I Know the Shortened URLs You Clicked on Twitter: Inference Attack using Public Click Analytics and Twitter Metadata

Jonghyuk Song
Dept. of CSE, POSTECH
Pohang, Republic of Korea
freestar@postech.ac.kr

Sangho Lee
Dept. of CSE, POSTECH
Pohang, Republic of Korea
sangho2@postech.ac.kr

Jong Kim
Div. of ITCE, POSTECH
Pohang, Republic of Korea
jkim@postech.ac.kr

ABSTRACT

Twitter is a popular social network service for sharing messages among friends. Because Twitter restricts the length of messages, many Twitter users use URL shortening services, such as bit.ly and goo.gl, to share long URLs with friends. Some URL shortening services also provide click analytics of the shortened URLs, including the number of clicks, countries, platforms, browsers and referrers. To protect visitors' privacy, they do not reveal identifying information about individual visitors. In this paper, we propose a practical attack technique that can infer who clicks what shortened URLs on Twitter. Unlike the conventional browser history stealing attacks, our attack methods only need publicly available information provided by URL shortening services and Twitter. Evaluation results show that our attack technique can compromise Twitter users' privacy with high accuracy.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Twitter; URL shortening service; Inference; Privacy leak

1. INTRODUCTION

Twitter is one of the most popular social network services for exchanging messages (tweets) among people. On April 5, 2012, Twitter announced that it has over 140 million active users and that more than 340 million messages are created every day [26]. Another interesting characteristic of Twitter is its ecosystem. On July 11, 2011, Twitter advertised that it has over one million registered applications built by more than 750,000 developers [25]. The third party applications include client applications for various platforms, such as Windows, Mac, iOS, and Android, and web-based applications such as URL shortening services, image-sharing services, and news feeds.

Among the third party services available to Twitter users, URL shortening services are one of the most essential services. Because Twitter restricts the length of a tweet to 140 characters and allows a tweet to contain only text, Twitter might not be able to include their complete thought in a tweet. Therefore, when a user wants to share more complicated information, such as news or multimedia pages, he will include a URL of the web page that contains the information into a tweet. However, when the length of an entire message, including the URL, is greater than 140 characters, the problem still exists. URL shortening services solve this length problem by providing a shortened URL that redirects visitors to the original, longer URL. Moreover, some URL shortening services, such as *bit.ly* and *goo.gl*, publicly publish *click analytics* which include the number of clicks, countries, browsers and referrers of visitors. Anyone can use such data to analyze statistics of visitors of a shortened URL. A curious user or an attacker might even want to obtain specific information about individual visitors of the shortened URL. However, to protect the privacy of visitors, URL shortening services only provide aggregated data; therefore, we cannot distinguish individual visitors using these data only. The main question is whether *we can extract information that can be used to identify individual visitor from the aggregated click analytics.*

Interestingly, Twitter itself provides a set of metadata that can be used to differentiate Twitter users. For instance, if a user, Alice, updates her messages using the official Twitter client application for iPhone, "Twitter for iPhone" will be included in the source field of the metadata of her messages. Using this information, we can determine that Alice is an iPhone user. Moreover, Alice might have disclosed on her profile page that she lives in the USA or she might have activated the location service of a Twitter client application to automatically fill the location field in the metadata. From this information, we can conclude that Alice is in the USA.

Along with the above example, let us consider a simple inference attack conducted by Bob – Alice's boyfriend. Bob posts a tweet with a URL shortened by *goo.gl*, and Alice sees the Bob's URL. If Alice clicks on the shortened URL, then *goo.gl* records {"country": "US", "platform": "iPhone", "referrer": "twitter.com"} in the click analytics of the shortened URL. Otherwise, no information may be added to the click analytics. Later, Bob retrieves the click analytics of the

shortened URL to know whether Alice clicked on his URL or not. If the click analytics has not changed or if its changes do not include information about the USA, iPhone, and twitter.com, he could infer that Alice did not click on his URL. Otherwise, he could infer that Alice clicked on his URL. This simple form of inference may include some errors because another Twitter user who also uses "Twitter for iPhone" in the USA could click on Bob's shortened URL. However, the main advantage of this inference attack is that *it is a passive attack relying on public information only*, unlike conventional browser history stealing attacks [3, 8, 10, 11, 14–18].

The goal of history stealing attacks is to know the URLs that a target browser (host or user) visited. However, all of the existing history stealing attacks are active attacks and require some private information. The required information includes Cascading Style Sheet (CSS) visited styles, browser cache, DNS cache and latency. To collect such information, we have to prepare a web page that contains scripts or malware that extract the CSS styles, browser cache or required time to load some pages from a visited browser, or monitor DNS requests to measure the DNS lookup time of a target host. In other words, we need to deceive or compromise a target user or his network to obtain the browsing history.

In this paper, we propose an attack technique to infer whether a specific user clicked on certain shortened URLs on Twitter. As shown in the above simple inference attack, our attack is based on the combination of publicly available information: click analytics from URL shortening services and metadata from Twitter. The goal of the attack is to know which URLs were clicked on by a target user. To perform the attack, we create *monitoring accounts* that monitor messages from all followings of a target user to collect all shortened URLs that the target user might click on. We then monitor the click analytics of those shortened URLs and compare them with the metadata of the target user. Such an attack could be used for targeted marketing, targeted spamming, or cyberstalking. Evaluation results show that our attack can successfully infer the click information with a high degree of probability.

In summary, the main contributions of this paper are as follows:

- We propose novel attack techniques to determine whether a specific user clicks on certain shortened URLs on Twitter. To the best of our knowledge, this is the first study that infers URL visiting history on Twitter.
- We only use public information provided by URL shortening services and Twitter; i.e., click analytics and Twitter metadata. We determine whether a target user visits a shortened URL by correlating the publicly available information. Our approach does not need complicated techniques or assumptions such as script injection, phishing, malware intrusion or DNS monitoring. All we need is publicly available information.

