Insights from a 1-million-site Measurement of Online Tracking

Steven Englehardt @s_englehardt
Dillon Reisman @dillonthehuman
Arvind Narayanan @random_walker

This research was funded by NSF award CNS 1526353, a grant from the Data Transparency Lab, and by a cloud credits for research grant from Amazon Web Services.
Visiting 2 websites results in 84 third parties contacted
Open Web Privacy Measurement (OpenWPM)

https://github.com/citp/OpenWPM
The Princeton Web Census

Monthly
1 Million Site Crawl

Collecting:
- Javascript Calls
- All javascript files
- HTTP Requests and Responses
- Storage (cookies, Flash, etc)
Results of the Princeton Web Census

New metric to rank third parties

Nearly all top 3rd parties cookie sync

News site have the most trackers

Tracking protection misses less popular 3rd parties and techniques

https://webtransparency.cs.princeton.edu/webcensus/
Insights from the Princeton Web Census
Consolidation of top trackers
Only 6 organizations are present on >10% of sites
Takeaways of consolidation

(1) Enforcement efforts can target large players, proactively set tracking norms.

(2) Large trackers can quickly deploy technique to a massive number of sites.

(3) Acquisitions can quickly shift tracking capability
Trackers Impede HTTPS Adoption

Firefox 47

Chrome 47
Trackers Impede HTTPS Adoption

Mixed content downgrades security indicator!
Trackers Impede HTTPS Adoption

Of sites with mixed content:

half is caused solely by third parties (10% by trackers)
Trackers Impede HTTPS Adoption

Half of all third-parties are HTTP-only

Of sites with mixed content:
half is caused solely by third parties (10% by trackers)

Half of all third-parties are HTTP-only
Takeaway: Tracking may have second-order privacy impacts

(1) Slow the adoption of encryption
(2) Identifier leakage in requests to
(3) Can aid network surveillance efforts
New Browser Features Used for Fingerprinting

Canvas

WebRTC

Audio

Battery

https://webtransparencies.cs.princeton.edu/webcensus/
Browsers remove BatteryStatus API citing privacy

---

| Status:    | VERIFIED FIXED                  |
| Whiteboard:|                              |
| Keywords:  | addon-compat, dev-doc-needed, privacy, site-compat |
| Product:   | Core (show info)                |
| Component: | DOM: Device Interfaces (show other bugs) (show info) |
| Version:   | unspecified                     |
| Platform:  | Unspecified Unspecified         |
| Importance:| -- normal (vote)                |
| Target Milestone: | mozilla52                     |
| Assigned To:| Chris Peterson [cpeterson]     |

**Reported:** 2016-10-27 23:28 PDT by Chris Peterson [cpeterson]

**Modified:** 2016-11-02 09:53 PDT (History)

**CC List:** 7 users (show)

**Flags:** ryanvm: in-testsuite-

**Crash Signature:** (edit)

**QA Whiteboard:**

**Iteration:** ---

**Points:** ---

**Has Regression Range:** ---

---

Persona is no longer an option for authentication on BMO. For more details see Persona Deprecated.
Browsers remove BatteryStatus API citing privacy
Takeaway: Expect any new API to be analyzed for its fingerprintability

1. Early detection of abuse can stem adoption

2. Browsers view fingerprinting as abuse
   a. Mitigate fingerprinting during standardization
   b. Remove APIs due to fingerprinting use
Our data is available!

The data is available as bzipped PostgreSQL dumps. The schema file used in all of the datasets is available here.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Million Site Stateless</td>
<td>Parallel Stateless Crawl</td>
</tr>
<tr>
<td>100k Site Stateful</td>
<td>Parallel Stateful Crawl -- 10,000 site seed profile</td>
</tr>
<tr>
<td>10k Site ID Detection (1)</td>
<td>Sequential Stateful Crawl -- Flash enabled -- Synced with ID Detection (2)</td>
</tr>
<tr>
<td>10k Site ID Detection (2)</td>
<td>Sequential Stateful Crawl -- Flash enabled -- Synced with ID Detection (1)</td>
</tr>
<tr>
<td>55k Site Stateless with cookie blocking</td>
<td>Parallel Stateless Crawl -- Firefox set to block all third-party cookies</td>
</tr>
<tr>
<td>55k Site Stateless with Ghostery</td>
<td>Parallel Stateless Crawl -- Ghostery extension installed and set to block all possible trackers</td>
</tr>
<tr>
<td>55k Site Stateless with HTTPS Everywhere</td>
<td>Parallel Stateless Crawl -- HTTPS Everywhere installed</td>
</tr>
</tbody>
</table>

https://webtransparency.cs.princeton.edu/webcensus/index.html#data
Getting third-party responses from our data

def get_host_plus_ps(url):
    ":Strip the URL down to just a hostname+publicsuffix.
    
    If the provided url contains an IP address, the IP address is returned.
    
    hostname = urllib.parse(url).hostname
    try:
        ip_address(hostname)
    return hostname
except ValueError:
    return psi.get_public_suffix(hostname)

def is_js(url, content_type):
    if get_top_level_domain(content_type) == 'script':
        return True
    if urlparse(url).path.split('/')[-1].lower() == 'js':
        return True
    if extension.lower() in image_types:
        return True
    return False

def is_img(url, content_type):
    if get_top_level_domain(content_type) == 'image':
        return True
    if extension.lower() in image_types:
        return True
    return False

def get_trackers(url_list, first_party, blacklist_parser=None, blacklist="easylist.txt"):  
    
    """Identify domains that are identified as trackers from list of URLs.
    
    Returns set of domains/IPs filtered by the given blacklist_parser.
    TODO: Better to return set of domains/IPs, or list of filtered urls?
    ""
    if not blacklist_parser:
        blacklist_parser = BlockListParser(blacklist)
    filtered_domains = set()
    for url in url_list:
        if is_tracker(url, first_party, blacklist_parser):
            filtered_domains.add(get_host_plus_ps(url))
    return filtered_domains

def is_tracker(url, first_party, blacklist_parser):
    pass

def is_url(url, content_type):
    if get_top_level_domain(content_type) == 'url':
        return True
    if extension.lower() in image_types:
        return True
    return False

def tp_query = "SELECT r.url, h.value FROM http_responses_view AS r \
    LEFT JOIN http_response_headers_view as h ON h.response_id = r.id" \
    WHERE r.top_url LIKE %s AND " \
    'url' not LIKE %s and h.name = 'Content-Type'"
cur = connection.cursor()
cur.execute(tp_query, (top_url, top_ps))
Getting third-party responses from our data
Getting third-party responses with Census.py

census.get_third_party_responses_by_domain(
    database_connection,
    "http://nytimes.com"
)
Getting third-party responses with Census.py

- get_third_party_responses_by_domain
- get_third_party_responses_by_domain
- get_cookie_syncs_on_domain
- is_tracker
- get_trackers
Getting third-party responses with Census.py

- get_third_party_responses_by_domain
- get_third_party_responses_by_domain
- get_cookie_syncs_on_domain
- is_tracker
- get_trackers

Contact us for access to “alpha” analysis server and library!
Thanks for listening!

Full Paper: senglehardt.com/papers/ccs16_online_tracking.pdf
Data and Analysis: webtransparency.cs.princeton.edu/webcensus/
Collaborate: webtap.princeton.edu/research/

Contact Me
Email: ste@cs.princeton.edu
Twitter: @s_englehardt
Web: senglehardt.com