XSS
Cross Site Scripting

JOÃO PAULO BARRACA

Prevalence and Detectability

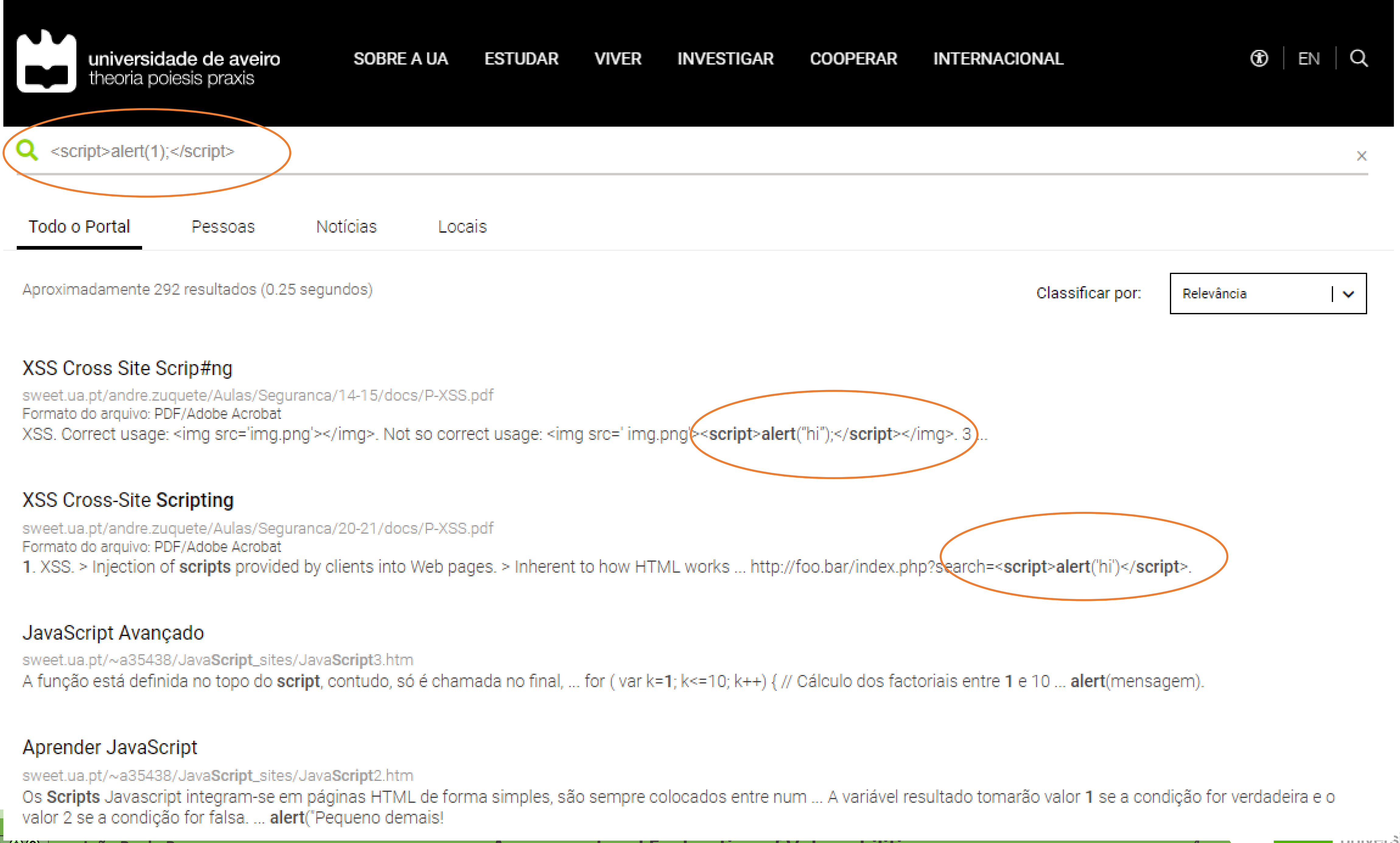
- Second most prevalent issue in the OWASP Top 10
  - Found in around two thirds of all applications.

- Automated tools can find some XSS problems automatically
  - particularly in mature technologies: PHP, J2EE / JSP, and ASP.NET.

Impact

- Moderate for reflected and DOM XSS

- Severe for stored XSS
  - with remote code execution on the victim’s browser
  - stealing credentials, sessions
  - delivering malware to the victim



<script>alert(1);</script>

Aproximadamente 292 resultados (0.25 segundos)

XSS Cross Site Scrip#ng

sweet.ua.pt/andre.zuquete/Aulas/Seguranca/14-15/docs/P-XSS.pdf

Formato do arquivo: PDF/Adobe Acrobat

XSS. Correct usage: <img src='img.png'></img>. Not so correct usage: <img src=' img.png'><**script**>**alert**("hi");</**script**></img>. 3 ...

XSS Cross-Site **Scripting**

sweet.ua.pt/andre.zuquete/Aulas/Seguranca/20-21/docs/P-XSS.pdf

Formato do arquivo: PDF/Adobe Acrobat

1. XSS. > Injection of **scripts** provided by clients into Web pages. > Inherent to how HTML works ... http://foo.bar/index.php?search=<**script**>**alert**('hi')</**script**>.