2. URL SHORTENING SERVICES

The first notable URL shortening service is TinyURL, which was launched in 2002. The success of TinyURL influenced the development of many URL shortening services. These services reduce the length of URLs for easy sharing. Shortened URLs are especially convenient for users of Twitter, which imposes a limit on the length of a message. In the past, Twitter used TinyURL and *bit.ly* as the default URL shortening services. As of October 10, 2011, Twitter started using its own URL shortening service, *t.co*, to wrap all URLs in tweets in order to protect Twitter users from malicious URLs [24, 28].

Some URL shortening services provide click analytics about each shortened URLs. Whenever a user clicks on a shortened URL, some information about the user is recorded. The click analytics is usually made public and can be accessed by anyone. Among such URL shortening services, we focus on *bit.ly* and *goo.gl* because they are broadly used and provide meaningful information.

2.1 goo.gl

In December 2009, Google launched a URL shortening service called Google URL Shortener at *goo.gl*. Its click analytics provides information about the visitors as follows:

- Referrers
- Countries
- Browsers
- Platforms

For example, let us assume a user uses a BlackBerry phone and is located in the USA. If he clicks on a shortened URL from *goo.gl* on Twitter, *t.co* is recorded in the Referrers field; *Mobile Safari* in the Browsers field; *US* in the Countries field; and *BlackBerry* in the Platforms field of *goo.gl*'s click analytics. The reason why *t.co* is recorded in the Referrers field is that all links shared on Twitter are wrapped using *t.co* by Twitter from October 10, 2011.

2.2 bit.ly

Bitly company launched a URL shortening service *bit.ly* in 2008. Its click analytics provides information about visitors as follows:

- Referrers
- Countries

bit.ly does not provide information about browsers and platforms. However, its Referrers field has more detailed information than that of *goo.gl*. When a user clicks on a shortened URL on Twitter, only "*t.co*" is recorded in the Referrers field in the *goo.gl* click analytics. However, *bit.ly* records the entire URL of the referrer site in the Referrers field, as "http://t.co/*****". With the information provided by *goo.gl*, we only know whether a visitor of a shortened URL comes from Twitter or not. However, if we use the information provided by *bit.ly*, we can determine the exact URL of the tweet containing the clicked shortened URL. This information makes our inference attack possible even without having information about browsers and platforms.

3. USER MATCHING

Whenever we notice that there is a visitor of the shortened URL by monitoring the click analytics, we compare the information about the visitor and Twitter users. If the shortened URL is *goo.gl*, the information about the visitor consists of four parts: Referrers, Countries, Platforms, and Browsers. If the shortened URL is *bit.ly*, only Referrers and

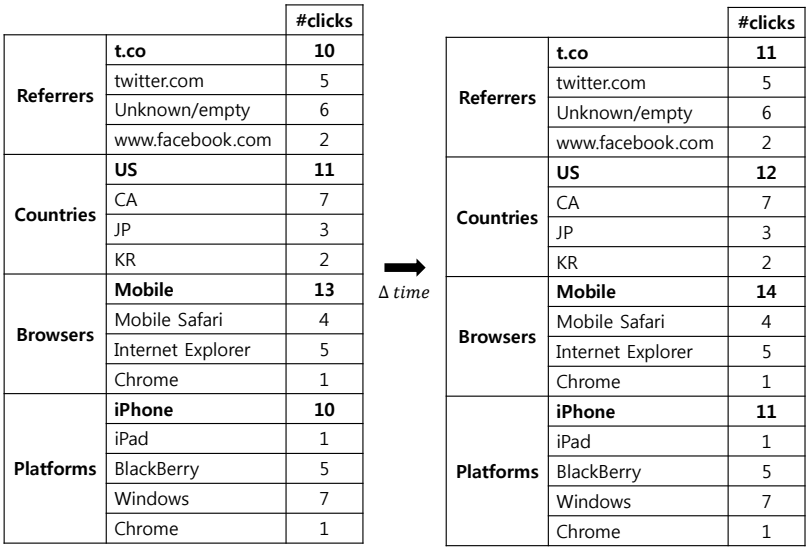

Figure 1: The proposed system notices that there is a visitor using differences in the click analytics. We can infer that the information of the visitor is {"country": "US", "platform": "iPhone", "referrer": "t.co"} (*goo.gl* case)

Countries are provided. We regularly monitor the click analytics to check whether the number of clicks on a shortened URL increased, which indicates a new visitor. Information about the visitor can be obtained from the differences between the new and the old click analytics. Figure 1 shows the example of the process we used to obtain the information about the visitor. Then we compared the information about the visitor who clicked on the shortened URL with information about the Twitter users whom we were tracking.

Twitter does not officially provide personal information about Twitter users such as country, browsers and platforms. Therefore, we need to infer the information about Twitter users by investigating their timeline and profile pages. Next, we describe how we extract the information from Twitter metadata.

3.1 Referrers

Our goal is to identify whether a known user clicked on a specific shortened URLs on Twitter. We can determine whether the visitor comes from Twitter by using the referrer information. The click analytics of *goo.gl* only records the hostname of the referrer site; therefore, if a visitor comes from Twitter, "*t.co*" or "*twitter.com*" is recorded in the Referrers field. In most cases, "*t.co*" is recorded because all links shared on Twitter are automatically shortened to *t.co* links. *t.co* handles redirections by context and user agents, so Referrer depends on the source of click [27]. In some cases, "*twitter.com*" is recorded because some Twitter applications use original links instead of *t.co* links. Therefore, if the Referrers information of the visitor is "*t.co*" or "*twitter.com*", we regarded the visitor as coming from Twitter.

When a shortened URL is provided by *bit.ly*, we can analyze it in greater detail, because the entire URL of the referrer site is provided in the click analytics of the shortened URL (Table 1). On Twitter, all URLs are converted into different *t.co* URLs. If the target user clicks on the shortened URL, the URL shortening services record the *t.co* URL in the Referrers field. The referrer match is considered successful when the *t.co* URL recorded in the click analytics is the same as the *t.co* URL of the target shortened URL.

