Handling Untrusted Data

Max McCarty

@maxRmccarty https://lockmedown.com

Overview

Fuzzing Data with Zed Attack Proxy

Identifying Untrusted Data

Where and When to Handle Trusted Data

Whitelist versus Blacklist

Validating Untrusted Data

Escaping Untrusted Data

Why Sanitizing Isn’t So Sanitary

Identifying Untrusted Data

What constitutes as untrusted data?

Form Input Values

TRUSTED or UNTRUSTED?

User-Agent HTTP Request Header

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/54.0.2840.71 Safari/537.36

TRUSTED or UNTRUSTED?

HTML Hidden Field

```
<input type=”hidden">
```

TRUSTED or UNTRUSTED?

Data from Application Database

TRUSTED or UNTRUSTED?

How can we identify untrusted data?

Rule #1:

Any data that data that is explicitly being supplied from an external source can be identified as untrusted.

Form input values

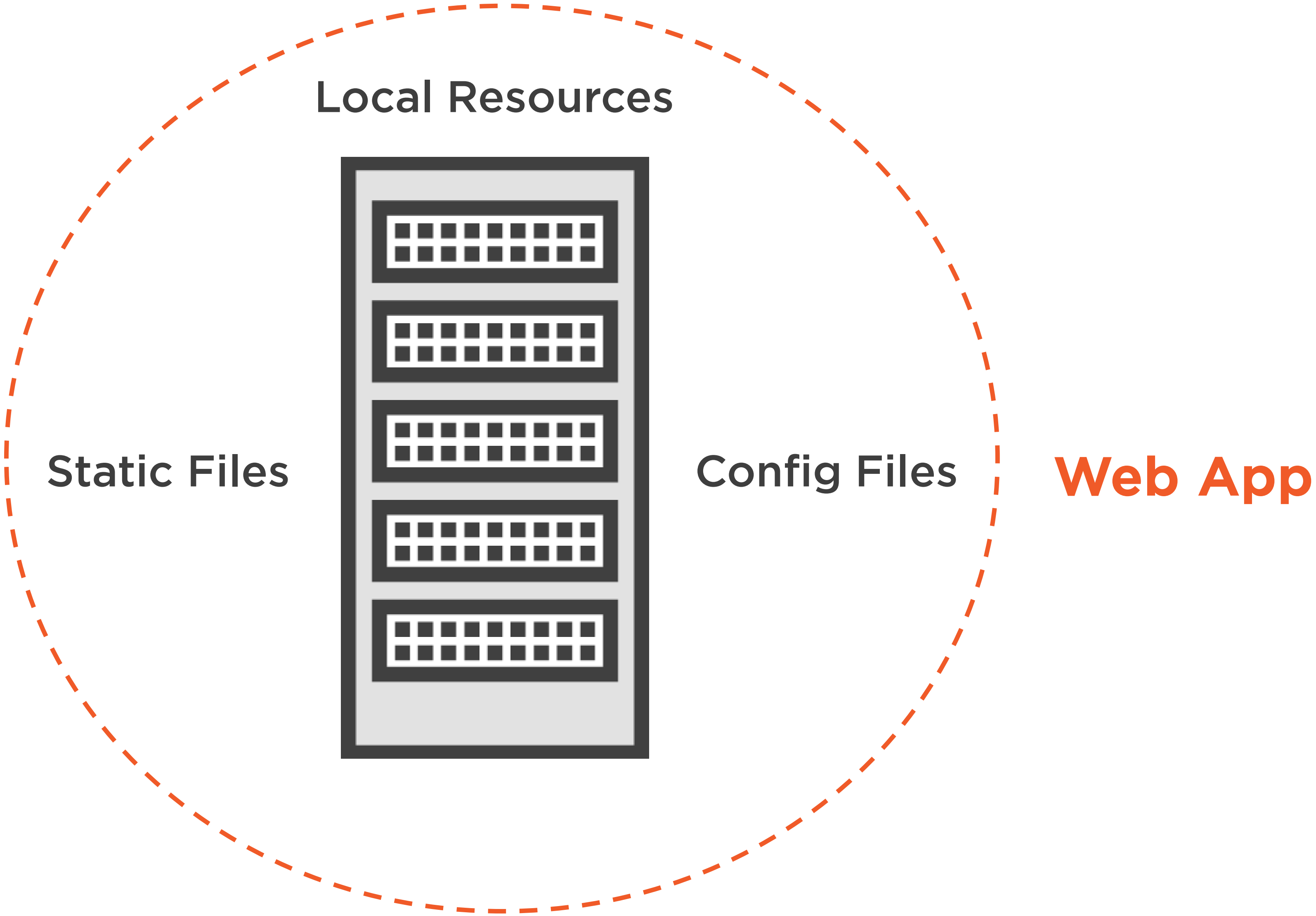
HTTP Request Headers

Proxy-Connection:	keep-alive
Content-Length:	117
Accept:	application/json, text/plain, */*
User-Agent:	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36
Content-Type:	application/json;charset=UTF-8
Accept-Language:	en-US,en;q=0.8

Rule #2

If the data has crossed a trust boundary, then it can be assumed to be untrusted data.

