Analyse de trafic avec Wireshark sur DVWA

Installation d'un environnement DVWA et capture d'identifiants en clair avec Wireshark

1. Présentation du scénario

Cette documentation présente la mise en place d'un environnement de test DVWA (Damn Vulnerable Web Application) sur une machine Kali Linux, puis l'analyse du trafic réseau généré lors d'une authentification avec Wireshark. L'objectif est de démontrer concrètement le risque d'une connexion HTTP non chiffrée : les identifiants saisis dans le formulaire de connexion circulent en clair sur le réseau et peuvent être interceptés par quiconque capture le trafic.

- Machine de test : Kali Linux, pile LAMP locale (Apache, MariaDB, PHP)
- Application cible : DVWA (Damn Vulnerable Web Application), volontairement vulnérable
- Accès : http://localhost/dvwa
- Outil d'analyse : Wireshark, capture sur l'interface de loopback (lo)

2. Installation de Wireshark

Les dépôts de paquets sont actualisés puis Wireshark est installé :

```
sudo apt update
sudo apt install wireshark
```

Lors de l'installation, la reconfiguration du paquet permet d'autoriser les utilisateurs non-root à capturer du trafic réseau sans avoir besoin de sudo à chaque lancement. L'utilisateur courant est ensuite ajouté au groupe wireshark, qui dispose des permissions nécessaires sur les interfaces de capture :

```
sudo dpkg-reconfigure wireshark-common
sudo usermod -aG wireshark $USER
```

Une déconnexion/reconnexion de session (ou un redémarrage) est nécessaire pour que l'appartenance au nouveau groupe soit prise en compte.

3. Activation du service SSH

Le service SSH est activé et démarré afin de permettre une administration à distance de la machine Kali en complément de son accès local :

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

Son état de fonctionnement et l'écoute effective du port 22 sont ensuite vérifiés :

```
sudo systemctl status ssh
sudo ss -tulpn | grep ssh
```

4. Installation de la pile LAMP

DVWA nécessite un serveur web Apache, une base de données MariaDB et PHP avec ses extensions MySQL et GD (nécessaire à la génération des CAPTCHA). L'ensemble est installé en une seule commande :

```
sudo apt update
sudo apt install apache2 mariadb-server php php-mysqli php-gd -y
```

5. Récupération et configuration de DVWA

Git est installé puis le dépôt officiel de DVWA est cloné directement dans le répertoire servi par Apache :

```
sudo apt install git -y
sudo git clone https://github.com/digininja/DVWA.git /var/www/html/dvwa
```

Le fichier de configuration type est dupliqué en fichier de configuration actif, puis la propriété de l'ensemble des fichiers de l'application est attribuée à l'utilisateur du serveur web (www-data), condition nécessaire pour qu'Apache et PHP puissent écrire dans certains répertoires de l'application :

```
sudo cp /var/www/html/dvwa/config/config.inc.php.dist /var/www/html/dvwa/config/config.inc.php
sudo chown -R www-data:www-data /var/www/html/dvwa
```

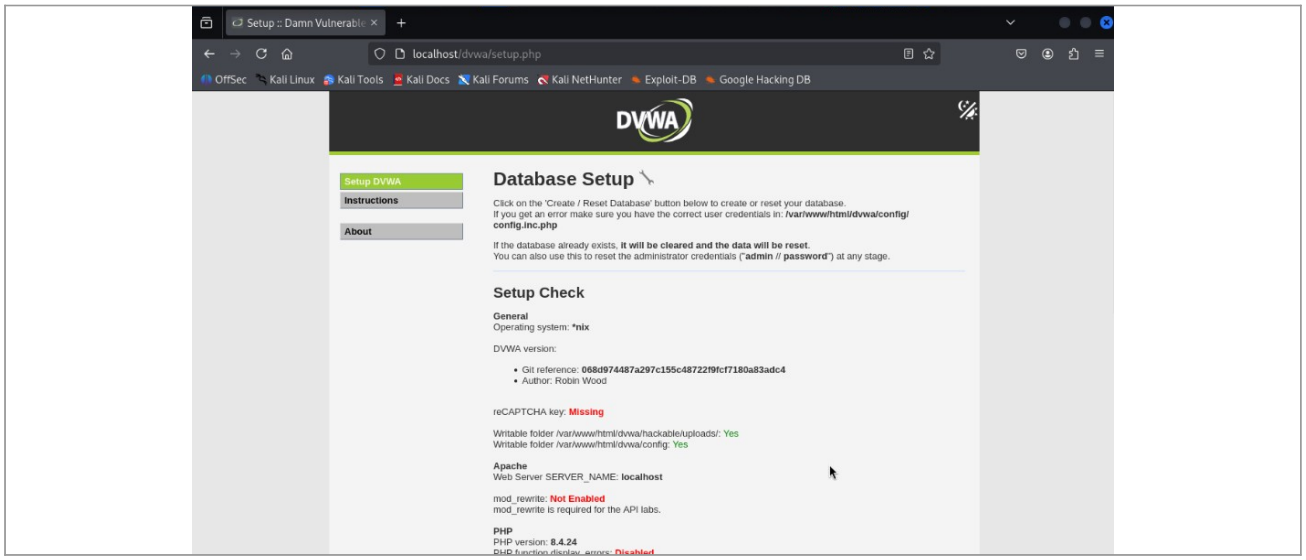
6. Démarrage de la base de données

Le service MariaDB est démarré, puis le client en ligne de commande est ouvert en tant qu'administrateur afin de pouvoir, si nécessaire, créer la base et l'utilisateur attendus par le fichier config.inc.php de DVWA :

```
sudo systemctl start mariadb
sudo mariadb
```

