significant attack firepower under their control.

1.2 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 presents the reasons that motivated us to examine if `facebook.com` can be used as an attack platform, *e.g.* to launch DDoS attacks against third parties. In Chapter 3 we describe the construction details of Facebook applications and list all tools provided by Facebook Platform for easy deployment of these applications. Chapter 4 describes in detail our proof of concept Facebook application, called *Photo of the Day*, that is instrumented to launch a DDoS attack against a victim web server. Chapter 5 outlines the experimental evaluation of our *FaceBot*. Chapter 6 explores countermeasures for defending and preventing a *FaceBot* based attack and presents pieces of advice on how to create safer social network platforms. In Chapter 7 we list other possible misuses of online social networks. Finally, Chapter 8 presents a brief survey of related work, and Chapter 9 concludes the thesis.

2
Motivation

In this Chapter we briefly describe the motivations that led us to study if and how online social networks can turn them into attack platforms. More specifically, Distributed Denial of Service (DDoS) attacks and Puppetnets [62] gave us the rise to analyze how Facebook applications can be used by an adversary in order to carry out network attacks. We discuss the details of each incentive and how they are related to this thesis.

2.1 Distributed Denial of Service Attacks

A Distributed Denial of Service attack is an attempt to make a computer resource unavailable to serve its well-intentioned users. This type of attack generally consists of the efforts of a network of zombies computers that simultaneously are orchestrated to prevent a network service from functioning as it is designed for. These zombies are under the control of a master machine. A strength is the fact that the zombie computers are often being infected, without their knowledge. Also, Internet Service Providers (ISPs) are rarely able to determine the master computer behind a DDoS attack. Thus, attackers are able to hide their footprints. A DDoS attack may have different aims, such as maximize the amount of ingress traffic towards the victim, the amount of egress traffic from the victim, *etc.* A standardized method for conducting DDoS attacks involves filling up the target computer with HTTP requests for data, such that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable. The inability of responding to legitimate browsing requests, is usually due to the consumption of resources, like bandwidth or disk space. This type of attacks do not compromise user data. Most of the times, victims of these attacks are web sites servers that attract a massive crowd of users daily. If popular web sites, like search engines, mailing services, social networks, *etc* are down for hours, they will suffer a significant loss of e-commerce and advertising revenue. Sometimes, depending on the amount of time a site is out of order, the loss can be amounted to million of dollars.

Traditional Denial of Service attacks have been known and analyzed by researchers for decades. However, DDoS attacks are more recent. The authors of [50] describe the first well-documented DDoS attack, occurred in August 1999. The attackers deployed a DDoS tool, called Trinoo [30], in at least 227 computer systems, managing to perform a distributed SYN DoS attack. Their aim was to flood a computer located in the University of Minnesota. Effectively, this computer was inaccessible for more than two days. On August 6, 2009, Twitter [32], a well-known online social network, where more than 40 million people announce what they are doing at any given moment, was the target of a DDoS attack [40]. The outage lasted about three hours. Barrett Lyon in [42] put out a possible reason that transformed Twitter into an obvious DDoS attack target. According to him, Twitter's computer resources and servers appear to have only one network provider, which was rather insane those days.

2.2 Puppetnets

Puppetnets [62] exploit the design principles (*e.g.* the programming languages, protocols, and security policies) of World Wide Web. Web pages can include links to elements located at different domains, other than the one they are hosted at. For example, plenty of web pages contain references to images, hosted at different pages, by using the common <*img/*> HTML tag. A malicious user can craft special web pages that contain thousands of links pointing at a victim site. When an unsuspecting user visits that pages, his browser unknowingly starts downloading elements, *e.g.* images, from the victim site and thus consuming its bandwidth. As is easily understood, Puppetnets can instrument Web browsers to participate in a Distributed Denial of Service attack. The firepower of this attack increases with the popularity of the malicious page, similar to the slashdot effect [58]. A huge set of indirectly misused browsers can create an impromptu botnet-like infrastructure that can cause significant damage to the victim site. The indirectly misuse of Web browsers, transmutes the attack to be less likely noticed by users who unwittingly participate on it, thus the attack maximizes its effectiveness.

Puppetnets use a number of techniques to make the attacks more effective. The use of JavaScript permits more flexible and powerful attacks as unsuspecting users can repeatedly download elements from victim sites or perform other kinds of attacks, such as port scanning and computational attacks. The firepower of Puppetnets depends on three main factors. First, the popularity of the malicious page. Second, the duration of visits to the malicious page. The more the unsuspecting user stays on the malicious page, the longer the attack takes place in the background. Third, the bandwidth of unsuspecting users and their latency to the victim site. These factors determine the number of downloads per second an attacker can achieve.

The HTML and JavaScript source code listed in Figure 2.1 demonstrates a Puppetnet DDoS attack.

```
<script type = ‘text/javascript’>

   img = new Image(30, 30);

   function ddos(){
      var now = new Date();
      img.src = ‘http://victim-site/image_1.png?’ + now.getTime();
      setTimeout(‘ddos()’, 30);
      return;
   }

</script>

<iframe name = ‘my_frame’ style = ‘width:0px; height:0px;
border: 0px’ src = ‘original_page.html’ onLoad = ‘ddos()’>
</iframe>
```

FIGURE 2.1: Sample source code for a Puppetnet DDoS attack.

The attacker uses a hidden frame to launch the attack-bearing page, so that the unsuspected user that browses the page that contains the above code can not observe frame’s contents. Iframe’s *onload* event causes the execution of a JavaScript function. This function instructs the browser to fetch an image from the victim site every 30 milliseconds, by using timeouts.

Through this process, the attacker has the ability to create a large number of requests towards the victim site. In order to prevent client-side caching, because all HTTP requests target the same image, the attacker appends an invariant modifier string (the number of milliseconds since midnight of January 1, 1970) to the attack URL. According to the URL specification[1], this modifier is ignored by web servers but not by clients, as it indicates whether the corresponding image is cached or not. When a large number of Internet users, around the world, simultaneously visit the above malicious web page, an effective DDoS attack is conducted against the victim site's web server. The programming skills and lines of source code needed for the attack are minimal enough.

Puppetnets gave us the rise to study and identify if Facebook applications can act like them and used by malicious developers to launch DDoS attacks against third parties.

In the next Chapter, we will examine how Facebook applications work and analyze all the essentials provided to the developers for easy deployment of applications that live inside the social network itself.

[1]RFC 1738 - Uniform Resource Locators (URLs):
`http://www.ietf.org/rfc/rfc1738.txt`

3

Facebook Applications

In this Chapter we present the construction details of Facebook applications. On May 24, 2007, Facebook builders launched the Facebook Platform, providing a framework for software developers to create lightweight applications, by leveraging the underlying social graph. This innovation attracted more and more users to create an account on Facebook. We summarize the core components and libraries provided to the developers, in order to build social experiences that give users the power to access applications that amplify their ability to interact with each other in new and interesting ways.

3.1 Platform Overview - Core Components

Facebook Platform provides all the essentials needed for easy deployment of applications that live inside the social network itself. A user who wants to build a Facebook application must simply add the Developer Application [9] to his account. One major requirement is the presence of a web server for hosting the new application. In order for an application to work correctly with Facebook, the user needs to take the following steps to prepare his web server:

1. He should verify that the server can provide an HTML file. That is, he can view a file in a browser installed on a different computer than his server.

2. He should upload the appropriate Facebook client library to his server. The server side part of the application can be developed in whichever development environment the user prefers. Facebook and other third party application developers have created client libraries for these environments. Facebook officially supports PHP, JavaScript, Connect for iPhone and Flash/ActionScript client libraries. Additionally, Facebook does not provide official support for the following client libraries: Android, ASP.NET, ASP (VBScript, JScript), Cocoa, ColdFusion, C++, C#, D, Emacs Lisp, Erlang, Google Web Toolkit, Java, Lisp, Perl, Python, Ruby on Rails, Smalltalk, Tcl, VB.NET and Windows Mobile.

3. If the application needs to store[1] user or application information in a database, he should install a relational database management system, such as MySQL, on the server.

[1]Storable Data:
`http://wiki.developers.facebook.com/index.php/Storable_Information`

A major advantage, in case the user does not occupy a web server, is the usage of a hosting service, specifically designed for Facebook applications. Facebook has partnered with some companies to supply developers everything they will need to create applications on Facebook Platform [12]. For example Joyent[2], a free hosting service, comes with the Facebook PHP client library installed, along with MySQL and PostgreSQL support.

After configuring the web server or the hosting service, the developer, using the Developer Application, fills out a form and submits the application. The form has many fields, such as the application's name, the application's description, the IP address of the hosting web server, *etc.* Two of the most important fields are: (*i*) the Canvas page URL, and (*ii*) the Canvas callback URL. A Canvas page is the main page of an application on Facebook. When users access the application they are redirected to this URL. Its format is '`http://apps.facebook.com/canvas_page_name`', where 'canvas_page_name' is usually the name of the application. The Canvas callback URL is the address of the web server or hosting service where the application lives on. Typically, a few days after submitting the application the Facebook Platform Team notifies the developer either that the application was successfully accepted or that it was rejected. If the application is accepted, it will be added to the Application Directory[3]. This allows users to sight the application when searching or browsing the Application Directory and install it to their profile.

