

## Activité 3 – Attaque de type injection SQL et mise en place de contre-mesures

Propriétés	Description
<b>Intitulé long</b>	Ce TP a pour objectif de réaliser une injection SQL et d'analyser la mise en place de sources sécurisée pour se protéger de ce type d'attaque.
<b>Formation(s) concernée(s)</b>	BTS Services Informatiques aux Organisations
<b>Matière(s)</b>	Bloc 3 – Cybersécurité des services informatiques
<b>Présentation</b>	Dans ce TP, l'objectif est d'appliquer les bonnes pratiques en matière de codage des applications web en PHP et de prévenir les attaques de type injection SQL.
<b>Compétences</b>	<ul style="list-style-type: none"><li>• Protéger les données à caractère personnel.</li><li>• Garantir la disponibilité, l'intégrité et la confidentialité des services informatiques et des données de l'organisation face à des cyberattaques.</li><li>• Assurer la cybersécurité d'une infrastructure réseau, d'un système, d'un service.</li><li>• Assurer la cybersécurité d'une solution applicative.</li></ul>
<b>Savoirs</b>	<ul style="list-style-type: none"><li>• Typologie des risques et leurs impacts.</li><li>• Cybersécurité : bonnes pratiques, normes et standards.</li></ul>
<b>Prérequis</b>	Connaissances de base concernant l'administration d'un système GNU/Linux et du langage PHP.
<b>Outils</b>	Mutillidae, conteneurs sur Docker (Laboratoire n°2) <a href="https://forge.apps.education.fr/reseau-certa/bts-sio/labos-kali-docker/lab2/">https://forge.apps.education.fr/reseau-certa/bts-sio/labos-kali-docker/lab2/</a>
<b>Mots-clés</b>	Injection SQL, exploitation de vulnérabilités, remédiations, hygiène numérique, respect des bonnes pratiques.
<b>Durée</b>	2h
<b>Auteurs</b>	Cécile Nivaggioni, Patrice DIGNAN avec la relecture de Valérie Martinez et Apollonie Raffalli. Laboratoire sur Docker : Apollonie Raffalli avec les tests de Christelle Thiry

## Préambule

Le TP proposé est uniquement à visée pédagogique. Son objectif est l'analyse de failles liées à l'usage de certains protocoles réseaux afin de proposer une amélioration de la sécurité informatique d'un système d'information et de l'hygiène numérique des étudiants. Il permet également l'acquisition de compétences associées au bloc 3 Cybersécurité SISR du BTS SIO.



**Les outils abordés dans ce support sont uniquement utilisés à des fins éthiques (Ethical Hacking) et pédagogiques. Leur usage est formellement interdit en dehors de ce cadre sur un réseau tiers sans autorisation explicite.**

Pour rappel, l'article 323-1 du code pénal stipule que le fait d'accéder ou de se maintenir, frauduleusement, dans tout ou partie d'un système de traitement automatisé de données est puni de deux ans d'emprisonnement et de 60 000 € d'amende.

Lorsqu'il en est résulté soit la suppression ou la modification de données contenues dans le système, soit une altération du fonctionnement de ce système, la peine est de trois ans d'emprisonnement et de 100 000 € d'amende.

## Plateforme utilisée

**Le TP qui vous est proposé utilise le laboratoire 2 contenant 5 conteneurs pré-configurés. Pour rappel :**

Machine	Nom de domaine pleinement qualifié	Configuration réseau	Applications et services
Serveur sous Debian 12	srvssh.local.sio.fr	Adresse IPv4 : 172.16.10.10/24 Passerelle : 172.16.10.254 Serveur DNS : 172.16.10.10	Service OpenSSH port 22/TCP Service DNS Bind port 53/UDP
Client sous Debian 12	clissh.local.sio.fr	Adresse IPv4 : 192.168.56.11/24 Passerelle : 192.168.56.254 Serveur DNS : 172.16.10.10	Environnement de bureau XFCE Service XRDP port 3389/TCP Client OpenSSH
Attaquant sous Kali Linux	kali.local.sio.fr	Adresse IPv4 : 192.168.56.12/24 Passerelle : 192.168.56.254 Serveur DNS : 172.16.10.10	Environnement de bureau XFCE Service XRDP port 3389/TCP Metasploit Netfilter/Iptables
Routeur sous Debian 12	routeur.local.sio.fr	Adresses IPv4 : eth0 – DHCP eth1 – 192.168.56.254/24 eth2 – 172.16.10.254 Serveur DNS : 172.16.10.10	Netfilter/Iptables
Serveur Metasploitable	srvm.local.sio.fr	Adresse IPv4 : 172.16.10.5/24 Passerelle : 172.16.10.254 Serveur DNS : 172.16.10.10	Service OpenSSH port 22/TCP Service Web port 80:TCP (site « mutillidae »)