7. Initialisation de la base DVWA

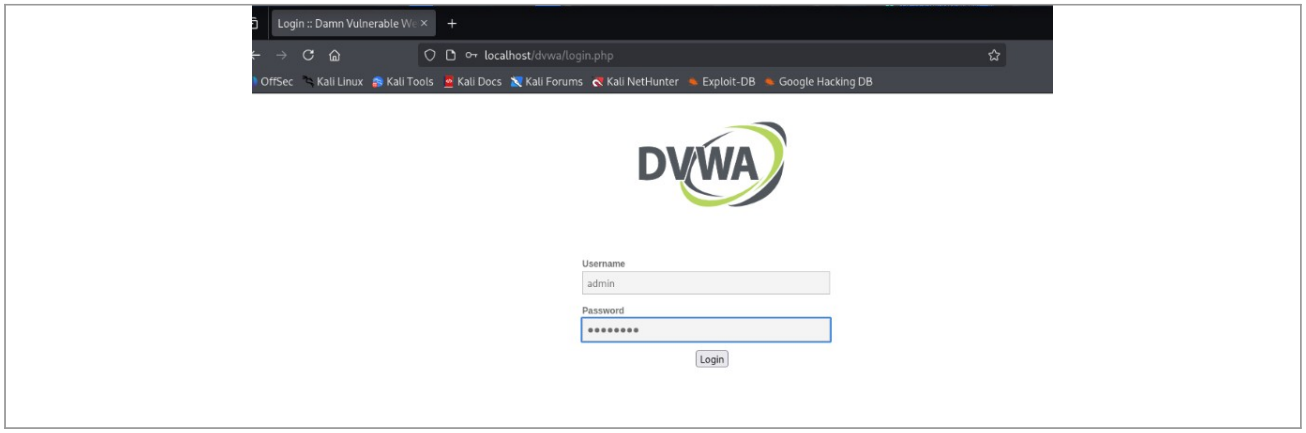
La page de configuration de DVWA (setup.php) liste les prérequis détectés automatiquement : système d'exploitation, version de DVWA, droits d'écriture sur les dossiers hackable/uploads et config, version de PHP, état du module reCAPTCHA et de mod_rewrite. Le bouton Create / Reset Database initialise ensuite les tables de l'application dans MariaDB :



Une fois la base créée, l'application redirige automatiquement vers l'écran de connexion.

8. Connexion à DVWA

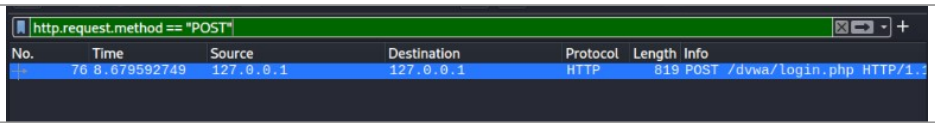
L'écran de connexion de DVWA accepte les identifiants par défaut de l'application (admin / password). C'est précisément cette requête de connexion qui va être interceptée à l'étape suivante :



9. Capture du trafic avec Wireshark

Avant de soumettre le formulaire de connexion, une capture est démarrée dans Wireshark sur l'interface de loopback (lo), puisque le navigateur et le serveur Apache tournent tous deux localement sur la machine. Un filtre d'affichage restreint la liste aux requêtes HTTP de type POST, qui correspondent aux soumissions de formulaires :

