PRINCETON UNIVERSITY

CENTER FOR INFORMATION TECHNOLOGY POLICY
PRINCETON UNIVERSITY

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

Steven Englehardt
@s_englehardt

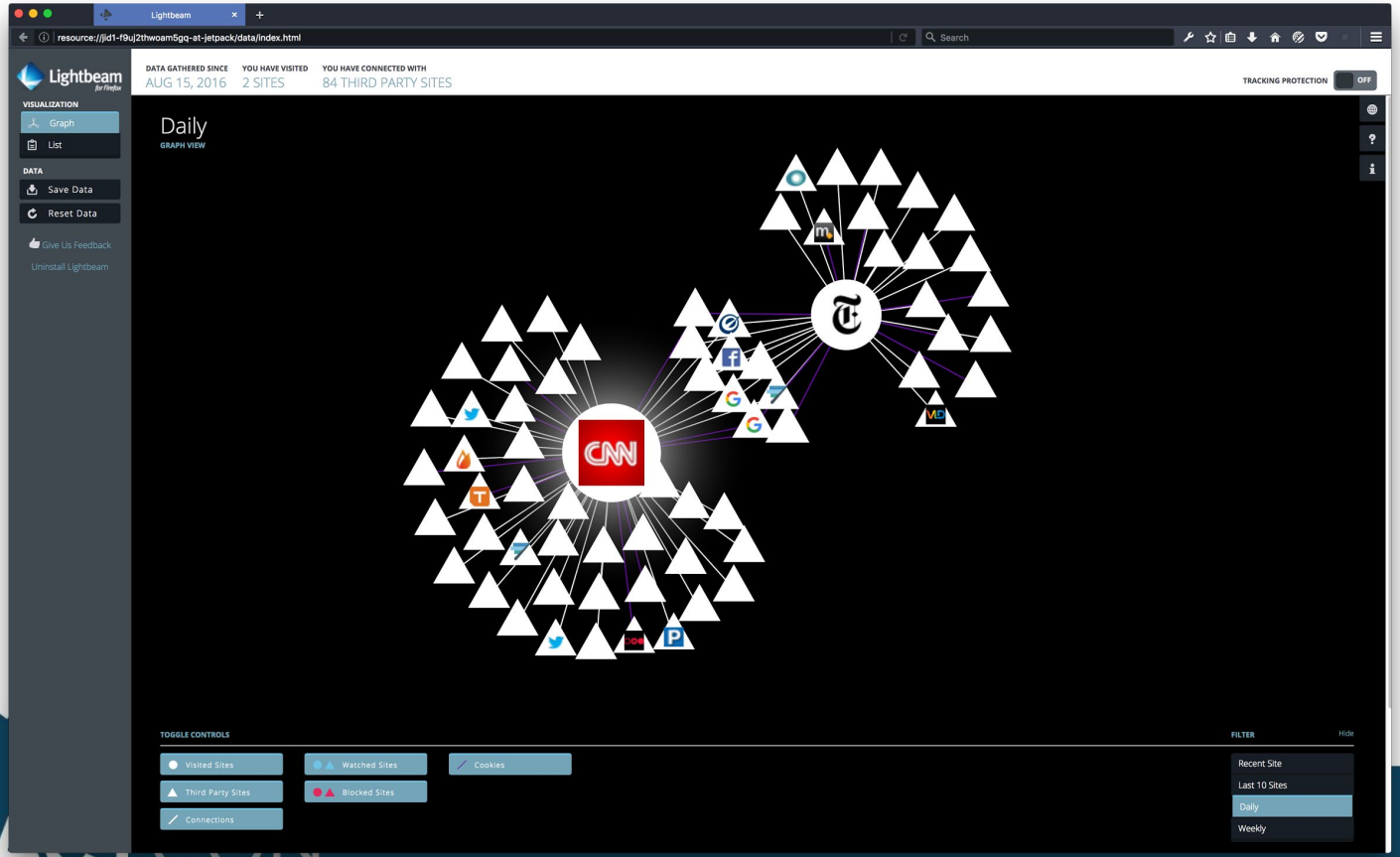
Dillon Reisman
@dillonthehuman

Arvind Narayanan
@random_walker

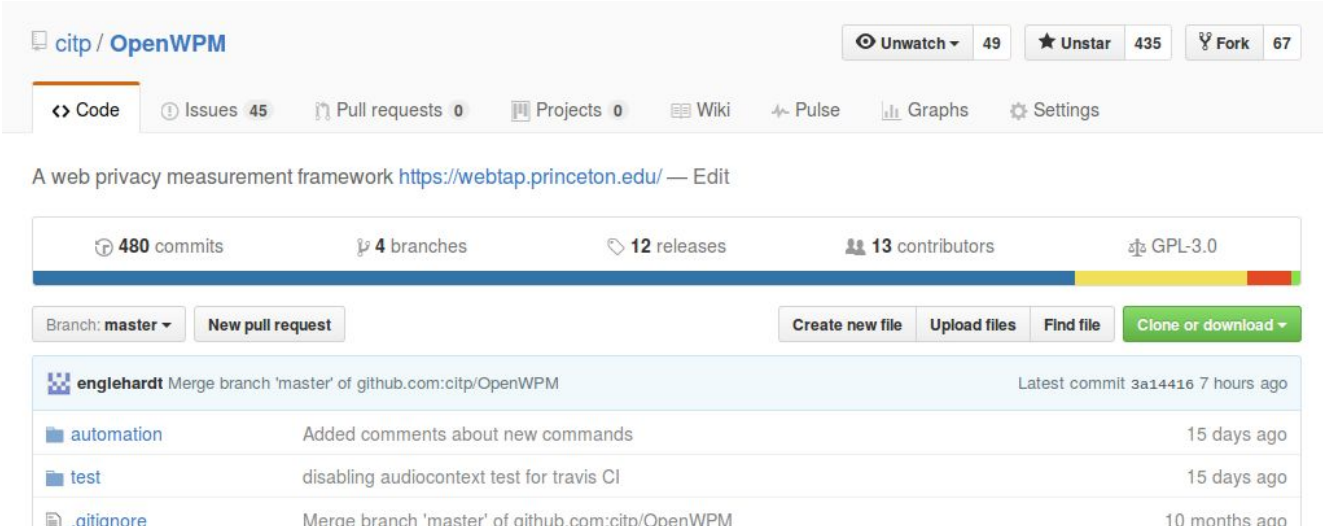
This research was funded by NSF award CNS 1526353, a grant from the Data Transparency Lab, and by a cloud credits for research grant from Amazon Web Services.

PRIVACYCON

Visiting 2 websites results in 84 third parties contacted



Open Web Privacy Measurement (OpenWPM)



https://github.com/citp/OpenWPM

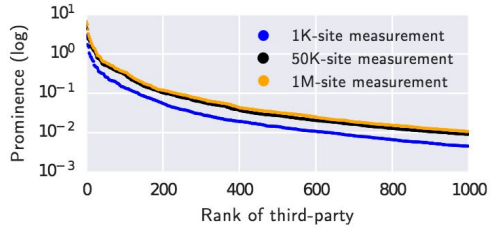
The Princeton Web Census

Monthly
1 Million Site Crawl

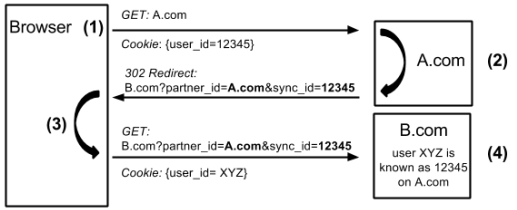
Collecting:

- Javascript Calls
- All javascript files
- HTTP Requests and Responses
- Storage (cookies, Flash, etc)

