Weird New Tricks for Browser Fingerprinting

yan (@bcrypt)
ToorCon 2015
real pic of me

also work on these things

HTTPS Everywhere
Privacy Badger
W3C
Now the Real Fight Against TPP Begins

JOIN THE TOR CHALLENGE

NSA
TAKE ACTION NOW
Oparise NSA Moss Spying!

STOP CISA

EFF staff photo, 2015
Come to the dark side...

BETTER TRACKING METHODS

...We have cookies
Tracking web users is all the rage

- Show ads!
- Inject QUANTUM malware
- Cybercatch cybercriminals
- Gather website analytics
- Detect fraud / droidnets
- Enforce paywalls
- etc.
A long time ago in a galaxy far, far away . . .

Obi-Wan tracked Luke using:

- cookies
- passive fingerprinting*
  (IP address, locales, user-agent, OS, etc.)
- sweet Jedi mind tricks

* In this presentation, fingerprinting == any non-cookie web tracking method.
THE ADBLOCKERS* STRIKE BACK

* In this presentation, adblocker == any tool that blocks web tracking (including non-advertising)
THE PHANTOM ADBLOCKER BLOCKERS

Matthew Keys
@MatthewKeysLive

Washington Post disables reading of articles for people with ad blocker software.

Your ad blocker is on.

Enter your e-mail address to unlock this story and subscribe to our First Reads newsletter.

Enter your email address  Go

I have read and agree to the Terms of Service and Privacy Policy.

Already a subscriber? Sign in
REVENGE OF THE ADBLOCKER BLOCKER BLOCKERS!!!
A New Hope: Browser Fingerprinting

- Evade blocking algorithms that blacklist domains based on cookie frequency (ex: Privacy Badger).
- Track users who disable 3rd party cookies (ex: Safari).
- Harder to delete than cookies.
- Can reveal new information about a user.
new web features ==
new fingerprinting techniques
Main Page

From Wikipedia, the free encyclopedia.

Welcome to Wikipedia, a collaborative project to produce a complete encyclopedia from scratch. We started in January 2001 and already have 44027 articles. We want to make over 100,000 complete articles, so let's get to work! Anyone, including you, can edit any article right now, without even having to log in. You can copyedit, expand an article, write a little or write a lot. See the Wikipedia FAQ for more background information about the project, and the help page for information on how to use and contribute to Wikipedia.
HOLY SHIT I HAVE 4 LIGHTSABERS ZOMG!!1

- active fingerprinting (HTML5 canvas, clock skew, installed fonts & plugins, WebRTC...)
- supercookies (Flash cookies, caches, HSTS, etags...
Fingerprinting attacks in the wild

WebRTC being used now by embedded 3rd party on nytimes.com to report visitors' local IP addresses.
Analytics: Cookie Leakage

Use cookies to identify Tor users when they are not using Tor

- Current: preliminary analysis shows that some cookies “survive” Tor use. Depends on how target is using Tor (Torbutton/Tor Browser Bundle clears out cookies).
- Goal: test with cookies associated with CT targets
  - Idea: what if we seeded cookies to a target?
  - Investigate Evercookie persistence

geez thx a lot Samy
#realtalk
How would you track a paranoid user who clears cookies & uses an adblocker?
Could fingerprint them, but adblockers & browsers will get better at blocking you...
...unless blocking causes too much collateral damage.
Collateral:

Privacy-conscious users usually care about security.
Can we fingerprint them using security features that are too important for them to turn off?
Trick #1: Abuse HTTP Public Key Pinning
HPKP (RFC 7469)

**Server:** One of these hashes must be in the TLS cert chain you receive from me.

**Browser:** DOPE!! NEXT TIME I SEE YOU I WILL CHECK IT BEFORE I WRECK IT
Public-Key-Pins:

max-age=3000;
pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZwM=";
pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLvKB9+xcprMF+44U1g=";
report-uri= “http://example.com/report”;
includeSubdomains;

How long to cache this shit for SHA-256 of a pub. key in the cert chain. Browser checks & caches this.

SHA-256 of a backup pub. key (required). Must NOT be in the cert chain. Browser caches this.

POST endpoint to report pin validation failures (optional).

Whether to pin for the host’s subdomains as well (optional).
Supercookie #1: fake backup pins

1. https://example.com sets a unique backup pin for each user + includeSubdomains + report-uri.
2. <img src="https://bad.example.com"> serves a chain that deliberately fails pin validation.
3. A validation failure report is sent which includes a unique cached backup pin!
Trick #2: Abuse HTTP Strict Transport Security + Content Security Policy
HSTS (RFC 6797)

Server: Hey, I just met you, and this is crazy, but please only call me over HTTPS for the next 604800 seconds.