Potential Trust Boundary



Local Resources

Static Files

Config Files

Web App

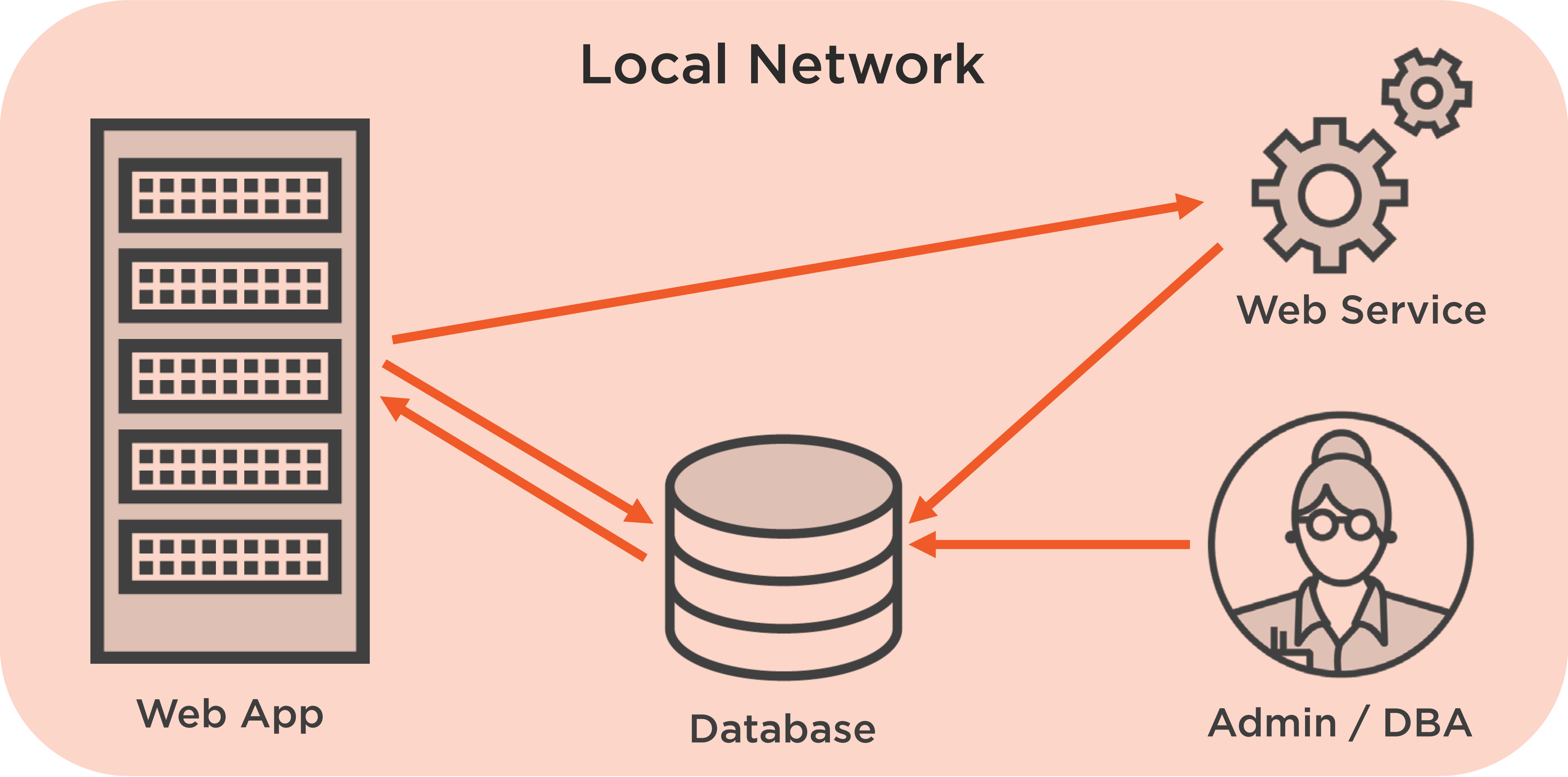
HTML Hidden Field

```
<input type=”hidden” value=“X3gAAAAOZMtj9d…” />
```

Rule #3

Be cognitive of who has access to the data.

