Cross-Site Scripting

Timo Pagel

OWASP


XSS
/CROSS/SITE/
SCRIPTING
BASICS


XSS
/CROSS/SITE/
SCRIPTING
ADVANCED

Cross Site Scripting Overview

Cross Site Scripting Overview

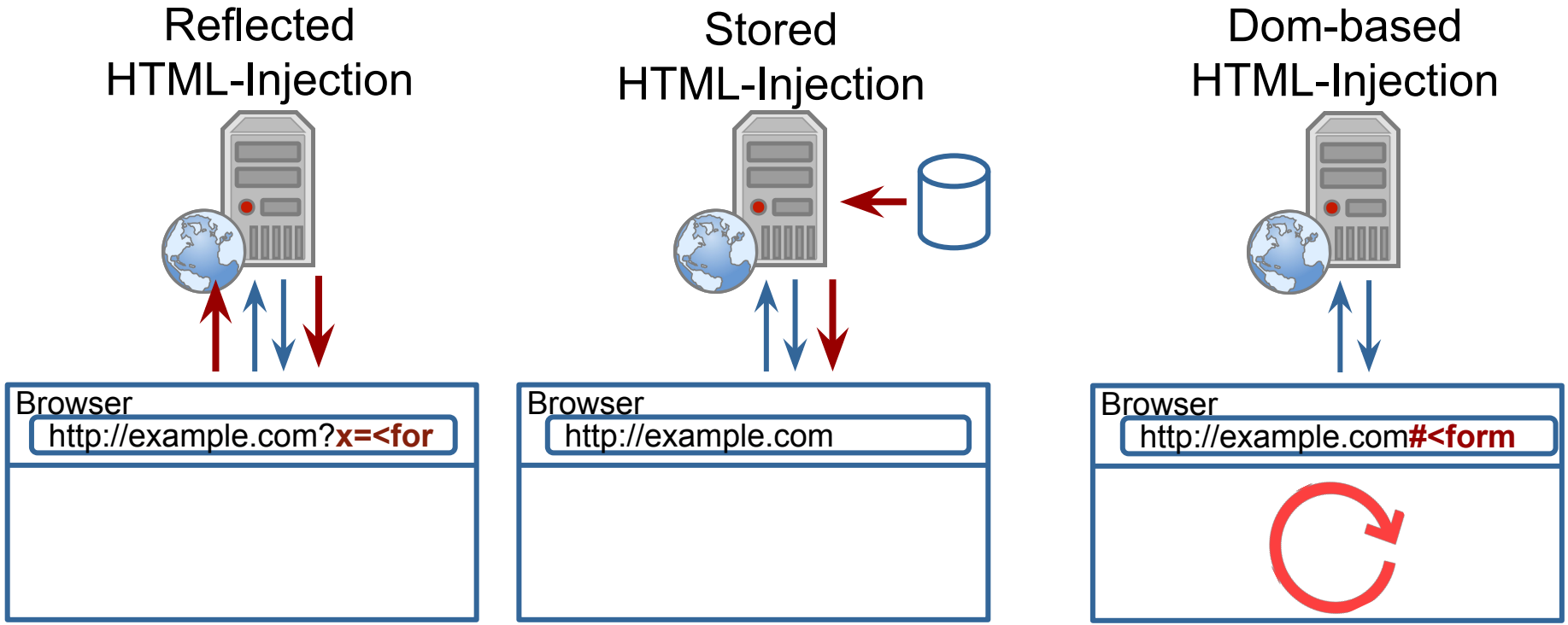
App. Specific	Exploitability ❸	Prevalence ❸	Detectability ❸	Technical ❷	Business ?
Automated tools can detect and exploit all three forms of XSS, and there are freely available exploitation frameworks.		XSS is the second most prevalent issue in the OWASP Top 10, and is found in around two thirds of all applications. Automated tools can find some XSS problems automatically, particularly in mature technologies such as PHP, J2EE / JSP, and ASP.NET.		XSS is the second most prevalent issue in the OWASP Top 10, and is found in around two thirds of all applications.	

Angular

Declarative Templating System

Angular

HTML Injection Overview (User-based View)



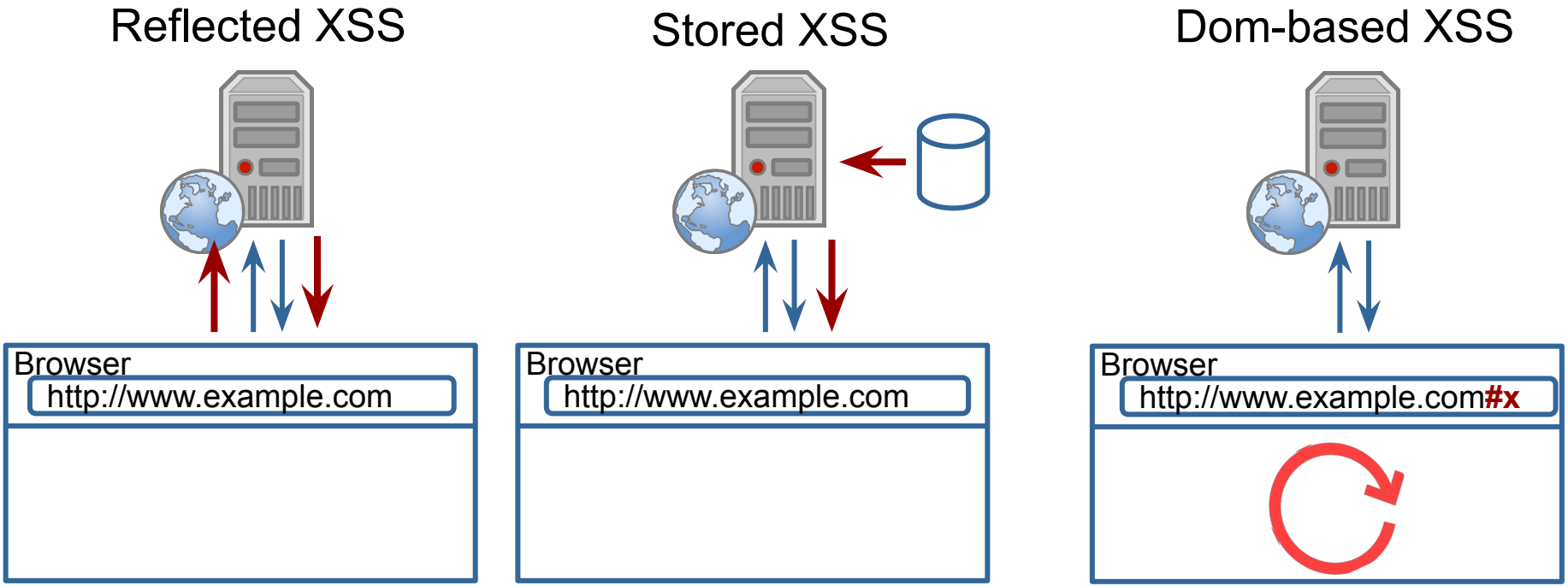
Basic Example of HTML-Injection

```
var userposition=location.href.indexOf("user=");

var user=location.href.substring(userposition+4);

document.getElementById("Welcome").innerHTML
=" Hello, "+user;
```

