<Modern XSS/>

The modern protections (and bypasses)

Philippe Arteau, Security Researcher

ConFoo.CA

GoSecure

Who am I ?

- Philippe Arteau
- Security Researcher at GoSecure
- Open-source developer

Security Guard	(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

GoSecure

Agenda

- Motivation and Overview
- Server Side Controls
  - Template Engine
  - ASP.net Request Validator
  - Web Application Firewall
- Client Side Controls
  - Chrome XSS Auditor
  - IE/Edge XSS Filter
- Content Security Policy
- Conclusion

Motivation and Overview

Motivation

Why learn about XSS protections even if they come by default?

- Developers can be more efficient at:
  - **Troubleshooting** client-side effect
  - **Working with** - not against - the protections in place
- **Avoid disabling protection** on the first side effect
- Know about **theirs limitations**

IE/Edge XSS Filter

Chrome XSS Auditor

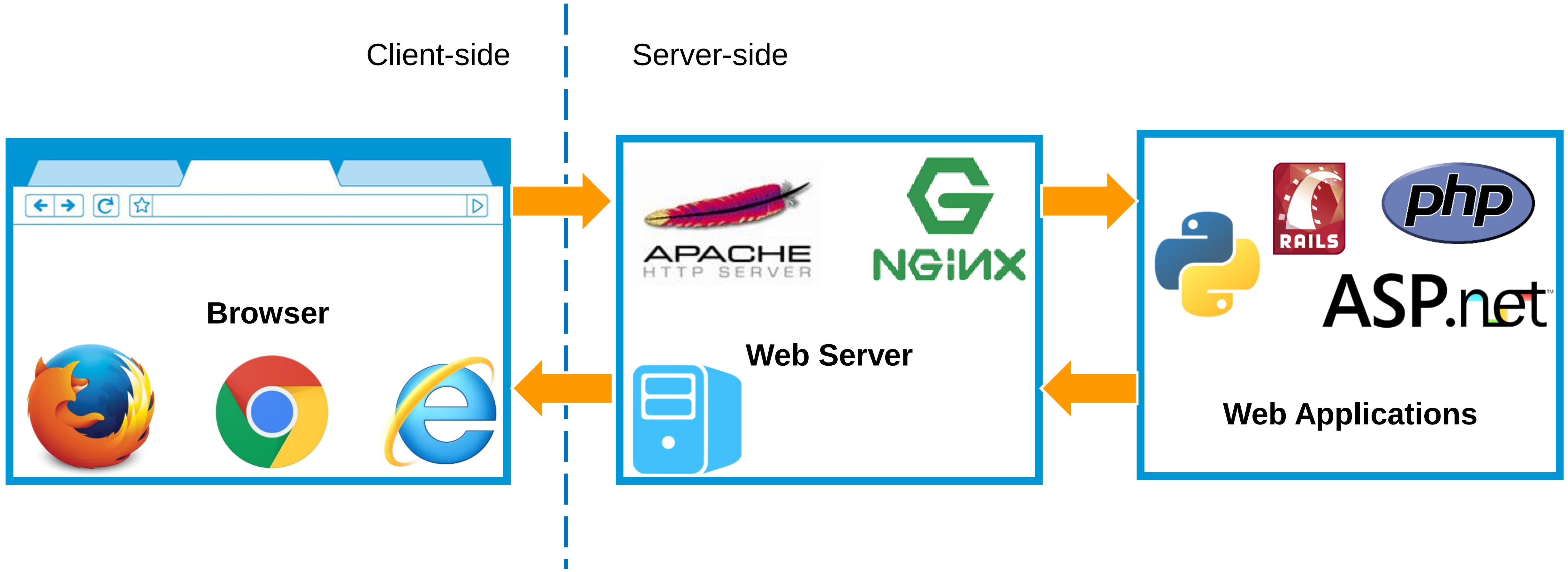
Template Engine Escaping

ASP.net Request Validator

Firefox XSS ... not yet

Which attack vectors are still **relevant** for XSS in **modern** web applications?

