HIRSCH Mateo
ECOTIERE Léo
Groupe 8

Etude comparative de solutions logicielles

1. Contextualisation

Date de l'étude	**27/11/2023**
Condition économiques	**Novembre 2023**
Sujet étudié	**IDS/IPS logiciel**
Cadre d'utilisation	**Supervision d'un réseau hospitalier**

2. Solution retenue

Notre choix final **s'est porté sur Snort** !
Nous avons choisi cet utilitaire car c'est celui qui répond le plus de manière positive aux comparaisons effectuées. De plus, Snort est un outil extrêmement polyvalent de par la définition de ses règles : il est possible de construire des règles "sur-mesure" pour des besoins très spécifiques.
Ses performances sont aussi dans les meilleurs du marché étudié, Suricata présente l'avantage de l'utilisation des ressources graphiques mais ce n'est pas nécessaire à l'échelle actuelle.
L'installation et la configuration prennent environ 20 minutes, le rendant disponible rapidement. Pour finir, il dispose d'une communautée très active : des réponses aux questions dans la journée.

3. Différence entre IPS et IDS

Un **I**ntrusion **D**etection **S**ystem (**IDS**) permet de repérer des activités anormales sur un réseau TCP/IP ou un hôte et envoie des alertes en fonction des paramètres configurés.

Un **I**ntrusion **P**revention **S**ystem (**IPS**) permet de repérer des activités anormales sur un réseau TCP/IP ou un hôte et agit en conséquence des paramètres configurés. Il peut par exemple bloquer un trafic ou éteindre un serveur ce que ne peut pas faire un IDS.

	IDS	**IPS**
Analyse	✔	✔
Alerte	✔	✔
Bloque le trafic	✗	✔
Se base sur une BDD	✔	✔
Effectue des recherches avec des plugins	✗	✔
Solution retenue	✗	✔

Nous avons **fait le choix de l'IPS** car il est plus complet et permet une analyse en profondeur des paquets (plugins) offrant donc une plus grande sécurité. De plus, l'IPS permet le blocage immédiat de paquets frauduleux, ce qui en fait une sorte *"d'IDS amélioré"*. Son installation et sa configuration n'étant pas beaucoup plus longues

4. Différence entre NIPS et HIPS

Un **N**etwork **I**ntrusion **P**revention **S**ystem (**NIPS**) est un IPS dédié à la supervision du réseau et l'analyse de paquets transitants.
Un **H**osts **I**ntrusion **P**revention **S**ystem (**HIPS**) est un IPS dédié à la supervision des machines, mettant l'accent sur la modification et suppression de données au sein d'une machine.

	NIPS	**HIPS**
Analyse le trafic réseau	✔	✗
Analyse les suppressions/modifications de fichiers	✗	✔
Solution retenue	✔	✗

Notre choix **s'est porté sur le NIPS** car étant donné que le réseau sera cloisonné, il est plus pertinent de superviser les différents réseaux et leurs flux plutôt que les machines hôtes. Une **supervision des hôtes** sera effectuée à l'aide du protocole **SNMP**.

5. Comparatif d'IPS

	Security Onion	Zabbix	Zeek	Snort	Suricata
Open Source	✔	✔	✔	✔	✔
IPS	✔	✗	✗	✔	✔
Super-compatibilité	✔	✔	?	✔	?
Capacité d'évolution	✔	✔	?	✔	✗
Optimisé réseau	✗	✗	✔	✔	✗
Installation et activation	~20 min	~20 min	~10 min	~15 min	~10 min
Configuration de base	~15 min	~15 min	~10 min	~10 min	~25 min
Communauté active	Quotidienne	Quotidienne	Hebdomadaire	Quotidienne	Quotidienne
Performance (en tant qu'IPS)	< Suricata. Il utilise suricata comme IPS mais ajoute une surcouche → moins performant	✗	✗	Multithreading 750 Mbit/s (max)	Multithreading 739 Mbit/s (max) Utilisation ressource GPU + CPU Hyperscan*
Type de détection	Signature + BDD	✗	✗	Signature + BDD	Signature + BDD
License	CC BY 4.0	GNU GPL	BSD license	GNU GPL NCUL	GNU GPL
Solution retenue	✗	✗	✗	✔	✗