		#
Referrers	direct	2
	http://t.co/3slAb	8
	http://t.co/xInA4	4
	https://twitter.com/[UID]/status/[TID]	3
Countries	US	9
	KR	5
	ID	1
	CH	2

Table 1: The examples of *bit.ly* click analytics. UID is a Twitter ID and TID is a numerical ID of each tweet.

3.2 Country

The country information of a Twitter user can be inferred using the location field in the profile page. In many cases, Twitter users fill in the location field with their city or place name. We can determine the user's country by searching GeoNames with the information in the location field of the user's Twitter profile [1]. GeoNames returns the country code that corresponds to the search keywords. The country information provided by the click analytics is also a country code; therefore, we have a successful country match if both country codes are the same.

Some Twitter users, hide their location by leaving the location field empty. Other users fill in the location field with meaningless information, such as "earth" and "in your heart." We cannot obtain accurate location information from these users. In those cases, we do not perform country matching. In our attack experiments, we avoided these problems by selecting only target users who filled in valid location names in the location field. However, even without location information, our attacks are still possible with other information. Location information increases the accuracy of our attacks, but it is optional.

Source	Browsers	Platforms
Twitter for iPhone	Mobile	iPhone
Twitter for iPad	Mobile	iPad
Twitter for Android	Mobile Safari	Linux
Twitter for BlackBerry	Mobile Safari	BlackBerry

Table 2: The examples of Browsers and Platforms corresponding to source

Additionally, we could rely on recent studies, which infer the location of Twitter users based on their posts [7, 13].

3.3 Browsers and Platforms

When our target user clicks a shortened URL provided by *goo.gl*, we can use the browser and platform information of the target user to increase the accuracy of inference because the click analytics of *goo.gl* provide such information unlike *bit.ly*. Twitter does not provide the browser and platform information of Twitter users, but Twitter provides the information what applications are used for posting the tweets. Whenever a Twitter user posts a tweet, the application name is recorded in the Source field of the tweet. For example, if a user uses the official Twitter client application for the iPhone, "via Twitter for iPhone" is recorded in the Source field. We can use this source information to infer the browser and platform used. Table 2 shows an example of the source values corresponding to browsers and platforms. Some values in the Sources field, however, correspond to several browsers and platforms because some applications support multiple platforms. For instance, TweetDeck is a multi-platform application that support the iPhone, Android, Windows, and Mac OS X. If a Twitter user uses a multi-platform application, we assume that the user uses all the platforms that the application supports.

4. INFERENCE ATTACK IN THE SIMULATED ENVIRONMENT

The definite ways that can exactly evaluate our system are asking the target users whether they really visited the shortened URLs or not, or monitoring their browsing activities by using logging software. However, both approaches are restrictive because we cannot survey all of them or require them to install logging software. Therefore, we built a simulated environment where we performed our experiments. Figure 2 shows the overall architecture of the attack in the simulated environment.

In this experiment, we used virtual users instead of Twitter users in the real world. The system tried to infer the shortened URLs clicked on by the virtual users. The processes involved in this attack system are as follows:

1. The system monitors the click analytics of the shortened URLs that are posted by Twitter users.
2. Changes in the shortened URL's click analytics indicate a new visitor, and the system extracts the visitor information from the click analytics.
3. The extracted visitor information is recorded in the simulated click analytics.
4. The system stochastically adds the information about the virtual users to the simulated click analytics to simulate the click of a real Twitter user.

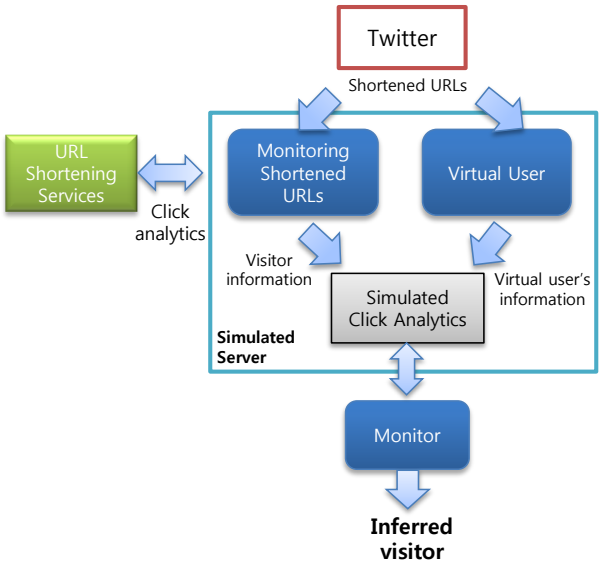

Figure 2: Overall architecture of the attack in the simulated environment

5. Changes in the simulated click analytics indicate a new visitor of the simulated server, an inferred visitor, and the system extracts the information of the inferred visitor from the simulated click analytics.

6. The system compares the inferred visitor and the information about the virtual user. If the information is matched, we infer that the shortened URL was clicked on by the virtual user.

Before the experiment could start, we selected 56 Twitter users who posted *goo.gl* or *bit.ly* URLs regularly. The clicks of virtual users are controlled by the system. Whenever a shortened URL is posted by a Twitter user, the virtual users click on the shortened URL with a probability of 0.7. We correctly know which shortened URLs were clicked on by the virtual users, so we can estimate the performance of the system.

We cannot test all types of Twitter users using the virtual users. Twitter users come from many countries, and they use many different platforms and browsers. Therefore, we had to limit the number of user types for our experiment. We selected six countries: United States (US), Great Britain (GB), Brazil (BR), Japan (JP), Italy (IT), and Rwanda (RW). The first five are among the top 20 countries with the largest number of Twitter accounts [23]. We added Rwanda to learn the effectiveness of the system when the target user lives in a country with only a few Twitter users. We also selected four smartphone platforms: an iPhone, Android, BlackBerry, and a Windows Phone. A combination of six countries and four platforms, gave us 24 types of users for the experiment.

