CSE509 Systems Security

A brief introduction to online tracking

Nick Nikiforakis
nick@cs.stonybrook.edu
“On the Internet, nobody knows you’re a dog.”
I need state!

• HTTP is a stateless protocol
  – The server does not know that two or requests originate from the same user

• No state -> No Personalization
  – No e-banking, e-shops, webmail, etc.

• Solution: Cookies!
<table>
<thead>
<tr>
<th>Domain</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>stonybrook.edu</td>
<td>username=nick; Date=30/09/2014;</td>
</tr>
<tr>
<td>google.com</td>
<td>g1=190233213; g2=afasfdioujewf;</td>
</tr>
<tr>
<td>slashdot.org</td>
<td>adhese_count=2; bcn=e4f5d957-00;</td>
</tr>
</tbody>
</table>
Ask Tom: Rants, raves and questions about DC dining scene

Nov 18, 2015

Washington Post food critic Tom Sietsema entertains your dining questions, rants and raves.

Find all of Tom Sietsema’s Washington Post work at washingtonpost.com/tomsietsema.

Q: The DC Food Scene From Afar

Tom – I’ve thoroughly enjoyed your America’s Best Food Cities series, and you seem to be having fun writing it. I’m curious what the restaurant communities in those cities think about DC. Is our restaurant explosion on the national food radar, or is the perception still that we’re all steak houses and all K Street? Do you find yourself defending/explaining the DC food scene to reviewers and restauranteurs in those cities?
A cookie’s life

red.com

ads.com

blue.com
3rd Party Tracking

• “Suddenly” all sorts of websites that you’ve never heard about, can create a browsing profile of you and sell it to advertising companies
  – quantserve.com
  – scorecardresearch.com
  – addthis.com
Flash Cookies

• The Flash plugin has the ability to store information in Local Shared Objects (LSOs) that are separate from HTTP cookies
  – Commonly known as Flash cookies

• These LSOs are there to remember things about the user in the context of a Flash application, like remembering user preferences, and game progress
Flash Cookies

• Flash cookies were the first technology that was used to supplement the traditional cookie-based third-party tracking

• Originally, when a user deleted his cookies, that would not have an effect on Flash cookies
  – Multiple stories started appearing in 2009 and 2010 where researchers and investigative journalists were finding websites that were using Flash cookies to remember
  – Flash cookies were often used to respawn traditional HTTP cookies, a practice which was named “zombie cookies”

• Modern browsers, collaborate with Flash so that when you “Delete all cookies” this now includes Flash cookies
Evercookies

• The idea of using Flash cookies to respawn traditional HTTP cookies was taken to the extreme with Samy Kamkar’s “evercookie”
  – https://github.com/samyk/evercookie

• “evercookie is a javascript API that produces extremely persistent, respawning cookies in a browser. Its goal is to identify a client even after they've removed standard cookies, Flash cookies (LSOs), HTML5 storage, SilverLight storage, and others.”
Evercookie

• The main idea is the same
  – Try to hide an identifier into as many “corners” of the browser as possible and rely on one copy to respawn the rest

• Methods used
  – Standard HTTP cookies
  – Flash LSOs
  – Silverlight Storage
  – HTML5 storage
  – HTML5 indexdb
  – HSTS Pinning
Tracking through HSTS Pinning

• HSTS is normally used to increase a user’s security by having your browser connect to your website over HTTPS even if the user asked to be connected over HTTP

• Last year, people realized that HSTS could be used for tracking
  – How?
Tracking through HSTS

1. Generate a tracking identifier of desired length, e.g. 32 bits (allowing for $2^{32}$ users)
2. Break that identifier down into binary
   
   $1 \quad 0 \quad 1 \quad 1 \quad \ldots \quad 0 \quad 1$

1. Have the user’s browser conduct 32 requests to different subdomains of your domain
   1. a.tracking.com
   2. b.tracking.com
   3. c.tracking.com
   4. Etc.
Tracking through HSTS (continued)

4. For every request sent, if the corresponding bit was 1, redirect to HTTPS and then send along an HSTS header telling the browser to remember that you want to be contacted only over HTTPS

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>...</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>a.tracking.com</td>
<td>→</td>
<td><a href="https://a.tracking.com">https://a.tracking.com</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>b.tracking.com</td>
<td>→</td>
<td><a href="http://b.tracking.com">http://b.tracking.com</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>c.tracking.com</td>
<td>→</td>
<td><a href="https://c.tracking.com">https://c.tracking.com</a></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Etc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. When the user returns, try to load all 32 domains again (over HTTP), and observe which requests will come to you over HTTPS (browser honoring your previous instructions communicated via the appropriate HSTS header)

1. https://a.tracking.com → 1
2. http://b.tracking.com → 0
3. https://c.tracking.com → 1

... ... 

5. Based on these requests you can now reconstruct the original identifier
Stateful tracking

• Stateful tracking is the most popular method for tracking users
  – Cookies (Browser and Flash)
  – HTML Storage
  – ETags
  – HTTP Strict Transport Security
    • Unrelated technologies that store state now become tracking vectors

• Regardless of the exact method, the idea is always the same
  – Find a nook in the browser where you can store and retrieve data
Users reacted...

- 1/3 of users delete first & third-party cookies within a month after they’ve been setup
- Multiple extensions revealing hidden trackers
  - Ghostery
  - Disconnect
- Private mode of browsers used to avoid traces of cookies from certain websites
Ghostery
PROBLEM = PROBLEM
Right?
Stateless tracking

• Stateless tracking was born out of a need
  – The need to track users, even if they delete their stateful identifiers
    • Clearing of cookies, Private Mode, 3rd Party tools, etc.
  