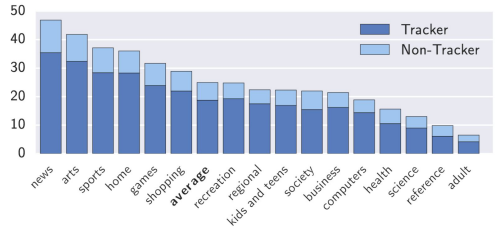
Results of the Princeton Web Census



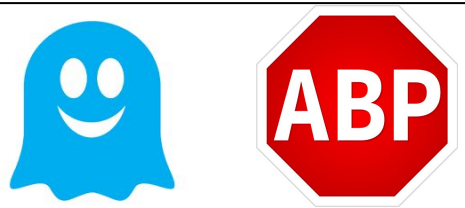
New metric to rank third parties



Nearly all top 3rd parties cookie sync



News site have the most trackers



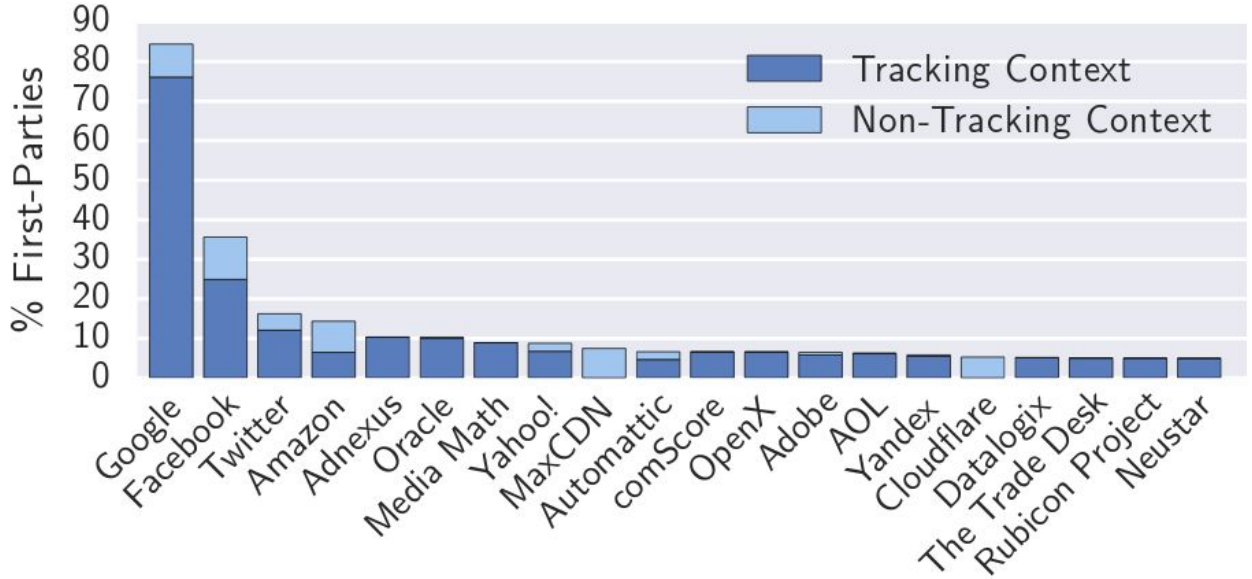
Tracking protection misses less popular 3rd parties and techniques

