



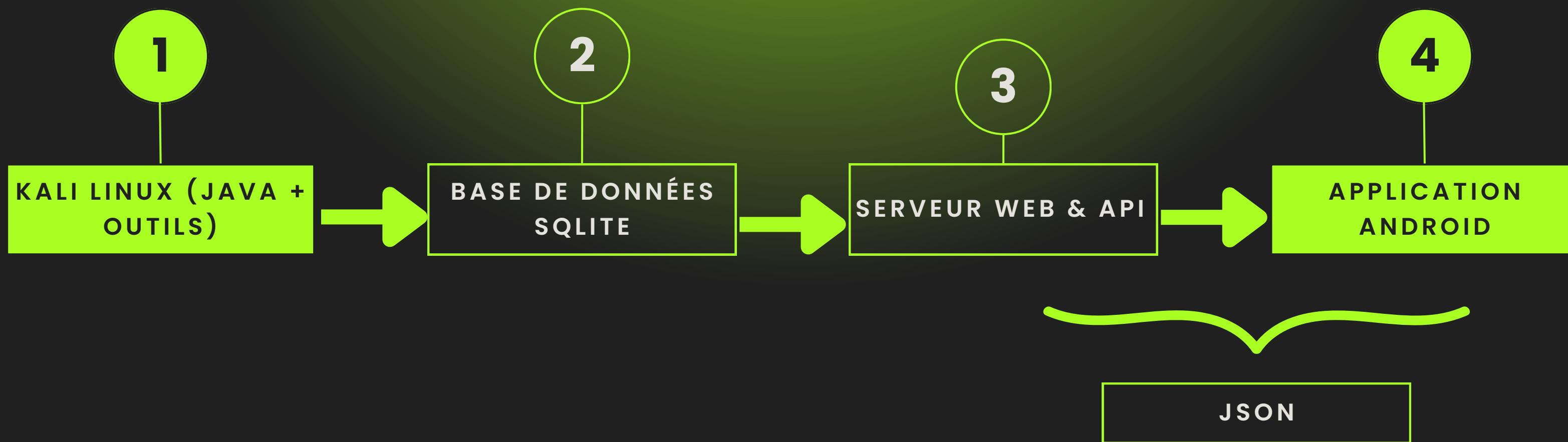
Présentation

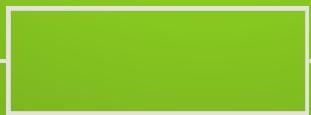
DÉVELOPPEMENT D'APPLICATIONS COMMUNICANTES

SAE302 Badaoui, Samperez, Cruz-Mermyn



ARCHITECTURE GLOBALE





LE CŒUR DU SYSTÈME : SCANNER JAVA

```
[root@kali)-[/opt/SAE32/src]
# # Exécution du programme
java -cp "../lib/sqlite-jdbc-3.51.0.0.jar::" App
[BDD] Connexion ouverte : /var/www/html/failles.db
Outil enregistré : Nmap Scanner
Outil enregistré : Gobuster (Dir Enum)
Outil enregistré : Nikto Web Scanner
✓ Application démarrée.
✓ Outils chargés : Nmap, Gobuster.
📁 Base de données : /var/www/html/failles.db

— MENU SAE302 TD2/TD3 —
1 - Créer la table (si besoin)
2 - Lancer détection simulée (TD2)
3 - Lister toutes les failles
4 - Chercher une faille par ID
5 - Filtrer par sévérité
6 - Enregistrer DummyTool (Test)
7 - Lister outils enregistrés
8 - Lancer un FULL SCAN (Nmap + Gobuster)
9 - Ajouter une cible (IP)
10 - Supprimer une faille par ID
11 - Mettre à jour une faille (ID)
0 - Quitter
Choix: ■
```

1

PILOTAGE SYSTÈME : UTILISATION DE
PROCESSBUILDER POUR EXÉCUTER
NMAP/NIKTO.

2

PARSING INTELLIGENT : ANALYSE DU TEXTE
BRUT POUR DÉTECTER LES MOTS-CLÉS
("OPEN", "CRITICAL").

3

AUTOMATISATION : STOCKAGE DIRECT EN
BASE SQLITE SANS INTERVENTION HUMAINE.



