Cache Me If You Can

Exposing your application using caching features

Philippe Arteau
Security Researcher for GoSecure

ConFoo.CA
DEVELOPER CONFERENCE

GoSecure
POWERED BY COUNTERTACK

$ whoami

- Philippe Arteau
- Security Researcher at GoSecure
- Open-source developer
  - Security Code Scan (Roslyn – Static Analysis for .NET)
  - Find Security Bugs (SpotBugs - Static Analysis for Java)
  - Burp and ZAP Plugins (Retire.js, CSP Auditor)

- Volunteer for the nsec conference and former trainer

Agenda

- Introduction

- Web Cache Deception
- ESI injection
- Cache poisoning

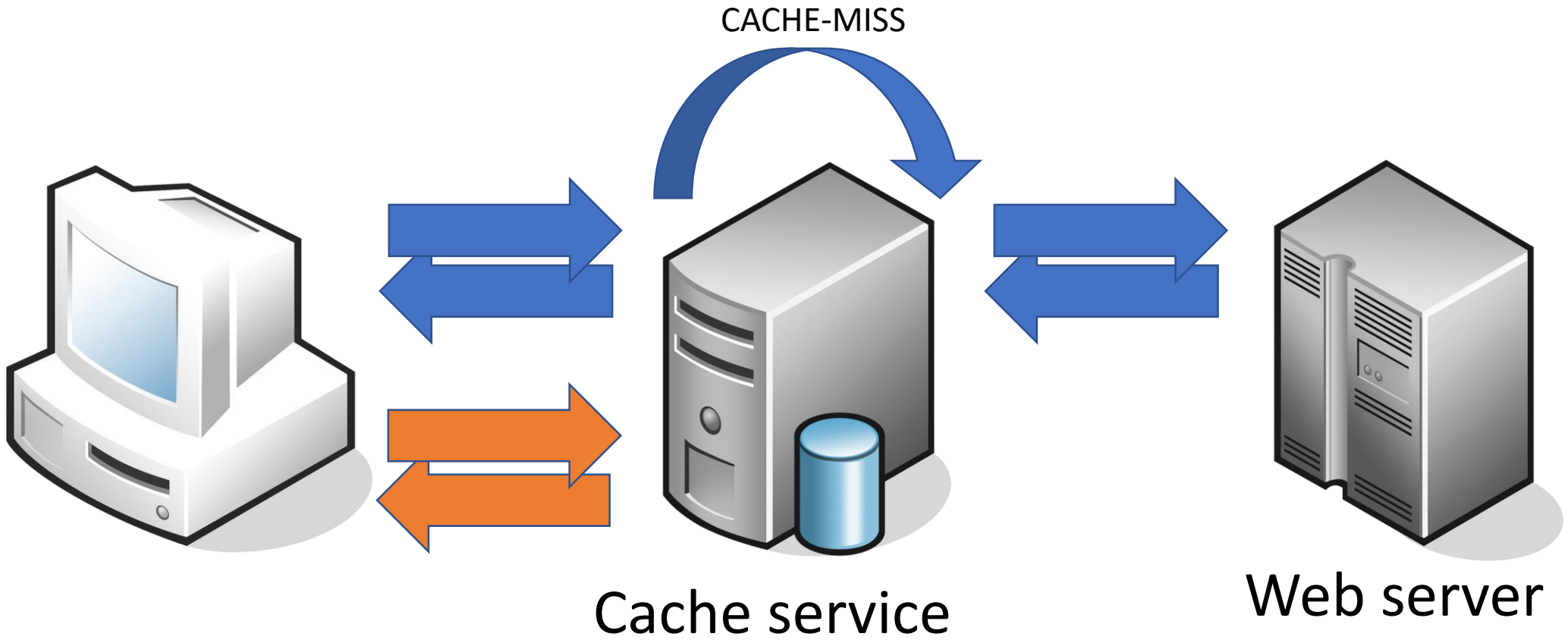
- Mitigations

Cache server
/kaSH ˈsərvər/

A **cache server** is a dedicated network **server** or service acting as a **server** that saves Web pages or other Internet content locally. By placing previously requested information in temporary storage, or **cache**, a **cache server** both speeds up access to data and reduces demand on an enterprise's bandwidth.