https://webtransparency.cs.princeton.edu/webcensus/

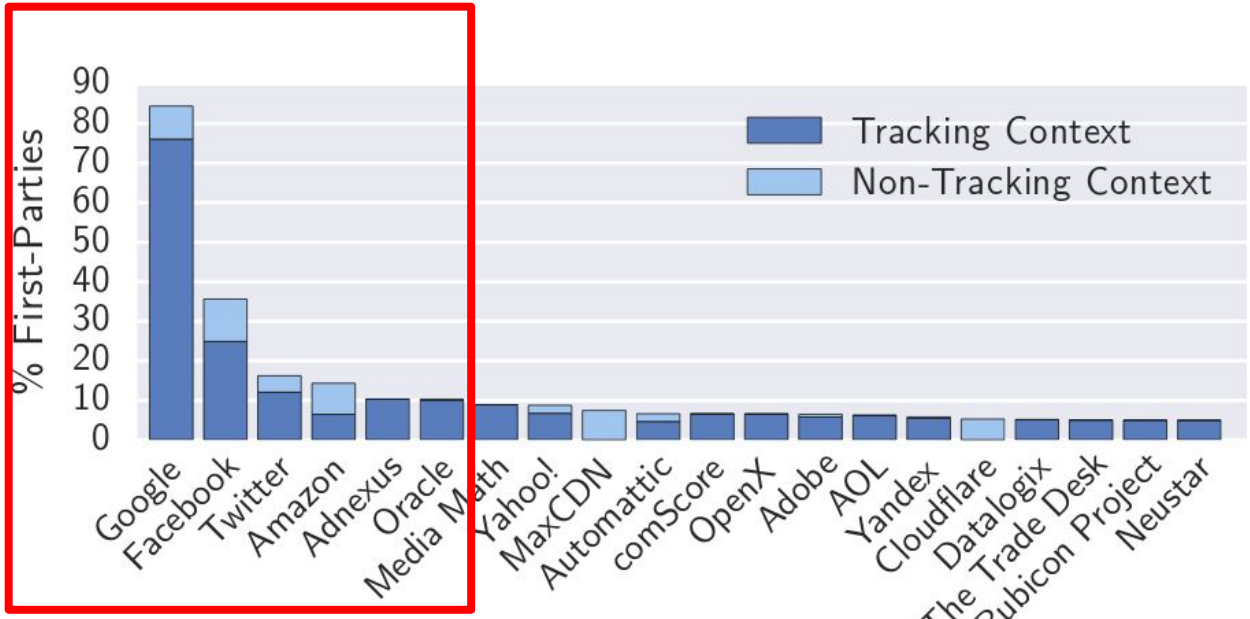
Insights from the Princeton Web Census

PRIVACYCON

Consolidation of top trackers



Only 6 organizations are present on >10% of sites

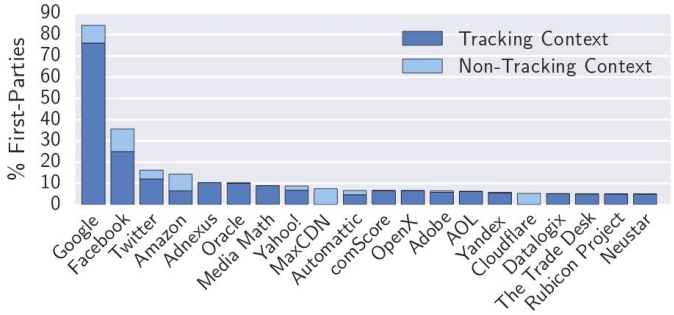


Takeaways of consolidation

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

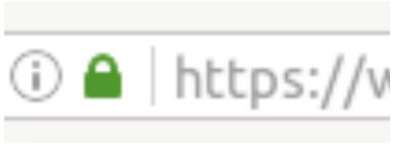
(2) Large trackers can quickly deploy technique to a massive number of sites.

