Best-of-Breed Content Blocking in Brave:
Three Projects to Improve the Depth, Breath, and Usefulness of Blocking at Scale

Pete Snyder
Senior Privacy Researcher
pes@brave.com
@pes10k
Hi, I’m Pete👋

- **Grew up in Chicago**
  ...actual Chicago

- **Law school -> freelance web stuff**
  Started: Anchorage, AK
  Ended: Judge Judy Show invitation

- **PhD in Computer Science**
  University of Illinois at Chicago
Me at Brave

● **Researcher at Brave**
  ...privacy, blocking, reliability

● **Co-Chair of PING**
  Privacy committee on W3C

● **Research <-> Engineering**
  Web compat, filter lists, fingerprinting, etc.
Brave in a Slide

- Privacy focused
- Alternative web funding model
  Fix incentive problems
- Research + Engineering
- Browsers and infrastructure now,
  more to come...
Overview

- **PageGraph**  
  Novel, open source, DOM attribution system

- **Behavioral Blocking**  
  Protecting Privacy Beyond URLs

- **Regional Filter List Generation**  
  Privacy protection for users in underserved regions

- **Automated Web Compatibility Testing**  
  Making sure we’re only breaking bad stuff…
Overview

- **PageGraph**
  Novel, open source, DOM attribution system

- **Behavioral Blocking**
  Protecting Privacy Beyond URLs

- **Regional Filter List Generation**
  Privacy protection for users in underserved regions

- **Automated Web Compatibility Testing**
  Making sure we’re only breaking bad stuff…
Anton Lazarev - alazarev@brave.com
Northeastern University
PageGraph: Motivation

- Lots of research requires understanding what script does what…
  Forensics, privacy, performance prediction, debugging…

- Seems simple…
  Doesn’t devtools do this already? Projects in research?

- Online and offline analysis
  Ordered, correct causal analysis
PageGraph: Challenges

- **Scripts**
  - Script injects another script?
  - Inline scripts?
  - Script in HTML attributes? JS URLs?
  - Eval, new Function(), callback functions, micro tasks…

- **Network**
  - Attribution for injected images / videos
  - Fetch? Ajax? Remote frames?

- **Web API use? Storage? DOM modifications? Etc…**
PageGraph: High Fidelity Cause Attribution

- **Modification Cause Attribution**
  Record cause of all DOM modifications, network requests, privacy-relevant WebAPI calls

- **Graph Representation**
  events -> edges, page elements -> nodes

- **Online and Offline Analysis**
  In memory graph, GraphML Export

- [https://github.com/brave/brave-browser/wiki/PageGraph](https://github.com/brave/brave-browser/wiki/PageGraph)
PageGraph: Toy Example

```javascript
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
   ...(rest removed)
</script>
```
PageGraph: Toy Example

```javascript
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0] …(rest removed)
</script>
```
PageGraph: Toy Example

```javascript
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
  ...(rest removed)
</script>
```
PageGraph: Toy Example

```javascript
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
... (rest removed)
</script>
```
PageGraph: Toy Example

```html
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
  ...(rest removed)
</script>
```
PageGraph: Toy Example

```html
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
...(rest removed)
</script>
```
PageGraph: Toy Example

```html
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
...(rest removed)
</script>
```
PageGraph: Toy Example

```html
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
...(rest removed)
</script>
```
PageGraph: Toy Example

```javascript
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0] ...
...(rest removed)
</script>
```
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
...(rest removed)
</script>
PageGraph: Toy Example

```javascript
<script>
const elm = document.createElement("img");
elm.src = "/pixel.gif";
document.body.appendChild(elm);
document.getElementsByTagName("script")[0]
  ...(rest removed)
</script>
```
Fingerprint2.js example...
All of CNN.com (for 30 sec...)
Limitations

● **Still stuff to instrument…**
  e.g. Module scripts

● **Bugs (but, few!)**
  < 5% of pages will have an attribution error

● **Dense graphs**
  Pages do a lot of things…
PageGraph: Summary

- Comprehensive causal attribution
  Attribute every request, DOM modification, compilation and privacy event to responsible script

- In modern chromium based browser
  Up to date Brave, up to date Chromium

- Open source and For Use Now
  Build it today!
Overview

- **PageGraph**
  Novel, open source, DOM attribution system

- **Behavioral Blocking**
  Protecting Privacy Beyond URLs

- **Regional Filter List Generation**
  Privacy protection for users in underserved regions