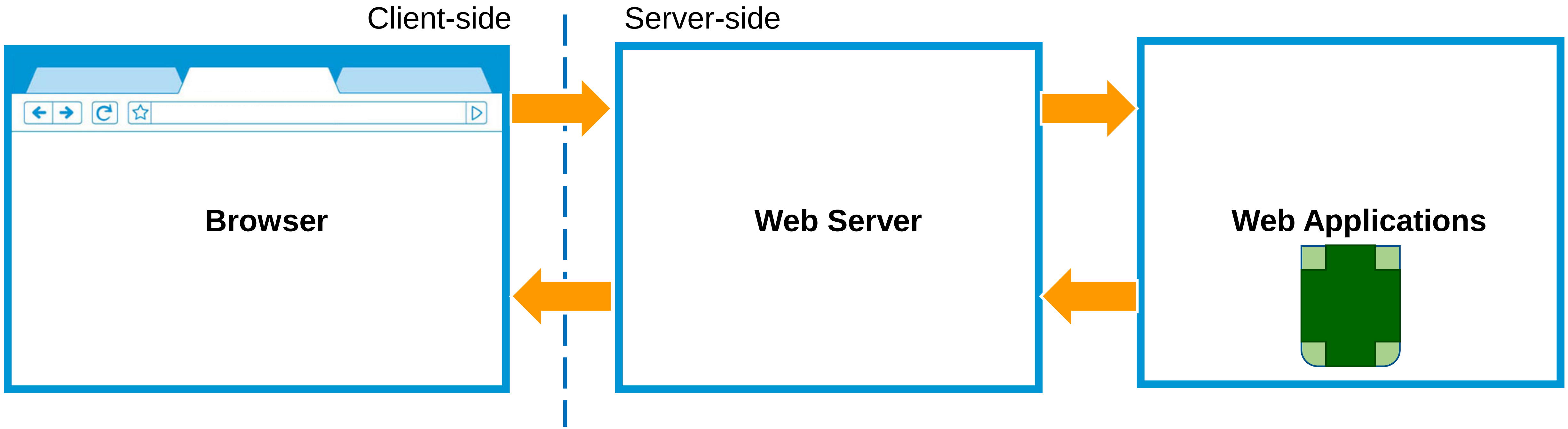
The Big Picture



Every protection will be effective... but most of them have limitations.

Server-Side Controls

Template Engine



Most template engine have **HTML encoding** by **default**

Edge cases:

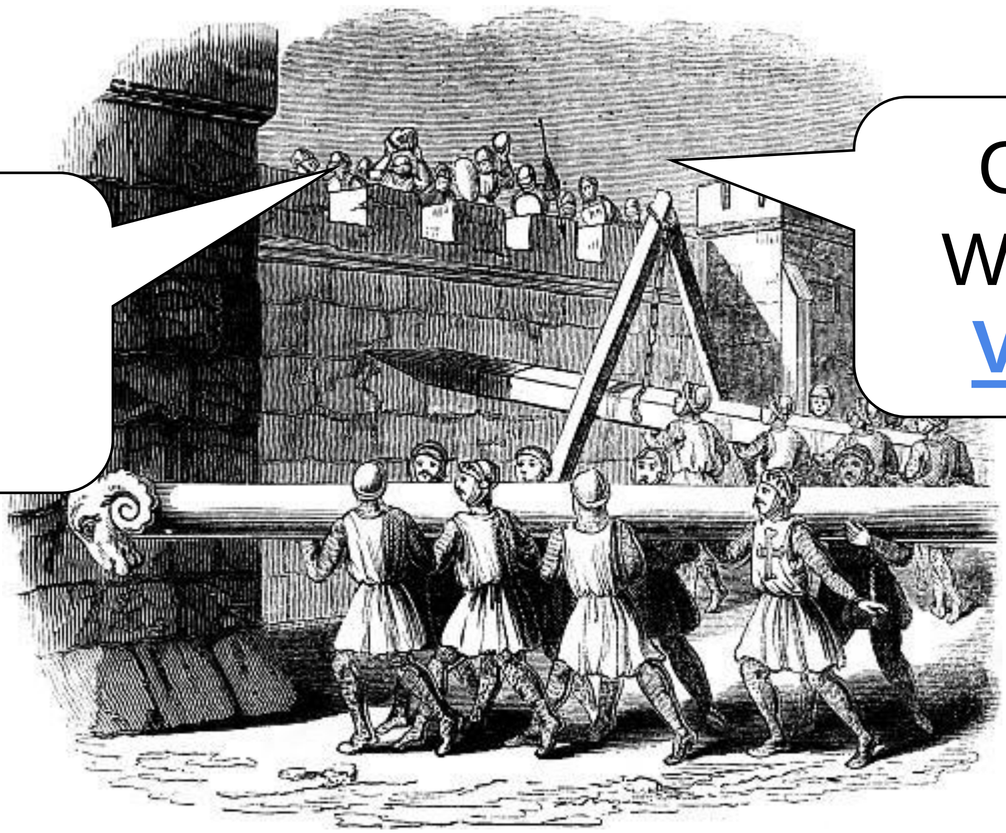
- XSS Contexts
- Unquote attributes

Demonstration: Template engine / XSS Contexts

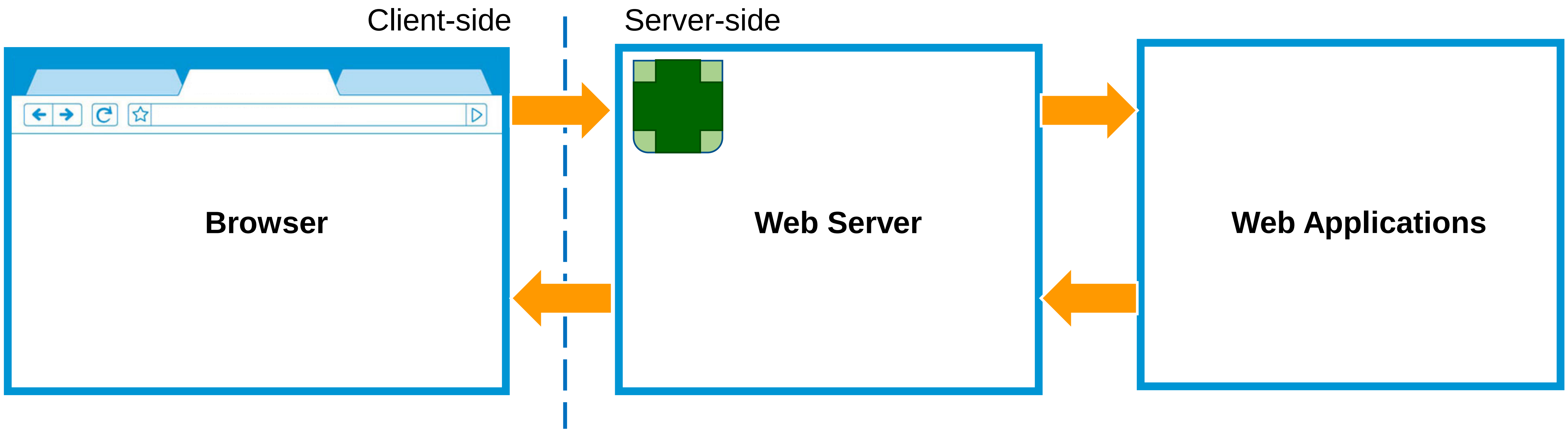
Bypass Included

GOSECURE

Web Application Firewall (WAF)