4.1 Data Collection

We collected data by crawling the click analytics of the shortened URLs, using the API methods offered by *goo.gl* and *bit.ly*. *goo.gl* APIs have a rate limit of 1,000,000 queries per day. Similarly, *bit.ly* allows users to create no more than five concurrent connections from one IP address. *bit.ly* also enforces per-hour limits, per-minute limits, and per-IP rate limits for each API method. However, *bit.ly* does not publish the exact number of allowed requests on each limit. In all, we monitored 31,525 *goo.gl* URLs and 24,144 *bit.ly* URLs from September to October 2012. Those shortened URLs were posted by 56 Twitter users (Table 3).

# of followers	*goo.gl*	*bit.ly*
100 - 1k	3	3
1k - 10k	10	5
10k - 100k	9	13
100k - 1M	7	6
total	29	27

Table 3: Twitter users used in the simulated experiment

		Actual value	
		Click	Non click
Prediction Value	Click	True positive	False positive
	Non click	False negative	True negative

Table 4: Confusion matrix

4.2 Evaluation

In this experiment, true positive rate (TPR) is meaningless because false negative is always zero. False negative cases are cases where a virtual user clicks on the shortened URL but the system infers that the virtual user has not click on the URL (Table 4). In the real world, the system is occasionally unable to obtain all information about the target user if he uses several platforms and browsers. In the simulated environment, however, the system knows all information about the virtual users, and the information is not changed during the entire experiment. Therefore, the system always knows what URLs are clicked on by the virtual user by monitoring the simulated click analytics. However, false positive cases are possible because some Twitter users have the same information as the virtual users. If such Twitter users click on the shortened URLs monitored, we get a false positive result. For these reasons, we used two metrics to evaluate the system: precision and false positive rate (FPR).

$$\textbf{Precision} = \frac{\textbf{True positive}}{\textbf{True positive} + \textbf{False positive}}.$$

$$\textbf{FPR} = \frac{\textbf{False positive}}{\textbf{False positive} + \textbf{True negative}}.$$

4.2.1 *goo.gl*

We created a Twitter account and followed 29 Twitter users who posted *goo.gl* URLs regularly. The accuracy of our system depends on the number of the followers of the those users because the shortened URLs posted on Twitter are exposed to the followers of the posting users. With a large number of followers, it is highly likely that many of those followers live in the same country and use the same platform or browser as the target user. Therefore, our system would guess incorrectly because the system misjudges those other users as the target user. We grouped the posting users based on the number of their followers to determine the effect of the number of followers on the results of the experiment.

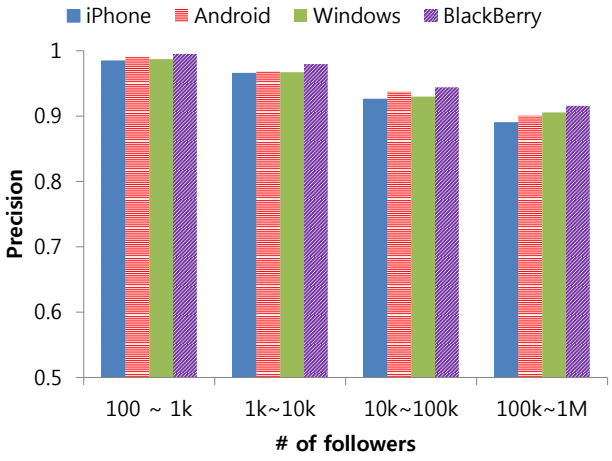

Figure 3: The precision of *goo.gl* URLs in terms of platforms. X axis means the number of followers of the updating users.

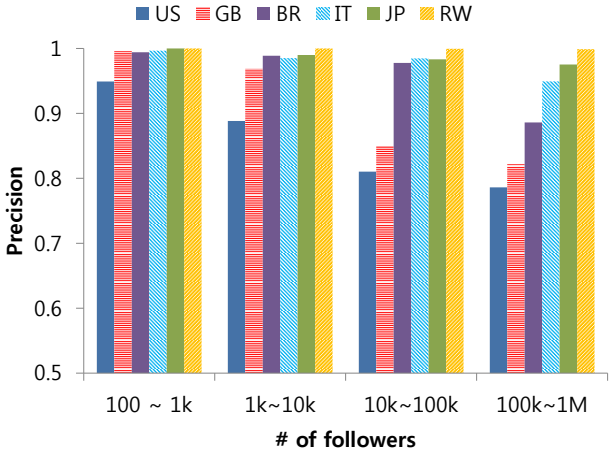

Figure 4: The precision of *goo.gl* URLs in terms of country. X axis means the number of followers of the updating users.

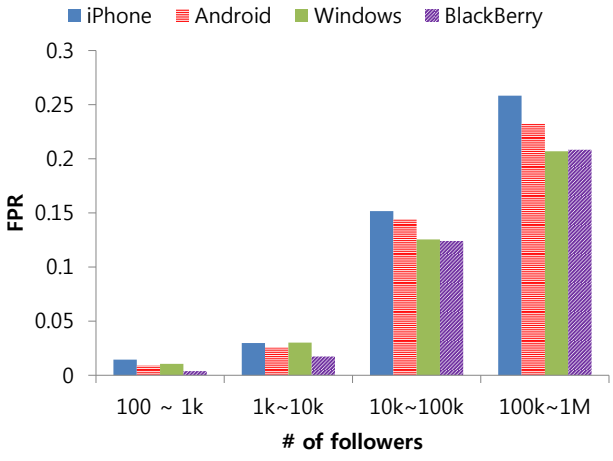

Figure 5: The FPR of *goo.gl* URLs in terms of platforms. X axis means the number of followers of the updating users.

