Poster: I Know Where You've Been: Geo-Inference Attacks via the Browser Cache

Yaoqi Jia*, Xinshu Dong†, Zhenkai Liang *, Prateek Saxena*
*Department of Computer Science, National University of Singapore. {jiayaoqi, liangzk, prateeks}@comp.nus.edu.sg
*Advanced Digital Sciences Center, Singapore. xinshu.dong@adsc.com.sg

***Abstract*—Many websites customize their services according to different geo-locations of users, including Google, Craigslist, etc. Recently, mobile devices further allow web applications to directly read users' geo-location information from GPS sensors. However, if such websites leave location-sensitive content in the browser cache, other sites can sniff users' geo-locations through side channels. In this paper, we demonstrate that such geo-location leakage channels are widely open in popular web applications today. With *geo-inference attacks* that measure the timing of browser cache queries, attackers can infer users' countries, cities, and neighborhoods.**

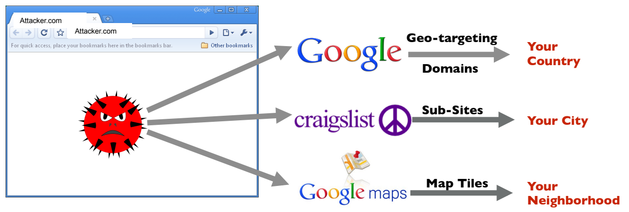

Fig. 1: Geo-inference attacks locate the victim's geo-location.

I. Introduction

Geo-location is a type of privacy-sensitive information. Websites have strong interests in obtaining users' geo-location information to provide personalized services. On the other hand, web attackers misuse victims' geo-locations for spear phishing, or user tracking. Geo-location leakage can cause tremendous damage to the user's privacy. That is why modern browsers disable the access to geo-location information by default.

Websites can identify users' locations from their IP addresses. However, IP addresses are unreliable in many ways. IP address-based geo-location tracking is unreliable for mobile networks. Moreover, users may intentionally use anonymization services, e.g., VPN, to hide their original IP addresses. IP address-based geo-location can be easily bypassed using anonymization services.

Recent advancement in mobile devices enables websites to obtain geo-location information from GPS sensors. Nevertheless in mobile browsers, it requires a user's explicit permission to access GPS data. In this work, we show how known side-channels can be utilized to conduct geolocating attacks, i.e., identify a user's geo-location with high accuracy without direct access to GPS. We term such attacks *geo-inference attacks via the browser cache*.

II. Problem Definition

A. Threat Model

The adversary in a geo-inference attack is a standard *web attacker*, who controls at least one web server and hosts a malicious site, e.g., *https://attacker.com*. To advertise the malicious site, the attacker can promote the popularity of the site via Search Engine Optimization (SEO), and disseminate its shortened URL through emails, social networks and advertisements. We assume that Alice is an ordinary web user who does not clear browser cache frequently, and she usually visits geo-location-oriented sites, e.g., Google, Craigslist and Google Maps, to avail herself of the local location-oriented services. When Alice visits the attacker's site (e.g., *https://attacker.com*) in her browser, she denies the unfamiliar site's access to the device's location.

B. Geo-Inference Attacks via the Browser Cache

As a typical case of attacks on browsing history, Wondracek et al. de-anonymize social network users by analyzing users' visited URLs [1]. In this work, we show the implication of timing attacks [2] in de-anonymizing the user's geo-location. In geo-inference attacks, the attacker makes guesses on the victim's geo-location, and then queries the cached URLs corresponding to the geo-location. As Figure 1 shows, by utilizing the location-sensitive resources left by location-oriented sites in the browser, geo-inference attacks provide an oracle for attackers to infer a user's country, city, and neighborhood.

III. Results

A. Locate Your Country

In order to infer a user's country, we check which page out of 191 Google's domains hits cache. We utilized Google's logo image, whose URL consists of Google's geo-targeting domain, e.g., *https://google.co.jp*, and */images/srpr/logo11w.png* to specify Google's website.

We measured the image load time of Google logos from Google's 191 geo-targeting domains to determine the country of the victim. We found that the image load time of each Google's domain without cache is much larger than the time with cache. Listing 1 is the code to measure the image load time. We can also utilize Cross-origin resource sharing (CORS) to send XMLHttpRequests, measure the requests load time for logos, and determine which country the victim lives in. As for the <img> tag in HTML, we utilized the *complete* property, which can indicate whether Google's logo is cached or not in Firefox, Safari, and IE, to locate the user's country, shown in Listing 2.

```
var image = document.createElement('img');
image.setAttribute('startTime', (new Date().getTime
    ()));
image.onload = function() {
    var endTime = new Date().getTime();
    var loadTime = endTime - parseInt(this.
        getAttribute('startTime'));
    ......}
```

Listing 1: Measuring the image load time with JavaScript

```
function cached(url) {
    var image = document.createElement('img');
    image.src = url;
    return image.complete || image.width+image.
        height > 0;}
```

Listing 2: Distinguishing whether the image is cached or not

B. *Locate Your City*

Several websites offer content specific to cities, e.g., Craigslist is a large classified advertisements website that covers 712 cities on the world. Since all the sub-sites are city-oriented and can be loaded in frames, we measured the page load time to locate the user's city. As Figure 2 shows, the attacks can reliably distinguish which city-specific page is cached in the user's browser.

