Browser history re:visited

Michael Smith[†] *Craig Disselkoen*[†] *Shravan Narayan*[†]
Fraser Brown[*] *Deian Stefan*[†]

[†]*UC San Diego* [*]*Stanford University*

Web Content


Web Content
sandboxing


Web Content
history data
sandboxing

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

- 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!"

2002

CSS visited pages disclosure

This message: [Message body] [Respond] [More options]
Related messages: [Next message] [Previous message]

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

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

Click here

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

Click here

visited = true

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

Click here

visited = false

global_rank	domain_name	rank_country	rank_value	categories	demog_male	demog_female	site_title	keywords	owner_name	own
1	google.com	United States	1	Computers/Internet/Searching /Search_Engines/Google...	49	50	Google	Mountain View	aa	dns-
2	facebook.com	United States	2	Society/Activism /We_Are_The_99_Percent, Computers/...	47.5	52.5	Facebook		TheFacebook, Inc.	dom
3	youtube.com	United States	3	Computers/Internet /On_the_Web /Web_Applications/Vid...	50	49	YouTube - Broadcast yourself			
4	yahoo.com	United States	5	Computers/Internet /On_the_Web/Web_Portals, Compute...	49	51	Yahoo!	On the Web, Web Portals		
5	baidu.com	China	1	World/Chinese_Simplified_CN /计算机/互联网络/搜寻/搜索引擎	31.5	75.5	Baidu.com	Chinese Simplified	2009 Baidu	baid
6	wikipedia.org	United States	6	Computers/Open_Source /Open_Content /Encyclopedias/W...	51.5	49	Wikipedia	On the Web, Tracking	Wikimedia Foundation, Inc.	info-
7	twitter.com	United States	7	Computers/Internet /On_the_Web /Online_Communities/S...	46	54	Twitter		Obvious Corp	twitt
8	amazon.com	United States	4	Shopping/Entertainment, Shopping/General_Merchandi...	40.5	61.5	Amazon.com	General Interest	Amazon.com	www
9	qq.com	China	2	World/Chinese_Simplified_CN /计算机/互联网络/门户网站, World/C...						bn

Source: Alexa Top Sites

bandwidth = URLs / second

global_rank	domain_name	rank_country	rank_value	categories	demog_male	demog_female	site_title	keywords	owner_name	own
1	google.com	United States	1	Computers/Internet/Searching /Search_Engines/Google...	49	50	Google	Mountain View	aa	dns-
2	facebook.com	United States	2	Society/Activism /We_Are_The_99_Percent, Computers/...	47.5	52.5	Facebook		TheFacebook, Inc.	dom
3	youtube.com	United States	3	Computers/Internet /On_the_Web /Web_Applications/Vid...	50	49	YouTube - Broadcast yourself			
4	yahoo.com	United States	5	Computers/Internet /On_the_Web/Web_Portals, Compute...	49	51	Yahoo!	On the Web, Web Portals		
5	baidu.com	China	1	World/Chinese_Simplified_CN /计算机/互联网络/搜寻/搜索引擎	31.5	75.5	Baidu.com	Chinese Simplified	2009 Baidu	baid
6	wikipedia.org	United States	6	Computers/Open_Source /Open_Content /Encyclopedias/W...	51.5	49	Wikipedia	On the Web, Tracking	Wikimedia Foundation, Inc.	info-
7	twitter.com	United States	7	Computers/Internet /On_the_Web /Online_Communities/S...	46	54	Twitter		Obvious Corp	twitt
8	amazon.com	United States	4	Shopping/Entertainment, Shopping/General_Merchandi...	40.5	61.5	Amazon.com	General Interest	Amazon.com	www
9	qq.com	China	2	World/Chinese_Simplified_CN /计算机/互联网络/门户网站, World/C...						br

Source: Alexa Top Sites

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

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)

2002 – initial disclosure

2010 – ~3,000 URLs/sec

Plugging the CSS History Leak

2011 – `MozAfterPaint` leak (~100 URLs/sec)

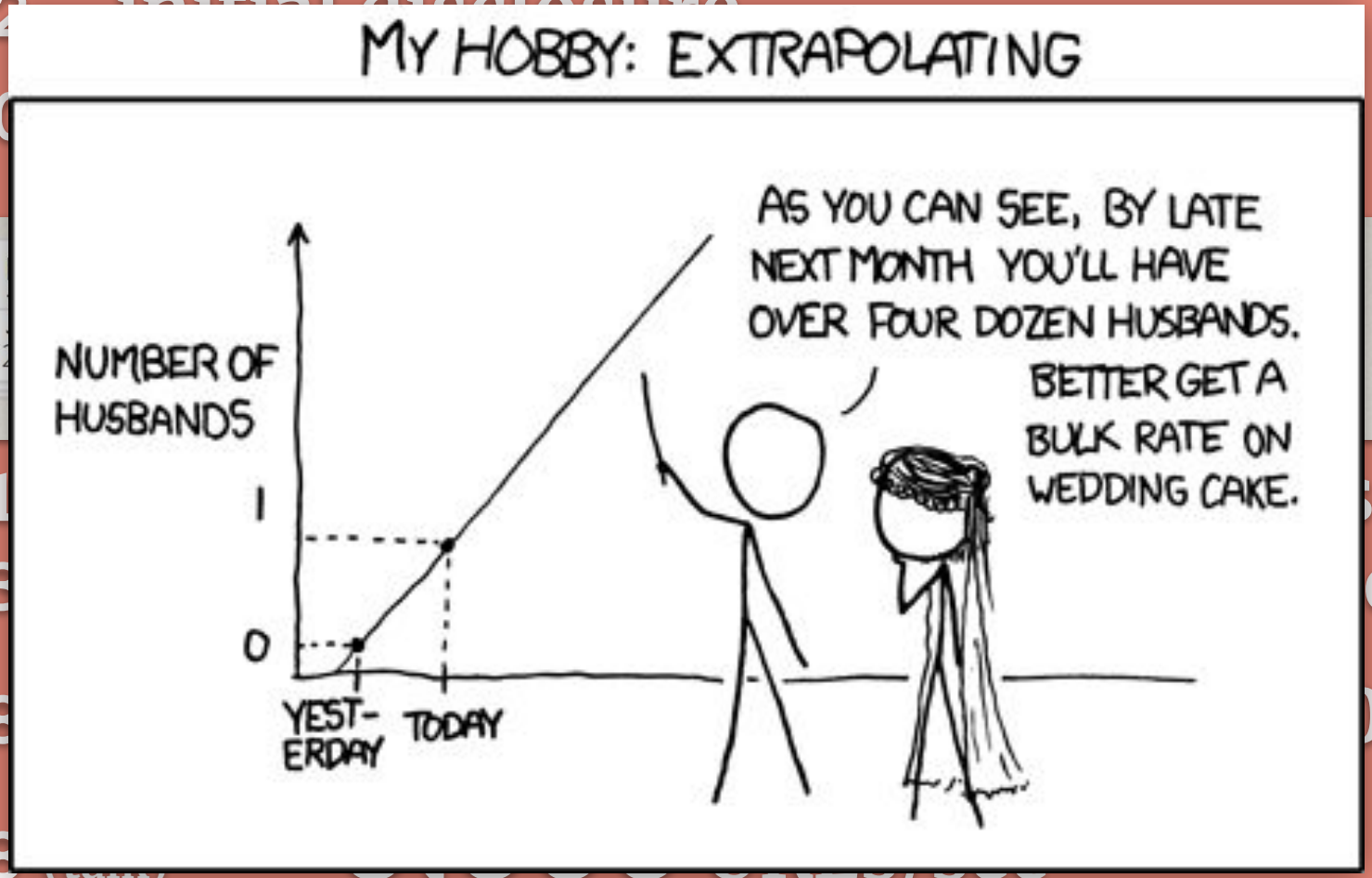
2013 – 'Pixel Perfect' attack (~60 URLs/sec)

2018 (CR deadline) – CVE-2018-6137 (~3,000)

2018 (talk) – ~6,000 URLs/sec

2002 initial disclosure

2010



2011 sec)

2013 ec)

2018 0))

2018 (talk)

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

CSS Painting API Level 1

W3C Working Draft, 10 April 2018

CSS Painting API Level 1

W3C Working Draft, 10 April 2018

CSS Painting API Level 1

W3C Working Draft, 10 April 2018

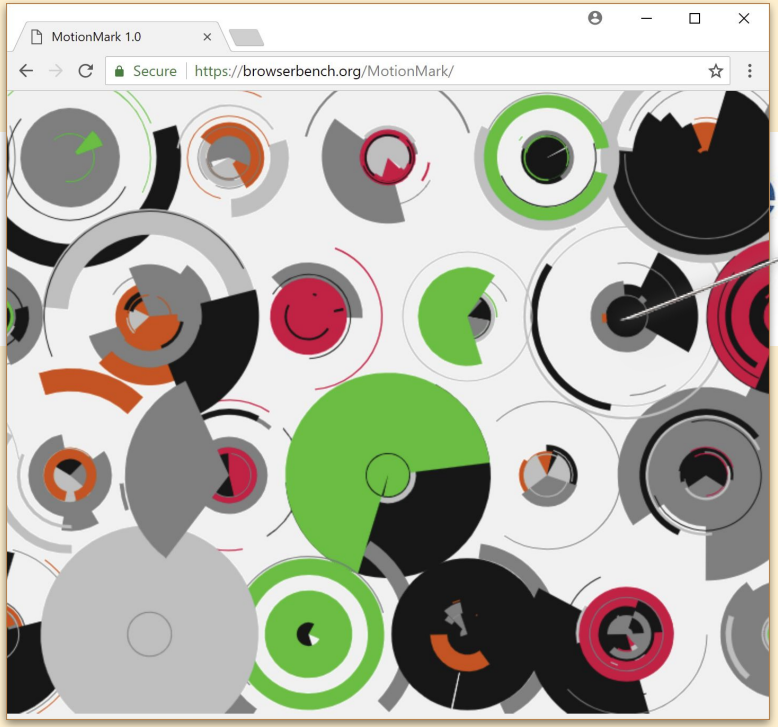

CSS Pai
W3C Working Draft, 10 April 2018
UNITED
N20904
JS


CSS Pai
W3C Working Draft, 10 Ap
UNITED
JS
Medtronic
1:31 THRESH
SUSPEND
MiniMed
B
ESC
ACT