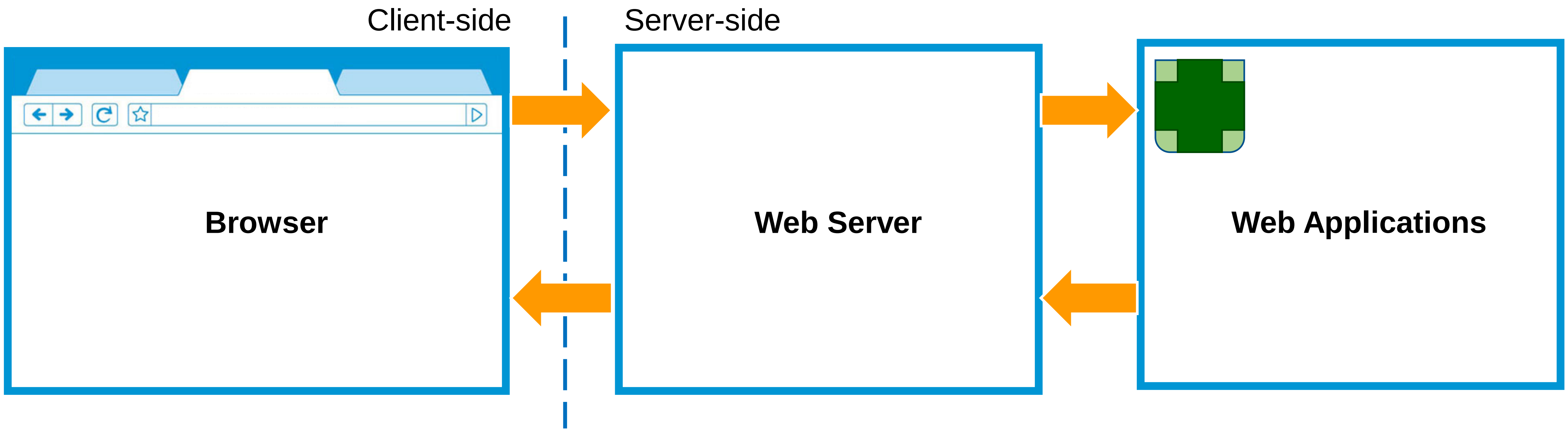


Web Application Firewall (WAF)



- Decoupled from the application (context is often missing)
    - Hard to understand request format such as JSON and XML.
- Regex patterns that take too long to process can be skipped /!\
- Transformation can lead to bypass

Request Validator (ASP.net)



- First, don’t disable it globally
- Transformation can lead to Request Validator bypass
- Request Validator focuses on HTML context (not Javascript, attribute or CSS)

Request Validator (ASP.net)

- Request Validator is a filter applied before the controller handles the parameters
- If a controller is transforming the value, the value may not be safe
  - Base64 decoding
  - URL decoding
  - SQL Server ascii column

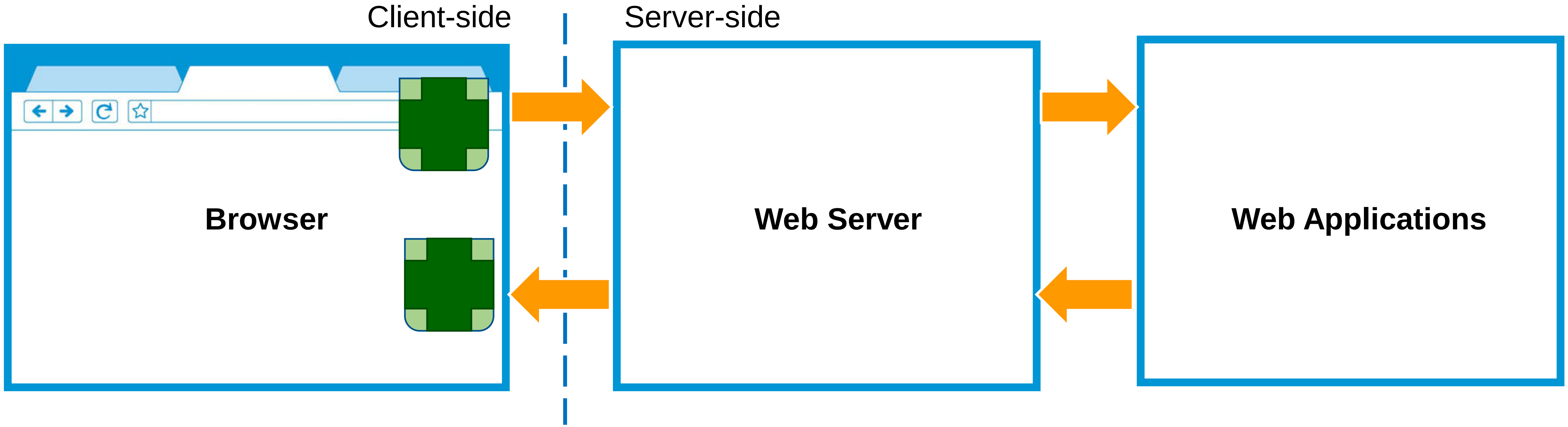
Character	Character After storage
U+FF1C (%EF%BC%9C)	U+003C (%3C) "<"
U+FF1E (%EF%BC%9E)	U+003E (%3E) ">"

Ref: http://gosecure.net/2016/03/22/xss-for-asp-net-developers/

Client-Side Controls

Parameters Inspection...

Browser Filters



- Does not apply to persistent XSS
- Transformations can often lead to filter bypasses
- Focus on the HTML and attribute contexts

Browser Filters: Adoption

- Mozilla Firefox
  - Inexistent
- Internet Explorer 8+
  - Active by default
- Google Chrome
  - Active by default
- Additional configurations (X-XSS-Protection: 1)
  - Mode=block: Stops the page loading if a malicious pattern is detected.
  - Report=URL: (Chrome and Safari only) The browser will post the blocked parameters to the URL

Chrome XSS Auditor (XSS Filter)

Chrome will **not execute scripts** that appear to have been reflected.