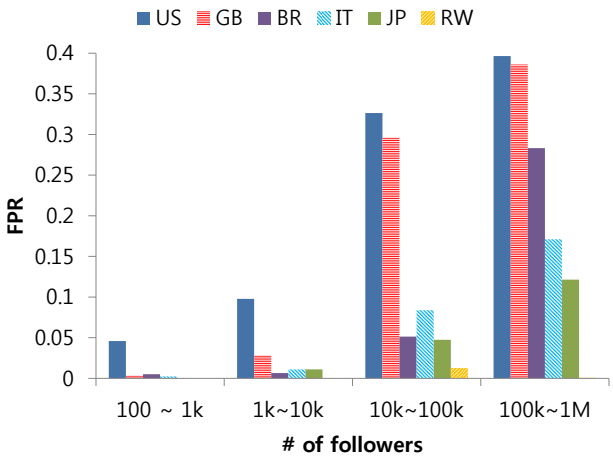

Figure 6: The FPR of *goo.gl* URLs in terms of country. X axis means the number of followers of the updating users.

Figure 3 shows the precision of each platform according to the number of followers. We expected to see low precision with iPhone and Android users because of the large number of users on those platforms. As expected, the results showed that our system had the lowest precision with iPhone users; however, the differences among the platforms were minimal. Average precisions of each platform were as follows: iPhone was 0.94, Android was 0.95, Windows Phone 0.95 and BlackBerry was 0.96.

Figure 4 shows the precision of each country according to the number of followers. The result is compared against the number of Twitter accounts in that country. We achieved the lowest precision with US users because they comprise a large percentage of the total number of Twitter accounts [23]. Average precisions of each country were as follows: US was 0.85, GB was 0.90, BR was 0.96, IT was 0.97, JP was 0.98 and RW was 0.99. The total average precision for all countries in our experiment was 0.94.

Both results showed that the precision decreased as the number of followers increased. The average precision was 0.99 when the number of followers was less than 1,000, but the average precision was 0.90 when the number of followers was greater than 100,000.

As shown in Figures 5 and 6, FPR results showed an inverse correlation with the precision results. US and iPhone users had higher FPR than others. The total average FPR was 0.1.

4.2.2 *bit.ly*

Our monitoring account also followed 27 Twitter users who updated *bit.ly* URLs regularly. Figures 7 and 8 show the results of the *bit.ly* cases, and the results were similar to the *goo.gl* cases. US users also have lower precision and higher FPR than others. The overall accuracy of the system was lower with *bit.ly* cases than with *goo.gl* cases, because *goo.gl* offers four types of information in the click analytics, whereas *bit.ly* offers only two types of information, namely, the Referrers and the Countries, as mentioned in Section 2.2. The system had to infer URL clicks based on less infor-

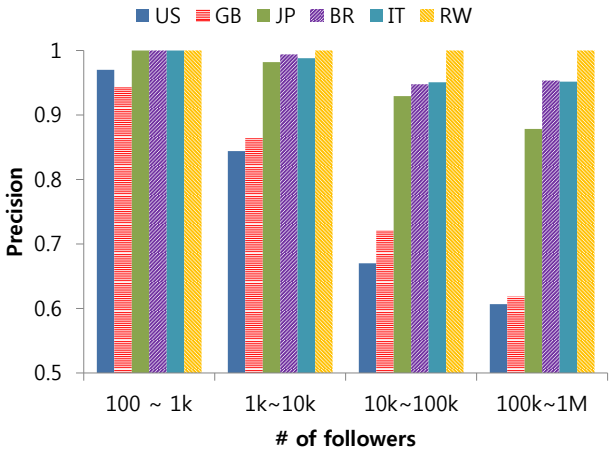

Figure 7: The precision of *bit.ly* URLs in terms of country. X axis means the number of followers of the updating users.

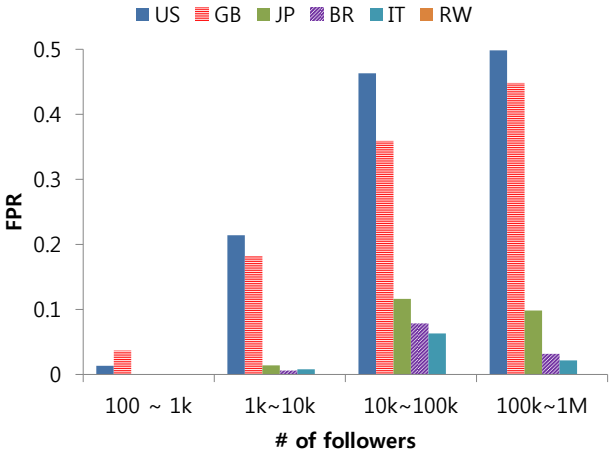

Figure 8: The FPR of *bit.ly* URLs in terms of country. X axis means the number of followers of the updating users.

mation. The total average precision was 0.87 and the total average FPR was 0.16.

4.2.3 *Discussion*

The most influential factor that affected the accuracy of the system is the number of followers who follow the same Twitter user and who have the same information as the target user. If no other user had the same information as the target user, the system could infer perfectly regardless of the number of the posting user's followers. In fact, most of the URLs clicked on by the Rwanda users were successfully inferred by the system regardless of the number of followers and platforms. In contrast, the system had the lowest accuracy if the target user lived in the US and used an iPhone. The user who lived in US and used an iPhone had the lowest precision with 0.81 and the highest FPR with 0.28. It means that even in the worst case our system has high performance. In general, the system successfully inferred the URLs clicked on by the target users with a high precision and a low FPR.

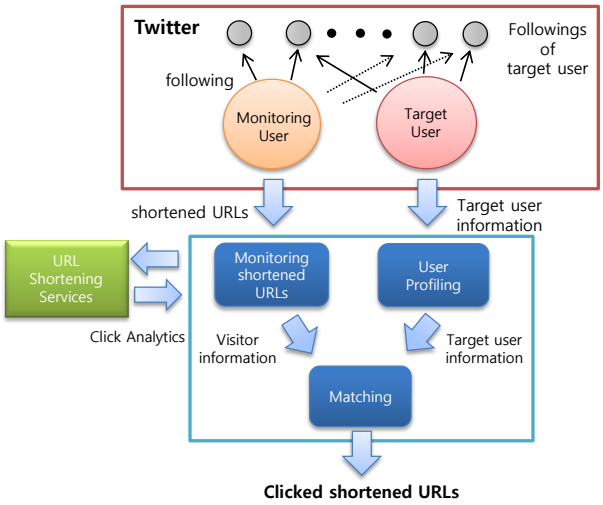

Figure 9: Overall architecture of the attack in the real world

5. INFERENCE ATTACK IN THE REAL WORLD

In this section, we introduce the inference attack in the real world. The system identifies whether a Twitter user clicked a shortened URL that were posted by his or her followings or not.

