

University of Nebraska - Lincoln

DigitalCommons@University of Nebraska - Lincoln

Computer Science and Engineering: Theses,
Dissertations, and Student Research

Computer Science and Engineering, Department of

12-1-2011

Location Cheating: A Security Challenge to Location-based Social Network Services

Mai Ren

University of Nebraska-Lincoln, ren1011@gmail.com

Follow this and additional works at: <http://digitalcommons.unl.edu/computerscidiss>

 Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Ren, Mai, "Location Cheating: A Security Challenge to Location-based Social Network Services" (2011). *Computer Science and Engineering: Theses, Dissertations, and Student Research*. 31.

<http://digitalcommons.unl.edu/computerscidiss/31>

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in Computer Science and Engineering: Theses, Dissertations, and Student Research by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

LOCATION CHEATING: A SECURITY CHALLENGE TO LOCATION-BASED
SOCIAL NETWORK SERVICES

by

Mai Ren

A THESIS

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Lisong Xu and Professor Wenbo He

Lincoln, Nebraska

December, 2011

LOCATION CHEATING: A SECURITY CHALLENGE TO LOCATION-BASED SOCIAL NETWORK SERVICES

Mai Ren, M. S.

University of Nebraska, 2011

Advisers: Lisong Xu and Wenbo He

Location-based mobile social network services such as Foursquare and Gowalla have grown exponentially over the past several years. These location-based services utilize the geographical position to enrich user experiences in a variety of contexts, including location-based searching and location-based mobile advertising. To attract more users, the location-based mobile social network services provide real-world rewards to the user, when a user checks in at a certain venue or location. This gives incentives for users to cheat on their locations.

In this thesis, we investigate the threat of location cheating attacks, find the root cause of the vulnerability, and outline the possible defending mechanisms. We use Foursquare as an example to introduce a novel location cheating attack, which can easily pass the current location verification mechanism (e.g., cheater code of Foursquare). We also crawl the Foursquare's website. By analyzing the crawled data, we show that automated large scale cheating is possible. Throughout this work, we aim to call attention to location cheating in mobile social network services and provide insights into the defending mechanisms.

COPYRIGHT

© 2011, Mai Ren

ACKNOWLEDGMENTS

First, I would like to express my appreciation to my advisor and committee chair, Professor Lisong Xu for his expertise, guidance and encouragement throughout the course of my master's study in UNL. I especially thank Dr. Lisong for his generous support during my difficult time before graduation.

I would also like to express my appreciation to my co-advisor, Professor Wenbo He for her expertise, guidance and encouragement throughout the course of my master's study. Her support has always been very generous in both time and research resources. This work would not have been done, or get published, without her guidance. Her technical and editorial advice and infinite patience were essential for the completion of this thesis. I feel privileged to have had the opportunity to study under her supervision.

I thank Professor Xue Liu for providing the initial inspiration, guidance and editorial advice about this work. This work would not have started without his broad vision and deep experience in research.

I thank Professor Mehmet Can Vuran for serving on my Master's Thesis defense committee. The comments and time given have greatly improved and clarified this work.

I thank all of the CSE Research Facilities, staff and friends, for their friendship and for all the memorable times at UNL. I also thank everyone at the University of New Mexico, for getting me started in graduate studies, and for their support for my transfer to UNL. I would also like to thank everyone who has helped me along the way.

I especially thank my wife, without her support none of this would have been possible.

Last, but not least, my deepest thanks go to my parents for their self-giving love and support throughout my life.

Contents

Contents	vi
List of Figures	viii
1 Introduction	1
1.1 Location-based Social Network Services	1
1.2 Contribution	3
1.3 Outline	3
2 Background	5
2.1 Business Model of LBS	5
2.2 Possible Location Cheating Scenarios	7
2.3 Cheater Code	8
3 Location Cheating Attack	10
3.1 Location Cheating against GPS Verification	10
3.2 Crawling Data from Foursquare's Website	15
3.3 Automated Cheating	19
3.4 Cheating with Venue Profile Analysis	22
4 Evaluation of Location Cheating on Foursquare	23

4.1	High Check-in Frequency in Recent Visitor List	23
4.2	Low Reward Rate	25
4.3	Suspicious Check-in Patterns	27
5	Possible Solutions against Location Cheating	30
5.1	Location Verification Techniques	30
5.2	Mitigating Threat from Location Cheating	32
6	Conclusions and Future Work	34
6.1	Conclusion	34
6.2	Future Work	34
6.2.1	Privacy Leakage	35
6.2.2	Defense of Location Cheating	35
	Bibliography	36
	A Multi-threading in Crawler	39
	B Screenshots	43

List of Figures

3.1	Illustration of location cheating	11
3.2	We checked into Fisherman’s Wharf Sign in San Francisco, got point reward, and maintained our mayor status.	15
3.3	Crawling architecture and the database to store crawled information from Foursquare.	18
3.4	Locations of Starbucks branches crawled from Foursquare’s website. . .	20
3.5	An illustration of location cheating check-ins along a virtual path in the city.	21
4.1	Recent check-ins vs. total check-ins: the average recent check-ins of the users who have a certain number of total check-ins.	24
4.2	Number of badges vs. number of check-ins: The average number of badges granted to users who have a certain number of total check-ins. .	25
4.3	Check-in locations of a suspected cheater.	27
4.4	Check-in locations of a “normal” user.	28
B.1	Page of a venue on Foursquare’s website, “Who’s been here” section reveals users location history, and has been removed.	44
B.2	The user page of our testing user.	45

B.3	Use Dalvik Debug Monitor to set the GPS of Android device emulator to Golden Gate Bridge.	46
B.4	The multi-threaded crawler we developed to crawl Foursquare's website.	47

Chapter 1

Introduction

1.1 Location-based Social Network Services

A recent surge of location-based services (LBS) led by Foursquare[1], Gowalla[2], GyPSii[3], Loopt[4], Brightkite[5] has attracted a great deal of attention. Take Foursquare as an example, it has become one of the top recommended applications for all smartphone platforms. As of August 2010, Foursquare had attracted 1.89 million users since its launch in March 2009, and it draws in more than 10,000 new members daily. Meanwhile, hundreds of other similar services have been set up to follow this growing trend.

To encourage the use of location-based social network services, the service providers offer virtual or real-world rewards to a user if he or she checks in at a certain *venue* (i.e., places like coffee shops, restaurants, shopping malls). Foursquare provides real-world rewards (e.g., a free cup of coffee from Starbucks), which gives users incentives to cheat on their location information so that they can check in at a venue far away from where they really are.

In this work, we use Foursquare as an example to investigate the vulnerability

in location-based social network services. The goal is to raise awareness of location cheating and suggest possible solutions to drive the success of the business model among service providers, registered venues, and users.

We first introduce a novel and practical attack on location cheating, where a user may claim he or she is at a certain location which is thousands of miles away from his/her actual location, thereby deceiving the service provider on location information. Though Foursquare has adopted the cheater code to stop location cheating, we show that an attack can easily pass the cheater code. This benefits attackers in the real world and can be more severe when combined with the analysis on venue (or location) profiles. In order to study Foursquare's vulnerability to location cheating, we also crawled Foursquare's website and used the crawling results to find suspicious cheaters on Foursquare.

We found that the root cause of the vulnerability to location cheating is the lack of proper location verification mechanisms. If a user explores the open source operating systems for smart phones (e.g., Android) to modify global-positioning-system-(GPS)-related application programming interfaces (APIs), the user is able to cheat on his/her location using falsified GPS information. Even if defending mechanisms like *cheater code* are deployed, the loosely regulated anti-cheating rules still leave space for location cheaters.

We would like to make the following clarifications about this work:

1. We have obtained consent from Foursquare to reveal the findings described in this thesis.
2. Part of this work has been published in the 31st International Conference on Distributed Computing Systems (ICDCS'11).

1.2 Contribution

The major contributions of our work are listed as follows:

1. We investigated the methods to cheating on Foursquare, the leading location-based social network. The methods may also apply to other similar LBSs.
2. We crawled data from Foursquare's website and showed how to use it to increase the effectiveness/damage of location cheating attacks.
3. We investigated current anti-cheating methods that Foursquare uses. Our investigation suggests that defending against location cheating requires improvement to location verification ability.
4. We outline the possible solutions to defend against location cheating. We suggest service providers take the following measures to prevent location cheating: (1) explore effective location verification technologies, and (2) limit profile crawling and analysis to mitigate the threat of location cheating.

We believe that this investigation on location cheating will have a great impact on mobile social network services, and it will be an active research topic with strong practical value.