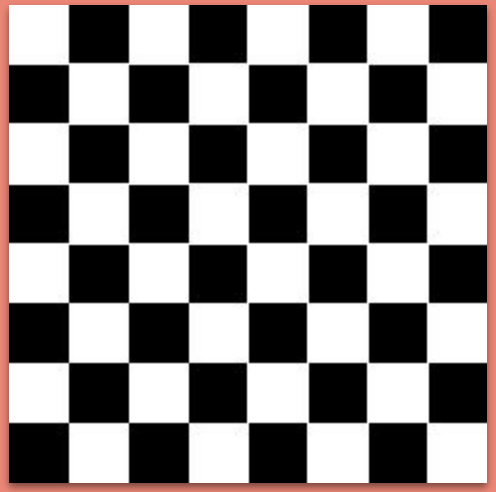

UNITED
JS
CSS
W3C Wor
1:31 THRESH
SUSPEND
Medtronic
MiniMed
B
ESC
ACT

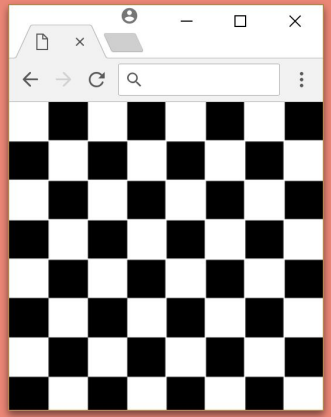
CSS Painting API Level 1

W3C Working Draft, 10 April 2018


MotionMark 1.0
Secure
https://browserbench.org/MotionMark/

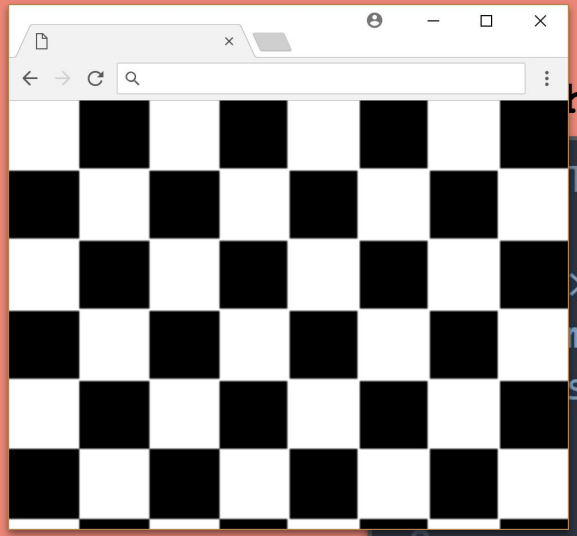

JS





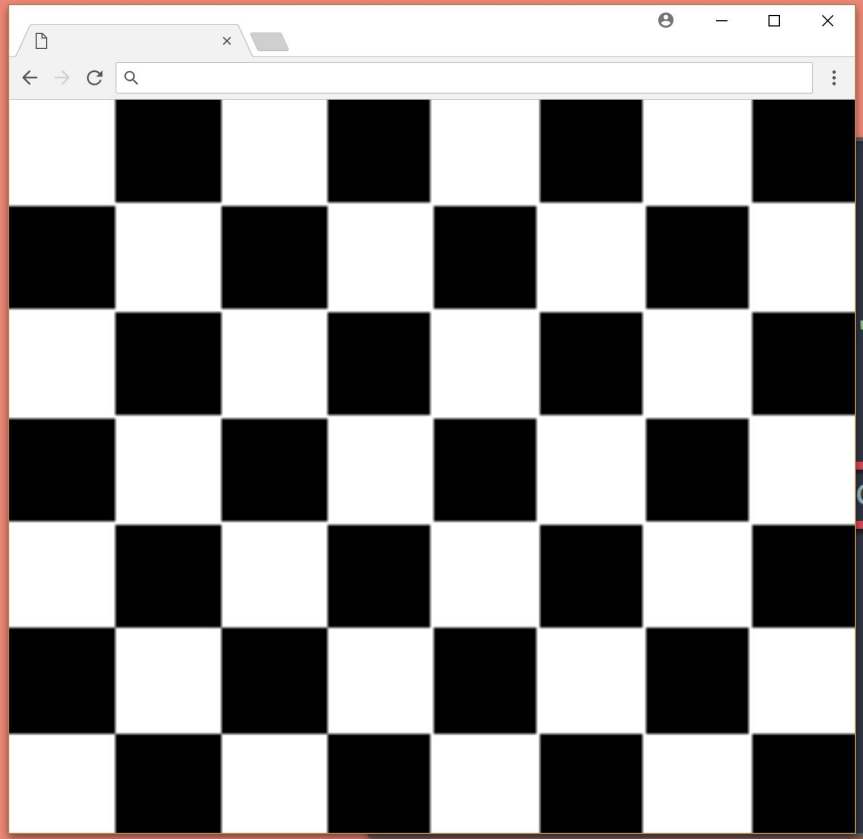
index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <style>
        body {
            background: url(chessboard.png) top left / cover;
        }
    </style>
</head>
<body>
</body>
</html>
```



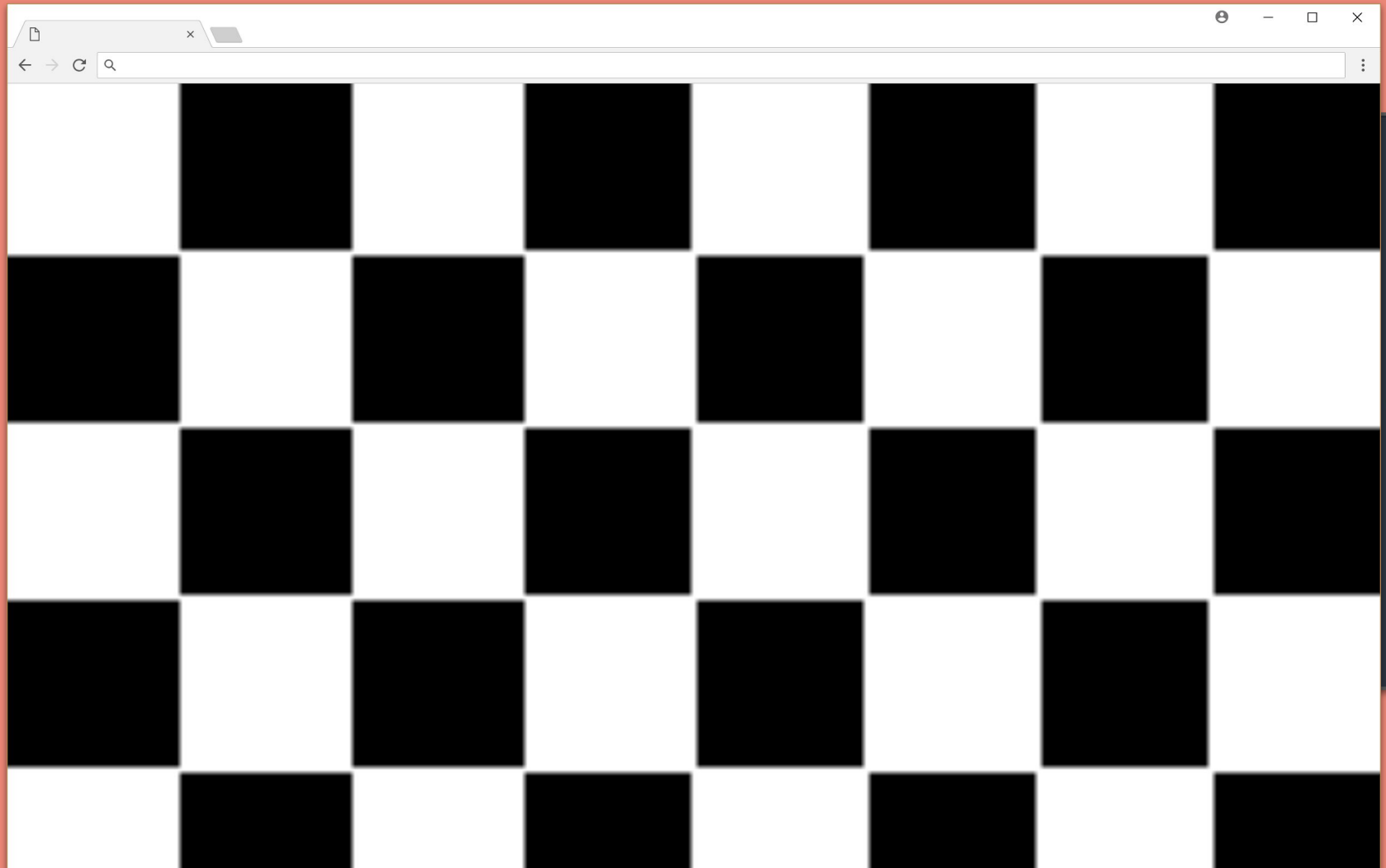
html

```
TYPE html>
 lang="en">
>
meta charset="utf-8"/>
style>
    body {
        background: url(chessboard.png) top left / cover;
    }
</style>
</head>
<body>
</body>
</html>
```



```
"utf-8"/>

ound: url(chessboard.png) top left / cover;
```



```
cover;
```

CSS Painting API Level 1

W3C Working Draft, 10 April 2018

paintlet.js

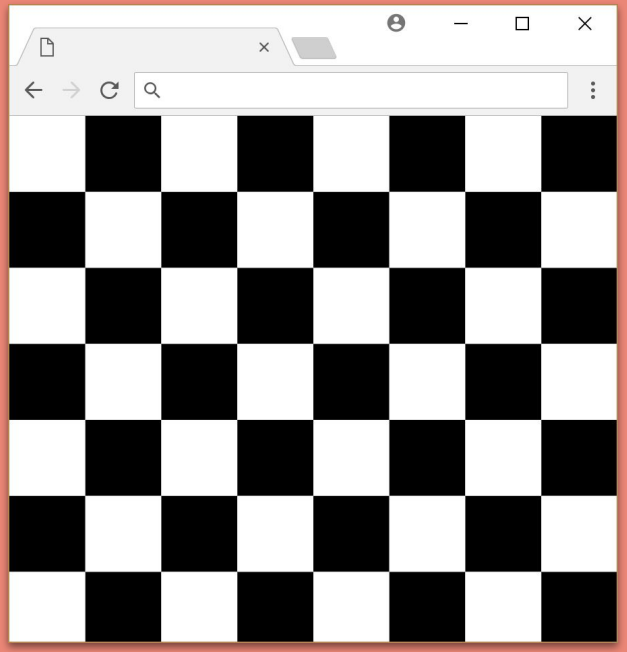
```
1 class ChessboardPainter {
2     paint (canvas, geometry, properties) {
3         // ... draw chessboard pattern ...
4     }
5 }
6
7 registerPaint('chessboard', ChessboardPainter);
```

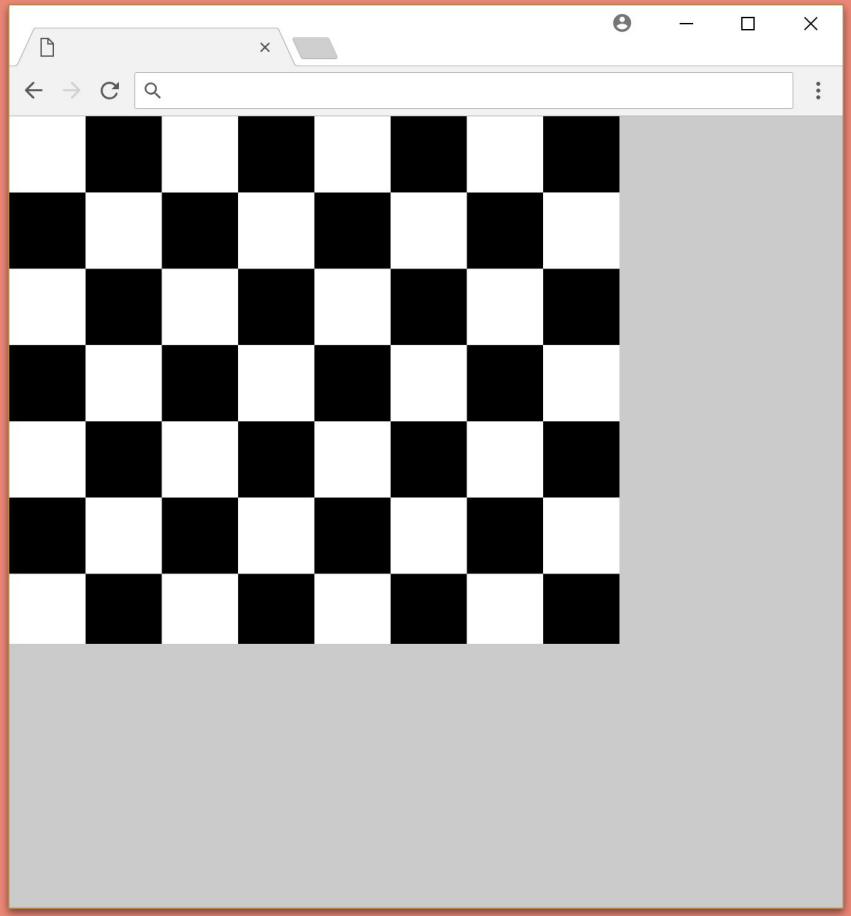
index.html

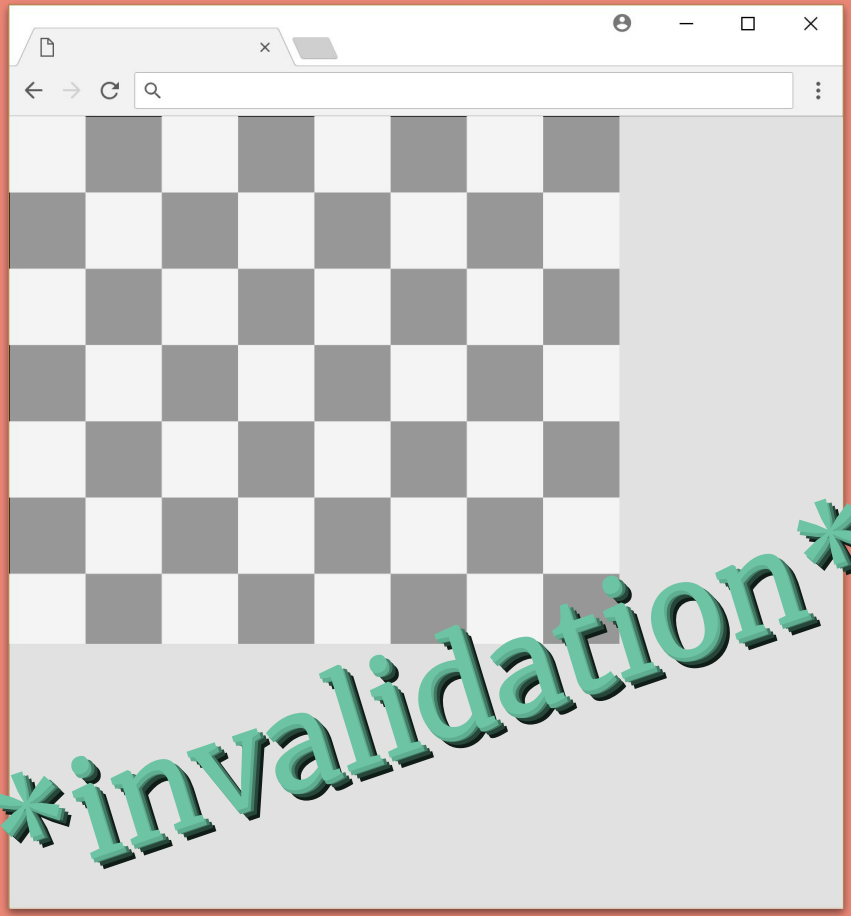
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <script>
        CSS.paintWorklet.addModule('paintlet.js');
    </script>
    <style>
        body {
            background: paint(chessboard);
        }
    </style>
</head>
<body>
</body>
</html>
```