Internal Resource / Threats



Identifying Untrusted Data

1. Apply Rules
2. Ask Questions
3. Make No Assumptions

Where and When to Handle Trusted Data

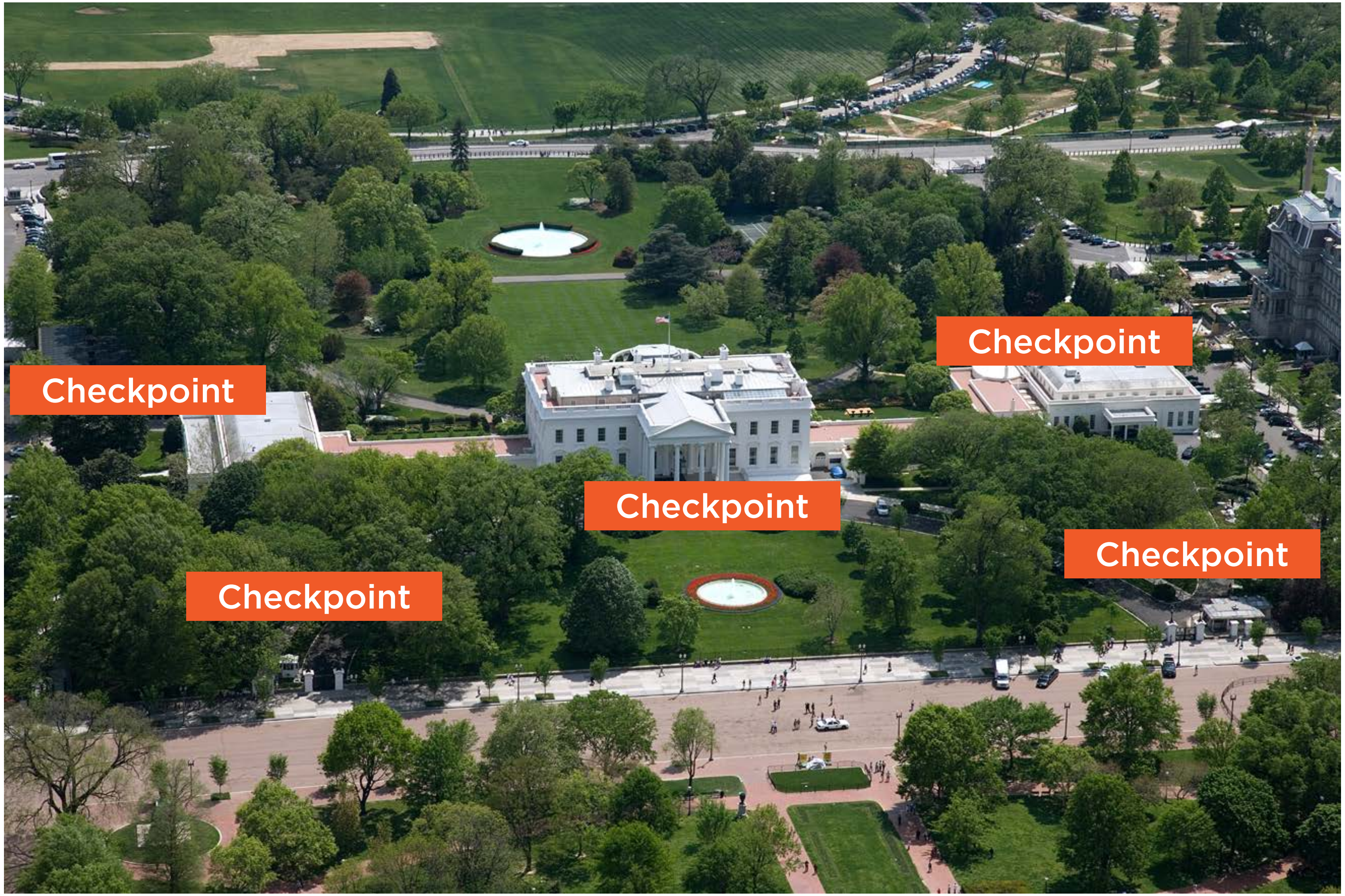
Difficulties of Determining When and Where