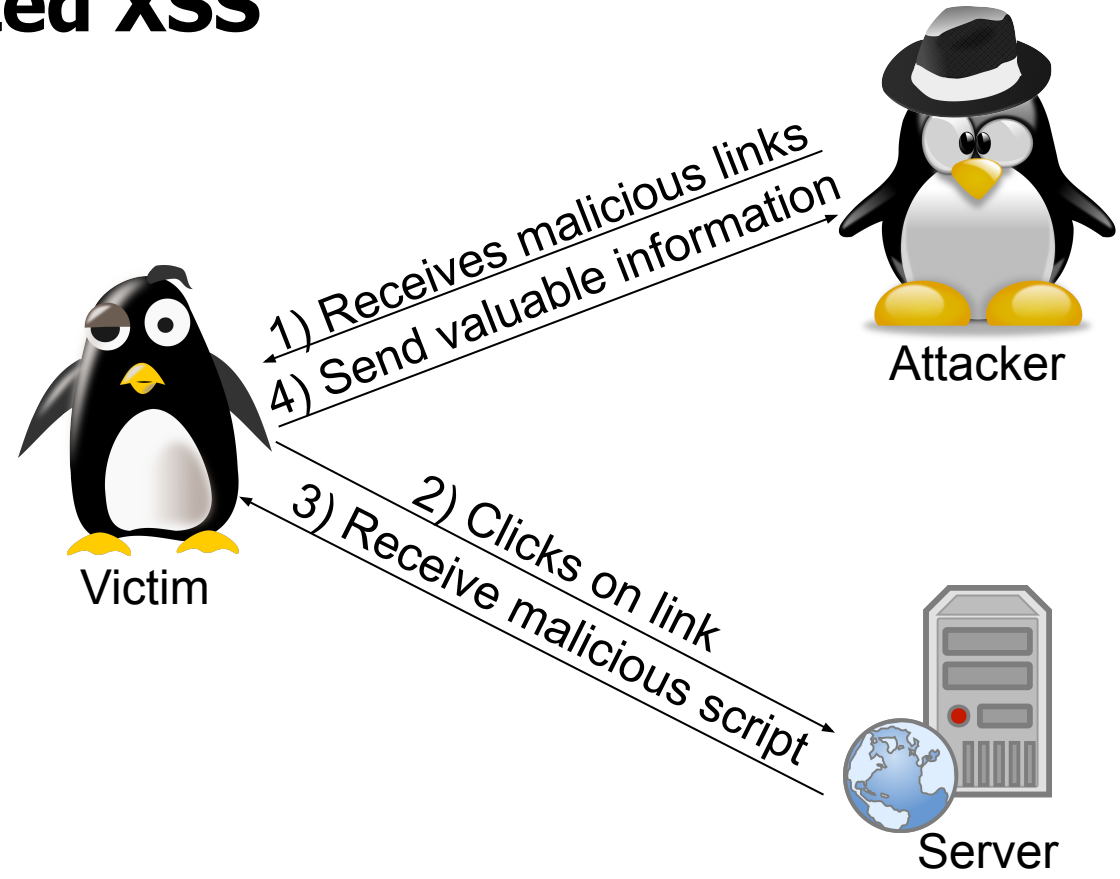
Intended URL: http://timo-pagel.de?user=timo

HTML-Injection: http://timo-pagel.de?user=<form>

XSS Overview (User-based View)



Reflected XSS



Demo: XSS impacts

Security by Design

Angular treats all input as untrusted by default

Server-Side Template Injection

- Logic is in JavaScript
- REST is used to perform CRUD-Operations
- The server **must not** generate templates

Any template received from the server is trusted

-> Static content (e.g. index.html) via nginx

Security Context in Angular

- HTML is used when interpreting a value as HTML, for example, when binding to `innerHtml`.

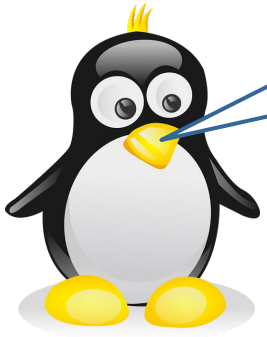
Security Context in Angular

- HTML is used when interpreting a value as HTML, for example, when binding to `innerHtml`.