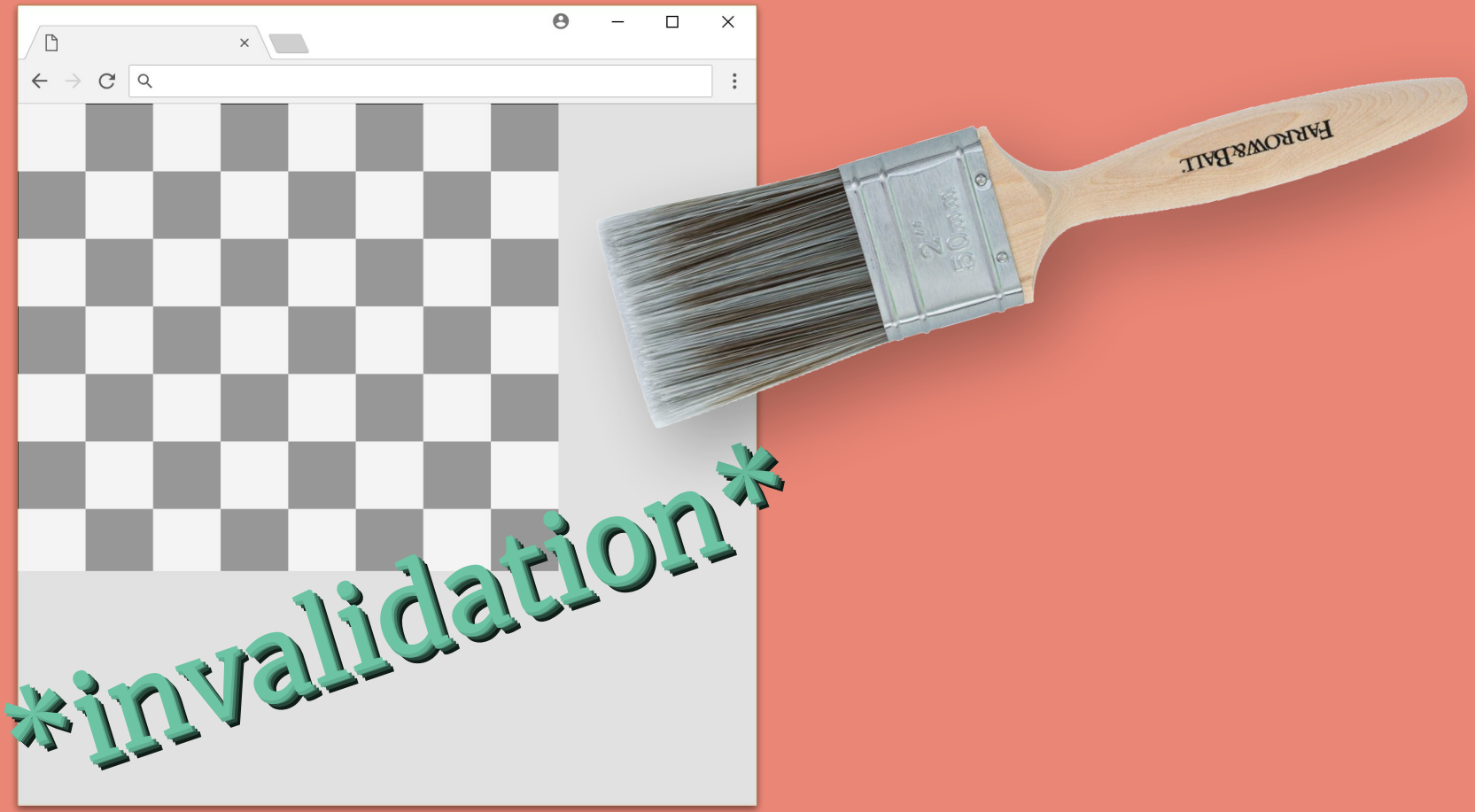
index.html

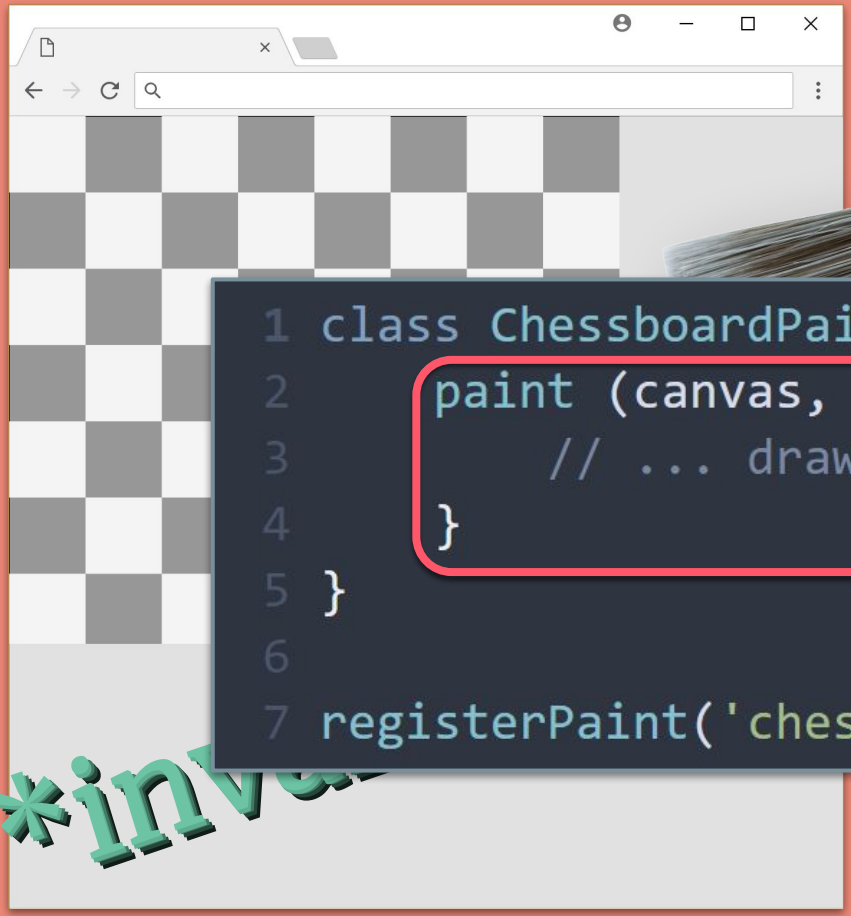
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <script>
        CSS.paintWorklet.addModule('paintlet.js');
    </script>
    <style>
        body {
            background: paint(chessboard);
        }
    </style>
</head>
<body>
</body>
</html>
```





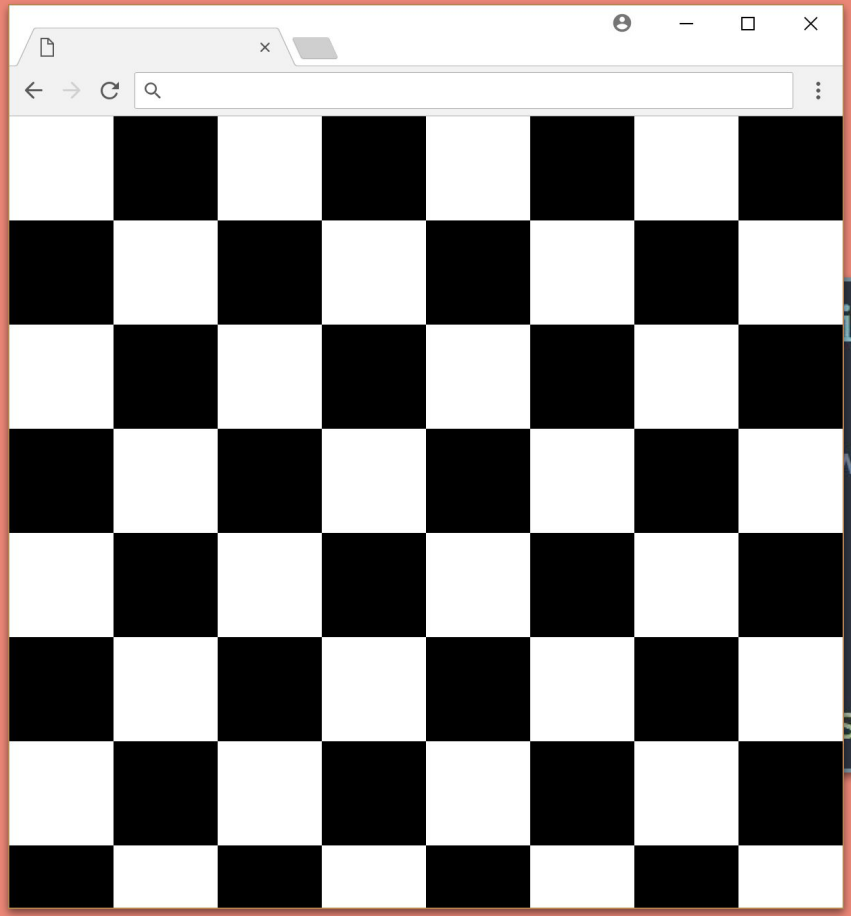

invalidation


FARROW&BALL
2"
50mm
invalidation



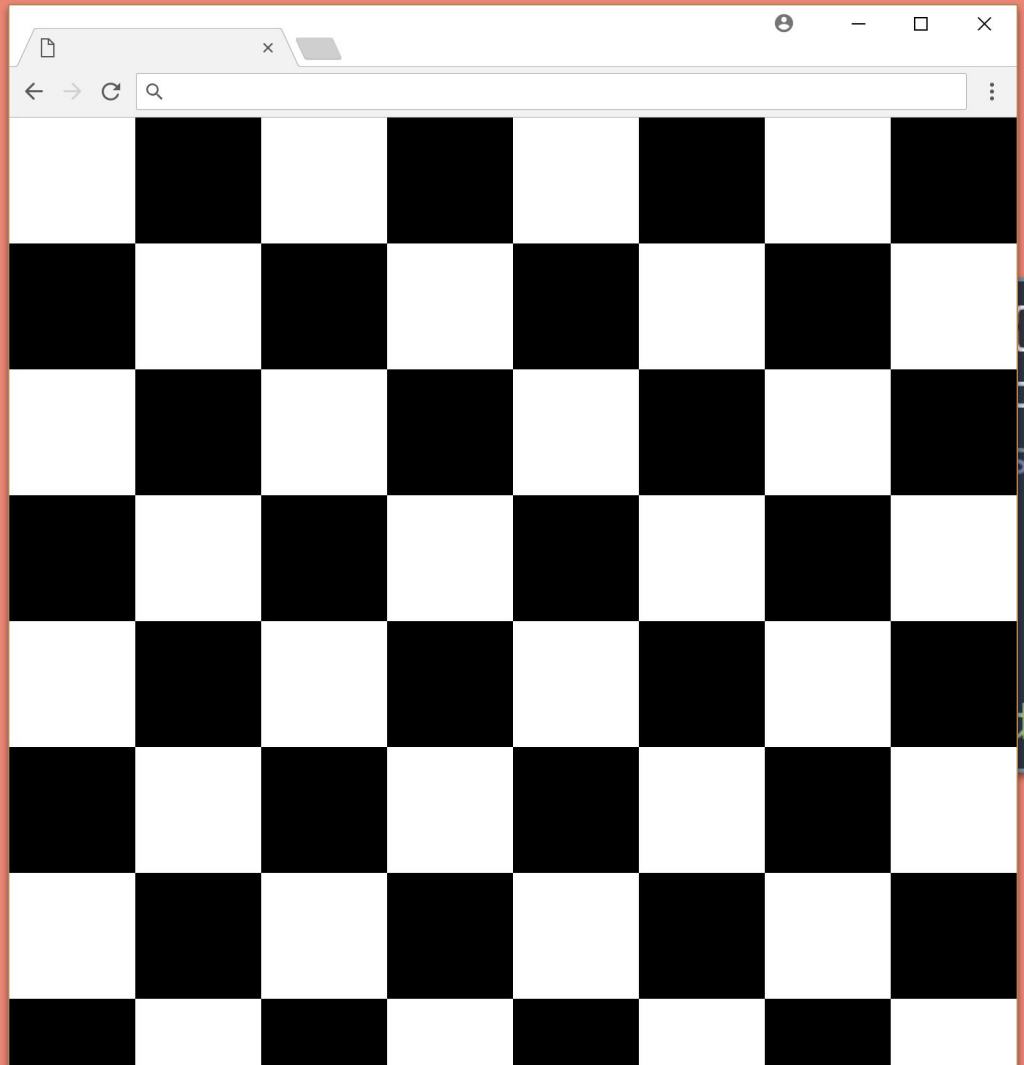

```
1 class ChessboardPainter {
2     paint (canvas, geometry, properties) {
3         // ... draw chessboard pattern ...
4     }
5 }
6
7 registerPaint('chessboard', ChessboardPainter);
```

*inva



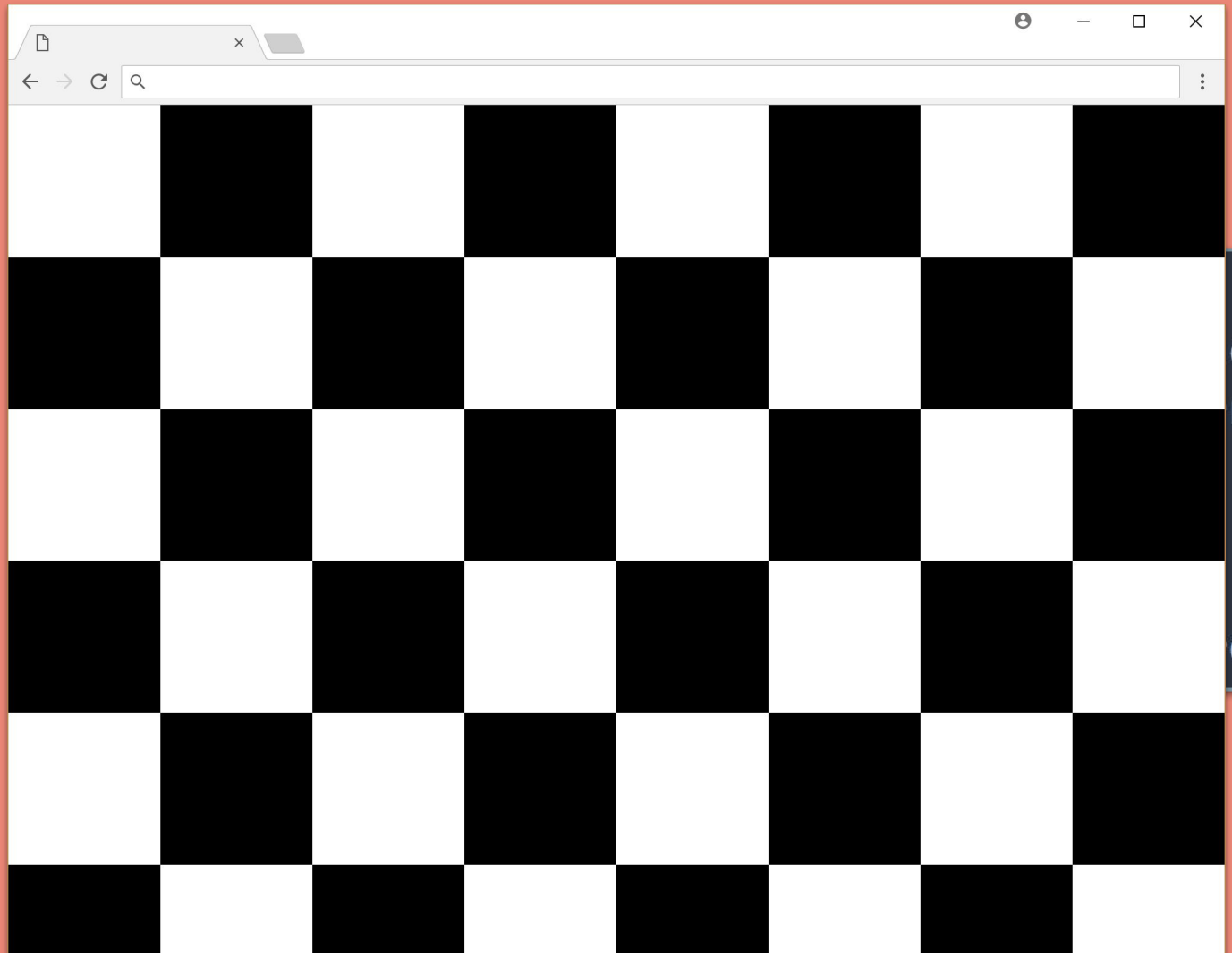
```
inter {
 geometry, properties) {
w chessboard pattern ...

ssboard', ChessboardPainter);
```



```
try, properties) {
sboard pattern ...

d', ChessboardPainter);
```



```
es) {
n ...

dPainter);
```

CSS Painting API Level 1

W3C Working Draft, 10 April 2018

§ 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.

