Latex Gloves: Protecting Browser Extensions from Probing and Revelation Attacks

Alexander Sjösten, Steven Van Acker, Pablo Picazo-Sanchez, Andrei Sabelfeld
Browser extensions

- Allows users to modify browser behaviour
  - Block advertisement & tracking scripts
  - Password managers

- Written in a combination of JavaScript, HTML and CSS
  - Content scripts
  - Background scripts

- User grants permissions

- Can inject content
  - One way through “web accessible resources”
  - `chrome-extension://` and `moz-extension://`
Google Cast example

 Detect google cast extension
  `chrome-extension://boadgeojelhngndaghljhdicfkm1lpafd/cast_sender.js`

 Discover Chromecast on the network
Probing attack

1) Web page makes request to

```
chrome-extension://boadgeojelhngndaghljhdicfmllpafd/cast_sender.js
```

Sjösten et al., CODASPY 2017
Gulyás et al., WPES 2018 (demo web page: https://extensions.inrialpes.fr/)
Sanchez-Rola et al., USENIX 2017
Probing attack

1) Web page makes request to
   `chrome-extension://boadgeojelhgdaghljhdicfkml1pafd/cast_sender.js`

2) If extension is installed, resource is returned.

Sjösten et al., CODASPY 2017
Gulyás et al., WPES 2018 (demo web page: https://extensions.inrialpes.fr/)
Sanchez-Rola et al., USENIX 2017
Mozilla’s solution

moz-extension://actual-extension-id/resource.js

Randomized

moz-extension://30bb95e6-4208-4633-ab7b-5623c0b09483/resource.js
Mozilla’s solution

`moz-extension://actual-extension-id/resource.js`

Randomized

`moz-extension://30bb95e6-4208-4633-ab7b-5623c0b09483/resource.js`

“It is randomly generated for every browser instance. This prevents websites from fingerprinting a browser by examining the extensions it has installed.”

- Mozilla documentation
Mozilla’s solution

moz-extension://actual-extension-id/resource.js

Randomized

moz-extension://30bb95e6-4208-4633-ab7b-5623c0b09483/resource.js

“It is randomly generated for every browser instance. This prevents websites from fingerprinting a browser by examining the extensions it has installed.”
- Mozilla documentation

“This is something we'd like to do when we have the opportunity to make a breaking change.”
- Chrome developer forum
  https://bugs.chromium.org/p/chromium/issues/detail?id=611420#c19
Revelation attack

1) Extension injects content

`moz-extension://30bb95e6-4208-4633-ab7b-5623c0b09483/resource.js`
Revelation attack

1) Extension injects content

```javascript
moz-extension://30bb95e6-4208-4633-ab7b-5623c0b09483/resource.js
```

2) Use the recently acquired random ID to probe for a unique resource in an extension.
Revelation attack

1) Extension injects content
   
   moz-extension://30bb95e6-4208-4633-ab7b-5623c0b09483/resource.js

2) Use the recently acquired random ID to probe for a unique resource in an extension.

3) If extension is installed, resource is returned.
Extensions susceptible to revelation attack

- Filter extensions which might inject content
- Check if they have (at least) one unique path to a resource
- Check if they have (at least) one resource with unique content
Extensions susceptible to revelation attack

- Filter extensions which might inject content
- Check if they have (at least) one unique path to a resource
- Check if they have (at least) one resource with unique content

<table>
<thead>
<tr>
<th></th>
<th>Extensions total</th>
<th>Susceptible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefox</td>
<td>1,378</td>
<td>1,301 (94.41%)</td>
</tr>
<tr>
<td>Chrome</td>
<td>11,633</td>
<td>10,459 (89.91%)</td>
</tr>
<tr>
<td>Total</td>
<td>13,011</td>
<td>11,760 (90.39%)</td>
</tr>
</tbody>
</table>
How can one reset the random UUID?

<table>
<thead>
<tr>
<th></th>
<th>Linux</th>
<th>Mac OSX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restarting browser</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Updating browser</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Re-installing browser</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Updating extension</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-installing extension</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/ browser restart</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>w/o browser restart</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Incognito mode</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clearing cache and cookies</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clearing the profile</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
Firefox has been installed before. Let's get you a new copy.

Re-install

- Restore default settings and remove old add-ons for optimal performance

Built for people, not for profit
How can one reset the random UUID?

<table>
<thead>
<tr>
<th>Action</th>
<th>Linux</th>
<th>Mac OSX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restarting browser</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Updating browser</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-installing browser</td>
<td>No</td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Updating extension</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Re-installing extension</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Incognito mode</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clearing cache and cookies</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clearing the profile</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
How many extensions reveal themselves?