JavaScript Avançado

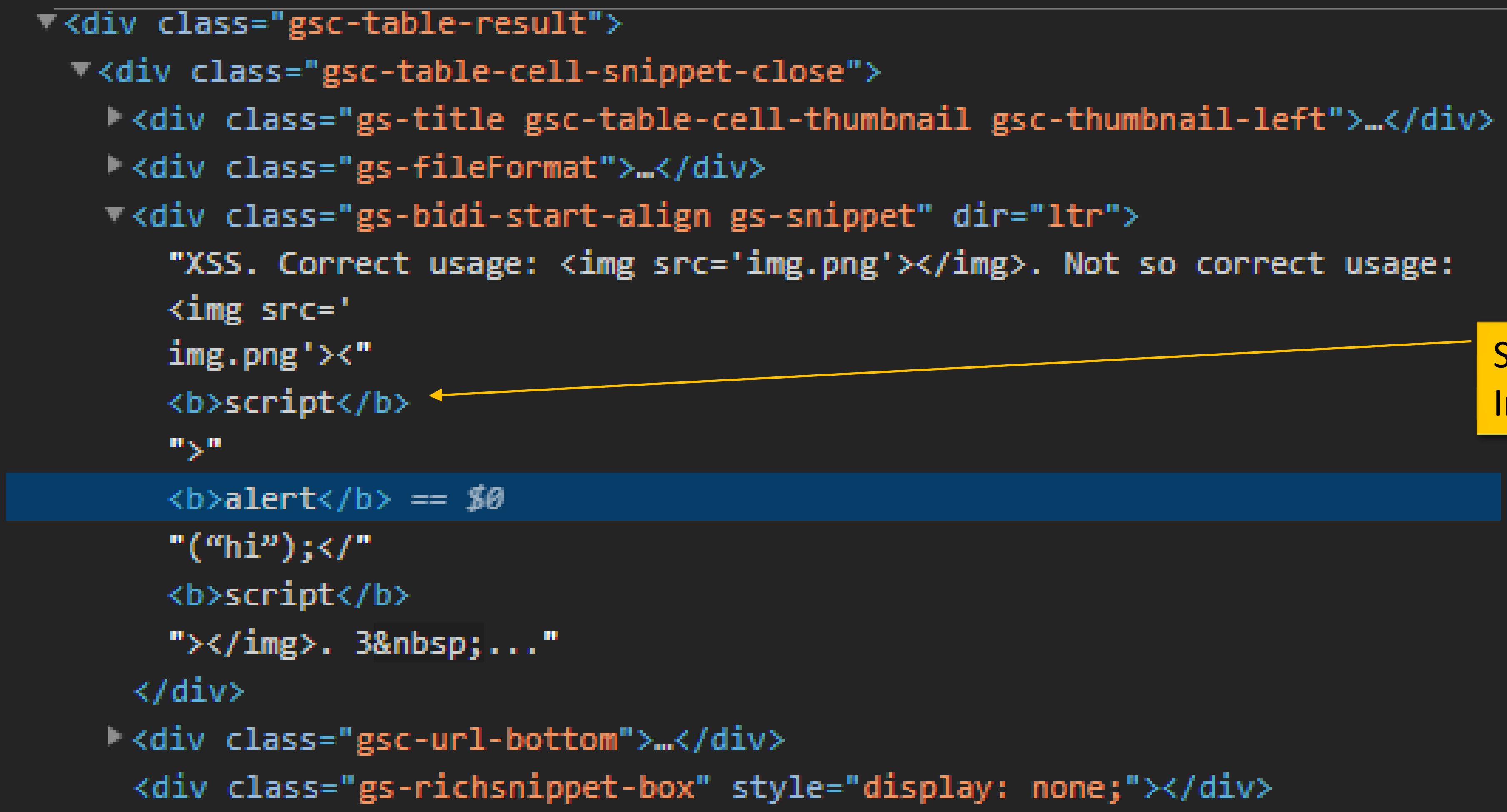
sweet.ua.pt/~a35438/Java**Script**_sites/Java**Script**3.htm

A função está definida no topo do **script**, contudo, só é chamada no final, ... for (var k=**1**; k<=10; k++) { // Cálculo dos factoriais entre **1** e 10 ... **alert**(mensagem).

Aprender JavaScript

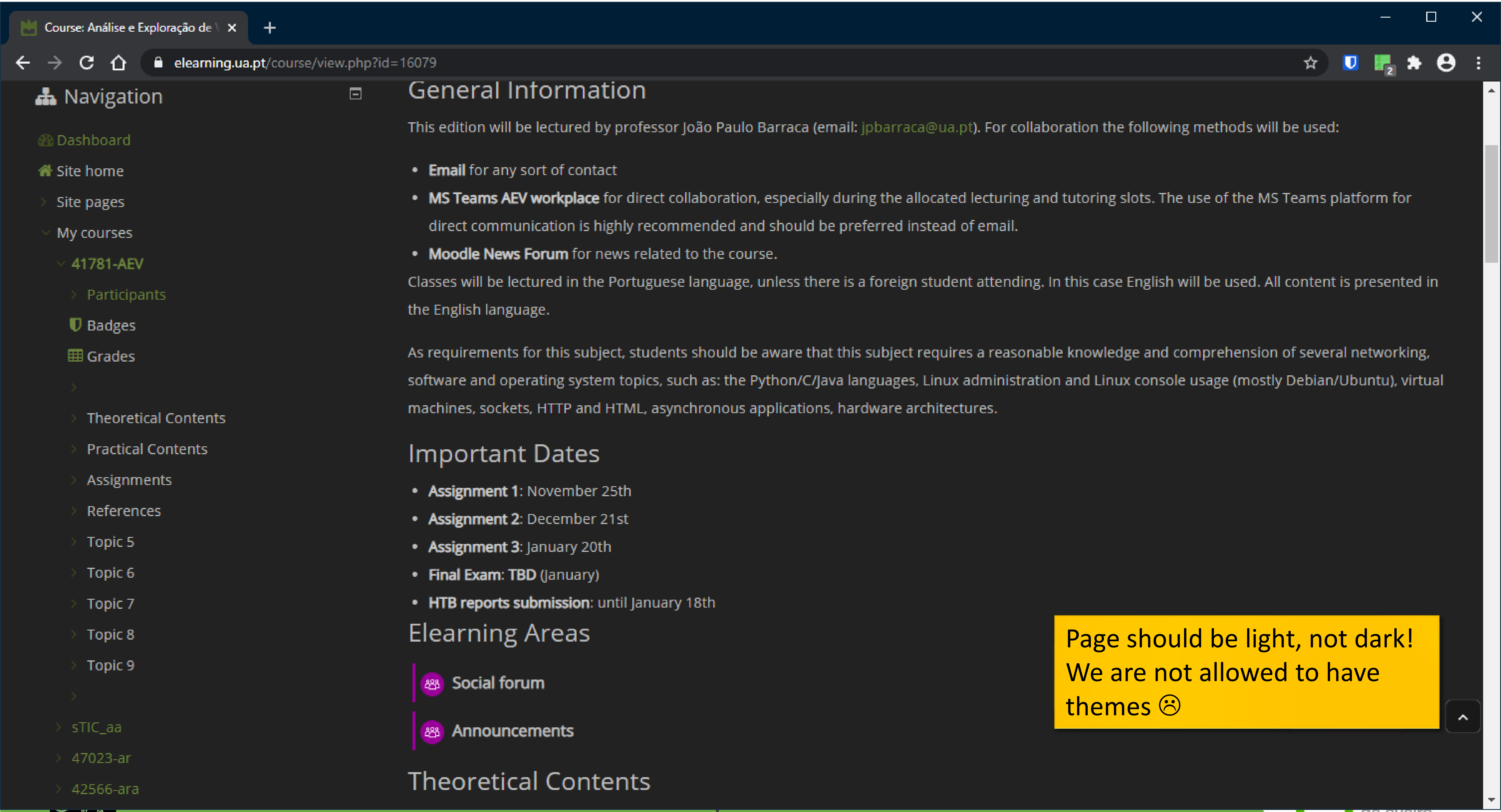
sweet.ua.pt/~a35438/Java**Script**_sites/Java**Script**2.htm

