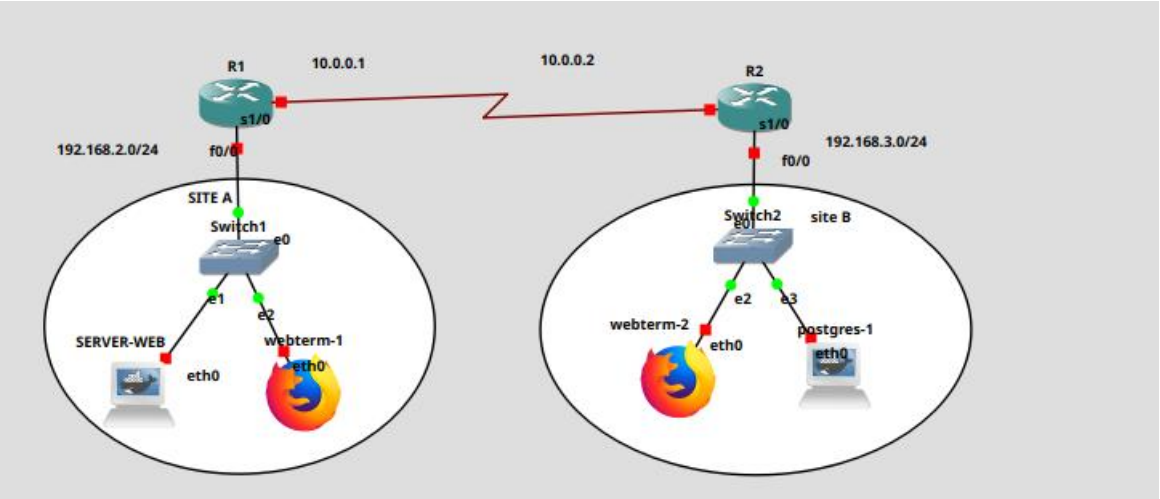
Projet 1 : VPN IPsec Site-à-Site Cisco avec services inter-sites

Ce document présente la mise en place d'un VPN IPsec site-à-site entre deux sites distants simulés dans GNS3 avec des routeurs Cisco. L'objectif est de sécuriser les échanges inter-sites et de permettre l'accès à des services hébergés localement .

I. Configurer les réseaux locaux des deux sites :

- **Architecture :**



L'architecture comprend deux sites distants (Site A et Site B) reliés via un VPN IPsec configuré entre deux routeurs Cisco (R1, R).

- **Site A** (192.168.2.0/24) héberge un serveur web.
- **Site B** (192.168.3.0/24) héberge une base de données.
  Les sites sont interconnectés par les sous-réseaux **10.1.1.0/24**(entre R1 et R2)

- **Configuration des routeurs et test**
- Paramétrage des interfaces réseau du routeur R1 avec les adresses IP correspondant au réseau local et au lien point à point avec R2

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#interface fa0/0
R1(config-if)# ip address 192.168.2.1 255.255.255.0
R1(config-if)# no shutdown
R1(config-if)#
R1(config-if)#interface s1/0
R1(config-if)# ip address 10.0.0.1 255.255.255.252
R1(config-if)# clock rate 64000
```

- Mise en place d'un tunnel VPN de site à site en utilisant le protocole IPSec, incluant la configuration des politiques ISAKMP, des transform-sets, et de la crypto map appliquée aux interfaces

```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#crypto isakmp policy 10
R1(config-isakmp)# encr aes 256
R1(config-isakmp)# authentication pre-share
R1(config-isakmp)# hash sha
R1(config-isakmp)# group 2
R1(config-isakmp)# lifetime 86400
R1(config-isakmp)#
R1(config-isakmp)#crypto isakmp key vpnkey123 address 10.0.0.2
R1(config)#
R1(config)#crypto ipsec transform-set MY-TRANSFORM esp-aes esp-sha-hmac
R1(cfg-crypto-trans)# mode tunnel
R1(cfg-crypto-trans)#
R1(cfg-crypto-trans)#$t ip 192.168.2.0 0.0.0.255 192.168.3.0 0.0.0.255
R1(config)#
R1(config)#crypto map VPN-MAP 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
R1(config-crypto-map)# set peer 10.0.0.2
R1(config-crypto-map)# set transform-set MY-TRANSFORM
R1(config-crypto-map)# match address 100
```

- Vérification de l'état du protocole ISAKMP : la session d'échange de clés est active, indiquant que la phase 1 du VPN IPSec est correctement établie

```
R1(config-crypto-map)#interface s1/0
R1(config-if)# crypto map VPN-MAP
R1(config-if)#
*Mar  1 00:01:58.179: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
```

- Vérification de la table de routage avec **sh ip route** : Et on voit bien que l'attribution a été une réussite.

