Browser history revisited

Michael Smith† Craig Disselkœn† Shravan Narayan† Fraser Brown* Deian Stefan†

†UC San Diego †Stanford University
Web Content

sandboxing
● https://www.google.com
● https://www.google.com/search?q=usenix+woot+2018
● https://bulkcheesewhizdelivery.com
● https://ashleymadison.com
Click here

“This points to something new!”

- https://www.google.com
- https://www.google.com/search?q=usenix+woot+2018
- https://bulkcheesewhizdelivery.com
- https://ashleymadison.com
Click here

“I’ve been there before!”
From: Andrew Clover <and@doxdesk.com>
Date: Thu, 14 Feb 2002 12:17:54 +0000
To: bugtraq@securityfocus.com
Message-ID: <20020214121754.B11742@doxdesk.com>
https://www.usenix.org/conference/woot18

Click here
https://www.usenix.org/conference/woot18

Click here  visited = false
<table>
<thead>
<tr>
<th>global_rank</th>
<th>domain_name</th>
<th>rank_country</th>
<th>rank_value</th>
<th>categories</th>
<th>demog_male</th>
<th>demog_female</th>
<th>site_title</th>
<th>keywords</th>
<th>owner_name</th>
<th>owner_domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>google.com</td>
<td>United States</td>
<td>1</td>
<td>Computers/Internet/Searching/Search_Engines/Google...</td>
<td>49</td>
<td>50</td>
<td>Google</td>
<td>Mountain View</td>
<td>aa</td>
<td>dns-</td>
</tr>
<tr>
<td>2</td>
<td>facebook.com</td>
<td>United States</td>
<td>2</td>
<td>Society/Activism/We_Are_The_99_Percent, Computers/...</td>
<td>47.5</td>
<td>52.5</td>
<td>Facebook</td>
<td>TheFacebook, Inc.</td>
<td>dom</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>youtube.com</td>
<td>United States</td>
<td>3</td>
<td>Computers/Internet/On_the Web/Web_Applications/Video...</td>
<td>50</td>
<td>49</td>
<td>YouTube - Broadcast your self</td>
<td>YouTube - Broadcast your self</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>yahoo.com</td>
<td>United States</td>
<td>4</td>
<td>Computers/Internet/On_the Web/Web_Portal...</td>
<td>49</td>
<td>51</td>
<td>Yahoo!</td>
<td>On the Web, Web_Portal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>baidu.com</td>
<td>China</td>
<td>5</td>
<td>World/Chinese_Simplified_CN/计算机/互联网/搜索/搜索引擎</td>
<td>31.5</td>
<td>75.5</td>
<td>Baidu.com</td>
<td>Chinese Simplified</td>
<td>2009 Baidu</td>
<td>baidu</td>
</tr>
<tr>
<td>6</td>
<td>wikipedia.org</td>
<td>United States</td>
<td>6</td>
<td>Computers/Open_Source/Open_Content/Encyclopedias/W...</td>
<td>51.5</td>
<td>49</td>
<td>Wikipedia</td>
<td>On the Web, Tracking</td>
<td>Wikimedia Foundation, Inc.</td>
<td>info-w</td>
</tr>
<tr>
<td>7</td>
<td>twitter.com</td>
<td>United States</td>
<td>7</td>
<td>Computers/Internet/On_the Web/Online_Communities/Social Media...</td>
<td>46</td>
<td>54</td>
<td>Twitter</td>
<td>Obvious Corp</td>
<td>twitter</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>amazon.com</td>
<td>United States</td>
<td>8</td>
<td>Shopping/Entertainment, Shopping/General_Merchandising, Shopping/...</td>
<td>40.5</td>
<td>61.5</td>
<td>Amazon.com</td>
<td>General Interest</td>
<td>Amazon.com</td>
<td>www</td>
</tr>
<tr>
<td>9</td>
<td>qq.com</td>
<td>China</td>
<td>9</td>
<td>World/Chinese_Simplified_CN/计算机/互联网/门户网站, World/C...</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>global_rank</td>
<td>domain_name</td>
<td>rank_country</td>
<td>rank_value</td>
<td>categories</td>
<td>demog_male</td>
<td>demog_female</td>
<td>site_title</td>
<td>keywords</td>
<td>owner_name</td>
<td>owner_name</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>--------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------</td>
<td>--------------</td>
<td>--------------------</td>
<td>----------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>1</td>
<td>google.com</td>
<td>United States</td>
<td>1</td>
<td>Computers/Internet/Searching/Search_Engines/Google...</td>
<td>49</td>
<td>50</td>
<td>Google</td>
<td>Mountain View</td>
<td>aa</td>
<td>dns-</td>
</tr>
<tr>
<td>2</td>
<td>facebook.com</td>
<td>United States</td>
<td>2</td>
<td>Society/Activism/We_Are_The_99_Percent, Computers/...</td>
<td>47.5</td>
<td>52.5</td>
<td>Facebook</td>
<td>TheFacebook, Inc.</td>
<td>dom</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>youtube.com</td>
<td>United States</td>
<td>3</td>
<td>Computers/Internet/On_the_Web/Web_Applications/Vide...</td>
<td>50</td>
<td>49</td>
<td>YouTube - Broadcast yourself</td>
<td>YouTube, Broadcast yourself</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>yahoom.com</td>
<td>United States</td>
<td>5</td>
<td>Computers/Internet/On_the_Web/Web_Portal/Compute...</td>
<td>49</td>
<td>51</td>
<td>Yahoo!</td>
<td>On the Web, Web_Portal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>baidu.com</td>
<td>China</td>
<td>1</td>
<td>World/Chinese_Simplified_CN/计算机/互联网/搜索/搜索引擎</td>
<td>31.5</td>
<td>75.5</td>
<td>Baidu.com</td>
<td>Chinese Simplified</td>
<td>2009 Baidu</td>
<td>baidu</td>
</tr>
<tr>
<td>6</td>
<td>wikipedia.org</td>
<td>United States</td>
<td>6</td>
<td>Computers/Open_Source/Open_Content/Encyclopedias/W...</td>
<td>51.5</td>
<td>49</td>
<td>Wikipedia</td>
<td>On the Web, Tracking</td>
<td>Wikimedia Foundation, Inc.</td>
<td>info-</td>
</tr>
<tr>
<td>7</td>
<td>twitter.com</td>
<td>United States</td>
<td>7</td>
<td>Computers/Internet/On_the_Web/Online_Communities/Social_Media...</td>
<td>46</td>
<td>54</td>
<td>Twitter</td>
<td>Obvious Corp</td>
<td>twitter</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>amazon.com</td>
<td>United States</td>
<td>4</td>
<td>Shopping/Entertainment,Shopping/General_Merchandise/Amazon.com...</td>
<td>40.5</td>
<td>61.5</td>
<td>Amazon.com</td>
<td>General Interest</td>
<td>Amazon.com</td>
<td>www</td>
</tr>
<tr>
<td>9</td>
<td>qq.com</td>
<td>China</td>
<td>2</td>
<td>World/Chinese_Simplified_CN/计算机/互联网/门户网站, World/China...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2002 – initial disclosure
2002 - initial disclosure
2010 - ~3,000 URLs/sec
2002 - initial disclosure
2010 - ~3,000 URLs/sec

