Programmation sécurisée (Secure coding)
Triade CIA

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

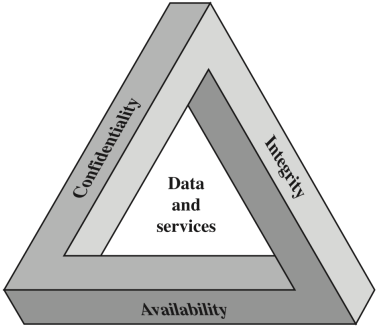
révision : 0.1

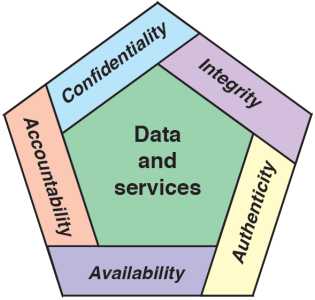
Plan

Introduction aux tests de sécurité

- L'élément clé de la sécurité est la **Triade de la Sécurité de l'Information** (*InfoSec Triad*), également appelée **Triade CIA**. Elle se compose des trois attributs suivants d'un système :
  - **Confidentialité** : Aucun utilisateur non autorisé ne peut lire les données.
  - **Intégrité** : Aucun utilisateur non autorisé ne peut modifier les données.
  - **Disponibilité** : Le système doit être accessible aux utilisateurs autorisés pour lire et écrire des données.
- Un système qui satisfait ces trois éléments en toutes circonstances est considéré comme sécurisé.
- En pratique, pour la plupart des logiciels, il peut souvent y avoir des circonstances particulières où un ou plusieurs de ces éléments ne sont pas respectés.

Au-delà de la Triade CIA





La sécurité : un élément fondamental

- La sécurité n'est pas quelque chose que l'on ajoute ...
- ... c'est quelque chose qui est intégré dès le départ, tout comme :
  - la qualité,
  - la scalabilité,
  - et la performance.

Les tests de sécurité sont particuliers

- Les tests de sécurité se distinguent des autres types de tests logiciels par la présence d'un adversaire intelligent -- ou plutôt de nombreux adversaires, bien que tous ne soient pas vraiment « intelligents » -- cherchant également des failles.
- Vous pouvez considérer les pirates malveillants qui tentent de s'introduire dans votre système comme une variante de testeurs :
  - Ils testent constamment les points faibles de votre logiciel.
  - Leur objectif est d'exploiter ces failles pour accéder à votre système ou voler des données.

Les défis des tests de sécurité

- Les tests de sécurité exigent de penser comme votre adversaire.
- Pour tester efficacement un système, vous devez arrêter de penser comme une personne honnête.
- À la place, adoptez la perspective de quelqu'un prêt à user de stratagèmes pour accéder au système de manière inattendue.
- En effet :
  - Si des vulnérabilités pouvaient être découvertes par une utilisation classique, elles auraient probablement déjà été identifiées et corrigées.
  - Les défauts les plus dangereux exploitent le système de façons que les utilisateurs normaux ne peuvent même pas concevoir.

Les utilisateurs : une vulnérabilité impossible à corriger

- Même si vous avez identifié et corrigé toutes les vulnérabilités dans le code de votre système pour le rendre parfaitement sécurisé. . .
- ...tout votre travail peut être réduit à néant par une simple attaque d'ingénierie sociale, par exemple :
  - Un individu appelle l'un de vos utilisateurs en prétendant être de l'équipe d'administration système et lui demande de confirmer son mot de passe.
- Cela arrive plus souvent qu'on ne le pense :
  - Beaucoup de personnes sont déroutées par la technologie et la sécurité de l'information.
  - D'autres ne prêtent simplement pas attention à ce qu'on leur demande.
- Comme le dit *Georgia Weidman*, auteure du livre **Penetration Testing** : « **Les utilisateurs sont une vulnérabilité qu'on ne pourra jamais corriger.** »

Tests de sécurité : normes et bonnes pratiques (1)

- **Famille ISO/IEC 27000 : Systèmes de gestion de la sécurité de l'information**
  - Fournit une vue d'ensemble des normes interconnectées, publiées ou en cours de développement.
  - Comprend des composants structuraux significatifs.
- **ISO 27001 : Système de gestion de la sécurité de l'information**
  - Définit les exigences pour établir, mettre en œuvre, maintenir et améliorer continuellement un système de gestion de la sécurité de l'information (SGSI).
- **ISO 27002 : Code de bonnes pratiques**
  - Introduit un ensemble de contrôles pour la sécurité de l'information.
- **British Standard 7799 Partie 3 (BSI Group)**
  - Publie des lignes directrices pour la gestion des risques liés à la sécurité de l'information.
- **COBIT (Control Objectives for Information and related Technology) : Gouvernance et gestion des TI**
  - Cadre de contrôle publié par ISACA (Standards Board of Information Systems Audit and Control Association) pour la gouvernance et la gestion des technologies d'entreprise.