```
R1#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     10.0.0.0/30 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Serial1/0
C    192.168.2.0/24 is directly connected, FastEthernet0/0
```

La commande **show ip route** permet de vérifier que les routes vers les réseaux distants ont bien été ajoutées. Cela confirme le bon fonctionnement du routage et l'accessibilité entre les sous-réseaux via le tunnel VPN.

- Configuration du routeur R2 et attribution des éléments TCP/IP

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#interface fa0/0
R2(config-if)# ip address 192.168.3.1 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)#
R2(config-if)#interface s1/0
R2(config-if)# ip address 10.0.0.2 255.255.255.252
R2(config-if)# no shutdown
R2(config-if)#
```

Le routeur R2 a été configuré avec les adresses IP appropriées sur ses interfaces : une interface reliée au réseau local (LAN B) et une autre connectée au réseau d'interconnexion avec R1 (10.0.0.0/30). Cette étape garantit que R2 peut communiquer avec les équipements de son réseau local ainsi qu'avec le routeur distant via le lien point-à-point.

- Définition du tunnel VPN avec le Protocol IPSec

```
R2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#crypto isakmp policy 10
R2(config-isakmp)# encr aes 256
R2(config-isakmp)# authentication pre-share
R2(config-isakmp)# hash sha
R2(config-isakmp)# group 2
R2(config-isakmp)# lifetime 86400
R2(config-isakmp)#
R2(config-isakmp)#crypto isakmp key vpnkey123 address 10.0.0.1
R2(config)#
R2(config)#crypto ipsec transform-set MY-TRANSFORM esp-aes esp-sha-hmac
R2(cfg-crypto-trans)# mode tunnel
R2(cfg-crypto-trans)#
R2(cfg-crypto-trans)#$t ip 192.168.3.0 0.0.0.255 192.168.2.0 0.0.0.255
R2(config)#
R2(config)#crypto map VPN-MAP 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
R2(config-crypto-map)# set peer 10.0.0.1
R2(config-crypto-map)# set transform-set MY-TRANSFORM
R2(config-crypto-map)# match address 100
R2(config-crypto-map)#
R2(config-crypto-map)#interface s1/0
R2(config-if)# crypto map VPN-MAP
R2(config-if)#
*Mar  1 00:02:08.803: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
R2(config-if)#exit
R2(config)#ip route 192.168.2.0 255.255.255.0 10.0.0.1
```

- Test avec la commande show crypto ipsec sa pour vérifier la bonne application des configurations IPSec : les résultats confirment que le tunnel VPN est opérationnel

```
R1#show crypto ipsec sa

interface: Serial1/0
    Crypto map tag: VPN-MAP, local addr 10.0.0.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (192.168.2.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (192.168.3.0/255.255.255.0/0/0)
   current_peer 10.0.0.2 port 500
     PERMIT, flags={origin_is_acl,ipsec_sa_request_sent}
    #pkts encaps: 2, #pkts encrypt: 2, #pkts digest: 2
    #pkts decaps: 2, #pkts decrypt: 2, #pkts verify: 2
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 1, #recv errors 0

     local crypto endpt.: 10.0.0.1, remote crypto endpt.: 10.0.0.2
     path mtu 1500, ip mtu 1500, ip mtu idb Serial1/0
```

❖ Test de connectivite entre les deux sites on voit que ca marche

```
/ # ping 192.168.3.2
PING 192.168.3.2 (192.168.3.2): 56 data bytes
64 bytes from 192.168.3.2: seq=0 ttl=62 time=51.541 ms
64 bytes from 192.168.3.2: seq=1 ttl=62 time=24.226 ms
```

❖ Test de l'état d'ISAKMP : la session est active et fonctionne correctement, comme observé également sur le routeur R2

```
R1#show crypto isakmp sa
dst             src             state          conn-id slot status
10.0.0.2        10.0.0.1        QM_IDLE              1    0 ACTIVE
```

Résumé de la configuration VPN IPsec

On commence par définir la politique ISAKMP (phase 1) pour négocier les clés sécurisées entre les routeurs, avec une clé pré-partagée. Ensuite, on configure le transform-set IPSec (phase 2) pour chiffrer les données. On crée une crypto map liant les paramètres à une liste d'accès qui définit le trafic à protéger, puis on applique cette crypto map à l'interface concernée.

Pour vérifier, on utilise show crypto isakmp sa et show crypto ipsec sa afin de s'assurer que le tunnel est actif et que les données sont chiffrées. Cette configuration permet de sécuriser les échanges entre les réseaux locaux via un tunnel IPsec.