Plugging the CSS History Leak
2002 - initial disclosure
2010 - ~3,000 URLs/sec

Plugging the CSS History Leak

2011 - MozAfterPaint leak (~100 URLs/sec)
2013 - ‘Pixel Perfect’ attack (~60 URLs/sec)
2002 - initial disclosure
2010 - ~3,000 URLs/sec
2011 - MozAfterPaint leak (~100 URLs/sec)
2013 - ‘Pixel Perfect’ attack (~60 URLs/sec)
2018 (CR deadline) - CVE-2018-6137 (~3,000)
2002 - initial disclosure
2010 - ~3,000 URLs/sec

Plugging the CSS History Leak

2011 - MozAfterPaint leak (~100 URLs/sec)
2013 - ‘Pixel Perfect’ attack (~60 URLs/sec)

2018 (CR deadline) - CVE-2018-6137 (~3,000)
2018 (talk) - ~6,000 URLs/sec
My Hobby: Extrapolating

As you can see, by late next month you'll have over four dozen husbands. Better get a bulk rate on wedding cake.

Number of Husbands

Yesterday TODAY

0 1
4 APIs, 4 attacks

- CSS Paint API
- CSS 3D transforms
- SVG fill-coloring
- JavaScript bytecode cache
Security-focused browsers affected

- Chrome + Site Isolation
- Chrome + ChromeZero add-on
- DeterFox
- FuzzyFox
- Brave

Unaffected

Tor Browser
historySniffer()