Tests de sécurité : normes et bonnes pratiques (2)

- **Common Criteria (ISO/IEC 15408)**
  - Ensemble de critères d'évaluation développés en alignement avec les organisations nationales de sécurité de pays tels que l'Australie, le Canada, la France, l'Allemagne, le Japon, les Pays-Bas, la Nouvelle-Zélande, l'Espagne, le Royaume-Uni et les États-Unis.
- **ITIL (ISO/IEC 20000)**
  - Introduit un ensemble de bonnes pratiques en gestion des services informatiques (ITSM).
  - Se concentre sur les processus de service IT et le rôle central de l'utilisateur.
- **National Information Security Technology Standard Specification**
  - Collection de normes nationales formulées par le Comité technique des normes nationales de sécurité de l'information.
  - Comprend la gestion de la sécurité, l'évaluation de la sécurité, l'authentification et l'autorisation, etc.
- **SANS Security Policy Resource**
  - Publiées par le SANS Institute pour un développement et une mise en œuvre rapides des politiques de sécurité de l'information.

Types d'attaques sur la sécurité d'un système

- **Attaques actives :**
  - Modifient le système attaqué de manière directe.
  - Exemples :
    - Ajouter un programme qui s'exécute en arrière-plan.
    - Changer les mots de passe des utilisateurs.
    - Modifier les données stockées sur le système.
  - Les attaques actives causent souvent plus de dégâts, mais elles sont généralement plus faciles à détecter.
- **Attaques passives :**
  - N'entraînent aucun changement dans le système.
  - Exemples :
    - Intercepter le trafic réseau.
    - Surveiller les réseaux sans fil non sécurisés.
  - Les attaques passives sont souvent difficiles à observer.

Attaque par interruption : une attaque sur la disponibilité

- **Objectif :** Réduire ou éliminer la disponibilité d'un système.
- **Exemples simples :**
  - Accès physique à un bâtiment pour débrancher tous les serveurs du réseau.
- **Exemples avancés :**
  - Envoi massif de requêtes non autorisées, empêchant les requêtes légitimes de passer (*attaque par déni de service - DoS*).
  - Modification des mots de passe de tous les utilisateurs, rendant le système inaccessible.

Attaque par interception : une attaque sur la confidentialité

- **Objectif :** Permettre à un utilisateur non autorisé de lire des données sensibles sans les modifier.
- **Impact :**
  - Même sans modification des données, ces attaques peuvent causer des dommages considérables.
  - Exemple : vol de numéros de carte de crédit, numéros de sécurité sociale ou autres informations personnelles sensibles.
- **Types d'attaques par interception :**
  - **Keylogging :** Programmes ou dispositifs matériels enregistrant les frappes au clavier.
  - **Packet sniffing :** Analyseurs de paquets inspectant le trafic réseau pour trouver des mots de passe ou des données sensibles.

Attaques sur l'intégrité : modification et fabrication

- **Types d'attaques sur l'intégrité :**
  - **Attaque par modification :** Modifie des données existantes. *Exemple :* Changer le solde actuel d'une carte cadeau sur un site e-commerce.
  - **Attaque par fabrication :** Ajoute des données fictives au système. *Exemple :* Créer un utilisateur fictif dans un compte.
- **Vulnérabilité :** Si un système permet l'une de ces attaques, il présente une vulnérabilité en matière d'intégrité.

Exploitation d'une vulnérabilité : **Exploits**

- **Vulnérabilité :** Un système peut avoir des paramètres par défaut non sécurisés, par exemple :
  - Utilisateur par défaut : `DEFAULT`.
  - Mot de passe par défaut : `123456`.

  Tant que cette vulnérabilité n'est pas découverte, aucun dommage réel n'est causé au système.
- **Exploitation :**
  - Lorsqu'un utilisateur découvre et utilise cette vulnérabilité, elle devient un **exploit**.
  - Un exploit est une technique ou un mécanisme utilisé pour compromettre un élément de la triade de la sécurité de l'information (confidentialité, intégrité, disponibilité).
- **Exemples d'exploits :**
  - Utiliser un mot de passe par défaut.
  - Logiciels complexes interagissant avec le système pour provoquer un comportement indésirable.