```
a {
    background-image: paint(chessboard);
}
```

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`

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

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

...unvisited

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

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

```
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="http://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

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

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...

...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 is...

...visited

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

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

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

...visited

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

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

```
a {
    background-image: paint(chessboard);
}
```

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

```
<a      ttps://dummy.com">
    Cli   here
</a>
```

...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 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

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

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

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

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

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

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

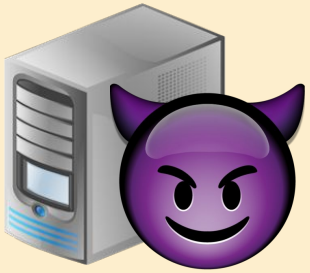
TODO

- ☑ find vulnerable feature
- ☑ leak visited bit for a URL
- ☐ exfiltrate visited bit
- ☐ amplify bandwidth

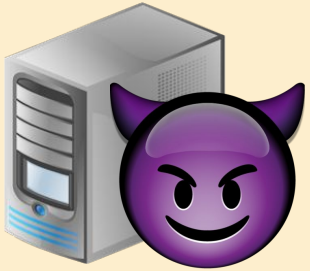
Paintlets can’t communicate

Paintlets can't communicate

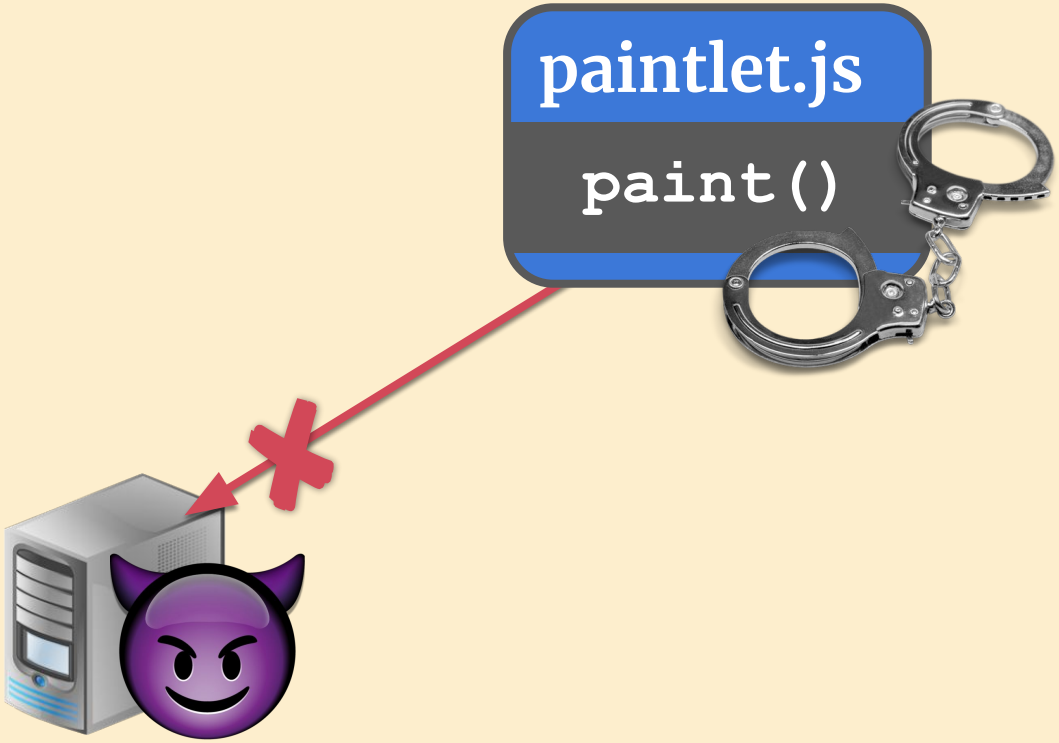
Paintlets can't communicate



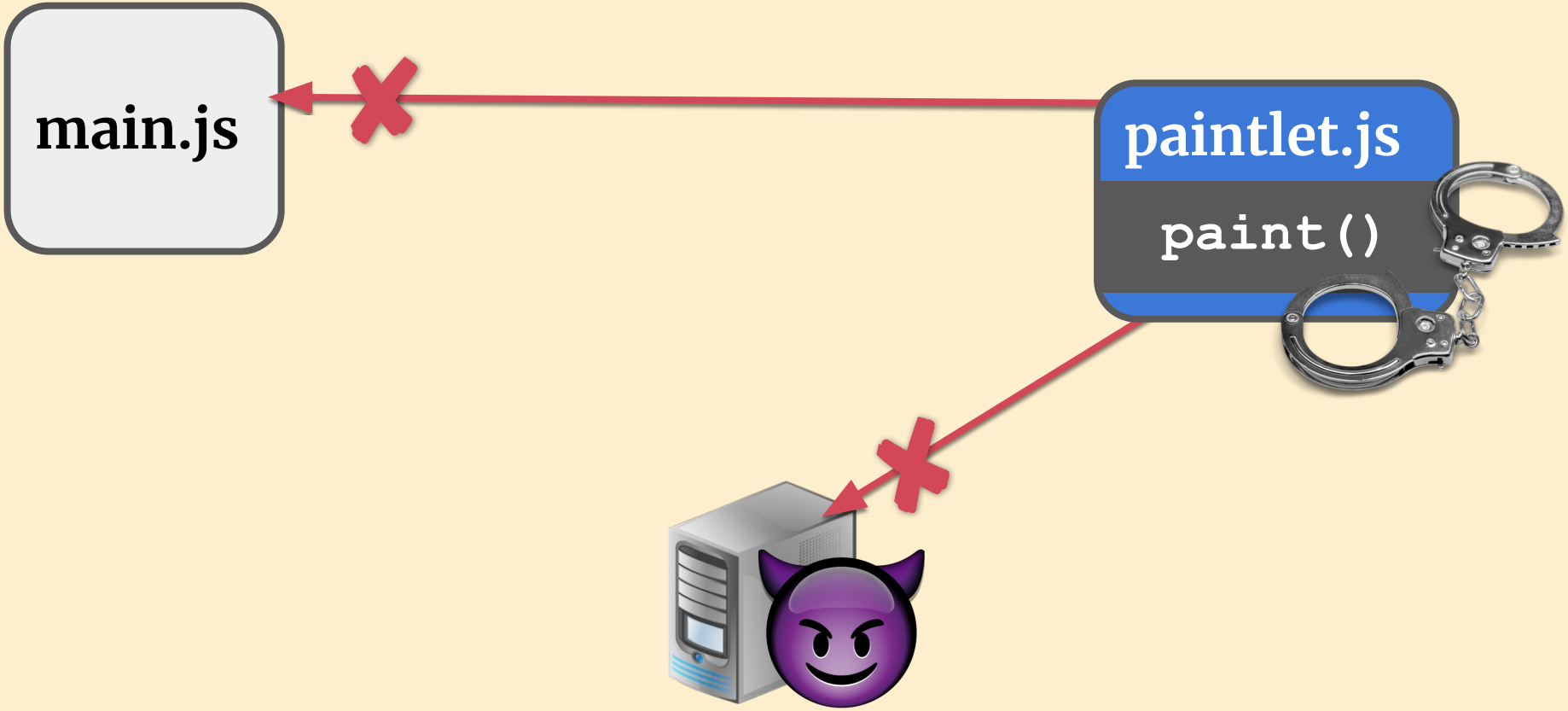
Paintlets can't communicate



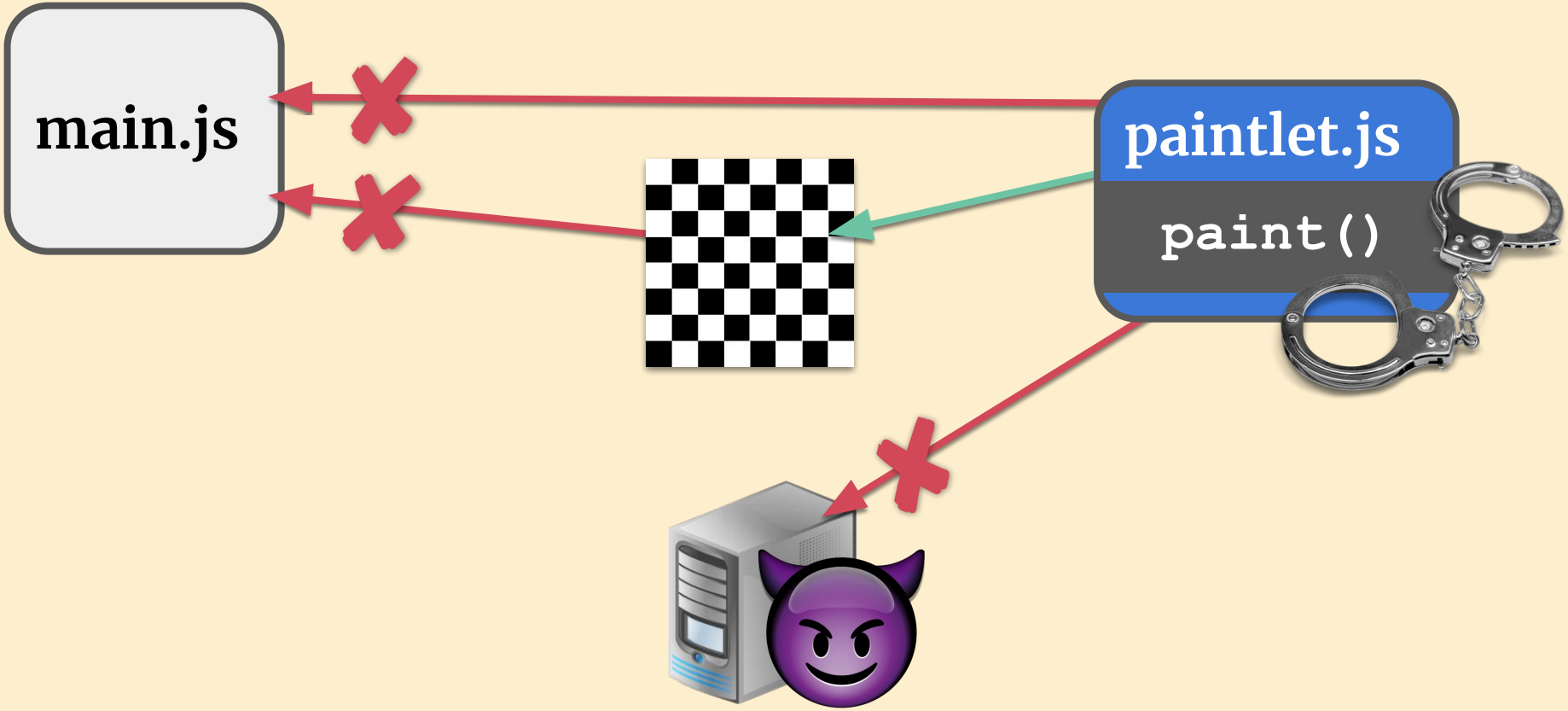
Paintlets can't communicate

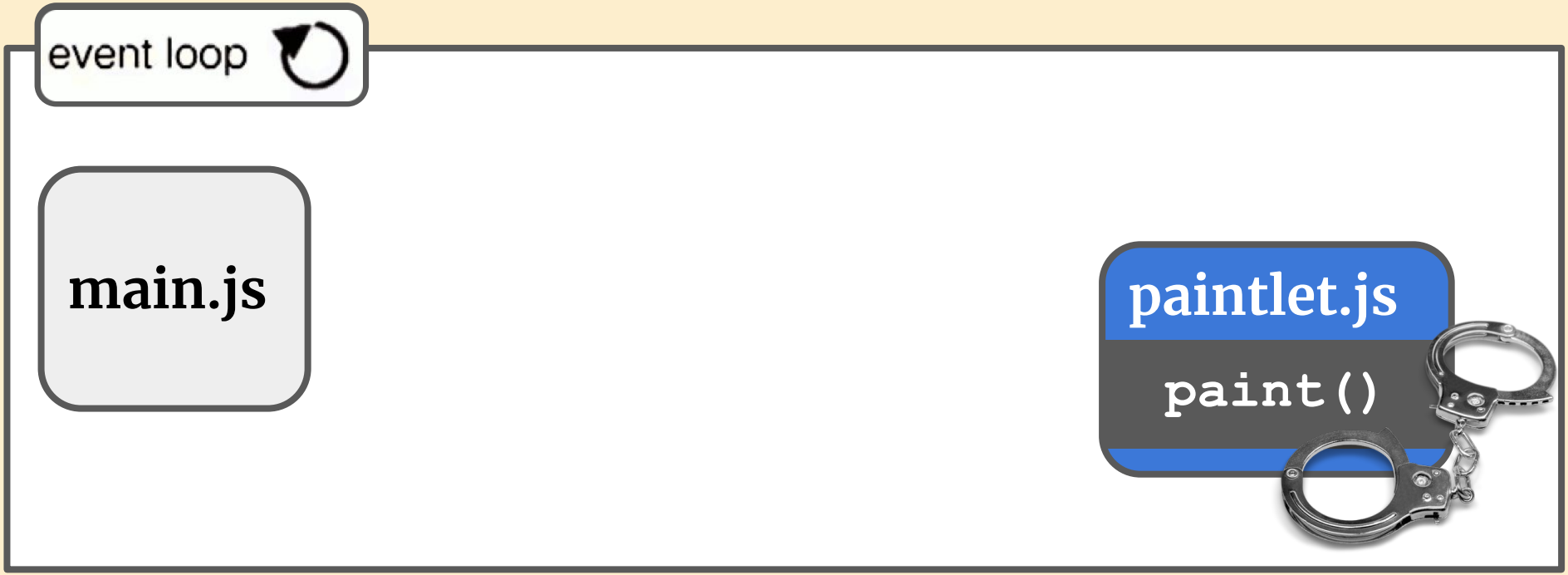


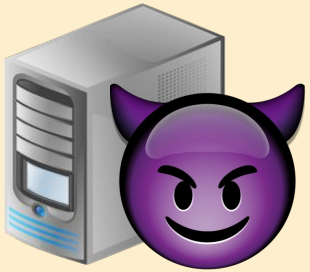
Paintlets can't communicate



Paintlets can't communicate




event loop
main.js
paintlet.js
paint()