Browser: OK
Strict-Transport-Security: max-age=3000;
includeSubdomains;

How long to remember to only connect to this host via HTTPS

Whether subdomains should also only be connected to over HTTPS (optional).
Supercookie #2: HSTS cache state

1. sneaky.com wants to fingerprint users.
2. example.com is known to support HSTS.
What happens then?

Case 1: Browser has never visited example.com

  -> makes a network round-trip, gets 301/302 to https://example.com

Case 2: Browser visited example.com before.

  -> HSTS causes an “internal” redirect (307) to https://example.com/ ~immediately
If we can measure the HTTP to HTTPS redirect latency, we can distinguish Case 1 from Case 2!
Q: How do we measure that?
A: Abuse one more browser security feature.
Content Security Policy (W3C spec)

Server: For your safety, please only allow resources of type <X> from origins <A> & <B> while on this page.

Browser: I GOT U FAM
Content-Security-Policy:

- `img-src: https://*;` - Allow images to load from HTTPS origins only.
- `script-src: ‘self’ *, scripts.com cdn.example.com` - Allow scripts to load from the page’s origin, *.scripts.com, and cdn.example.com only.
The Missing Ingredient:
Set CSP to ‘img-src http:///*’

HTTPS image requests are blocked and fire an error event to JS listeners.
Why is this useful?

1. JS only lets us listen for `img` `onerror` and `onload` events. Turns out CSP violation triggers `onerror` consistently and early in the fetch pipeline.

2. If browser ever completes a request for `https://example.com`, it will get the HSTS pin and future results are polluted. CSP prevents this from happening!
After setting CSP:

Case 1: Browser has never visited example.com

  -> makes network request, gets 301/302 to https://example.com, img onerror fires.

Case 2: Browser visited example.com before.

  -> HSTS rewrites src to https://example.com/~immediately, img onerror fires.
How long does the HTTP to HTTPS redirect take?

Case 1: Browser has never visited example.com

  -> Order of 100ms depending on network latency and site response time.

Case 2: Browser visited example.com before.

  -> Order of 1ms, independent of the site and network conditions.
Putting it all together
Remember the CSS visited-selector bug?

CSS History Sniffing

- Determine user’s browsing habits with CSS
- Visited link different than non-visited link
- CSS and element inspection determines visited pages
- Issued fixed March 2010

```javascript
if (getComputedStyle(link, '').color == "rgb(0, 0, 128)") {
    // link.href has not been visited
} else {
    // link.href has been visited
}
```

http://dbaron.org/mozilla/visited-privacy
That was soooo 2010

New plan:

1. Scrape Alexa Top 1M for hosts that send HSTS and aren’t preloaded.
2. Load all the HSTS hosts asynchronously on one page.
3. Measure the onerror timing & separate hosts into visited and unvisited.
Turns out...

Redirect timing is hard to measure accurately for 300+ async image loads at once.

Improved by calibrating timing drift using a request to a preloaded HSTS host every other request.

Chrome still had many false positives; confirmed timings for positive results using synchronous loads.
demo:
http://zyan.scripts.mit.edu/sniffly
Sites you’ve probably visited:
- angel.co
- www.xoom.com
- www.ocadux.com
- www.ipnja.com
- vine.co
- www.hackerrank.com
- www.ing-diba.de
- www.yola.com
- www.wealthfront.com
- noscript.net
- notepad-plus-plus.org
- www.ing.al
- www.6ixt.com
- mhsura.io
- www.kickstarter.com
- giustizia.it
- www.adtractor.com
- creativemarket.com
- upjex.com
- www.nabobbank.nl
- mscashembank.com
- www.crazydomains.com.au
- www.commbank.com.au
- www.xing.com
- podio.com
- www.sofort.com
- www.auajobs.gov
- www.koecp.com.tw
- about.gitlab.com
- vylan.com

Sites you probably haven’t visited:
scraper + tracker code:
https://github.com/diracdeltas/sniffly
Your mileage may vary

- Results depend on latest HSTS preload list.
- HTTPS Everywhere & other extensions cause false positives.
- Doesn’t work as-is in Tor Browser thanks to 100 ms timing buckets.
Your mileage may vary

- Only leaks origin, not full path . . . or does it?

Actually, looks feasible to adapt this attack to leak the 301 redirect cache instead of the HSTS cache. :)
TO BE CONTINUED...
The End

Call me maybe:
yan@mit.edu / @bcrypt

Many <3’s to White Ops for sponsoring my trip to ToorCon!

Special thanks to Scott Helme, Jan Schaumann, Chris Palmer, and Chris Rohlf for feedback and demo testing.