LA PASSERELLE : API & WEB

```
if ($target_ip) {
    // — CAS 2 : ON A CLIQUÉ SUR UNE IP → On affiche ses failles —
    $query = "SELECT * FROM failles WHERE ip = :ip ORDER BY id DESC";
    $stmt = $db->prepare($query);
    $stmt->execute(['ip' => $target_ip]);
    $failles = $stmt->fetchAll(PDO::FETCH_ASSOC);
} else {
    // — CAS 1 : ACCUEIL → On affiche la liste des machines —
    // On groupe par IP et on compte le nombre de failles pour chaque machine
    $query = "SELECT ip, COUNT(*) as count, MAX(dateDetection) as last_seen FROM failles GROUP BY ip ORDER BY last_s
    $result = $db->query($query);
    $machines = $result->fetchAll(PDO::FETCH_ASSOC);
}
```

```
▼ 0:
  id:          44
  nom:         "Vulnérabilité Web (Config)"
  ▼ description: "8102 requests: 0 error(s) and 5 item(s) reported on remote host"
    ip:          "192.168.204.128"
    severite:    "LOW"
    source:      "Nikto"
    dateDetection: "2025-12-24"
```

RÔLE PIVOT

Le script PHP expose les données de la base SQLite sur le réseau HTTP.

FORMAT JSON

Standardisation des données pour qu'elles soient lisibles par n'importe quel client (Android, iOS, Web).



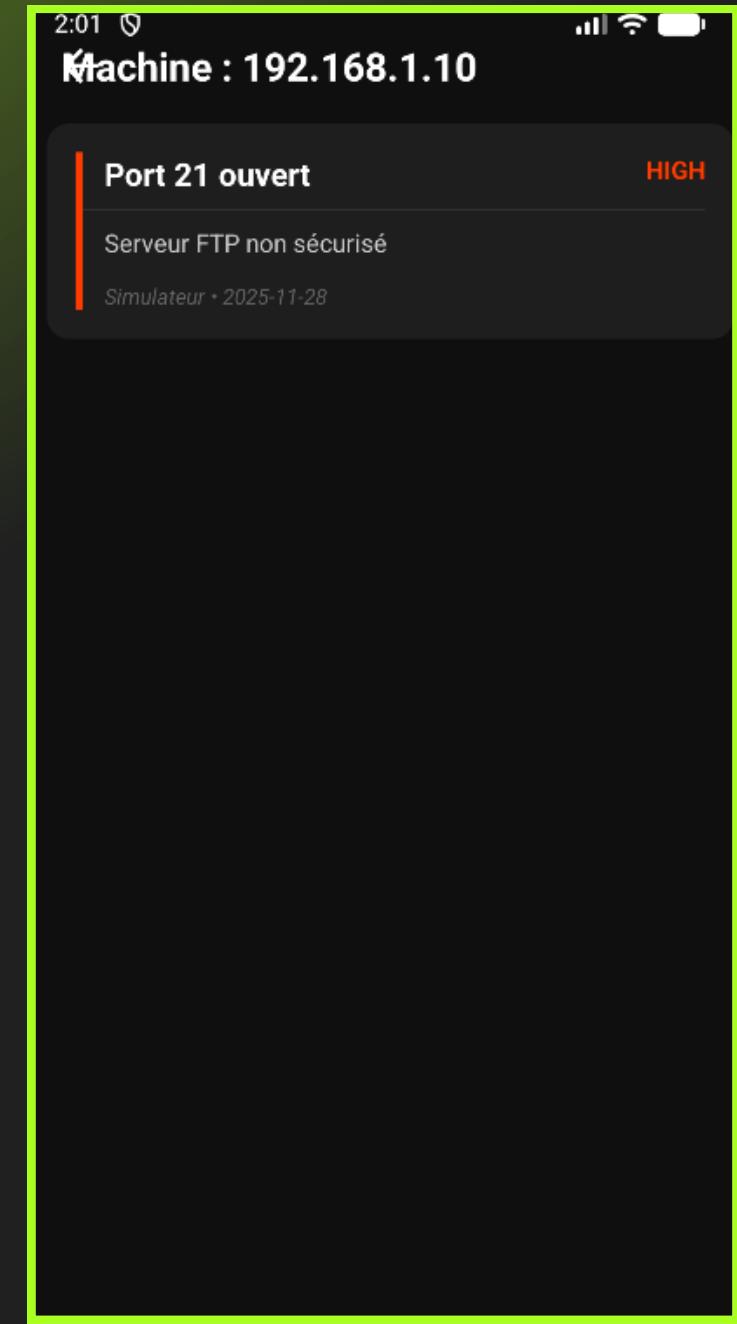
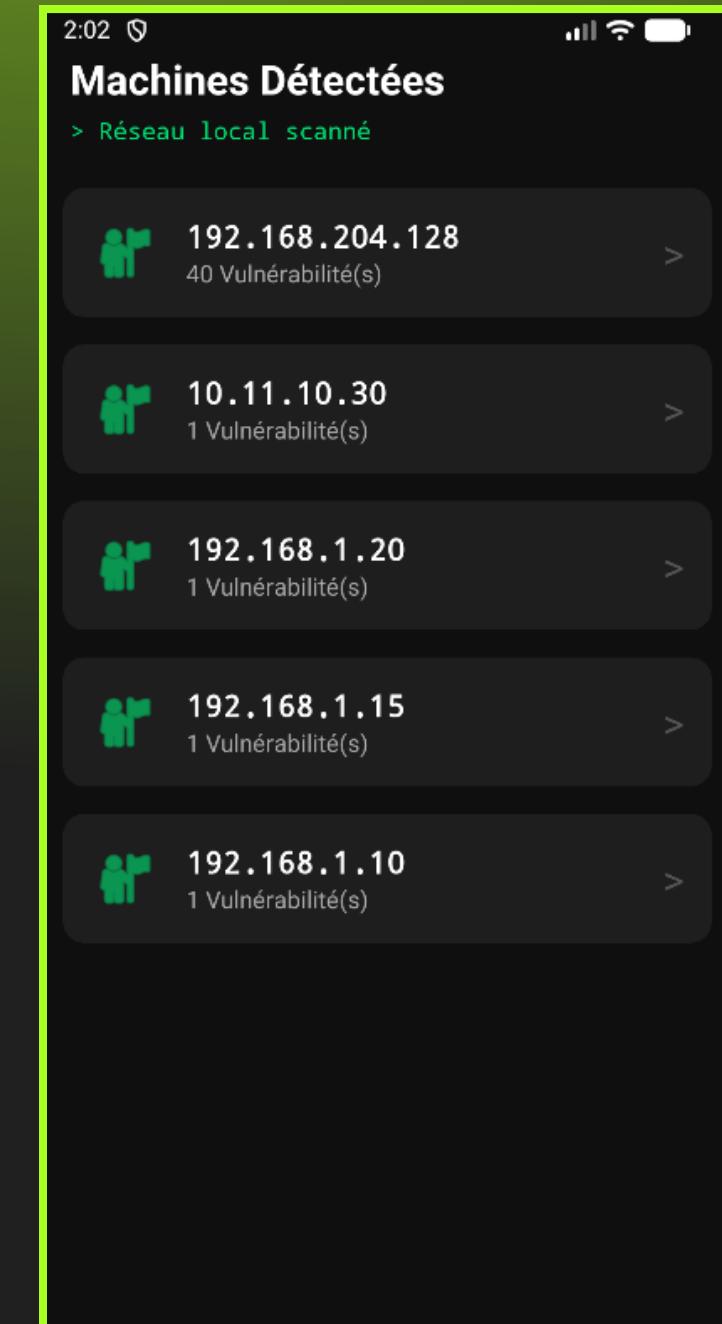
L'INTERFACE MOBILE (ANDROID)