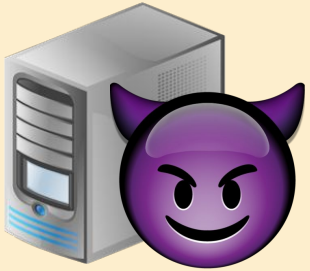
event loop

main.js

```
class ChessboardPainter {
    paint (canvas, geometry, properties) {
        sleep(20); // milliseconds
    }
}
```

paintlet.js

paint()



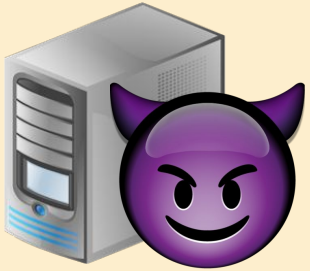
event loop

main.js

```
class ChessboardPainter {
    paint (canvas, geometry, properties) {
        sleep(20); // milliseconds
    }
}
```

paintlet.js

paint()



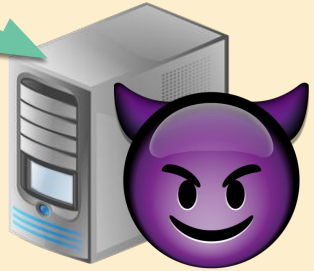
event loop

main.js

```
class ChessboardPainter {
    paint (canvas, geometry, properties) {
        sleep(20); // milliseconds
    }
}
```

paintlet.js

paint()



