

# Nizar Session

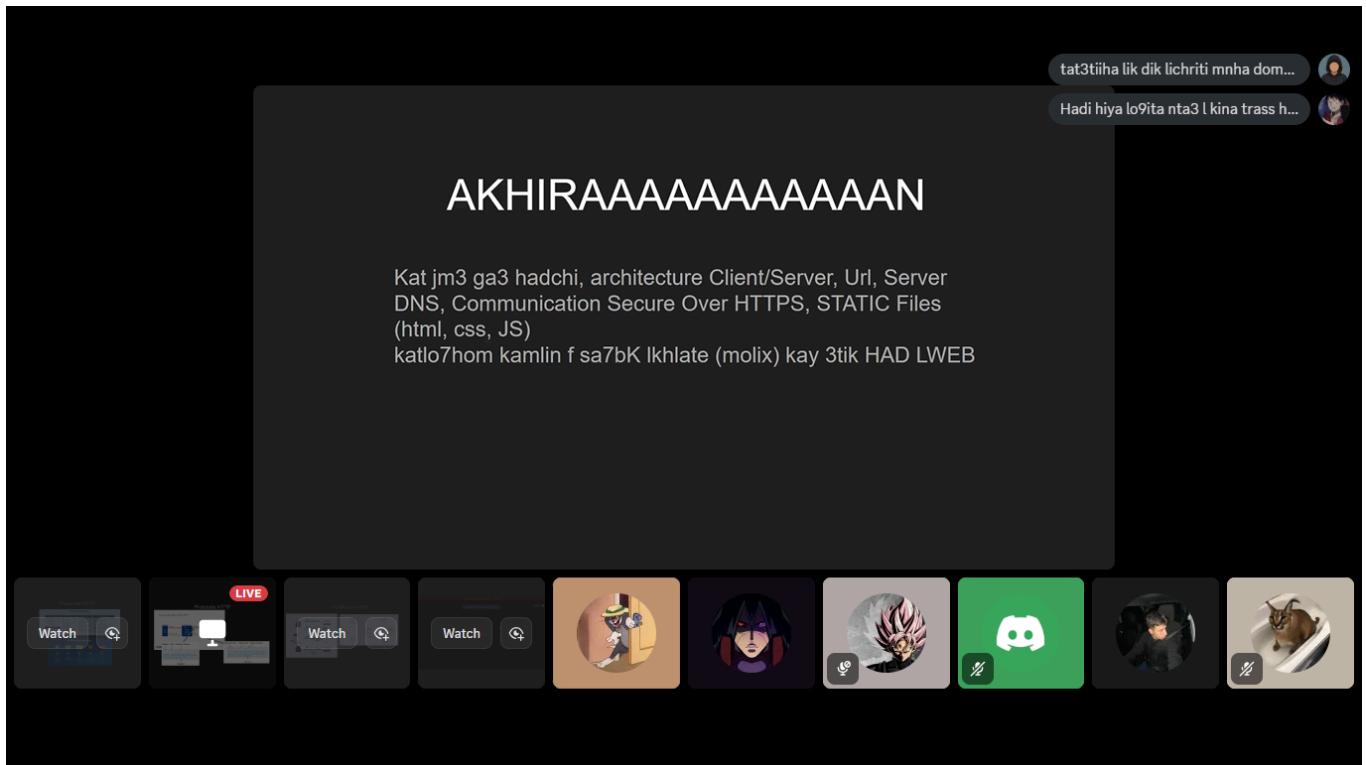
## DNS

- In a URL, we have TLD (Domaine de premier niveau) (Top Level domain), it is important. (It is usually the final part of the url (.org, .ma, .com, ...))
- You can see all DNS with this website : <https://securitytrails.com/>

## HTTPS

- Fun Fact, HTTPS = HTTP + SSL Encryption.
- SSL is used with a certification, SSL certification.