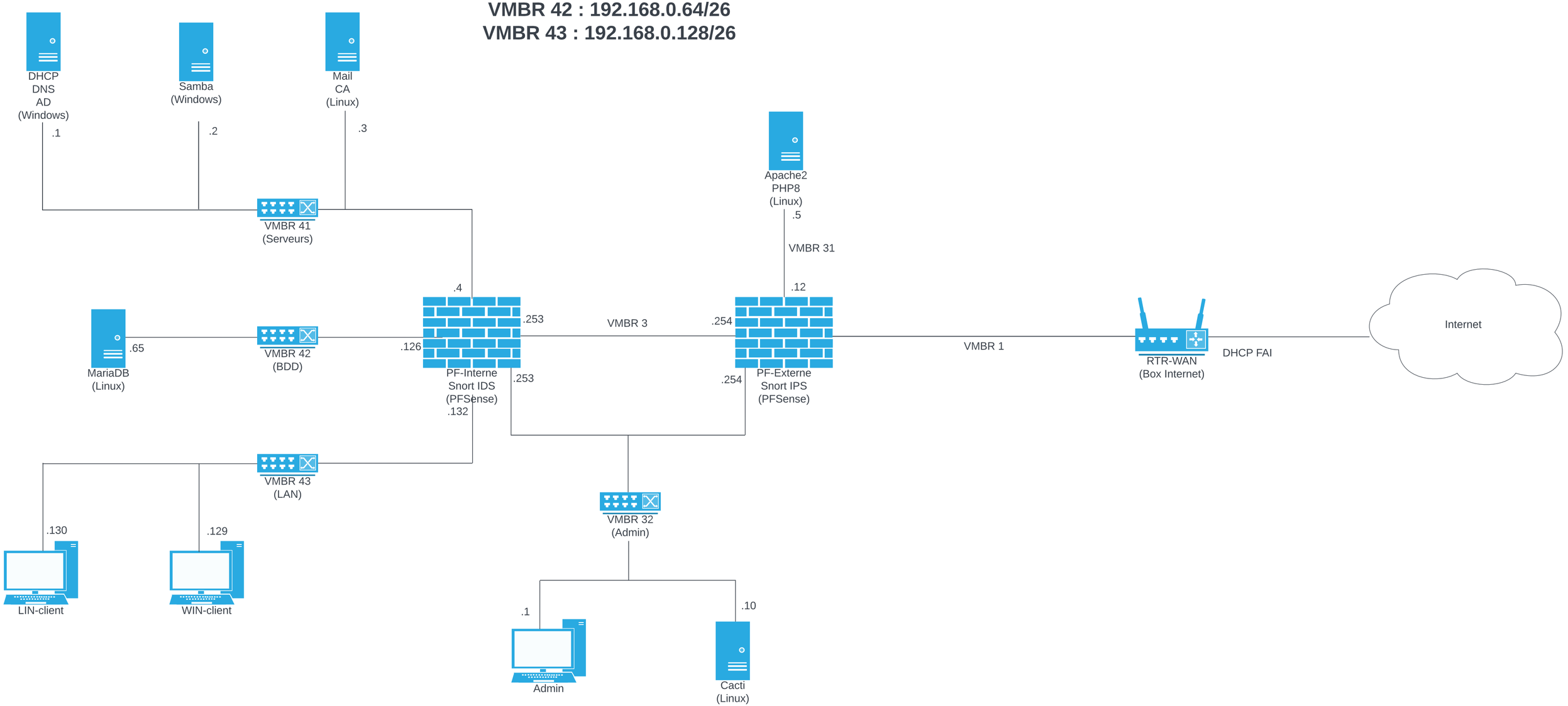
Lexique :

- **GNU GPL** : GNU's Not Unix General Public License (libre de d'utilisation et de modification)
- **BSD License** : Berkeley Software Distribution License (libre de d'utilisation et de modification)
- **CC BY 4.0** : Utilisation et modifications libres avec crédits
- **NCUL** : Non-Commercial Use License (commercialisation des codes sources et règles interdite)
- **Hyperscan** : Plugin disponible pour Suricata permettant l'utilisation de règles regex à très haute performance.

Sources :

- *Site officiel Zabbix*
- *Site officiel Security Onion*
- *Site officiel Zeek*
- *Site officiel Snort*
- *Site officiel Suricata*
- *White Paper de M. Jonathan H. Stammler (CISO) sur les performances de Suricata*
- *Etude académique de Mme. Alka Gupta & M. Lalit Sen Sharma (University of Jammu) sur les performances de Snort*

VMBR 3 : 192.168.2.252/30
VMBR 31 : 192.168.1.4/28
VMBR 32 : 192.168.100.0/24
VMBR 41 : 192.168.0.0/26
VMBR 42 : 192.168.0.64/26
VMBR 43 : 192.168.0.128/26



HIRSCH Mateo
ECOTIERE Léo
Groupe 8

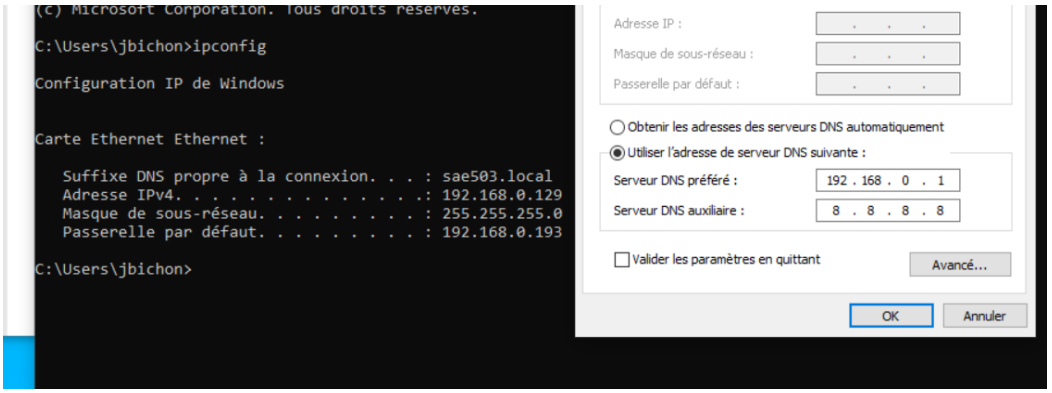
Rapport de test

des fonctionnalités services et réseaux du projet

Système	Description test	Résultat attendu	Résultat obtenu (Voir ANNEXES)
Dynamic Host Configuration Protocol (DHCP)	Récupération d'adresse IPv4 via DHCP sur les 2 clients du LAN	Adresse IP dans le réseau 192.168.0.128/26 avec les réservation attribuées (129 pour Windows & 130 pour Linux)	Check
Domain Name Service (DNS)	Réalisation d'un ping sur un nom de serveur	Possibilité de ping win-srv-1.sae503.local	Check
SaMBa (SMB)	Connexion sur les comptes utilisateurs de l'AD sur les clients Linux & Windows et accès au répertoire partagé (Windows) et personnel (Linux)	Montage automatique des lecteurs sur un point de montage défini LINUX /home/SAE503/jbichon/partage & /home/SAE503/jbichon/partage WINDOWS Z:	Check
Active Directory (AD)	- Connexion au domaine des clients Linux & Windows. - Connexion avec les utilisateurs déclarés dans l'AD	Validation de la connexion sur les clients avec les utilisateurs déclarés dans l'AD	Check
Linux Apache MySQL PHP (LAMP)	- Accès au site web en local / distant et à travers le pare-feu (NAT) - Ajout de données dans la BDD depuis le serveur Web - Utilisation de fonctions PHP	- Page Web accessible - Ajout des données confirmé - Pas d'erreurs/warnings php	Check