Exploits : Malware -- Logiciels malveillants et leurs effets

- **Définition :** Le malware désigne tout logiciel conçu intentionnellement pour produire un effet indésirable sur un système informatique, souvent à l'insu de l'utilisateur autorisé.
- **Types de malware :**
  - **Bactérie :** Un programme qui consomme excessivement les ressources du système, par exemple en saturant les descripteurs de fichiers ou l'espace disque.
  - **Bombe fork :** Une forme particulière de bactérie qui se duplique en boucle (fork), saturant toutes les ressources CPU en créant constamment de nouvelles copies d'elle-même.

Malware (suite)

- **Bombe logique :** Code intégré dans un programme qui exécute une fonction non autorisée à un moment précis. *Exemple :* Supprimer toutes les données le premier jour du mois.
- **Porte dérobée (Trapdoor) :** Une partie de programme qui offre un accès secret à un système ou une application.
- **Cheval de Troie :** Logiciel se faisant passer pour un autre pour tromper les utilisateurs. *Exemple :* Un programme prétend offrir des curseurs amusants pour la souris, mais une fois installé, il supprime toutes les données du disque dur.
- **Virus :** Programme informatique souvent petit, qui se réplique avec l'intervention humaine. *Exemple :* Cliquer sur un lien ou exécuter un programme reçu en pièce jointe.

Malware (suite)

- **Ver (Worm) :** Programme informatique, souvent petit, qui se réplique sans intervention humaine. *Exemple :* Une fois installé, il peut utiliser une machine infectée pour tenter de pénétrer d'autres systèmes et copier son code.
- **Zombie :** Un ordinateur avec un logiciel installé permettant à des utilisateurs non autorisés d'exécuter des fonctions malveillantes. *Exemple :* Utiliser un programme de messagerie sur votre machine pour envoyer du spam, rendant l'expéditeur difficile à tracer.
- **Botnet :** Une collection de zombies contrôlés par un maître. *Utilisation :* Lancer des attaques massives comme des attaques par déni de service (DDoS) ou envoyer du spam à grande échelle.

Malware (suite)

- **Spyware :** Logiciel qui surveille secrètement les actions de l'utilisateur. *Exemple :* Un logiciel peut enregistrer toutes les frappes au clavier et les transmettre quotidiennement à un tiers.
- **Outils DoS :** Outils permettant de réaliser des attaques par déni de service (DoS), visant à rendre un système ou un service indisponible.
- **Rançongiciel (Ransomware) :** Logiciel exécutant une action indésirable, comme chiffrer votre disque dur, et exigeant une rançon pour la restaurer. *Remarque :* La rançon est généralement payée aux créateurs ou utilisateurs du logiciel.

Risques de sécurité pour un téléphone Android

- **Risque 1 :** Exploitation de bugs ou vulnérabilités pour obtenir des permissions non accordées.
- **Risque 2 :** Affichage de messages non sollicités, en particulier des publicités.
- **Risque 3 :** Résistance ou tentative de résistance à la désinstallation par l'utilisateur.
- **Risque 4 :** Propagation automatique vers d'autres appareils.
- **Risque 5 :** Masquage des fichiers et/ou des processus pour éviter la détection.
- **Risque 6 :** Divulgation d'informations privées à un tiers sans consentement.
- **Risque 7 :** Destruction des données de l'utilisateur ou du périphérique sans consentement.
- **Risque 8 :** Usurpation de l'identité de l'utilisateur, par exemple en envoyant des e-mails ou en effectuant des achats.
- **Risque 9 :** Épuisement de la batterie, des données mobiles, des minutes d'appel ou des SMS/MMS restants.
- **Risque 10 :** Dégradation de l'expérience utilisateur avec le téléphone.

Gestion des permissions pour les applications Android

- **Déclaration des permissions nécessaires :** Une application doit déclarer les permissions qu'elle requiert, par exemple :
  - Accéder à Internet.
  - Écrire dans des fichiers locaux.
  - Consulter les contacts.
  - Utiliser le Bluetooth.
  - Accéder à la position GPS.
  - Envoyer des SMS.
- **Autorisation de l'utilisateur :** Les utilisateurs doivent manuellement accorder les permissions requises par l'application.
- **Contrôle d'accès détaillé :** Les permissions sont définies de manière fine dans le fichier `Manifest.xml` de l'application :
  - Permissions spécifiques aux fichiers/URL.