**input:** target URLs

**output:** visited URLs
TODO
☐ find vulnerable feature
☐ leak visited bit for a URL
☐ exfiltrate visited bit
☐ amplify bandwidth
CSS Painting API Level 1
W3C Working Draft, 10 April 2018
CSS Painting API Level 1
W3C Working Draft, 10 April 2018
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <style>
        body {
            background: url(chessboard.png) top left / cover;
        }
    </style>
</head>
<body>
</body>
</html>
```html
<html lang="en">
<head>

<meta charset="utf-8"/>

<style>
  body {
    background: url(chessboard.png) top left / cover;
  }
</style>

</head>

<body>

</body>

</html>
```
```javascript
1 class ChessboardPainter {
2     paint (canvas, geometry, properties) {
3         // ... draw chessboard pattern ...
4     }
5 }
6
7 registerPaint('chessboard', ChessboardPainter);
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <script>
        CSS.paintWorklet.addModule('paintlet.js');
    </script>
    <style>
        body {
            background: paint(chessboard);
        }
    </style>
</head>
<body>
</body>
</html>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8"/>
  <script>
    CSS.paintWorklet.addModule('paintlet.js');
  </script>
  <style>
    body {
      background: paint(chessboard);
    }
  </style>
</head>
<body>
</body>
</html>
class ChessboardPainter {
  paint (canvas, geometry, properties) {
    // ... draw chessboard pattern ...
  }
}

registerPaint('chessboard', ChessboardPainter);
paintlet.js

```javascript
Painter {
  geometry, properties) {
    new chessboard pattern ...
    Chessboard', ChessboardPainter);
```
paintlet.js

```javascript
// Chessboard pattern ...

var painter = new ChessboardPainter;
```
paintlet.js

```javascript
// ...
§ 9. Security Considerations

There are no known security issues introduced by these features.

§ 10. Privacy Considerations

There are no known privacy issues introduced by these features.
Click here
TODO
☑ find vulnerable feature
☐ leak visited bit for a URL
☐ exfiltrate visited bit
☐ amplify bandwidth
If https://ashleymadison.com is...

...unvisited

...visited
If https://ashleymadison.com is...

...unvisited

Attacker creates link pointing to https://dummy.com; visited = false

```html
<a href="https://dummy.com">
  Click here
</a>
```
If `https://ashleymadison.com` is... 

...unvisited

Attacker creates link pointing to `https://dummy.com`; visited = false

```html
<a href="https://dummy.com">Click here</a>
```

```css
a {
    background-image: paint(chessboard);
}
```
If https://ashleymadison.com is...

...unvisited

Attacker creates link pointing to https://dummy.com; visited = false

Browser does initial paint of link

```
<a href="https://dummy.com">
  Click here
</a>
```
If **https://ashleymadison.com** is...

...unvisited

Attacker creates link pointing to **https://dummy.com**; visited = false

Browser does initial paint of link
If **https://ashleymadison.com** is...

...**unvisited**

- Attacker creates link pointing to **https://dummy.com**; visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method

[Click here](#)
If [https://ashleymadison.com](https://ashleymadison.com) is...

...unvisited

- Attacker creates link pointing to [https://dummy.com](https://dummy.com); visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method

[Click here]
If \texttt{https://ashleymadison.com} is...

...unvisited

- Attacker creates link pointing to \texttt{https://dummy.com}; visited = \texttt{false}
- Browser does initial paint of link
- Browser calls paintlet’s \texttt{paint} method
- Attacker updates link to point to \texttt{https://ashleymadison.com}; visited remains \texttt{false}

\textbf{Click here}
If https://ashleymadison.com is...

...unvisited

Attacker creates link pointing to https://dummy.com; visited = false

Browser does initial paint of link

Browser calls paintlet’s `paint` method

Attacker updates link to point to https://ashleymadison.com; visited remains false
If https://ashleymadison.com is... 

...visited
If [https://ashleymadison.com](https://ashleymadison.com) is visited...

...visited

Attacker creates link pointing to [https://dummy.com](https://dummy.com); visited = false

```html
<a href="https://dummy.com">Click here</a>
```
If https://ashleymadison.com is visited... 

...visited

Attacker creates link pointing to https://dummy.com; visited = false

```html
<a href="https://dummy.com">Click here</a>
```

```
a {
    background-image: paint(chessboard);
}
```
If https://ashleymadison.com is...

...visited

Attacker creates link pointing to https://dummy.com; visited = false

Browser does initial paint of link
If https://ashleymadison.com is...  

...visited

Attacker creates link pointing to https://dummy.com; visited = false

Browser does initial paint of link
If \texttt{https://ashleymadison.com} is visited

...visited

- Attacker creates link pointing to \texttt{https://dummy.com}; visited = \texttt{false}
- Browser does initial paint of link
- Browser calls paintlet’s \texttt{paint} method
If https://ashleymadison.com is...