## In general



## OSINT

### - Google dorking,

also known as Google hacking, is a technique that uses advanced Google Search queries to find security vulnerabilities in websites. It involves using specific keywords and operators to

locate exposed information or misconfigured systems, such as finding sensitive files or login pages that were not intended to be public.

The screenshot shows a video call interface with a presentation slide titled "Subdomain Enumeration". The slide contains text and links for finding subdomains:

- you can use a website called <https://crt.sh> is a Certificate Transparency (CT) log search tool that allows users to find SSL/TLS certificates issued for specific domains or organizations
- you can <https://securitytrails.com/> for full DNS records
- you can also use brute forcing tools like **ffuf**
- you can use [archive.org](https://archive.org) to look for old pages that may hide some subdomains
- or you can use **Google Dorking**
- you can find network range that is used by the organization and start finding alive hosts in that range and do the reverse ip lookup

The video call interface includes a toolbar at the bottom with various icons and a list of participants on the left.

It's here where we use the Authentification ByPass

## - File Inclusion:

The screenshot shows a video call interface with a presentation slide titled "File Inclusion". The slide contains text about file inclusion vulnerabilities:

- well there is two types of this file inclusion vulnerability
- there is RFI and LFI
- RFI : remote file inclusion this envolve to call an remote file from another server by the backend and presented to the user may cause RCE in php language and XSS in some other cases
- LFI this is : local file inclusion this one envolve calling internal file from the server and present it to the user

Lab LFI : <http://challenge01.root-me.org/web-serveur/ch16/>

The video call interface includes a toolbar at the bottom with various icons and a list of participants on the left.

RFI and LFI are both **file inclusion vulnerabilities** in web applications that occur due to improper input validation, but they differ in the **location of the files** an attacker can include.