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 :(

Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C

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

`registerPaint()` covert channel

1) create weird HTML element outside paintlet

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

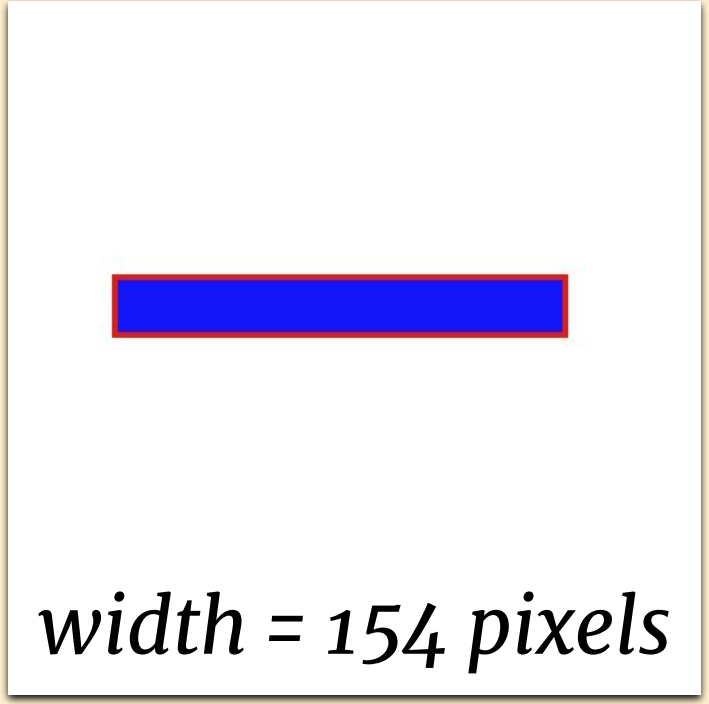
`registerPaint()` covert channel

2) call `registerPaint()` inside paintlet

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

`registerPaint()` covert channel

3) weird element gets big width value



`registerPaint()` covert channel

3) weird element gets big width value

width = 154 pixels

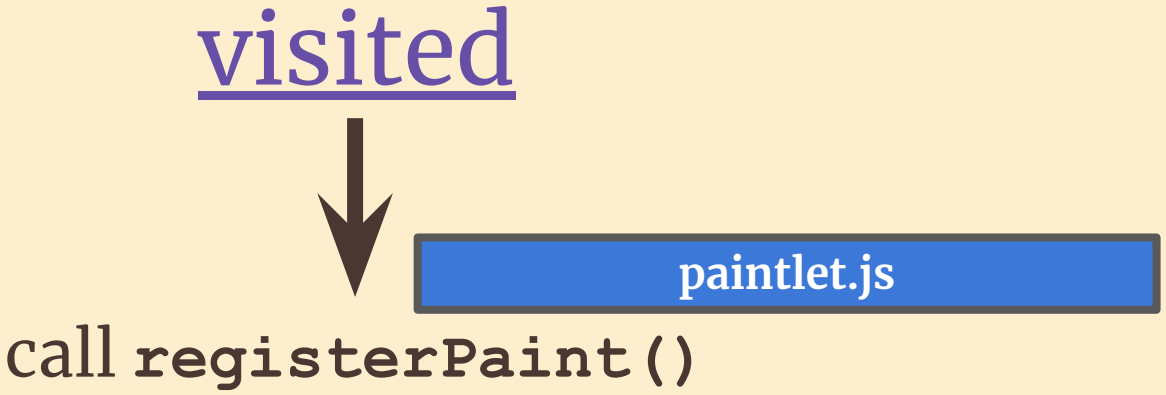
vs

width = 4 pixels

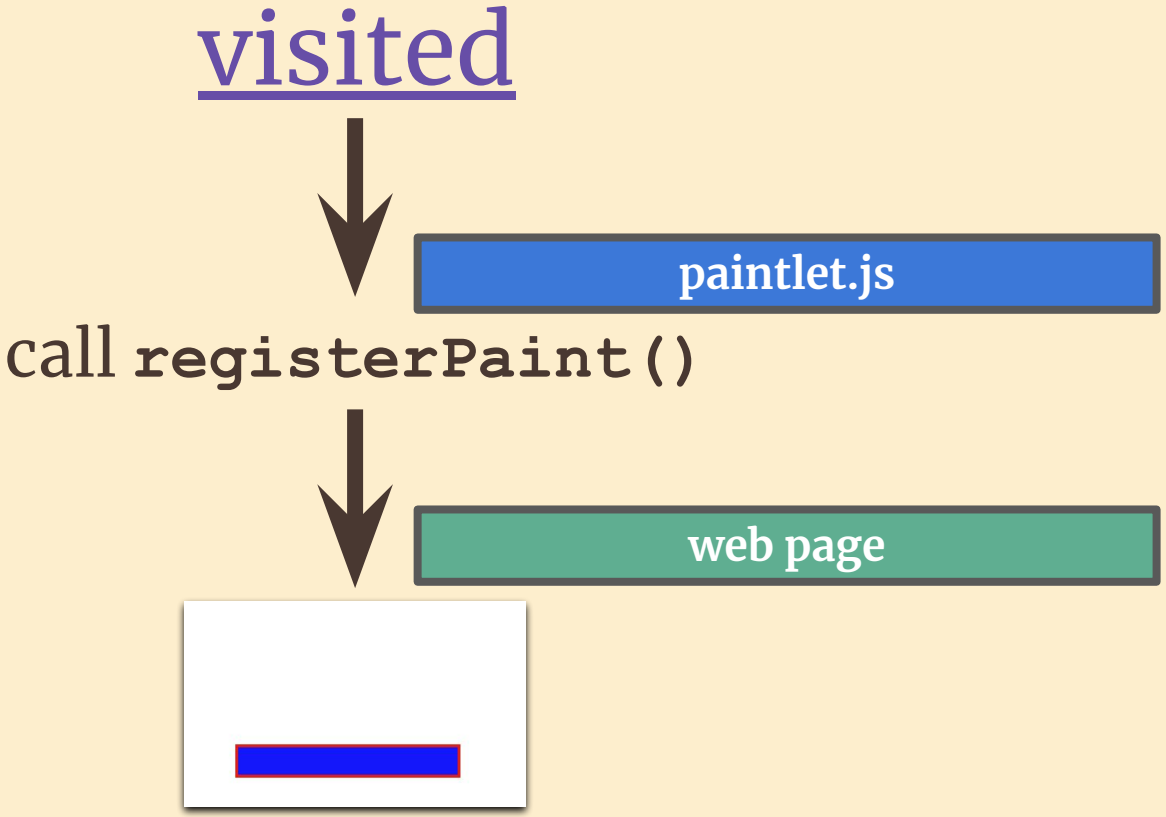
`registerPaint()` covert channel

visited

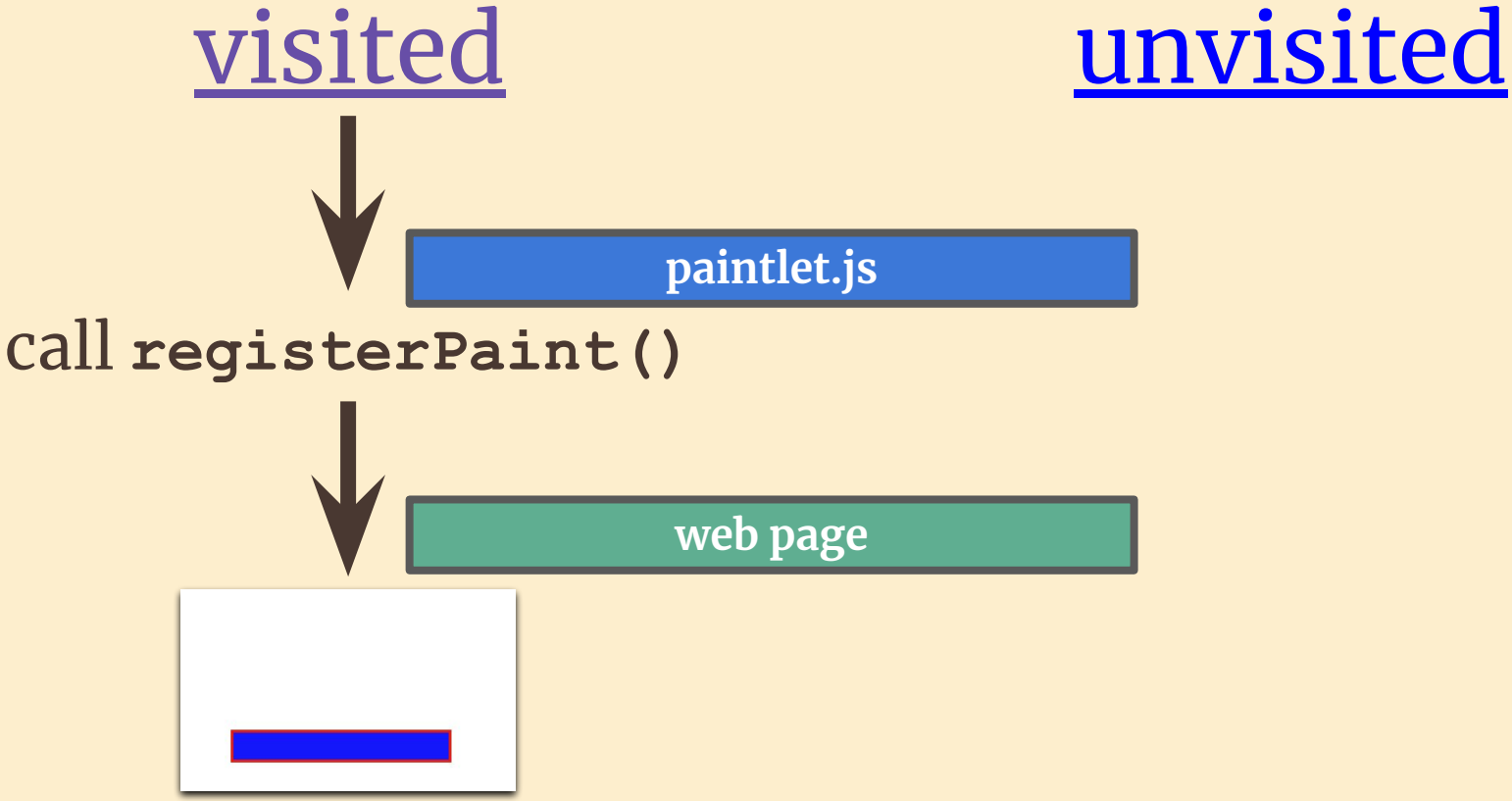
`registerPaint()` covert channel



`registerPaint()` covert channel

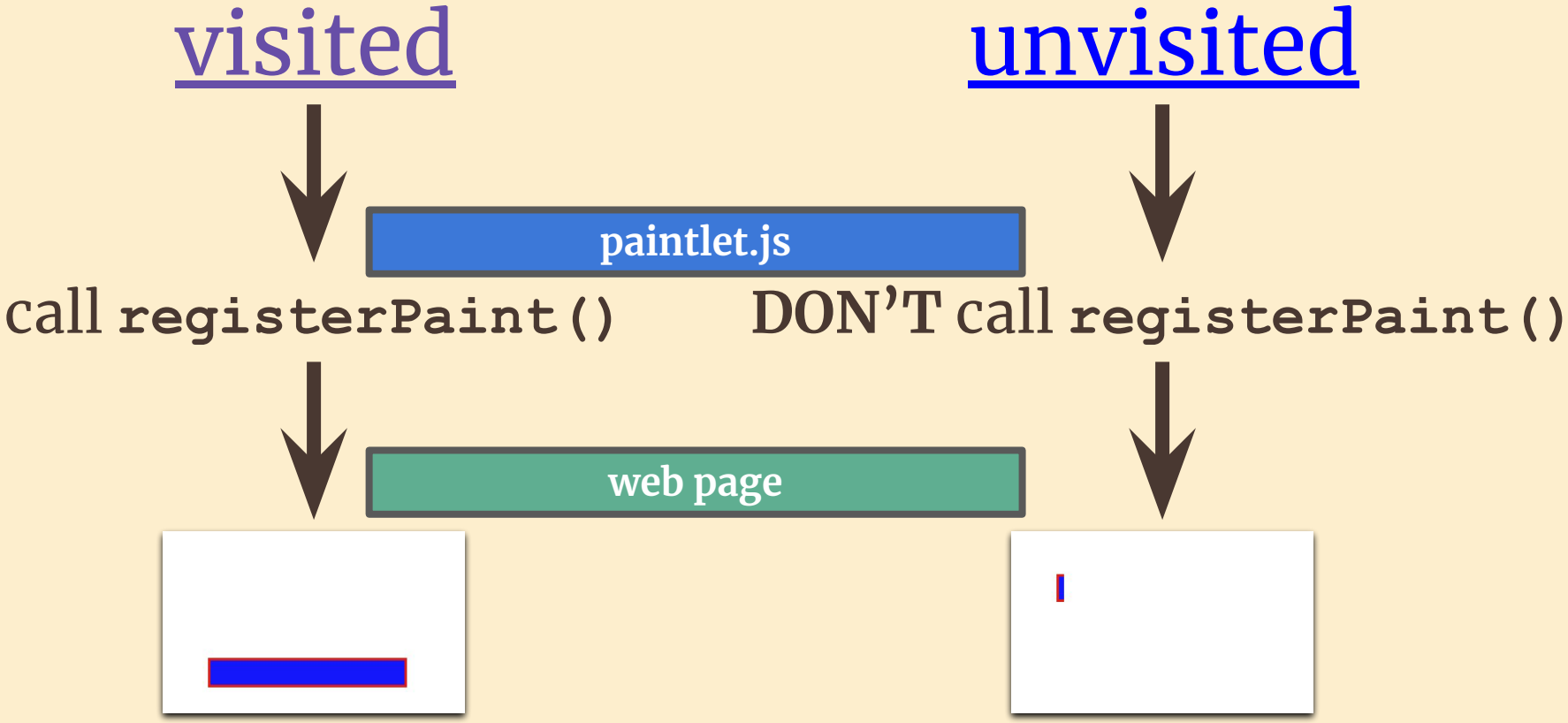


`registerPaint()` covert channel

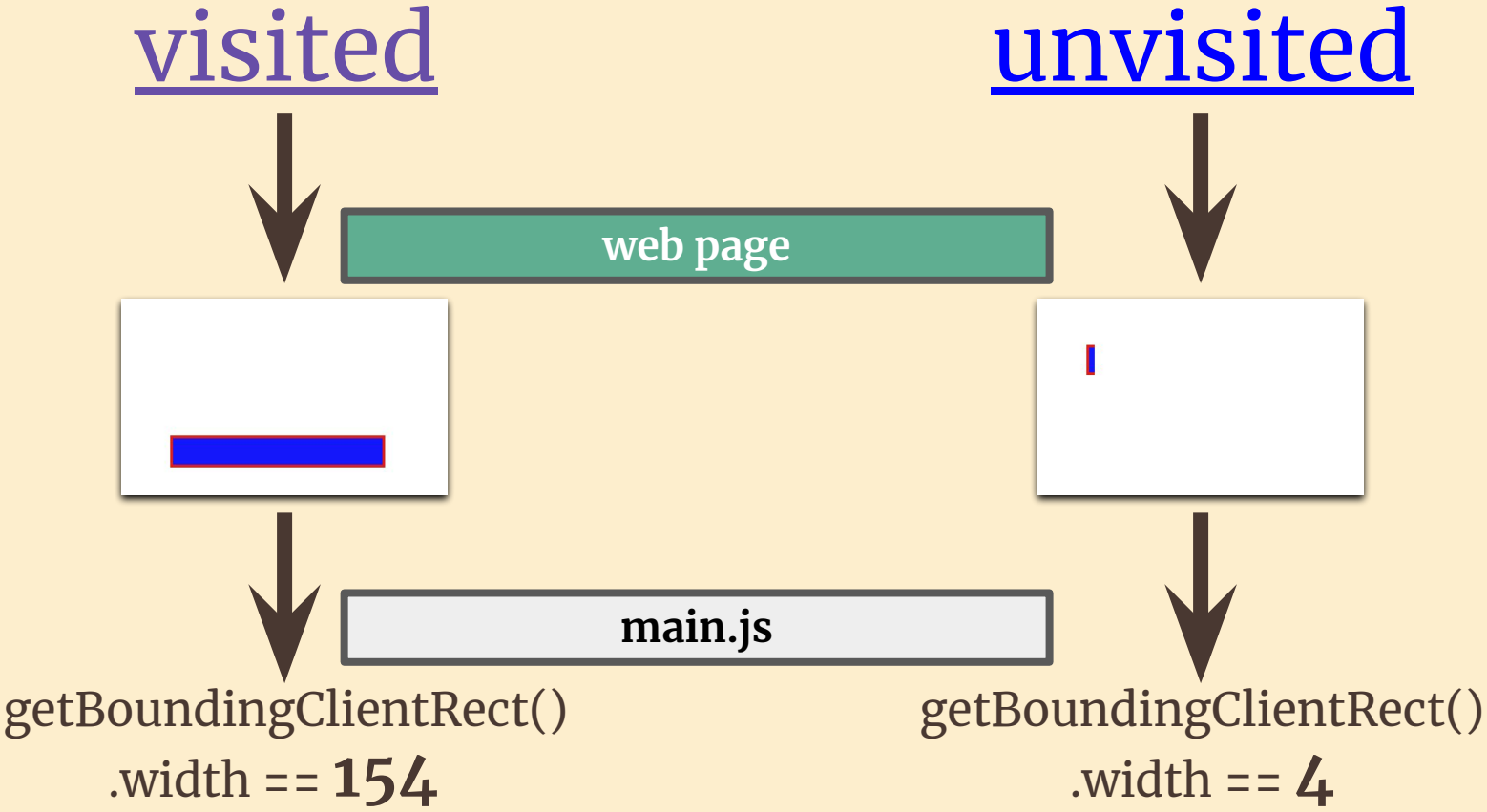


`registerPaint()` covert channel

`registerPaint()` covert channel



`registerPaint()` covert channel



`registerPaint()` covert channel

Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C
Click here Click here Click here Click here Click here Click here Click here Click here C

Demo!

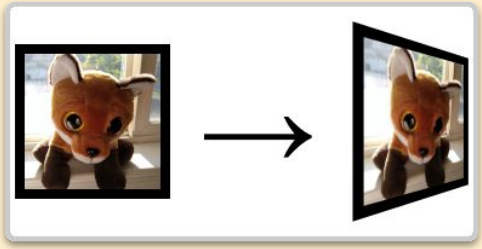
TODO

- ☑ 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*

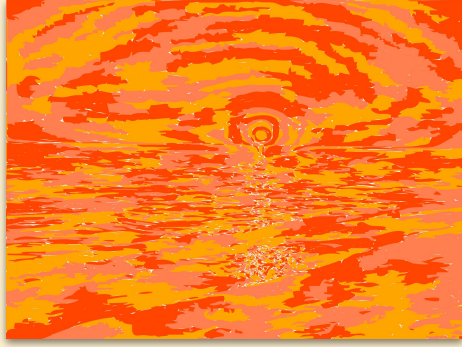
Browser doesn't need to re-render the link
→ paint performance is FAST

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*

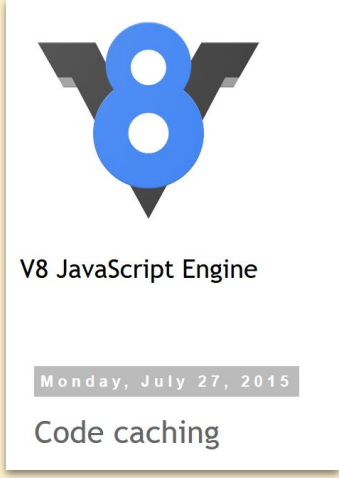
Browser doesn't need to re-render the link
→ paint performance is FAST

→ ***unvisited***

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

→ ***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

unvisited

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

visited

4 APIs, 4 attacks

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

2002 – initial disclosure

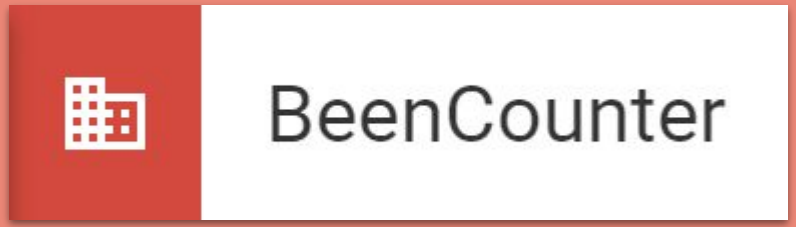
2010 – ~3,000 URLs/sec

An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications

Dongseok Jang Ranjit Jhala Sorin Lerner Hovav Shacham

"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

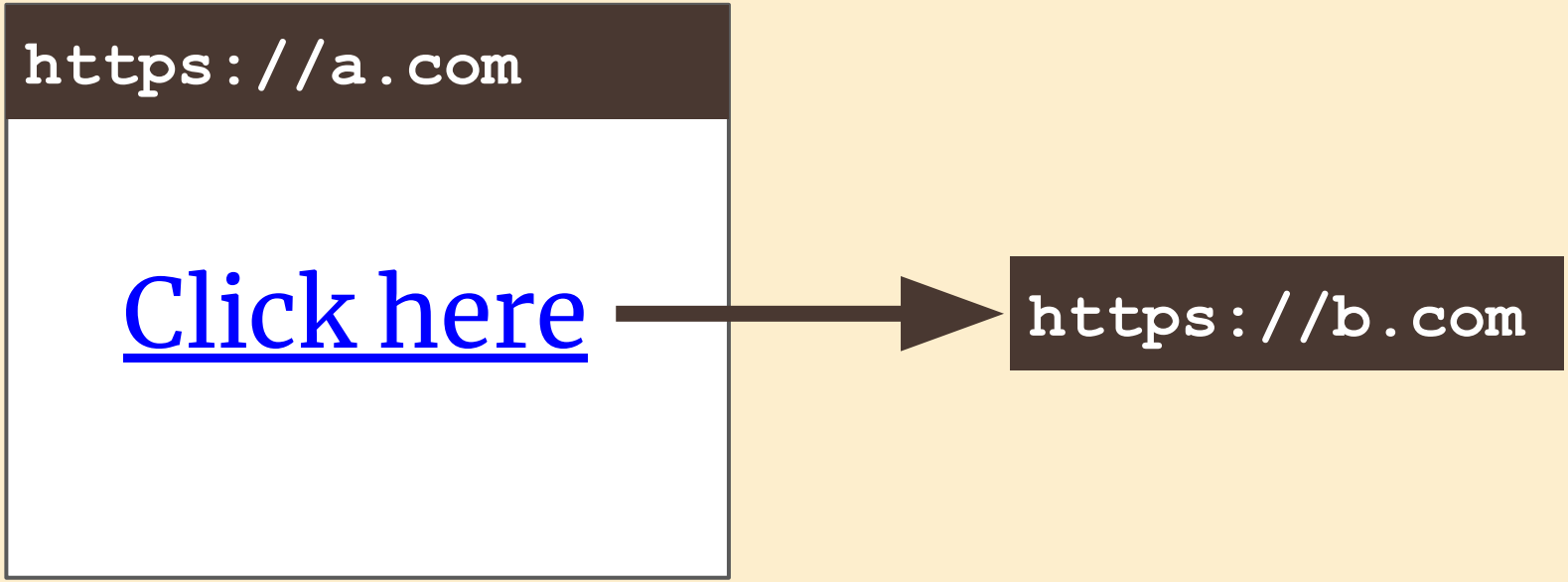


"Track which sites your visitors visit. Learn how many of them have been to your competitor's site or your advertising partner's site."