Request:

```
?input=<h1>Hello <script>alert(1)</script></h1>
```

Response: (highlighted value is not executed but remains in the DOM)

```
<h1>Hello <script>alert(1)</script></h1>
```

Chrome XSS Auditor (XSS Filter)

Chrome trusts resources that are hosted on the **same origin** (domain).

- <script src="//**xss.lol/**malicious.js"></script>
- <script src="/jsonp?**callback=**test"></script> (Exception)

- <script src="/api/users/files/23840238492.txt"></script>

Demonstration: Chrome XSS Filter

GoSecure

IE/Edge XSS Filter: How Does it Work?

IE and Edge will **modify potentially malicious values** that appear to have been reflected.

Request:

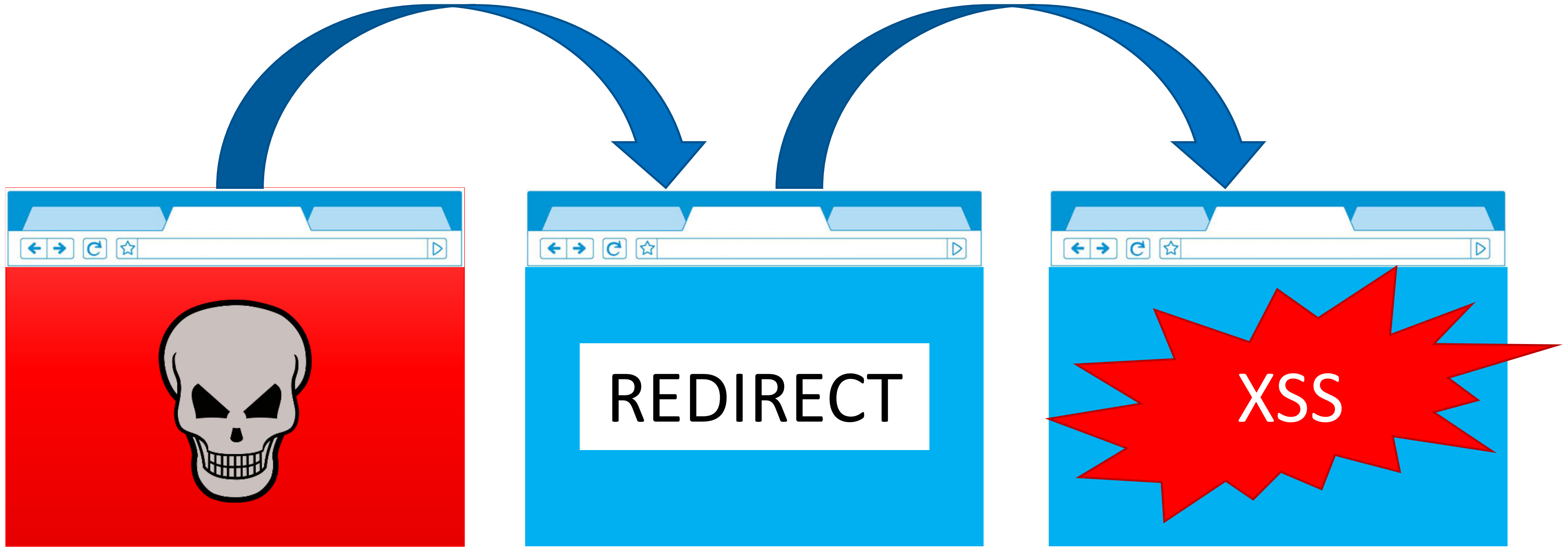
```
?input=test" autofocus="" onfocus="alert(1)
```