...visited

- Attacker creates link pointing to https://dummy.com; visited = false
- Browser does initial paint of link
- Browser calls paintlet’s paint method

Click here
If https://ashleymadison.com is... 
...visited

Attacker creates link pointing to https://dummy.com; visited = false

Browser does initial paint of link

Browser calls paintlet’s paint method

Attacker updates link to point to https://ashleymadison.com; visited becomes true, invalidates link
If https://ashleymadison.com is... 

...visited

Attacker creates link pointing to https://dummy.com; visited = false

Browser does initial paint of link

Browser calls paintlet’s paint method

Attacker updates link to point to https://ashleymadison.com; visited becomes true, invalidates link

Browser re-paints link

Click here *invalidation*

Click here *invalidation*

Click here *invalidation*
If `https://ashleymadison.com` is...

...visited

Attacker creates link pointing to `https://dummy.com`; visited = `false`

Browser does initial paint of link

Browser calls paintlet’s `paint` method

Attacker updates link to point to `https://ashleymadison.com`; visited becomes `true`, invalidates link

Browser re-paints link
If **https://ashleymadison.com** is... 

...visited

- Attacker creates link pointing to **https://dummy.com**; visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to **https://ashleymadison.com**; visited becomes `true`, invalidates link
- Browser re-paints link
- Browser calls paintlet’s `paint` method

Click here
If [https://ashleymadison.com](https://ashleymadison.com) is...

...unvisited

- Attacker creates link pointing to [https://dummy.com](https://dummy.com); visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to [https://ashleymadison.com](https://ashleymadison.com); visited remains false

...visited

- Attacker creates link pointing to [https://dummy.com](https://dummy.com); visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to [https://ashleymadison.com](https://ashleymadison.com); visited becomes true, invalidates link
- Browser re-paints link
- Browser calls paintlet’s `paint` method
If **https://ashleymadison.com** is...

...unvisited

- Attacker creates link pointing to [https://dummy.com](https://dummy.com); visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to [https://ashleymadison.com](https://ashleymadison.com); visited remains false

...visited

- Attacker creates link pointing to [https://dummy.com](https://dummy.com); visited = false
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to [https://ashleymadison.com](https://ashleymadison.com); visited becomes true, invalidates link
- Browser re-paints link
- Browser calls paintlet’s `paint` method
If `https://ashleymadison.com` is...

...unvisited

- Attacker creates link pointing to `https://dummy.com`; visited = `false`
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to `https://ashleymadison.com`; visited remains `false`

...visited

- Attacker creates link pointing to `https://dummy.com`; visited = `false`
- Browser does initial paint of link
- Browser calls paintlet’s `paint` method
- Attacker updates link to point to `https://ashleymadison.com`; visited becomes `true`, invalidates link
- Browser re-paints link
- Browser calls paintlet’s `paint` method
TODO
☑ find vulnerable feature
☑ leak visited bit for a URL
☐ exfiltrate visited bit
☐ amplify bandwidth
Paintlets can’t communicate
Paintlets can’t communicate
Paintlets can’t communicate

paintlet.js
paint()
Paintlets can’t communicate
Paintlets can’t communicate
Paintlets can’t communicate
Paintlets can’t communicate

```
main.js

paintlet.js
```
```javascript
class ChessboardPainter {
    paint (canvas, geometry, properties) {
        sleep(20); // milliseconds
    }
}
```
class ChessboardPainter {
    paint (canvas, geometry, properties) {
        sleep(20); // milliseconds
    }
}
```javascript
class ChessboardPainter {
    paint(canvas, geometry, properties) {
        sleep(20); // milliseconds
    }
}
```
TODO
☑ find vulnerable feature
☑ leak visited bit for a URL
☑ exfiltrate visited bit
☐ amplify bandwidth
Timing attacks are slow :(
Timing attacks are slow :(  

Click here 

[max bandwidth: 60 URLs/sec]
Timing attacks are slow :(
Timing attacks are slow :( 
Other covert channels are fast :)
Timing attacks are slow :(  
Other covert channels are fast :)  
registerPaint() covert channel
registerPaint() covert channel

- registerPaint() function can be called inside paintlet sandbox
- Unintended behavior: can use registerPaint() to control width of element outside paintlet sandbox
1) create weird HTML element outside paintlet

```html
<!-- in web page HTML: -->
<div id="weirdElement">&nbsp;</div>
<style>
    #weirdElement { display: inline; }
    #weirdElement::after {
        content: paint(myPainterIdentifier);
    }
</style>
```
registerPaint() covert channel

2) call registerPaint() inside paintlet

```javascript
// inside paintlet script:
registerPaint('myPainterIdentifier', PainterClass);
```
registerPaint() covert channel