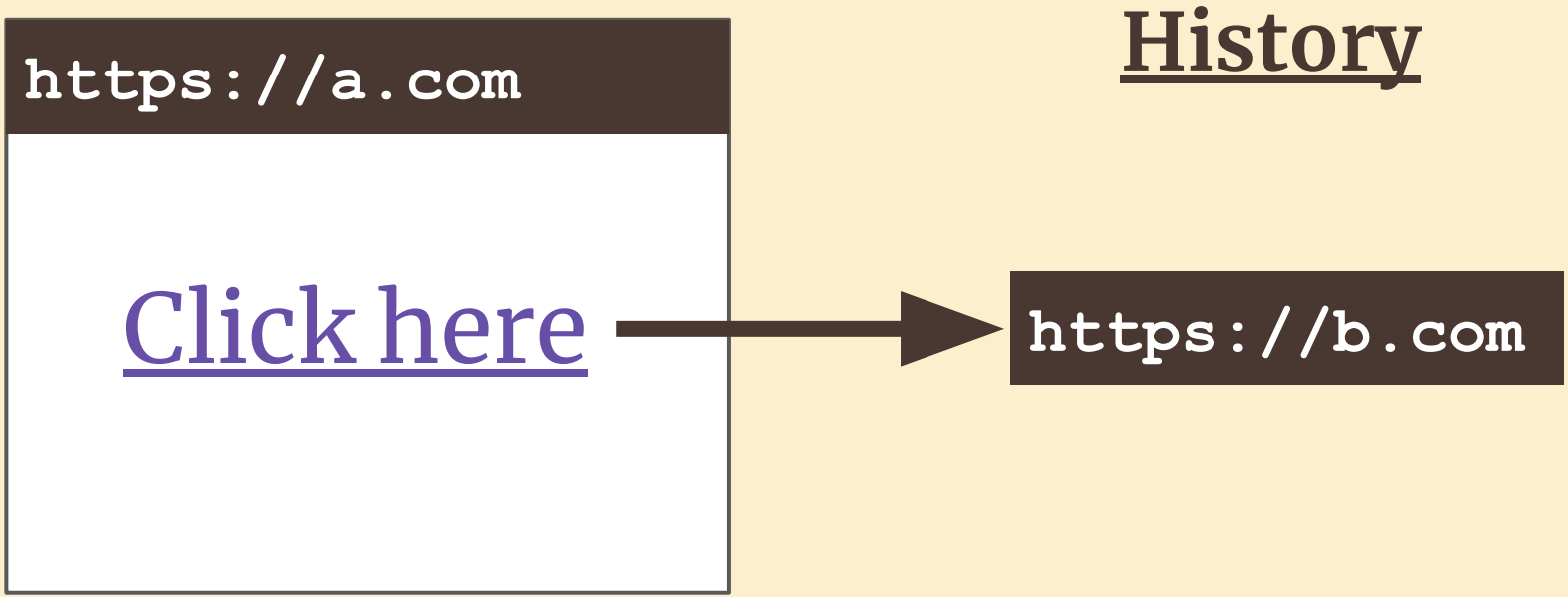
"Tealium's *patent-pending technology* lets you see the view-through traffic to your site by those who've been exposed to your press, or blog or video coverage."

Defense: “referrer-origin labels”

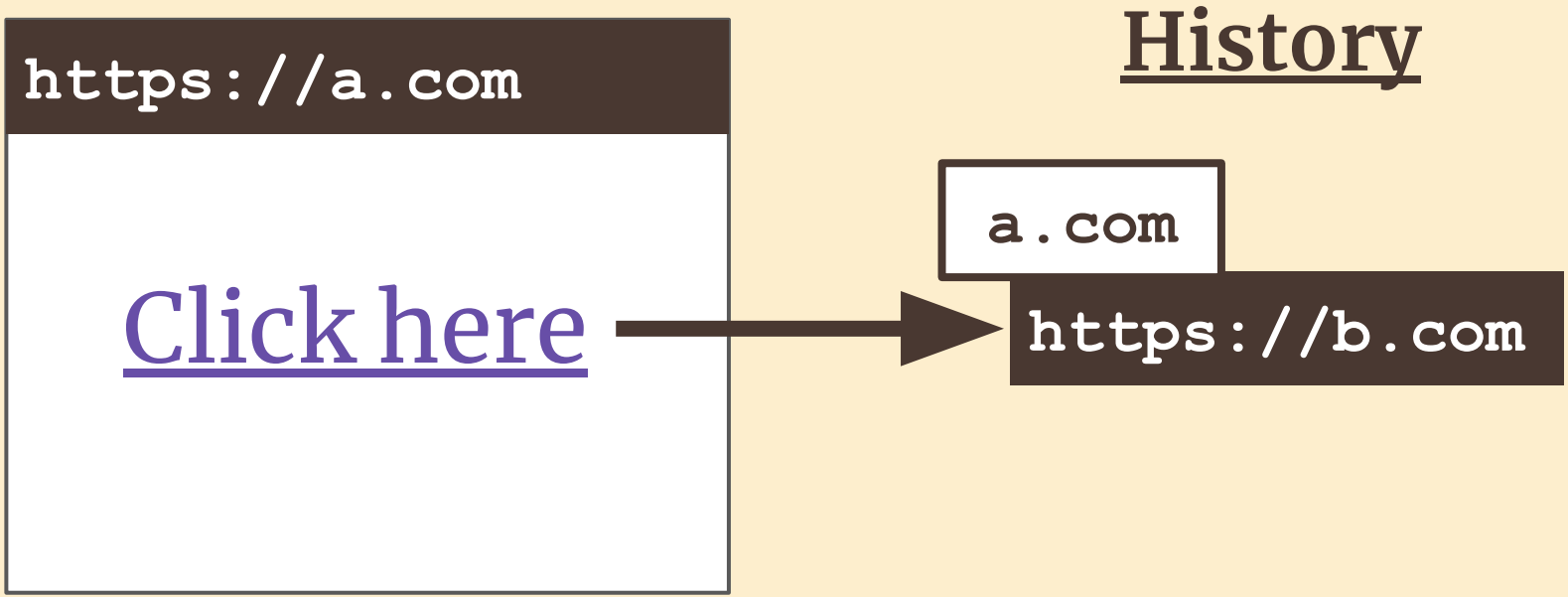
Defense: "referrer-origin labels"



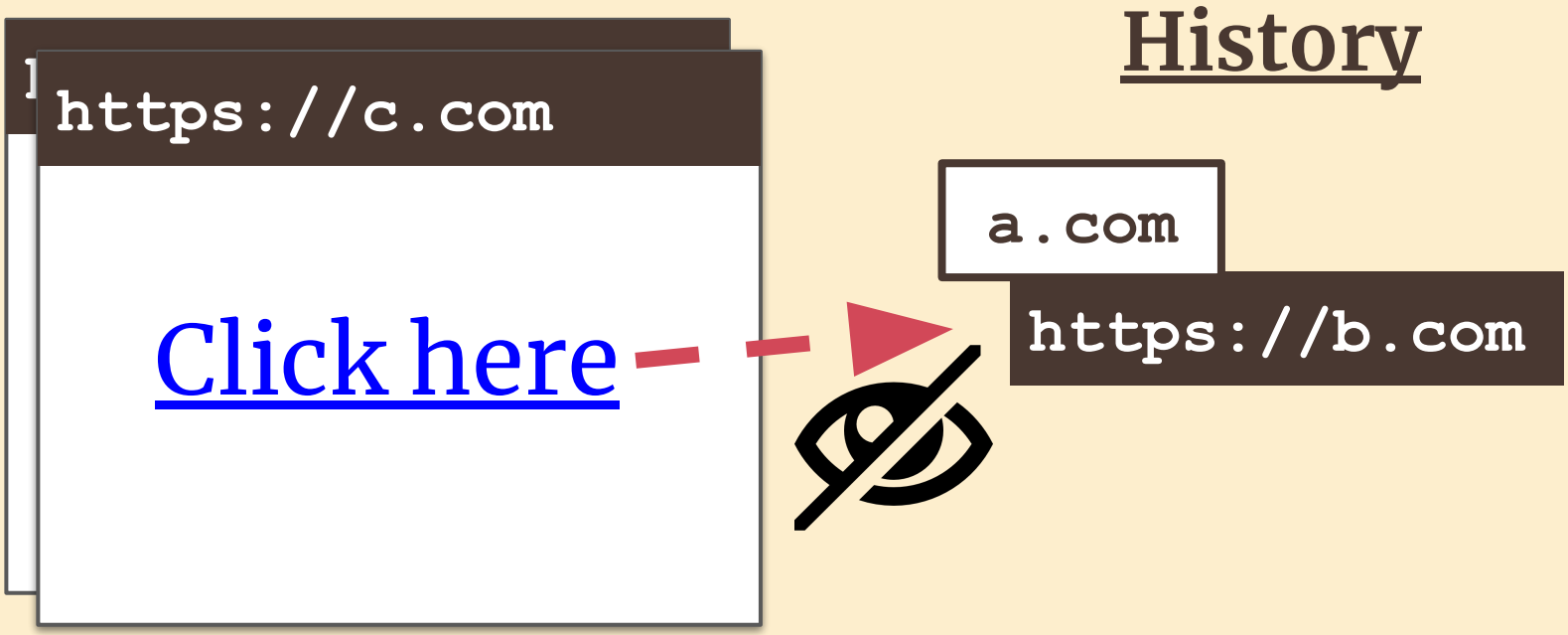
Defense: "referrer-origin labels"



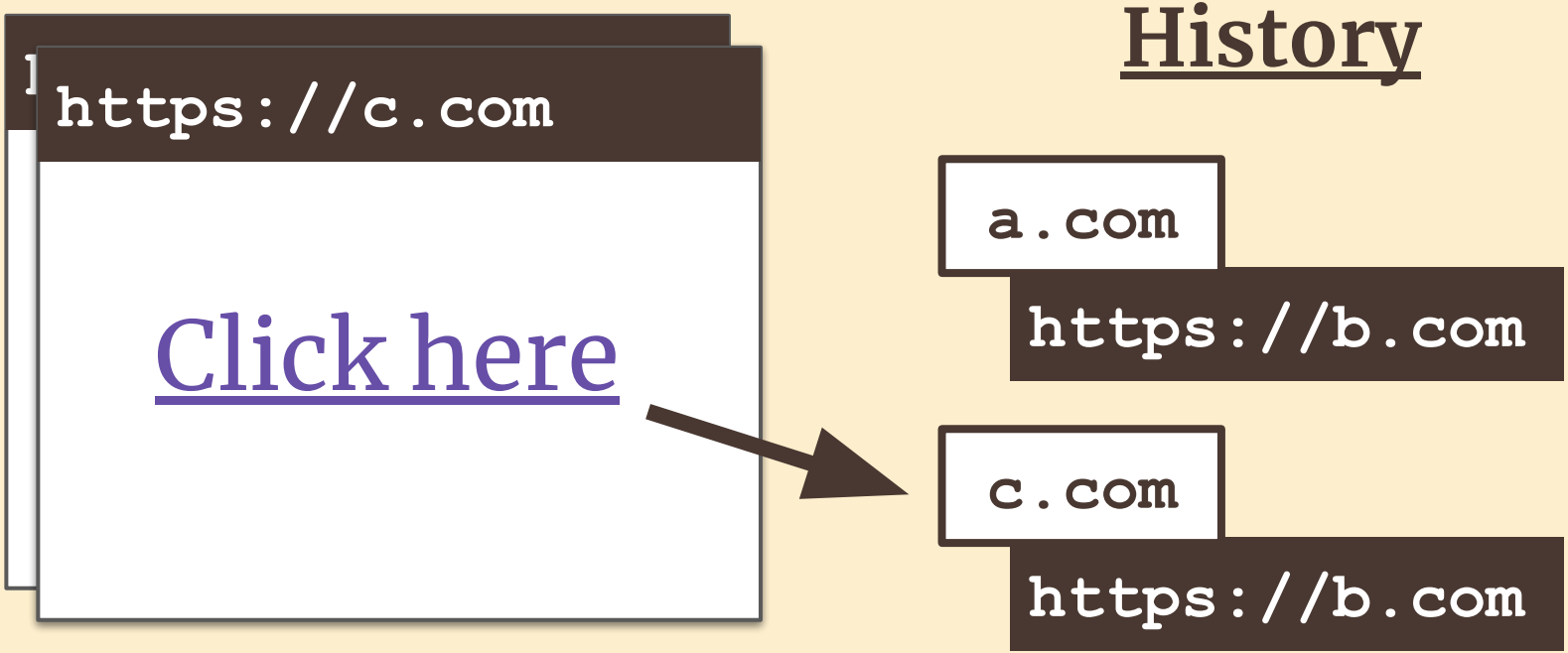
Defense: "referrer-origin labels"