ANNEXES

I. Serveur Dynamic Host Configuration Protocol (DHCP)

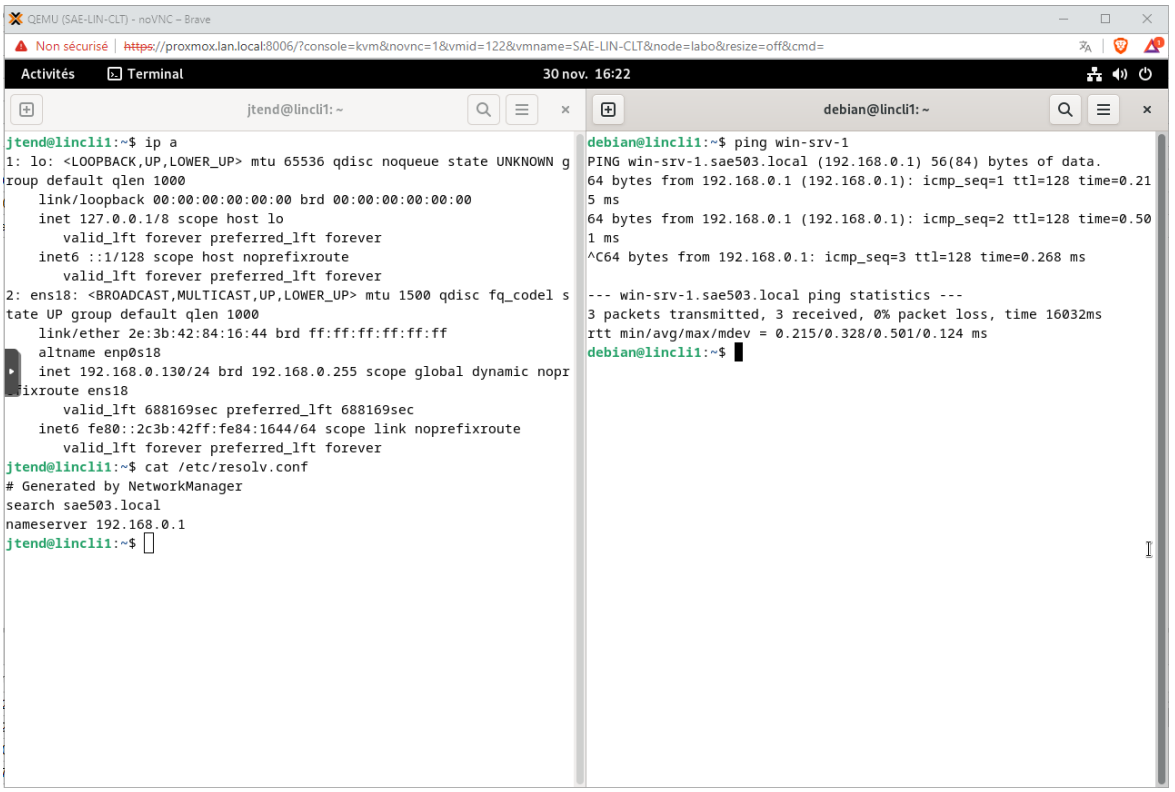


Adresse IPv4 et configuration DHCP sur le client Windows

```
jtend@lincli1:~$ ip a && cat /etc/network/i
if-down.d/      if-post-down.d/ if-pre-up.d/    if-up.d/
jtend@lincli1:~$ ip a && cat /etc/network/interfaces
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNK
    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: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_c
    link/ether 2e:3b:42:84:16:44 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.0.130/24 brd 192.168.0.255 scope global dynami
       valid_lft 687999sec preferred_lft 687999sec
    inet6 fe80::2c3b:42ff:fe84:1644/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
# This file describes the network interfaces available on your
# and how to activate them. For more information, see interface

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto ens18
iface ens18 inet dhcp
jtend@lincli1:~$
```

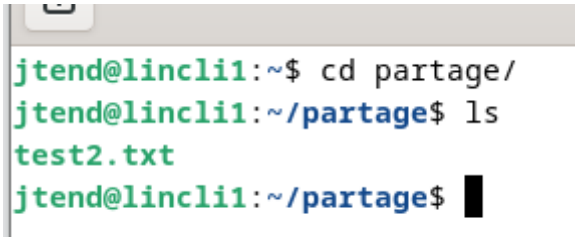
Adresse IPv4 et configuration DHCP sur le client Linux