Facebook Platform comprises a number of core entities for easy creation of Web applications that live inside Facebook and which are freely available to every Facebook user. Through the following components, the developer has access to the social graph.

[2]Free Facebook Applications Developer Program:
`http://www.joyent.com/products/joyent-developer-programs/free-facebook-dev-program`

[3]Facebook Application Directory: `http://www.facebook.com/apps/directory.php`

3.1.1 Facebook Markup Language

The Facebook Markup Language (FBML) [11] is a subset of HTML along with some additional tags specific to Facebook. Specific tags have a common form: <*fb:tagName/*>. FBML lets applications to interact with their users and users' friends. There are plenty of FBML tags, which can be organized into the following categories: social data tags, sanitization tags, design tags, component tags and control tags.

- *Social data tags* retrieve and format data to the user accessing the application. These data can have many forms, such as user information (first name, last name, *etc*), group[4] or photos information. A widely used social tag is the <*fb:name/*>, which displays the user's name in a variety of ways.

- FBML uses *sanitization tags* to enforce site standards both internally and on developers' applications created for Facebook. For example, the <*fb:swf/*> tag controls how SWFs begin playing inside applications.

- By providing predefined *design tags*, Facebook Platform helps developers mingle their applications into the style and look & feel of the host site (`www.facebook.com`).

- *Component tags* create widget-like components that allow user interaction with an application, such as the ability to provide comments through a comment board inside the application.

- *Control tags* control how FBML renders information on a page. For example, a developer has the ability to show content only to the owner of an application profile box, through the <*fb:visible-to-owner/*> tag.

[4]According to Facebook statistics [15], more than 45 million active user groups exist on the site.

3.1.2 Facebook Query Language

The Facebook Query Language (FQL) [14] allows a developer to use an SQL-style interface to easily query some Facebook social data, such as the full name or profile picture of a user. If the developer knows how to use SQL, it should be pretty straightforward to execute FQL queries. Facebook Platform provides several tables[5], as a reference for constructing FQL queries.

3.1.3 Facebook JavaScript

Facebook JavaScript (FBJS) [10] permits developers to use JavaScript in their applications. FBJS has the same syntax, as the traditional JavaScript. FBJS source code gets parsed, and any identifiers (function and variable names) get prepended with the application's unique identifier, preventing the sandboxing of the code, without using iframes. Also, FBJS supplies a powerful AJAX object and an animation library for developers.

3.1.4 Facebook API

Concerning the server side part of an application and the available client libraries, we should take into account the Facebook API [5]. Using this API, a developer can add social context to his application by utilizing profile, friend, fan page, group, photo, and event data. For example, through specific methods, a developer can collect users' hometown location, high school information, favorite quotes, *etc.* The API uses a REST-based interface. This means that the Facebook API method calls are made over the Internet by sending HTTP GET or HTTP POST requests to the Facebook API REST server. Thus, almost any programming language can be used to communicate over HTTP protocol with the REST server to retrieve all the social data needed.

[5]FQL Tables: `http://wiki.developers.facebook.com/index.php/FQL_Tables`

3.2 How an FBML Canvas Page Works

When an user accesses a Canvas page, several steps occur in the Facebook REST server and the hosting server in order to render application's contents to the user's browser. These steps are listed in Figure 3.1. Initially, the user's browser requests the Canvas page URL. The Facebook server, that receives the request, sends an HTTP POST to the Callback URL on the hosting server, asking for the FBML of the Canvas page. If the developer makes calls to the Facebook API to retrieve social data, then the hosting server sends an HTTP POST or HTTP GET request to the Facebook REST server in order to receive the needed data. After executing all API method calls, the hosting server returns the resulted FBML to the Facebook REST server. The Facebook server transforms that FBML into HTML and sends it back to the user's browser.

From the above, we can observe that the Facebook REST server acts as a proxy between the user's browser and the hosting server.

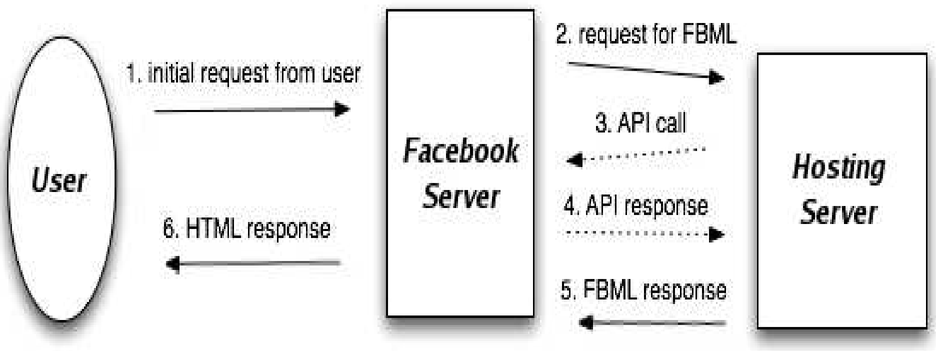

FIGURE 3.1: How an FBML Canvas page is rendered.

4
A Proof of Concept Application

In this Chapter we present our Facebook application, that can turn its users into a *FaceBot*. We analyze the design of the *Photo of the Day* application, and explain the properties that can transform it to an application that could perform malicious actions covertly.

4.1 Photo of the Day: From Facebook to FaceBot

To take advantage of a social web site, like Facebook, for launching DoS attacks, the adversary needs to create an application, which embeds URIs to a victim web server. These URIs must point to documents hosted by the victim, like images, text files, media files, *etc.* When a user interacts with the application, the victim host will receive unsolicited HTTP GET requests. These requests are triggered through Facebook, since the application lives inside the social network, but they are actually generated by the Web browsers used by the users that access the application. We define as ***FaceBot*** the collection of the users' Web browsers that are forced to generate requests upon viewing a malicious Facebook application. Schematically, a FaceBot is presented in Figure 4.1. The cloud groups a collection of Facebook users who browse a malicious application in Facebook. This causes a series of requests to be generated and directed towards the victim.

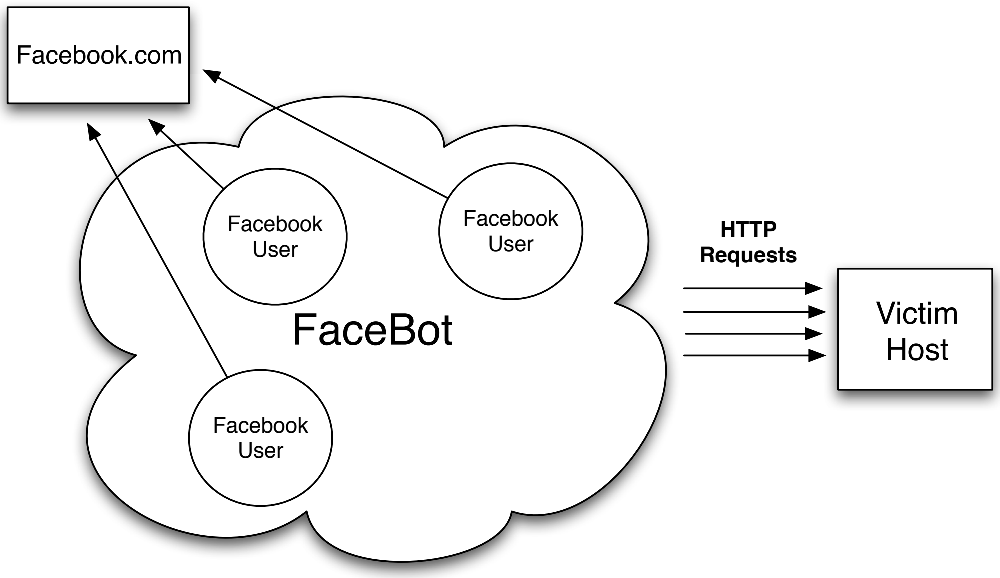

FIGURE 4.1: The architecture of a FaceBot. Users access a malicious application in the social site (`facebook.com`) and subsequently a series of HTTP requests are created, which target the victim host.

4.2 FaceBot Design

Our initial vision was to create a first *proof of concept FaceBot* for demonstration purposes, while at the same time not causing any harm to real

Facebook users. Thus, we created a real-world Facebook application, which we call "*Photo of the Day*" [25], that presents a different photo from National Geographic to Facebook users every day. Each time a user visits the *Photo of the Day* application, an image from the respective service of National Geographic[1] appears [23]. When a Facebook user accesses the application, he can see the photo of the day, along with a brief description below it. Also, when the user clicks on the photo, a wallpaper version is provided. Moreover, the name of the photographer who took the picture is displayed. Finally, the user can invite his friends, who do not have *Photo of the Day*, to add it to their profile.