Every application is different

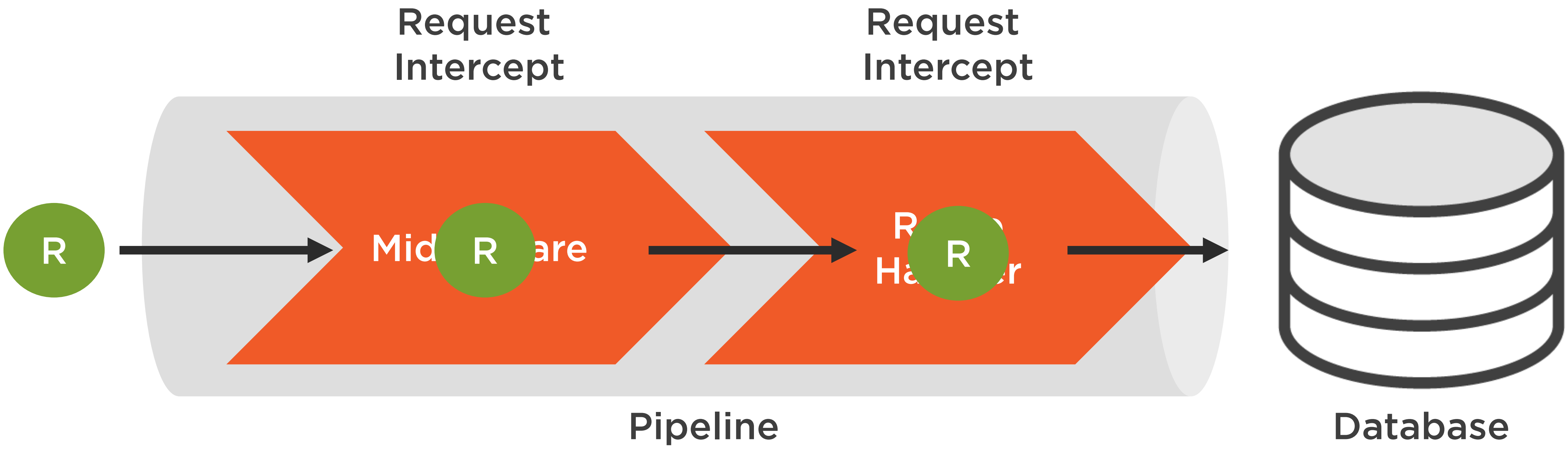
Application architectures is not the same

The number of data sources vary

White House Grounds



Targeted Database Example



Request Intercept

Request Intercept

R

Mid R are

R R
Ha er

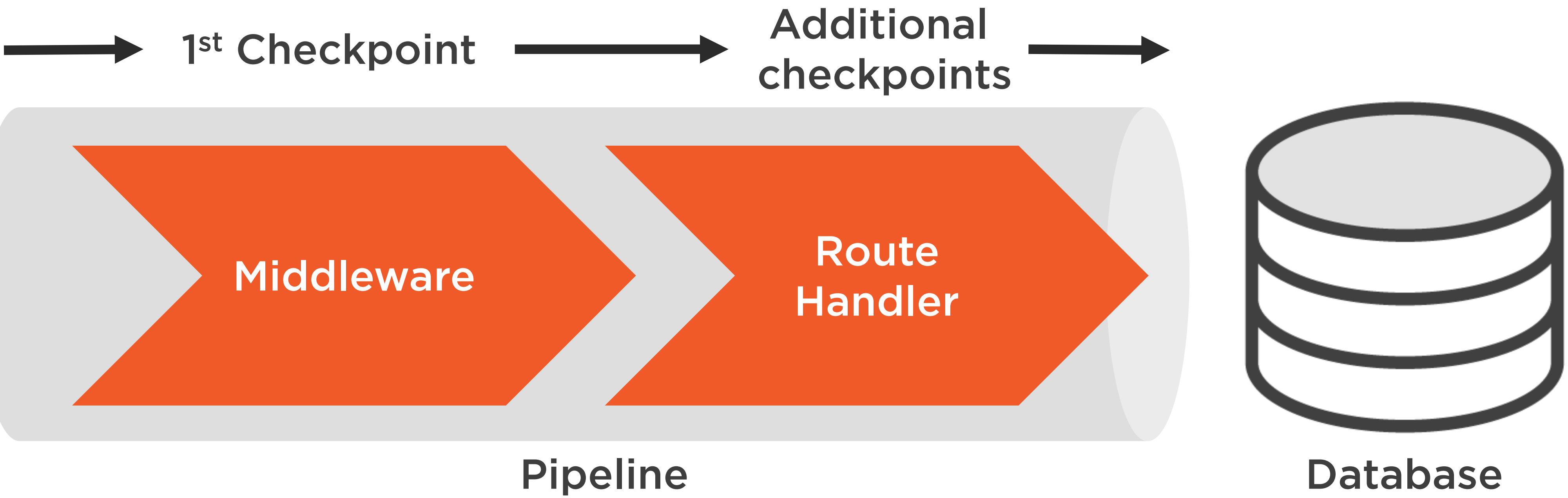
Pipeline

Database

Rule

Keep untrusted data as far away from critical systems as possible.