Exemple dans `Manifest.xml`

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
    <uses-permission
        android:name="android.permission.RECEIVE_SMS"
    />
    ...
</manifest>
```

Applications signées : Sécurité et confiance

- **Applications signées :**
  - Codées avec une clé privée de développeur.
  - Sur Android et iPhone, les applications doivent être signées avant d'être mises sur le marché.
  - L'approbation manuelle réduit les risques d'applications malveillantes.
- **Confiance dans les applications officielles :**
  - Les applications disponibles sur l'App Store ou le Google Play Store sont généralement auditées.
  - Les utilisateurs peuvent évaluer les applications et laisser des commentaires.
- **Facteurs influençant la perception de la sécurité :**
  - Une application est-elle plus sûre si :
    - Elle provient d'un éditeur ou d'une entreprise connue ?
    - Un ami l'a déjà installée ?
    - Elle est payante ?

Problèmes liés aux permissions et aux applications spammeuses

- **Problèmes avec les permissions :**
  - Les applications peuvent demander trop de permissions.
  - Les utilisateurs ne comprennent pas toujours les implications des permissions.
  - Face à une surcharge d'informations, les utilisateurs cliquent souvent sur « Oui » sans réfléchir.
  - Une fois accepté, l'application peut avoir accès à presque tout.
- **Modifications des permissions via des mises à jour :**
  - Les mises à jour peuvent changer les permissions d'une application.
  - Exemple : Une mise à jour récente de l'application Facebook.
  - Les utilisateurs cliquent souvent sur « Oui » par confiance dans l'application. *(Élévation des privilèges)*.
- **Caractéristiques des applications spammeuses :**
  - Résistent à la désinstallation.
  - Affichent des publicités imitant l'interface système ou OS.
  - Divulguent ou endommagent les données personnelles de l'utilisateur.
  - Usurpent l'identité de l'utilisateur (envoi de messages, achats frauduleux, etc.).

Sécurité par l'obscurité : une pratique risquée

- **Définition :** La sécurité par l'obscurité repose sur le fait que les attaquants ne connaissent pas une information nécessaire pour nuire au système.
- **Exemples :**
  - **Fichier non sécurisé :** « Si un attaquant accédait à `http://foo.com/passwordscachette.txt`, il aurait nos mots de passe. Mais personne ne sait que ce fichier existe, donc nous sommes en sécurité. »
  - **Failles dans la base d'authentification :** « Notre base d'authentification est indisponible pendant 2 minutes chaque matin (nuit) à 4h. Durant ce laps de temps, tout utilisateur peut se connecter sans restrictions. Mais personne ne sait cela, et la probabilité de connexion à ce moment est minime. »
- **Problème :** La sécurité par l'obscurité est une stratégie fragile :
  - Si l'information est découverte, la sécurité du système est immédiatement compromise.
  - Cette approche ne remplace pas des mesures de sécurité robustes.

Bonnes pratiques pour la sécurité des utilisateurs

- **Authentification :** Forcer les utilisateurs à se connecter avant d'effectuer des opérations sensibles. *Raison :* Réduire le risque d'accès non autorisé.
- **Protocoles sécurisés :** Utiliser des protocoles sécurisés comme HTTPS pour empêcher l'interception des données.
  - Certaines plateformes utilisent HTTPS uniquement pour la page de connexion, puis repassent en HTTP pour les pages suivantes. **Est-ce une bonne pratique ? Non.** *Raison :* Les données peuvent être interceptées lors des futures interactions.
- **Mots de passe forts :** Imposer des mots de passe robustes pour les utilisateurs.
  - Éviter des mots de passe évidents comme `"password"`, `"abc"` ou identiques au nom d'utilisateur.
  - Inclure des règles pour renforcer les mots de passe (longueur minimale, caractères spéciaux, etc.).

Principe du moindre privilège : Sécurité proactive

- **Définition :** Accorder uniquement les privilèges nécessaires pour accomplir une tâche (et rien de plus).
- **Exemples :**
  - **Code :** Ne pas exécuter de code avec les privilèges `root` ou un utilisateur hautement privilégié, sauf si absolument nécessaire.
  - **Serveur web :** Limiter l'accès du serveur web uniquement aux fichiers HTML qu'il doit servir, rien d'autre.
- **Désactivation des services inutiles :**
  - Désactiver SSH, VNC, sendmail, etc., si ces services ne sont pas requis.
  - Fermer tous les ports, sauf :
    - Port `80` pour le trafic HTTP.
    - Tout autre port strictement nécessaire pour les opérations du serveur.

Validation et filtrage des données des utilisateurs

- **Principe général :** Encoder et filtrer les données des utilisateurs non fiables avant de les accepter dans un système de confiance.
- **Bonnes pratiques :**
  - Vérifier que les données acceptées respectent :
    - Le bon **type**.
    - Le bon **format**.
    - La bonne **longueur**.
  - Refuser les données incorrectes ou malveillantes dans des formulaires graphiques.
  - **Prévention des attaques SQL :** Supprimer tout code SQL des noms d'utilisateur ou autres champs soumis.
  - **Affichage sécurisé :** Encoder et/ou désinfecter tout texte d'entrée affiché en retour à l'utilisateur pour éviter des attaques comme XSS (Cross-Site Scripting).

Sécurité à chaque étape du développement logiciel

- **Avant l'écriture du code :**
  - Intégrer la sécurité dès la phase de conception.
- **Pendant l'écriture du code :**
  - Réaliser des **revues de code**.
  - Effectuer des **audits de sécurité du code**.
  - Utiliser la **programmation en binôme** pour réduire les erreurs.
- **Après l'écriture du code :**
  - Organiser des **revues de conception** (walkthroughs).
  - Procéder à des **audits de sécurité du système**.
  - Réaliser des tests de sécurité fonctionnels et système.
  - Effectuer des tests d'intrusion (*penetration tests*) pour identifier les failles potentielles.

Outils et techniques pour évaluer la sécurité

- **Audit de sécurité :**