Os **Scripts** Javascript integram-se em páginas HTML de forma simples, são sempre colocados entre num ... A variável resultado tomarão valor **1** se a condição for verdadeira e o valor 2 se a condição for falsa. ... **alert**("Pequeno demais!



```
▼<div class="gsc-table-result">
  ▼<div class="gsc-table-cell-snippet-close">
    ▶<div class="gs-title gsc-table-cell-thumbnail gsc-thumbnail-left">…</div>
    ▶<div class="gs-fileFormat">…</div>
    ▼<div class="gs-bidi-start-align gs-snippet" dir="ltr">
        "XSS. Correct usage: <img src='img.png'></img>. Not so correct usage:
        <img src='
        img.png'><"
        <b>script</b>
        ">"
        <b>alert</b> == $0
        "(“hi”);</"
        <b>script</b>
        "></img>. 3&nbsp;..."
      </div>
    ▶<div class="gsc-url-bottom">…</div>
      <div class="gs-richsnippet-box" style="display: none;"></div>
```

Seems to be OK
Input is escaped



General Information

This edition will be lectured by professor João Paulo Barraca (email: jpbarraca@ua.pt). For collaboration the following methods will be used:

- **Email** for any sort of contact
- **MS Teams AEV workplace** for direct collaboration, especially during the allocated lecturing and tutoring slots. The use of the MS Teams platform for direct communication is highly recommended and should be preferred instead of email.
- **Moodle News Forum** for news related to the course.

Classes will be lectured in the Portuguese language, unless there is a foreign student attending. In this case English will be used. All content is presented in the English language.

As requirements for this subject, students should be aware that this subject requires a reasonable knowledge and comprehension of several networking, software and operating system topics, such as: the Python/C/Java languages, Linux administration and Linux console usage (mostly Debian/Ubuntu), virtual machines, sockets, HTTP and HTML, asynchronous applications, hardware architectures.

Important Dates

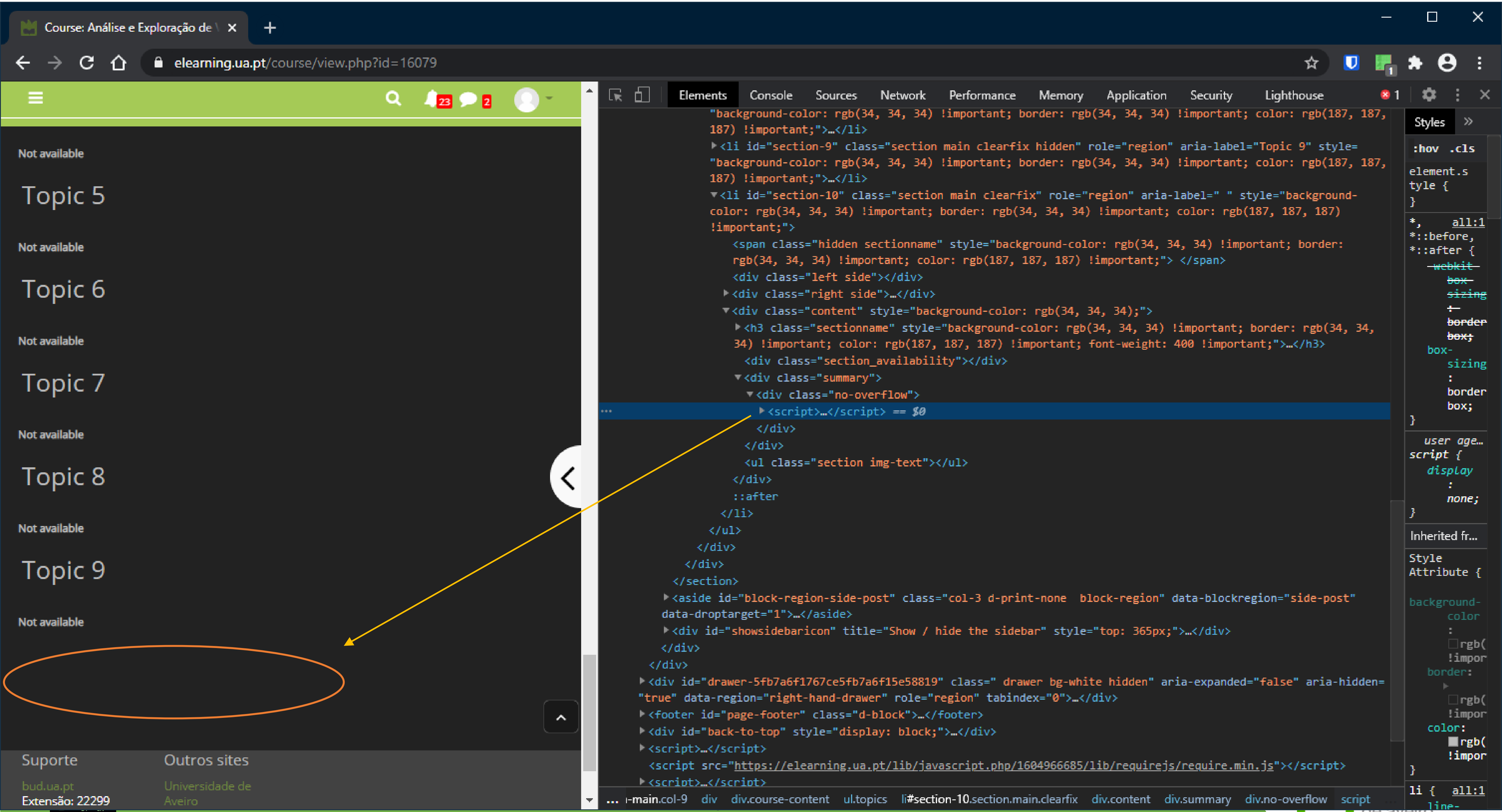
- **Assignment 1**: November 25th
- **Assignment 2**: December 21st
- **Assignment 3**: January 20th
- **Final Exam: TBD** (January)
- **HTB reports submission**: until January 18th