**Toutes les machines sont accessibles en SSH (sur le port 22 à partir de l'hôte qui les héberge) et à partir de l'extérieur.**

**La machine Kali Linux et le client Debian bénéficient d'une interface graphique accessible via un bureau à distance (protocole RDP sur le port 3389 à partir de l'hôte qui les héberge) et à partir de l'extérieur.**

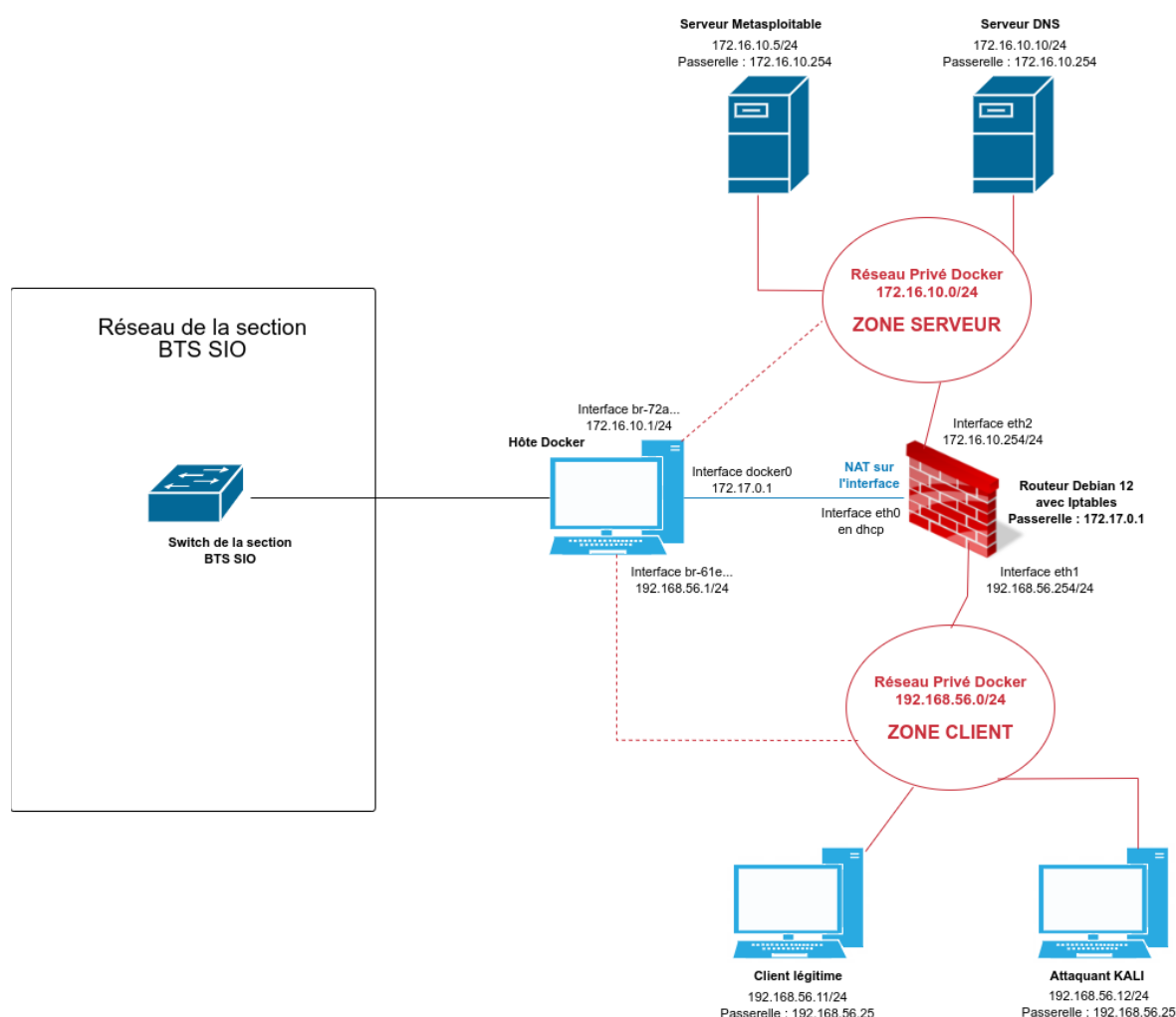
Voici les comptes, les mots de passe et les numéros de ports **accessibles de l'extérieur**, vous permettant d'accéder aux différents conteneurs :