  - Série de vérifications et de questions pour évaluer la sécurité d'un système.
  - Peut être réalisé par un auditeur interne ou externe.
  - *Meilleure pratique :* Considérer l'audit comme un processus continu et non un événement ponctuel.
- **Test d'intrusion (Penetration Test) :**
  - Tentative ciblée et éthique (white-hat) de compromettre la sécurité d'un système.
- **Analyse des risques :**
  - Évaluation des risques relatifs liés à ce qui peut mal tourner en cas de compromis de la sécurité.
- **Outil leader :** Mentionner un outil ou une suite logicielle de référence pour chacune de ces pratiques. *Exemple : Nessus, Burp Suite, OWASP ZAP.*

Liste de contrôle pour la sécurité des systèmes

- **Authentification sécurisée :**
  - Votre système nécessite-t-il une authentification sécurisée avec des mots de passe ?
  - Les mots de passe sont-ils difficiles à craquer ?
- **Contrôle d'accès et journalisation :**
  - Des listes de contrôle d'accès (ACL) sont-elles en place sur les appareils réseau ?
  - Existe-t-il des journaux d'audit pour enregistrer qui accède aux données ?
  - Ces journaux d'audit sont-ils régulièrement examinés ?
- **Système d'exploitation et applications :**
  - Les paramètres de sécurité du système d'exploitation respectent-ils les normes de l'industrie ?
  - Toutes les applications et services inutiles ont-ils été éliminés ?
  - Les systèmes d'exploitation et applications sont-ils à jour avec les correctifs ?
- **Sauvegarde et récupération :**
  - Comment les supports de sauvegarde sont-ils stockés ? Qui y a accès ? Sont-ils à jour ?
  - Existe-t-il un plan de reprise après sinistre ? A-t-il été testé ?

Liste de contrôle pour la sécurité des systèmes (2)

- **Chiffrement et applications personnalisées :**
  - Disposez-vous d'outils cryptographiques fiables pour gérer le chiffrement des données ?
  - Les applications personnalisées ont-elles été développées en tenant compte de la sécurité ?
  - Ces applications ont-elles été testées pour détecter des failles de sécurité ?
- **Documentation et gestion des modifications :**
  - Comment les modifications de configuration et de code sont-elles documentées ?
  - Ces enregistrements sont-ils revus régulièrement, et par qui ?

Table de classification des données

- Pour chaque type de données que votre application utilise ou enregistre, posez-vous les questions suivantes :

Type de données	**pers/sensible**	**Que fait l'app avec ?**	**Lieu/Format de stockage**	**Est-ce envoyé ?**
Exemple : Adresse e-mail	Oui	Notifications utilisateur	Crypté dans une base de données locale	Oui, via HTTPS
Exemple : Coordonnées GPS	Oui	Fournit des recommandations localisées	Enregistré temporairement en mémoire	Oui, en clair
Exemple : Historique d'achat	Oui	Historique des commandes	Crypté sur le serveur	Non

- **Questions supplémentaires pour chaque ligne :**
  - Ces données sont-elles réellement nécessaires ?
  - Peut-on limiter leur utilisation ou supprimer leur collecte ?
  - Le stockage et le transfert sont-ils sécurisés ?

Méthodes de stockage et confidentialité des données

- Où votre application stocke-t-elle ses données et pourquoi ? Est-ce sécurisé ?

Méthode de stockage	Description	Confidentialité des données
Préférences partagées (Shared Preferences)	Permet de stocker des types de données primitifs (par exemple, `int`, `Boolean`, `float`, `long` et `String`) qui persistent au-delà de la session de l'appareil. Même si votre application n'est pas en cours d'exécution, vos données persisteront jusqu'au redémarrage de l'appareil.	Vous pouvez définir quatre modes de confidentialité : `MODE_PRIVATE`, `MODE_WORLD_READABLE`, `MODE_WORLD_WRITABLE`, et `MODE_MULTI_PROCESS`. Le mode par défaut est `MODE_PRIVATE`.
Stockage interne (Internal Storage)	Permet de stocker vos données dans la mémoire interne de l'appareil. Généralement, ces données ne sont pas accessibles par d'autres applications ou par l'utilisateur final. C'est une zone de stockage privée. Les données stockées ici persistent même après un redémarrage de l'appareil. Lorsque l'utilisateur final supprime votre application, Android supprime également ces données.	Vous pouvez définir trois modes de confidentialité : `MODE_PRIVATE`, `MODE_WORLD_READABLE`, et `MODE_WORLD_WRITABLE`. Le mode par défaut est `MODE_PRIVATE`.

LISV Laboratoire d'Ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Méthodes de stockage et confidentialité des données

Méthode de stockage	Description	Confidentialité des données
Stockage externe (External Storage)	Les données stockées ici sont accessibles en lecture à tous. Les utilisateurs de l'appareil et d'autres applications peuvent lire, modifier et supprimer ces données. Le stockage externe est associé aux cartes SD ou à la mémoire interne de l'appareil (non amovible).	Les données sont accessibles en lecture par défaut.
Bases de données SQLite (SQLite Databases)	Si vous devez créer une base de données pour votre application afin de tirer parti des capacités de recherche et de gestion des données de SQLite, utilisez ce mécanisme de stockage.	Les bases de données que vous créez sont accessibles par toutes les classes de votre application. Les applications externes n'ont pas accès à cette base de données.
Connexion réseau (Network Connection)	Vous pouvez stocker et récupérer des données à distance via des services web. Vous pouvez lire davantage selon vos paramètres de service web.	Dépend des paramètres de votre service web.

Top 10 des problèmes OWASP [1] pour les app mobiles communiquant avec des app web

1. Identifier et protéger les données sensibles sur l'appareil mobile.
2. Gérer de manière sécurisée les identifiants de mot de passe sur l'appareil.
3. Assurer la protection des données sensibles en transit.
4. Implémenter correctement l'authentification des utilisateurs et la gestion des sessions.
5. Sécuriser les API (services) du back-end et la plateforme (serveur).

1. Open Web Application Security Project