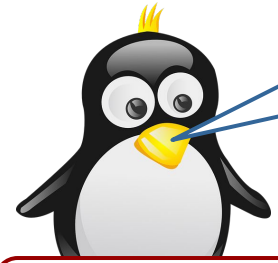
Will <script>alert(‘x’)</script> be interpreted with innerHtml?

OWASP

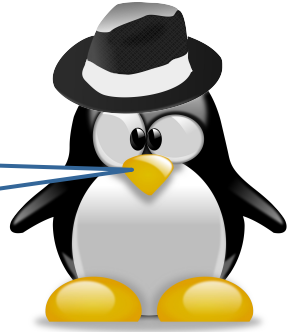
I remove every “<script>” tag from user input



I remove every “<script>” tag from user input

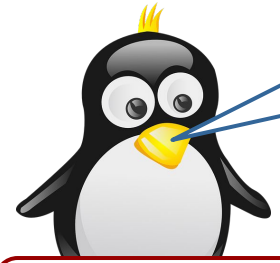


What is about JavaScript in tags?

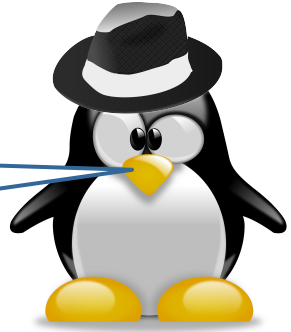


```
<img src=x:alert(alt) onerror=eval(src) alt=0>
```

I remove every “<script>” tag from user input



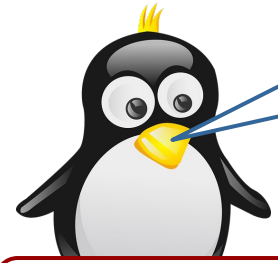
What is about JavaScript in tags?



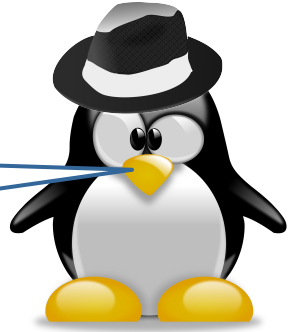
```
<img src=x:alert(alt) onerror=eval(src) alt=0>
```

```
<iframe src="javascript:alert(0)">
```

I remove every “<script>” tag from user input



What is about JavaScript in tags?

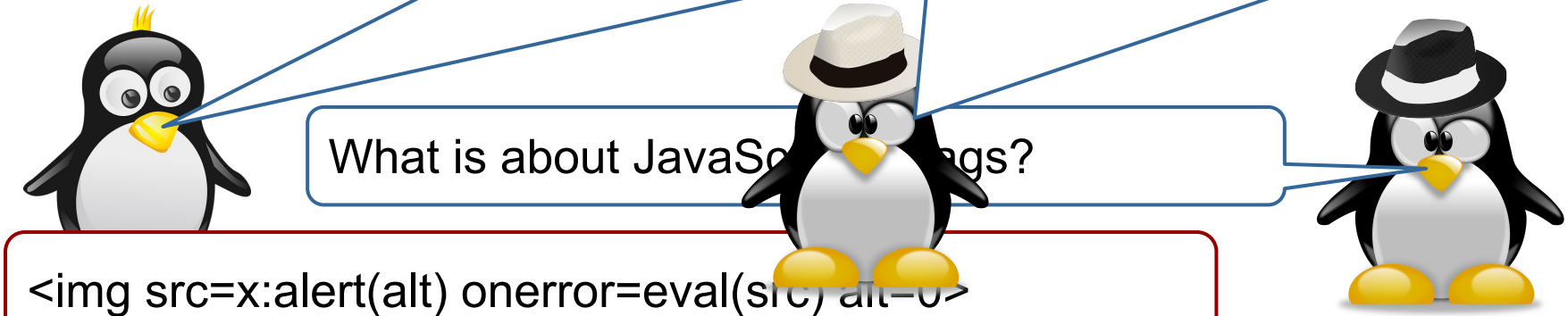


```
<img src=x:alert(alt) onerror=eval(src) alt=0>
```

```
<iframe src="javascript:alert(0)">
```

```
<svg:g onload="alert(8)"/>
```

The force is strong in **WHITELISTS**



What is about JavaSc ags?


```
<img src=x:alert(alt) onerror=eval(src) alt=0>
```

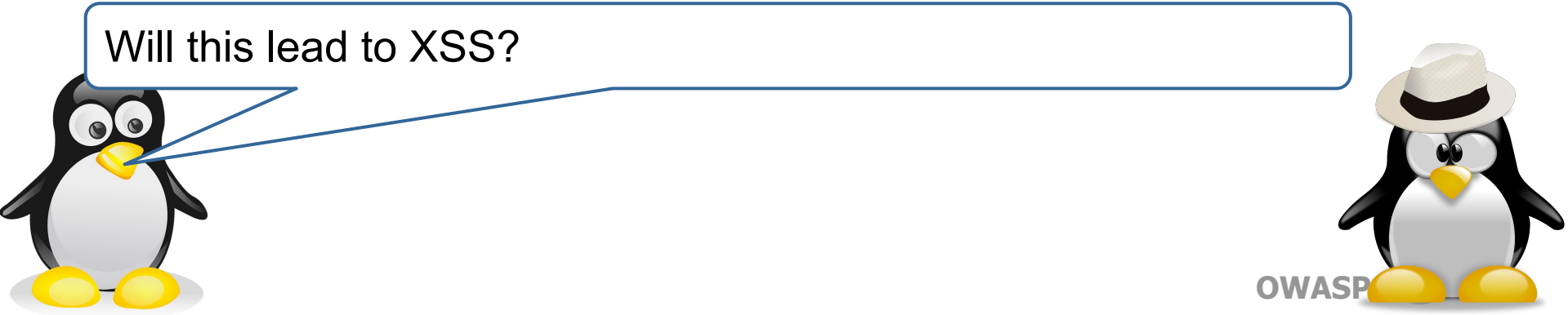
```
<iframe src="javascript:alert(0)">
```

```
<svg:g onload="alert(8)"/>
```