II. Serveur Domain Name Service (DNS)

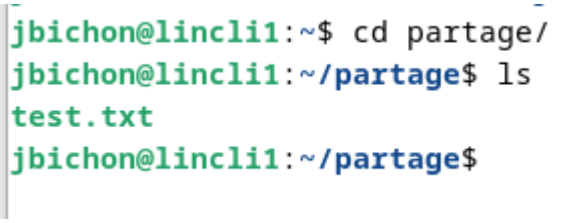


Configuration IPv4 & DNS & ping sur le serveur Windows 1 depuis le client Linux

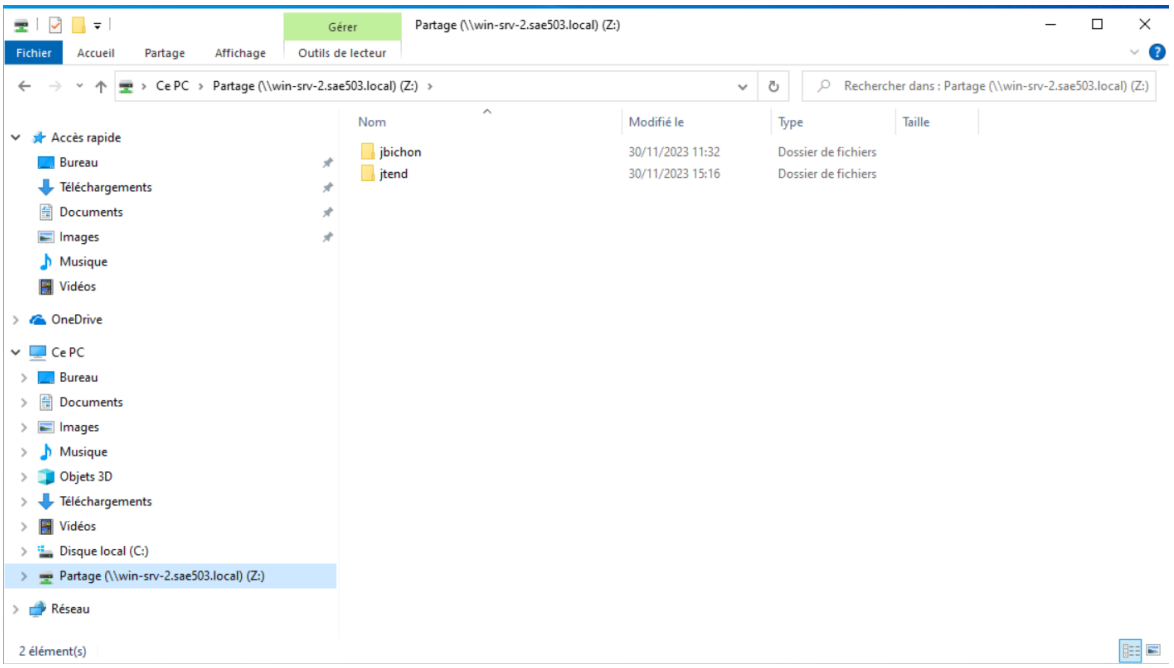
III. Serveur SaMBa (SMB)



Accès au partage personnel depuis Jean Tend sur le client Linux



Accès au partage personnel depuis Joseph Bichon sur le client Linux

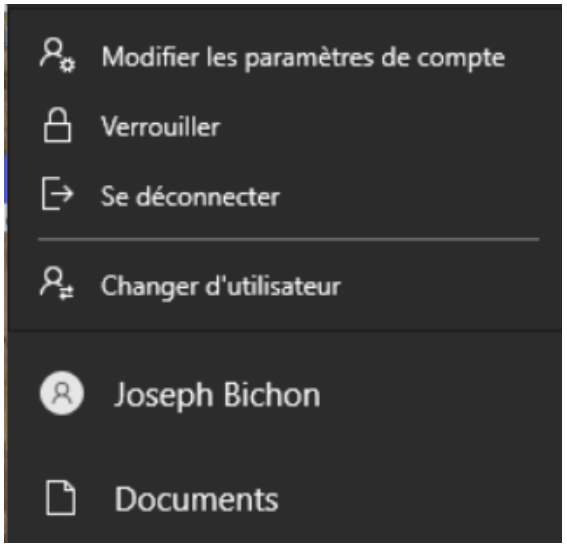


Accès au partage commun depuis Joseph Bichon sur le client windows

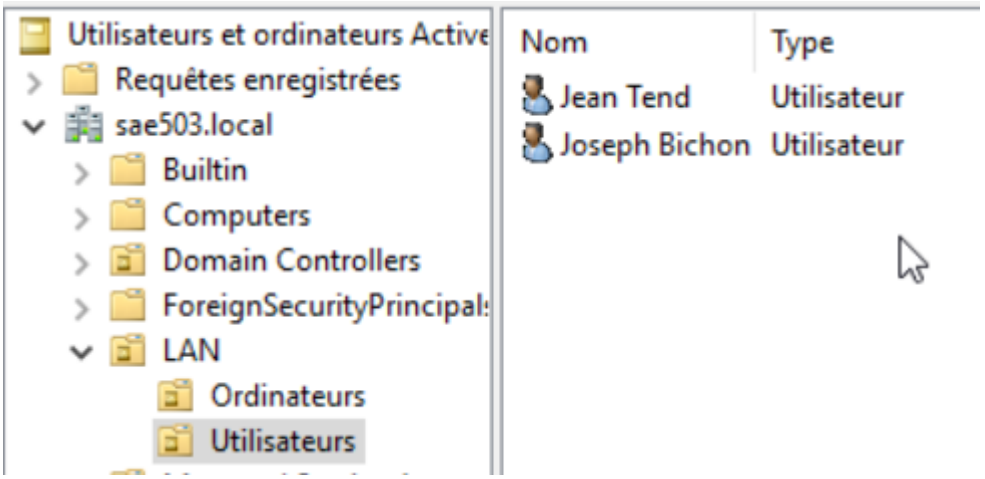
IV. Serveur Active Directory (AD)

```
debian@lincli1:~$ cat /etc/passwd | grep jbichon
debian@lincli1:~$ su jbichon
Password:
jbichon@lincli1:/home/debian$ cd
jbichon@lincli1:~$ ls
Bureau     Images   Musique  Public           Vidéos
Documents  Modèles  partage  Téléchargements
jbichon@lincli1:~$ pwd
/home/SAE503/jbichon
jbichon@lincli1:~$
```

