

# SÉCURITÉ DES APPLICATIONS

École Supérieure de Gestion, d'Informatique et des Sciences  
Informatique Option Architecture logicielle

Dr Nelson Saho<sup>1</sup>  
saho.nelson@gmail.com

<sup>1</sup>Institut de Formation et de Recherche en Informatique

13 janvier 2025

# Contenu

- 1 Sécurisation des Applications Mobiles
- 2 Les outils de sécurisation des applications

# Contenu

## 1 Sécurisation des Applications Mobiles

## 2 Les outils de sécurisation des applications

# Contexte et enjeux

- **Croissance des applications mobiles** : Les applications mobiles sont omniprésentes dans divers secteurs (santé, finance, commerce, etc.). Cette popularité en fait une cible privilégiée pour les cyberattaques
- **Statistiques** : Selon les rapports de sécurité, plus de 70 % des applications mobiles présentent des vulnérabilités exploitables.

# Principales menaces

- **Attaques par injection** : SQL, XSS ou code malveillant.
- **Exploitation des failles réseau** : Interception des données via des attaques de type man-in-the-middle (MITM).
- **Malwares mobiles** : Applications malveillantes déployées sur les appareils des utilisateurs.
- **Reverse engineering** : Décompilation et analyse du code source pour trouver des failles.

# Spécificités Android et iOS

- **Android** : Système ouvert, plus personnalisable mais plus exposé.
- **iOS** : Système fermé, plus sécurisé mais pas infaillible.

# Modélisation des menaces, Authentification et autorisation

- **Définir le périmètre à protéger** : Données sensibles, communications, accès utilisateur.
- **Outils de modélisation** : STRIDE, DREAD.
- **Bonnes pratiques** :
  - Utilisation d'authentification multi-facteurs (MFA).
  - Gestion des sessions avec expiration automatique.
- **Exemples** : *OAuth2* pour la gestion des accès, JSON Web Tokens (JWT).

# Protection des données sensibles

## ■ Chiffrement des données :

- Données en transit : Utilisation de TLS (Transport Layer Security).
- Données au repos : Chiffrement AES 256.

## ■ Stockage sécurisé :

- Android : Keystore pour les clés cryptographiques.
- iOS : Keychain pour les mots de passe et données sensibles.



# Développement sécurisé pour Android

## ■ Gestion des permissions

- Réduire les permissions demandées à celles strictement nécessaires.
- Utiliser des permissions contextuelles (runtime permissions).

## ■ Protection des composants Android

- *Intents* : Utiliser des intents explicites pour empêcher les attaques de redirection.
- *Services* : Protéger les services avec des permissions appropriées.

## ■ Obfuscation et anti-reverse engineering

- Utilisation de ProGuard ou R8 pour obscurcir le code.
- Intégration de techniques anti-tampering.

# Développement sécurisé pour iOS

## ■ Stockage des données sensibles

- Utiliser *Keychain* pour stocker les informations sensibles comme les mots de passe.
- Éviter de stocker des données sensibles dans *UserDefaults*.

## ■ Sécurisation des communications

- Activer *App Transport Security* (ATS) pour exiger des connexions HTTPS.
- Utiliser des certificats SSL/TLS pour chiffrer les communications.

## ■ Problématiques liées au Jailbreaking

- Détecter les appareils jailbreakés et limiter leur accès.
- Utiliser des bibliothèques tierces pour effectuer des vérifications de sécurité.

# Développement sécurisé pour iOS

## ■ Introduction aux tests de sécurité

- Importance des tests périodiques pour identifier les vulnérabilités.
- Cadres de référence : OWASP Mobile Application Security Testing Guide (MASTG).

## ■ Outils de tests

- **Android** : *Mobile Security Framework* (MobSF), *Drozer* pour les analyses de failles.
- *iOS* : *Frida* pour l'analyse dynamique, *Objection* pour l'exploitation des vulnérabilités courantes.

# Conformité et sensibilisation

## ■ Normes et réglementations

- *RGPD* : Protection des données personnelles en Europe.
- *PCI DSS* : Normes de sécurité pour les applications traitant les paiements.

## ■ Sensibilisation au développement sécurisé

- Former les développeurs aux principes de DevSecOps.
- Encourager l'intégration de la sécurité dans le cycle de vie du développement logiciel (SDLC).

# Conclusion

- La sécurisation des applications mobiles est une étape critique pour protéger les données utilisateur et préserver la confiance des clients.
- Elle nécessite une combinaison d'architecture sécurisée, de bonnes pratiques de développement, de tests rigoureux et de conformité aux normes.
- En restant vigilant face aux nouvelles menaces, les développeurs peuvent garantir des applications mobiles plus sécurisées.

# Contenu

1 Sécurisation des Applications Mobiles

2 Les outils de sécurisation des applications

Ces outils détectent les vulnérabilités dans le code, les dépendances ou les configurations.