During the deployment of the *Photo of the Day* application, we didn't employ any obligatory invitations during its installation in a user's profile. We mention this, because it is very common that Facebook applications require a user to invite a subset of his friends, and thus advertise the application to the Facebook community, prior the installation. This practice helps in the further propagation of the application in Facebook. Typically, a user must announce the application to about 20 of his friends in order to proceed with the installation[2].

Concerning the client libraries, as mentioned in Section 3.1, the *Photo of the Day* application is implemented on top of the PHP[3] library.

4.2.1 Malicious Attributes of "Photo of the Day"

In order to modify the *Photo of the Day* from an innocent-looking application to an application that could perform malicious actions covertly, we have placed special FBML tags in its source code, so that every time a user

[1]National Geographic has specific terms for content distribution, which are not violated by this work [22].

[2]Currently, Facebook bans 'forced' invites [13].

[3]Official PHP Client Library:
`http://wiki.developers.facebook.com/index.php/PHP`

visits the application's Canvas page, HTTP requests are generated towards a victim host. More precisely, the application embeds four hidden frames with inline images hosted at the victim. Each time the user interacts with the application, the inline images are fetched from the victim, causing the victim to serve a request of 600 KBytes, but the user is not aware of that fact, because the images are never displayed. We list a portion of our sample source code which is responsible for fetching an inline image from a victim host and placing it to a hidden frame inside the *Photo of the Day* application, in Figure 4.2.

Notice, that our proof of concept application was absorbing a fixed amount of traffic from the victim host. An adversary could employ more sophisticated techniques and create a JavaScript snippet, as the one presented in Section 2.2, which continuously requests documents from a victim host over time. In this way the attack may be significantly amplified.

4.2.2 Why Hidden Frames ?

The hidden frames are included in the application's HTML source code via the *<fb:iframe/>*[4] FBML tag. The traditional *<iframe/>* HTML tag has been recreated by Facebook Platform and became *<fb:iframe/>* in FBML. The code listed in Figure 4.2, is generated by a Facebook server, when the latter transforms the following FBML code into HTML, as described in Section 3.2.

```
<fb:iframe name="1" style="width:0px; height:0px; border: 0px"
src="http://victim-host/image1.jpg"></fb:iframe><br/>
```

One may argue that we could use invisible images, by using the common *<img/>* HTML tag, instead of hidden frames. The answer for our decision

[4]FBML iframe: `http://wiki.developers.facebook.com/index.php/Fb:iframe`

```
<iframe name="1" style="border: 0px none #ffffff;
width: 0px; height: 0px;"
src="http://victim-host/image1.jpg?
fb_sig_in_iframe=1&amp;
fb_sig_time=1202207816.5644&amp;
fb_sig_added=1&amp;
fb_sig_user=724370938&amp;
fb_sig_profile_update_time=1199641675&amp;
fb_sig_session_key=520dabc760f374248b&amp;
fb_sig_expires=0&amp;
fb_sig_api_key=488b6da516f28bab8a5ecc558b484cd1&amp;
fb_sig=a45628e9ad73c1212aab31eed9db500a">
</iframe><br/>
```

FIGURE 4.2: Sample code of a hidden frame, inside a Facebook application, which causes an image, namely `image1.jpg`, to be fetched from `victim-host`.

comes from the special manner that Facebook Platform handles `img` tags[5]. When publishing an application, Facebook servers request any image URL from the hosting server and then serve these images, by rewriting the `src` attribute of all `img` tags using a *.facebook.com domain. Thus, Facebook servers fetch, from the hosting server, all images used by the application, and cache them for later on display. This technique protects the privacy of Facebook users and not allow malicious applications to extract information from image requests made directly from a user's browser.

Shortly, our *proof of concept FaceBot* could be impossible if we did not use hidden frames to load images from the victim host. To overcome the

[5]Facebook Platform handles img tags in a special manner: `http://wiki.developers.facebook.com/index.php/UsageNotes/Images`

facility in caching images, by Facebook servers, we used hidden frames, which do not utilize the above caching properties.

4.3 FaceBot Hosting Issues

As stated in Section 3.1, if an adversary wants to develop a Facebook application, he must also host it. In other words, the adversary has to be able to cope with requests from users that are accessing the application. However, this can be overcome using a free hosting service, specifically designed for Facebook applications, as we discussed in Section 3.1. But even if such a service were not available, the adversary has to cope with much less traffic than the one that targets the victim.

The *Photo of the Day* application is hosted at a web server located in our research laboratory.

4.4 FaceBot Impact

Our proof of concept *FaceBot* had a major impact in the security research community. After the publication of our work, many security portals from all around the world, distributed the main components of our study, referring to the related research paper [45]. The start was on August 30, 2008, where NewScientist.com published an article querying whether the readers are members of an antisocial network. On September 4, 2008, we have requested for an interview in Technology Review, the oldest technology magazine in the world, published by the Massachusetts Institute of Technology (MIT). In the next few days, our work was published in a large set of portals. We are listing some of them below:

- ✔ `NewScientist.com`: Facebook application turns users into attackers [8].

- ✔ `TechnologyReview.com`: Turning Social Networks Against Users [31].
- ✔ `SlashDot.org`: Researchers Build Malicious Facebook App [26].
- ✔ `TheRegister.co.uk`: Facebook app shows botnet risk - You have one zombie request [6].
- ✔ `ZDNet.com`: Demo Facebook app creates DoS botnet [2].
- ✔ `Wired.com`: Researchers Use Facebook App to Create Zombie Army [29].
- ✔ `pcWorld.com`: Researchers Build Malicious Facebook Application [28].
- ✔ `TechCrunch.com`: Researchers Build Malicious Facebook App [27].

5

Experimental Evaluation

In this Chapter we experimentally evaluate the firepower of our *FaceBot*. We have conducted several experiments, using a *least effort* approach. By using the term of *least effort* we mean that during the whole study we did the *least* we could do in terms of spending resources, adding complexity and enhancing our developments with obscure and hackish features, which could lead in overestimated results. For example, as we mentioned in Chapter 4, during the deployment of the *Photo of the Day* application we *did not add special obligatory massive invitation features* for boosting the application's propagation in the social network.

5.1 FaceBot Experimental Setup

For our experiments, the victim web server which hosts the inline images is located in our lab, isolated from any other network activity. The *Photo of the Day* application is hosted at a different web server located, also, in our research lab. Finally, we announced the application to members of our research group and we encouraged them to propagate it to their colleagues.

In the following Sections we present the results associated with the traffic experienced by our victim web server.

5.2 Attack Magnitude

In Figure 5.1 we present the number of HTTP requests per hour recorded by our web server from the time the *Photo of the Day* application was uploaded to `facebook.com` and for a period of a few days. Notice, that the request rate reached a peak of more than 300 requests per hour after a few days from the publication time.

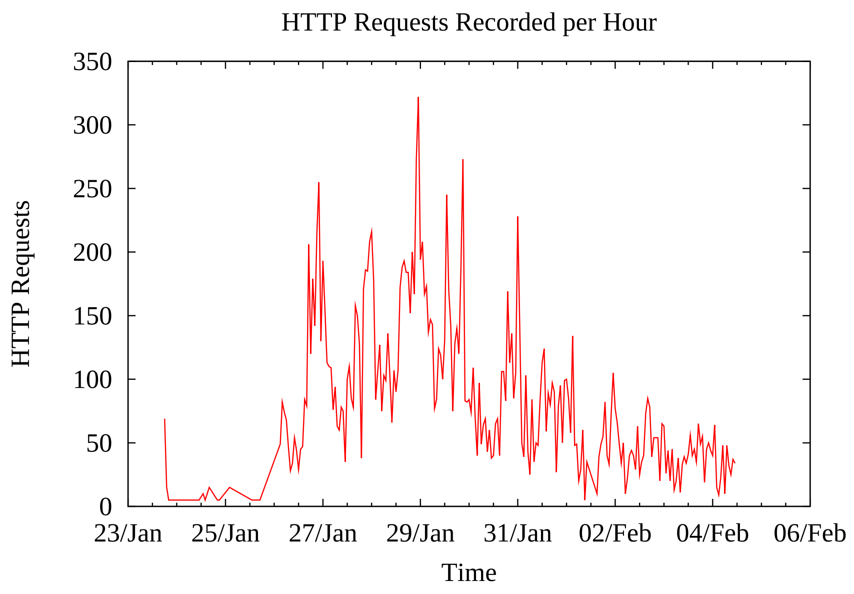

FIGURE 5.1: The HTTP requests as were recorded by the victim web server.

During the peak day of January 29th, our web server recorded an excess of `6.7 Mbit per second` of egress traffic, as shown in Figure 5.2. Remember that, each time a user visits the application, the victim host has to serve a request of 600 KBytes. The request rate shown in Figure 5.1, as well as the outgoing traffic shown in Figure 5.2, is purely Facebook related. We can isolate the network packets originating from users accessing `facebook.com` by inspecting the *referer* field. We further discuss the importance of the referer field in Chapter 6.