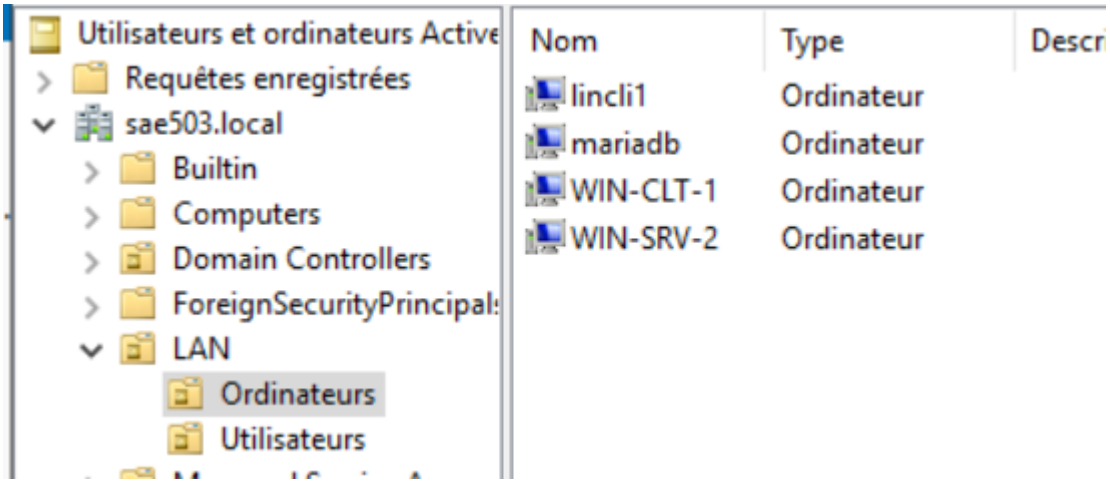
Login en temps que Joseph Bichon sur le client Linux via l'Active Directory (Absence de Joseph Bichon en local)