Elearning Areas

- Social forum
- Announcements

Theoretical Contents

Page should be light, not dark!
We are not allowed to have themes ☹



Not available

Topic 5

Not available

Topic 6

Not available

Topic 7

Not available

Topic 8

Not available

Topic 9

Not available

Suporte

bud.ua.pt
Extensão: 22299

Outros sites

Universidade de Aveiro

```
"background-color: rgb(34, 34, 34) !important; border: rgb(34, 34, 34) !important; color: rgb(187, 187,
187) !important;">…</li>
▶<li id="section-9" class="section main clearfix hidden" role="region" aria-label="Topic 9" style=
"background-color: rgb(34, 34, 34) !important; border: rgb(34, 34, 34) !important; color: rgb(187, 187,
187) !important;">…</li>
▼<li id="section-10" class="section main clearfix" role="region" aria-label=" " style="background-
color: rgb(34, 34, 34) !important; border: rgb(34, 34, 34) !important; color: rgb(187, 187, 187)
!important;">
  <span class="hidden sectionname" style="background-color: rgb(34, 34, 34) !important; border:
  rgb(34, 34, 34) !important; color: rgb(187, 187, 187) !important;"> </span>
  <div class="left side"></div>
 ▶<div class="right side">…</div>
 ▼<div class="content" style="background-color: rgb(34, 34, 34);">
   ▶<h3 class="sectionname" style="background-color: rgb(34, 34, 34) !important; border: rgb(34, 34,
   34) !important; color: rgb(187, 187, 187) !important; font-weight: 400 !important;">…</h3>
    <div class="section_availability"></div>
   ▼<div class="summary">
     ▼<div class="no-overflow">
…      ▶<script>…</script> == $0
      </div>
    </div>
    <ul class="section img-text"></ul>
  </div>
  ::after
 </li>
</ul>
</div>
</div>
</section>
▶<aside id="block-region-side-post" class="col-3 d-print-none  block-region" data-blockregion="side-post"
data-droptarget="1">…</aside>
▶<div id="showsidebaricon" title="Show / hide the sidebar" style="top: 365px;">…</div>
</div>
</div>
▶<div id="drawer-5fb7a6f1767ce5fb7a6f15e58819" class=" drawer bg-white hidden" aria-expanded="false" aria-hidden=
"true" data-region="right-hand-drawer" role="region" tabindex="0">…</div>
▶<footer id="page-footer" class="d-block">…</footer>
▶<div id="back-to-top" style="display: block;">…</div>
▶<script>…</script>
 <script src="https://elearning.ua.pt/lib/javascript.php/1604966685/lib/requirejs/require.min.js"></script>
▶<script>…</script>
```

```
… ı-main.col-9   div   div.course-content   ul.topics   li#section-10.section.main.clearfix   div.content   div.summary   div.no-overflow   script
```

```
element.style {
}
*,                    all:1
*::before,
*::after {
  ~~-webkit-box-sizing: border-box;~~
  box-sizing: border-box;
}
user age…
script {
  display: none;
}
Inherited fr…
Style Attribute {
  background-color: rgb( !impor
  border: ▶ rgb( !impor
  color: rgb( !impor
}
li {                  all:1
  line-
```

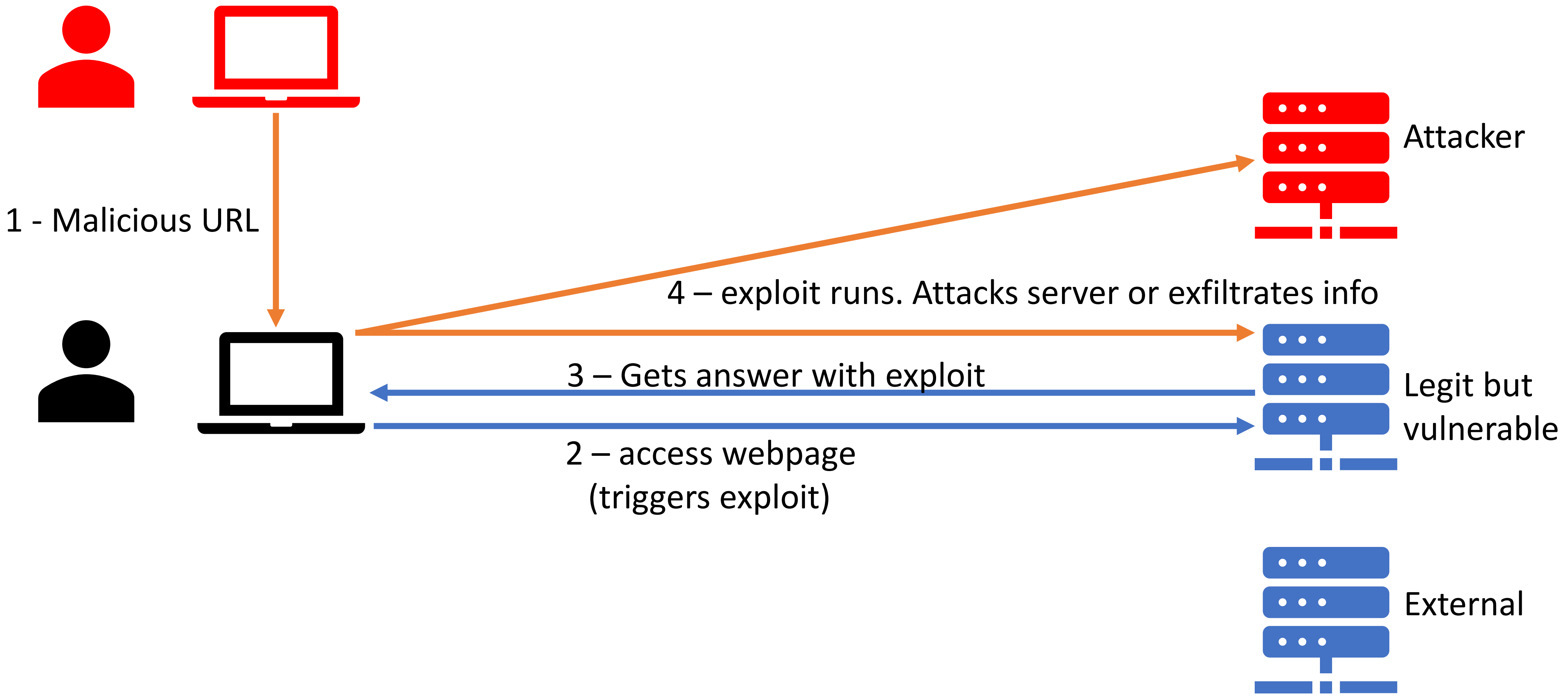
Reflected XSS

- The application or API includes unvalidated and unescaped user input as part of HTML output
  - That is: the HTML displays a string sent by the user

- The attacker will send a malicious link to the victim, pointing to an attacker-controlled page
  - Through email, posted on a chat, etc..