Top 10 des problèmes OWASP [2] pour les app mobiles communiquant avec des app web

7. Effectuer l'intégration des données avec des services/applications tiers de manière sécurisée.
8. Porter une attention particulière à la collecte et au stockage du consentement pour l'utilisation des données de l'utilisateur.
9. Mettre en place des contrôles pour empêcher l'accès non autorisé aux ressources payantes (par exemple, portefeuille, SMS et appels téléphoniques).
10. Garantir une distribution/provisionnement sécurisé des applications mobiles.
11. Vérifier soigneusement les erreurs dans toute interprétation du code en temps d'exécution.

2. Open Web Application Security Project

Attaque de l'homme du milieu (Man-in-the-Middle)

- **Définition :** Une attaque où un tiers non autorisé intercepte et écoute le trafic web pendant son transit entre le client et le serveur.
- **Impact :**
  - Compromet la confidentialité des données.
  - Peut permettre des modifications ou des vols de données sensibles.
- **Bonne pratique :** Pour sécuriser votre application :
  - Utilisez le protocole HTTPS pour tout le trafic web.
  - HTTPS repose sur le protocole **SSL (Secure Socket Layer)** pour garantir une communication cryptée et sécurisée.

Denial-of-Service (DoS)

- **Définition :** Une attaque où l'attaquant rend un serveur web indisponible.
- **Comment l'attaque est menée :**
  - L'attaquant envoie fréquemment un grand nombre de requêtes à votre site web.
  - **DDoS (Déni de service distribué) :** Utilisation de nombreux ordinateurs pour amplifier l'attaque.
  - Résultat : Votre serveur ne peut pas gérer autant de requêtes et devient très lent ou totalement inutilisable.
- **Problèmes causés par cette attaque :**
  - Les utilisateurs ne peuvent pas accéder à votre site.
  - Si le serveur d'une boutique en ligne plante, cela entraîne une perte de revenus potentiels.
  - Le serveur peut planter et entraîner une perte ou une corruption des données importantes.
  - Toute la bande passante utilisée par les attaquants peut vous coûter très cher.

Sniffing de paquets : Risques et solutions

- **Définition :** Écoute du trafic envoyé sur un réseau. De nombreux protocoles Internet (HTTP, AIM, e-mail) ne sont pas sécurisés.
- **Risques :** Si un attaquant est sur le même réseau local (LAN) que vous, il peut :
  - Lire vos e-mails ou messages instantanés (IM) pendant que vous les envoyez.
  - Voir les sites web que vous consultez.
  - Intercepter votre mot de passe pendant qu'il est transmis au serveur.
- **Solutions :**
  - Utilisez des protocoles sécurisés comme **SSH** et **HTTPS**.
  - Chiffrez vos communications pour protéger les données sensibles.
  - Protégez votre réseau local (LAN) et Wi-Fi en empêchant les intrus d'y accéder.

Password cracking

- **Définition :** Tentative de deviner les mots de passe des utilisateurs privilégiés de votre système.
- **Méthodes d'attaque :**
  - **Attaque par force brute :** L'attaquant utilise un logiciel qui essaie séquentiellement toutes les combinaisons possibles de mots de passe.
  - **Attaque par dictionnaire :** L'attaquant utilise un logiciel qui essaie des mots de passe basés sur :
    - Tous les mots dans un dictionnaire.
    - Des combinaisons de mots, de nombres, etc.
- **Solutions :**
  - Forcer les utilisateurs à choisir des mots de passe sécurisés :
    - Longueur minimale.
    - Inclusion de lettres, chiffres et caractères spéciaux.
  - Bloquer une adresse IP après un certain nombre d'échecs de connexion ($\mathbb{N}$ tentatives).

Phishing : Manipulation et risques

- **Définition :**
  - Le phishing consiste à utiliser des e-mails ou des sites web masqués pour tromper les utilisateurs.
  - Technique de **social engineering (ingénierie sociale)** : Manipuler les utilisateurs pour obtenir frauduleusement des informations sensibles telles que des mots de passe ou des numéros de carte bancaire.
- **Problèmes :**
  - Si les utilisateurs de confiance de votre système sont trompés et donnent leurs informations personnelles :
    - Les attaquants peuvent utiliser ces informations pour se connecter en tant qu'eux.
    - Cela peut entraîner une compromission totale de votre système.

Élévation des privilèges (Privilege escalation)

- **Définition :** Une Élévation de privilèges se produit lorsqu'un attaquant parvient à exécuter du code sur votre serveur avec des droits d'utilisateur privilégié.
- **Exemple :**
  - Les utilisateurs normaux ne peuvent pas directement interroger votre base de données.
  - Mais un attaquant découvre une faille dans votre sécurité, lui permettant de se faire passer pour un administrateur et d'effectuer des requêtes directement.
- **Conséquences :**
  - Une fois qu'un attaquant fonctionne en tant que `root` (administrateur) :
    - Il contrôle totalement le serveur.
    - Il peut faire absolument tout ce qu'il veut : accéder aux données sensibles, modifier la configuration, supprimer des fichiers, etc.

Cross-site scripting (XSS)

- **Définition :** Une attaque XSS se produit lorsqu'un script malveillant d'un site est exécuté par un utilisateur naviguant sur un autre site.
  - **Exemple :** Vous visitez `google.com`, mais le JavaScript de `evil.com` s'exécute.
- **Comment l'attaque est réalisée :**
  - L'attaquant identifie un code non sécurisé sur le site cible.
  - Il utilise cette faille pour injecter un script JavaScript dans la page web.
  - Lorsque l'utilisateur visite cette page, il voit le code malveillant s'exécuter.
- **Conséquences :**
  - Vol d'informations sensibles (cookies, données de session, etc.).
  - Modification malveillante de l'interface utilisateur.
  - Redirections vers des sites dangereux.

Injection SQL

- **Définition :** Une injection SQL se produit lorsqu'un attaquant provoque l'exécution de requêtes SQL non désirées sur votre base de données.
- **Cause fréquente :** Des entrées non fiables sont directement insérées dans une requête SQL.
  - **Exemple en PHP :**