- **Automated Web Compatibility Testing**
  Making sure we’re only breaking bad stuff…
Quan Chen - qchen10@ncsu.edu
North Carolina State University
Blocking Arms Race

Tracking Domains → /etc/hosts, Safari ITP DNS filtering, etc… → Move “bad” code to “good” domains
Blocking Arms Race

Tracking Domains

/etc/hosts, Safari ITP
DNS filtering, etc…

Move “bad” code to
“good” domains

Mixed Domains

Filter Lists, ITP2.0

Mix “bad” code with
“good” code
Blocking Arms Race

Tracking Domains
- /etc/hosts, Safari ITP
  - DNS filtering, etc…
  - Move “bad” code to “good” domains

Mixed Domains
- Filter Lists, ITP2.0
  - Mix “bad” code with “good” code

Mixed Code
Blocking Arms Race

Tracking Domains /etc/hosts, Safari ITP DNS filtering, etc… Move “bad” code to “good” domains

Mixed Domains Filter Lists, ITP2.0 Mix “bad” code with “good” code

Mixed Code
Behavioral Blocking Roadmap

1. Motivation
2. Methodology
3. Results
4. Whats next
Behavioral Blocking Roadmap

1. Motivation
2. Methodology
3. Results
4.Whats next
Motivating Example

<html>
<head>
  <script src="(google-analytics.js)">
  <script src="(jQuery)">
  <script src="(Site stuff)">
</head>
<body>...
</body>
</html>
Motivating Example

<html>
<head>
<script src="(google-analytics.js)"></script>
<script src="(jQuery)"></script>
<script src="(Site stuff)"></script>
</head>
<body>...</body>
</html>
Motivating Example

<html>
<head>
<script src="(google-analytics.js + jQuery + site stuff).min.js"></script>
</head>
<body>...</body>
</html>
Core Problem: URLs

- Moved code
- Changing URLs
- Inlining code
- Bundling code
- etc
Project Goals

- Deliver the same privacy protections, independent of code delivery...
- Independent of URL...
- w/r/t Network requests, storage access, fingerprinting endpoints
Straw Proposals

Matching hashes?

Trivial to circumvent
Straw Proposals

- Matching hashes?
  - Trivial to circumvent

- ASTs?
  - Code bundlers / mungers
Straw Proposals

- Matching hashes?
  - Trivial to circumvent

- ASTs?
  - Code bundlers / mungers

- Identifying tokens
  - Minification / obfuscation / etc
Behavioral Blocking Roadmap

1. Motivation

2. Methodology

3. Results

4. Deployment
Methodology

1. Behavioral signatures of scripts
   Page graph, event loop turns

2. Build set of known privacy-harming behaviors
   EasyList + EasyPrivacy as ground truth

3. Check to see if new scripts do known bad things
   e.g. apply extracted signatures
Signature Granularity

- ** Entire Script / Library?**
  Too non-deterministic, too much variation between sites
  e.g. false negatives

- ** Individual API Calls?**
  Not enough information
  e.g. false positives

- ** Event Loop Turns**
  Good balance
  Allows some variation between library use
Fingerprint2.js (e.g. bad)
Fingerprint2.js (e.g. bad)
Obvious Fingerprint2.js is… obvious
site.js (e.g. benign)
combined.min.js

canvas

<canvas src=/logo.png>

Image resource

screen.height

Image resource

<body>

class=js-yes

doc.cookie
Putting it together…

- Catch script activities between event loop yields
- Graph represents as sequential edges and nodes
- Extract EL+EP subgraphs using the above algo
- Look for these recurring patterns in non-labeled JS units
- Enforce protections as if it came from original labeled script
Behavioral Blocking Roadmap

1. Motivation

2. Methodology

3. Results

4. Whats next
## Raw data (partial)

<table>
<thead>
<tr>
<th></th>
<th>Alexa 100k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crawl set</td>
<td>1,996,093</td>
</tr>
<tr>
<td>Generated signatures</td>
<td>400,287</td>
</tr>
<tr>
<td>Privacy affecting, &quot;bad&quot; behaviors</td>
<td>11,443</td>
</tr>
<tr>
<td>Num sites with &gt;= 1 missed behavior</td>
<td>11,443</td>
</tr>
</tbody>
</table>
Taxonomy: Moving Scripts