Login en temps que Joseph Bichon sur le client Windows

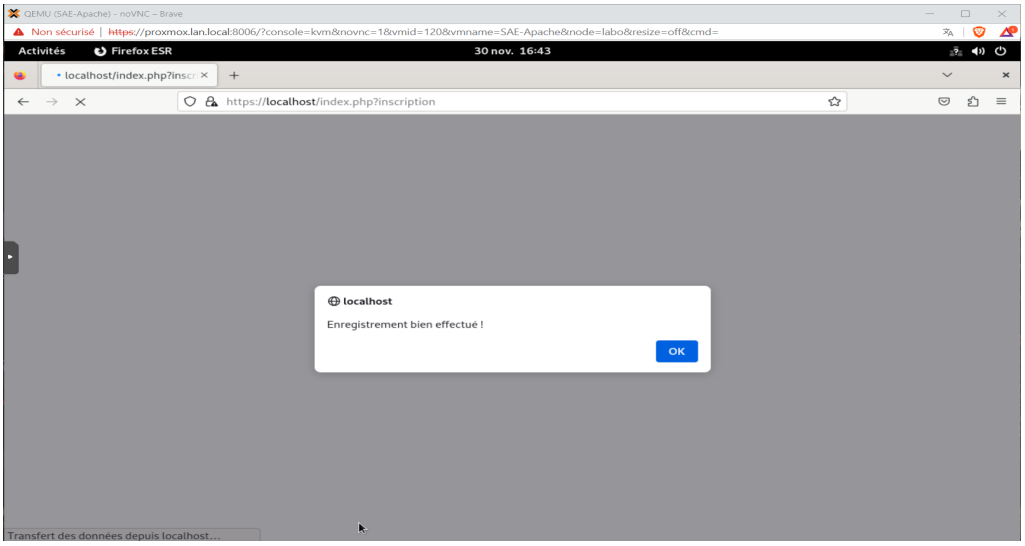


Utilisateurs créés dans l'Active Directory sur le serveur Windows 1

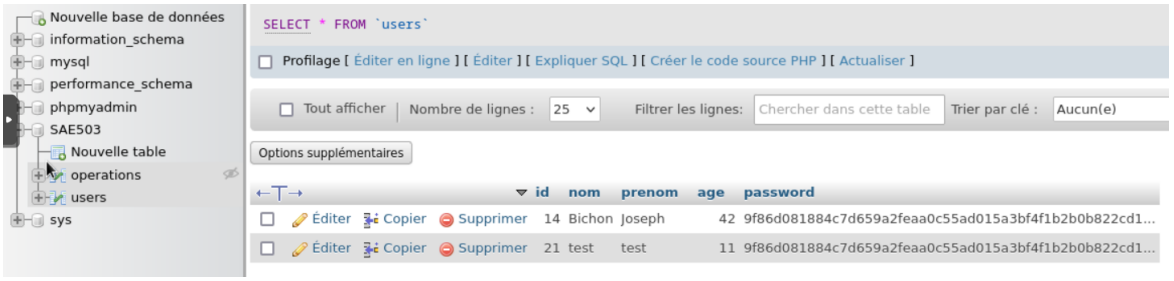


Ordinateurs renseignés dans l'Active Directory sur le serveur Windows 1

V. Serveur Linux Apache Mysql PHP (LAMP)

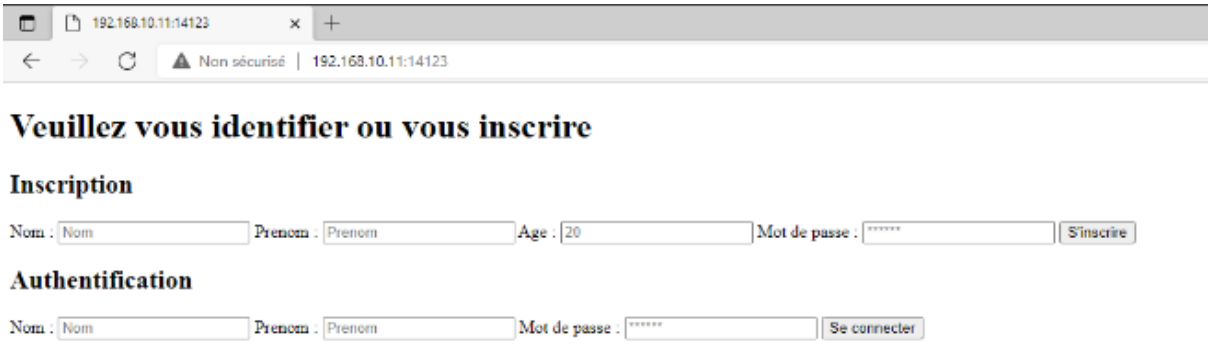


Confirmation de l'inscription pour l'utilisateur Test Test sur le serveur Apache (en localhost ici)



Présence de l'utilisateur Test Test dans la base de donnée MariaDB sur le serveur MariaDB

Test de connexion sur le serveur via son adresse DNS sur le réseau LAN



Test de connexion sur le serveur via son adresse NAT sur le réseau WAN de la PF externe (ici on sort sur la VMBR1 du proxmox pour ne pas sortir un internet)

Point de vue Red Team

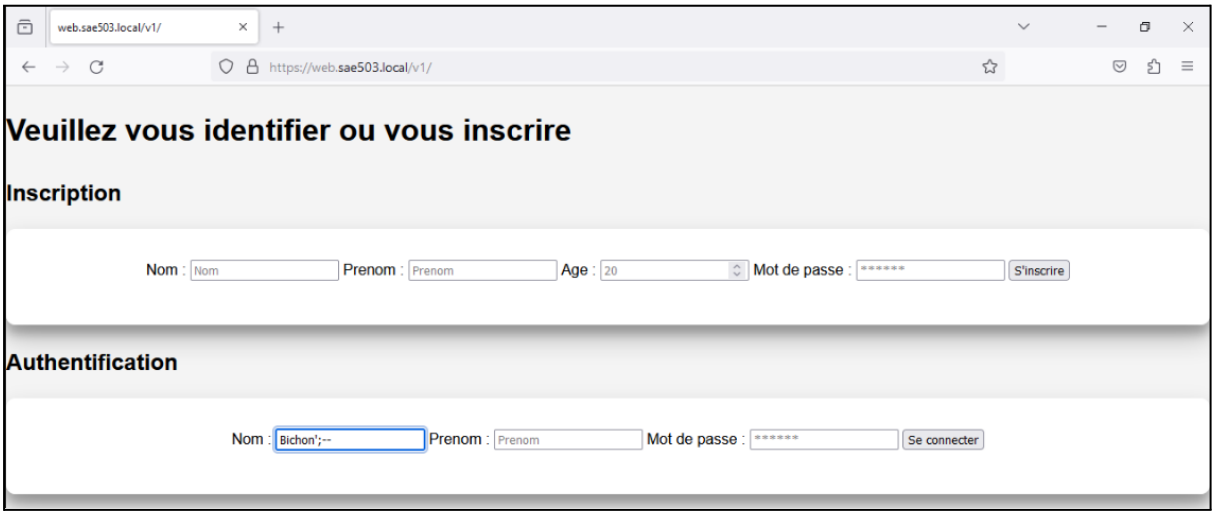
1) Enumeration

On note la présence d'un service HTTPS sur le serveur Web. La page d'accueil est une page d'inscription/authentification.