Interpolating and Binding (example from https://angular.io/guide/security)

src/app/inner-html-binding.component.html

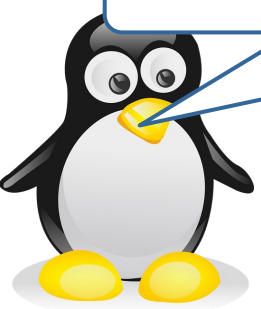
```
<h3>Binding innerHTML</h3>
<p>Bound value:</p>
<p class="e2e-inner-html-interpolated">{{htmlSnippet}}</p>
```

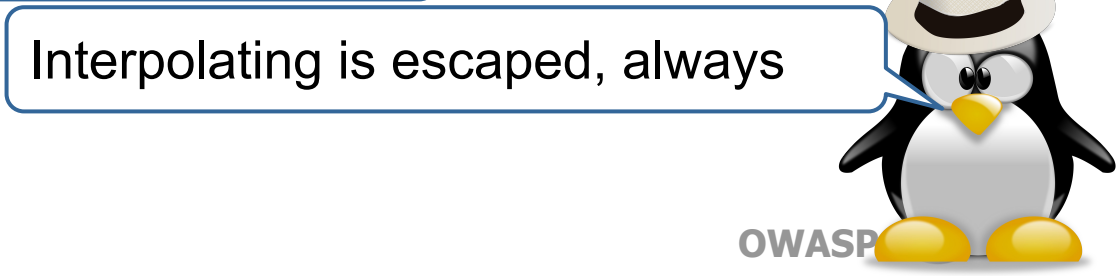


Interpolating and Binding

src/app/inner-html-binding.component.html

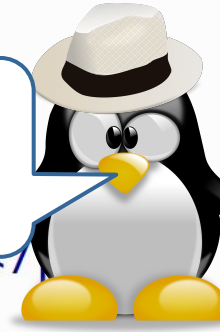
```
<h3>Binding innerHTML</h3>
<p>Bound value:</p>
<p class="e2e-inner-html-interpolated">{{htmlSnippet}}</p>
```





Interpolating and Binding

src/app/inner-html-binding.component.html



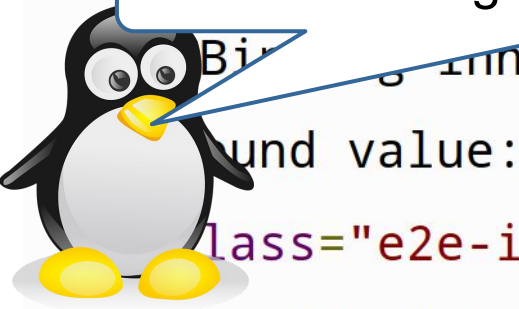

```
...innerHTML</h3>
...und value:</p>
...lass="e2e-inner-html-interpolated ></p>
  <p>Result of binding to innerHTML:</p>
  <p class="e2e-inner-html-bound" [innerHTML]="htmlSnippet"></p>
```

src/app/inner-html-binding.component.ts (class)

```
export class InnerHtmlBindingComponent {
  // For example, a user/attacker-controlled value from a URL.
  htmlSnippet = 'Template <script>alert("Owned")</script> <b>Syntax</b>';
}
```

Interpolating and Binding

src/app/inner-html-binding.component.html




```
Bi    innerHTML</h3>
und value:</p>
lass="e2e-inner-html-interpolated">{{htmlSnippet}}</
<p>Result of binding to innerHTML:</p>
<p class="e2e-inner-html-bound" [innerHTML]="htmlSnippet"></p>
```

src/app/inner-html-binding.component.ts (class)

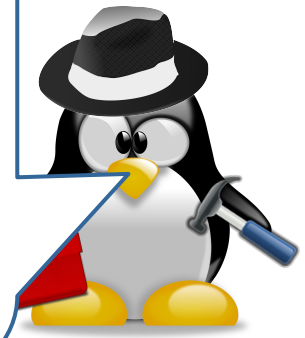
```
export class InnerHtmlBindingComponent {
  // For example, a user/attacker-controlled value from a URL.
  htmlSnippet = 'Template <script>alert("Owned")</script> <b>Syntax</b>';
}
```

/CROSS/SITE/
SCRIPTING

shop.pagel.pro: Perform a reflected XSS attack with *<iframe src="javascript:alert(`xss`)">*.

Hint 1/1:

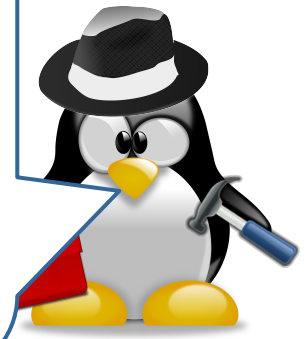
Next hint: 5 min
10 Minutes left



shop.pagel.pro: Perform a reflected XSS attack with *<iframe src="javascript:alert(`xss`)">*.