Multilayer Approach (Security In-depth)



1st Checkpoint

Additional checkpoints

Middleware

Route Handler

Pipeline

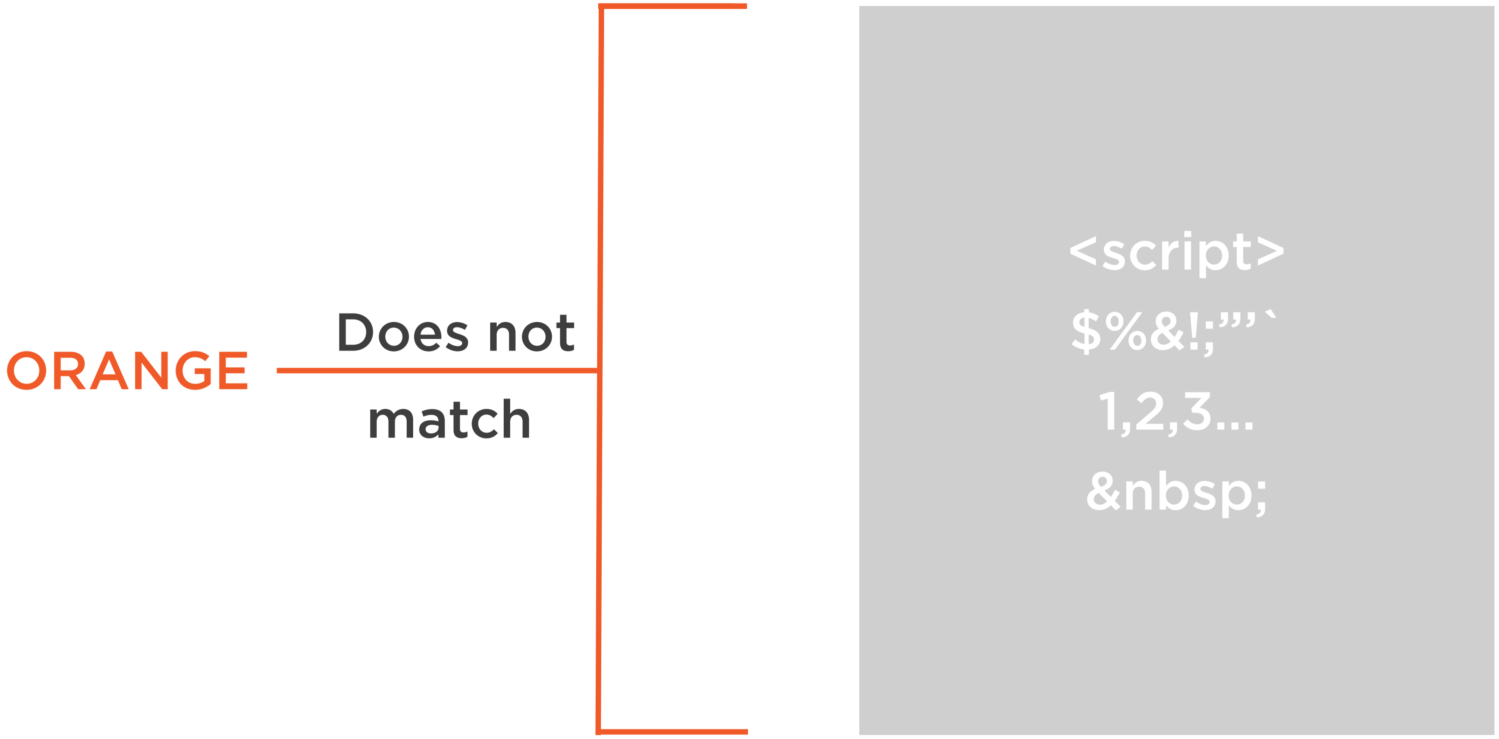
Database

Whitelist versus Blacklist Approaches

Blacklist Approach

Color Preference

Blacklist Values



ORANGE

Does not match

<script>

$%&!;”’`

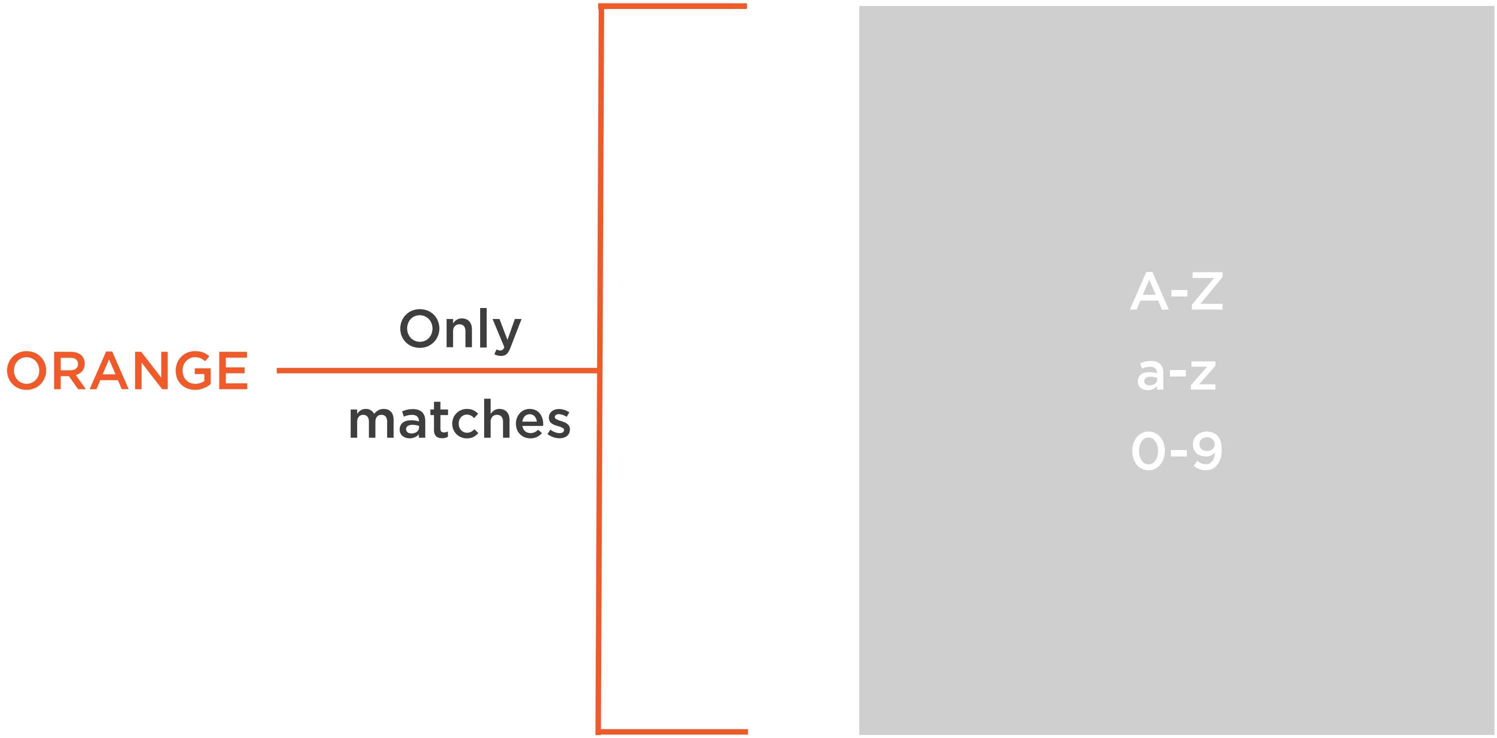