"exclude_matches": [ "*:/*/__chrome/newtab*" ],
"js": [ "dist/content_script_bundle.js" ],
"matches": [ "http:///*/", "https:///*/" ],
How many extensions reveal themselves?

- 3 sets of URLs
  - “real” URLs: derived from the matches attribute
  - “attackerhost” URLs: replace hostname with attacker.invalid
    http://www.example.com/abc ⇒ http://www.attacker.invalid/abc
  - “buydns” URLs: for more fine-grained regexps, e.g. http://*.com/abc
    http://www.example.com/abc ⇒ http://www.attacker.com/abc
How many extensions reveal themselves?

● 3 sets of URLs
  ○ “real” URLs: derived from the matches attribute
  ○ “attackerhost” URLs: replace hostname with attacker.invalid
    http://www.example.com/abc ⇒ http://www.attacker.invalid/abc
  ○ “buydns” URLs: for more fine-grained regexps, e.g. http://*.com/abc
    http://www.example.com/abc ⇒ http://www.attacker.com/abc

● Extract the regular expressions
  ○ 24,398 unique regular expressions
How many extensions reveal themselves?

- 3 sets of URLs
  - “real” URLs: derived from the matches attribute
  - “attackerhost” URLs: replace hostname with attacker.invalid
  - “buydns” URLs: for more fine-grained regexps, e.g. `http://*.com/abc`

- Extract the regular expressions
  - 24,398 unique regular expressions

- Performed crawling using CommonCrawl database
  - Contains ~4.57 billion URLs
  - For each regular expression: consider only first 100 matching URLs
  - For each extension: take random set of max 1000 URLs
How many extensions reveal themselves?

<table>
<thead>
<tr>
<th></th>
<th>Content-dependent</th>
<th>Any content</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromium</td>
<td>508</td>
<td>2,176</td>
<td>2,684</td>
</tr>
<tr>
<td></td>
<td>(5,908,381)</td>
<td>(31,903,741)</td>
<td>(37,812,122)</td>
</tr>
<tr>
<td>Firefox</td>
<td>68</td>
<td>154</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>(115,720)</td>
<td>(676,318)</td>
<td>(792,038)</td>
</tr>
<tr>
<td>Either browser</td>
<td>576</td>
<td>2,330</td>
<td>2,906</td>
</tr>
<tr>
<td></td>
<td>(6,024,101)</td>
<td>(32,580,059)</td>
<td>(38,604,160)</td>
</tr>
</tbody>
</table>
How many extensions reveal themselves?

<table>
<thead>
<tr>
<th></th>
<th>Content-dependent</th>
<th>Any content</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromium</td>
<td>508</td>
<td>2,176</td>
<td>2,684</td>
</tr>
<tr>
<td></td>
<td>(5,908,381)</td>
<td>(31,903,741)</td>
<td>(37,812,122)</td>
</tr>
<tr>
<td>Firefox</td>
<td>68</td>
<td>154</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>(115,720)</td>
<td>(676,318)</td>
<td>(792,038)</td>
</tr>
<tr>
<td>Either browser</td>
<td>576</td>
<td>2,330</td>
<td>2,906</td>
</tr>
<tr>
<td></td>
<td>(6,024,101)</td>
<td>(32,580,059)</td>
<td>(38,604,160)</td>
</tr>
</tbody>
</table>

Each of the 792,038 Firefox users are uniquely identifiable.
Revealed and susceptible to revelation attack?

<table>
<thead>
<tr>
<th></th>
<th>Revealed</th>
<th>Susceptible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromium</td>
<td>2,684</td>
<td>2,606 (97.09%)</td>
</tr>
<tr>
<td>Firefox</td>
<td>222</td>
<td>216 (97.30%)</td>
</tr>
<tr>
<td>Total</td>
<td>2,906</td>
<td>2,822 (97.11%)</td>
</tr>
</tbody>
</table>
Measures: Latex Gloves

(a) Probing defense

Probing defense policy:
ALLOW http://example.com → Ext.A

(b) Revelation defense

Revelation defense policy:
ALLOW Ext. A → http://example.com

Our solution
Measures: Latex Gloves

- Blacklists from browser vendors
Measures: Latex Gloves

- Blacklists from browser vendors
- Allow web pages to specify whitelists
Measures: Latex Gloves

- Blacklists from browser vendors
- Allow web pages to specify whitelists
- Users classify web pages

Sensitive | Insensitive
Countermeasures

- Long term
  - Latex Gloves
Countermeasures

- **Long term**
  - Latex Gloves

- **Short term**
  - Re-generate the random UUID more often
    - When starting the browser
  - Re-generate the random UUID when entering private browsing mode
  - Randomize the full URL, including the path
    - Helps, but is not perfect...
  - Use data URIs
Thank you!

Questions?