- A successful attack can allow the attacker to execute arbitrary HTML and JavaScript in the victim's browser

Reflected XSS


Attacker
1 - Malicious URL
4 – exploit runs. Attacks server or exfiltrates info
3 – Gets answer with exploit
Legit but vulnerable
2 – access webpage (triggers exploit)
External

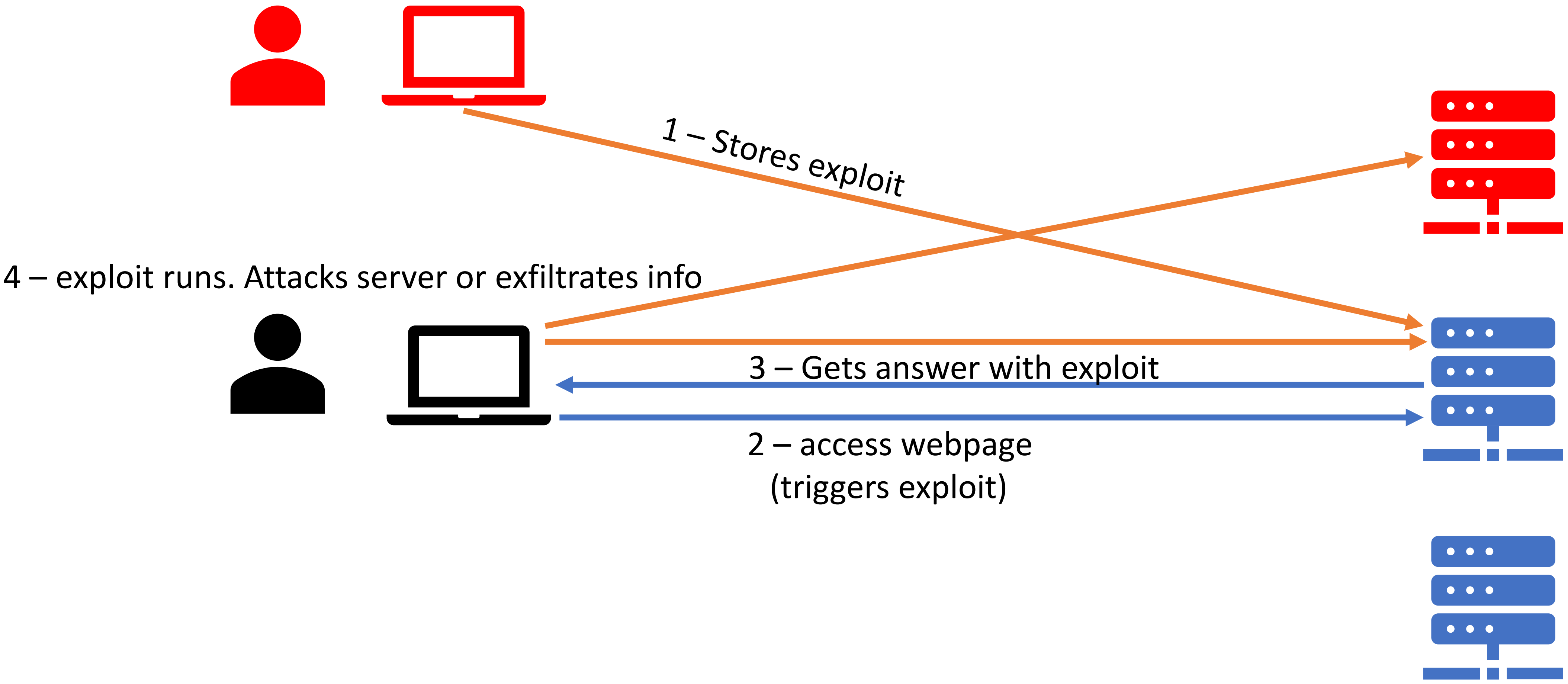
Stored XSS

- The application or API stores unsanitized user input
  - Injected by an attacker

- Input is viewed later by another user or an administrator and payload is executed

- Stored XSS is considered a high risk as actions may be executed with administrator permissions
  - When the site admin access the webpage

Stored XSS



1 – Stores exploit

4 – exploit runs. Attacks server or exfiltrates info

3 – Gets answer with exploit

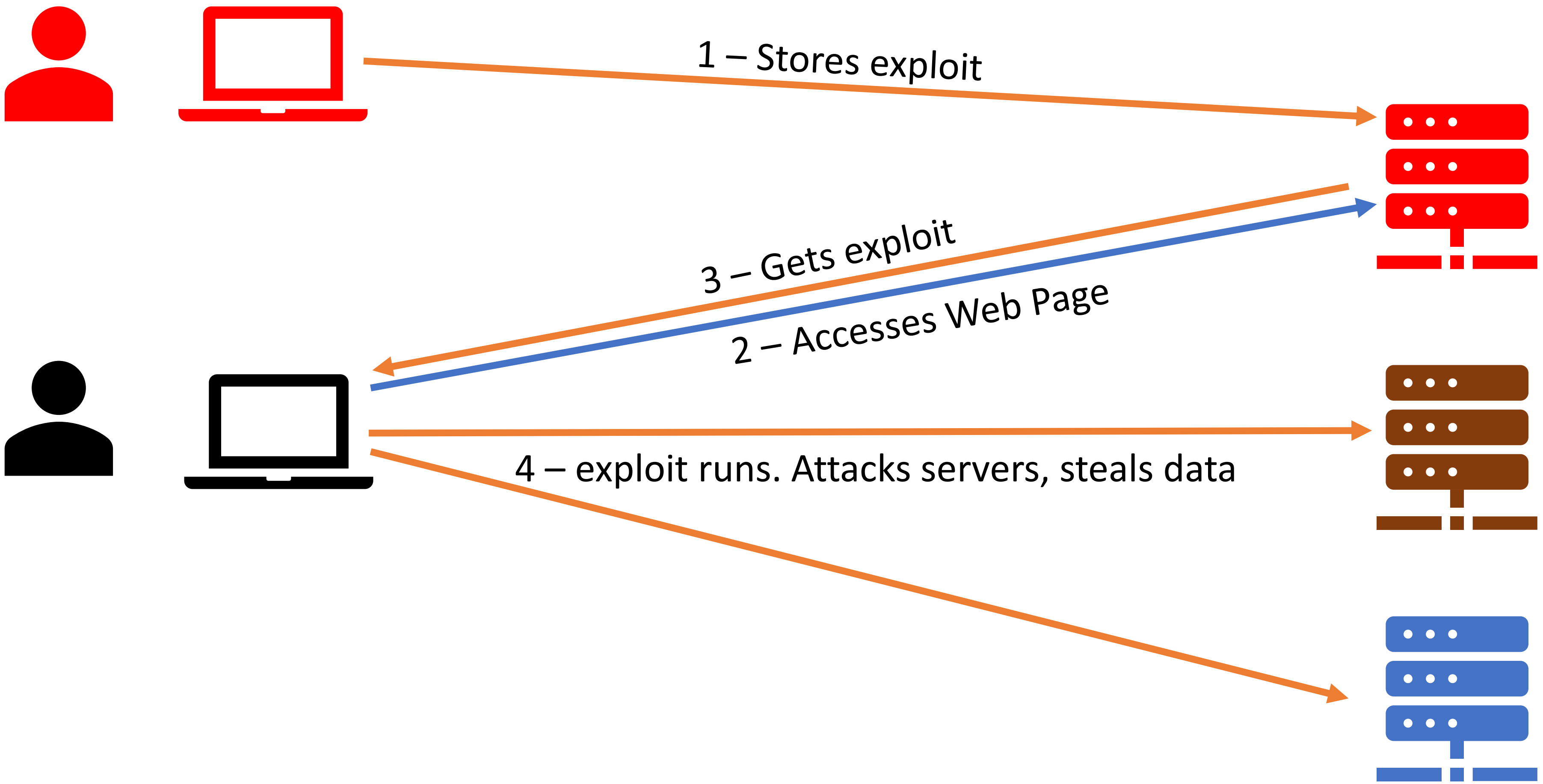
2 – access webpage
(triggers exploit)