1.3 Outline

The rest of this thesis is organized as follows. In chapter 2, we briefly describe the background of LBS and its associated business model, cheating scenarios and cheater code. In chapter 3, we introduce a basic location cheating attack, demonstrate how to automate the cheating, and optimize the benefits to the attacker through crawling and profile analysis. In chapter 4, we show the results from our

experiments on location cheating and examine the seriousness of current cheating threats. In chapter 5, we discuss possible solutions to prevent location cheating. In chapter 6, we provide our conclusions and discuss future work.

Chapter 2

Background

In this chapter, we provide background and describe current practices used by location-based mobile social network services.

2.1 Business Model of LBS

Location-based social networking services allow users to share their location-related information. Users can add comments about a restaurant, find out what's happening, let their friends know where they are, and meet friends nearby for a cup of coffee. To report the geolocation to a service provider (e.g., Foursquare[1]), a user needs to "check in" to the location/venue where the user is. The service provider may broadcast the user's location information to his/her friends or even the public. The check-in is done by hand, which means a user is able to determine if he/she wants to check in, thereby controlling their location privacy. Services like this are not new, but they all have lacked incentives for people to use them, until Foursquare introduced a new business model.

Foursquare uses a progressive reward mechanism to provide four types of

reward incentives to its users. Listed from the easiest to the hardest to obtain, they are: points, badges, mayorships, and real world rewards. The first three are virtual rewards: (1) *points* are provided for all valid check-ins (e.g., the first time to check in to a venue, check in to the same venue multiple times); (2) *badges* are awarded for specific achievements, such as “30 check-ins in a month” or “checked into 10 different venues”; (3) *mayorship* of a venue is granted to the user who checked in to that venue the most days in the past 60 days. Only the number of days with check-ins to this venue are counted, without consideration of how many check-ins occurred per day or the total number of check-ins. Unlike points and badges which are solely dependent on a user’s activities, the title of “Mayor” is given on a competitive basis. There is only one mayor for each venue. This will create vulnerability that if an attacker got the mayorship of this venue and kept checking in to it every day, no other user can get the mayorship from the attacker.

(4) Real-world rewards, like a free cup of coffee, are provided by businesses (e.g., restaurants or bars) that set up a partnership with Foursquare. We crawled the information for all venues (discussed in more details later) and found that more than 90% of the rewards were only for mayors. This setup provides benefits for both Foursquare and its partner businesses: On the one hand Foursquare does not need to pay for those real world rewards; On the other hand, the user’s desire for discounts and their competition for the mayorship will likely bring more users (customers) to the partner businesses, thereby increasing their profits. While the business model benefits multiple parties in the game, it makes Foursquare a lucrative target of attacks by location cheating.

2.2 Possible Location Cheating Scenarios

In the context of location-based social network services, a user may cheat on his/her location for various reasons. A user may want to get rewards from venues or impress others by claiming a false location. A business owner may use location cheating to check into a competing business, and badmouth that business by leaving negative comments.

Similar to most location-based social network services, Foursquare initially relied on users' self-regulation to maintain the authenticity of the check-ins. Hence, the check-ins to any place a user can find in the Foursquare *client application* (using the suggested list of nearby venues, searching for a venue by name, or browsing and locating the venue on the map) were valid. Software tools are available on the market that can automatically check people into their desired venues, e.g., "Autosquare" for Android. The basic cheating method worked in the early days of Foursquare. It is rather simple and obviously does not work now after the introduction of location verification mechanism, which requires location information to complete the check-in process. However, location cheaters can modify the location information and send false locations to the server.

The objective of the attacks is to automatically check into as many businesses as possible and as frequently as possible to maximize benefits through location cheating. A more sophisticated attack is automated cheating. To make automated cheating easier, the cheaters may use venue profile analysis to identify victims, which can be the venues who provide discounts or users who are aiming to get mayorships in specific venues. Hence, an attacker is able to select the venues where the "Mayor" title is less competitive and the rewards are more desirable or use a minimum number of check-ins to prevent another user from getting a

mayorship.

2.3 Cheater Code

Foursquare has adopted the *cheater code* to defend against location cheating attacks. One of its functions is to verify the location of a device by using the GPS function of that device. If a user claims that he/she is currently in a location far away from the location reported by the GPS of his/her phone, this check-in will be considered invalid and won't yield any rewards.

Apart from utilizing GPS for location verification, the *cheater code* also incorporates multiple rules which runs on Foursquare servers to determine if a user cheats on its location. The details of the *cheater code* are concealed from users. But we managed to detect a few rules, through experiments, that are important to maneuvering the location cheating to pass the scrutiny of the *cheater code*. A part of the criteria used in determining location cheating in the *cheater code* are listed as follows.

Frequent check-ins: We found a user cannot check in to the same venue again within one hour. This rule prevents a user from checking in frequently to get as many points as possible and keep his/her name always on top of the recent check-in list, making it more likely for people to contact the user for comments about the venue.

Super human speed: If a user continuously checks into locations that are located far away from each other, Foursquare will indicate that the user is moving at "super human speed" and refuse to give any reward for his/her check-ins. This rule limits location cheating by a single user to a small geographic area.

Rapid-fire check-ins: If a user checks into multiple venues that are located

within a 180 meters by 180 meters square area (which is well within a short walking distance such as in a mall) with a 1 minute interval, Foursquare issues a warning about “rapid-fire check-ins” on the fourth check-in. This rule stops a user from checking in to multiple venues in a small area and in a short time period.

These rules essentially limit the number of check-ins a user can perform daily, thus reducing the potential for automated cheating. Clearly identifying these rules helps attackers to design the best way to work around them.

Chapter 3

Location Cheating Attack

In this chapter, we outline three levels of attack: cheating via GPS, automated cheating, and cheating with the assistance of venue profile analysis. They will severely interrupt the operation of LBSs when combined together. We first introduce four location cheating methods which can pass the validation from Foursquare and other similar location-based social network services at least once. After that, we crawl data from Foursquare's website and evade Foursquare's *cheater code* to automate the cheating process. Finally, by analyzing the crawled data, we focus on a cheating attack on high valued targets such as those who provide real world rewards.

3.1 Location Cheating against GPS Verification

Location-based services like Foursquare use their *client applications* installed on their users' smartphones to get GPS location readings, since this happens completely on the client side, it is relatively easy to hack. We analyzed Foursquare's *client application* source code and confirmed that it gets the GPS location data

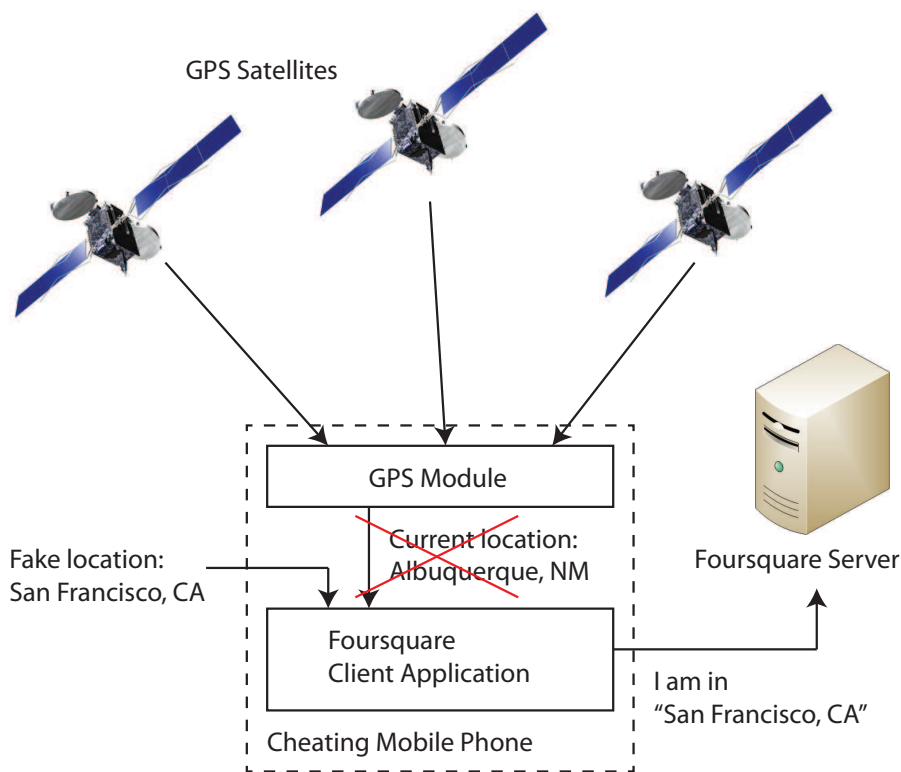


Figure 3.1: Illustration of location cheating.

from the phone's GPS-related APIs. Figure 3.1 shows the concept of such location cheating: Normally, the GPS module in a mobile phone will return the current location information to the LBS application, but an attacker blocks this and feeds fake location information to the LBS application so that it makes its server believe that this phone is really at the falsified location. The cheating check-in will then be approved.

