Série TD N° 3

Exercice 01 :

Les attaques par injection SQL représentent une menace sérieuse pour tout site utilisant sur une base de données. Les méthodes derrière une attaque sont faciles à apprendre et les dommages causés peuvent aller d'une compromission considérable à une compromission complète du système. Malgré ces risques, un nombre important de systèmes sur Internet sont sensibles à cette forme d'attaque.

Considérez une application web d'achat en ligne qui affiche des produits dans différentes catégories. Lorsque l'utilisateur clique sur la catégorie « Gifts », par exemple, son navigateur web demande l'URL :

https://insecure-website.com/products?category=Gifts

Cela amène l'application à effectuer la requête SQL suivante pour récupérer les détails des produits concernés dans la base de données :

SELECT id, nom, category, price FROM products WHERE category = 'Gifts' AND released = 1

Dont la requête originale est :

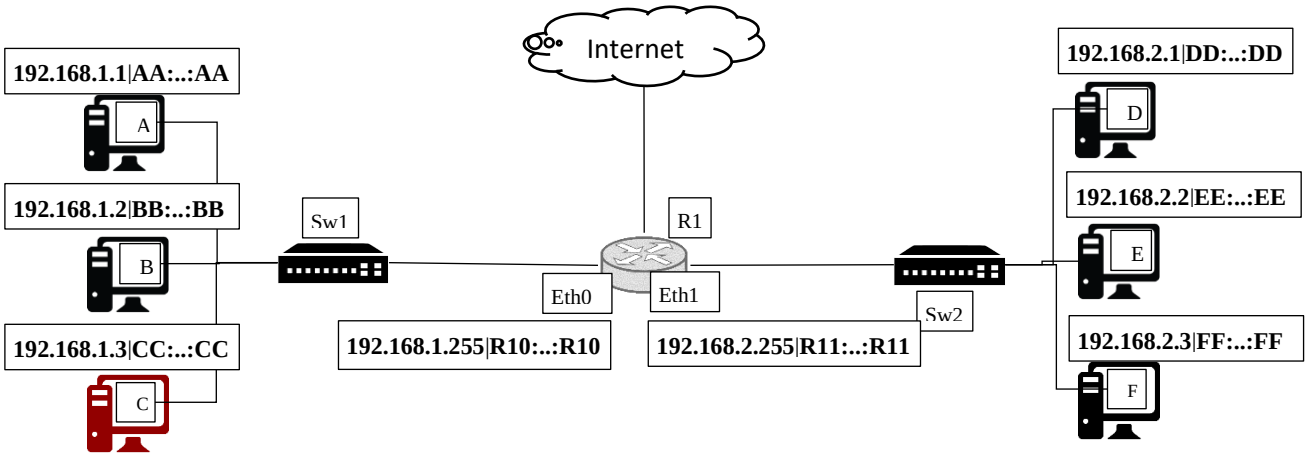
SELECT id, nom, category, price FROM products WHERE category = '"$category"' AND released = 1

Cette requête SQL demande à la base de données de renvoyer **tous les détails (id, nom, category, price)** de la **table des produits** où l**a catégorie est Gifts** et le **produit soit publié (released est 1)**.

1. Comment vérifier que le site est vulnérable à l'attaque injection SQL.
2. Construire une attaque qui permet d'afficher tous les produits de n'importe quelle catégorie, y compris des catégories qu'il ne connaît pas.
3. Les résultats de la requête SQL sont renvoyés et affichés dans les réponses de l'application.
4. Construire une attaque qui permet de récupérer des données à partir d'autres tables de la base de données. Récupérer les données de la table **user** (**id, name, utilisateur, password, type_account, date**).

Exercice 02:

Soit le réseau câblé suivant :



1. Quel est le rôle du protocole ARP ? Expliquez comment les tables ARP sont mises à jour ?
2. Quels sont les trames que la machine **C** peut sniffer ? Expliquez ?
3. Quel est le principe de l'attaque ARP spoofing ?

4. Quels sont les machines qui peuvent lancer une attaque ARP spoofing sur le réseau relié à l'interface 0 (eth0) du routeur **R1** ? Expliquez ?

La machine **C** veut intercepter tout le trafic qui circule sur le réseau relié à l'interface 0 (eth0), grâce à une attaque ARP spoofing.

5. Donnez l'état de la table ARP des machines **B** et **R1** avant l'attaque ?
6. Donnez les étapes que la machine **C** doit effectuer pour réaliser l'attaque ?
7. Une fois l'attaque achevé, décrivez l'état des tables ARP des machines **B**, **R1** et **C**.

Exercice 03 :

On rappelle que l'attaque **IP spoofing** consiste pour un pirate à se faire passer pour une machine B auprès d'une machine A. L'attaque se compose de trois étapes :

- Le pirate paralyse la machine B.
- Le pirate devine le procédé utilisé par A pour générer ses numéros de séquence initiaux (ISN).
- Le pirate se fait passer pour B auprès de A.

1. Que se passerait-il si le pirate ne paralysait pas la machine B ?
2. Pourquoi est-il nécessaire de déterminer la manière dont A génère ses ISN ?
3. Quel peut être l'intérêt pour le pirate de se faire passer pour la machine B ?
4. Représenter les différentes étapes de l'attaque sur un schéma.

Exercice 04 :

Quelle valeur le programme en C suivant va-t-il afficher ? Pourquoi ?

```c
#include <stdio.h>
void main()
{
   char buffer[10];
   char *ptr;
   buffer[0] ='A';
   buffer[1] ='B';
   buffer[2] ='C';
   buffer[3] ='D';
   ptr = buffer + 2;
   *ptr ='Z';
   printf("%c %c %c %c\n", buffer[0], buffer[1], buffer[2], buffer[3]);
}
```

Exercice 05 :

1. Quelle valeur le programme en C ci-dessous va-t-il afficher ? Pourquoi ?
2. Dessiner un digramme de la pile en considérant que toutes les variables sont alignées sur des multiples de 4 octets et que les adresses sont stockées sur 4-octets.
3. A quoi correspondant les valeurs 12 et 10 dans la procédure "function" ?

```c
#include <stdio.h>
void function(int a, int b, int c)
{
```

```
4    char buffer1[5];
5    char buffer2[10];
6    char *ptr;
7    ptr = buffer1 + 12;
8    *ptr += 10;
9  }
10 void main()
11 {
12   int x;
13   x = 0;
14   function(1,2,3);
15   x = 1;
16   printf("%d\n", x);
17 }
```