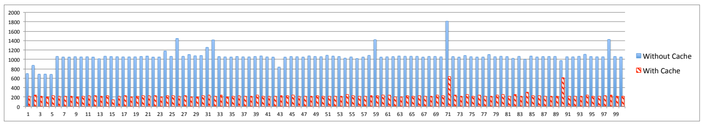

Fig. 2: The difference between the page load time (in millisecond) of 100 Craigslist sites without cache (> 1000 ms) and that with cache (≈ 220 ms)

C. *Locate Your Neighborhood*

To extend our study to finer granularity than city level, we leveraged the cached resources from online map service websites, e.g., Google Maps. In fact, the maps are tessellations of map tiles, and each tile localizes at a predicable URL. As Figure 3 shows, we can derive the coordinates (12627, 23720) of Grand Loop Rd from the requesting URL. Though the coordinates are not the real geographic coordinates, we can predict the URLs of one specific area's map tiles corresponding with the coordinates. As Google Maps usually zooms in to the user's current location when the user starts to browse Google Maps in the browser, by sniffing the victim's recently visited map tiles, we can locate the victim's streets and neighborhood.

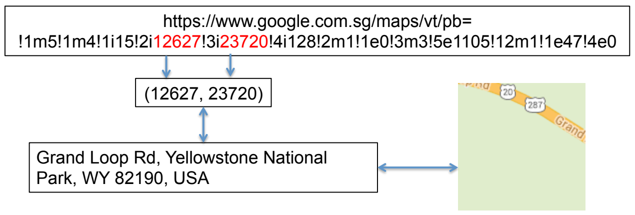
https://www.google.com.sg/maps/vt/pb=
!1m5!1m4!1i15!2i12627!3i23720!4i128!2m1!1e0!3m3!5e1105!12m1!1e47!4e0
(12627, 23720)
Grand Loop Rd, Yellowstone National Park, WY 82190, USA
20
287
Grand

Fig. 3: URLs of map tiles statically depend on geo-locations in Google Maps.

TABLE I: Geo-inference attacks on browsers

	I	II	III	IV	V
Chrome (Linux, Windows & OS X + Android & iOS)	✓	✓	–	✓	✓
Firefox (Linux, Windows & OS X + Android)	✓	–	✓	✓	✓
Safari (Windows & OS X + iOS)	✓	–	✓	✓	✓
Opera (Linux, Windows & OS X + Android & iOS)	✓	✓	–	✓	✓
IE (Windows)	✓	–	✓	✓	✓

I : Locate Your Country with Image Load Time
II : Locate Your Country with Cross-Origin Resource Sharing
III: Locate Your Country with <img>'s *complete* Property
IV: Locate Your City with Craigslist
V : Locate Your Neighbourhood with Google Maps
✓ : Support – : Not Support

D. *Affected Browsers and Websites*

We further study the browsers and websites affected by geo-inference attacks. As Table I shows, all five mainstream browsers (Chrome, Firefox, Safari, Opera and IE) as well as TorBrowser [1] are susceptible to geo-inference attacks. Resorting to VPN (Hotspot Shield), we visited 55 Alexa Top 100 websites (not highly related to specific countries) in five different countries (USA, UK, Australia, Japan and Singapore) and recorded all the URLs of cached resources for each website. We found that 62% of websites contain location-sensitive resources, which can be used in geo-inference attacks.

IV. Solution

Security researchers have proposed various defenses against browser cache sniffing. Jackson et al. propose a defense solution by segregating browser cache based on the same-origin policy [3], thus limiting web attackers from observing timing differences between accessing cached and non-cached cross-origin web resources. Through a prototype of the same-origin caching policy on Chromium, we found that the policy triggers more than 50% performance overhead among Alexa Top 100 websites. A more balanced solution is to segregate location-sensitive resources instead of cacheing them in cache shared across all websites.

References

[1] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel, "A practical attack to de-anonymize social network users," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 223–238.

[2] E. W. Felten and M. A. Schneider, "Timing attacks on web privacy," in *Proceedings of the 7th ACM conference on Computer and communications security*. ACM, 2000, pp. 25–32.

[3] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell, "Protecting browser state from web privacy attacks," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 737–744.

[1] We run the experiments on TorBrowser 3.5.2.1, which is based on Firefox 24.3.0.