II. Déployer les services :

Dans cette partie, nous allons installer et configurer les services principaux sur chaque site afin de tester la connectivité et l'utilité du VPN.

- Installation de apache2 pour fournir un service web

```
/ # apt install apache2 php libapache2-mod-php php-pgsql
```

- Installation de PostgreSQL

```
/ # apt install postgresql postgresql-contrib systemctl nano -y
```

- Téléchargement d'une image Docker de PostgreSQL avec la commande `docker pull postgres`, en vue d'un déploiement conteneurisé.

```
root@server-VMware:~# sudo docker pull postgres
```

- Après avoir téléchargé l'image Docker PostgreSQL, nous avons utilisé la commande **pg_createcluster 17 main --start** pour créer et démarrer un cluster PostgreSQL local.
  Cette étape initialise l'environnement de la base de données sur le Site B, rendant PostgreSQL opérationnel pour accepter des connexions depuis le Site A via le tunnel VPN.

```
/ # sudo pg_createcluster 17 main --start
```

- Préparation d'une base de données sécurisée avec un utilisateur spécifique qui pourra gérer les tables d'utilisateurs. Puis création de deux tables pour stocker des utilisateurs et leurs mots de passe

```
/ # sudo -u postgres psql
psql (17.5 (Debian 17.5-1.pgdg120+1))
Type "help" for help.

postgres=#
postgres=# CREATE DATABASE securite;
CREATE DATABASE
postgres=# CREATE USER aziz WITH PASSWORD 'aziz123';
CREATE ROLE
postgres=# GRANT ALL PRIVILEGES ON DATABASE securite TO aziz;
GRANT
postgres=# \c securite
You are now connected to database "securite" as user "postgres".
securite=# CREATE TABLE utilisateurs (
    id SERIAL PRIMARY KEY,
    username TEXT NOT NULL,
    password TEXT NOT NULL
);
CREATE TABLE
securite=# CREATE TABLE utilisateurs_clair (
    id SERIAL PRIMARY KEY,
    username TEXT NOT NULL,
    password TEXT NOT NULL
);
CREATE TABLE
securite=# GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO aziz;
GRANT
securite=# GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO aziz;
GRANT
securite=#
```

Explication :

- Connexion à PostgreSQL en tant qu'utilisateur **postgres**
- Création d'une base de données **securite**
- Création d'un utilisateur **aziz** avec un mot de passe
- Attribution des droits sur la base **securite** à l'utilisateur **aziz**
- Connexion à la base **securite** en tant que superutilisateur **postgres**
- Création de deux tables dans la base securite
  - utilisateurs : avec trois colonnes — un identifiant unique auto-incrémenté id, un **username** (nom d'utilisateur) obligatoire, et un **password** (mot de passe) obligatoire.
  - **utilisateurs_clair** : une autre table identique à la première (apparemment pour stocker des mots de passe en clair, ce qui n'est pas conseillé en pratique pour la sécurité).
- Attribution des droits à l'utilisateur aziz sur toutes les tables et séquences

➢ Ensuite, nous avons configuré PostgreSQL pour qu'il accepte les connexions distantes.

- Nous avons modifié le fichier **postgresql.conf** qui par défaut écoute sur tout les adresses listen_addresses = '*'

```
listen_addresses = '*'
```

- Puis précisé les adresses autorisées (`localhost,192.168.2.2`)

```
# - Connection Settings -

listen_addresses = 'localhost,192.168.2.2'
```

- Nous avons autorisé l'utilisateur aziz à se connecter à la base de données **securite** depuis l'adresse IP 192.168.2.2/32 en utilisant l'authentification md5.

```
# hostnogssenc  DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
  host          securite  aziz  192.168.2.2/32  md5
```

- Création du fichier html pour la page web

```
/var/www/html # nano index.html
```

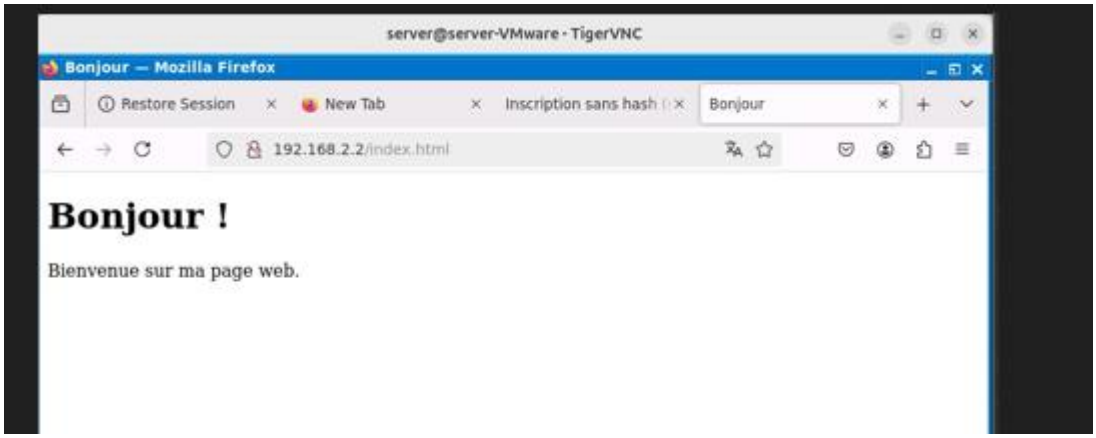
- On édite le code qui affiche juste un message de bienvenue

```
  GNU nano 7.2                        index.html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8" />
    <title>Bonjour</title>
</head>
<body>
    <h1>Bonjour !</h1>
    <p>Bienvenue sur ma page web.</p>
</body>
</html>
```