There are at least four ways for an attacker to pass the GPS verification by providing Foursquare's *client application* with fake GPS coordinates:

1. *Via GPS APIs:*

This method modifies the GPS-related APIs in a smartphone's operating system to return fake GPS data. This is easy because the prevalence of

open source smartphone operating systems like Android. These APIs can be modified to get GPS locations from sources other than the phone's GPS module, for example, from a server that returns fake GPS coordinates, or simply from a local file. This method is limited to open source operating systems, because it is difficult to modify a closed source system like iOS from Apple; but since LBSs like Foursquare provide their *client applications* on multiple major smartphone platforms (Android, iPhone, Blackberry), this is a universal cheating method. A hack into Android is representative to cover cross-platformed LBSs.

2. *Via GPS module:*

Directly hacking into a smartphone's GPS module is another way of falsifying the location. There are two ways to do this: one via hardware and the other is via software. The former modifies the physical GPS hardware inside the phone, making it capable of faking data, so that the cheating is transparent to the mobile phone's operating system. The latter simulates a GPS device. For example, an attacker can write a program on a computer that simulates the behavior of a Bluetooth GPS receiver and let the phone connect to this simulated Bluetooth GPS receiver, enabling the simulated GPS to return fake coordinates. In fact, there are already a number of such tools on the market (e.g., Skylab GPS Simulator[6], Zyl Soft[7], GPS Generator Pro[8]), that were originally developed to help debug GPS-related software or gadgets.

3. *Via server APIs:*

Foursquare provides a set of application APIs that allow developers to create new applications for them, like an application for uploading GEO-tagged

photos. These APIs can be employed by a location cheater to check into a place. The drawback is that not all LBS service providers provide such public server side APIs. But this method is more convenient to issue a large-scale cheating attack.

4. *Via device emulator:*

Smartphone manufacturers (like Apple, Google, and Microsoft) provide device emulators to developers for easier debugging and testing. A device emulator is a full featured virtual machine of that device. One of the basic features of these device emulators is that they are configurable, including their simulated GPS module. Taking the Android device emulator for example, we can send it a specific command to set a location to the simulated GPS module. The GPS module of this emulator will return the coordinates we set to whichever applications that need GPS info. We conducted our experiments with this method, because this one is the easiest and most reliable when compared to the first three methods. Almost all potential attackers with a basic knowledge of mobile developing can master this method with no difficulty.

We chose Android emulator to conduct our experiments. By default, the emulator prohibits the android market so we have to hack the emulator first to get the Foursquare application installed. The reason the Android emulator has this limit is that Google only wants developers to test their own programs in the emulator. We bypassed this limitation by using a full system recovery image from a device manufacturer's website to run the Android Virtual Device. This will restore the emulator back to a full featured system with the Android Market, thus enabling us to install Foursquare or any other LBS applications on it.

We registered a user on Foursquare for testing purposes, and conducted all of our experiments in Albuquerque, New Mexico and Lincoln, Nebraska. Our goal was to check in to venues outside of the two states, so we knew the cheating method was working. We used the tool “Dalvik Debug Monitor”, which is part of the Android SDK to connect to the emulator and set GPS coordinates in it. We obtained the coordinates of the target venues by looking up Google Earth, which shows the exact coordinates of where the mouse is pointing on its map.

The entire cheating process can be described as: hack the emulator; install and run Foursquare application; find the coordinates of the target venue in Google Earth; use “Dalvik Debug Monitor” to set the coordinates in the emulator; find the target venue in the list of nearby venues in Foursquare application; and check into the target venue.

The results of our experiments showed that the check-ins to distant venues were all accepted, and we received rewards successfully. We got points for each of the check-ins, and we got badges like a normal user as well, i.e. after checking in to 10 different venues, we got the badge “Adventurer: You’ve checked into 10 different venues!”. We also tried to get a mayorship, we chose the venue “Fisherman’s Wharf Sign” in San Francisco which is a well-known tourist spot as the target venue, and we kept checking in to it once a day for 4 consecutive days. After 9 days, we had found our test user became the mayor of the venue. Figure 3.2 shows the result of a later check-in to “Fisherman’s Wharf Sign”. This experiment shows that the *device emulator method* works and can receive the rewards.



Figure 3.2: We checked into Fisherman's Wharf Sign in San Francisco, got point reward, and maintained our mayor status.

3.2 Crawling Data from Foursquare's Website

Getting the big picture of Foursquare users and venues is a great help for the location cheating attacks, although the crawling itself is not an attack. There are two types of information that we crawled: users' profiles and venues' profiles. In this section, we describe the crawling procedure, in which we only accessed Foursquare's public webpages. Wondracek, Holz, Kirda and Kruegel[9] introduced a similar crawling and attacking approach. We will also use the crawling results to show the evidence of existing location cheating attacks on Foursquare

and identify the suspicious location cheaters in next section.

To increase performance, we developed a multi-thread crawler to download and process a large amount of webpages (over 7 million). This architecture has proven to be highly effective, for example, Cho, J. and Garcia-Molina used a parallel crawler to increase performance[10], and Chau, Pandit, Wang, Faloutsos focus on crawling social networks with parallel crawling [11].

We wrote the crawler in C# and used MySQL as the database. We ran the crawler on 3 Windows PCs at the same time, each with a 2.0GHz Intel Core 2 Duo processor and 1GB RAM. The fourth computer with the same hardware specifications, but running Ubuntu 8.10 server operating system and functions served as a database server. In our design, we set 14 to 16 threads on each of the three crawling machines to crawl 100,000 users per hour for user profile crawling, and set 5 to 6 threads on each machine to crawl around 50,000 venues per hour for venue profile crawling.

In total, we crawled more than 1.89 million users and 5.6 million venues, which coincide with the Foursquare's reported number of users. This means we can update all user profiles in less than two days, or update all venue profiles in about 5 days. The crawling performance is an important design concern, because by repeatedly crawling data and comparing the differences between each set of crawling results, we can further investigate the behaviors of its users and extract more information. For example, the venue's recent visitor list does not have a time stamp to indicate when a user visited this venue; but if we crawl the venues daily, then we will be able to determine how frequently a user checks into a venue. We can further analyze the user behavior to show if the user is suspicious of location cheating.

Each user on Foursquare has a profile that contains personal information. A

user's profile provides information such as name, current location, number of check-ins, reward information, and a list of friends. A user's mayorships and check-in history are hidden from the public, since these two types of information may expose his/her location privacy. However, we can infer a user's mayorship information and partial check-in activities from venue profiles, which contain lists of their recent visitors and links to their mayors. In addition, a venue's profile also provides its name, address, location, number of users who checked in, unique visitors and tips.

To crawl these profiles, we need to know the URLs of these profile pages. We discovered that Foursquare uses incrementing numerical IDs to identify their users and venues. By changing the ID in the URL, we can crawl almost all of the user and venue profiles. We believe this is a serious security weakness and should be patched soon.

Two types of URLs can be used to access user profiles. The first one is with an internal user ID in URL, like "http://Foursquare.com/user/-1852791". To access another user with ID 23456, we just replace the "1852791" in the URL with "23456", and we can visit the public profile page with the new URL. We believe that we can access all users just by increasing or decreasing the user ID in the URL. We implemented a web crawler to do so, and we discovered around 1.89 million users in August 2010. Another type of URL contains the username, like "http://Foursquare.com/user/test", where "test" is the username of a user. Not every user has a username-based URL to the profile page. Out of 1.89 million users, only 26.1% have usernames, so we used the URL with ID in our crawling tool. For venue profiles, Foursquare only uses numbered IDs in the URL of the profile pages, like "http://Foursquare.com/venue/1235677".