3) weird element gets big width value

width = 154 pixels
3) weird element gets big width value

```
width = 154 pixels
```

```
width = 4 pixels
```
registerPaint() covert channel

visited
registerPaint() covert channel

visited

call registerPaint()
visited

call `registerPaint()`

`paintlet.js`

`web page`
`registerPaint()` covert channel

**visited**

- paintlet.js

**unvisited**

- web page

Call `registerPaint()`
registerPaint() covert channel

visited

unvisited

call registerPaint()  DON’T call registerPaint()

paintlet.js

web page
registerPaint() covert channel

visited

unvisited

call registerPaint()  DON’T call registerPaint()
registerPaint() covert channel

visited

unvisited

web page

getBoundingClientRect()
.width == 154

getBoundingClientRect()
.width == 4

main.js
registerPaint() covert channel
Demo!
TODO
☑ find vulnerable feature
☑ leak visited bit for a URL
☑ exfiltrate visited bit
☑ amplify bandwidth
4 APIs, 4 attacks

- CSS Paint API
- CSS 3D transforms
- SVG fill-coloring
- JavaScript bytecode cache
Attack: CSS 3D transforms

- Attacker makes a link expensive to render with CSS 3D transforms
- Attacker rapidly toggles the link’s destination between a dummy URL and a target URL

- Browser doesn’t need to re-render the link → paint performance is FAST
- Browser does lots of expensive re-renders for the link → paint performance is SLOW

unvisited

visited
Attack: SVG `fill`-coloring

Attacker puts a complex SVG image inside a link

Attacker sets `fill`-styles to change SVG image’s colors if link is visited

Attacker rapidly toggles the link’s destination between a "dummy URL" and a "target URL"

Browser doesn’t need to re-render the link
→ paint performance is FAST

Browser does lots of expensive re-renders for the link
→ paint performance is SLOW

unvisited

visited
Attack: JavaScript bytecode cache

Attacker injects script from target site into their own page

Attacker measures script’s compilation time

- Browser has to compile script from scratch
  → compilation time is LONG

- Browser has script’s bytecode in cache, skips most of compilation
  → compilation time is SHORT

unvisited

visited
4 APIs, 4 attacks

- CSS Paint API
- CSS 3D transforms
- SVG fill-coloring
- JavaScript bytecode cache
2002 – initial disclosure
2010 – ~3,000 URLs/sec

Plugging the CSS History Leak
“Our survey shows that several popular sites, including Alexa global top-100 sites, use privacy-violating flows to exfiltrate information about users’ browsing behavior.”
SaaS = Sniffing as a Service

“Track which sites your visitors visit. Learn how many of them have been to your competitor's site or your advertising partner's site.”

“Tealium's patent-pending technology lets you see the view-through traffic to your site by those who’ve been exposed to your press, or blog or video coverage.”
Defense: “referrer-origin labels”

https://a.com

Click here
Defense: “referrer-origin labels”

https://a.com

Click here

https://b.com
Defense: “referrer-origin labels”

Click here

History

https://a.com

Click here

https://b.com
Defense: “referrer-origin labels”

Click here

History

https://a.com

a.com

https://b.com
Defense: “referrer-origin labels”

History

Click here

https://c.com

https://b.com

Click here

https://a.com
Defense: “referrer-origin labels”

History

- a.com
- https://b.com
- c.com
- https://b.com
“Is https://b.com visited?”

---

**HTML**

**web content**

https://c.com

*Click here*

---

**IPC layer**

**renderer**

**storage engine**

a.com

https://b.com
“Is https://b.com visited?”
“Is https://b.com visited?”

web content
https://c.com
Click here

IPC layer

storage engine
a.com
https://b.com

renderer
“Is **https://b.com** visited?”

- **web content**: `https://c.com`
- **renderer**
- **IPC layer**
- **storage engine**: `https://b.com`

Click here
Defense: “referrer-origin labels”

1) Applies to:
   history data + cache data
2) Replaces prior mitigations

The Paint API vulnerability was unique because of its high-throughput. But the fundamental problem is that rendering of a web page relies on information that the web page authors should not be able to access. And adding mitigations for each rendering pathway seems like trying to patch leaks with duct tape when you could just turn off the water at the faucet. And all that duct tape just makes a sticky tangle of so many language features. E.g., Other open bugs for :visited include #2263 #2844 #2884 #2037
• Attack: invisibly determine whether exact URLs are visited
  ○ 4 APIs, 4 attacks
  ○ Major browsers affected
  ○ CVE-2018-6137:
    our highest bandwidth (~6,000 URLs/sec)
• Defense: “referrer-origin labels”