Hint 1/1: Check the tracking process

Next hint: 5 min
10 Minutes left



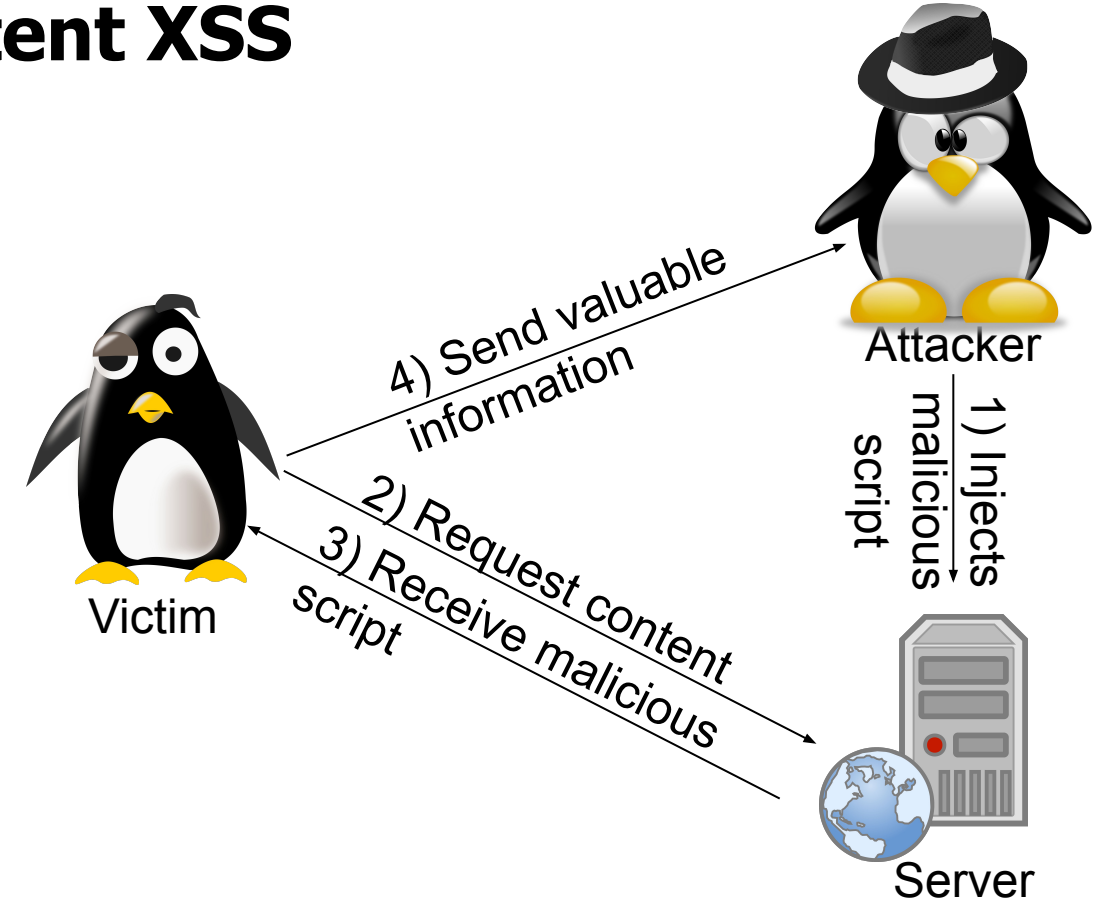
shop.pagel.pro: Perform a reflected XSS attack with *<iframe src="javascript:alert(`xss`)">*.

Hint 1/1: Check the tracking process

Next hint: 5 min
10 Minutes left

Solution:
https://shop.pagel.pro/#/track-result?id=%3Ciframe%20src%3D%22javascript:alert(0)%22%3E

Persistent XSS



Based on: http://www.opensourceforu.com/wp-content/uploads/2010/09/Figure-3_stored-attack.jpg, April 2015

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> bypassing a client-side security mechanism.*

Hint 1/3: Check the whole registration/authentication process

Next Hint: 5 Minutes

20 Minutes left

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> bypassing a client-side security mechanism.*

Hint 1/3: Check the whole authentication process
Hint 2/3: Check http://shop.pagel.pro/#/administration

15 Minutes left

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> bypassing a client-side security mechanism.*

Hint 1/3: Check the whole authentication process
Hint 2/3: Check http://shop.pagel.pro/#/administration
Hint 3/3: Content-Length set?
10 Minutes left

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> bypassing a client-side security mechanism.*

Hint 1/3: Check the whole authentication process
Hint 2/3: Check http://shop.pagel.pro/#/administration
Hint 3/3: Content-Length set?

Solution:

Methode Adresse

POST https://shop.pagel.pro/api/Users/

Request-Header:

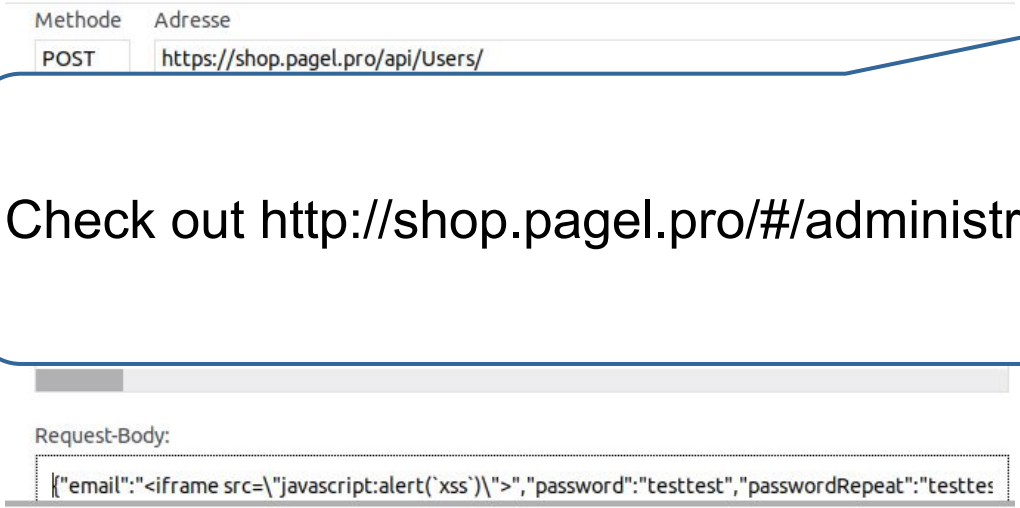
```
Content-Type: application/json
Connection: keep-alive
Cookie: JuceShopStickySessionID=.13020; language=en; token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpX
Pragma: no-cache
Cache-Control: no-cache
[...]
```