After we had the URL of a profile page, we sent HTTP Get to this URL and got

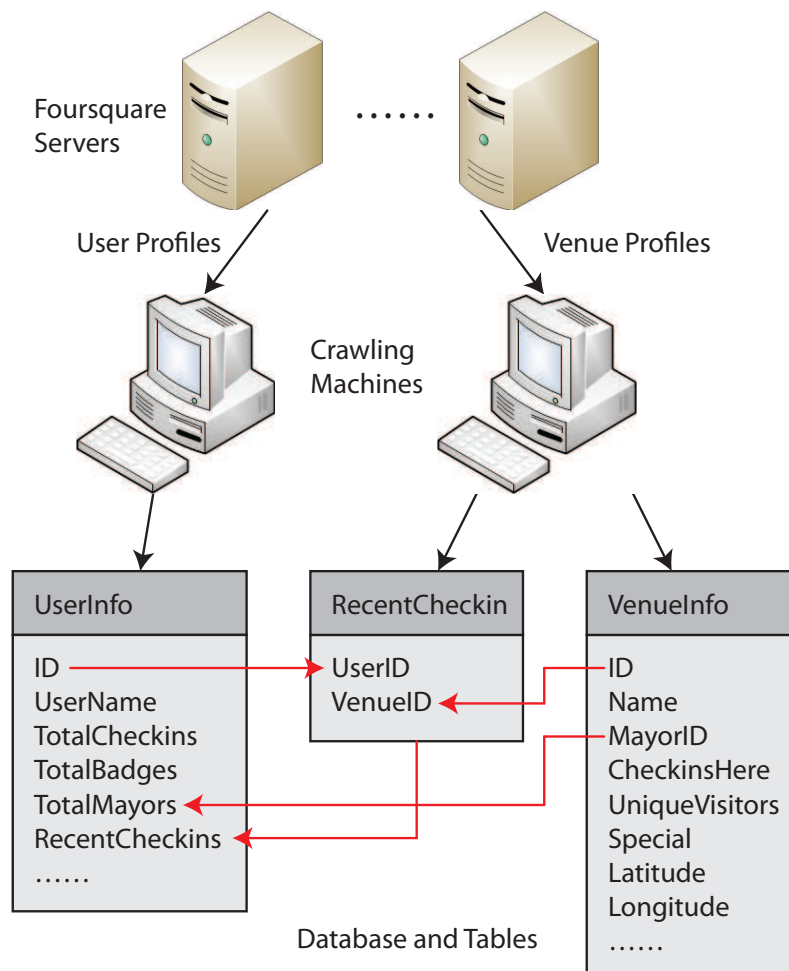


Figure 3.3: Crawling architecture and the database to store crawled information from Foursquare.

the HTML source code from the server's response. To extract data from the HTML source code, we let the crawler perform a set of regular expression matches. After extracting the data, we stored user and venue information in a database. Figure 3.3 shows the structure of the database; the arrows indicate the relationships between the tables. We stored user and venue profiles in tables *UserInfo* and *VenueInfo* respectively; and we also created a table called *RecentCheckins* to record the relations between venues and users. We put each venue's recent visitors in this ta-

ble; and by counting the number of records for a user, we recorded the number of recent check-ins of this user and stored it in *RecentCheckins* of *UserInfo*. Similarly, by analyzing the *MayorID* of each venue, we calculated how many mayorships each user had and put the result in *TotalMayors* of *UserInfo*.

3.3 Automated Cheating

To achieve significant benefits from location cheating, attackers need to be able to control a large number of users and make them check in automatically. This requires the location cheaters to (1) automatically find location coordinates of victim venues, and (2) automatically select a list of venues to check in to in order to pass the *cheater code*. We met the first requirement by crawling, and we could easily use SQL commands to get the location coordinates of the selected venues from the database.

Figure 3.4 shows the coordinates of all Starbucks branches in the US, where x axes and y axes are real coordinates. The location coordinates form the shape of the United States territory, because Starbucks' branches are distributed all over the US. We draw this map by SQL command:

```
SELECT Longitude, Latitude FROM VenueInfo WHERE Name LIKE "%Starbucks%".
```

Second, to pass the anti-cheating verification, the key is to avoid triggering any of the rules in the *cheater code*, since it detects cheating behavior on a per user basis, we focus on the strategy of a single user. An attacker needs to organize coordinates from the first step into a schedule, which states the sequence of venues to check into and the time interval between the check-ins; and the schedule must follow all rules from the *cheater code*. The attacker could create a tool to do this automatically.

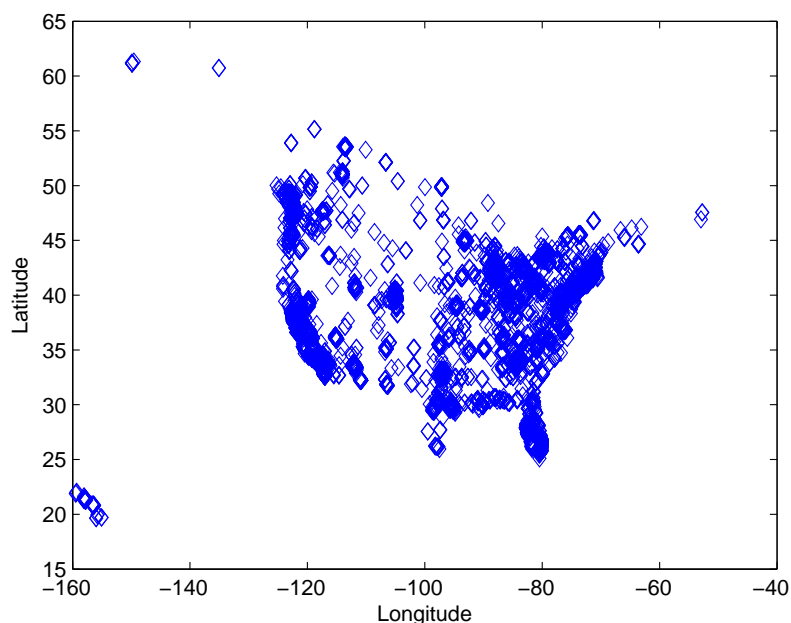


Figure 3.4: Locations of Starbucks branches crawled from Foursquare’s website.

To determine the sequence of venues in which to check in, an attacker can create a virtual user, compute a virtual path to visit the target venues using Google Map’s APIs, and build the check-in schedule along the virtual path. We also need to determine the time interval T between check-ins, which is determined by the distance between the check-ins in the schedule. Based on our experiments, we can check into venues less than 1 mile apart with a 5-minute interval without being detected as a cheater. So for distance D less than 1 mile, we should set T to 5 minutes, if $D > 1$ mile, we let $T = D * 5$ minutes.

In our proof of concepts experiment, we created a semiautomatic location cheating tool. With the tool, an attacker can use any venue as the starting point. The attacker can then set the next cheating location by setting the moving direction and distance, for example, “move 500 yards to the west”, the tool will search for the venue that is the closest to the target location and then automatically set

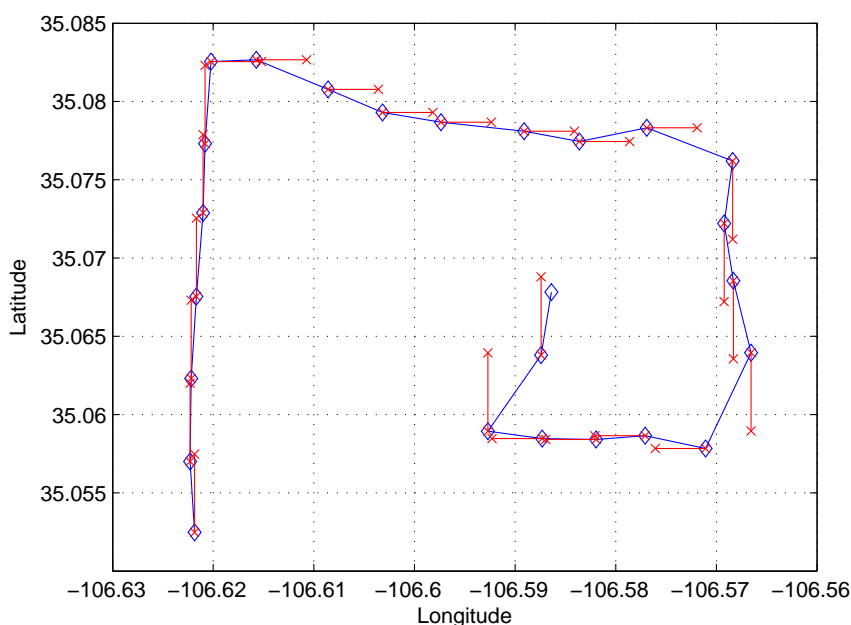


Figure 3.5: An illustration of location cheating check-ins along a virtual path in the city.

the coordinates of the venue found to the emulator or generate a list for fully automated check-ins later. The tool also automatically limits the process to avoid triggering any anti-cheating rules in the *cheater code* as we presented before.

Figure 3.5 shows the path of a virtual tour, the diamond points are the locations of venues the tool actually selected and checked into, and the cross-points and lines to them show the intended moving directions and target locations. The starting point in this tour is at the lower left point of Figure 3.5. We started by moving north and then kept turning right. The desired moving distance for each step was 0.005 degrees, either longitude or latitude, equivalent to about 550 meters in latitude direction or about 450 meters in longitude direction around this location. We set the interval between check-ins to 5 minutes since the moving distance is less than 1 mile. We continued checking into 25 venues without being

detected as a cheater, and we received reward points and badges accordingly.