DOM XSS

- Vulnerable apps: JS frameworks, single-page applications, and APIs that dynamically include external JS
  - Ideally, applications would not send attacker-controllable data to unsafe JavaScript APIs.

- Attacker controls remote resource (or injects resource)
  - All aspects of the client facing app may be diverted

DOM XSS

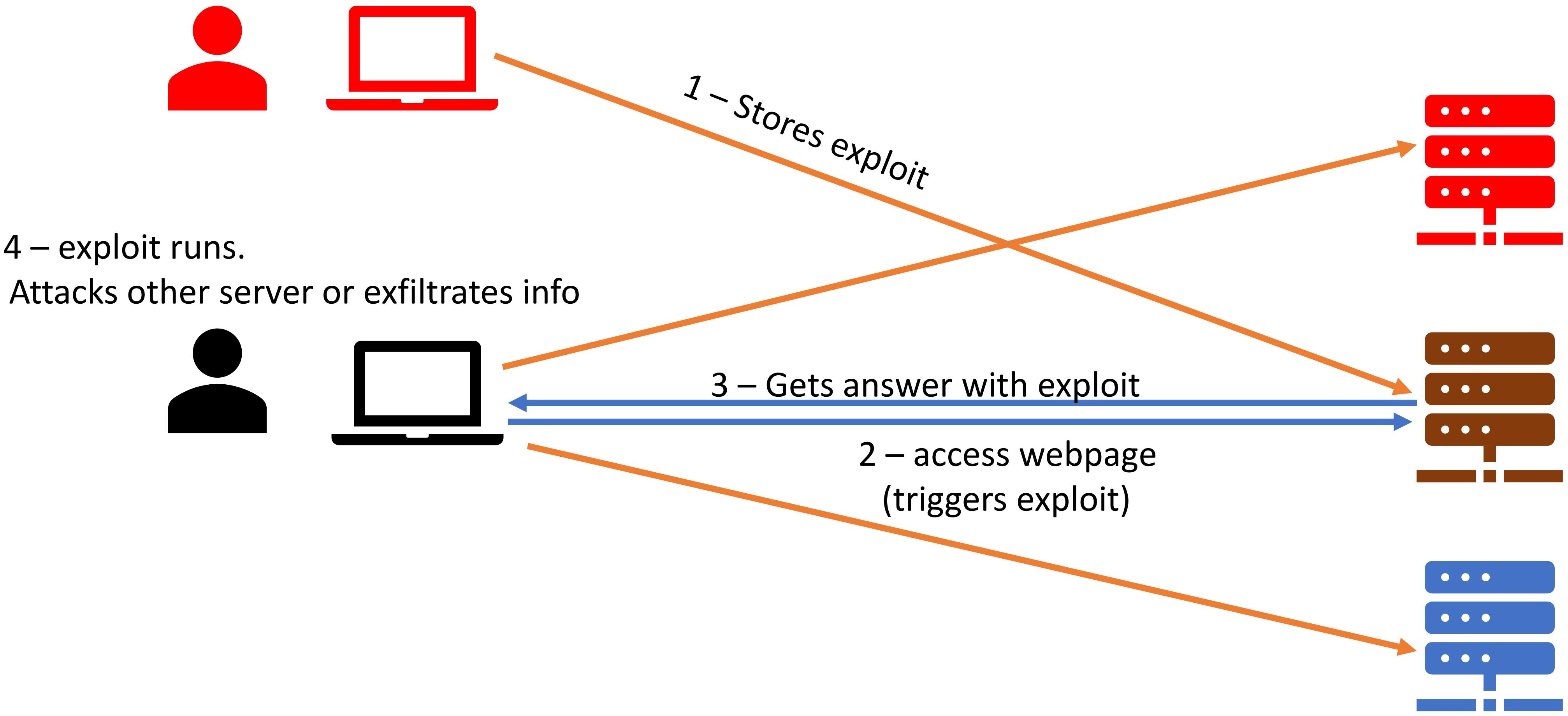

1 – Stores exploit
3 – Gets exploit
2 – Accesses Web Page
4 – exploit runs. Attacks servers, steals data

Cross Site Request Forgery

- Attacker subverts client DOM
  - Using a crafted web page
  - Using a vulnerable web page that was subverted
  - Using a XSS attack

- Client browser issues requests to external server
  - Browser will send cookies authenticating requests

Cross Site Request Forgery



Avoiding XSS: Synchronizer Tokens

- Add hidden tokens to forms so that every post requires the correct token.
  - Token is random and unique for each form
  - Server-side code verifies if the correct token is provided

- Why: If a script makes a direct POST it will not have access to the latest token

```
<form>
      <input type=“text” name=“login”></input>
      <input type=“password” name=“password”></input>
      <input type="hidden" name="csrf_token" value="KbyUmhTLMpYj7CD2di7JKP1P3qmLlkPt"/>
</form>
```

Avoiding XSS: Cookie-to-header

- Upon the establishment of a session, a cookie with a random value is provided to the client
- The JS in the Client gets the cookie and resends the cookie in the header
- Why: Assumes that only JS provided on a specific HTTPS connection may access the cookie.
  - Assumes correct browser behavior
  - The browser will not let a script called from an external source have access to external cookies
  - SameSite=Lax will only allow using cookies from same requests (GET, not POST), in a top-level operation
    - Top level operation: A click or something that changes the location

Server will set:

```
Set-Cookie: csrf_token=i8XNjC4b8KVok4uw; Expires=some_date; Max-Age=some_age; Path=/; Domain=.site.org;
SameSite=Lax; Secure
```