We selected a number of Twitter users as our target users. Our goal was to identify the shortened URLs that were clicked on by a target user. The result of this attack is a set of URLs that could have been clicked on by a target user. The procedures of this attack system are as follows:

1. The system selects a target Twitter user who follows some accounts that post shortened URLs.
2. The system monitors the click analytics of all shortened URLs that are posted by the followings of the target user.
3. When the system notices changes in the click analytics, which indicates a new visitor to the shortened URL, the system extracts the visitor's information from the click analytics.
4. The system compares the information about the visitor with known information the target user. If both pieces of information match, it infers that the shortened URL was clicked on by the target user.

Figure 9 shows the overall architecture. The architecture consists of three modules: profiling, monitoring, and matching. The profiling module gets the information of the target user from the target user's profile and timeline, as mentioned in Sections 3.2 and 3.3. We created a Twitter user (monitoring user) who followed all the followings of the target user in order to access all tweets that might be viewed by the target user. The monitoring module extracts the shortened URLs from the tweets posted by the followings of the target user and monitors the changes in the click analytics of those shortened URLs. When the monitoring module notices the change, which indicates a new visitor to the shortened URL, the matching module compares the information of the visitor

with information about the target user. After the matching procedure, all shortened URLs that were clicked on by visitors with the same information as the target user will be included in a set of candidate URLs.

We identify a set of candidate URLs that could be visited by the target user whenever shortened URLs are clicked. The candidate URLs, however, may not be accurate because other Twitter users who have the same information as the target user could click on the candidate URLs. There are many Twitter users who have received the same shortened URLs seen by the target user. All the followers of that user who has sent the shortened URLs to the target user receive the same shortened URLs. Among them, someone who has the same information with the target user may click on a shortened URL that is being monitored by our system. The system could mistakenly conclude that the shortened URL was clicked on by the target user. However, the probability that the clicks are from the target user was significant.

On the other hand, it is also possible that the candidate set might not include a shortened URL that is clicked on by the target user, particularly if the target user clicks on the shortened URL in an unusual environment that is atypical for that user. For example, if a target user typically uses an iPhone in the USA, our system would only monitor changes of click analytics involving the iPhone and the USA. However, it is possible for the target user to change his smart phone or to use a personal computer for using Twitter. If he clicks on the shortened URLs in such environments, our system cannot notice those events. However, this kind of situation temporarily occurs because if the target user posts a tweet using the new environment at least once, the profiling module will add the new environment into his profile information. Therefore, we can successfully identify the user's information and perform the inference attack with high accuracy.

5.1 Target User Selection

The main goal of the attack is to identify the shortened URLs that are clicked on by a target user. There are a number of criteria used to select target users for our experiments. First, we needed to select the target users whose exact information could be identified by us. Their profile must be public and they must use well-known applications, such as the official Twitter applications for the smartphone. Second, the target user must follow some users who post shortened URLs frequently because we want to obtain enough experimental results. If no shortened URL appears in the timeline of the target user, we cannot attempt an attack. Third, the target users must actively use Twitter. If we select an inactivate user as a target user, we cannot obtain enough experimental data. Our ideal target users are Twitter users who frequently check their timeline and click on URLs on their timeline. Another important condition of a target user is that the user needs to post or retweet a tweet that includes the shortened URLs that he clicked on. We assume that we successfully inferred the click on the shortened URLs if the target user posts a tweet with the shortened URL that is one of the URLs in a candidate set. However, the criteria listed above are used only to obtain enough experimental data and to conduct evaluation, which will be covered later in the Section 5.3. They are not strongly related to the accuracy of the attack. Any Twitter user who can be identified by an attacker could be a target user.

In order to find qualified target users for the experiments, we manually searched *goo.gl* or *bit.ly* strings on Twitter and reviewed the user's timeline.

5.2 Data Collection

First, we crawled Twitter data using two sets of Twitter API methods: Streaming APIs and REST APIs. The Streaming APIs enable us to monitor target users in real time. We used the REST APIs for crawling profile pages, timelines, followers, and followings. However, the REST APIs have a rate limit: a host is permitted 150 requests per hour. In order to overcome the rate limit, we changed the IP address of the crawling servers when the servers exceeded the rate limit. We used 10 servers and 100 IP addresses to crawl Twitter data. Second, we crawled the click analytics of the shortened URLs as mentioned in Section 4.2.

We selected 27 target users and crawled their profiles, timelines, and favorites. We monitored 2,278 *goo.gl* URLs and 25,816 *bit.ly* URLs. The collection lasted for about two months from March to April 2012.

5.3 Evaluation

As mentioned in Section 4, it is difficult to evaluate the system properly. Therefore, we use a different method to evaluate our system. We assume that if a URL is included in the tweets or favorites of a Twitter user, the Twitter user had already visited them. To validate the correctness of our inference that a user visited a URL, we checked whether the user included the same URL in his tweets or favorites in the near future.

To clarify, suppose that our system inferred that Twitter user A visited the shortened URL B. We collect the timeline and the favorites of user A and check whether a tweet containing the shortened URL B exists. If we find the shortened URL B in the timeline or favorites, then we are certain that the system successfully infers the shortened URL visited by the candidates.