As we can see in Figure 3.5, most of the time, the actual venues we checked into are not very far from the desired location, this is due to the high density of venues in the city. To move across large distances, we should increase the moving distance of each step, which will reduce the probability that we drift too far from the desired direction, like the second to last move in Figure 3.5.

3.4 Cheating with Venue Profile Analysis

Since brute-force check-ins increase the chance that a cheater is caught, a location cheater may gain intelligence from venue analyses after crawling. For example, an attacker may select the victim venues that provide special offers to their mayors and don't have a mayor yet (or are less competitive for mayorship) as targets. It is relatively easy to become the mayor of these venues. Amongst the venues we have crawled, around 1000 venues fall into this category.

Through profile analysis, we found a user on Foursquare is the mayor of 865 venues but with a total number of check-ins of only 1265. It is interesting to observe that most of the 865 venues have no other visitors during the past 60 days, so only one check-in is enough to get the mayorship. We also discovered some special offers that do not require mayorship which are much easier to obtain, it's difficult to find such information without crawling the venue profiles.

The attack can also target other users. For example, to stop a user from getting any mayorship, the attacker will analyze venue profiles and find venues that the victim user is mayor of or has been to. Then the attacker will apply an automated cheating attack on those venues in order to attack the mayorships of the victim.

Chapter 4

Evaluation of Location Cheating on Foursquare

We have demonstrated how to do location cheating attacks on Foursquare. Next, we will show a big picture of location cheating through our crawling and analysis. In this chapter, we examine the signs of location cheating on Foursquare. We found three identifying factors that are related to location cheating. They are: (1) above normal level of activity, (2) below normal level of rewards, and (3) suspicious check-in patterns.

4.1 High Check-in Frequency in Recent Visitor List

If a user checks in too frequently and at too many venues, it is suspicious, because it is unlikely the user visits so many places in a short amount of time.

We crawled the record of 20 million check-ins, and each of them represents a user visiting a venue once. That means, on average, each user on Foursquare has checked into at least ten venues, and a venue has had at least four visitors. The

actual number should be higher since only recent check-ins were shown on the website and were crawled.

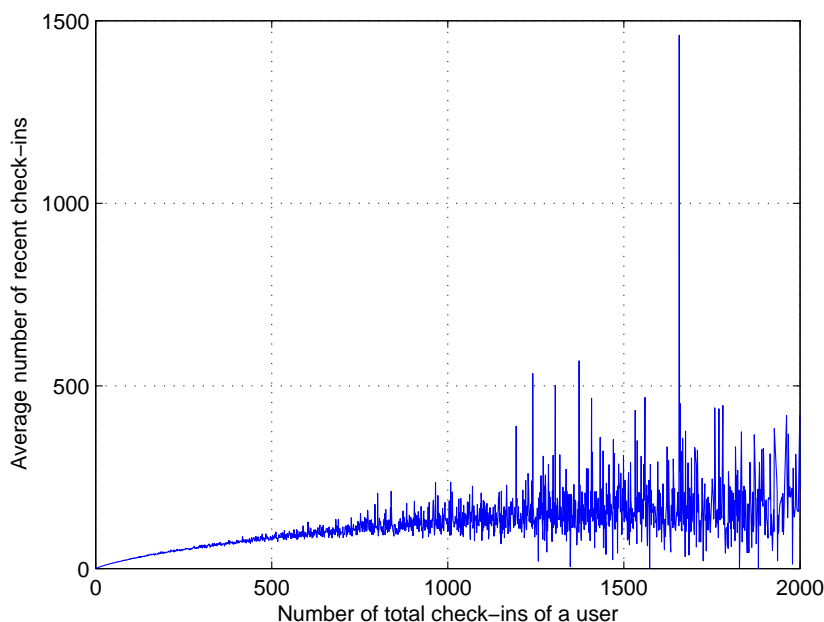


Figure 4.1: Recent check-ins vs. total check-ins: the average recent check-ins of the users who have a certain number of total check-ins.

Figure 4.1 shows a relationship between the number of total check-ins and the recent check-ins. A recent check-in of a user means that the user is in a venue's recent visitor list, but we cannot directly know when this check-in happened. If this user is the only visitor of this venue, then he/she will stay in the recent visitor list even if this check-in happened a year ago. In fact, there are 1,291,125 venues that have only one check-in; and 2,014,305 venues have had only one visitor ever. Though it is not a hard proof, the high ratio of *recent check-ins* to *total check-ins* of a user indicates that it is likely a user plays tricks in order to stay in the recent visitor list, which is a sign of cheating. Here, we only included users with 2000 or less total check-ins since they cover 99.98% of users. We get the number of recent

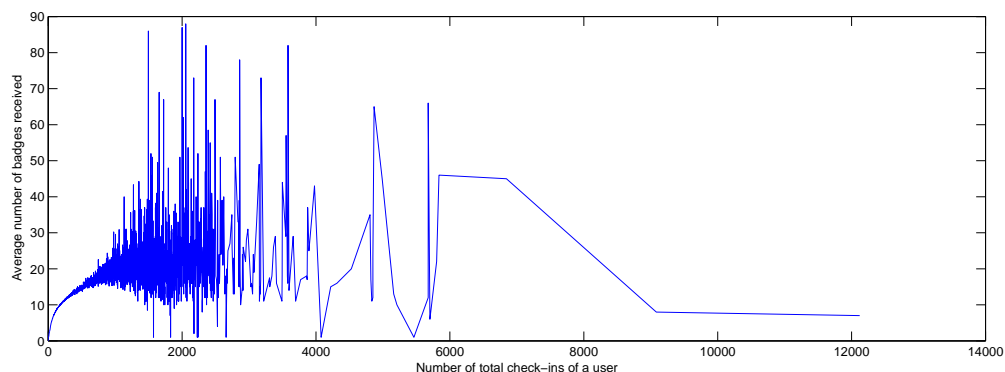


Figure 4.2: Number of badges vs. number of check-ins: The average number of badges granted to users who have a certain number of total check-ins.

check-ins vs. the number of total check-ins of each user, and then we compute the average number of recent check-ins for users who have a given number of total check-ins (see Figure 4.1). We can see that some users with more than 1,000 check-ins have an unusually high percentage of recent check-ins, which suggests that those users are possibly cheaters, since it is not very likely for users to always check into a large number of different venues in a short time period.

From Figure 4.1, we can see that, on average, we get around 100 recent check-ins of a user, if the user did more than 500 check-ins total. There are 25,074 users that have a total check-in number falling in between 500 and 2000. It's not difficult to determine where they have been or are likely to go from this data.

4.2 Low Reward Rate

If a user has a large amount of check-ins but little rewards like badges, the user may have been detected as a cheater by Foursquare so those check-ins were invalidated toward rewards, although they still increase the total number of check-ins of those users under Foursquare's current policy. Figure 4.2 shows the relation

between rewards (badges) and the number of check-ins. We first get the number of badges vs. the number of total check-ins of each user, and then we compute the average number of badges for users who have a given number of total check-ins. As shown in Figure 4.2, for users with 1000 or less check-ins, the relation between the number of check-ins and badges is stable. It illustrates that a user will be likely to get more badges after doing more check-ins. It is reasonable because the rewards are usually granted to those who have checked in over a certain number of times to a venue. For the users with a larger number of check-ins, we can see that the curve in Figure 4.2 oscillates dramatically. Actually, many users with more than 1000 check-ins only have less than 10 badges. We think the best explanation for this is that they are location cheaters and were caught by Foursquare and, thus, their check-ins yielded no rewards. For almost all users with more than 9000 check-ins, the reward level is low. The average check-ins per day for these users is over 16 times since the Foursquare service was launched in March 2009, which is strong evidence that these users are cheaters.