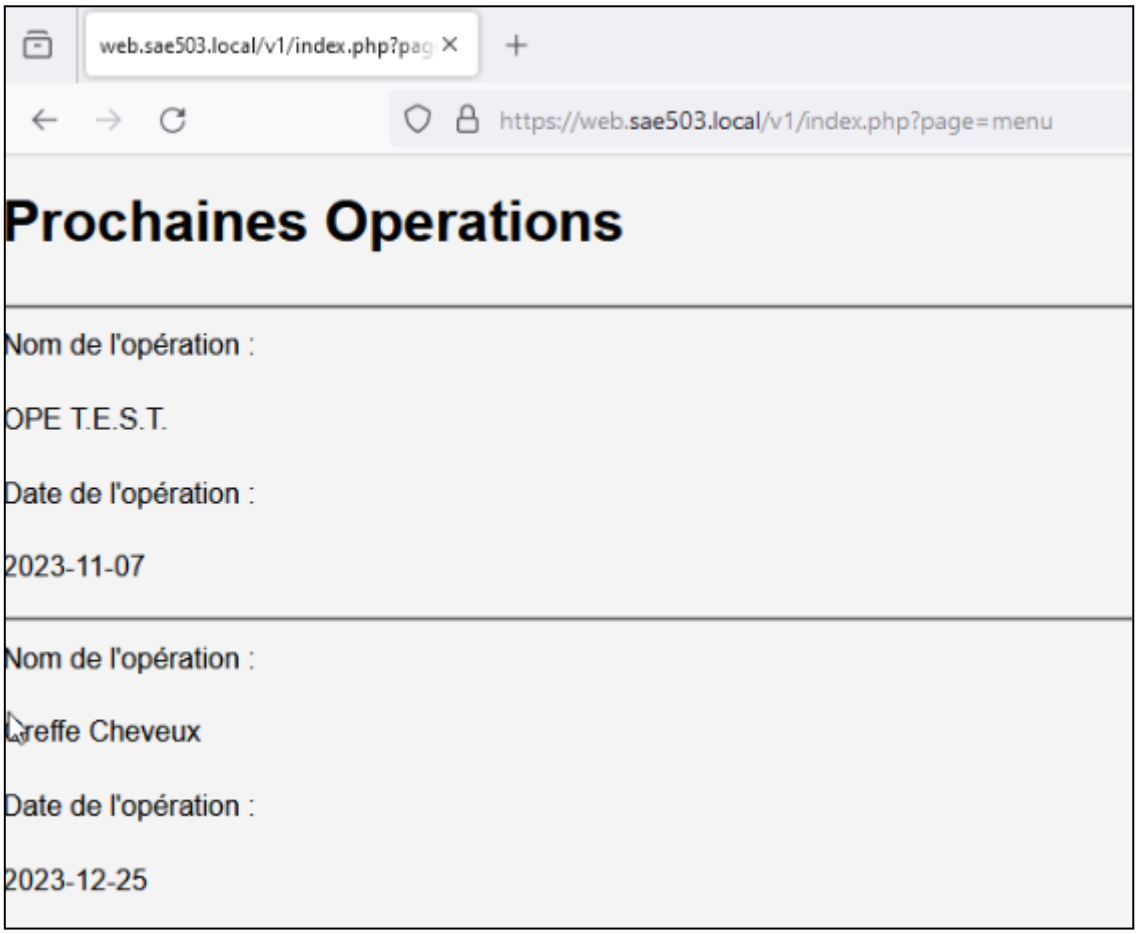
Création d'un compte utilisateur pour explorer l'application web.

Des opérations sont listées sur cette page, il est aussi possible d'en saisir.
On sait que Joseph Bichon est utilisateur de ce domaine, puisqu'il nous en a parlé à la dernière soirée foot.

2) Exploitation

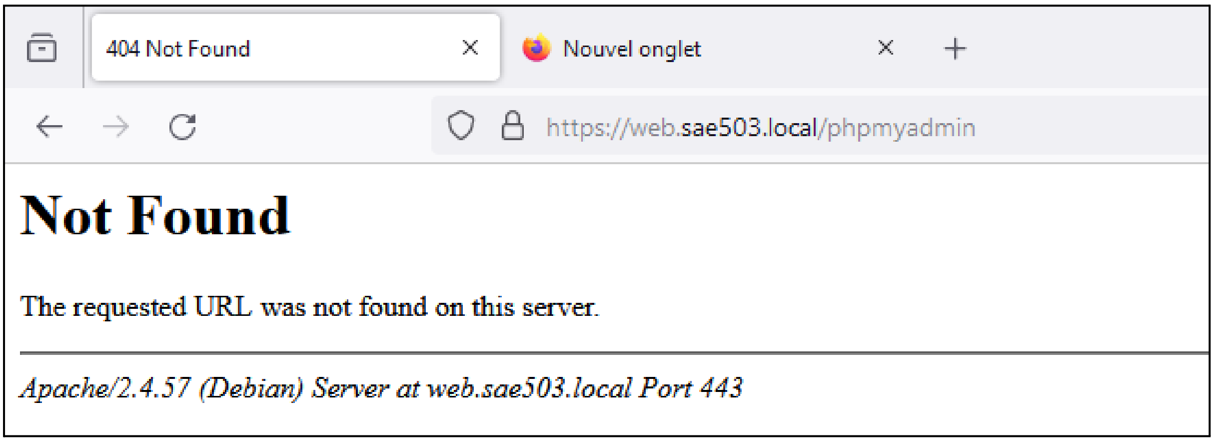


On teste donc la connexion en fermant immédiatement les guillemets de la variable ainsi que la ligne avec le ";" et en commentant le reste du code sur cette même ligne en utilisant "--".



C'est parfait ! On récupère les infos sur les opérations de l'utilisateur Joseph Bichon, l'injection à fonctionner.

3) Post-Exploitation



Puisqu'il semble qu'il y ait une base de données, on va essayer de se connecter sur la page d'administration la plus répandue PHPMyAdmin. Cela ne donnera rien, la page est en réalité inaccessible depuis l'extérieur.