Intitulé de la machine	Nom d'utilisateur	Mot de passe	Ports SSH	Port RDP
Serveur sous Debian 12	etusio	Fghijkl1234*	12222	
Client sous Debian 12	etusio	Fghijkl1234*	22222	23389
Attaquant sous Kali Linux 2023.3	etusio	Fghijkl1234*	32222	33389
Routeur sous Debian 12	etusio	Fghijkl1234*	42222	
Serveur Metasploitable	msfadmin	msfadmin	52222	

Lorsque des commandes nécessitant des privilèges administrateurs seront utilisées, il sera nécessaire d'utiliser la commande **sudo**.

```
etusio@srvssh:~$ sudo service ssh restart
```

Voici une représentation logique de la maquette proposée dans le cadre de ce laboratoire :



🔗 Lancer le laboratoire : **bash gestion\_lab2.sh -c**

🔗 Vérifier que les 5 conteneurs sont actifs : **docker ps**

```
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
0d08664c36d   tleencjr/metasploitable2            "sh -c '/bin/service..." 6 days ago    Up About a minute
4136eb106769   aporaf/kalirolling-lab2             "/lib/systemd/system..." 6 days ago    Up About a minute    22/tcp, 3389/tcp
e879e10e57a3   aporaf/clientdebian12-lab2          "/lib/systemd/system..." 6 days ago    Up About a minute    22/tcp, 3389/tcp
7c1ef0ce2e65   aporaf/serveurdebian12-lab2         "/lib/systemd/system..." 6 days ago    Up About a minute    22/tcp, 53/tcp, 53/udp
13a1f9aae620   aporaf/routeurdebian12-lab2         "/lib/systemd/system..." 6 days ago    Up About a minute    0.0.0.0:12222->12222/tcp, :::12222->12222/tcp, 0.0.0.0:22222->22222/tcp, :::22222->22222/tcp, 0.0.0.0:23389->23389/tcp, :::23389->23389/tcp, 0.0.0.0:32222->32222/tcp, :::32222->32222/tcp, 0.0.0.0:33389->33389/tcp, :::33389->33389/tcp, 0.0.0.0:52222->52222/tcp, :::52222->52222/tcp, 0.0.0.0:42222->42222/tcp, :::42222->42222/tcp
routeur-lab2
```

## Réalisation de l'attaque

### Découverte des hôtes et services présents sur un réseau local

En tant qu'attaquant, la première étape consiste en général à recueillir des informations sur le réseau dans lequel nous nous trouvons. Ainsi, à l'aide de l'outil nmap présent sur Kali Linux, il est possible de réaliser un scan des réseaux logiques locaux :

```
etusio@kali:~$ nmap -sP 192.168.56.0/24
etusio@kali:~$ nmap -sP 172.16.10.0/24
```

Puis de scanner les différents hôtes afin de savoir quels ports sont ouverts sur ceux-ci et quels services sont proposés. Par exemple, sur la zone des serveurs :

```
etusio@kali:~$ nmap -sV 172.16.10.5
etusio@kali:~$ nmap -sV 172.16.10.10
etusio@kali:~$ nmap -sV 172.16.10.254
```

L'extrait du scan du serveur 172.16.10.5 montre que les services Apache2 et MySQL sont à l'écoute :