Overview

The caching services covered will be mainly **reverse proxy**.



New Cache Vulnerabilities Discovered

Caching vulnerabilities taken from "Top10 Web Hacking Techniques"

From 2017

- **Web Cache Deception**
- Cloudbleed
- Binary Webshell Through OPcache in PHP 7

From 2018

- **Web Cache Poisoning**
- **Edge Side Include Injection**
- *Path Normalization Parser Logic Flaw (can apply to cache/proxy server)*

https://portswigger.net/blog/top-10-web-hacking-techniques-of-2018
https://portswigger.net/blog/top-10-web-hacking-techniques-of-2017

Demonstration

Simple caching mechanism

Web Cache Deception

Web Cache Deception

Discovered by Omer Gil in 2017

This allow an attacker to **force to a <u>specific user</u> cache private information** and view the cache later.

It takes of advantage of **different path interpretation**.

Sample configuration

The demonstration will be using the following configuration

```
location ~* .+\.(ico|jpg|gif|jpeg|css|js|flv|png|swf)$ {

        proxy_pass http://webapp;
        rewrite ^/(.*)$ /$1 break;

        expires 10m;
        proxy_cache cache;
        proxy_cache_key $host$uri#is_args$args;
        proxy_cache_valid 200 304 12h;
        proxy_cache_valid 302 301 12h;
        proxy_cache_valid any 1m;
}
```

https://www.nginx.com/blog/nginx-caching-guide/
https://serversforhackers.com/c/nginx-caching

To cache or not to cache ?

Sensible information is available at a given path:

```
location ~* .+\.(ico|jpg|gif|jpeg|css|js|flv|png|swf)$ { […] }
```

http://website.com/transactions**.php**	NOT CACHED
http://website.com/css/jquery**.css**	CACHED
http://website.com/images/favicon**.ico**	CACHED
http://website.com/files/song1**.mp3**	NOT CACHED