1,2,3…

&nbsp;

Whitelist Approach

Color Preference

Whitelist Values



ORANGE

Only matches

A-Z

a-z

0-9

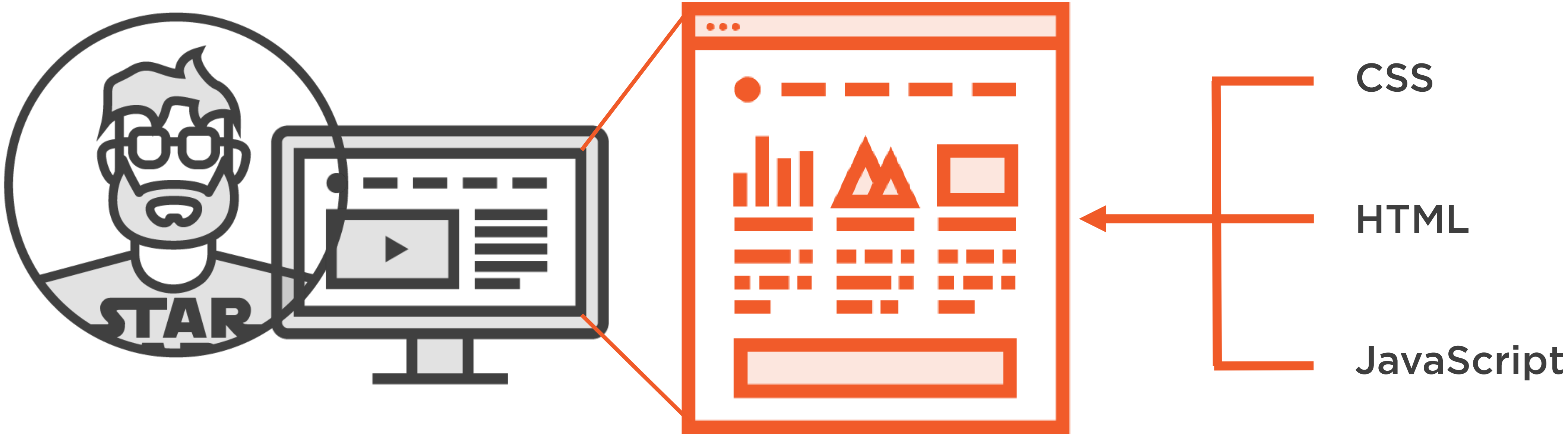
Escaping Untrusted Data

Escaping (Output Encoding)

Is a technique used to ensure that characters are treated as data, not as characters that are relevant to the interpreter’s parser.