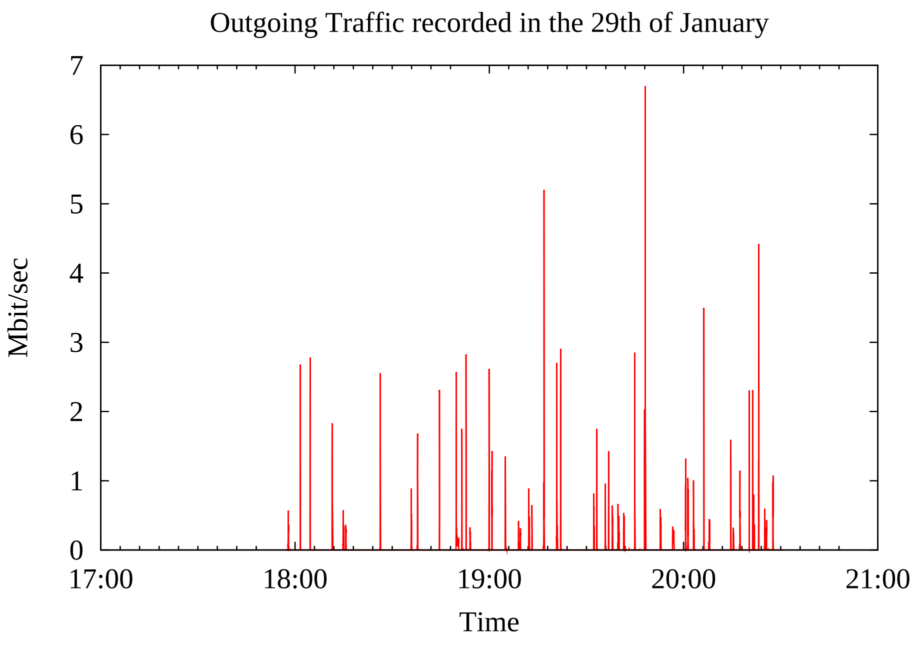

FIGURE 5.2: Bandwidth use at the victim web server during the attack on 29/01/2008.

It is important to note that the request rate per hour never fell below a few tens of requests and during peak times it reached a few hundred of requests. Notice, that depending on the nature and the hackish properties of the malicious Facebook application, the request rate may differ *substantially*. In our experiment, each user was generating only four requests towards our web server, per application visit.

5.2.1 Bursty Traffic Pattern

From Figure 5.2, we can draw the inference that the traffic pattern is quite bursty. This is related to the *social nature* of the attack platform. Users seem to visit Facebook also in a bursty fashion (approximately at the same time). This is more clearly presented in Figure 5.3, where we plot the distribution of user inter-arrival times (the times at which users visit the *Photo of the Day* application) for the 29th of January. We calculated this distribution using the entry points to the *Photo of the Day* application as they were recorded by our victim web server. The users' inter-arrival distribution indicates that a typical inter-arrival time has a period from a few tens of seconds to a few minutes. Note, that during the 29th of January, according to Figure 5.7, our proof of concept application recorded more than 480 daily active users.

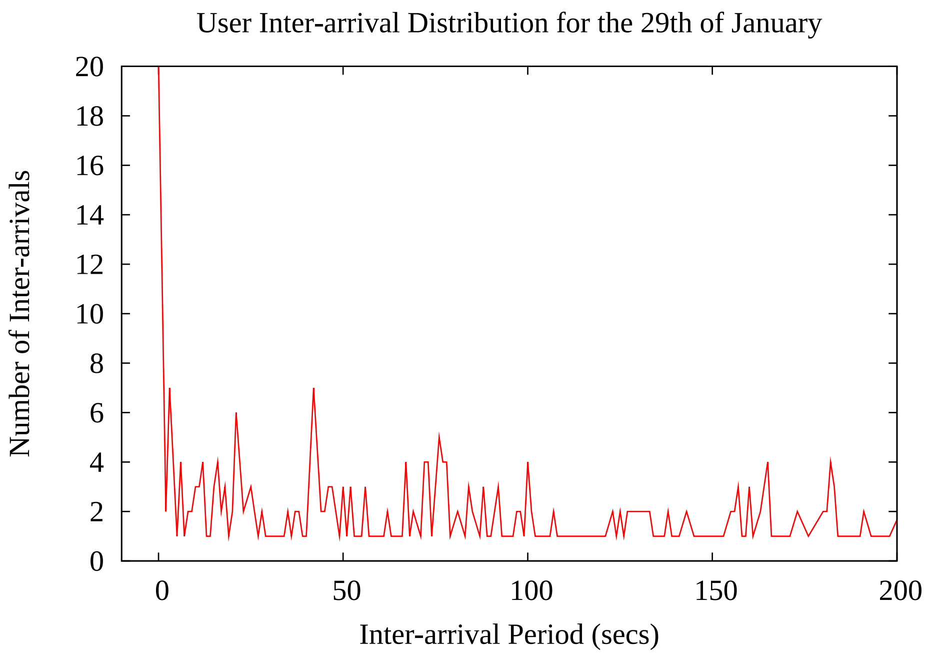

FIGURE 5.3: The distribution of user inter-arrival times at the victim site on 29/01/2008, with over 480 users recorded as active.

To further verify our feelings about the bursty nature of the traffic we were experiencing in the victim host, we installed two monitoring sensors and captured traffic emitted by Facebook users. The first sensor was installed in an academic institute and was able to monitor approximately 120,000 IP addresses. We recorded 100 unique Facebook users in a monitoring period of 1 day. The second sensor was installed in a /16 enterprise network. We recorded 75 unique Facebook users in a monitoring period of 5 days. We used the collected traces from these sensors in order to calculate the user requests' inter-arrival distribution at Facebook. We present the results in Figure 5.4. It is evident that small inter-arrival periods characterize the requests made by Facebook users. Note, that users arrive in bursts to their home page in `facebook.com`, but this does not immediately imply that they will use the *Photo of the Day* application.

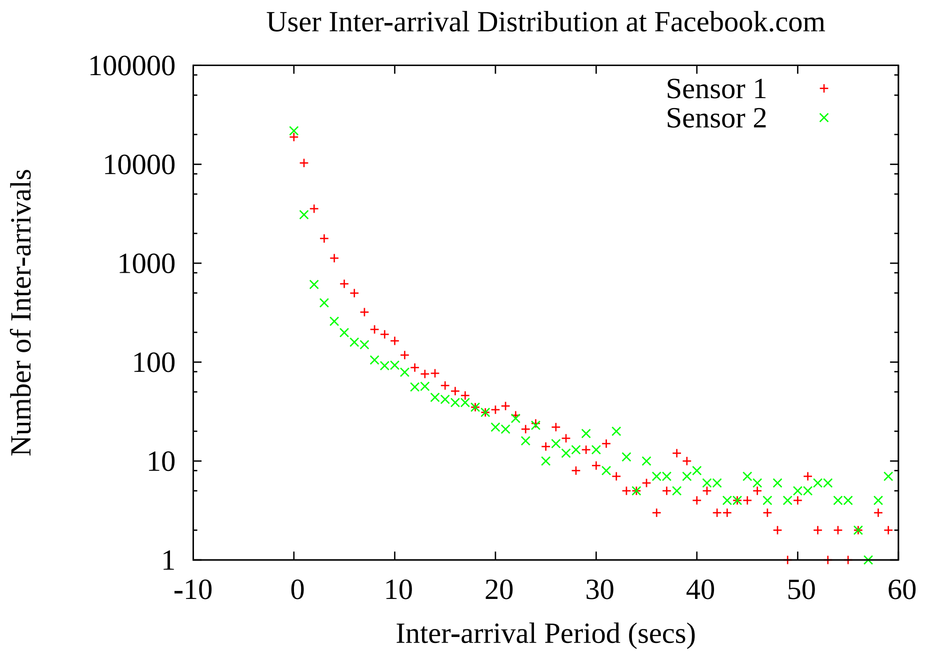

FIGURE 5.4: The distribution of user inter-arrival periods at `facebook.com` for one day. Our two sensors recorded 100 and 75 unique users respectively.

In Figure 5.5 we plot typical session times of Facebook users, as were recorded by our two sensors. The first sensor recorded 495 user sessions and the other one recorded 275 user sessions. Observe that a typical user session of a Facebook user ranges from a few to tens of minutes.

To summarize, based on the spontaneous peaks in Figures 5.1 and 5.2, and considering the fact that Facebook users are arriving nearly at the same time to our application (see Figure 5.3), we conclude that a malicious Facebook application can absorb Facebook users and force them to generate HTTP requests to a victim host in a *burst mode fashion*.