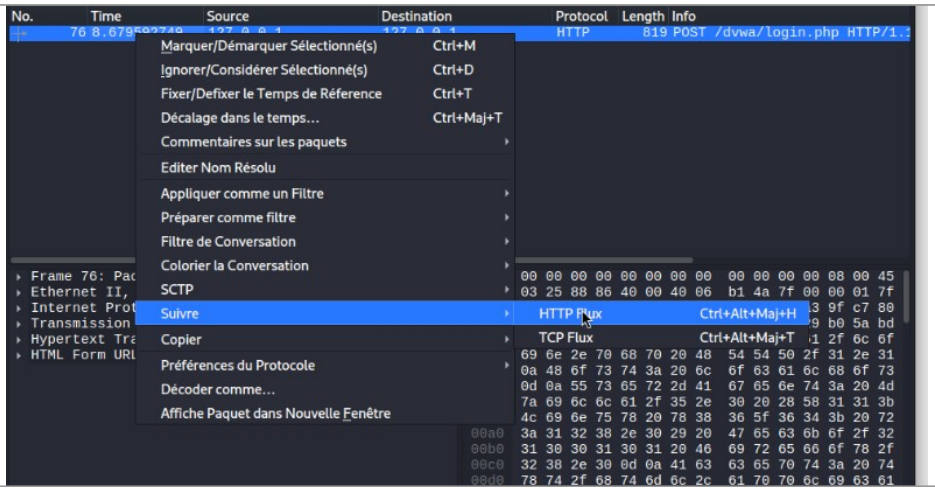
```
http.request.method == "POST"
```



Le paquet repéré correspond à la requête POST /dvwa/login.php HTTP/1.1, envoyée par le navigateur vers le serveur local au moment de la connexion.

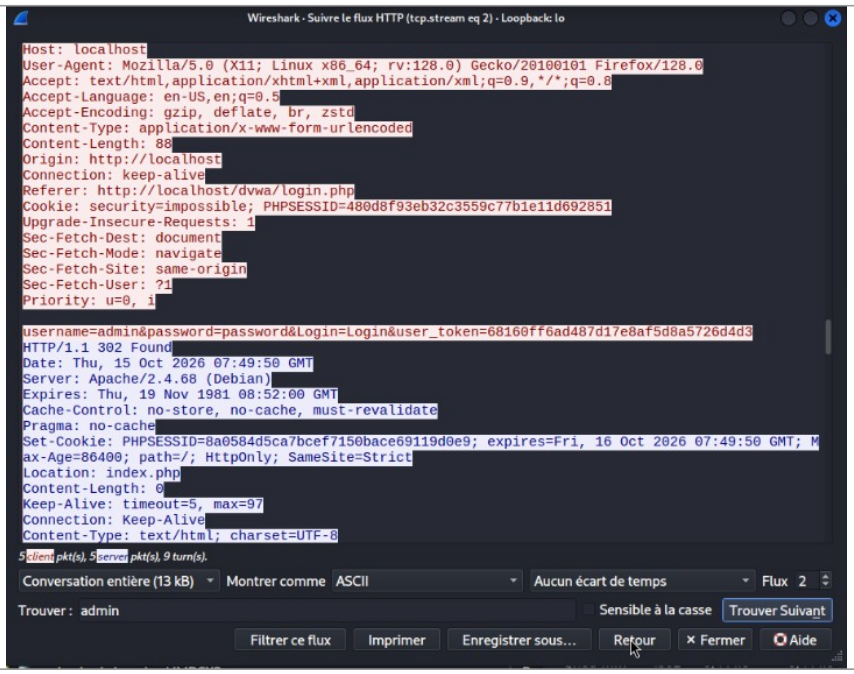
9.1 Suivi du flux HTTP

Un clic droit sur ce paquet donne accès au menu contextuel de Wireshark. L'option Suivre → Flux HTTP reconstitue l'intégralité de l'échange entre le navigateur et le serveur pour cette connexion TCP, requête et réponse comprises :



9.2 Identifiants en clair dans la requête

La reconstruction du flux HTTP expose le corps de la requête POST envoyée par le navigateur. Les champs du formulaire de connexion y apparaissent en texte brut, sans aucun chiffrement, puisque la connexion s'est faite en HTTP simple et non en HTTPS :



- username=admin&password=password&Login=Login&user_token=68160ff6ad487d17e8af5d8a5726d4d3
- Cookie de session : PHPSESSID=480d8f93eb32c3559c77b1e11d692851

- Réponse du serveur : HTTP/1.1 302 Found, redirection vers index.php après authentification réussie

Le nom d'utilisateur (admin) et le mot de passe (password) apparaissent intégralement dans le flux capturé, exactement comme ils ont été saisis dans le formulaire. N'importe quel outil d'écoute réseau positionné sur le même segment — ou, comme ici, sur la machine elle-même — peut donc reconstituer ces identifiants sans avoir besoin de les déchiffrer.

10. Conclusion

Cette démonstration illustre de façon très concrète pourquoi le HTTP simple ne doit jamais être utilisé pour transmettre des formulaires d'authentification : sans chiffrement TLS, un identifiant et un mot de passe transitent en clair et sont immédiatement lisibles dans n'importe quel outil de capture réseau. La bascule vers HTTPS (certificat TLS sur le serveur Apache) et l'usage de mécanismes complémentaires (jetons anti-CSRF, cookies marqués Secure et HttpOnly) permettent de corriger ce type d'exposition. DVWA, conçue comme cible volontairement vulnérable, sert ici de terrain d'entraînement sûr pour observer ce risque sans l'exploiter contre un système réel.

Démonstration réalisée dans un cadre pédagogique, sur une application volontairement vulnérable (DVWA) installée localement à des fins d'entraînement. L'interception de trafic réseau ne doit être pratiquée que sur des systèmes et réseaux dont on est propriétaire ou pour lesquels une autorisation explicite a été obtenue.