Escaping simply lets the interpreter know that the data is not intended to be executed, and therefore prevents attacks from working.

Various Application Contexts

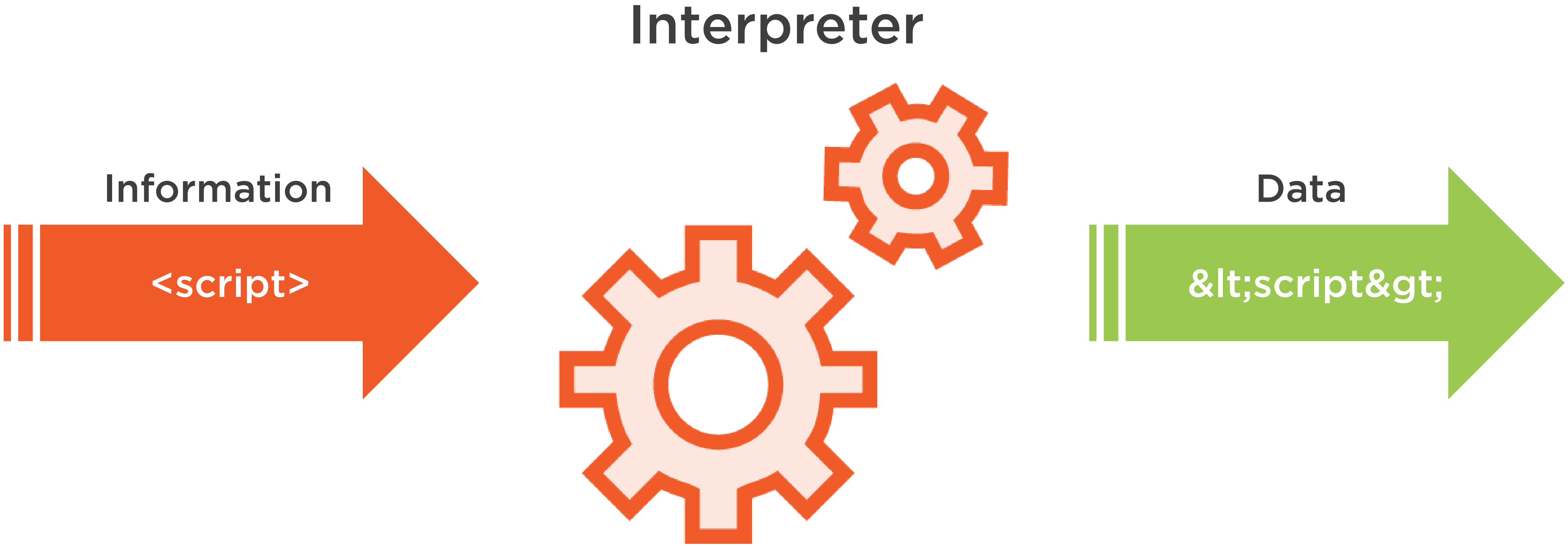


CSS

HTML

JavaScript

Escaping Process



Interpreter

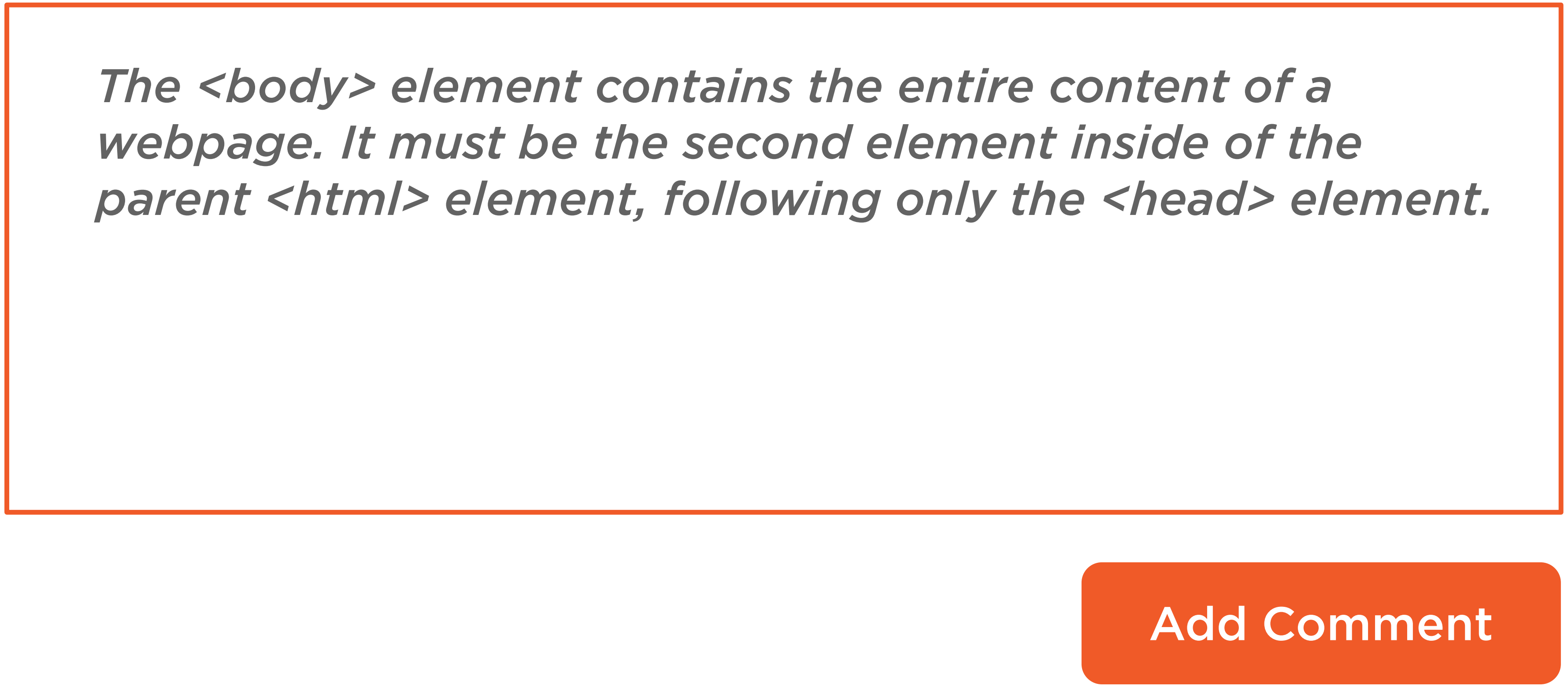
Information

<script>

Data

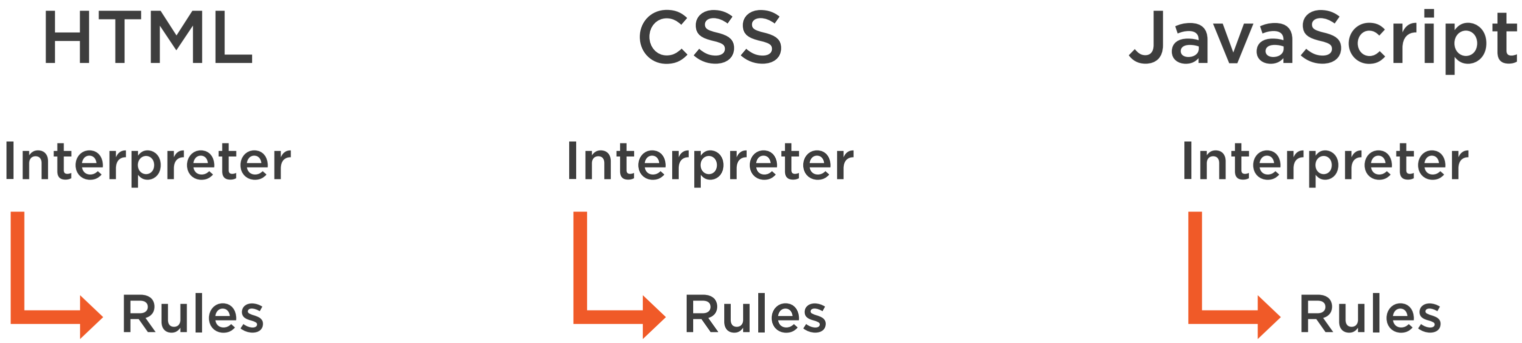
&lt;script&gt;

Escaping HTML Example



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.

Escaping Rules Are Specific to an Interpreter



HTML

Interpreter

Rules

CSS

Interpreter

Rules

JavaScript

Interpreter

Rules

Why Sanitizing Isn’t So Sanitary

Sanitizing

Is a process that attempts to sanitize the data by removing known values to be potentially malicious in order to make the data safe.

Potential Issues with Sanitizing

- Blacklist approach
- Maintenance requirements
- Context bound
- Potentially inadequate

Summary

Fuzzing Data with Zed Attack Proxy

Identifying Untrusted Data

Where and When to Handle Trusted Data

Whitelist versus Blacklist Approaches

Validating Untrusted Data

Escaping Untrusted Data

Why Sanitizing Isn’t So Sanitary