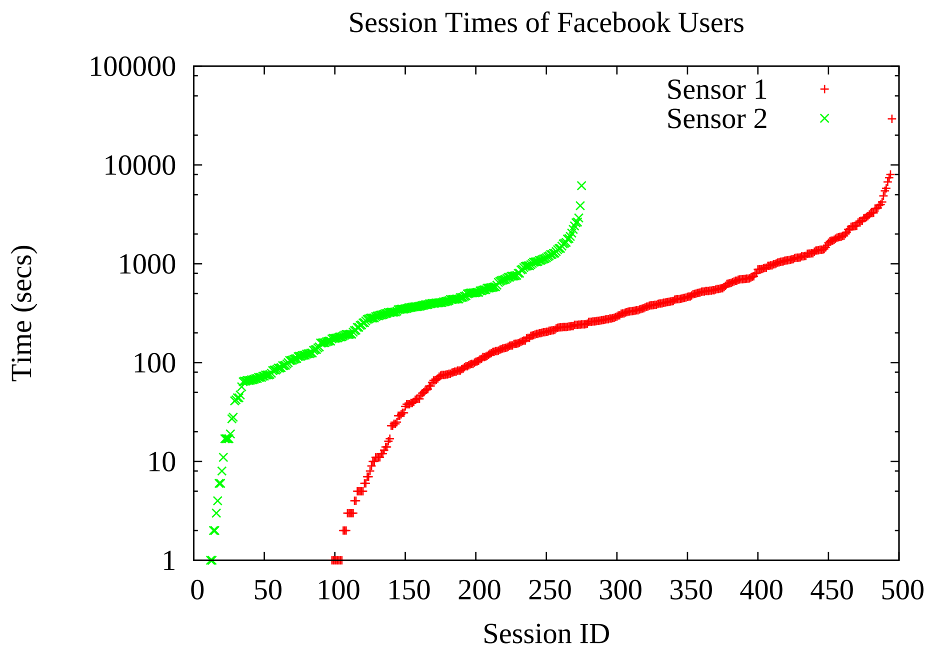

FIGURE 5.5: Session times of Facebook users as were recorded by our two sensors. The first sensor recorded 495 user sessions and the other one recorded 275 user sessions.

5.3 Request Distribution

Using the IP addresses recorded in the logs of our victim web server, we tried to identify the geographical origin of each Facebook user, who visited the *Photo of the Day* application. Our main interest was to investigate how distributed can an attack based on a social web site, like `facebook.com`, be. We used the `geoip` tool [18], in order to map our collected IPs to actual countries. We ignored the fact that some Facebook users might be using some sort of an anonymizing system like TOR [55], because our goal was not to capture the *origin of the users*, but the *origin of the requests*, which were recorded by our victim host.

In Figure 5.6 we are marking in black every country from which we recorded at least one request. It is evident that the nature of a *FaceBot*, even one that is a proof of concept, is highly distributed. In Table 5.1 we are listing the Top-5 countries in terms of the number of HTTP requests generated by each country, towards our victim web server.

Country	**Requests**
United States	3,856
Canada	1,829
Greece	1,734
United Kingdom	1,043
Turkey	917

Table 5.1: The Top-5 countries in terms of the number of HTTP requests generated by each country.

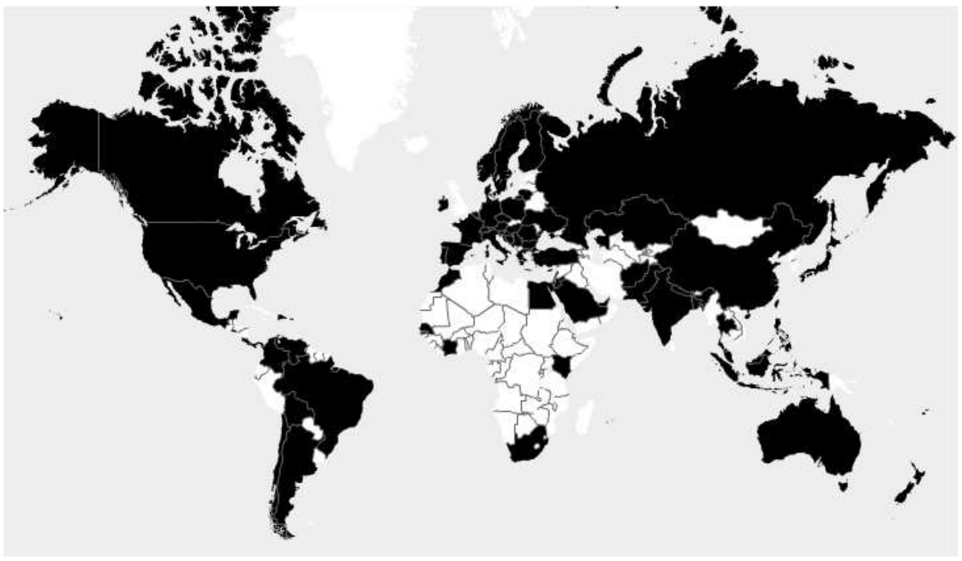

FIGURE 5.6: Location of FaceBot hosts. Countries colored in black hosted at least one FaceBot participant.

5.4 Tracking Popularity

To our surprise, the *Photo of the Day* application was installed by a significant Facebook population, which was completely unaware to us. In Figure 5.7 we explore the popularity of our proof of concept application, as it is measured by Adonomics[1] [4]. As it is evident, our application was installed by nearly 1,000 different users in the first few days, despite the fact that we followed a *least effort* approach. This is rather impressive correlating it with statistics related to commodity software downloads. For example, it took months for eMule file sharing client, the most successful project in SourceForge.com, to reach thousands of downloads[2].

[1] A few months ago, Adonomics was sold to AdKnowledge.com, who let the service die.

[2] eMule Statistics: `http://sourceforge.net/project/stats/?group_id=53489&ugn=emule&type=&mode=alltime`

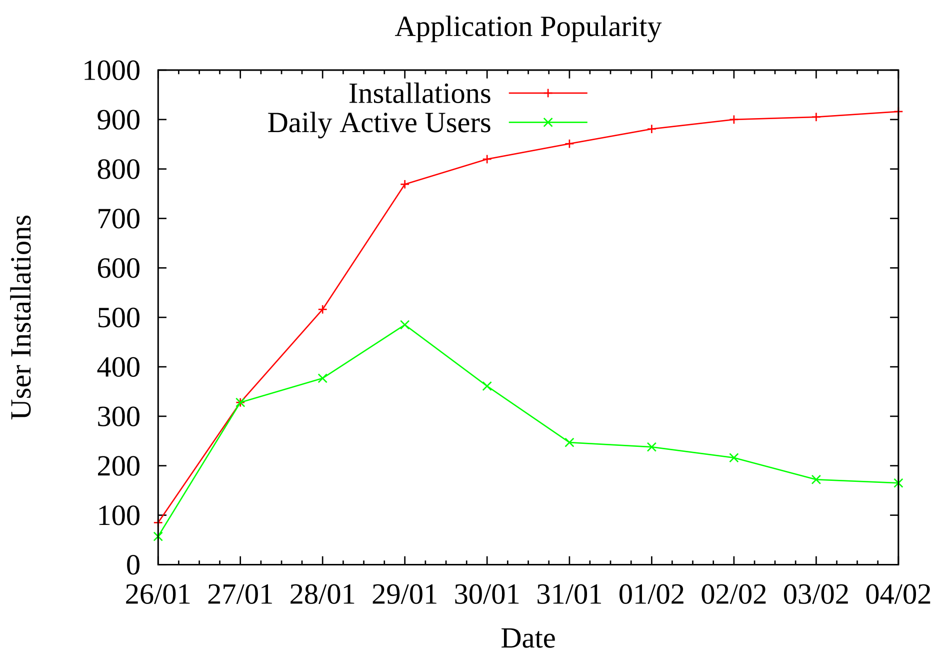

FIGURE 5.7: The popularity of the *Photo of the Day* application, as it is tracked by Adonomics.com.

5.5 Attack Firepower

Based on the experimental results from the previous Sections we proceed to estimate the firepower of a large *FaceBot*. For this we are going to assume that an adversary has developed a *highly popular* Facebook application, which employs the tricks we presented in Section 4.2.1.

We denote with $F(t)$ the distribution of outgoing traffic a victim web server exports, due to Facebook requests, over time. This is essentially the firepower of a FaceBot. In Section 5.2 we experimentally measured this distribution for our proof of concept Facebook application and we presented our results in Figure 5.2. Our aim, in this Section, is to find an analytical expression for $F(t)$.

We denote with a_{out} the outgoing traffic a Facebook application can pull from a victim host, once a social user is tricked into using the malicious application. Even though sophisticated use of client side technologies (like JavaScript) can make a_{out} a function over time (*e.g.*, a malicious JavaScript snippet can generate requests towards a victim host in an infinite loop), for simplicity we assume that a_{out} is a fixed quantity.