1. COMMUNICATION RÉSEAU (RETROFIT)

Utilisation de la librairie Retrofit pour consommer l'API JSON avec l'ip du serveur web.

AFFICHAGE OPTIMISÉ (RECYCLERVIEW)

Implémentation du pattern Adapter & ViewHolder via des RecyclerView. Cela permet d'afficher des listes de centaines de failles de manière fluide sans consommer toute la mémoire du téléphone.





WALID

**RESPONSABLE DE TOUT LE BLOC
BACKEND/SYSTÈME (JAVA ET LA
CRÉATION DE LA BDD)**

ALEXANDRE

**SUR LA COUCHE INTERMÉDIAIRE (LE
SERVEUR WEB ET L'API QUI EXPOSE
CETTE BDD)**

JULIEN

**S'EST CONCENTRÉ SUR LE FRONTEND
MOBILE**

ORGANISATION & RÉPARTITION

Activité	Walid	Alexandre	Julien
Développement Java Parse	R	A	I
Développement Java Scan	A	R	I
Développement site web	C	R	A
Conception de la BdD	R	A	I
Gestion de projet (Github, RACI, Rapport)	R	I	A
Dev appli Android	A	I	R



CHALLENGES TECHNIQUES



MAÎTRISE DE L'ÉCOSYSTÈME JAVA

La difficulté était d'appréhender l'ampleur des notions à connecter (Commandes système, SQL, POO) sans connaissances préalables.



INTERCONNEXION WEB / MOBILE

Nous avions l'interface Web et le Mobile, mais aucune méthode claire pour synchroniser les données entre ces deux environnements différents.



TRAITEMENT DES DONNÉES

Les outils de sécurité (Nmap) génèrent des rapports en texte brut, impossibles à stocker proprement en base de données.



FIN



MERCI POUR VOTRE ATTENTION !

DEMO ?