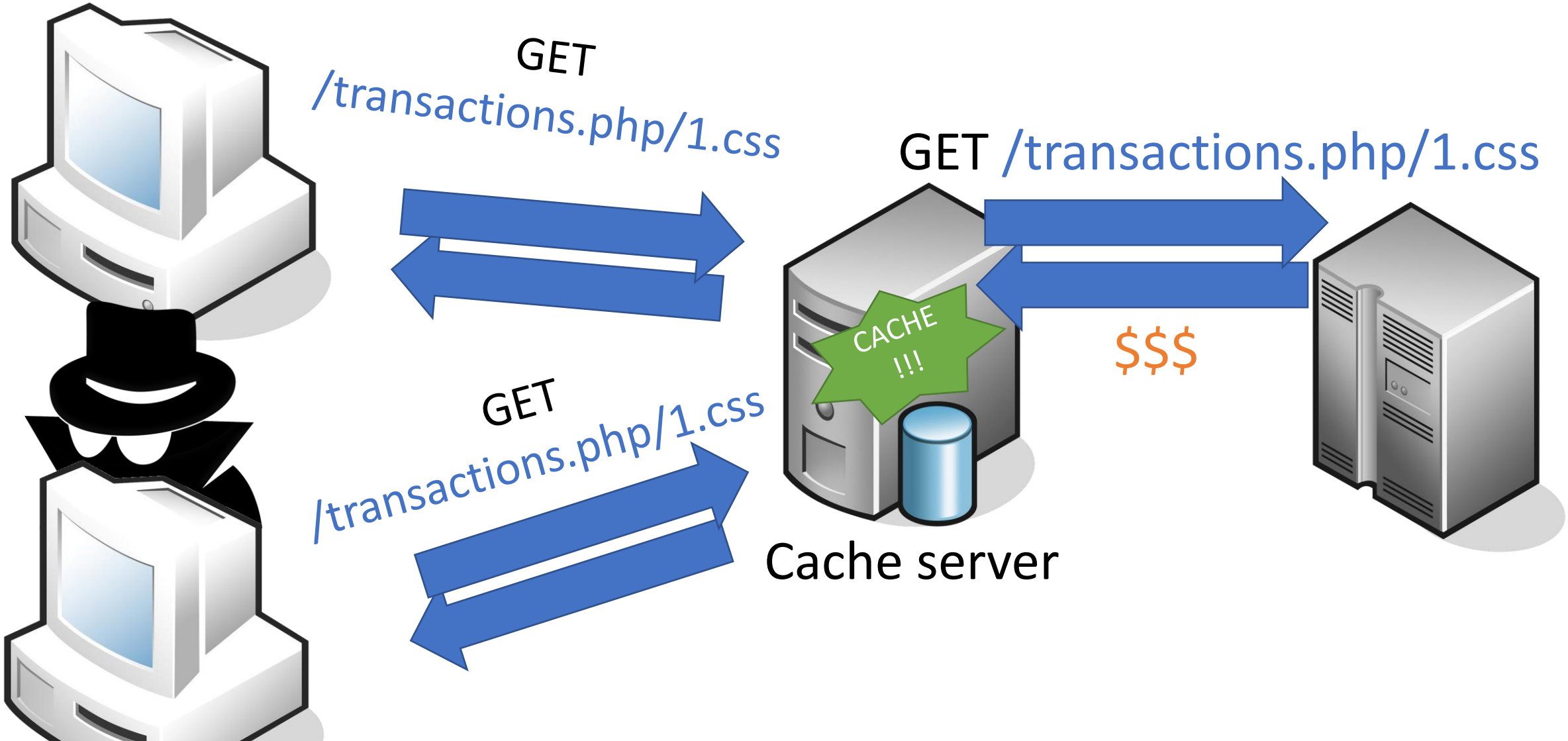
To cache or not to cache ? (part 2)

Sensible information is available at a given path:

```
location ~* .+\.(ico|jpg|gif|jpeg|css|js|flv|png|swf)$ { […] }
```

http://website.com/transactions.php	NOT CACHED
http://website.com/transactions.css	CACHED
http://website.com/transactions.php.css	CACHED
http://website.com/test.css/%2E%2E/transactions.php	NOT CACHED
http://website.com/transactions.php/test.css	CACHED

Request flow



Impacts

- Access to sensitive information
  - Private documents, images, etc.
  - Transactions
  - API Token
  - Security questions
- Also leakage of CSRF Token
  - Action can be performed on behalf of users

Is it language specific?

PHP : /page.php => /page.php**/test.css**

Python Django : /page => /page**/test.css**

ASP.net Web Forms : /page.aspx => /page.aspx**/test.css**

JSP pages : /page.jsp => /page.jsp**;test.css**

Demonstration

Web Cache Deception

NGINX + PHP

Mitigations

Cache-Control

Return `Cache-Control: private`

on pages that return sensible information

Pragma

Return `Pragma: no-cache`

Expires

Return `Expires: [past date]`

Content-Type Validation

The cache service can validate that the Content-Type returned match the extension.

- *.css → `text/css`
- *.js → `text/javascript`
- …

https://blog.cloudflare.com/understanding-our-cache-and-the-web-cache-deception-attack/