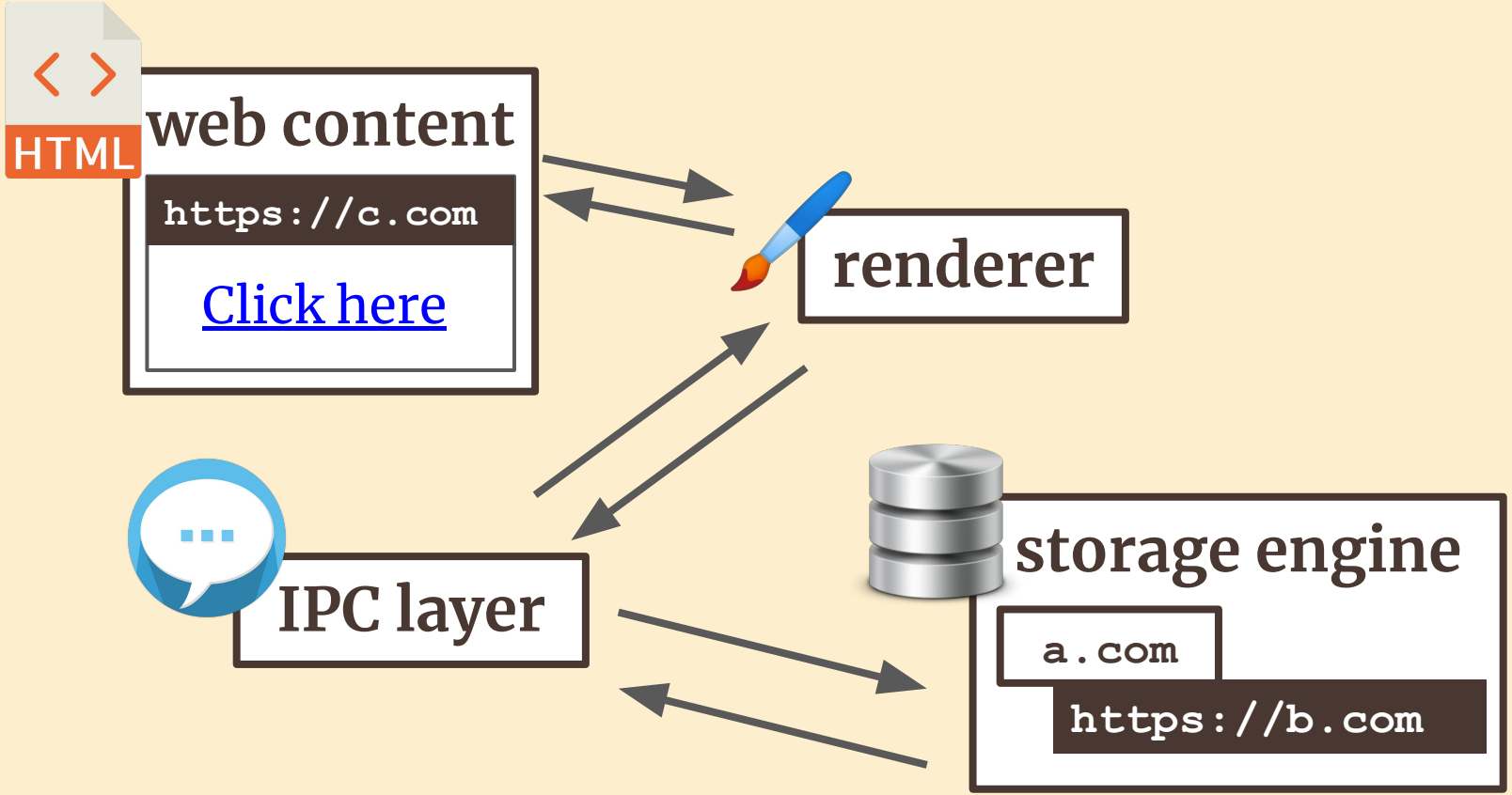
Defense: “referrer-origin labels”



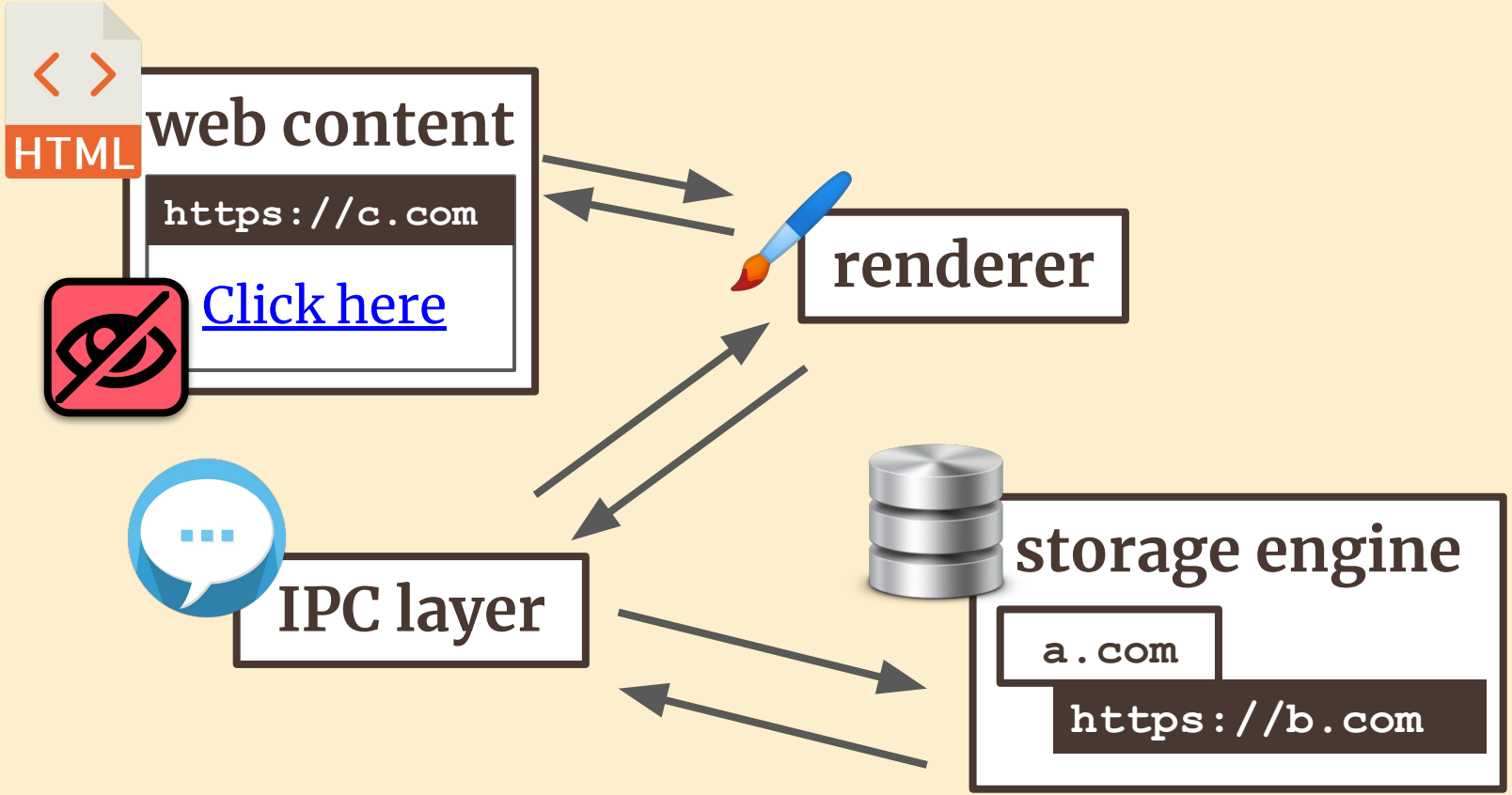
Defense: "referrer-origin labels"



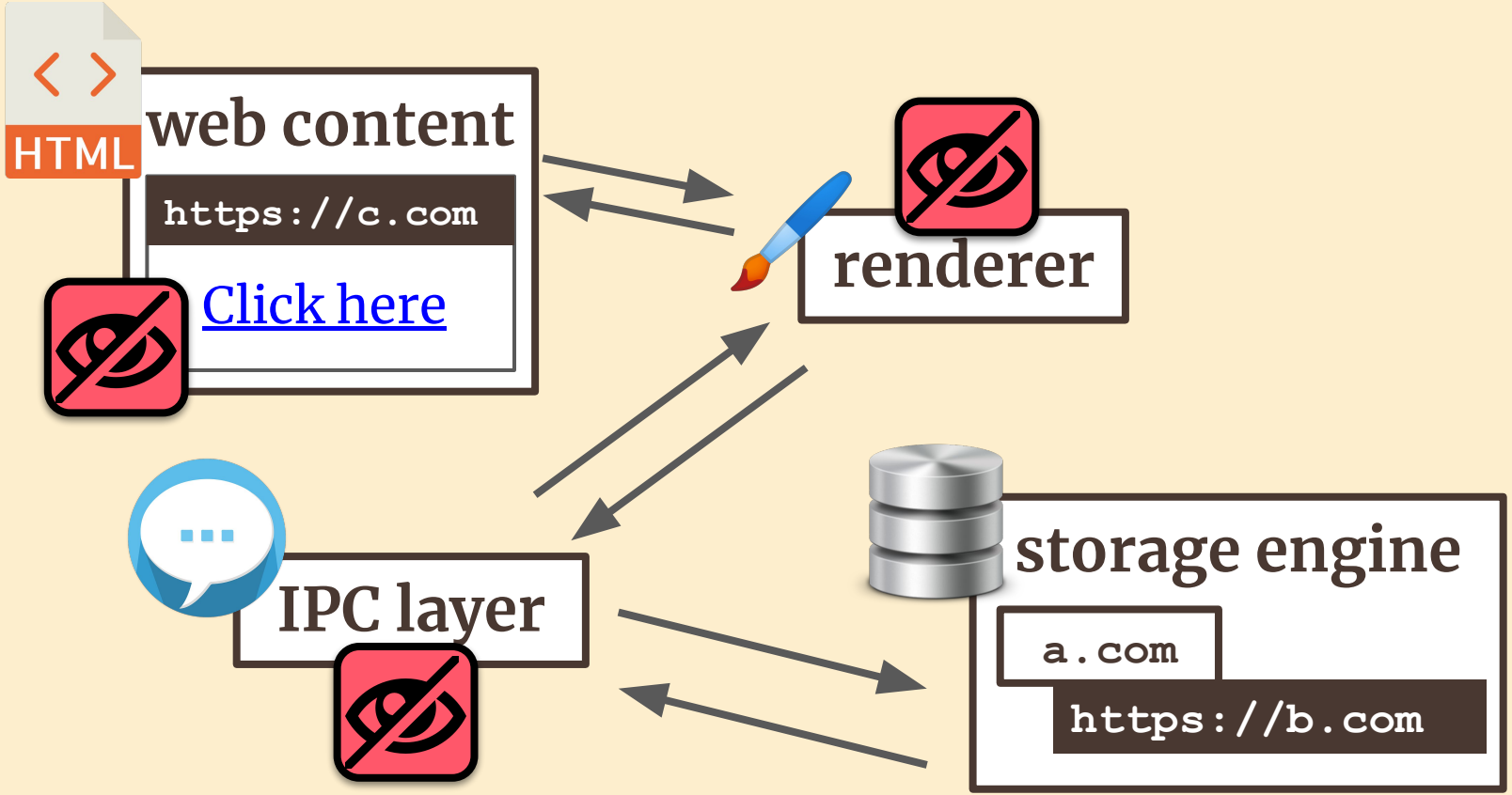
“Is https://b.com visited?”



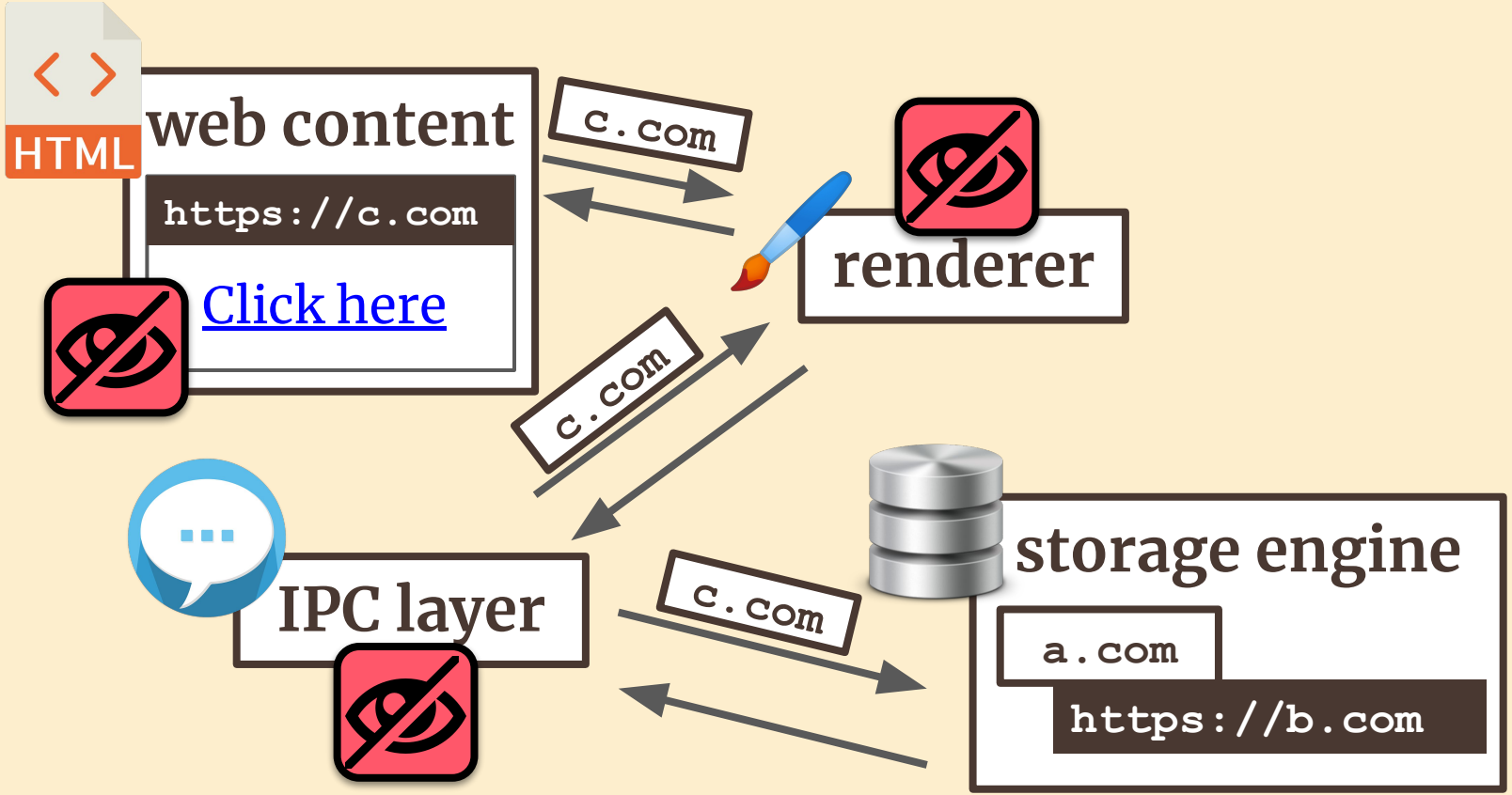
“Is https://b.com visited?”



“Is https://b.com visited?”



“Is https://b.com visited?”



Defense: “referrer-origin labels”

1) Applies to:
   history data + cache data

2) Replaces prior mitigations

AmeliaBR commented

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”