Response:

```
< [...] value="test" % autofocus="" #nfocus="alert#1#" >
```

IE/Edge XSS Filter: Potential bypasses

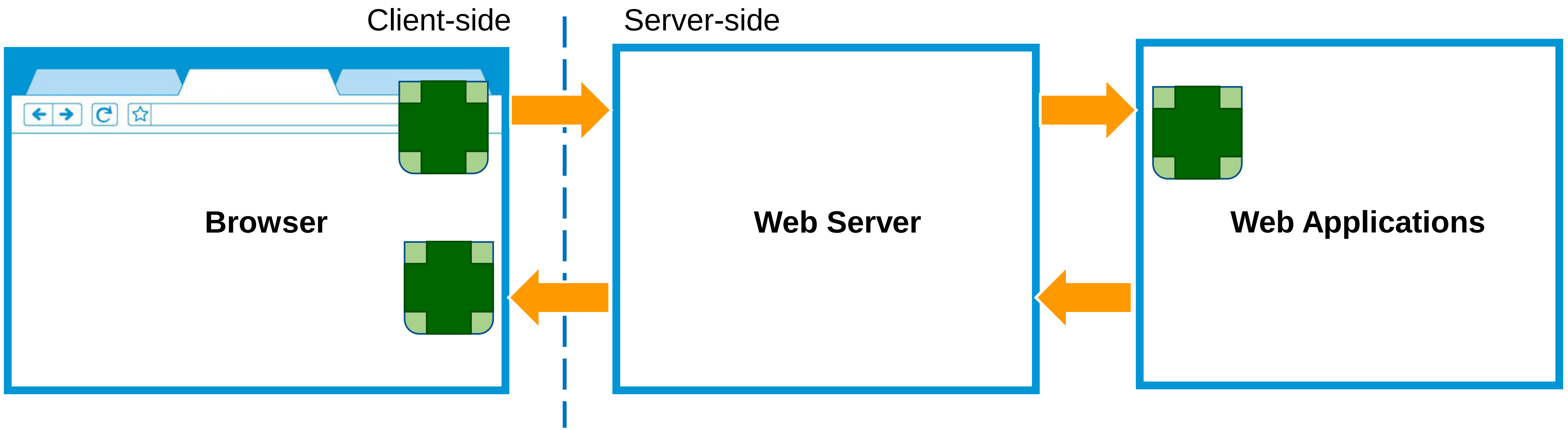
- If the **referrer is the same origin** has the current page, it is consider safe.



Demonstration: IE/Edge XSS Filter

Content Security Policy

Content Security Policy



- Supported by all modern browsers
- Small adoption among web frameworks
- Hard to configure manually
- Mode “Report-Only” available

Common Misconfigurations

- **‘unsafe-inline’ misconfiguration**
- ‘unsafe-eval’ may lead to DOM XSS
- Use of wildcards *
- **Allowing CDN servers, googleapi.com, etc.**
- **Allow file upload on the same domain**
- Use of deprecated header
- Unexpected inheritance from “default-src:”