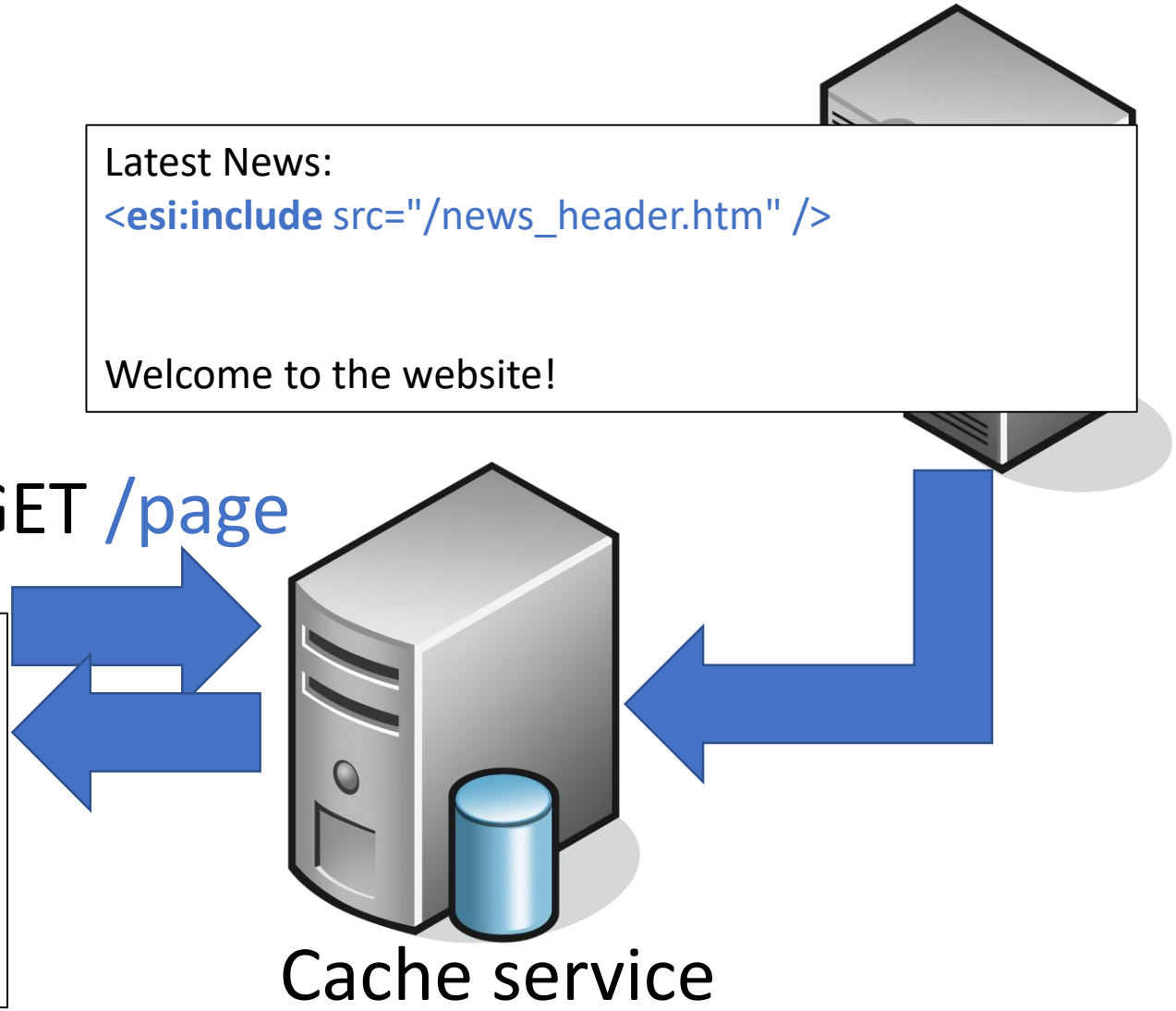
ESI Injection

ESI Injection

Discovered by Laurent Desaulniers and Louis Dion-Marcil in 2018