## SAST (Static Application Security Testing)

- Analyse le code source à la recherche de failles.
- Exemples
  - SonarQube : Analyse statique de code avec des règles de sécurité.
  - Fortify Static Code Analyzer : Détection avancée de vulnérabilités dans le code source.
  - Checkmarx : Identifie les problèmes de sécurité au niveau du code.

## DAST (Dynamic Application Security Testing)

- Analyse les applications en cours d'exécution pour détecter les vulnérabilités.
- Exemples
  - OWASP ZAP : Scanner gratuit pour tester les failles d'une application web.
  - Burp Suite : Outil de test d'intrusion pour applications web.



## IAST (Interactive Application Security Testing)

- Combine SAST et DAST en surveillant une application pendant son exécution.
- Exemples
  - Contrast Security : Intègre des tests de sécurité interactifs dans les applications.
  - Seeker : Détecte les vulnérabilités au moment de l'exécution.

Les dépendances non sécurisées sont une source majeure de failles.

### Exemples d'outils :

- OWASP Dependency-Check : Identifie les bibliothèques vulnérables.
- Snyk : Analyse et corrige les vulnérabilités des dépendances.
- npm audit (pour Node.js) : Vérifie les vulnérabilités dans les packages npm.
- pip-audit (pour Python) : Analyse des dépendances Python.

Pour protéger les données sensibles dans les applications.

## Cryptographie et gestion des clés

- HashiCorp Vault : Stockage et gestion des secrets.
- AWS Key Management Service (KMS) : Service de gestion des clés dans le cloud.
- Azure Key Vault : Gestion des certificats, des clés et des secrets.

## Masquage des données et tokenisation

- Protegrity : Tokenisation et chiffrement des données sensibles.
- Informatica Data Masking : Masquage de données pour les environnements non sécurisés.

Pour détecter et prévenir les attaques en cours.

## WAF (Web Application Firewall)

- Filtre et surveille le trafic HTTP pour protéger contre les attaques courantes comme les injections SQL ou les attaques XSS.
- Exemples
  - ModSecurity : WAF open source.
  - Cloudflare WAF : Solution de protection des applications web basée sur le cloud.
  - AWS WAF : Protection des applications exécutées sur AWS.

## RASP (Runtime Application Self-Protection)

- Intègre la sécurité dans l'application pour surveiller et bloquer les attaques en temps réel.
- Exemples
  - Imperva RASP : Sécurise les applications à l'exécution.
  - AppDynamics RASP : Solution RASP intégrée à la surveillance des performances.

Garantit que seules les personnes autorisées accèdent aux ressources.

### Exemples d'outils :

- Auth0 : Plateforme d'authentification et d'autorisation.
- Okta : Gestion des identités et des accès pour les applications.
- Keycloak : Solution open source pour l'authentification et l'autorisation.

Les API sont souvent la cible des attaquants.

### Exemples d'outils :

- Postman API Security Testing : Tests de sécurité pour les API.
- OWASP API Security Project : Outils et recommandations pour sécuriser les API.
- 42Crunch : Protection et test de sécurité des API.

Intégration de la sécurité dans le pipeline CI/CD.

### Exemples d'outils :

- GitLab Security Scanner : Analyse de sécurité intégrée à GitLab.
- Trivy : Scanner de sécurité pour conteneurs, dépendances et infrastructures.
- Aqua Security : Protection des conteneurs et du cloud.
- JFrog Xray : Analyse des vulnérabilités dans les dépendances.



Pour tester la résistance des applications face à des menaces connues.

### Exemples d'outils :

- OWASP Top 10 : Liste des 10 vulnérabilités les plus critiques (SQL Injection, XSS, CSRF, etc.).
- Metasploit : Framework d'exploitation des vulnérabilités.
- SQLMap : Test d'injection SQL.

Les outils seuls ne suffisent pas ; il faut sensibiliser les équipes.

### Exemples d'outils :

- Hack The Box : Simulations d'attaques et défis de cybersécurité.
- TryHackMe : Formations interactives sur la sécurité.
- OWASP Juice Shop : Application web vulnérable pour apprendre à identifier et corriger les failles.

Des outils pour auditer la sécurité des applications selon des normes (ISO, GDPR, PCI-DSS).

### Exemples d'outils :

- Nessus : Scanner de vulnérabilités pour les applications et réseaux.
- OpenSCAP : Vérification de conformité pour Linux et applications.
- Qualys : Plateforme de gestion des vulnérabilités.

Pour sécuriser efficacement les applications, il est important d'utiliser une combinaison de :

- Outils d'analyse (SAST, DAST, IAST)
- Solutions de protection (WAF, RASP, IAM)
- Sécurisation des API et gestion des données sensibles
- Sensibilisation des équipes.

# Thanks for your attention.

# ?