- On redémarre apache2

```
/var/www/html # systemctl restart apache2
```

- On affiche la page en passant par l'adresse du serveur web



5 ET 6 : **Tester les accès inter-sites ET Sécuriser davantage** (ACL, NAT si besoin)

On va d'abord faire un ping de la base de donnée (SITE B) vers le serveur Web (SITE A)

```
/etc/postgresql/17/main # ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2): 56 data bytes
64 bytes from 192.168.2.2: seq=0 ttl=61 time=63.410 ms
64 bytes from 192.168.2.2: seq=1 ttl=61 time=51.261 ms

--- 192.168.2.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 51.261/57.335/63.410 ms
```

On voit que ca marche !!

Testons le chemin inverse (SITE A) vers (Site B)

```
/var/www/html # ping 192.168.3.1
PING 192.168.3.1 (192.168.3.1): 56 data bytes
64 bytes from 192.168.3.1: seq=0 ttl=253 time=42.059 ms
64 bytes from 192.168.3.1: seq=1 ttl=253 time=28.108 ms
64 bytes from 192.168.3.1: seq=2 ttl=253 time=29.277 ms
^C
--- 192.168.3.1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 28.108/33.148/42.059 ms
```

Test de sécurité avec le hachage de mot de passe et capture Wireshark

Nous avons réalisé deux scénarios de test afin d'évaluer la sécurité des échanges réseau en fonction de l'utilisation d'un tunnel VPN IPsec et du hachage des mots de passe :

1. Communication sécurisée : VPN activé + mot de passe haché

- Un mot de passe est transmis depuis le client vers le serveur après avoir été **haché avec un algorithme sécurisé** (comme **bcrypt**).
- La communication entre les routeurs **A et B** passe à travers un **VPN Ipsec** .

2. Communication non sécurisée : VPN désactivé + mot de passe en clair

- Le mot de passe est transmis **sans hachage**, directement dans le corps de la requête.
- La connexion entre les routeurs A et B se fait **sans VPN**, donc **aucun chiffrement réseau** n'est appliqué.

Notre allons créer un formulaire pour chaqu'un des cas.

```
 GNU nano 7.2                    formulaire_hash.php *
    </style>
</head>
<body>
    <div class="form-container">
        <h2>Formulaire SECURISER</h2>
        <form action="insert_hash.php" method="POST">
            <label for="username">Nom d'utilisateur :</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Mot de passe :</label>
            <input type="password" id="password" name="password" required>

            <input type="submit" value="S'inscrire">

            <div class="info"> ^=^t^r Le mot de passe sera hash   avant stockag>
        </form>
    </div>
</body>
</html>
```

Formulaire_hash.php représente notre interfaces

```
 GNU nano 7.2                        insert_hash.php
<?php
// insert_hash.php

// Connexion    PostgreSQL distant (VPN actif suppos  )
$db = pg_connect("host=192.168.3.2 dbname=securite user=aziz password=aziz123");

if (!$db) {
    die("Erreur de connexion    la base PostgreSQL.");
}

// Recuperation des donnees POST
$username = $_POST['username'] ?? '';
$password = $_POST['password'] ?? '';

// Validation simple
if (empty($username) || empty($password)) {
    die("Veuillez fournir un nom d'utilisateur et un mot de passe.");
}
```

Nous nous connect0ns a notre base de donnée en spécifiant l'ip de la base de donnée et ses identifiants.

```
// Hashage du mot de passe avec BCRYPT
$hashed_password = password_hash($password, PASSWORD_BCRYPT);

// Insertion dans la table utilisateurs_hash (mot de passe s  curis  )
$query = "INSERT INTO utilisateurs_hash (username, password) VALUES ($1, $2)";
$result = pg_query_params($db, $query, array($username, $hashed_password));

if ($result) {
    echo "Utilisateur ajout   avec succes (mot de passe hash  ).";
} else {
    echo "Erreur lors de l'ajout de l'utilisateur : " . pg_last_error($db);
}

pg_close($db);
?>
```