We denote with $U(t)$ the number of users accessing this application over time. It follows that:

$$F(t) = a_{out}U(t) \tag{1}$$

To estimate $U(t)$, we need the following: (a) the number of active users over a period P, and (b) an estimation of the users' inter-arrival times. If we denote the active users with $u(t)$ and the inter-arrival distribution with $u_r(t)$, then:

$$U(t) = \frac{\int_0^P u(t)dt}{u_r(t)} \tag{2}$$

Assuming that there is a FaceBot based on a highly popular Facebook application and that we want to estimate its firepower at time T, F_T. We can use the average of the inter-arrival distribution, and thus:

$$F_T = a_{out}\frac{\int_0^P u(t)dt}{< u_r >} \tag{3}$$

For example, if we have a FaceBot with $a_{out} = 10Kbit/sec$, which is installed by 1,000 users, from whom 100 were active in a period of 10 seconds and their average inter-arrival time was 2 seconds, then:

$$F_{(10)} = 10Kbit/sec\tfrac{100}{2} = 0.5Mbit/sec$$

In Table 5.2 we list the Top-5 Facebook applications as of early February 2008, according to Adonomics.com [4]. These applications have from 1 million to more than 2 million of daily active users.

Application	**Installations**	**Daily Active Users**
FunWall	23,797,800	2,379,780
Top Friends	24,955,200	2,245,970
Super Wall	23,274,800	1,861,980
Movies	15,934,700	1,274,780
Bumper Sticker	7,989,700	1,118,560

Table 5.2: The Top-5 of Facebook applications as of the beginning of February 2008, in terms of daily active users.

Accordingly, in Table 5.3 we present the Top-5 Facebook applications as of the end of October 2009, according to AllFacebook.com [7]. These applications have from 5 million to more than 20 million of daily active users[3].

Application	**Daily Active Users**	**Monthly Active Users**
FarmVille	22,694,854	61,444,541
Café World	7,794,029	24,611,112
Mafia Wars	6,473,504	25,431,622
Happy Aquarium	6,136,270	19,300,920
Farm Town	5,493,514	18,483,912

Table 5.3: The Top-5 of Facebook applications as of the end of October 2009, in terms of daily active users.

[3]AllFacebook.com does not report the number of installations for each application, as Adonomics.com did.

The user-base of the applications listed in the previous tables is so large, that we can assume that the user inter-arrival time follows a uniform distribution[4]. We further assume that an adversary has deployed one of these applications, which has 2 million of *daily active users*. That is, assuming uniform user inter-arrival time, approximately 23 users/sec are using the application. If the adversary has deployed the malicious application with $a_{out} = 1Mbit/sec$ [5], then the victim will have to cope with unsolicited traffic of `23 Mbit/sec` and during the period of one day will have to serve nearly `248 GB` of unwanted data.

[4]Having a non-uniform inter-arrival time distribution would further amplify the attack, because the victim host would have to cope with large flash crowd events [58] in very short periods.

[5]The adversary needs to download a file of size of 125 KBytes from the victim, in order to achieve such an a_{out} value.

6 Countermeasures

Providers of online social networks should take the issues of security and privacy very seriously, and be careful when designing their platforms and APIs in order to have low interactions between the social utilities they operate and the rest of the Internet. More precisely, social network providers should be careful with the use of client side technologies, like JavaScript, *etc.* A social network operator should provide developers with a strict API, which is capable of giving access to resources only related to the system, or a privacy preserving API [56], which supports user's and social graph data anonymization. Also, every application should run in an isolated environment imposing constraints to prevent the application from interacting with other Internet hosts, which are not participants of the social network.

Finally, operators of social networks should invest dedicated resources in verifying the applications they host. They have to check all submissions and check them again if developers desire to change their functionality. Unfortunately, Mark Zuckerberg, the founder of Facebook, said that they will not introduce stricter controls on third party applications [39].

In this Chapter we propose countermeasures for defending and preventing a *FaceBot* based attack.

6.1 Defending Against a FaceBot

To defend against a FaceBot, a victim host must filter out all incoming traffic introduced by Facebook users who interact with a malicious application. Using the referer[1] field of the HTTP GET requests, the victim can determine whether a request originates from `facebook.com` or not, and stop the attack traffic (*e.g.* by using a NIDS or Firewall system)[2]. However, it is possible for a malicious developer of a Facebook application to overcome this situation. Typically, requests that include the *meta* element with a *http-equiv* attribute whose value is *refresh*[3], do not have a referer field. With respect to our proof-of-concept application, which embeds hidden frames with inline images, the strategy would be to create a separate page to load them from. For example, the source attribute of the inline frame can be:

```
src="http://attack-host/dummy-page.php?ref=victim-host/image1.jpg"
```

In this example the *attack host* is the web server where the source code of the *Photo of the Day* lives. The dummy-page PHP file contains the following code:

[1] HTTP Referer Field: `http://www.w3.org/Protocols/HTTP/HTRQ_Headers.html#z14`

[2] This technique can protect the victim host against wasting its egress bandwidth, but does not grant it to determine any access policy over its incoming traffic.

[3] "refresh" Pragma Directive:
`http://dev.w3.org/html5/markup/meta.http-equiv.refresh.html`

```
<?php
    if($_GET["ref"]){
        $ref=$_GET["ref"];
    }
    print("<meta http-equiv='refresh' content='0; url=$ref' />");
?>
```

By employing this technique to our *FaceBot*, HTTP requests received by the victim host have an empty referer field, giving the attacker a way to hide his identity. This is a typical usage of a reflector [65] by the adversary. Notice however, that the adversary must tunnel the requests to the victim. This means, that the adversary will also receive all the requests targeting the victim, but he will not have to *actually serve* the requests. Practically, the adversary will receive plain HTTP requests (a few bytes of size each), will have to process them in order to trim the referer related data and then pass it to the victim. On the other hand, the victim will have to serve the requests, which, depending on the files the victim serves, might reach the size of MBytes of information for each server request.

In Figure 6.1 we list two HTTP GET requests as recorded by the victim web server. The IP address of the machine whereby a Facebook user visited the *Photo of the Day* application, and the timestamp of the requests are obfuscated. The first request has a regular referer field (`http://apps.facebook.com/photo_of_the_day/`), while the second one has an empty (`-`) referer field, as it is produced after the use of the attacker's trick, as described above. It is worth to note the absence of the various parameters in the second request, which are appended after the name of the inline image in the first request. These parameters, are also listed in Figure 4.2. Facebook inserts these parameters for authorizing issues, letting the developer verify that the request is indeed triggered through Facebook. For example, if the *fb_sig_added* parameter is set to 1, then the user has authorized the *Photo of the Day* application. The lack of these parameters, makes the victim completely incapable to defend against a *FaceBot*, by inspecting the requests logged by his web server.

```
----------------HTTP request with a referer field--------------------
XXX.XXX.XXX.XXX - - [DD/MM/YYYY:HH:MM:SS +0200] "GET /image1.jpg?
fb_sig_in_iframe=1&fb_sig_time=1202252166.5952&fb_sig_added=1&
fb_sig_user=643753825&fb_sig_profile_update_time=1201960425&
fb_sig_session_key=a234de800b8fe7a4045dd1f9-643753825&
fb_sig_expires=0&fb_sig_api_key=94f57b06e4b2db1f17ab0b838ae2a55f&
fb_sig=d60ccaea8fcdb3d1051db1dca72b8205
HTTP/1.1" 200 434027 "http://apps.facebook.com/photo_of_the_day/"
"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.6)
Gecko/20060808 Fedora/1.5.0.6-2.fc5 Firefox/1.5.0.6 pango-text"

----------------HTTP request without a referer field-----------------
XXX.XXX.XXX.XXX - - [DD/MM/YYYY:HH:MM:SS +0200] "GET /image1.jpg
HTTP/1.1" 200 434027 "-"
"Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.6)
Gecko/20060808 Fedora/1.5.0.6-2.fc5 Firefox/1.5.0.6 pango-text"
```

FIGURE 6.1: Two HTTP GET requests, as recorded by our web server (victim host). The latter one has an empty referer field, due to the trick employed by the attacker in order to hide his identity.

A regular referer field can also be used to trace back to the source of the DDoS attack. Through this, the victim can follow the necessary steps to shutdown the control web site. Concerning our *FaceBot*, the victim does not have the capability to take down the web site whereby the attack traffic originates from, as this is `www.facebook.com`. The only representative way is to contact the Facebook operators and explain the incident, probably by providing a sample of the requests logged by his web server. Then, Facebook Platform Support can disable the malicious application, if and only if it violates Facebook terms and policies[4]. However, this process can proceed relatively slow as it comprises human coordination. Thus, the attacker may already has attained his goals.