JS will call:

```
GET /index?csrf_token=i8XNjC4b8KVok4uw
```

Avoiding XSS: SameSite cookie attribute

- Setting the SameSite to Strict instructs browser to only provide the cookie to requests from that site
  - Similar to Lax, but without exceptions to safe requests

- Why: If the SameSite is set, an external script will not have access to the token

Server sets:

```
Set-Cookie: csrf_token=i8XNjC4b8KVok4uw; Expires=some_date; Max-Age=some_time;
Path=/; Domain=.wikipedia.org; SameSite=Strict; Secure
```

Legit JS will have access to the cookie, External JS won’t

Avoiding XSS: Double cookie submission

- Two cookies are used
  - Session Cookie: identifies the user, stable across the session duration
  - CSRF cookie: dynamically changing for each request

- Why: External requests will not have information about the last CSRF cookie
  - May allow sites to force a specific interaction sequence as CSRF cookies may identify the previous location

Same Origin Policy

- Sites may require external resources (Cross Origin Resources)
  - Javascripts, Images, Styles
  - However this should be controlled

- Current site perspective: where my resources are being loaded from?
  - Images may be remote, JS should be local, as well as styles…

- Other sites: who is accessing my resources?
  - I do not want to be spreading malware (act as a storage for Stored XSS)

Same Origin Policy

- Web servers may state a header that sets the Same Origin Policy
- What is Same Origin Policy?
  - SOP restricts how a document or script loaded from one origin can interact with a resource from another origin
- define: origin. In relation to http://store.comp.com/dir/page.html
  - http://store.comp.com/dir2/other.html, Success
  - http://store. comp.com/dir/inner/another.html Success
  - https://store. comp.com/secure.html, Failure - Different protocol
  - http://store. comp.com:81/dir/etc.html, Failure - Different port
  - http://news. comp.com/dir/other.html, Failure Different host

Same Origin Policy

- Origin is permitted to send data to another origin but not read
- Interactions between origins are placed in three categories:
  - Cross origin writes (redirects, links, form action etc.)
  - Cross origin embedding (html tag with src/hrefs)
  - Cross origin reads (not allowed without CORS etc.)

Same Origin Policy

Cross Origin Embedding

- JavaScript <script src="..."></script>.
- CSS with <link rel="stylesheet" href="...">.
- Images with <img>.
- Media files with <video> and <audio> tags.
- Plug-ins with <object>, <embed> and <applet>.
- Fonts with @font-face.
- Anything with <frame> and <iframe>.

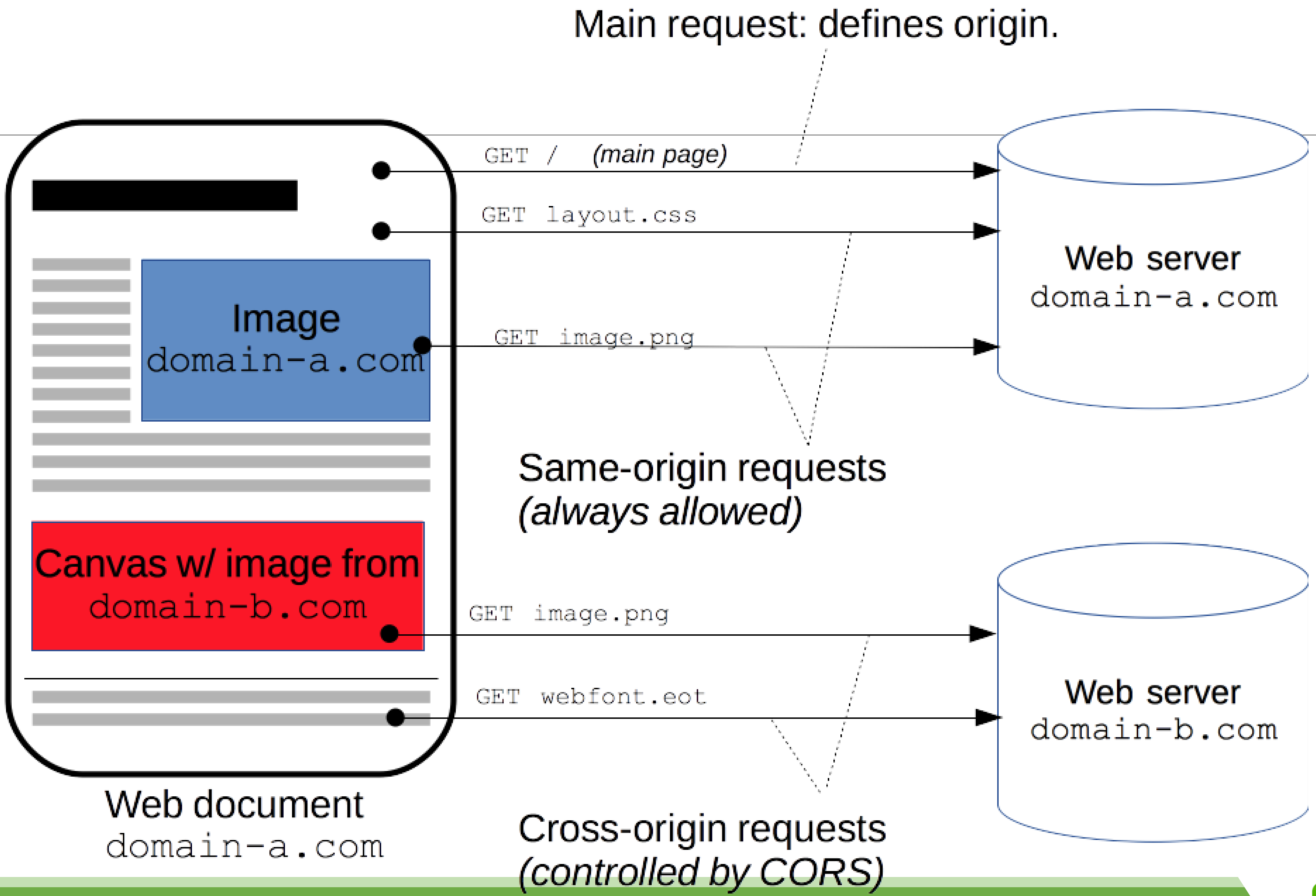
Cross Origin Request Sharing

Why is CORS needed?