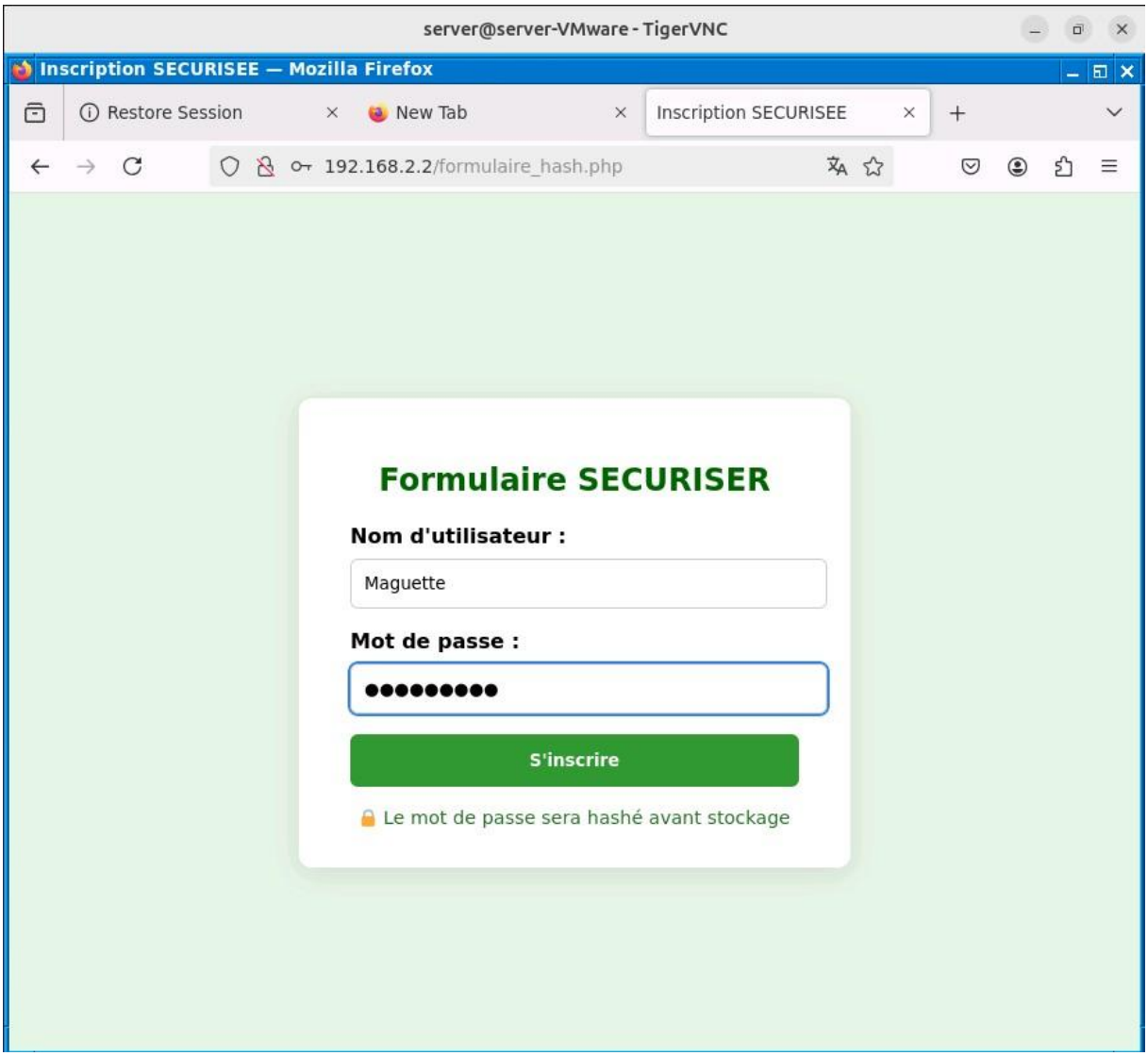
On fait le **hashage** du mot de passe avec **password_hash** , une fonction se trouvant dans **php** puis nous stockons le mot de passe hasher dans notre base de donnée

Nous installons le paquet **postgresql-client** pour pouvoir communiquer avec notre base de donnée

```
/var/www/html # apt install postgresql-client
```