Request-Body:

```
{"email":"<iframe src=\"javascript:alert(`xss`)\">","password":"testtest","passwordRepeat":"testtes
```

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> bypassing a client-side security mechanism.*

Hint 1/3: Check the whole authentication process
Hint 2/3: Check http://shop.pagel.pro/#/administration
Hint 3/3: Content-Length set?
Solution:

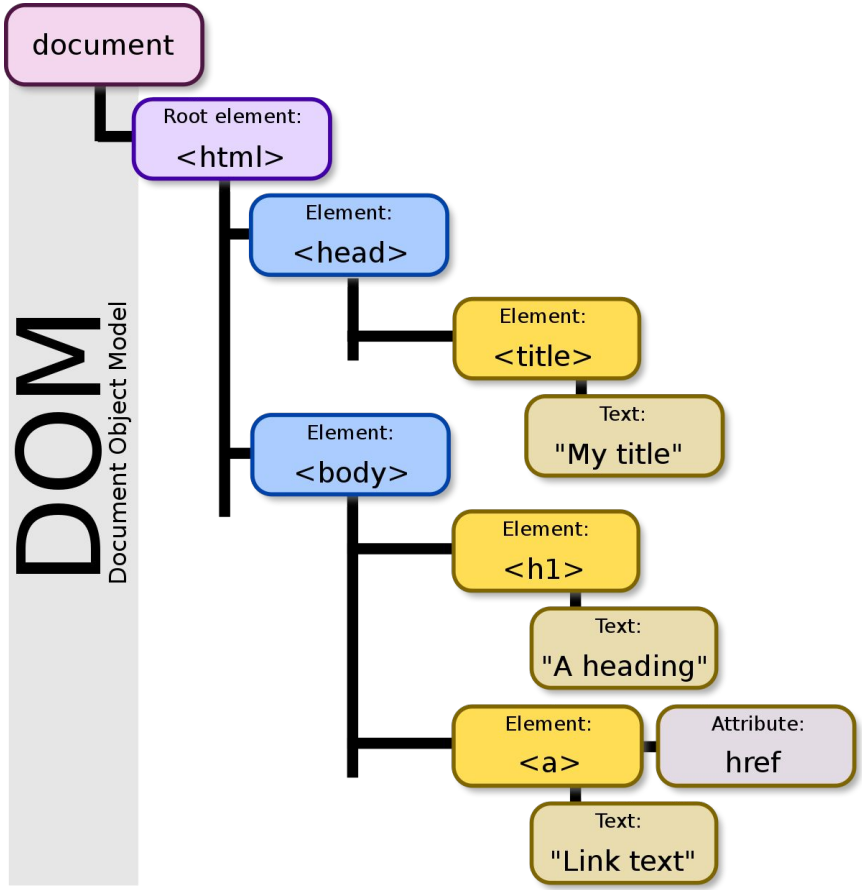


Check out http://shop.pagel.pro/#/administration

Dom Based XSS

Embedding by browser (not server)

Document Object Model



Hands On

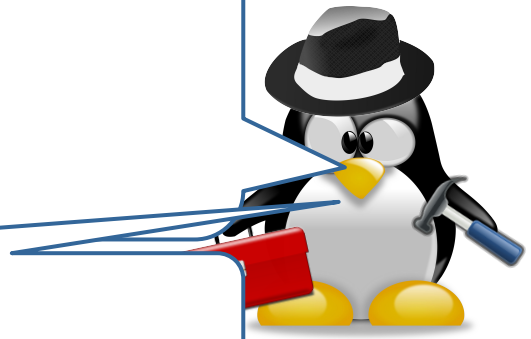
Show an alert box with the content `"XSS"` on http://shop.pagel.pro in an alert box

2 Minutes

Hands On

Show an alert box with the content `"XSS"` on http://shop.pagel.pro in an alert box

2 Minutes till hint

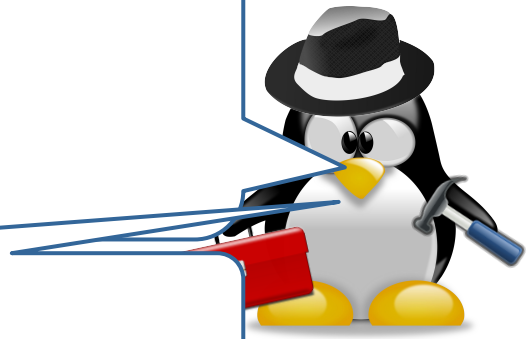


Hint: <iframe src="....">

Hands On

Show an alert box with the content `"XSS"` on http://shop.pagel.pro in an alert box

2 Minutes till hint



Hint: <iframe src="....">

Hands On

Show an alert box with the content `"XSS"` on http://shop.pagel.pro in an alert box

2 Minutes till hint

Hint: <iframe src="....">

Solution: <iframe src="javascript:alert(`xss`)">

Defense against Cross Site Scripting

- Input Validation
- HTTP-Headers
  - Content Security Policy
  - HttpOnly-Flag
- POST instead of GET
- Block JavaScript (Client Side)

Hands On Again!