We notice that among the 1.89 million users, 36.3% have never checked into any venues, 20.4% have one to five check-ins, which means more than half of the users have only checked in less than six times. On the other hand, 0.2% of the users have checked in at least 1,000 times; and 11 users have checked in at least 5,000 times. These 11 users who have made no less than 5,000 check-ins can be divided into two distinct groups by the number of mayorships they have. The first group has six users, each of whom is mayor of tens of venues, which are all concentrated in a city area. The other five users in the second group, including the one with over 12,000 check-ins, the highest among all users, do not have any mayorships, and they received much less badges than the first group. A further analysis indicates that four of the five users in the second group appeared in a

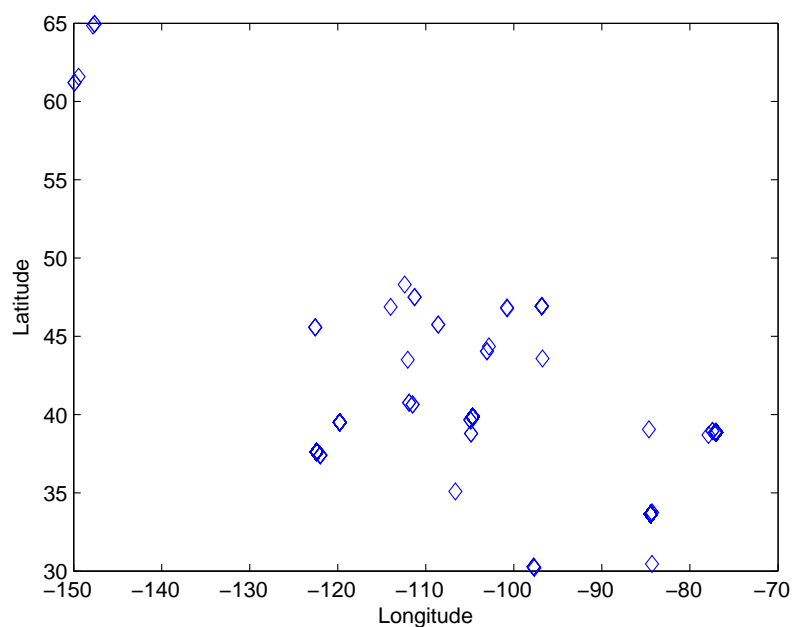


Figure 4.3: Check-in locations of a suspected cheater.

recent visitor list of a venue, while the users in the first group are all in the recent visitor lists of a large amount of venues. This provides us with strong evidence that the users in the second group are cheaters and were caught, so their check-ins were invalidated.

4.3 Suspicious Check-in Patterns

Next, we will examine if the check-in pattern or history can tell if a user is a location cheater through further analysis of the crawled data.

We analyze a user's check-in pattern based on the recent check-in records. Figure 4.3 shows the recent check-in locations of a suspected cheater. We draw the venues to which a user has checked in on a map, so that we have a general idea of the places the user has "visited". This user is in the recent visitor lists of

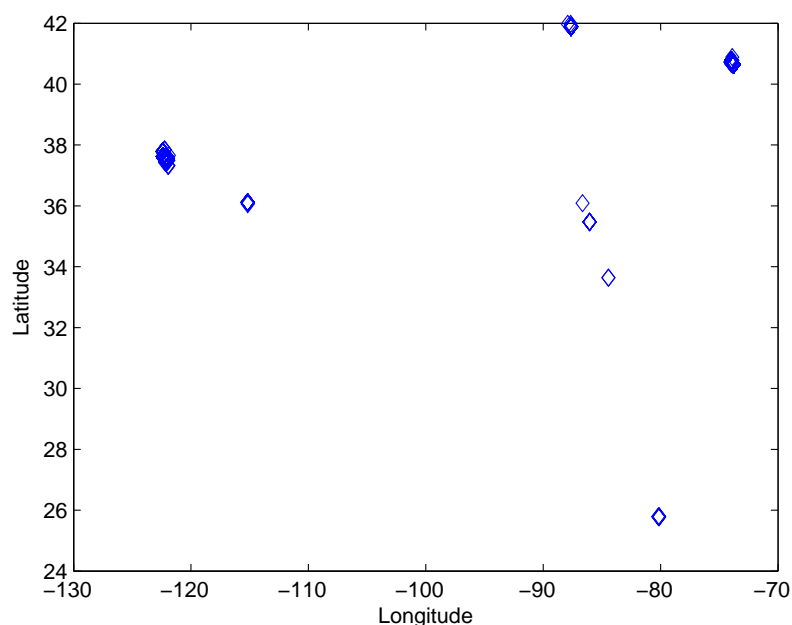


Figure 4.4: Check-in locations of a “normal” user.

over 1000 venues. As we can see in Figure 4.3, those venues are scattered pretty far apart and spread over 30 different cities throughout the United States, including Alaska, and Europe. Judging from this user’s ID (Foursquare increments this ID as user registers), we believe that the user has used Foursquare for less than one year. It illustrates that within a year, the user has “visited” at least 30 different cities, hence this user is suspected of location cheating.

Figure 4.4 shows the recent check-in locations of a user with a similar number of recent check-in records and similar ID (it means the two users registered for the Foursquare service at almost the same time) as the user in Figure 4.3. But the venues he/she visited are concentrated in three cities (places with darker markers) and a few other places, where he/she may have visited on vacation. After examining the users with more than 1,000 recent check-in records, users with more than 2000 total check-ins, and users with more than 100 mayorships,

we believe that the check-in pattern in Figure 4.4 is normal.

In the future, we will focus on those cheaters that haven't been detected by Foursquare's current system. Foursquare implemented its *cheater code* anti-cheating system online around April 2010. Since then, all detected cheating check-ins still count in the total number of check-ins, but do not receive any rewards. By the time this work was conducted (August, 2010), all mayors passed the scrutiny of the *cheater code*. So any cheaters we found in this group of users were new discoveries. There are 425,196 users who have the mayor title, and there are 2,315,747 venues which have mayors. On average, each user with a mayorship is the mayor of 5.45 venues. Those who are mayors of many venues are likely to be cheaters.

Chapter 5

Possible Solutions against Location Cheating

The investigation presented in this work allows an attacker to launch automated location cheating attacks against a large number of victims, including service providers, business partners, and users. The root cause of the vulnerability is the lack of effective location verification mechanisms which can be deployed on a large scale. However, it is possible to counter these attacks. In this chapter, we list possible techniques to thwart location cheating, and we suggest that location security be enhanced by limiting the access to user and venue profiles.

5.1 Location Verification Techniques

Distance bounding: Distance bounding protocols [12], [13], [14] that exploit the limitation on transmission range or speed of a communication signal for location verification, which does not rely on GPS inputs. This solution requires the deployment of verifiers around the registered venues; hence it's expensive to deploy

location verification based on distance bounding.

Address mapping: Using address mapping to geolocate IP addresses has been proposed in various applications, such as *Tracert Map* and *Google Location Service*. Researchers have adopted IP address mapping to locate mobile phones [15]. A challenge of applying IP address mapping to verify location is that mobile phones may access the Internet from nonlocal IP addresses, and the IP addresses can be changed dynamically.

Venue side location verification: The Wi-Fi routers that provide the Wi-Fi hotspot services can work as location verifiers. This technique provides an intrinsic distance bounding since only devices that are physically within the radio communication range of a Wi-Fi router can communicate with it. According to previous literature [16, 17], the radio range of a Wi-Fi router is generally no more than one hundred meters. This range level helps identify cheaters that are miles away from the venue. However, for the cheaters within the transmission range a Wi-Fi router, this approach does not work. For example, a cheater sitting inside a McDonald's can check-in to the Wendy's next door, which is only 50 meters away. In this case, the Wendy's owner can configure the Wi-Fi router to limit the communication within the restaurant via hardware or firmware configuration tools (i.e., DD-WRT [18]).

In this solution, a Wi-Fi router takes the responsibility to measure if a check-in message was sent from a device in a legal area by checking the communication delay between the Wi-Fi router and the device. If so, the Wi-Fi router sends the verification information to the corresponding LBS server. In order to provide location verification service, the Wi-Fi router must be registered to the LBS server

and establish trusted communication with the server to block the impersonating attacks by location cheaters.

When comparing the three solutions, *Distance Bounding* provides the most accurate location data, and it can be used anywhere, but it is difficult to implement and has the highest cost. *Address Mapping* is the least accurate in terms of the location data it provides, it can be used anywhere, and it has the lowest cost and is the easiest to implement. *Venue Side Location Verification* has enough location accuracy, and it incurs no extra hardware purchase or installation cost for the venues. Owners of the venues can simply update the software on their existing routers to make these routers capable of defeating location cheaters.

5.2 Mitigating Threat from Location Cheating

As alluded above, with the assistance of profile analysis, an attacker may optimize the location cheating strategies. To limit the effect of potential location cheating attacks, we need to reduce the information exposed to the public. Along this direction we can employ the following techniques.

Access control for crawling: To prevent large-scale profile analysis by attackers, a direct solution is to take counter measures to stop or limit crawling. If a user must login to view the publicly available profile pages, it's easier to detect the crawling users and block them. This can be combined with IP address blocking, if the service provider can detect the crawler's IP address. Even if the crawlers hide behind network address translations (NATs), blocking their IP addresses causes limited collateral damage. Casado and Freedman [19] show most NATs only have a few hosts behind them, and proxies generally have much more. Crawling

behind a public proxy cannot achieve enough performance. Although tools like Tor [20] may provide a high level of anonymity on the Internet, it also suffers from limited performance for the purpose of crawling.