• Unlike stateful tracking, we are no longer searching for a nook to hide data

• We are searching for variability that is already present in the user’s browsing environment
  – Requirements: Relatively stable over time, different between different users
Web-based device fingerprinting

• Eckersley showed in 2010 that certain attributes of your browsing environment can be used to accurately track you

• These attributes, when combined, created a quite unique fingerprint of your system?
  – How?

• Other properties
  – Nothing to inspect
  – Very hard to opt-out
Properties fingerprinted by Panopticlick
How?

navigator.userAgent
navigator.appCodeName
navigator.appName
navigator.platform
navigator.product
navigator.productSub
navigator.vendor

screen.width
screen.height
screen.availWidth
screen.availHeight
screen.colorDepth
How?

navigator.plugins

(new Date()).getTimezoneOffset()
How?

• Fonts are trickier
  – No official JavaScript API to extract fonts

• Solution used by Eckersley
  – Use Flash plugin where there are font-discovering APIs

• Solution found later
  – Side-channel in JavaScript
Font Detection through JavaScript

<table>
<thead>
<tr>
<th>String</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_DO_NOT_NEED_FLASH</td>
<td>500 x 84</td>
</tr>
<tr>
<td>I_DO_NOT_NEED_FLASH</td>
<td>620 x 90</td>
</tr>
<tr>
<td>I_DO_NOT_NEED_FLASH</td>
<td>600 x 87</td>
</tr>
<tr>
<td>I_DO_NOT_NEED_FLASH</td>
<td>510 x 82</td>
</tr>
</tbody>
</table>

1. Create a large string for each font family that you want to test
2. Measure it’s width and height (offsetWidth, offsetHeight properties)
3. Compare it to the width and height of the Arial font family (typical fallback font when your browser does not have the one you asked for)
4. If dimensions are different, then the font-family installed
Resulting fingerprints

- 94.2% of the users with Flash/Java could be uniquely identified
- Simple heuristic algorithms could track updates of the same browser
Sites where you can test your fingerprint

• Panopticlick (platform of first paper quantifying the dangers of browser fingerprinting)
  – https://panopticlick.eff.org/ (Now changed to include more modern fingerprinting techniques)

• AMIUUnique
  – https://amiunique.org/ (Website using additional fingerprinting techniques)
Other proposed ways

• Eckersley paved the way of stateless tracking through fingerprinting

• After Eckersley, other researchers proposed ways of fingerprinting browsers, based on:
  – Speed
  – Implementation coverage
  – Rendering of elements
They will know you by your speed...

• Mowery et al. (W2SP 2011) proposed the use of performance benchmarks to tell different JavaScript engines apart
  – Different JavaScript engine -> Different browser

• Collected performance signatures (39 tests) from approx. 1000 users
  – 98.2% correct browser family detection
  – Overall accuracy (versions included): 79.8%
As well as your features...

• Mulazzani et al. (W2SP 2013) proposed the use of missing functionality in JavaScript engines
  – Different browsers, implement JavaScript standards, at a different rate

<table>
<thead>
<tr>
<th>Browser</th>
<th>Win 7</th>
<th>WinXP</th>
<th>Mac OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox 3.6.26</td>
<td>3955</td>
<td>3955</td>
<td>3955</td>
</tr>
<tr>
<td>Firefox 4</td>
<td>290</td>
<td>290</td>
<td>290</td>
</tr>
<tr>
<td>Firefox 5</td>
<td>264</td>
<td>264</td>
<td>264</td>
</tr>
<tr>
<td>Firefox 6</td>
<td>214</td>
<td>214</td>
<td>214</td>
</tr>
<tr>
<td>Firefox 7</td>
<td>190</td>
<td>190</td>
<td>190</td>
</tr>
</tbody>
</table>
As well as your artistic talent

- Mowery et al. (W2SP) proposed the use of the HTML5 canvas to detect browser-specific renderings of the same string
  - Write some text in canvas, read it out as an image
  - Different browsers/hardware combinations will create slightly different images

- [http://jsbin.com/ePAheCi/2/edit](http://jsbin.com/ePAheCi/2/edit)
What is fingerprinting used for?
Available defenses against online tracking

• Stateful tracking
  – Clearing cookies/Using browser’s Private Mode
  – Using multiple browser profiles for different activities
  – Setting up your browser to reject third-party cookies
  – Anti-tracking extensions that block specific scripts

• Stateless tracking
  – Anti-tracking extensions that block specific scripts
Problems with existing solutions

• Most popular anti-tracking browser extensions work with long lists of already known trackers

• These lists can be compiled:
  – Either by the company behind an extension (Ghostery)
  – Or crowdsourced by the community (EasyPrivacy list of Adblock)
  – Or a mix of the two

• The problem is that if a tracker is not part of those lists then it will not be blocked
  – New trackers are not covered
  – Trackers who change their domain names / names of scripts are not covered
Potential answers to these problems

• Tools like EFF’s PrivacyBadger attempt to “learn” in real time who is a tracker based on what information they exchange in a third-party fashion
  – No need for known list of trackers
  – Will not block third-party trackers who do not match their heuristics

• Other tools (typically academic prototypes) try to systematically lie by adding noise to your fingerprint so that each time that you are fingerprinted, your fingerprint is different than last time
Recap

• A user’s desire to browse privately is in direct contradiction with a website’s desire to monetize user visits
  – The more they know about users, the more targeted the ads, the higher the chance of a user interacting with an ad
• Tracking is either stateful or stateless
  – Many of the defenses against the former do not work against the latter
“Remember when, on the Internet, nobody knew who you were?”