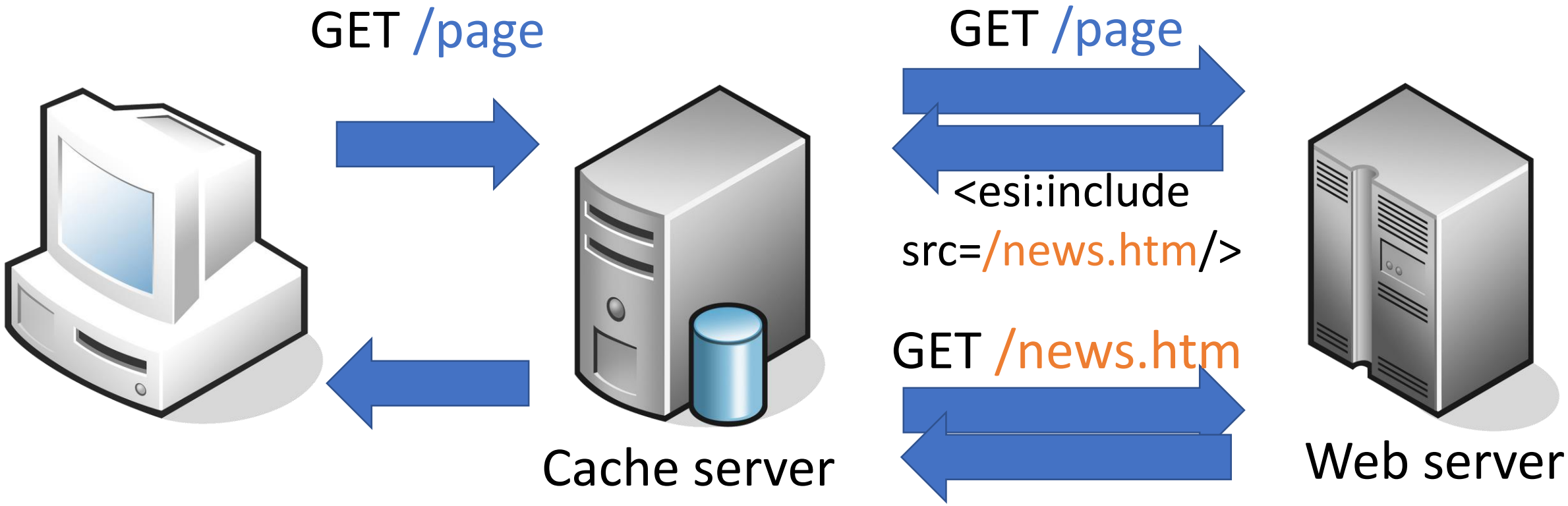
- Malicious use of **XML cache directives** place in **HTTP response**