Hiding information from profiles: To reduce the information leak, we hope that even if an attacker successfully crawled the website, the information that can be extracted from the data is still limited. But if a subset of information in the profiles is removed, the usability of the location-based social networking service will suffer. For example, if the recent check-in list is removed from the venue's profile, users cannot contact the recent visitors to the venue for their comments about the venue. Hence, removing the information from profiles is not a good solution to prevent profile analysis. Rather, the service provider may use the hash function to hide necessary information (such as user IDs in the recent check-in list). Recently, the information leak has been studied. Griffith and Jakobsson [21] use public records to infer individuals' mothers' maiden names, and Heatherly et al. [22], as well as Zheleva and Getoor [23], show how public data provided by social networks can be used to infer private information.

Chapter 6

Conclusions and Future Work

In this chapter we present conclusions and future work of our project.

6.1 Conclusion

In this thesis, we introduced a novel and practical location cheating attack that enables an attacker to make the location-based service providers believe that the attacker is in a place far away from his/her real location. Through real world experiments on Foursquare, the leading location-based social network, we demonstrate that our attacking approach works as expected; and location cheating really threatens the development and deployment of location-based mobile social network services. The counter measures against location cheating in current systems are not perfect.

6.2 Future Work

We plan on two directions for future work: one focuses on crawling-based privacy collecting, the other is the defense of location cheating.

6.2.1 Privacy Leakage

We show the leakage of personal location history in Figure B.1, the “Who’s been here” section reveals the list of recent visitors to this venue. After we crawled webpages for all venues, we built a personal location history for each user on Foursquare. Unfortunately, there are two obstacles that stopped us from further investigation into privacy leakage in this work: (1) the “Who’s been here” does not give the full list of visitor history; (2) the “Who’s been here” section was removed right after we finished all the crawling.

In the future, we would like to take two approaches to investigate privacy leakage: First, we will investigate more LBSs and providers of other services that involve sensitive yet partially public personal information. Second, we will try to collect location privacy from traditional social network services like Facebook and Twitter, because many other services including LBSs are tightly integrated with them. For example, Foursquare allows users to sign in using their Facebook account, and after each successful check-in, Foursquare app wants to broadcast this message on Facebook and Twitter by default.

6.2.2 Defense of Location Cheating

We have suggested several techniques for enhancing the security of location information. In the future, we will investigate further to find better solutions to identify possible cheaters, especially those whom haven’t been found by the existing anti-cheating mechanisms. We also would like to seek better solutions to the balance between usability and security in order to make location-based mobile social networking services more attractive.

Bibliography

- [1] <http://www.foursquare.com>. 1.1, 2.1
- [2] <http://www.gowalla.com>. 1.1
- [3] <http://www.gypsii.com>. 1.1
- [4] <http://www.loopt.com>. 1.1
- [5] <http://www.brightkite.com>. 1.1
- [6] http://www.skylab-mobilesystems.com/en/products/gps_sim.html. 2
- [7] <http://www.zylsoft.com/vgps.htm>. 2
- [8] <http://www.avangardo.com/software/gps-generator-pro.html>. 2
- [9] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel, "A practical attack to de-anonymize social network users," in *2010 IEEE Symposium on Security and Privacy*, pp. 223–238, IEEE, 2010. 3.2
- [10] J. Cho and H. Garcia-Molina, "Parallel crawlers," in *Proceedings of the 11th international conference on World Wide Web*, pp. 124–135, ACM, 2002. 3.2
- [11] D. Chau, S. Pandit, S. Wang, and C. Faloutsos, "Parallel crawling for online social networks," in *Proceedings of the 16th international conference on World Wide Web*, pp. 1283–1284, ACM, 2007. 3.2

- [12] G. Hancke and M. Kuhn, "An RFID distance bounding protocol," 2005. 5.1
- [13] J. Chiang, J. Haas, and Y. Hu, "Secure and precise location verification using distance bounding and simultaneous multilateration," in *Proceedings of the second ACM conference on Wireless network security*, pp. 181–192, ACM, 2009. 5.1
- [14] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Proceedings of the 2nd ACM workshop on Wireless security*, pp. 1–10, ACM, 2003. 5.1
- [15] M. Balakrishnan, I. Mohamed, and V. Ramasubramanian, "Where's that phone?: geolocating IP addresses on 3G networks," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pp. 294–300, ACM, 2009. 5.1
- [16] W. Lehr and L. McKnight, "Wireless Internet access: 3G vs. WiFi?* 1," *Telecommunications Policy*, vol. 27, no. 5-6, pp. 351–370, 2003. 5.1
- [17] A. Howard, S. Siddiqi, and Sukhatme, "An experimental study of localization using wireless ethernet," in *Field and Service Robotics*, pp. 145–153, Springer, 2006. 5.1
- [18] <http://www.dd-wrt.com>. 5.1
- [19] M. Casado and M. Freedman, "Peering through the shroud: The effect of edge opacity on IP-based client identification," in *Proceedings of the 4th Networked Systems Design and Implementation*, 2007. 5.2

- [20] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, p. 21, USENIX Association, 2004. 5.2
- [21] V. Griffith and M. Jakobsson, "Messin' with Texas Deriving Mother's Maiden Names Using Public Records," in *Applied Cryptography and Network Security*, pp. 91–103, Springer, 2005. 5.2
- [22] R. Heatherly, M. Kantarcioglu, B. Thuraisingham, and J. Lindamood, "Preventing private information inference attacks on social networks," 2009. 5.2
- [23] E. Zheleva and L. Getoor, "To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles," in *Proceedings of the 18th international conference on World wide web*, pp. 531–540, ACM, 2009. 5.2

Appendix A

Multi-threading in Crawler

Control of multi-threading in the crawler(C#):

```
// Mutex for multi-thread access to critical variables.
private static Mutex m_mutex = new Mutex();

// Desired thread count.
private int m_threadCount = 0;

// Set this to false to stop launching new threads.
private bool m_bRunning = false;

// Launch new threads until desired number was reached.
public void StartThread()
{
    // User can adjust thread count in numericUpDown1
    while (m_threadCount < numericUpDown1.Value)
    {
```

```
m_mutex.WaitOne();
m_threadCount++;
m_mutex.ReleaseMutex();

textBox2.Text = m_id.ToString();

// Launch a new crawling thread. Each thread crawls
// one URL which corresponds to one user or venue.
Thread newThread = null;
if (m_mode == Mode.User) // Crawl user information
{
    newThread = new Thread(new ThreadStart(GetUserInfo));
}
else // Crawl venue information
{
    newThread = new Thread(new ThreadStart(GetVenueInfo));
}
newThread.Start();
}
}

// This is a delegate to the real ThreadTerminated function.
delegate void ThreadTerminatedCallback(bool bSuccess);

// Define a callback function for the crawling thread, when it
// finishes, it will call this function.
```

```
public void ThreadTerminated(bool bSuccess)
{
    if (this.textBox1.InvokeRequired)
    {
        // This part is to go back to the thread of user
        // interface (main thread), so that user can see
        // the update immediately.
        ThreadTerminatedCallback d =
            new ThreadTerminatedCallback(ThreadTerminated);
        this.Invoke(d, new object[] { bSuccess });
    }
    else
    {
        m_mutex.WaitOne();

        m_threadCount--;

        // Record process history
        m_processed++;
        if (!bSuccess)
        {
            m_failed++;
        }

        // Update user interface
        textBox3.Text = m_processed.ToString();
    }
}
```

```
textBox4.Text = m_failed.ToString();

// Display remaining thread count during stopping.
if (false == m_bRunning)
{
    Log("Working threads: " + m_threadCount);

    if (m_threadCount == 0)
        Log("Stopped");
}

m_mutex.ReleaseMutex();

// Always try to max the thread count after a thread
// finishes.
if (true == m_bRunning)
{
    StartThread();
}
}
}
```

Appendix B

Screenshots

Screenshots of Foursquare's website and some tools used in location cheating.

UNMCS1 Find Friends Add Things Apps Help Settings Logout

foursquare™ Find places, people, tags **SEARCH**


ME HISTORY STATS FRIENDS Currently in **Ottawa, Ontario**

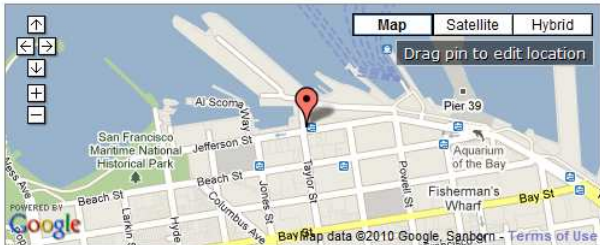
Fisherman's Wharf Sign

Taylor St
@ Jefferson St
San Francisco, CA 94133

Like


Are you the manager of this business?