(3) Acquisitions can quickly shift tracking capability



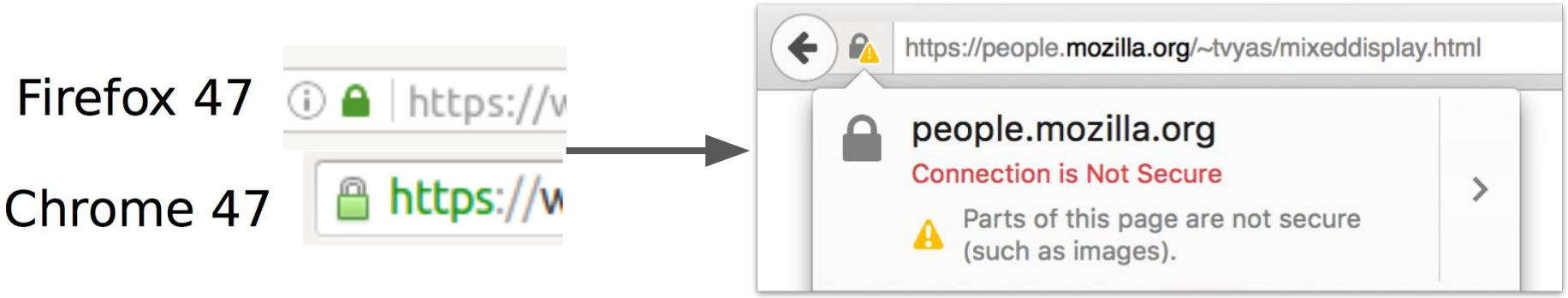
Trackers Impede HTTPS Adoption

Firefox 47



Chrome 47

Trackers Impede HTTPS Adoption

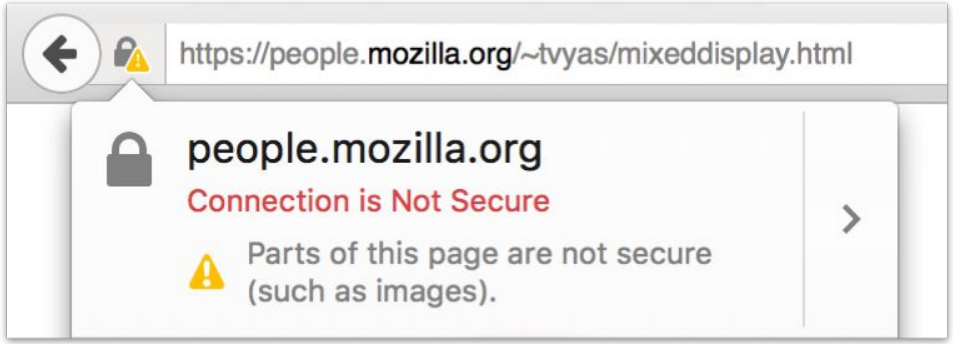


Mixed content downgrades security indicator!

Trackers Impede HTTPS Adoption

Firefox 47

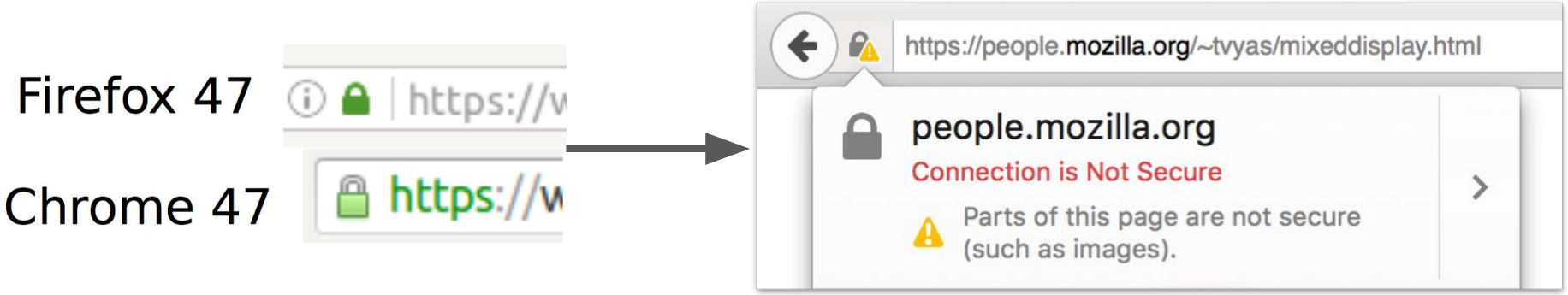
Chrome 47



Of sites with mixed content:

half is caused solely by third parties (10% by trackers)

Trackers Impede HTTPS Adoption



Of sites with mixed content:

half is caused solely by third parties (10% by trackers)