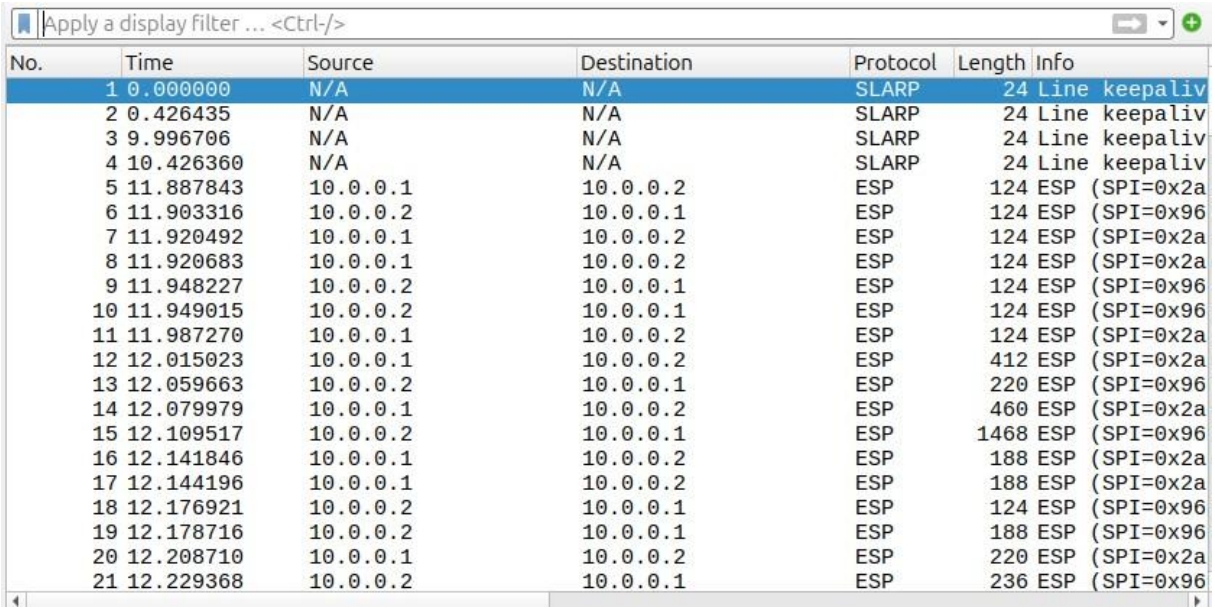
Et nous redémarrons **apache2**

```
/var/www/html # systemctl restart apache2
```




```
securite=> SELECT * FROM utilisateurs;
 id | username |                          password
----+----------+------------------------------------------------------------
  1 | aziz     | $2y$10$xM9A8fa79G9J6pF6qRoHiuThftp.6lNsBRrw0F51zhv.v6Gbi5u8O
(1 row)
```

On avait entrer le mot de passe : passer123 qui a été hasher et est maintenant un text incompréhensible.


Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	N/A	N/A	SLARP	24	Line keepaliv
2	0.426435	N/A	N/A	SLARP	24	Line keepaliv
3	9.996706	N/A	N/A	SLARP	24	Line keepaliv
4	10.426360	N/A	N/A	SLARP	24	Line keepaliv
5	11.887843	10.0.0.1	10.0.0.2	ESP	124	ESP (SPI=0x2a
6	11.903316	10.0.0.2	10.0.0.1	ESP	124	ESP (SPI=0x96
7	11.920492	10.0.0.1	10.0.0.2	ESP	124	ESP (SPI=0x2a
8	11.920683	10.0.0.1	10.0.0.2	ESP	124	ESP (SPI=0x2a
9	11.948227	10.0.0.2	10.0.0.1	ESP	124	ESP (SPI=0x96
10	11.949015	10.0.0.2	10.0.0.1	ESP	124	ESP (SPI=0x96
11	11.987270	10.0.0.1	10.0.0.2	ESP	124	ESP (SPI=0x2a
12	12.015023	10.0.0.1	10.0.0.2	ESP	412	ESP (SPI=0x2a
13	12.059663	10.0.0.2	10.0.0.1	ESP	220	ESP (SPI=0x96
14	12.079979	10.0.0.1	10.0.0.2	ESP	460	ESP (SPI=0x2a
15	12.109517	10.0.0.2	10.0.0.1	ESP	1468	ESP (SPI=0x96
16	12.141846	10.0.0.1	10.0.0.2	ESP	188	ESP (SPI=0x2a
17	12.144196	10.0.0.1	10.0.0.2	ESP	188	ESP (SPI=0x2a
18	12.176921	10.0.0.2	10.0.0.1	ESP	124	ESP (SPI=0x96
19	12.178716	10.0.0.2	10.0.0.1	ESP	188	ESP (SPI=0x96
20	12.208710	10.0.0.1	10.0.0.2	ESP	220	ESP (SPI=0x2a
21	12.229368	10.0.0.2	10.0.0.1	ESP	236	ESP (SPI=0x96