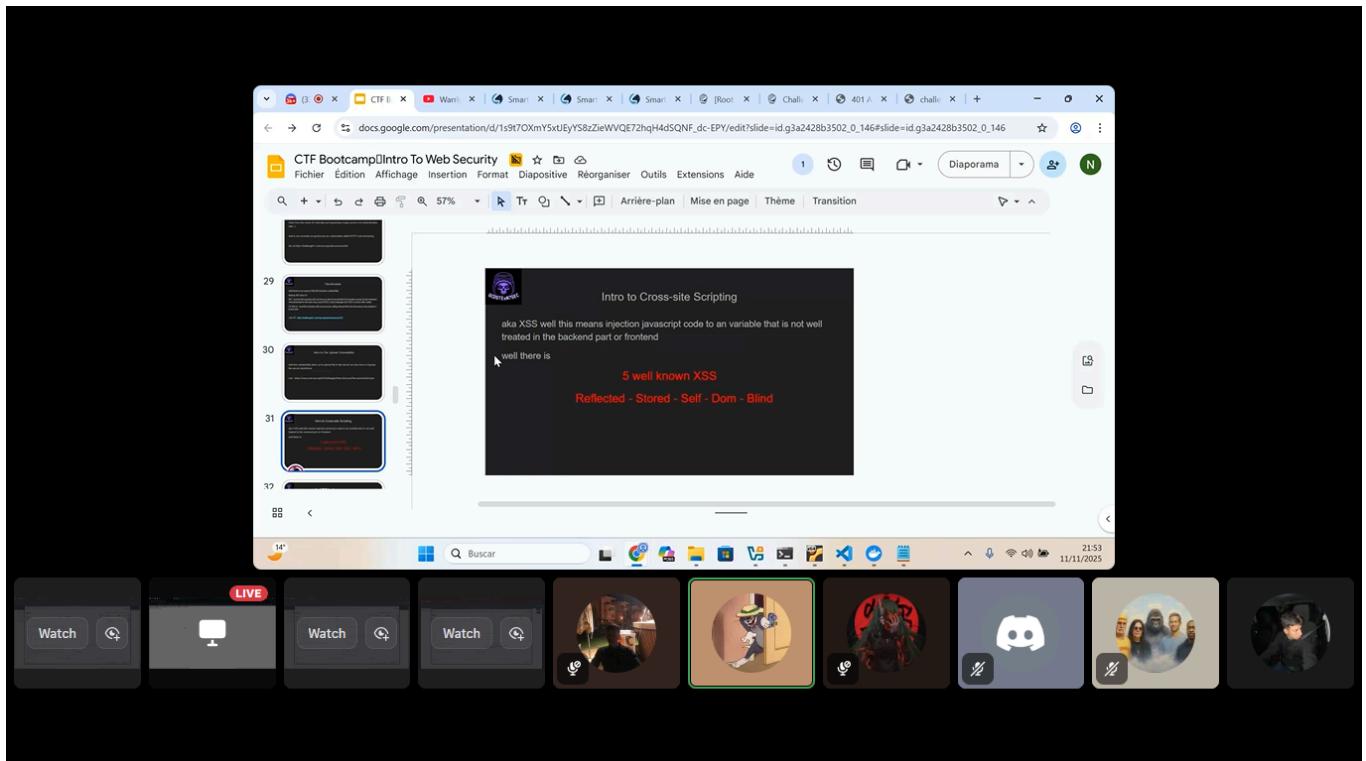
## RFI: Remote File Inclusion

- **Definition:** RFI allows an attacker to include files hosted on a **remote server** (an external source controlled by the attacker) into the vulnerable web application.
- **Mechanism:** The web application fetches and executes the attacker's remote malicious code, typically specified via a URL in a user-supplied input parameter.
- **Severity:** RFI is generally considered more severe than LFI because it can immediately lead to **remote code execution (RCE)** on the target server, allowing the attacker to run arbitrary commands and potentially compromise the entire system.
- **Prerequisites:** A successful RFI attack often requires specific server configurations, such as the `allow_url_include` setting being enabled in PHP.

## LFI: Local File Inclusion

- **Definition:** LFI allows an attacker to include and access files that are already present on the **local server** where the application is running.
- **Mechanism:** Attackers manipulate file paths using techniques like directory traversal (e.g., `../..`) in user input to access sensitive local files (like configuration files, source code, or system logs) that were not intended to be exposed.
- **Severity:** While LFI primarily leads to **sensitive information disclosure**, it can sometimes be escalated to RCE if the attacker can find a way to place controlled data (like through a log file or an insecure upload function) on the server and then include it.