We computed three probabilities as follows:

$$\mathbf{P1} = \frac{|\mathbf{U} \bigcap \mathbf{RT}|}{|\mathbf{U}|},$$

$$\mathbf{P2} = \frac{|\mathbf{C}_{urls} \bigcap \mathbf{RT}|}{|\mathbf{C}_{urls}|},$$

$$\mathbf{P3} = \frac{|\mathbf{N}_{urls} \bigcap \mathbf{RT}|}{|\mathbf{N}_{urls}|}.$$

Let $\mathbf{U}$ be a set of all shortened URLs that are posted by followings of the target user. $\mathbf{U}$ is classified into two sets $\mathbf{C}_{urls}$ and $\mathbf{N}_{urls}$ where $\mathbf{C}_{urls}$ is a set of shortened URLs inferred as visited by the target user, candidate URLs set, and $\mathbf{N}_{urls}$ is a set of shortened URLs inferred as unvisited by the target user. $\mathbf{RT}$ is a subset of $\mathbf{U}$ that includes the shortened URLs which are in the target user's timeline including retweeted or favorited by the target user.

The resulting probabilities were as follows: **P1** was 0.032, **P2** was 0.048, and **P3** was 0.003. **P2** was 1.5 times higher than **P1** and 16 times higher than **P3**. This implies that we can successfully categorize all shortened URLs into a set of visited URLs and a set of unvisited URLs. The target users normally posted tweets containing shortened URLs that are included in the candidate URLs set. They rarely posted

	# of shortened URLs	**RR**
goo.gl	2,278	0.584
bit.ly	25,816	0.674
Total	28,094	0.669

Table 5: The monitored shortened URLs and RR for each URL shortening services in the real world attack

tweets with shortened URLs outside the candidate URLs set. According to boyd et al. [4], about 3% of tweets are likely to be retweets. That percentage was similar to our calculation of **P1** which was 0.032; therefore, the value of **P1** is also trustworthy.

To view the results from a different angle, we also calculated two other metrics.

$$\mathbf{P4} = \frac{|\mathbf{C}_{urls} \bigcap \mathbf{RT}|}{|\mathbf{RT}|}.$$

$$\mathbf{P5} = \frac{|\mathbf{N}_{urls} \bigcap \mathbf{RT}|}{|\mathbf{RT}|}.$$

P4 indicates the fraction of candidate URLs that are in **RT**, and **P5** indicates the fraction of non-candidate URLs are in **RT**. The results were as follows: **P4** was 0.952 and **P5** was 0.048. We found that **P4** was much higher than **P5**. Most of the shortened URLs that are in the timeline or favorites of the target users were inferred as candidate URLs. Therefore, we can say with confidence that a shortened URL is highly likely to be retweeted or favorited by the target user if it is included in the candidate set.

We also computed the reduction ratio **RR**, which represents how much we reduced the number of candidate URLs from the number of all shortened URLs posted by the followings of the target user. **RR** is computed as follows:

$$\mathbf{RR} = \frac{|\mathbf{C}_{urls}|}{|\mathbf{U}|}.$$

RR depends on click tendency of the target users. When the target user clicks on all of the shortened URLs in **U**, **RR** becomes 1. Therefore, a higher **RR** does not always indicate that the system is performing poorly. Table 5 shows the results for each URL shortening service. The average value of the reduction ratio is 66.9%. This means that our system inferred that the target users clicked on 66.9% of the shortened URLs posted by their followings. The reduction ratio in the *goo.gl* case is lower than in the *bit.ly* case, because *goo.gl* provides more information than *bit.ly* in the click analytics. Since the number of *bit.ly* shortened URLs is fairly larger than that of *goo.gl* on Twitter, we have a larger number of *bit.ly* shortened URLs than that of *goo.gl* shortened URLs.

6. DISCUSSION

6.1 Limitations

Our inference attack method has some limitations due to the restrictions in the given information. We cannot guarantee the correctness of the given location information because some users do not reveal their exact location information on Twitter. Moreover, the given browser and platform information is also restricted because some client applications do not reveal the exact platforms that they use. Even when we are able to identify specific Twitter users, many users have the same information as the identified Twitter users have. Therefore, the results of inference cannot be 100% guaranteed. However, with more information about the target users, the accuracy of our system will improve. For example, if we know when the target user frequently uses Twitter, we can further reduce the number of the candidates. One way to infer this timeframe is by analyzing the time history of the target user's tweets. We will use this time history for future work. Further, if we could obtain information about a target user from different channels (e.g., if we are personally acquainted with the target), we could increase the probability of succeeding with our inference attack.

6.2 Countermeasures

We only need public information provided by Twitter and the URL shortening services. Therefore, the published information must be changed to prevent our inference attacks. A simple measure of prevention is by delaying the update to the click analytics of shortened URLs. If the click analytics is updated every minute or every tens of minutes, the changes of the click analytics would more likely include a larger number of click events, so that inference attacks would have difficulties in differentiating an individual from the group of click events. In addition, providers could add noise information to the click analytics in order to prevent exact inference, as the differential privacy does [9].

6.3 Applications

Using our inference attack method, attackers can determine the URLs that the target user visited. Based on the visited URLs, the attackers could infer the target user's preferences, such as music interests, political inclination, or favorite products. This information could be used for targeted marketing or targeted spamming. Moreover, we discovered that it is very easy to cyber-stalk on Twitter. Anyone can stalk a target user by creating a Twitter account that follows everyone whom the target user follows (if the target user is not a private user). This way, the attacker receives the same tweets that appear in the target user's timeline.

Some active inference attacks are also possible. We did inference attacks after we identified the information of the target user. On the contrary, we can use our inference attacks to obtain information about the target user. If an attacker creates a shortened URL and sends the shortened URL to the target user, who then clicks on the shortened URL, the attacker can obtain information, such as the target user's current location and platform, from the click analytics.

7. RELATED WORK

7.1 Browser History Stealing

There are several types of history stealing attacks. First, the cached data of the browser was used for sniffing browser history [10, 14, 15]. There is a time difference between retrieving cached resources and retrieving non-cached resources. The attackers can know which pages were visited by analyzing the differences in latency. DNS cache was also used for history stealing attacks [10, 11, 18]. In general, most of the history stealing attacks are based on Cascading Style Sheet (CSS) visited styles [3, 8, 16, 17]. They use the fact that browsers display visited links differently from unvisited

links. These history stealing attacks assume that victims visit a malicious web page or that victims are infected by malware. However, our inference attacks do not need to make these assumptions. The inference attacks only use the combinations of publicly available information. Therefore, anyone can be an attacker, and anyone can also be a victim.