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-15 00:36 CEST
Nmap scan report for 172.16.10.5
Host is up (0.00013s latency).
Not shown: 980 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login        OpenBSD or Solaris rlogind
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  landesk-rc   LANDesk remote management
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8180/tcp  open  unknown     ISC BIND 9.4.2
```

Dans ce TP nous allons nous intéresser à ces services, en particulier à la notion d'injection SQL pour la récupération de données stockées en base.

**Pour rappel**, la machine scannée est un serveur Metasploitable (version 2.0.0). Il s'agit d'une distribution Linux (Ubuntu) intentionnellement vulnérable pour apprendre à tester les principales vulnérabilités. **C'est sur ce serveur qu'est disponible l'application web pédagogique Mutillidae (du groupe OWASP) à partir duquel vous allez travailler.**

## MISE À JOUR DE L'APPLICATION MUTILLIDAE

Pour ce TP, nous allons travailler à partir de la base de données « owasp10 ». Pour cela, une modification de configuration sur la machine virtuelle « Mutillidae 2 » est nécessaire.

➤ Vérifier, en ligne de commande, l'existence de cette base de données.

Voici un rappel de commandes de base pour l'administration de bases de données d'un serveur MySQL :

- `SHOW DATABASES ;` ⇒ Liste les bases disponibles.
- `USE nom_de_la_base ;` ⇒ Définit la base de données à utiliser (sur laquelle se positionner).
- `SHOW TABLES ;` ⇒ Liste les tables d'une base de données.
- `DESCRIBE nom_de_la_table ;` ⇒ Affiche la structure d'une table (champs et types).

➤ Donner le schéma relationnel de cette base de données.

Pour l'application web Mutillidae, les paramètres d'accès à la base de données sont définis dans le fichier « config.inc » situé à la racine des sources de l'application.

➤ Mettre à jour ces paramètres pour utiliser la base de données « owasp10 » (les autres paramètres ne sont pas à modifier).

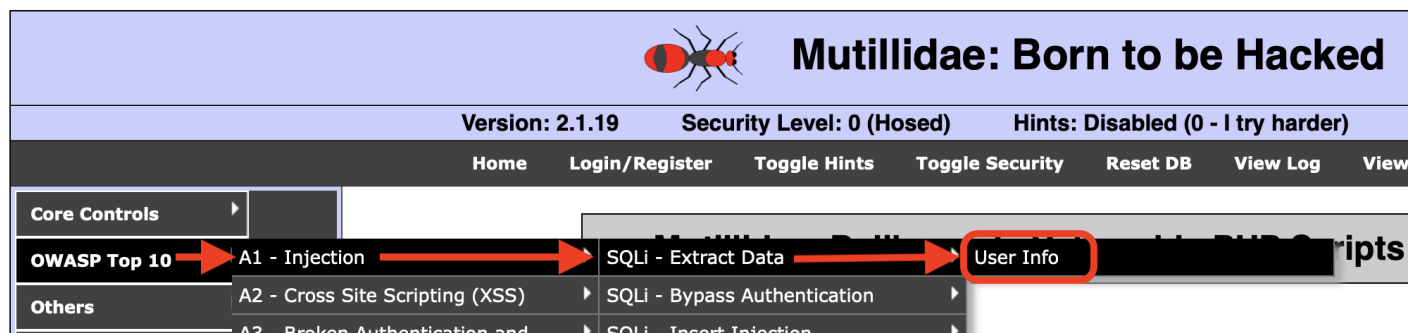
*Pour pouvoir utiliser nano saisir « export TERM=xterm ». Vous pouvez également utiliser vim.tiny.*

## Réalisation d'une injection SQL

L'accès à l'application Mutillidae se fait ici à partir de l'URL <http://172.16.10.5/mutillidae/>.

➤ Se connecter sur cette application.

Pour tester l'injection SQL, vous devez vous positionner sur le menu « Owasp Top 10 → A1 Injection → SQLi Extract Data → « User info » :



L'application mise à disposition dans ce cas est un formulaire chargé d'afficher les informations de l'utilisateur qui se connecte :

**Please enter username and password  
to view account details**

**Name**

**Password**

**View Account Details**

Un attaquant va dans un premier temps essayer de récupérer des informations sur la structure de la base de données en tentant de provoquer une erreur de syntaxe au moment de l'exécution de la requête SQL utilisée pour valider l'authentification sur l'application.

Pour cela, un exemple d'identifiants peut être :

- login : **hacker**
- mot de passe : '



**Attention** : la valeur du mot de passe est bien une apostrophe.



Tenter une connexion avec ces paramètres.



Expliquer en quoi les informations affichées peuvent être utiles pour un attaquant.

Les paramètres de connexion précédent (avec un caractère spécial, l'apostrophe) étant autorisés, l'attaquant sait qu'il va pouvoir tenter une attaque de type injection SQL.

Le principe est d'injecter une opération toujours vraie afin que la requête SQL exécutée au moment de la validation de l'authentification soit correcte même si le login n'existe pas dans la base.

Vous allez tester les identifiants suivants :

- login = **hacker**
- mot de passe = '**or 'a' = 'a**



Tenter une connexion avec ces paramètres.



Expliquer l'affichage obtenu.

## Mise en place d'une contre-mesure

- Positionner le niveau de sécurité de l'application Mutillidae à 5 en cliquant deux fois sur le bouton « Toggle Security » (barre de menu haute).



- Tenter à nouveau l'injection SQL précédente et expliquer le résultat obtenu.

Le code source gérant l'authentification sur l'application Mutillidae est dans la page « login.php » (dans /var/www/mutillidae).

- Comparer le code dans sa version sécurisée et dans sa version non sécurisée.
- Expliquer comment le code sécurisé de la page login.php permet d'empêcher l'injection SQL.
- Proposer une conclusion sur l'intérêt d'un codage sécurisé au regard des principes fondamentaux de la sécurité (disponibilité, intégrité et confidentialité).