Le **protocole ESP (Encapsulating Security Payload)** est un composant principal de la suite IPsec. Il est utilisé pour **sécuriser les paquets IP** en assurant les services suivants :

🎯 Fonctions principales du protocole ESP :

1. **Confidentialité (chiffrement)**
   Il chiffre la charge utile (données transportées) pour que seule la partie destinataire puisse la lire.
2. **Authentification (optionnelle)**
   ESP peut aussi garantir que le paquet n'a pas été modifié en transit, via un code d'authentification (HMAC par exemple).
3. **Intégrité et anti-rejeu**
   ESP utilise un compteur de séquence pour protéger contre les attaques par rejeu (retransmission de paquets capturés).
4. **Encapsulation complète (mode tunnel)**
   En **mode tunnel**, ESP chiffre tout le paquet IP original, y compris l'en-tête IP source/destination, et ajoute un nouvel en-tête IP (souvent entre deux routeurs VPN).
   En **mode transport**, seul le contenu (les données) est chiffré, pas l'en-tête IP.

GNU nano 7.2 formulaire_clair.php

```
</head>
<body>
    <div class="form-container">
        <h2>Formulaire NON s  curis  </h2>
        <form action="insert_clair.php" method="POST">
            <label for="username">Nom d'utilisateur :</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Mot de passe :</label>
            <input type="password" id="password" name="password" required>

            <input type="submit" value="S'inscrire">

            <div class="warning"> ^z   ^o Ce formulaire transmet le mot de pass>
        </form>
    </div>
</body>
</html>
```