```
"SELECT * FROM Users WHERE name='$name';"
```

- **Exemple d'attaque :**
  - Un attaquant spécifie un nom d'utilisateur comme suit :

```
"a'; DROP TABLE Users;"
```

  - Résultat : La table `Users` est supprimée.
- **Conséquences :**
  - Perte ou corruption des données.
  - Compromission de la sécurité du système.
  - Vol de données sensibles.

Ingénierie sociale : La « vulnérabilité impossible à corriger »

- **Définition :** Manipuler des personnes (souvent des utilisateurs autorisés) pour les inciter, de manière sournoise, à effectuer des actions mettant en danger la sécurité d'un système.
- **Pourquoi est-ce si dangereux ?**
  - Cette attaque exploite le facteur humain, une faiblesse que les correctifs ou les outils techniques ne peuvent résoudre.
- **Méthode courante : Phishing**
  - Obtenir des informations personnelles ou sensibles via des e-mails ou autres communications frauduleuses.
  - Ces messages sont généralement envoyés à un grand nombre d'adresses e-mail en espérant que quelqu'un tombe dans le piège.
- **Conséquences :**
  - Accès non autorisé à des systèmes critiques.
  - Vol d'informations sensibles (mots de passe, numéros de carte bancaire, etc.).
  - Compromission totale de la sécurité du système.

Tester la résistance à l'ingénierie sociale : un défi unique

- **Pourquoi est-ce difficile ?**
  - Il n'y a pas d'aspect technique à vérifier :
    - Le système fonctionne correctement pour les utilisateurs autorisés.
    - Mais ces utilisateurs autorisés peuvent involontairement exécuter les ordres d'un utilisateur non autorisé.
  - Les personnes sont le facteur ultime d'état externe imprévisible :
    - Quelqu'un qui ignore habituellement les e-mails de phishing peut être trompé par un appel téléphonique.
    - Une personne vigilante peut, après une mauvaise nuit, manquer de discernement.
    - Une autre, submergée par le travail, peut cliquer sur un lien malveillant.
- **En quoi cela diffère des tests techniques ?**
  - Contrairement aux fonctions "impures" (dépendant d'états globaux ou externes mutables), les humains introduisent une variabilité bien plus grande.
- **Conclusion :** Tester la résistance humaine à l'ingénierie sociale nécessite :
  - Des simulations réalistes (tests de phishing, scénarios d'appels frauduleux).
  - Une sensibilisation et une éducation continues pour réduire les risques.

Exemple : Buffer Overflow (débordement de tampon)

- Le langage C est un langage de programmation de haut niveau, mais il suppose que le programmeur est responsable de l'intégrité des données.

Exemple Buffer Overflow