- Moving script file from blocked code to new URL
- https://example.org/tracker.js -> https://helpful.org/site.js
- Long tail, hard to crowdsourced, etc.
- 4,730 sites
Google Analytics

- https://www.google-analytics.com/analytics.js
- Blocked by `||google-analytics.com/analytics.js`
Google Analytics

- https://www.google-analytics.com

- Blocked by `||google-analytics.com` precaution

Taxonomy: Inlining

- Moving script from a URL to txt
- `<script src=”X”>` to `<script>(code)</script>`
- Difficult for most tools to address at all
- Impossible for current tools to address well
- 223 sites
Dynatrace Example

- https://www.dynatrace.com/*
  (e.g. https://js-cdn.dynatrace.com/jstag/157944990f8/fsf84414/…)

- Blocked by `||dynatrace.com^$third-party`

- But look at https://www.dynatrace.com…
Taxonomy: Bundling

- Including harmful code with benign / user-serving code
- jQuery + fingerprint2.js + site.js -> combined.min.js
- No win choice for blockers…
- 122 sites
Microsoft Insights

- E.g. https://az416426.vo.msecnd.net/scripts/a/ai.0.js

- Blocked by `||msecnd.net/scripts/a/ai.0.js`

Taxonomy: Common Library

- Third party libraries used by larger libraries
- Original library URL may never (rarely?) be fetched directly
- Not *anything* to block off
- 6,141 sites
Adobe Visitor API

- Adobe has many tracking libraries (e.g. Adobe Target) that all use the Adobe Visitor API
- Visitor API is the tracking part
- Much is blocked with `||adobedtm.com^$third-party`
- Other libs suck in Visitor API too
<table>
<thead>
<tr>
<th>Technique</th>
<th># Instances</th>
<th>Unique Scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving</td>
<td>4,730</td>
<td>586</td>
</tr>
<tr>
<td>Inlining</td>
<td>450</td>
<td>450</td>
</tr>
<tr>
<td>Bundling</td>
<td>122</td>
<td>92</td>
</tr>
<tr>
<td>Common Code</td>
<td>6,141</td>
<td>2,342</td>
</tr>
</tbody>
</table>
Behavioral Blocking Roadmap

1. Motivation

2. Methodology

3. Results

4. What's next
What To Do Next

● Defenses
  Moved scripts -> new filter rules
  Others -> maybe runtime fingerprint enforcement?

● Is the problem getting worse?
  Longitudinal measurements (and WayBack Machine)
  Measurements going forward

● Other places too
  JS urls, HTML attributes, service workers, etc.
Take Aways…

- Filter lists are porous
- Focusing on URLs is the problem
- Behavior >> delivery
- Brave has plans to protect users in a novel, more comprehensive way
Overview

- **PageGraph**
  Novel, open source, DOM attribution system

- **Behavioral Blocking**
  Protecting Privacy Beyond URLs

- **Regional Filter List Generation**
  Privacy protection for users in underserved regions

- **Automated Web Compatibility Testing**
  Making sure we’re only breaking bad stuff…
Alexander Sjösten - sjosten@chalmers.se
Chalmers University of Technology
Filter Lists, Unsung Heros…

- **Crowdsourced Lists**
  Network Rules: Requests
  Cosmetic Rules: Display
  Exceptions: Reverse the Above

- **EasyList: Blocks Ads**
  73,079 rules

- **EasyPrivacy: Block Trackers**
  17,024 rules
Crowdsourcing Is Tricky
What About Everyone Else

- Regions with Fewer Speakers?
- Regions with Less Affluent Speakers?
- Less Internet Users?
- Higher Data Plans?
- etc…
Project In a Nutshell

- **Image and Frame Classifier**
  Heavy, offline, contextual

- **Block High in Request Chain**
  e.g. PageGraph to determine request causes

- **Don’t Break Pages**
  e.g. PageGraph to understand impact of blocking

- **Generate AdBlock Compatible Rules**
  e.g. shippable and shareable
Project In a Nutshell