shop.pagel.pro: *Perform an XSS attack on a legacy page within the application*

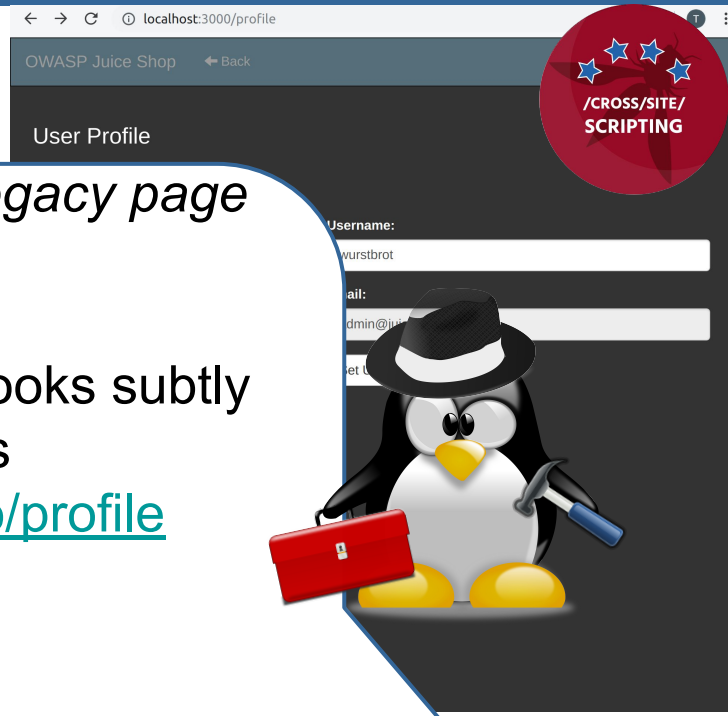
Hint 1/4: Find a screen in the application that looks subtly odd and dated compared with all other screens

10 Minutes left

Hands On Again!

shop.pagel.pro: *Perform an XSS attack on a legacy page within the application*

Hint 1/4: Find a screen in the application that looks subtly odd and dated compared with all other screens
Hint 2/4: Login and check http://shop.pagel.pro/profile

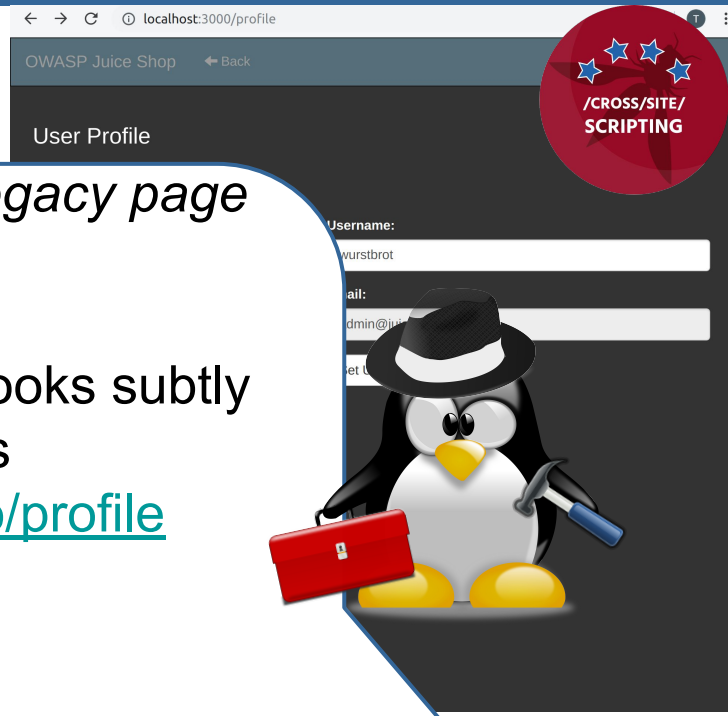


10 Minutes left

Hands On Again!

shop.pagel.pro: *Perform an XSS attack on a legacy page within the application*

Hint 1/4: Find a screen in the application that looks subtly odd and dated compared with all other screens
Hint 2/4: Login and check http://shop.pagel.pro/profile
Hint 3/4: Identify a XSS-able input field

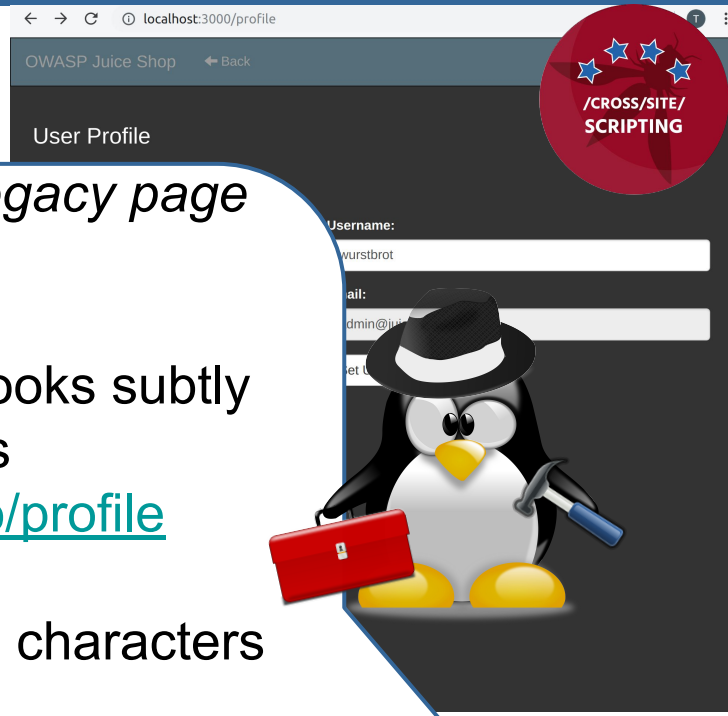


10 Minutes left

Hands On Again!