- For legitimate and trusted requests to gain access to authorized data from other domains
  - Think cross application data sharing models

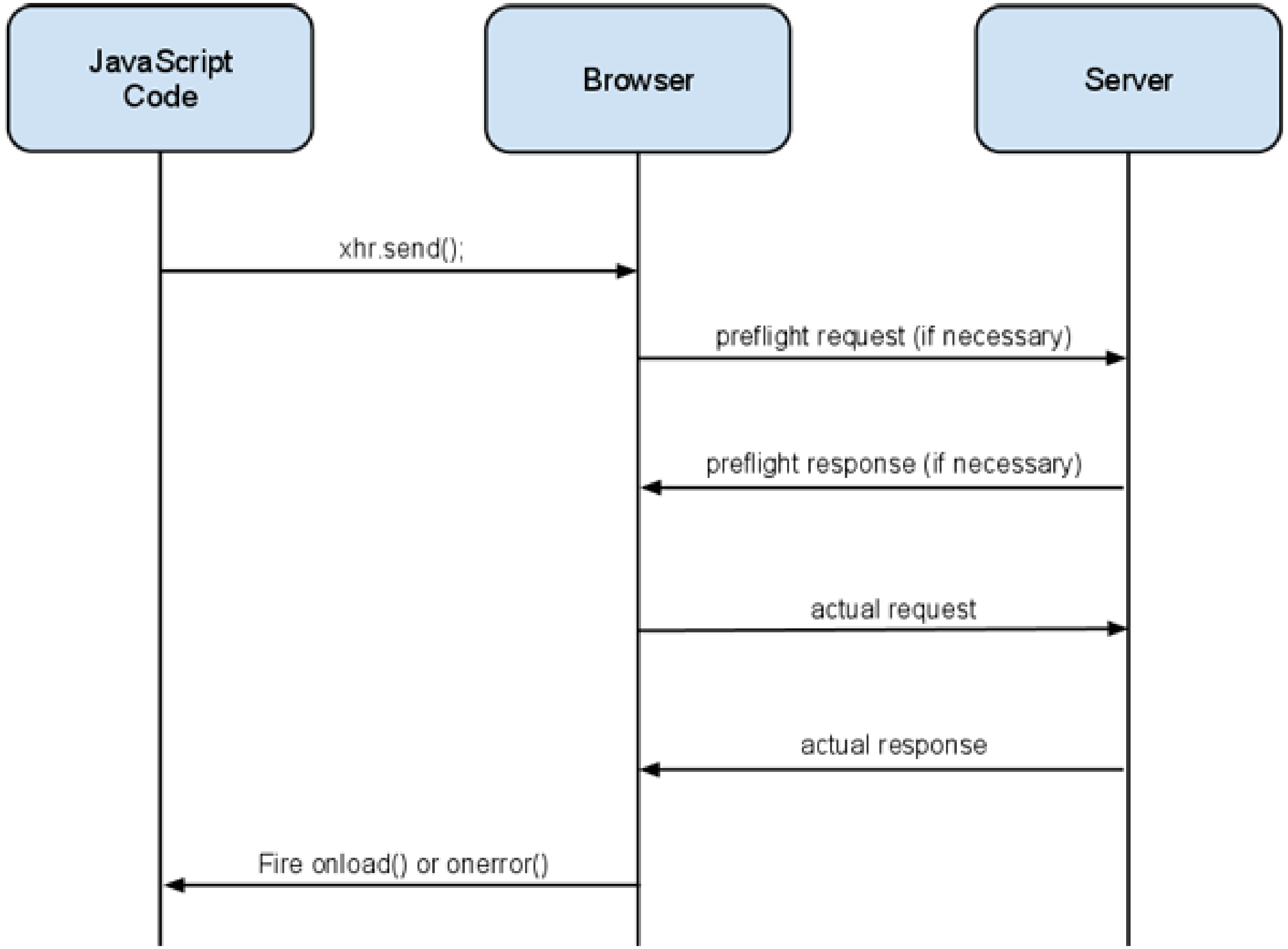
- Allows data to be exchanged with trusted sites while using a relaxed Same Origin policy mode.

- Application APIs exposed via web services and trusted domains require CORS to be accessible over the SOP


Main request: defines origin.

Same-origin requests *(always allowed)*

Cross-origin requests *(controlled by CORS)*


JavaScript Code
Browser
Server
xhr.send();
preflight request (if necessary)
preflight response (if necessary)
actual request
actual response
Fire onload() or onerror()

CORS Requests

- Preflight is not needed if
  - Request is a HEAD/GET/POST via XHR
  - No Custom headers
  - Body is text/plain
- Server responds with a CORS header
  - Browser determines access
  - Neither the request, nor response contain cookies

CORS Headers

- Simple Request
  - Origin: Header set by the client for every CORS request
  - Value is the current domain that made the request

- Access-Control-Allow-Origin
  - Set by the server and used by the browser to determine if the response is to be allowed or not.
  - Can be set to * to make resources public (bad practice!)

CORS Insecurity

- Several security issues arise from the improper implementation of CORS, most commonly using a universal allow notation (*) in the server headers
- Clients should not trust the received content completely and eval or render content without sanitization which could result in misplaced trust
- The application that allows CORS may become vulnerable to CSRF attacks

CORS Insecurity

- Prolonged caching of Preflight responses could lead to attacks arising out of abuse of the Preflight Client Cache

- Access control decisions based on the Origin header could result in vulnerabilities as this can be spoofed by an attacker

CORS Insecurity: Misplaced Trust

- Data exchange between two domains is based on trust
  - If one of the servers involved in the exchange of data is compromised, then the model of CORS is put at risk

- Scenarios?
  - An attacker can compromise site A and host malicious content, knowing site B trusts the data that site A sends to site B.

CORS Insecurity: Access Control based on Origin

- The Origin header indicates that the request is from a particular domain, but does not guarantee it
  - Header can be controlled by the attacker

- Spoofing the Origin header allows access to the page if access is based on this header

- Scenarios?
  - An attacker sets the Origin header to view sensitive information that is restricted
  - Using cURL to set a custom origin header: curl --header 'origin:http://someserver.com'

CORS Insecurity: Caching of Preflight responses

- The Access-Control-Max-Age header is set to a high value, allowing browsers to cache Preflight responses
  - It’s very important for performance reasons
  - But caching the preflight response for longer duration can pose a security risk.

- If the access-control policy is changed on the server the browser would still follow the old policy available in the Preflight Result Cache

- Scenario:
  - During updates to sites, the access policy will be out of sync until the cache expires

CORS Insecurity: Universal Allow

- Setting the 'Access-Control-Allow-Origin' header to *
  - Effectively turns the content into a public resource, allowing access from any domain
  - Very common during development, and somewhat during production
- Scenarios?
  - An attacker can steal data from an intranet site that has set this header to * by enticing a user to visit an attacker controlled site on the Internet.
  - An attacker can perform attacks on other remote apps via a victim’s browser when the victim navigates to an attacker controlled site.