- **Image and Frame Classifier**
  Heavy, offline, contextual

- **Block High in Request Chain**
  e.g. PageGraph to determine request causes

- **Don’t Break Pages**
  e.g. PageGraph to understand impact of blocking

- **Generate AdBlock Compatible Rules**
  e.g. shippable and shareable
Nike Air Zoom Fencing Shoes Black and Pink
## Existing Work Doesn’t Generalize

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trained Data</strong></td>
<td>95.9%</td>
<td>95.5%</td>
<td>96.4%</td>
</tr>
<tr>
<td><strong>New Data</strong></td>
<td>77.0%</td>
<td>48.8%</td>
<td>87.4%</td>
</tr>
</tbody>
</table>
Contextual Classifier

- **Existing Image Classifier**
  Tigas, Panagiotis, Samuel T. King, and Benjamin Livshits. "Percival: Making In-Browser Perceptual Ad Blocking Practical With Deep Learning."

- **PageGraph Contextual Information**
  First or third party URL?
  Injected by script?
  Parent node degree?
Context Helps a Lot…

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trained Data</td>
<td>95.9%</td>
<td>95.5%</td>
<td>96.4%</td>
</tr>
<tr>
<td>New Data</td>
<td>77.0%</td>
<td>48.8%</td>
<td>87.4%</td>
</tr>
<tr>
<td>Contextual</td>
<td>97.6%</td>
<td>92.0%</td>
<td>75.0%</td>
</tr>
</tbody>
</table>
Project In a Nutshell

- **Image and Frame Classifier**
  Heavy, offline, contextual

- **Block High in Request Chain**
  e.g. PageGraph to determine request causes

- **Don’t Break Pages**
  e.g. PageGraph to understand impact of blocking

- **Generate AdBlock Compatible Rules**
  e.g. shippable and shareable
Ads Aren’t (Usually) Static URLs

- Programmatically Inserted Scripts
- Real Time Bidding
- Header Bidding
- Targeting Final URLs -> Stale Rules
FOTOT që flasin/ “Dreka e gabuar”: Tallja e PD-së me luftën ndaj oligarkëve dhe ato zgjedh

19 Nëntor, 2019 10:27 | 3 Komenta

“Naziqëm në Bullgari dhe racizëm në Mailin e Zi, ndërsa në Kosovë na qerqosën me birra”
19 Nëntor, 2019 22:30 | 0 Komenta

FOTOT: 800 km për të humb drejtësynë me burrin 24 vite më të madh dhe... të dashurën e tij
19 Nëntor, 2019 22:20 | 0 Komenta

43-vjeçari në rrezik për jetën, mjekët zbulojnë se shkaktar ishte jorgani
19 Nëntor, 2019 22:15 | 0 Komenta

Ngërqi në Kushitetuesë/ Ironia e Patozë: Kritë të ‘fabrikuar! Kur Meta thotë të marrit amrët, Bashë është tash fshini fermerët
19 Nëntor, 2019 22:10 | 0 Komenta

ME SUPER SMART PARASHIKIMIN E BËN TI!

19 Nëntor, 2019 10:25 | 3 Komenta
Pse bën Ilir Meta si i çakallosur dhe pse Basha nuk mund të ndihmojë!

19 Nëntor, 2019 21:11 | 1 Komante
We next discuss how we generate generalized attribution. We observed a non-trivial number of corner cases where AdGraph records. For example, PageGraph tracks image requests (e.g. JavaScript stack is reset through events like timer callbacks), but AdGraph captures only these and a large number of similar corner cases.

### 3.3.2 Diagnosing which Scripts Cause Requests

In addition to blocking higher in request chains reducing the total number of our image classification mistakes, we use the browser instrumentation to build the entire request chain that caused the script request to be included in the page (e.g. the script that fetched the URL of the ad library itself will rarely change. Approaches that do not consider request chains yield a more consistent set of URLs. While the request chain as a whole is the key reason for the URL differences, the script that inserted the image), and then again use the browser instrumentation to determine how far up each request chain we can block without breaking the page.

#### 3.3.3 Block in Request Chains

To generate optimized results, we build these request chains for both images (and frames) our system ads into pages from scripts that perform more complex, and potentially, the scripts that injected those scripts, etc.). We refer to the list of elements that participate in an advertisement being included as its request chain. The most relevant related work identifies ads as an ad, and for resources identified by netloc, the latter allows us to block in a request chain without breaking the page. Our approach is "conservative" (i.e. prefers false negatives over false positives), and the goal of determining the earliest "upstream" request we can block in a request chain without breaking the page. We use this technique to start blocking, using the approach described in the following subsections.