7.2 Privacy Leaks from Public Information

Many previous studies proposed attack techniques that cause privacy leaks in social networks, such as inferring private attributes or de-anonymizing users. Most of these studies used public information to infer hidden information. Some studies combined information from several different data sets. First, there are studies introducing de-anonymzing attacks in social networks. Backstrom *et al.* [2] tried to identify edge existence in anonymized network and Narayanan and Shmatikov [21] identified Netflix records of known users using only a little bit of data about the users. Furthermore, they combined their results with IMDb data and inferred user's political preferences or religious view. Narayanan and Shmatikov [22] also proved that users who have accounts in both Twitter and Flickr can be recognized in the anonymous Twitter graph. Wondracek *et al.* [29] proposed the de-anonymized attack using group membership information obtained by browser history stealing attack. There are also studies inferring private attributes of users in the social networks. He *et al.* [12] and Lindamood *et al.* [19] built a Bayesian network to predict undisclosed personal attributes. Zheleva and Getoor [30] showed how an attacker can exploit a mixture of private and public data to predict private attributes of a target user. Similarly, Mislove *et al.* [20] inferred the attributes of a target user by using a combination of attributes of the user's friends and other users who are loosely (not directly) connected to the target user. Calandrino *et al.* [5] proposed algorithms inferring customer's transactions in the recommender systems, such as Amazon and Hunch. They combined public data of the recommender systems and some of the transactions of a target user in order to infer the target user's unknown transactions. Chaabane *et al.* [6] proposed an inference attack to predict undisclosed attributes by using only music interests. They derived semantics using Wikipedia ontology and measured the similarity between users.

8. CONCLUSION

In this paper, we proposed an inference attack that infers shortened URLs that are clicked on by the target user. All the information needed in our attack is public information; that is, the click analytics of URL shortening services and Twitter metadata. Both information are public and can be accessed by anyone. We combined two pieces of public information with inferred candidates. To evaluate our system, we crawled and monitored the click analytics of URL shortening services and Twitter data. Throughout the experiments, we have shown that our attack can infer the candidates in the majority of cases. To the best of our knowledge, this is the first study that infers URL visiting history on Twitter. We also proved that if an attacker knows some information about the target user, he could determine whether the target user clicks on the shortened URL.

9. ACKNOWLEDGEMENTS

This research was supported by World Class University program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea (R31-10100). Also, this research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency). (NIPA-2012-H0301-12-3002)

10. REFERENCES

[1] geonames. `http://www.geonames.org/export/client-libraries.html`.

[2] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *WWW*, 2007.

[3] D. Baron. :visited support allows queries into global history, 2002. `https://bugzilla.mozilla.org/show_bug.cgi?id=147777`.

[4] D. boyd, S. Golder, and G. Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *HICSS*, 2010.

[5] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov. "you might also like:" privacy risks of collaborative filtering. In *IEEE Security and Privacy*, 2011.

[6] A. Chaabane, G. Acs, and M. A. Kaafar. You are what you like! information leakage through users' interests. In *NDSS*, 2012.

[7] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: A content-based approach to geo-locating twitter users. In *ACM CIKM*, 2010.

[8] A. Clover. Css visited pages disclosure, 2002. `http://seclists.org/bugtraq/2002/Feb/271`.

[9] C. Dwork. Differential privacy. In *ICALP*, 2006.

[10] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. In *ACM CCS*, 2000.

[11] L. Grangeia. Dns cache snooping or snooping the cache for fun and profit. In *SideStep Seguranca Digitial, Technical Report*, 2004.

[12] J. He, W. W. Chu, and Z. V. Liu. Inferring privacy information from social networks. In *ISI*, 2006.

[13] B. Hecht, L. Hong, B. Suh, and E. H. Chi. Tweets from justin bieber's heart: The dynamics of the location field in user profiles. In *ACM CHI*, 2011.

[14] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell. Protecting browser state from web privacy attacks. In *WWW*, 2006.

[15] M. Jakobsson and S. Stamm. Invasive browser sniffing and countermeasures. In *WWW*, 2006.

[16] A. Janc and L. Olejnik. Feasibility and real-world implications of web browser history detection. In *W2SP*, 2010.

[17] A. Janc and L. Olejnik. Web browser history detection as a real-world privacy threat. In *ESORICS*, 2010.

[18] S. Krishnan and F. Monrose. Dns prefetching and its privacy implications: When good things go bad. In *USENIX LEET*, 2010.

[19] J. Lindamood, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. Inferring private information using social network data. In *WWW*, 2009.

[20] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel. You are who you know: Inferring user profiles in online social networks. In *WSDM*, 2010.

[21] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse dataset. In *IEEE Security and Privacy*, 2008.

[22] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Security and Privacy*, 2009.

[23] Semiocast. Twitter reaches half a billion accounts more than 140 millions in the u.s., 2012. `http://semiocast.com/publications/2012_07_30_Twitter_reaches_half_a_billion_accounts_140m_in_the_US`.

[24] Twitter blog. Links and twitter: Length should't matter, 2010. `http://blog.twitter.com/2010/06/links-and-twitter-length-shouldnt.html`.

[25] Twitter blog. One million registered twitter apps, 2011. `http://blog.twitter.com/2011/07/one-million-registered-twitter-apps.html`.

[26] Twitter blog. Shutting down spammers, 2012. `http://blog.twitter.com/2012/04/shutting-down-spammers.html`.

[27] Twitter developers. t.co redirection behavior, 2012. `https://dev.twitter.com/docs/tco-redirection-behavior`.

[28] Twitter developers. The t.co url wrapper, 2012. `https://dev.twitter.com/docs/tco-url-wrapper`.

[29] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users. In *IEEE Security and Privacy*, 2010.

[30] 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 *WWW*, 2009.