GNU nano 7.2 insert_clair.php

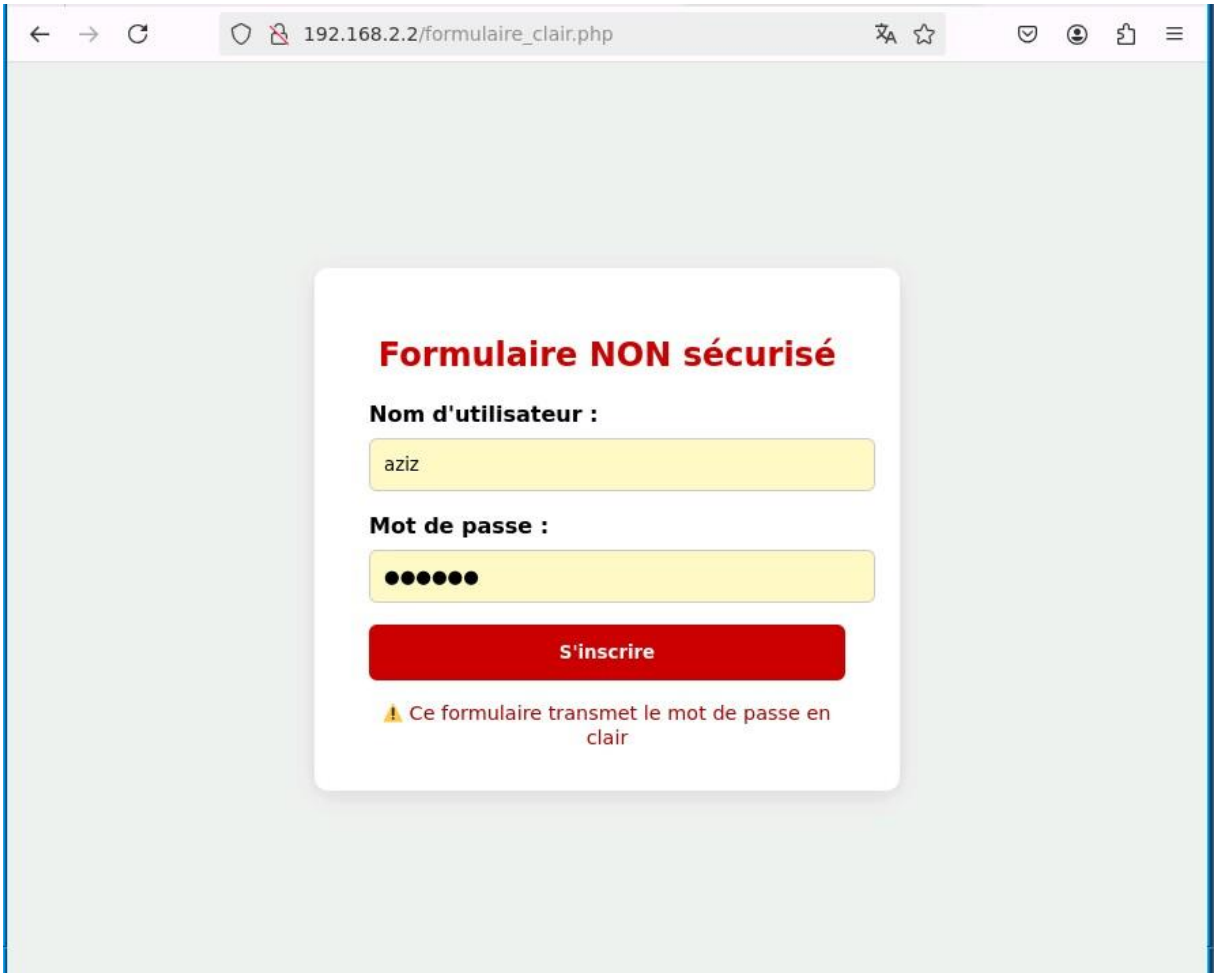
```
<?php
// Recuperation des donnees du formulaire
$username = $_POST['username'];
$password = $_POST['password'];

// Connexion    PostgreSQL
$host = "192.168.3.2"; // IP du serveur PostgreSQL
$db = "securite";
$user = "aziz";
$pass = "aziz123";

try {
    $pdo = new PDO("pgsql:host=$host;dbname=$db", $user, $pass);

    // Requete d'insertion (sans hash, donc non s  curis  )
    $stmt = $pdo->prepare("INSERT INTO utilisateurs_clair (username, password) >
    $stmt->execute(['username' => $username, 'password' => $password]);
```

```
    echo "Erreur : " . $e->getMessage();
}
?>
```




```
......./N...%....user.aziz.database.securite..R.......
SCRAM-SHA-256..p...6SCRAM-SHA-256.... n,,n=,r=tO33X5Zdl3LxFbdNb5alEG
IeR...\....r=tO33X5Zdl3LxFbdNb5alEGIeX0mJBPCylQbTa52mMV8WB3mH,s=j5wk
IgzQsnDAa71dK9b35A==,i=4096p...lc=biws,r=tO33X5Zdl3LxFbdNb5alEGIeX0m
JBPCylQbTa52mMV8WB3mH,p=X9c+UYB8awVe89CKth2gYZkIhDVQDVMRbxAZVppGAzo=
R...6....v=Z+1vDTCUvAOHetAEYxBHtl5AwRKh/8nr7s7CfX4Ej0U=R........S...
.in_hot_standby.off.S....integer_datetimes.on.S....TimeZone.Etc/UTC.
S....IntervalStyle.postgres.S....is_superuser.off.S....application_n
ame..S...&default_transaction_read_only.off.S....scram_iterations.40
96.S....DateStyle.ISO, MDY.S...#standard_conforming_strings.on.S....
session_authorization.aziz.S....client_encoding.UTF8.S...2server_ver
sion.17.5 (Debian 17.5-1.pgdg120+1).S....server_encoding.UTF8.K.....
....|..Z....IP...dpdo_stmt_00000001.INSERT INTO utilisateurs_clair (
username, password) VALUES ($1, $2)...........S....1....Z....IB...5.
pdo_stmt_00000001.............aziz....passer....D....P.E...    .....
S....2....n....C....INSERT 0 1.Z....IQ...!DEALLOCATE pdo_stmt_000006
01.C....DEALLOCATE.Z....IX....
```

6.BILAN

1. Problème avec HTTP seul

HTTP (port 80) **n'est pas sécurisé** :

- Les **données circulent en clair** (texte lisible).

- Tout ce que vous envoyez (identifiants, mots de passe, requêtes...) **peut être intercepté** par un attaquant avec un outil comme Wireshark.
- Exemple :

  > Une personne dans le même réseau (ex : Wi-Fi public) peut capturer votre requête HTTP et lire directement

---

🔐 2. Ce que fait un VPN IPsec

IPsec (Internet Protocol Security) est un protocole réseau qui chiffre **tout le trafic IP** entre deux points (routeurs, serveurs, clients, etc.).

Avec IPsec :

- Les paquets sont **chiffrés (encryptés)**.
- L'**intégrité** des données est garantie.
- L'**authenticité** des communications est vérifiée.

Conclusion

La mise en place du VPN IPsec a permis d'assurer une **communication sécurisée entre sites distants**, en protégeant les échanges contre les interceptions et les altérations. Grâce au chiffrement et à l'authentification fournis par IPsec, ce projet garantit la **confidentialité, l'intégrité et l'authenticité** des données transitant sur un réseau potentiellement non fiable comme Internet. Ce mécanisme renforce ainsi la sécurité globale de l'infrastructure et constitue une solution fiable pour les organisations souhaitant interconnecter des réseaux distants en toute confiance.