#### 3.4.2 Building Request Chains

For each script, we determine the optimal place to block. The general approach is to generalize the benefits of our image classification mistakes. We use these request chains to determine the optimal place to block. The general approach is to generalize the benefits of our image classification mistakes. We use these request chains to determine the optimal place to block.

A simplification of the request chain for a script that inserted another script that matches condition #1, the latter allows us to block in a request chain without breaking the page. Our approach is "conservative" (i.e. prefers false negatives over false positives), and the goal of determining the earliest "upstream" request we can block in a request chain without breaking the page. We use this technique to start blocking, using the approach described in the following subsections.

#### 3.5 Safe Blocking in Request Chains

To generate optimized results, we build these request chains for both images (and frames) our system ads into pages from scripts that perform more complex, and potentially, the scripts that injected those scripts, etc.). We refer to the list of elements that participate in an advertisement being included as its request chain. The most relevant related work identifies ads as an ad, and for resources identified by netloc, the latter allows us to block in a request chain without breaking the page. Our approach is "conservative" (i.e. prefers false negatives over false positives), and the goal of determining the earliest "upstream" request we can block in a request chain without breaking the page. We use this technique to start blocking, using the approach described in the following subsections.

#### 3.4.2 Building Request Chains

For each script, we determine the optimal place to block. The general approach is to generalize the benefits of our image classification mistakes. We use these request chains to determine the optimal place to block. The general approach is to generalize the benefits of our image classification mistakes. We use these request chains to determine the optimal place to block.
We next discuss how we generate generalized filter rules in existing work requests, among many others. This additional attribution to PageGraph is the AdGraph project, which also modifies AdGraph records. For example, PageGraph tracks image requests (e.g., JavaScript stack is reset through events like timer callbacks), and image URLs. Increased Attribution Breadth.

The most relevant related references from AdGraph in several significantly improved ways.

3.3 Diagnosing & Fixing Corner Cases
3.3.1 Diagnosing Corner Cases

We observed a non-trivial number of corner cases where AdGraph would attribute modifications to the wrong script unit, use the browser instrumentation to build the entire request chain that caused the advertisement to be included in the page (e.g., the script that fetched the script that inserted the image), and then again use the browser instrumentation. The general approach is to use the data gathered by the previously described image classification.

3.4 Generalizing Filter Rules

3.4.1 Building a New Filter List

Generalization of filter rules that quickly go stale; rules that target ad library scripts (as the URL of the ad library itself will rarely change). Approaches that the request chain yields a more consistent set of URLs. While the classiﬁers for detecting whether a script is an ad, and for resources identiﬁed by netloc are as an ad, and for resources identiﬁed by netloc.

3.4.2 Building Request Chains.

Blocking a single "upstream" ad library may prevent the browser from needing to consider several "downstream" requests. These heuristics are designed to distinguish scripts that only inject ads into pages from scripts that perform more complex, and that second script inserted an advertising image into the page. In the initial HTML ("script one"), that script programmatically anticipated in an advertisement being included as its "request chain."

We consider it to block. To determine how "high" in each request chain we can block, with the goal of determining the earliest "upstream" request we can block in a request chain without breaking the page. Our approach dictates the set of page events tracked in the graph, beyond what filter rules from existing filter lists (i.e., EasyList, EasyPrivacy and the classiﬁcation.

3.5 Safe Blocking in Request Chains

We use a pair of simple heuristics to determine which JavaScript unit is responsible for each modification: (1) If a script inserts another script that matches condition #1, (2) If a script creates more than two subtrees in the document, (3) Otherwise, append the script element into the request chain, where a script was included in a page, we determine when to start blocking, using the approach described in the following section.
We next discuss how we generate generalized attribution rules in existing work to PageGraph is the AdGraph project, which also modifies AdGraph records. For example, PageGraph tracks image requests frequently changing, one-o

cases where Ad rules that quickly go stale; rules that target ad library scripts (as

target the frequently changing image URLs will result in the URL of the ad library itself will rarely change. Approaches that generalize the bene

to maximize the bene

3.4.1 Motivation.

3.3.2 Di

3.5.1 Determining Page Breakage.

3.5 Safe Blocking in Request Chains

This subsection describes how we determine whether blocking a request chain without breaking the page. Our approach is to start blocking, using the approach described in the following section.