ESI Code Sample



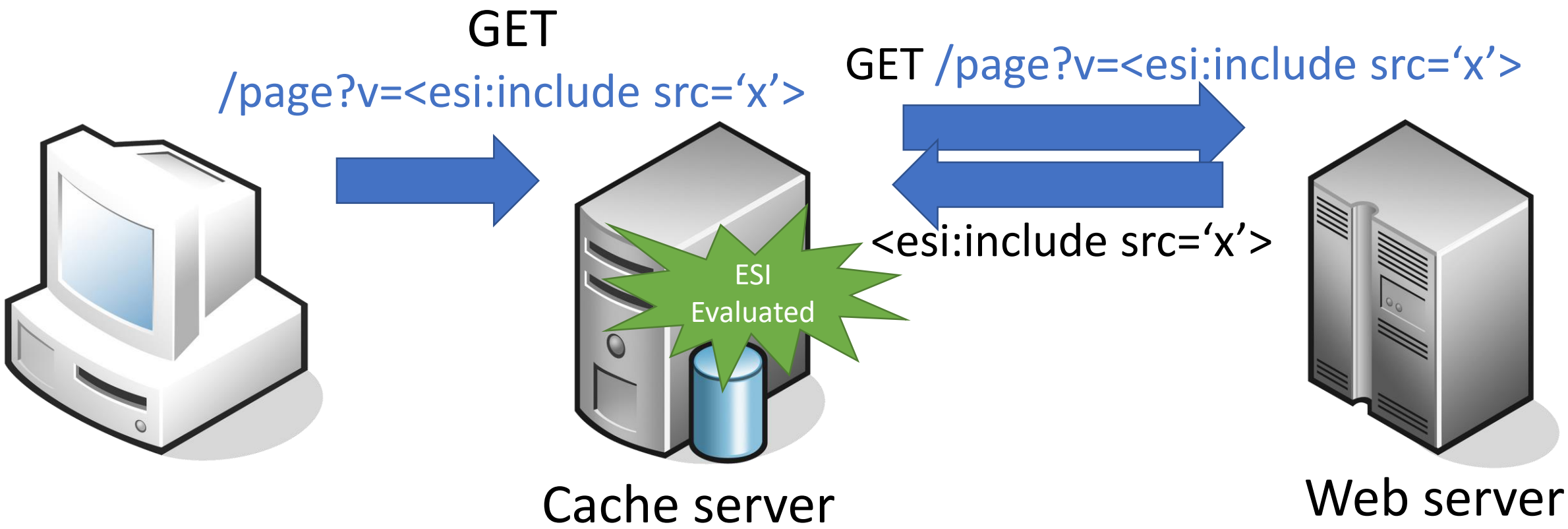
ESI include tags

Expected flow



ESI include tags

Malicious flow



Potential ESI instructions

```
<esi:include src="/page.htm"/>

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

<esi:inline name="/news_page.html" fetchable="yes">
<script>prompt('Malicious script')</script>
</esi:inline>

<!esi-- -->
```

Demonstration

ESI:include leaking session cookie

Mitigations

Mitigations (ESI)

- Escape properly value return in HTML context in your web pages.
  - Framework/template automatic escaping should cover this case
- Verify that ESI is not enable on your cache service
- Avoid returning Surrogate-Control: content="ESI/1.0" on page that contains user content.

Cache Poisoning

Cache Poisoning

- Discovered by James Kettles from PortSwigger in 2018

- Use Caching server in order to propagate
  - Cross-Site Scripting (XSS)
  - Open-redirect
  - Other content poisoning (file download, resources, …)

It require vulnerabilities **based on reflected HTTP header**.

Can we exploit this?

Request

Raw | Params | Headers | Hex

```
GET /index.php HTTP/1.1
Host: 123456789
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
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: pref_lang=fr; PHPSESSID=sahbsvf2gfsr6h5jh65i8k42vp
Connection: close
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache
```