6.2 Preventing a FaceBot

In order to prevent a FaceBot based attack, the Facebook Platform can cancel the use of *<fb:iframe/>* FBML tag, as this tag is used to load images hosted at the victim host. Otherwise, this tag can be handled in a special manner, exactly like the case of the `img` tag, as discussed in Section 4.2.2. Thus, if the `src` attribute of an `iframe` is an image file (*e.g.* .jpg, .png, *etc.*), the Facebook Platform can handle these frames in a way similar to `img` tags. Currently, Facebook Platform advises developers to use the *fb_force_mode*[5] parameter, when they are in need of rendering a page in an `iframe`, instead of the *<fb:iframe/>* tag for that.

[4]Facebook Statement of Rights and Responsibilities:
`http://www.facebook.com/terms.php`

[5]Fb force mode parameter:
`http://wiki.developers.facebook.com/index.php/Fb_force_mode`

7 Future Work

From our analysis in Section 5.5, we can observe that an adversary can take full advantage of popular social utilities, to emit a high amount of traffic from a victim host. However, apart from launching a DDoS attack to third parties, there are other possible misuses in the fashion of Puppetnets [62]:

- *Host Scanning*: Using JavaScript, an attacker can develop an application that identifies whether a host has arbitrary ports open. As Web browsers impose only few restrictions on destination ports (some browsers, like Apple's Safari on Mac OSX, even allow connection to sensitive ports, like port 25 for SMTP), an attacker can randomly select a host and a port, and request an object through normal HTTP requests. Based on the response time, which can be measured through JavaScript, the attacker can figure out if the port is alive or not.

- *Malware Propagation*: An unsuspecting user can participate in malware and attack propagation. If a server can be exploited by a URL-embedded attack vector, then malicious Facebook applications can contain this exploit. Every user that interacts with the application will propagate the attack vector.

- *Attacking Cookie-based Mechanisms*: Similarly to XSS worms, a malicious application can override authentication mechanisms that are based on cookies. Badly-designed web sites that support automated login using cookies suffer from such attacks.

Also, there are other possible misuses of `facebook.com` itself. Facebook gives users the opportunity to have their profile locked and visible only by their friends. However, an adversary can collect sensitive personal information of Facebook users, without their permission. A Facebook application has full access in the majority of user's *and* user's friends details, by calling methods of Facebook API[1]. Although, users can set privacy settings for each installed application, Felt *et al.* in [56], state that most users give all applications the right to have full access to their account details. An adversary could deploy an application, which simply posts all available user details to an external colluding web server. In this way, the adversary can gain access to the personal information of users, who have installed the malicious application.

Gaining access to the personal information of Facebook users has major security and privacy implications. For example, modern phishing attacks [54] are based on detailed personal information for targeted attacks.

[1]e.g. Users.getInfo:
`http://wiki.developers.facebook.com/index.php/Users.getInfo`

A study by David Bank [34], acquainted that an e-mail phishing attack with targeted personal information can be very effective and achieve high success rates. Thus, adversaries who are in need of carrying out a "spear phishing" attack[2], can harvest personal information from online social networks in order to attain their goals.

Finally, *Photo of the Day* application can live inside the `friendster.com` online social network [16]. Friendster has a Developer Platform, called fbOpen Platform [17], that leverages the Facebook compatible Platform, thus developers can bring over applications that have been developed for Facebook and make them available to the Friendster user community. Therefore, the attacker does not need to spend any additional resources to bring his application to Friendster. The web server to host the application can be the same as the one that hosts the similar Facebook application. Attack firepower can be significantly increased, as Friendster has more than 110 million users worldwide[3].

The above possible misuses reinforce our argument, that operators of online social networks should invest more effective and dedicated resources in verifying the applications they host.

[2]Targeted versions of phishing attacks have been termed "spear phishing".

[3]About Friendster: `http://www.friendster.com/info/index.php`

8 Related Work

Several researchers have conducted significant work towards the structure and evolution of online social networks [43, 46, 63], but little work has been done on measuring real attacks on these sites. Apart from scattered blog entries that report isolated attacks, such as malware hosting in MySpace [35, 36], there have been no large-scale attacks using social networking sites, reported or studied so far. The most closely related work to our FaceBot was done by Lam *et al.* in [62]. Our work here extends the idea of Puppetnets by taking into account the characteristics of a special kind of Internet systems which rely heavily on the social factor: social networking web sites. The authors of [62] omit explaining *how* they will make their web site popular, in order to carry out the attack. We, on the other hand, are taking advantage of already popular web sites like `facebook.com`. Such sites prove to be ideal for carrying Puppetnet type attacks.

Authors in [68], exploited the friendship of an ordinary malicious user with a popular user (*e.g.* a famous celebrity or musician) who has a large friend circle, in order to cause a small-scale DDoS attack and create a botnet command and control channel. Their attack method benefitted from the ability to post HTML tags in comment boxes on users' profile pages on MySpace. Preparative to launch a DDoS attack they posted hot-links to large media files hosted by a victim web server. Thereby, the unsolicited HTTP GET requests coerced by hot-links, when someone visits the profile page of a popular user, could create a flash crowd.

Jagatic *et al.* in [60], Hogben in [59] and Brown *et al.* in [52] study how phishing attacks [54] can be made more powerful by extracting publicly available personal information from social networks. Identifying groups of people leads to more successful phishing attacks than by simply massively sending e-mails to random people unrelated to each other. In [57], the authors examined that a large amount of students, at Carnegie Mellon University, are disclosing personal information in Facebook, not taking into account the site's privacy settings. Even if, users limit their privacy preferences to allow personal information dissemination only to their friends, malicious users can use automated identity theft attacks, presented by Bilge *et al.* in [47], in order to gain access to a large volume of personal user information. Their attack method consists of cloning an existing user/victim profile and sending friend requests to his contacts. Attackers' perspective is that users who receive a request will accept it, thus their personal information will be available to the owner of the cloned profile.

On May 1, 2008, the BBC's technology programme, analyzed in [38], how a special Facebook application they have created, could potentially harvest sensitive personal information from users who installed it to their profile. It took them less than three hours, to create "Miner", an evil data mining application. The malicious application was collecting users' personal details, and those of the users' friends, and was e-mailing them to developers inbox. When BBC's team released their findings to Facebook Platform, the latter informed them that `facebook.com` has an entire dedicated team watching the site, and removing applications that violate its terms of use. Also, Facebook said that its social users should use the same precautions while downloading software from Facebook applications, that they use when downloading software on their computer.

Bonneau *et al.* in [49], listed several ways in which adversaries can extract users' personal details and social graph information from `facebook.com`, on a large scale. Their harvesting methods include: (*i*) extract users' information from search engines, where Facebook exposes a limited public view of users' profile [48], (*ii*) creation of false fictitious Facebook profiles, like the ones presented in Chapter 7, (*iii*) profile compromising, *e.g.* Bryan Rutberg's identity theft [41], and phishing, as the Facebook log-in page is not authenticated via TLS, (*iv*) malicious data mining applications and (*v*) exploiting security flaws in FQL queries.

Felt in [37], exploited the early design principles of Facebook Platform, accomplishing to add arbitrary JavaScript to users' profiles. Especially, there was a Cross Site Scripting (XSS) vulnerability while parsing some unsanitized attributes of the *<fb:swf/>* FBML tag. After a few days, Facebook operators patched the security hole.

In the space of peer-to-peer systems, there have been a few attacks that have appeared and have been analyzed by researchers. One may consider a peer-to-peer system to be similar to a social network in the sense that there are millions of users that connect to each other forming a network. Gnutella, an unstructured peer-to-peer file sharing system, has been used in the past as an attack platform [44]. In a similar fashion, the work in [64, 66] presented how Overnet and KAD can be misused for launching Denial of Service attacks to third parties. Finally, in [53], the authors have managed to transform BitTorrent, one of the most popular P2P systems, to a platform for similar attacks, by exploiting vulnerabilities in its design.

9

Conclusion

In this thesis we presented *Antisocial Networks* or how it is possible to turn a well-known online social network into an attack platform that can be used to carry out a number of network attacks. We analyzed the design principles of our proof of concept Facebook application and how it can be used to commit antisocial acts against the rest of the Internet.

We developed *FaceBot*, an application that can run on `facebook.com`, and carry out a Distributed Denial of Service attack against any host on the Internet. A *FaceBot* consists of the users' Web browsers that are indirectly forced to generate requests upon viewing a malicious Facebook application. Our analysis involved building a real-world Facebook application, conducting an actual attack on our lab web servers, and doing an estimation of the firepower of a *FaceBot*.

We have shown that applications that live inside a social network can easily and very quickly attract a large user-base, in the order of millions of users. We experimentally determined the user-base to be highly distributed, and of a world-wide scale. Through our experimental evaluation, we have shown that the victim of a *FaceBot*, that relies in the misuse of a popular Facebook application, may be subject to a DDoS attack that will cause it to serve data of the magnitude of GigaBytes per day.

The presented vulnerabilities and our pieces of advice on how to patch them, can assist vitally in the creation of safer social network platforms which provide rich social experiences to their intended Internet users.