We use these request chains to determine the optimal place to block. For each

Figure 5: Example of a request chain, ending in an inserted ad image.
Project In a Nutshell

- **Image and Frame Classifier**
  Heavy, offline, contextual

- **Block High in Request Chain**
  e.g. PageGraph to determine request causes

- **Don’t Break Pages**
  e.g. PageGraph to understand impact of blocking

- **Generate AdBlock Compatible Rules**
  e.g. shippable and shareable
We observed a non-trivial number of corner cases where Adwork requests, among many others. This additional attribution work to PageGraph is the AdGraph project, which also modifies these and a large number of similar corner cases.

AdGraph records. For example, PageGraph tracks image requests (e.g. the number of requests, bringing privacy and performance improvements. Frequently changing, one-off range of users. Moving higher in the request chain means we are more likely to be useful over time, and to a wider target the frequently changing image URLs will result in

3.4 Generalizing Filter Rules

3.3.2 Diagnosing Filter Rules from Existing Work.

We build these request chains for both images (and frames) our advertisement to be included in the page (e.g. the script that fetched the data gathered by the previously described image classification attribute (e.g. the script that fetched the image). Instrumentation to determine how far up each request chain we can block is not easy user serving, page operations. PageGraph differs in addition to

3.5 Safe Blocking in Request Chains

Figure 5: Example of a request chain, ending in an inserted ad image.
Where to Safely Cut?

1. Start with identified ad…

2. If a script creates more then two subtrees in the document, stop

3. If script inserts a script that that does #2, stop

4. Otherwise, block and continue up
We observed a non-trivial number of corner cases where AdGraph rules in existing work requests, among many others. This additional attribution work to PageGraph is the AdGraph project, which also modifies AdGraph records. For example, PageGraph tracks image requests (e.g. JavaScript stack is reset through events like timer callbacks frequently changing, one-on-one, more likely to programmatically identify ad libraries for a range of users. Moving higher in the request chain means we are more likely to be useful over time, and to a wider URL of the ad library itself will rarely change. Approaches that specifically benefit from an understanding of context can block without breaking the page. Our approach to maximize the benefit also generalizes the benefit to maximize the benefit to a wider range of users.

3.4 Generalizing Filter Rules

allows for greater understanding of the context scripts execute in. PageGraph handles these and a large number of similar corner cases.

3.3.2 Determining Page Breakage.

proves cause-attribution in the graph, or correctly determines which JavaScript unit is responsible for each modification. We build these request chains for both images (and frames) our instrumented blink and v8 systems in chromium to build a graph-representation of page execution. PageGraph determines whether blocking a script is “conservative” (i.e. prefers false negatives over false positives), and that second script inserted an advertising image into the page. This subsection describes how we determine whether blocking a script is “conservative” (i.e. prefers false negatives over false positives), and that second script inserted an advertising image into the page.

3.5 Safe Blocking in Request Chains

To generate optimized serving ad images and frames using the classification data gathered by the previously described image classification, we determine whether blocking a script is “conservative” (i.e. prefers false negatives over false positives), and that second script inserted an advertising image into the page. This subsection describes how we determine whether blocking a script is “conservative” (i.e. prefers false negatives over false positives), and that second script inserted an advertising image into the page.

If a script inserts another script that matches condition #1, then block in a request chain without breaking the page. Our approach to maximize the benefit also generalizes the benefit to maximize the benefit to a wider range of users.

3.5.1 Determining Page Breakage.

To generate optimized serving ad images and frames using the classification data gathered by the previously described image classification, we determine whether blocking a script is “conservative” (i.e. prefers false negatives over false positives), and that second script inserted an advertising image into the page. This subsection describes how we determine whether blocking a script is “conservative” (i.e. prefers false negatives over false positives), and that second script inserted an advertising image into the page.

3.3 Diagnosing Ad Image Presence

The most relevant related work to the PageGraph project is “erasing references from Existing Work.” The most relevant related work to the PageGraph project is “erasing references from Existing Work.”