? < + > Type a search term

Response

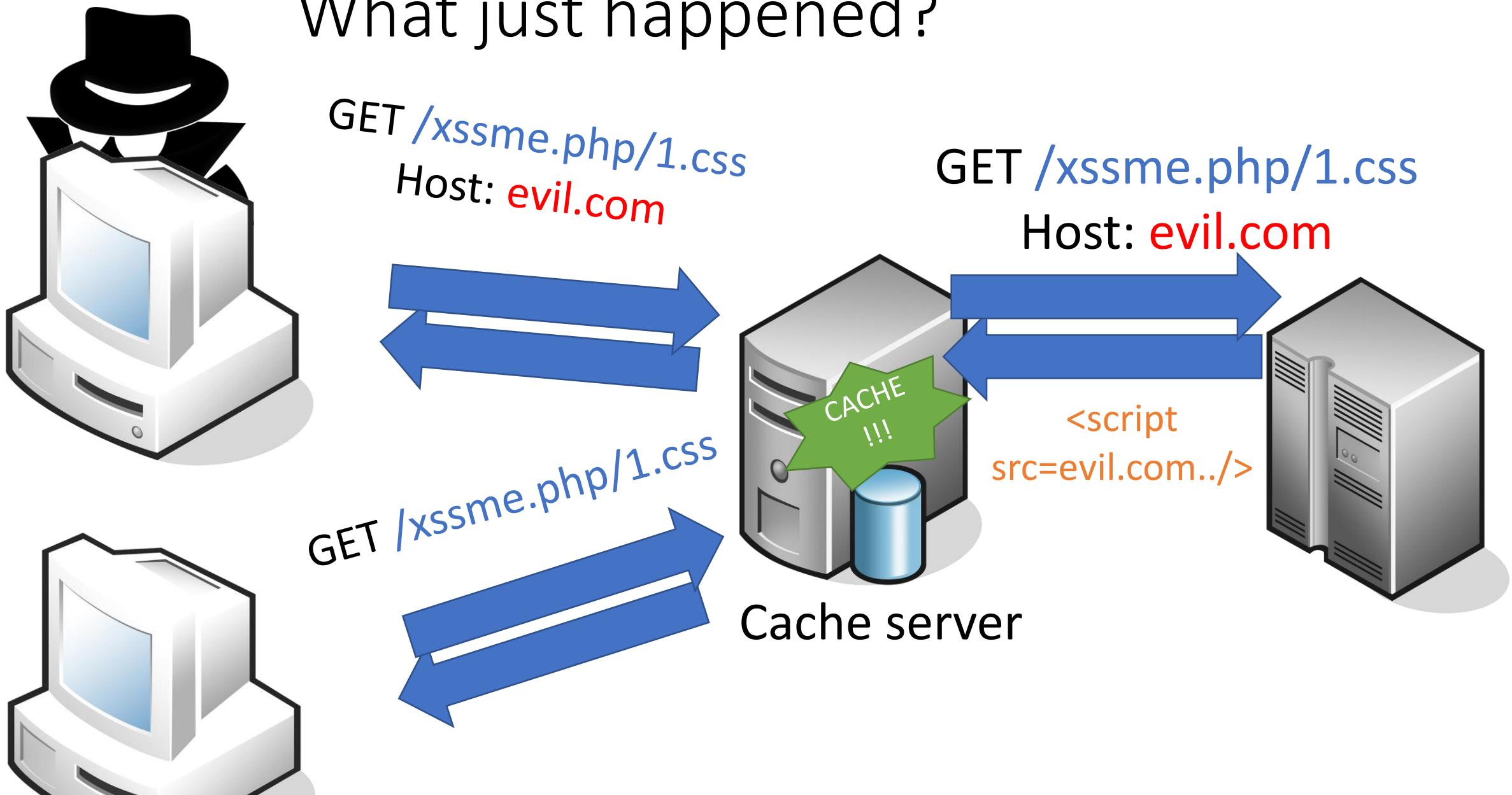
Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 200 OK
Server: nginx/1.15.5
Date: Thu, 14 Mar 2019 15:17:05 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Content-Length: 299

<html>
<head>
    <title>Insecure Bank</title>

    <link href="main.css" rel="stylesheet"/>

    <script src="http://123456789/script/jquery.js"></script>
</head>
<body>
```

Demonstration

Cache poisoning

What just happened?



Mitigations

Mitigations (Cache Poisoning)

- Same as Cache Deception

- Also, considered the risk when “unexploitable” XSS are found.

Conclusion

In summary

- The **cache services** deployed can affect your application
- **Review the configuration** of those services
  - Features enable
  - Regex for the caching resources to include
- Make sure cache headers are returned (`Cache-Control`, `Pragma`, ..) on authenticated page

Questions ?

Contact
parteau@gosecure.ca
https://gosecure.net/
@h3xStream @GoSecure_Inc

References

Web Cache Deception

- http://omergil.blogspot.com/2017/02/web-cache-deception-attack.html
- https://www.blackhat.com/docs/us-17/wednesday/us-17-Gil-Web-Cache-Deception-Attack-wp.pdf

ESI Injection

- https://gosecure.net/2018/04/03/beyond-xss-edge-side-include-injection
- https://www.gosecure.net/blog/2019/05/02/esi-injection-part-2-abusing-specific-implementations

Cache poisonning

- https://portswigger.net/blog/practical-web-cache-poisoning
- https://portswigger.net/blog/bypassing-web-cache-poisoning-countermeasures

- Credit to Mike McDonald / Ember Studio for the Computers Icons