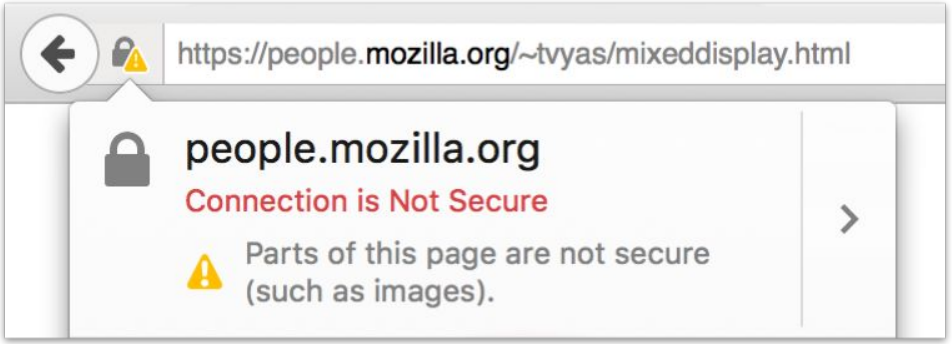
Half of all third-parties are HTTP-only

Takeaway: Tracking may have second-order privacy impacts

(1) Slow the adoption of encryption

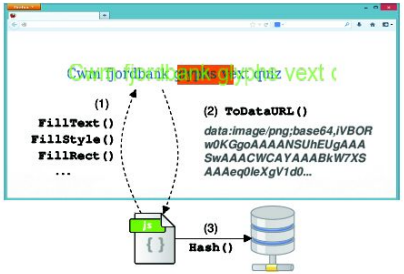
(2) Identifier leakage in requests to

(3) Can aid network surveillance efforts

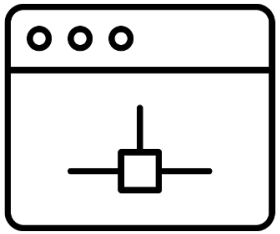