Figure 5: Example of a request chain, ending in an inserted ad image.
Project In a Nutshell

- **Image and Frame Classifier**  
  Heavy, offline, contextual

- **Block High in Request Chain**  
  e.g. PageGraph to determine request causes

- **Don’t Break Pages**  
  e.g. PageGraph to understand impact of blocking

- **Generate AdBlock Compatible Rules**  
  e.g. shippable and shareable
URLs → Filter Rules

1. Take URL
2. Reduce to eTLD+1
3. Remove query params and fragment
4. Remove protocol

1. https://a.good.example.com/ad.html#banner?id=3
2. https://example.com/ad.html#banner?id=3
3. https://example.com/ad.html
4. ||example.com/ad.html
<table>
<thead>
<tr>
<th>Country</th>
<th>Current Lists</th>
<th>Classifier</th>
<th>∪ Chains</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albania</td>
<td>3,021</td>
<td>451</td>
<td>521</td>
<td>17.2%</td>
</tr>
<tr>
<td>Hungary</td>
<td>2,619</td>
<td>549</td>
<td>736</td>
<td>28.1%</td>
</tr>
<tr>
<td>Sri Lanka</td>
<td>2,204</td>
<td>512</td>
<td>644</td>
<td>29.2%</td>
</tr>
<tr>
<td>Total</td>
<td>7,844</td>
<td>1,512</td>
<td>1,901</td>
<td>24.2%</td>
</tr>
</tbody>
</table>
Take Aways…

- Filter lists are good (if not great)
- Right now they fail folks who need the most
- ML + serious consideration can help
- Brave will be shipping soon
Overview

- **PageGraph**
  Novel, open source, DOM attribution system

- **Behavioral Blocking**
  Protecting Privacy Beyond URLs

- **Regional Filter List Generation**
  Privacy protection for users in underserved regions

- **Automated Web Compatibility Testing**
  Making sure we’re only breaking bad stuff…
Michael Smith - mds009@ucsd.edu
University of California at San Diego
Automated WebCompat: In A Slide…

- **Brave makes aggressive modifications**
  And not just Brave…

- **We’re Flying Blind**
  Privacy community only has vague intuitions if / when we’re breaking things

- **This Hurts Privacy**
  Uncertainty -> risk averseness, hard to iterate to improvements

- **This Makes People Want Other Problems**
  Brenden’s Twitter blows up, I work late, my plants die 😥
WebCompat Problem Cavalcade

- Missing desired images
- Missing / broken event registrations
- Media playback
- Navigation problems
- Unstyled content

- Authentication
- Broken redirections
- Missing embedded content
- Form submissions
- Etc etc etc etc etc (etc!)
What Makes Web Compat Difficult

- **Existing work focuses on fulfilling promises to site owners**
  - WebCompat.org: [https://webcompat.com/](https://webcompat.com/)
  - etc...

- **We want to measure subjective user experience**
  - “If I break standard X, will users get upset”

- **Limited existing work**
  - Academic work is useful but naive
  - Remaining, subjective work has to do with accessibility
Bounding the Problem (First Cut)

- Recall > Precision
- Top level frame and local frames
- Target things that are broken from the get go
Random script
jQuery
Take Aways…

- Privacy community is drowning in suggestions
- We ignore half the ledger
- Useable privacy -> More privacy
Wrapping Up

- **PageGraph**
  Novel, open source, DOM attribution system

- **Behavioral Blocking**
  Protecting Privacy Beyond URLs

- **Regional Filter List Generation**
  Privacy protection for users in underserved regions

- **Automated Web Compatibility Testing**
  Making sure we’re only breaking bad stuff…
Internships

- **Research focused**
  Papers published at S&P, WWW, USENIX, MADWeb, etc.

- **Engineering focused**
  Shipping features to related to measurement, improved blocking, detection, and more

- **San Francisco and London offices**
  [https://brave.com/careers/?gh_jid=896018](https://brave.com/careers/?gh_jid=896018)
Thank You

• Brave is Uniquely Doing Deployment-Minded Privacy Research
• Short Term: Better Regional Filter Lists
• Medium Term: Behavioral Based Blocking
• Long(ish) Term: Automatic WebCompat

Pete Snyder
Privacy Researcher
pes@brave.com
@pes10k