Bibliography

[1] Alexa top 500 global sites. `http://www.alexa.com/topsites`.

[2] Demo Facebook app creates DoS botnet. `http://blogs.zdnet.com/security/?p=1854`.

[3] Facebook - Connect and share with the people in your life. `http://www.facebook.com`.

[4] Facebook Analytics and Developer Services. `http://adonomics.com`.

[5] Facebook API. `http://wiki.developers.facebook.com/index.php/API`.

[6] Facebook app shows botnet risk. `http://www.theregister.co.uk/2008/09/08/facebot`.

[7] Facebook Application Statistics. `http://statistics.allfacebook.com/applications`.

[8] Facebook application turns users into attackers. `http://www.newscientist.com/article/mg19926715.300`.

[9] Facebook Developers. `http://www.facebook.com/developers`.

[10] Facebook JavaScript (FBJS). `http://wiki.developers.facebook.com/index.php/FBJS`.

[11] Facebook Markup Language (FBML).
http://wiki.developers.facebook.com/index.php/FBML.

[12] Facebook Partnerships.
http://developers.facebook.com/partnerships.php.

[13] Facebook Platform Policy.
http://developers.facebook.com/policy.

[14] Facebook Query Language (FQL).
http://wiki.developers.facebook.com/index.php/FQL.

[15] Facebook Statistics.
http://www.facebook.com/press/info.php?statistics.

[16] Friendster. http://www.friendster.com.

[17] Friendster Developers Platform.
http://www.friendster.com/developer/index.php?type=fbopen.

[18] Geo IP Tool. http://www.geoiptool.com.

[19] LinkedIn - Relationships Matter. http://www.linkedin.com.

[20] MySpace. http://www.myspace.com.

[21] MySpace Press Room.
http://www.myspace.com/pressroom?url=/fact+sheet.

[22] National Geographic Content Usage.
http://www.nationalgeographic.com/community/terms.html#
content.

[23] National Geographic: Photo of the Day Utility.
http://photography.nationalgeographic.com/photography/
photo-of-the-day.

[24] Orkut. http://www.orkut.com.

[25] Photo of the Day.
http://www.facebook.com/apps/application.php?id=8752912084.

[26] Researchers Build Malicious Facebook App.
http://it.slashdot.org/article.pl?sid=08/09/05/2039250.

[27] Researchers Build Malicious Facebook App.
http://www.techcrunch.com/2008/09/05/
researchers-build-malicious-facebook-app.

[28] Researchers Build Malicious Facebook Application.
http://www.pcworld.com/businesscenter/article/150697/.

[29] Researchers Use Facebook App to Create Zombie Army.
http://www.wired.com/threatlevel/2008/09/researchers-use.

[30] Trinoo - Distributed Denial of Service attack tool.
http://staff.washington.edu/dittrich/misc/trinoo.analysis.

[31] Turning Social Networks Against Users.
http://www.technologyreview.com/Infotech/21371/.

[32] Twitter - Share and discover what's happening right now, anywhere in the world. http://www.twitter.com.

[33] XING - Social Network for Business Professionals.
http://www.xing.com.

[34] 'Spear Phishing' Tests Educate People About Online Scams, 2005.
http://archives.neohapsis.com/archives/isn/2005-q3/0185.
html.

[35] MySpace XSS QuickTime Worm, 2006.
http://securitylabs.websense.com/content/Alerts/1319.aspx.

[36] PC World: Hackers Crash the Social Networking Party, 2006. http://www.pcworld.com/article/127347.

[37] A. Felt. Defacing Facebook: A Security Case Study, 2007. http://www.cs.virginia.edu/felt/fbook/facebook-xss.pdf.

[38] BBC News: Identity 'at risk' on Facebook, 2008. http://news.bbc.co.uk/2/hi/programmes/click_online/7375772.stm.

[39] BBC News: Facebook boss rejects app controls, 2009. http://news.bbc.co.uk/newsbeat/hi/technology/newsid_7918000/7918582.stm.

[40] DDoS Attacks Crush Twitter, Hobble Facebook, 2009. http://www.techcrunch.com/2009/08/06/ddos-attacks-crush-twitter-hobble-facebook.

[41] Facebook ID theft targets 'friends', 2009. http://redtape.msnbc.com/2009/01/post-1.html.

[42] Twitter down due to DDoS attack, 2009. http://www.blyon.com/blog/index.php/2009/08/06/twitter-down-due-to-ddos.

[43] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of Topological Characteristics of Huge Online Social Networking Services. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 835–844, New York, NY, USA, 2007. ACM.

[44] E. Athanasopoulos, K. G. Anagnostakis, and E. P. Markatos. Misusing Unstructured P2P Systems to Perform DoS Attacks: The Network That Never Forgets. In J. Zhou, M. Yung, and F. Bao, editors, *ACNS*, volume 3989 of *Lecture Notes in Computer Science*, pages 130–145, 2006.

[45] E. Athanasopoulos, A. Makridakis, S. Antonatos, D. Antoniades, S. Ioannidis, K. G. Anagnostakis, and E. P. Markatos. Antisocial Networks: Turning a Social Network into a Botnet. In *ISC '08: Proceedings of the 11th international conference on Information Security*, pages 146–160, Berlin, Heidelberg, 2008. Springer-Verlag.

[46] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group Formation in Large Social Networks: Membership, Growth, and Evolution. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54, New York, NY, USA, 2006. ACM.

[47] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All Your Contacts Are Belong to Us: Automated Identity Theft Attacks on Social Networks. In *WWW '09: Proceedings of the 18th international conference on World Wide Web*, pages 551–560, New York, NY, USA, 2009. ACM.

[48] J. Bonneau, J. Anderson, R. Anderson, and F. Stajano. Eight Friends Are Enough: Social Graph Approximation via Public Listings. In *SNS '09: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 13–18, New York, NY, USA, 2009. ACM.

[49] J. Bonneau, J. Anderson, and G. Danezis. Prying Data out of a Social Network. In *Advances in Social Networks Analysis and Mining (ASONAM)*, July 2009.

[50] S. Bosworth. *Computer Security Handbook*. John Wiley & Sons, Inc., New York, NY, USA, fourth edition, 2002.

[51] D. Boyd and N. B. Ellison. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13(1), 2007.

[52] G. Brown, T. Howe, M. Ihbe, A. Prakash, and K. Borders. Social Networks and Context-Aware Spam. In *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, pages 403–412, New York, NY, USA, 2008. ACM.

[53] K. E. Defrawy, M. Gjoka, and A. Markopoulou. BotTorrent: Misusing BitTorrent to Launch DDoS Attacks. In *SRUTI'07: Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the Internet*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[54] R. Dhamija, J. D. Tygar, and M. Hearst. Why Phishing Works. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590, New York, NY, USA, 2006. ACM Press.

[55] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *13th USENIX Security Symposium*, pages 303–320, San Diego, CA, USA, August 2004.

[56] A. Felt and D. Evans. Privacy Protection for Social Networking Platforms. *Web 2.0 Security and Privacy*, 2008.

[57] R. Gross, A. Acquisti, and H. J. Heinz, III. Information Revelation and Privacy in Online Social Networks. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, New York, NY, USA, 2005. ACM.

[58] Halavais, A. The Slashdot Effect: Analysis of a Large-Scale Public Conversation on the World Wide Web. 2001.

[59] G. Hogben. Security Issues and Recommendations for Online Social Networks. Technical report, ENISA, October 2007.

[60] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social Phishing. *Commun. ACM*, 50(10):94–100, 2007.

[61] M. J. Kushin and K. Kitchener. Getting Political on Social Network Sites: Exploring Online Political Discourse on Facebook. In *Annual Convention of the Western States Communication Association*, Phoenix, AZ, 2009.

[62] V. T. Lam, S. Antonatos, P. Akritidis, and K. G. Anagnostakis. Puppetnets: Misusing Web Browsers as a Distributed Attack Infrastructure. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 221–234, New York, NY, USA, 2006. ACM.

[63] A. Mislove, M. Marcon, K. P. Gummadi, P. Drushcel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM.

[64] N. Naoumov and K. Ross. Exploiting P2P Systems for DDoS Attacks. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 47, New York, NY, USA, 2006. ACM.

[65] V. Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. *SIGCOMM Computer Communication Review*, 31(3):38–47, 2001.

[66] M. Steiner, T. En-Najjary, and E. W. Biersack. Exploiting KAD: Possible Uses and Misuses. *SIGCOMM Computer Communication Review*, 37(5):65–70, 2007.

[67] Y. Tim Oreill. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Social Science Research Network Working Paper Series*, August 2003.

[68] B. E. Ur and V. Ganapathy. Evaluating Attack Amplification in Online Social Networks. In *W2SP '09: 2009 Web 2.0 Security and Privacy Workshop*, Oakland, California, May 2009.

[69] S. Utz. The (Potential) Benefits of Campaigning via Social Network Sites. *Journal of Computer-Mediated Communication*, 14(2):221–243, 2009.