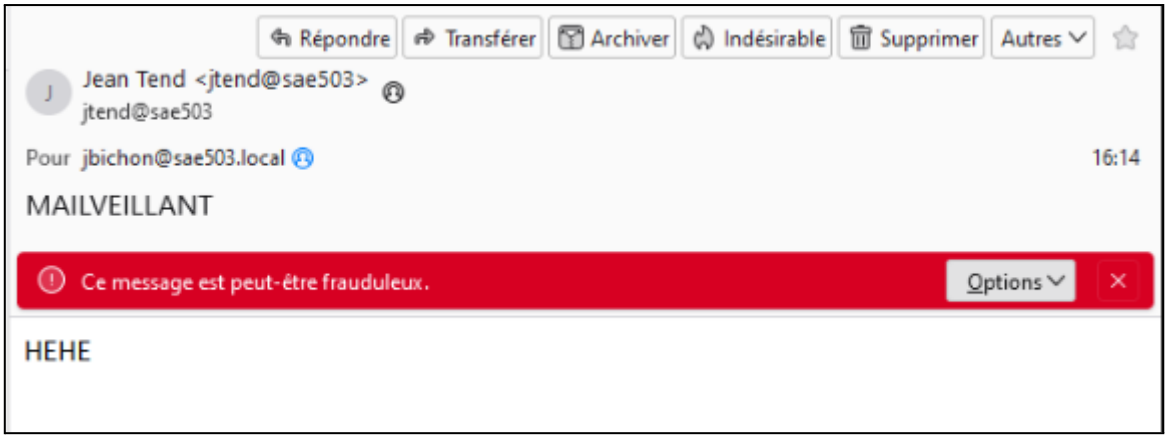
En énumérant les fichiers grâce à des outils on remarque l'utilisation d'un fichier Database.php, on va chercher à voir son contenu.
Pour cela on encode en BASE64 notre payload constitué d'un wrapper PHP récupérant le contenu d'un fichier avant son exécution par le serveur.

```php
<?php
/**
 * Classe MySQL utilisant un design pattern Singleton
 *
 * @author  Julien Crego <prenomnom@gmail.com>
 * @version 1.0
 */

error_reporting(E_ALL);
ini_set('display_errors', TRUE);
ini_set('display_startup_errors', TRUE);

class MySQL {

    static private $oPDO = NULL;
    static private $oInstance = NULL;

    /**
     * Constructeur défini en privé pour le rendre inaccessible
     */
    private function __construct() {
        self::$oPDO = new PDO('mariadb:host=192.168.1.12;dbname=SAE503', 'phpmyadmin', 'p@ssw0rd');
        self::$oPDO->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        self::$oPDO->query("SET NAMES 'utf8'");
    }
```

Il suffit ensuite de décoder le code en BASE64 afin de le récupérer en ASCII. On obtient alors un nom d'utilisateur ainsi que son mot de passe pour l'administration de la base de données, qui se confirme bien être phpmyadmin au vu du nom de l'utilisateur.

```html
<form action="https://web.sae503.local/menu.php" method="POST" name="formulaire">
        <input type="hidden" name="nom" value="HEHE"/>
        <input type="hidden" name="date" value="2023-11-10"/>
        <input type="hidden" name="id" value="14"/>
</form>
<script>document.formulaire.submit();</script>
```

Sur la page où il est possible d'ajouter une opération on va chercher à faire ajouter une opération à distance bien que nous ayons un accès permanent à son compte. Pour se faire on va utiliser une attaque CSRF avec le payload ci-dessus correspondant au formulaire légitime assaini.



Joseph Bichon reçoit donc un mail de notre part avec comme contenu notre payload (invisible à l'œil comme souhaité mais détecté par Thunderbird). A sa lecture notre payload s'exécute et ajoute donc une opération sous son nom à son insu !

Thunderbird est actuellement très bien protégé contre ce type d'attaque ce qui n'a pas permis de la réaliser jusqu'au bout. Le payload est fonctionnel et le principe reste toujours le même, ici seul le client mail nous bloque de par le fait qu'il soit à jour dans ses versions.

Point de vu Blue Team

1) Identification :

Pour ce qui est de **l'injection SQL** la faille est au niveau des **variables insérées** dans la requête, comme ci-dessous :

```php
function login($nom, $prenom, $pass) {
        $pdo = connexion();
        $pass = hash("sha256", $pass);
        $req = "SELECT id FROM users WHERE nom = '$nom' AND prenom = '$prenom' AND password = '$pass';";
        $sth = $pdo->query($req);
        $res = $sth->fetchAll();

        session_start();
        $_SESSION['id'] = $res[0]['id'];
        echo "<script>document.location='index.php?page=menu'</script>";
}
```

Concernant **la LFI** permettant la visualisation de code source PHP la faille est dans le **paramètre GET "page"** permettant l'injection de **wrappers** PHP.

https://web.sae503.local/v1/index.php?page=menu

2) Correction :

Il suffit simplement **d'assainir les variables** avant de les ajouter à la requête avec notamment la fonction PHP **htmlspecialchars()** remplaçant les caractères spéciaux par leur encodage HTML (ce qui **empêche l'injection SQL**).

```php
function login($nom, $prenom, $pass) {
        $nom = htmlspecialchars($nom);
        $prenom = htmlspecialchars($prenom);
        $pass = htmlspecialchars($pass);
        $pdo = connexion();
        $pass = hash("sha256", $pass);
        $req = "SELECT id FROM users WHERE nom = '$nom' AND prenom = '$prenom' AND password = '$pass';";
        $sth = $pdo->query($req);
        $res = $sth->fetchAll();

        session_start();
        $_SESSION['id'] = $res[0]['id'];
        echo "<script>document.location='menu.php'</script>";
}
```

De plus en **modifiant la structure même du site** afin de ne plus avoir recours à l'inclusion de page, on **élimine la faille** qu'est le paramètre "page", comme ci-dessous :

https://web.sae503.local/v2/menu.php

3) Vérification :

MariaDB server version for the right syntax to use near ''Bichon';' ANI
dex.php(14): login() #2 {main} thrown in **/var/www/html/Functions.php** on line **27**

Pour la LFI **la vérification n'est pas possible** tout simplement car il **n'existe plus** de champs injectables !

4) Résumé des failles → protections :

- Injection SQL dans le login de la page -> utilisation de htmlspecialchars()
- CSRF -> client mail à jour et/ou n'exécute pas le HTML & JS
- LFI -> pas d'arguments "modifiables" liés à des "include" du code PHP