## Cross-site scripting (XSS):



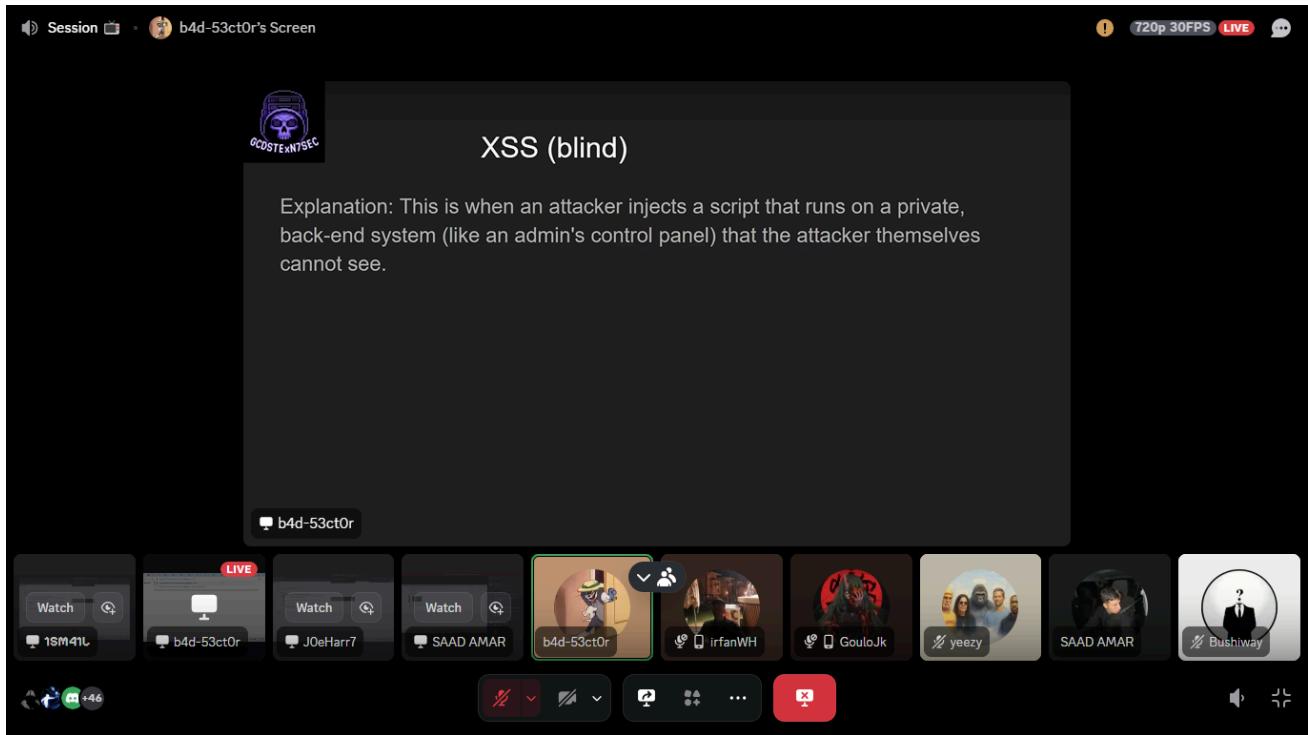
The screenshot shows a video player interface with a dark theme. At the top left is a logo featuring a skull wearing a beret and the text 'OCOSTEXTSEC'. The main title 'why XSS is dangerous' is centered above five bullet points. Below the title is a horizontal row of nine video thumbnail previews.

- Stealing Information: Attackers can steal any information your users type into the site. This includes usernames, passwords, credit card numbers, and personal details like names and addresses.
- Taking Over User Accounts: An attacker can "hijack" a user's login session. This means they can act as that user, change their password, access their private messages, make purchases, or transfer money without the user ever knowing.
- Spreading Malware: The attacker can use your website to show pop-ups that trick users into downloading viruses, spyware, or ransomware onto their own computers.
- Tricking Your Users (Phishing): An attacker can change parts of your website to show a fake login form. When a user tries to log in, their password is sent directly to the attacker.
- Changing Your Website (Defacement): Attackers can change what your website looks like. They could add offensive content, delete pages, or post their own messages, making your site look unprofessional or broken.