```c
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
  int value = 5; char buf1[8], buf2[8];
  strcpy(buf1, "one"); /* Put "one" into buf1 */
  strcpy(buf2, "two"); /* Put "two" into buf2 */
  printf("[AVANT] buf2 @ %p contains \'%s\'\n", buf2, buf2);
  printf("[AVANT] buf1 @ %p contains \'%s\'\n", buf1, buf1);
  printf("[AVANT] value @ %p is %d (0x%08x)\n", &value, value, value);
  printf("\n[STRCPY] copying %d bytes into buf2\n\n", strlen(argv[1]));
  strcpy(buf2, argv[1]); /* Copy first argument into buf2. */
  printf("[APRES] buf2 @ %p contains \'%s\'\n", buf2, buf2);
  printf("[APRES] buf1 @ %p contains \'%s\'\n", buf1, buf1);
  printf("[APRES] value @ %p is %d (0x%08x)\n", &value, value, value);
}
```

Résultat 1

```
[AVANT] buf2 @ 0x7fff192c484c contains 'two'
[AVANT] buf1 @ 0x7fff192c4854 contains 'one'
[AVANT] value @ 0x7fff192c485c is 5 (0x00000005)

[STRCPY] copying 3 bytes into buf2

[APRES] buf2 @ 0x7fff192c484c contains 'abc'
[APRES] buf1 @ 0x7fff192c4854 contains 'one'
[APRES] value @ 0x7fff192c485c is 5 (0x00000005)
```

Résultat 2

```
$./a.out abcdefghijklmn
[AVANT] buf2 @ 0x7fff3932a37c contains 'two'
[AVANT] buf1 @ 0x7fff3932a384 contains 'one'
[AVANT] value @ 0x7fff3932a38c is 5 (0x00000005)

[STRCPY] copying 14 bytes into buf2

[APRES] buf2 @ 0x7fff3932a37c contains 'abcdefghijklmn'
[APRES] buf1 @ 0x7fff3932a384 contains 'ijklmn'
[APRES] value @ 0x7fff3932a38c is 5 (0x00000005)
```

Résultat 3

```
$ ./a.out  AAAAAAAAAAAAAAAAAAAAAAAAAAAA
[AVANT] buf2 @ 0x7ffdca89421c contains 'two'
[AVANT] buf1 @ 0x7ffdca894224 contains 'one'
[AVANT] value @ 0x7ffdca89422c is 5 (0x00000005)

[STRCPY] copying 29 bytes into buf2

[AFTER] buf2 @ 0x7ffdca89421c contains 'AAAAAAAAAAAAAAAAAAAAAAAAAAAA'
[AFTER] buf1 @ 0x7ffdca894224 contains 'AAAAAAAAAAAAAAAAAAAA'
[AFTER] value @ 0x7ffdca89422c is 1094795585 (0x41414141)
Erreur de segmentation
```