New Browser Features Used for Fingerprinting

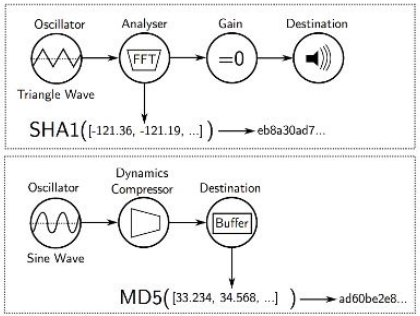
Canvas



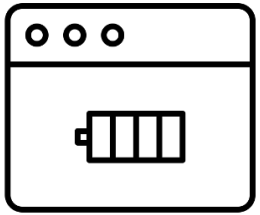
WebRTC



Audio

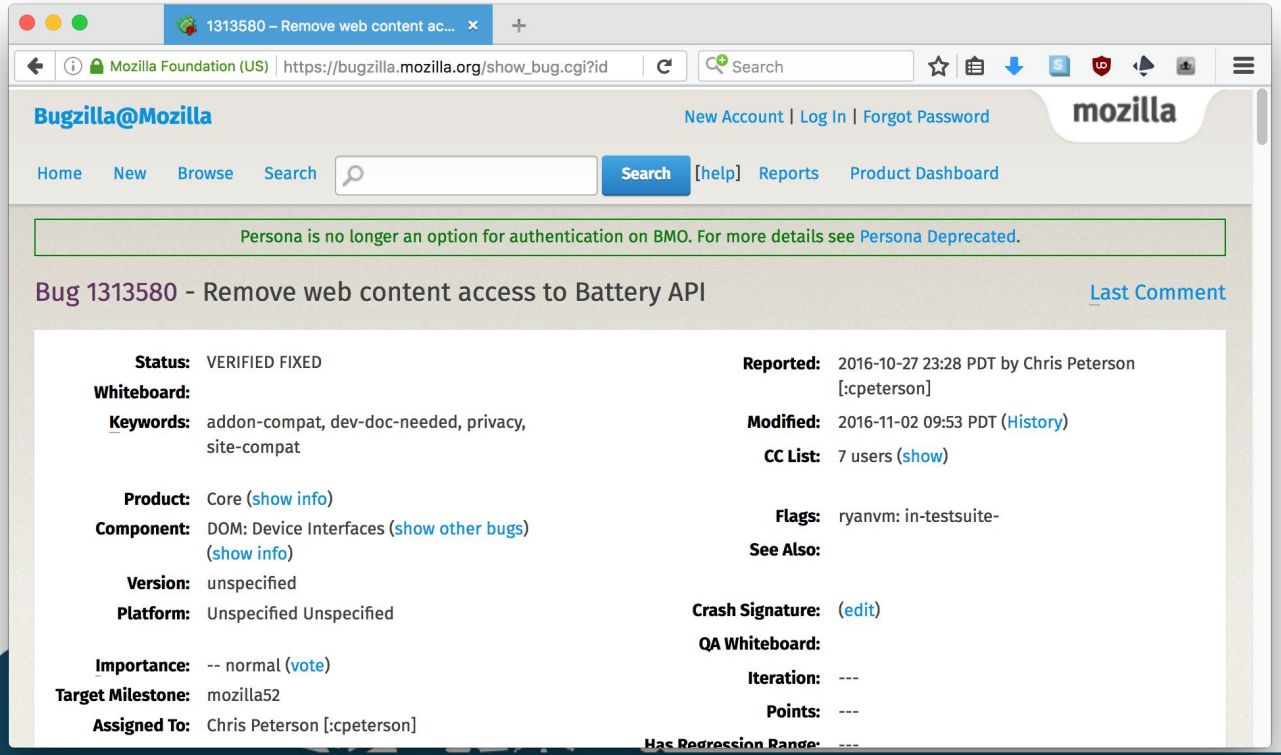


Battery



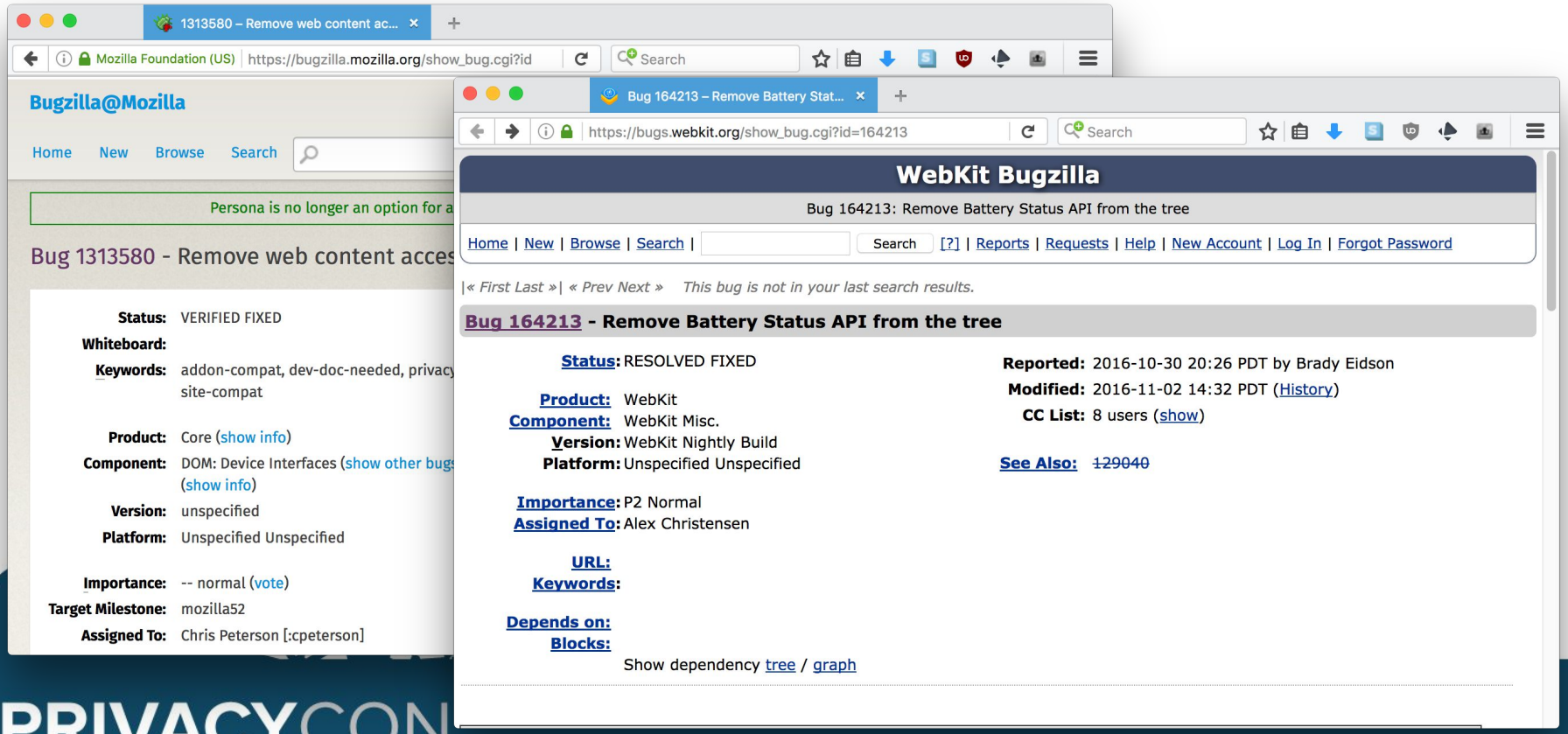
https://webtransparency.cs.princeton.edu/webcensus/