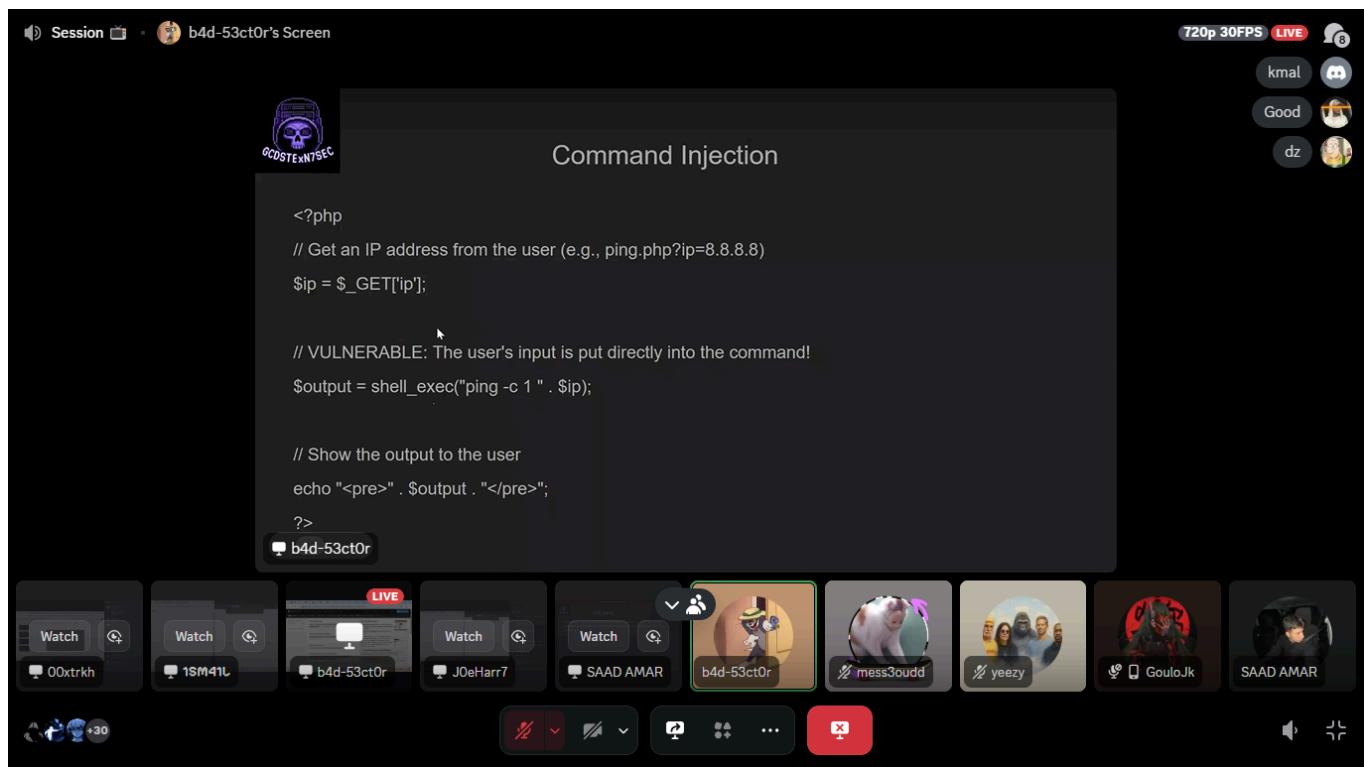
- It can be used to steal cookies and stuff

The screenshot shows a video player interface with a dark theme. At the top left is a logo featuring a skull wearing a beret and the text 'OCOSTEXTSEC'. The main title 'XSS (Stored)' is centered above three bullet points. A message from a viewer named 'z3ma ba9i had xss' is visible in the top right corner. Below the title is a horizontal row of nine video thumbnail previews.

- The attack is permanently saved on the website for all visitors to see.
- How it works: An attacker posts a malicious comment on a blog post or a product review: "Nice article! <script>stealEveryoneCookies()</script>"
- What happens: The website saves this dangerous comment to its database. Now, every single person (including admins) who views that blog post will have the malicious script run in their browser, potentially stealing everyone's login session.



## - Command Injection



## - SQL Injection

Session b4d-53ct0r's Screen

720p 30FPS LIVE

## SQL Injection

```
<?php  
// Get user input (BAD: not sanitized)  
$user = $_POST['username'];  
$pass = $_POST['password'];  
  
// VULNERABLE: Building the query by concatenating strings  
$sql = "SELECT * FROM users WHERE username = '$user' AND password = '$pass'";  
// ...then the code runs this $sql query against the database...  
?>
```

b4d-53ct0r

Watch 00xtrkh Watch 1SM41L Watch LIVE b4d-53ct0r Watch J0eHarr7 Watch SAAD AMAR Watch MR.KATANA Watch Mr.katana Watch mess3oudd Watch GouloJk Watch SAAD AMAR

Showing rows 0 - 0 (1 total, Query took 0.0011 seconds.)

```
SELECT * FROM users WHERE username='kiki' union select null,"nono" as username,"fofo" as password , null;
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all Number of rows: 25 Filter rows: Search this table

Extra options

id	username	password	reg_date
NULL	nono	fofo	NULL

- [onlinephp.io/password-hash](http://onlinephp.io/password-hash)