Ref: http://gosecure.net/2016/06/28/auditing-csp-headers-with-burp-and-zap/

Content Se

HTTP/1.1 200 OK

Server: nginx

Date: Wed, 17 Fe

Content-Type: te

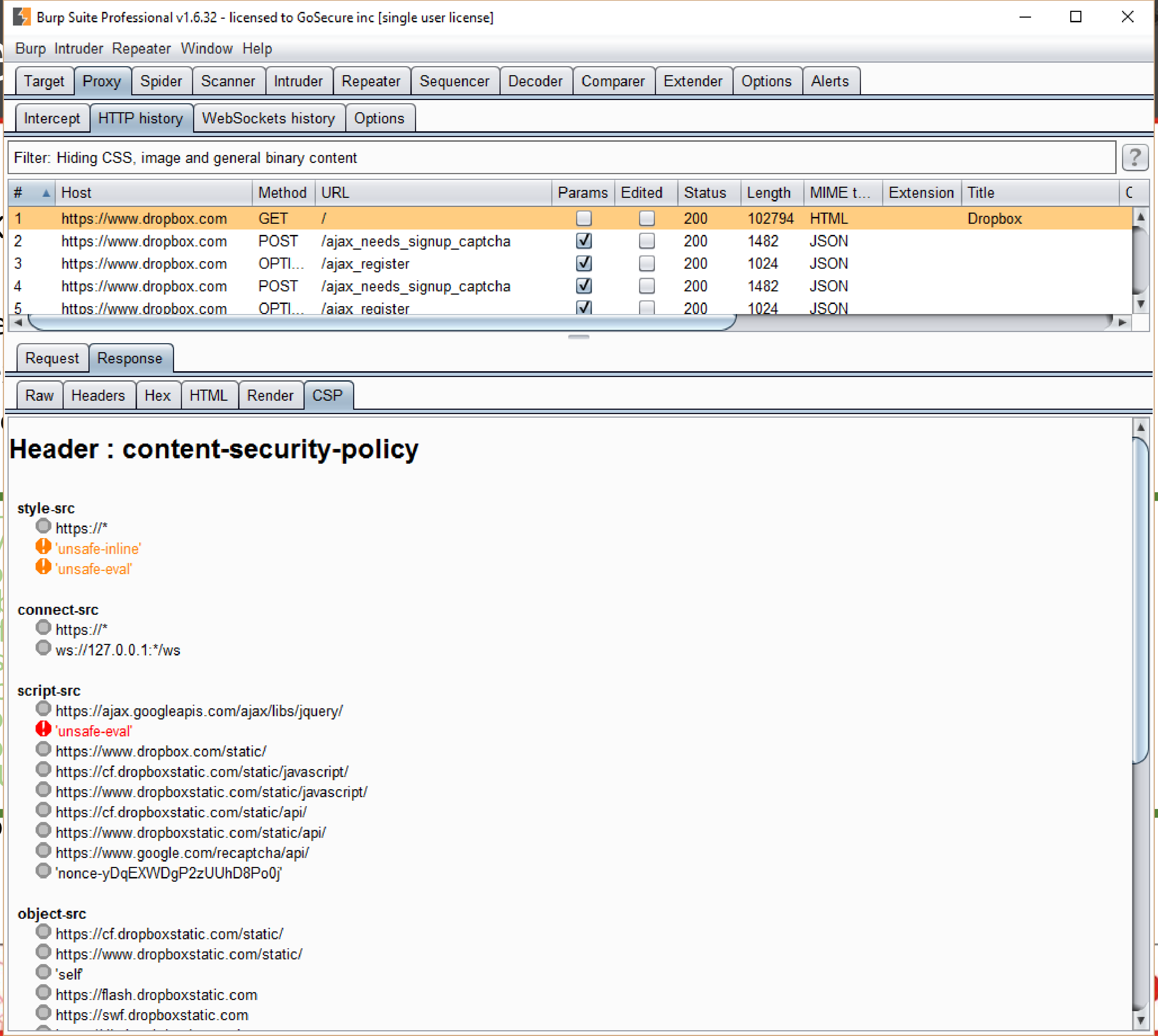
Connection: clos

x-xss-protection:

content-security-
media-src https:/
https://www.dro
https://swf.dropb
https://* data: ;
src https://* 'uns
'unsafe-eval' http
https://www.dro
https://www.dro
yDqEXWDgP2zUU

x-content-type-o

[…]



Burp Suite Professional v1.6.32 - licensed to GoSecure inc [single user license]

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME t...	Extension	Title
1	https://www.dropbox.com	GET	/	☐	☐	200	102794	HTML		Dropbox
2	https://www.dropbox.com	POST	/ajax_needs_signup_captcha	✓	☐	200	1482	JSON		
3	https://www.dropbox.com	OPTI...	/ajax_register	✓	☐	200	1024	JSON		
4	https://www.dropbox.com	POST	/ajax_needs_signup_captcha	✓	☐	200	1482	JSON		
5	https://www.dropbox.com	OPTI...	/ajax_register	✓	☐	200	1024	JSON		

Header : content-security-policy

style-src
- https://*
- 'unsafe-inline'
- 'unsafe-eval'

connect-src
- https://*
- ws://127.0.0.1:*/ws

script-src
- https://ajax.googleapis.com/ajax/libs/jquery/
- 'unsafe-eval'
- https://www.dropbox.com/static/
- https://cf.dropboxstatic.com/static/javascript/
- https://www.dropboxstatic.com/static/javascript/
- https://cf.dropboxstatic.com/static/api/
- https://www.dropboxstatic.com/static/api/
- https://www.google.com/recaptcha/api/
- 'nonce-yDqEXWDgP2zUUhD8Po0j'

object-src
- https://cf.dropboxstatic.com/static/
- https://www.dropboxstatic.com/static/
- 'self'
- https://flash.dropboxstatic.com
- https://swf.dropboxstatic.com