CHECKINS HERE **532** UNIQUE VISITORS **463** YOUR VISITS **9** MAYOR 



Map Satellite Hybrid Drag pin to edit location

WHO'S BEEN HERE



CATEGORIES

Add categories below or on your [history page](#).

- + Sculpture primary category
- + Plaza / Square [set as primary](#)

Add a category:

TIPS

Paolo F. did this...
Tourist trap (April 10, 2010)

E Drew P. did this...
Taylor at Jefferson is the Heart of Fisherman's Wharf and where you will find a 3 story Ships Wheel with a Crab in the center welcoming you to the Wharf. Get a picture when a Street Car goes by! (February 23, 2010) [Link]

Robert B. did this...
A great "Hey, I'm in San Francisco" photo opportunity to be had here. Also lots of great restaurants and entertainment within walking distance. (February 20, 2010)

Morris C. did this...
Go to Macy's and adopt a friend from the SPCA. (December 14, 2009) [Link]

TAGS

Use tags to let people know what they can expect to find here:

- + official fisherman's wharf crab wheel sign
- + artists
- + photo op
- + tourist attraction

Add a tag:

Figure B.1: Page of a venue on Foursquare's website, "Who's been here" section reveals users location history, and has been removed.

UNMCS1 Find Friends Add Things Apps Help Settings Logout

foursquare™ Find places, people, tags SEARCH

ME HISTORY STATS FRIENDS Currently in Ottawa, Ontario

UNMCS1
This is you!

TOTAL NIGHTS OUT	TOTAL CHECKINS	TO DOS NOW DONE	TOTAL THINGS DONE
10	55	0	0

TIPS TO DOS + Add a Tip
You haven't added any tips near Ottawa, Ontario yet. [Get to it!](#)

BADGES

MAYORSHIP
You are currently the mayor of 1 place! [huh?]
+ [Fisherman's Wharf Sign](#) (San Francisco, CA) [X]

FEED

- @Electric Medialand (2 weeks ago)
- @Sports Authority (2 weeks ago)
- @LivingWell Docklands (2 weeks ago)
- @Saratoga Flats Raceway (2 weeks ago)
- @Bath Sense & More (2 weeks ago)
- @Copernicus Theatre (2 weeks ago)
- @King Solomon Restaurant (2 weeks ago)
- @Boathouse Group (2 weeks ago)
- @Martini Room (2 weeks ago)
- @Sports Authority - Milpitas (2 weeks ago)

FRIENDS
Looks like you don't have any foursquare friends. Let's [find your friends](#) or [invite](#) new ones!

ALSO ON FOURSQUARE
Follow these brands to unlock badges and find interesting tips around your city

HUFF POST PA visitPA.com TLC bon appetit H AskMen.com Lucky

about | press | jobs | privacy | terms | help | contact | blog © 2010 foursquare

Figure B.2: The user page of our testing user.

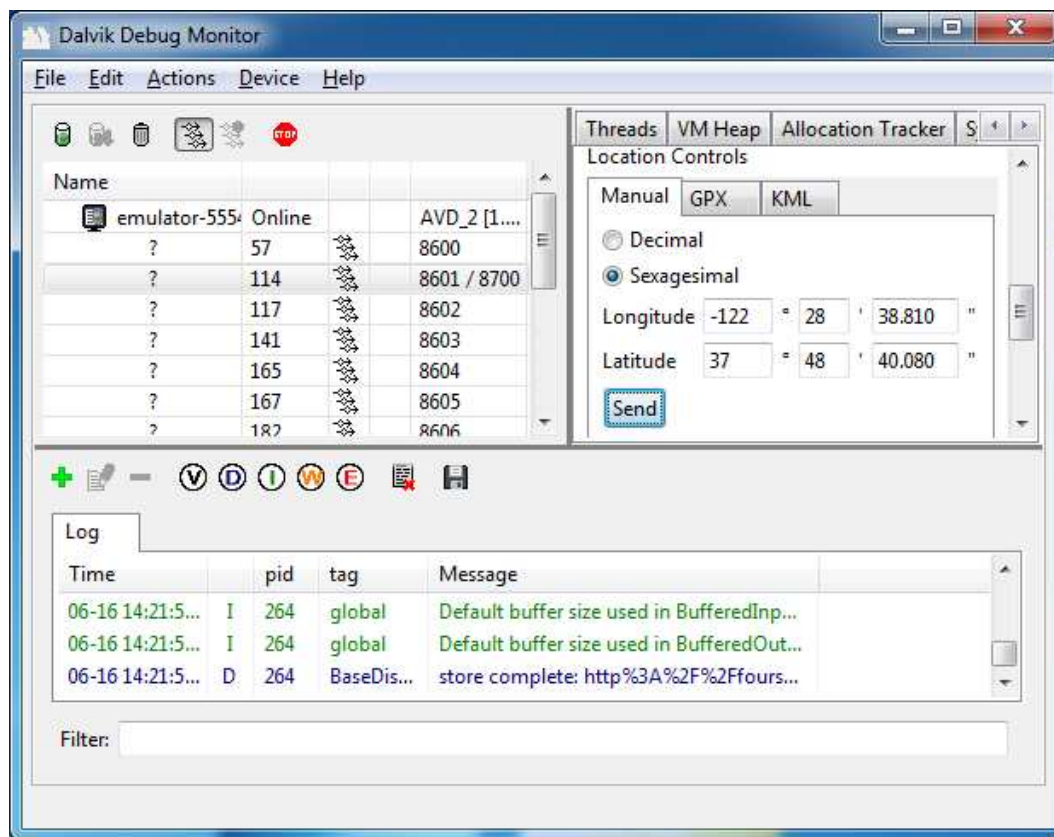


Figure B.3: Use Dalvik Debug Monitor to set the GPS of Android device emulator to Golden Gate Bridge.

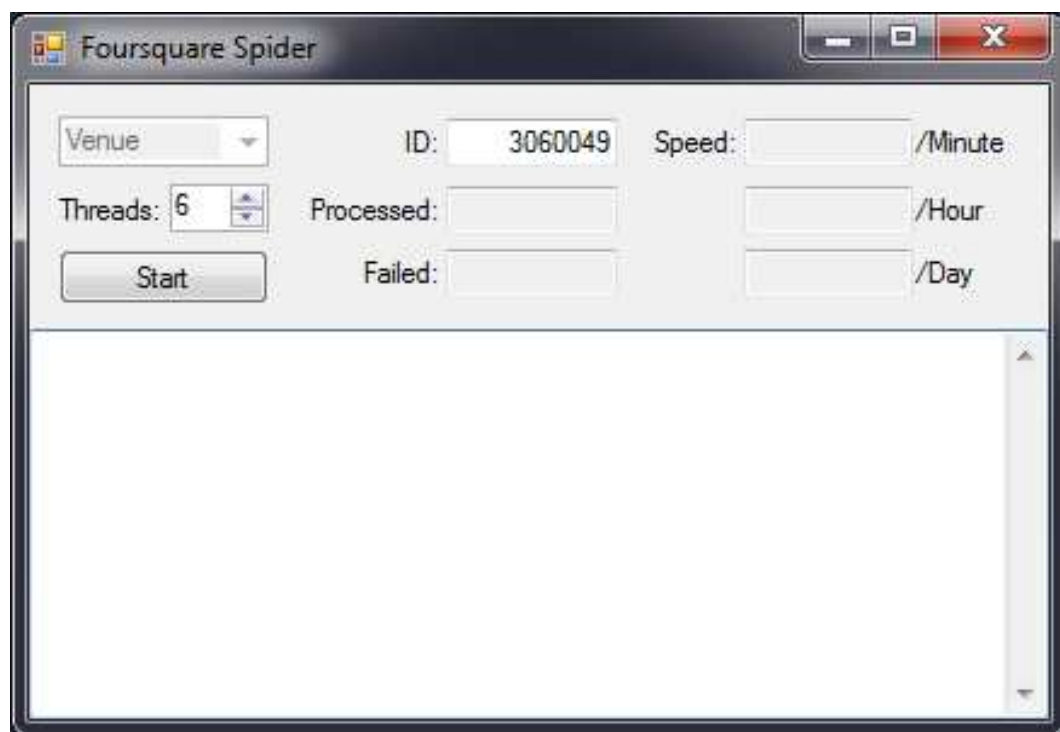


Figure B.4: The multi-threaded crawler we developed to crawl Foursquare's website.