Browsers remove BatteryStatus API citing privacy



PRIVACYCON

Browsers remove BatteryStatus API citing privacy



PRIVACYCON

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.

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

https://webtransparency.cs.princeton.edu/webcensus/index.html#data

Getting third-party responses from our data

```
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.itersize = 100000
try:
    top_ps = utils.get_host_plus_ps(top_url)
except AttributeError:
    print("Error while finding public suffix of %s" % top_url)
    return None

cur.execute(tp_query, (top_url, top_ps))

el_parser = BlockListParser('easylist.txt')
ep_parser = BlockListParser('easyprivacy.txt')
response_data = defaultdict(dict)
for url, content_type in cur:
    if utils.should_ignore(url):
        continue

    url_data = dict()

    url_ps = utils.get_host_plus_ps(url)
    if url_ps == top_ps:
        continue
    url_data['url_ps'] = url_ps

    is_js = utils.is_js(url, content_type)
    is_img = utils.is_img(url, content_type)
    is_el_tracker = utils.is_tracker(url,
                                     is_js=is_js,
                                     is_img=is_img,
                                     first_party=top_url,
                                     blocklist_parser=el_parser)
    is_ep_tracker = utils.is_tracker(url,
                                     is_js=is_js,
                                     is_img=is_img,
                                     first_party=top_url,
                                     blocklist_parser=ep_parser)
    is_tracker = is_el_tracker or is_ep_tracker

    url_data['is_js'] = is_js
    url_data['is_img'] = is_img
    url_data['is_tracker'] = is_tracker
    response_data[url] = url_data
```

```
def is_js(url, content_type):
    if get_top_level_type(content_type) == 'script':
        return True
    if urlparse(url).path.split('.')[-1].lower() == 'js':
        return True
    return False
```

```
def is_img(url, content_type):
    if get_top_level_type(content_type) == 'image':
        return True
    extension = urlparse(url).path.split('.')[-1]
    if extension.lower() in IMAGE_TYPES:
        return True
    return False
```

```
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 = urlparse(url).hostname
    try:
        ip_address(hostname)
        return hostname
    except ValueError:
        return psl.get_public_suffix(hostname)
```

```
def get_trackers(url_list, first_party, blocklist_parser=None, blocklist="easylist.txt"):
    """Identify domains that are identified as trackers from list of URLs.

    Returns set of domains/IPs filtered by the given blocklist_parser.
    TODO: Better to return set of domains/IPs, or list of filtered urls?
    """
    if not blocklist_parser:
        blocklist_parser = BlockListParser(blocklist)

    filtered_domains = set()
    for url in url_list:
        if is_tracker(url, first_party, blocklist_parser):
            filtered_domains.add(get_host_plus_ps(url))

    return filtered_domains
```

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

Image Assets from the Noun Project:
Browser Network and Browser Battery by Aybige