Demonstration: CSP

Bypass Included

GoSecure

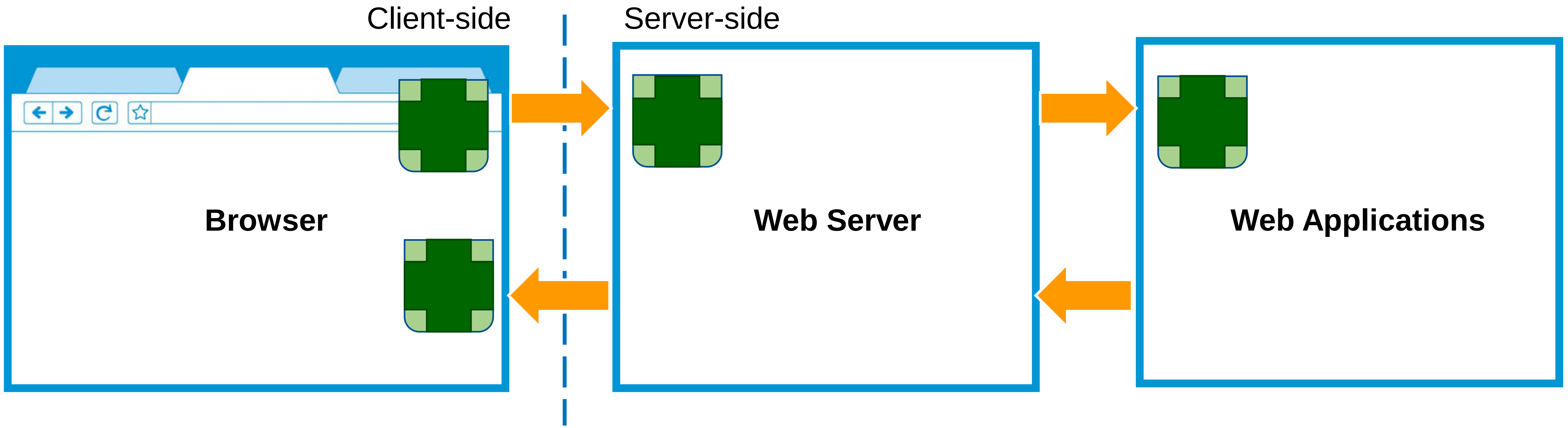
Conclusion

Guideline for Developers

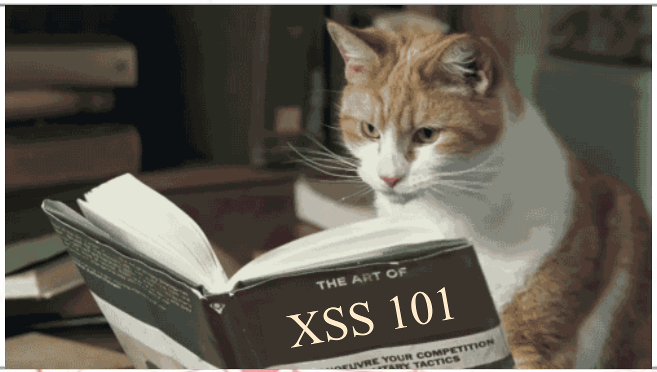
- Use a modern template engine
  - HTML encoding by default PLEASE!
- Encoding context is very important
  - HTML != Attribute != CSS != JavaScript
- Be careful when allowing HTML from user
- Be careful with file uploads
- Transformation can often lead to filter bypasses

Keep in Mind...

- No protection layer will be bullet proof
- Defense in depth
  - Avoid relying on a single layer



References


THE ART OF
XSS 101
OUTMANOEUVRE YOUR COMPETITION
MILITARY TACTICS

Recommended reading..

- WASP: Cross-site Scripting (XSS)
- OWASP: XSS Filter Evasion Cheat Sheet
- XSS without HTML: Client-Side Template Injection with AngularJS by Gareth Heyes James Kettle
- CSP 2015 by filedescriptor
- Bypassing ASP.NET ValidateRequest for stored XSS attack by *InfoSecAuditor*
- XSS Auditor bypass / Another one by Gareth Heyes
- X-XSS-Nightmare: XSS Attacks Exploiting XSS Filter (IE/Edge) by Masato Kinugawa

More recommended reading..

- Revisiting XSS Sanitization by Ashar Javed
- UTF-7 XSS attacks in modern browsers (Security Stack Exchange)
- DOM Clobbering by Gareth Heyes
- Towards Elimination of XSS Attacks with a Trusted and Capability Controlled. DOM by Mario Heiderich
- CSP Bypass using Angular and GIF by Mario Heiderich

Questions ?

Contact
parteau@gosecure.ca
gosecure.net/blog/
@h3xStream @GoSecure_Inc