shop.pagel.pro: *Perform an XSS attack on a legacy page within the application*

Hint 1/4: Find a screen in the application that looks subtly odd and dated compared with all other screens
Hint 2/4: Login and check http://shop.pagel.pro/profile
Hint 3/4: Identify a XSS-able input field
Hint 4/4: Try to bypass the XSS “sanitizer” with characters like << >> |*a* ||*a* in the username input field



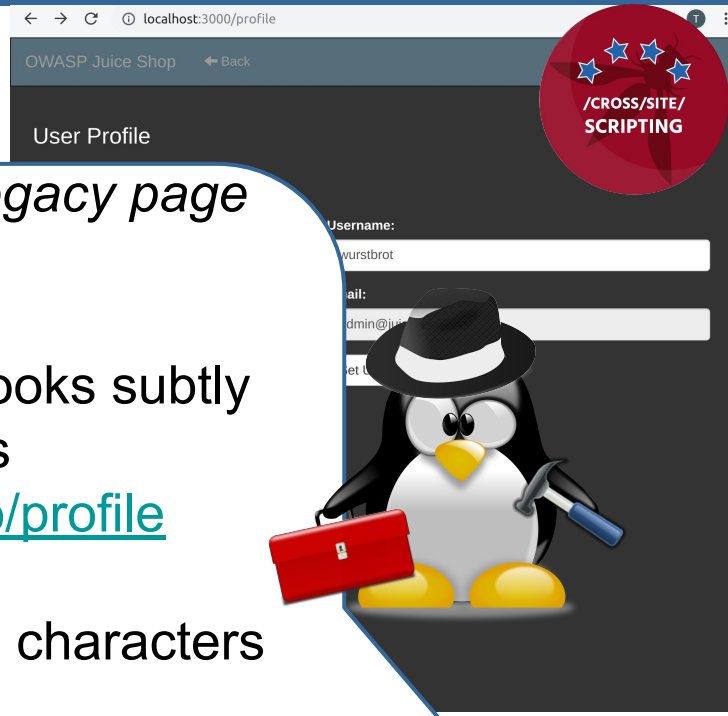
10 Minutes left

Hands On Again!

shop.pagel.pro: *Perform an XSS attack on a legacy page within the application*

Hint 1/4: Find a screen in the application that looks subtly odd and dated compared with all other screens
Hint 2/4: Login and check http://shop.pagel.pro/profile
Hint 3/4: Identify a XSS-able input field
Hint 4/4: Try to bypass the XSS “sanitizer” with characters like << >> |*a* ||*a* in the username input field
Solution: <<a|ascript>alert(`xss`)</script>

10 Minutes left



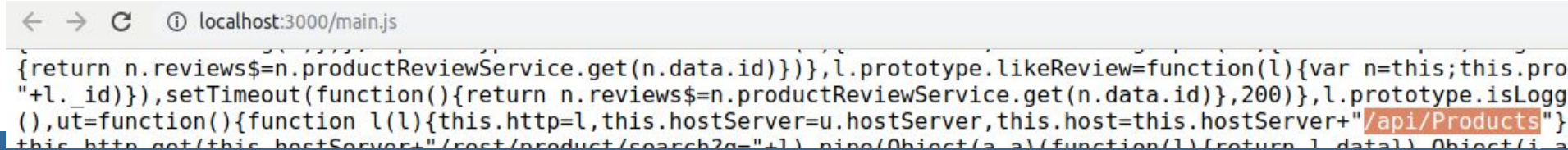
Backup Slides

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> without using the frontend application at all.*

Hint 1/3: Careless developers might have exposed API methods that the client does not even need -> find the XHR endpoint

Solution:

10 Minutes left



shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> without using the frontend application at all.*

Hint 1/3: Careless developers might have exposed API methods that the client does not even need -> find the XHR endpoint
Hint 2/3: Perform a OPTIONs request on */api/Products*
Hint 3/3

Solution:
10 Minutes left

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> without using the frontend application at all.*

Hint 1/3: Careless developers might have exposed API methods that the client does not even need -> find the XHR endpoint
Hint 2/3: Perform a OPTIONs request on */api/Products*
Hint 3/3: Try to post title/name/description of a product

Solution:
10 Minutes left

shop.pagel.pro: *Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> without using the frontend application at all.*

Hint 1/3: Careless developers might have exposed API methods that the client does not even need -> find the XHR endpoint
Hint 2/3: Perform a OPTIONs request
Hint 3/3: Try to post title/name/descrip

Solution:
10 Minutes left

$ curl -X OPTIONS -D - 'https://shop.pagel.pro/api/Products/1'
HTTP/1.1 204 No Content
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: **GET,HEAD,PUT,PATCH,POST,DELETE**
Vary: Access-Control-Request-Headers
Content-Length: 0
Date: Sun, 02 Jun 2019 18:58:43 GMT
Connection: keep-alive

localhost:3000/main.js

```
{return n.reviews$=n.productReviewService.get(n.data.id)})
"+l._id)}),setTimeout(function(){return n.reviews$=n.produ
(),ut=function(){function l(l){this.http=l,this.hostServer
```

shop.pagel.pro: *Perform a persisted XSS attack with*

<i

fr

Methode Adresse

PUT http://localhost:3000/api/Products/1

H

Request-Header:

m

Content-Type: application/json

X

H

H

S

1

Request-Body:

{"description":"JA<iframe src=\"javascript:alert(`xss`)\">"}

{return n.reviews$=n.productReviewService.get(n.data.id)})},l.prototype.likeReview=function(l){var n=this;this.pro
"+l._id)}),setTimeout(function(){return n.reviews$=n.productReviewService.get(n.data.id)},200)},l.prototype.isLogg
(),ut=function(){function l(l){this.http=l,this.hostServer=u.hostServer,this.host=this.hostServer+"/api/Products"}