Sécurité des Systèmes et Applications Web

Cours : Introduction à la Sécurité Informatique

Mohamed Said M'Bareck

SUPNUM
ms.mbareck@supnum.mr

2025-2026

Plan du Cours

1. Sécurité des Systèmes : Architecture et Fondations
   - Principes du Système d'Exploitation
   - Modèles de Contrôle d'Accès Mandataire (MAC)
   - Architecture d'Hébergement et Isolation
2. Sécurité Applicative : Attaques et Défauts de Code
   - La Surface d'Attaque Web
   - Vulnérabilités Basées sur les Entrées (Injections)
   - Failles de Logique Applicative et XML
   - Attaques Côté Client (XSS) et Fuites d'Informations
3. Hardening et Sécurité Positive (Défense)
   - Configuration Sécurisée (Hardening)
   - Gestion et Validation des Données
4. Tests, Assurance et Outils
   - Méthodologie et Outils de Test
5. Conclusion
   - Points Clés à Retenir

- **État de Protection** (*Protection State*) : Défini par une matrice d'accès.
- **Matrice de Contrôle d'Accès de Lampson** : Représentation bidimensionnelle des sujets ($s \in S$) et des objets ($o \in O$). Les cellules stockent les opérations ($ops(s, o)$) autorisées.
- **Principe du Moindre Privilège (PLP)** : Un principe essentiel pour limiter l'impact des compromissions.

Le Moniteur de Référence (Reference Monitor)

Mécanisme d'application d'accès classique.

- Doit être **isolé/à l'abri de la falsification** (*tamperproof*).
- Composants : Interface, Module d'Autorisation, et Magasin de Politique (*Policy Store*).

- **MAC** : Systèmes où la politique de protection est fixée par des administrateurs de confiance et est immuable pour les processus non fiables.
- **Modèle Bell-LaPadula (BLP)** : Un modèle MLS (Multilevel Security) pour la **confidentialité**.
  - *Simple-Security Property* : Lecture autorisée si $SC(s) \geq SC(o)$.
  - *Star-Security Property (β-security)* : Écriture autorisée si $SC(s) \leq SC(o)$.
- **Modèle Biba** : Pour l'**intégrité**.
  - *Simple-Integrity Property* : Lecture autorisée si $SC(s) \leq SC(o)$ (plus haute intégrité).
  - *Star-Integrity Property* : Écriture autorisée si $SC(s) \geq SC(o)$ (plus basse intégrité).

- **Architecture en Couches** (*Tiered Architecture*) : Améliore la modularité et la sécurité par rapport aux conceptions monocouches.
- **DMZ (Zone Démilitarisée)** : Le système de serveur WWW (World Wide Web) doit être placé dans la DMZ pour isoler le réseau interne.
- **Confinement** (*Confinement Problem*) :
  - Nécessaire pour les processus potentiellement malveillants.
  - Le **Sandboxing** (bac à sable) et la **Virtualisation (VM)** sont des mécanismes de confinement.

Assurance et Cycle de Vie

La sécurité doit être intégrée dès les phases de **Spécification** et de **Conception**. L'utilisation d'une méthodologie réduit les vulnérabilités et améliore la qualité et la sécurité du code.

- Le problème fondamental : l'utilisateur peut soumettre des **entrées arbitraires** (*arbitrary input*).
- La majorité de la surface d'attaque provient des mécanismes suivants :
  - **Authentification**.
  - **Gestion de Session** (e.g., Session Tokens).
  - **Contrôle d'Accès** (*Access Control*) : Source de vulnérabilités critiques.

Analyse des Requêtes HTTP

Les paramètres sont transmis via la chaîne de requête (URL) ou le corps de la requête POST. Ces paramètres peuvent inclure des données cachées (`hidden parameter`) ou des cookies (`Cookie`) qui contrôlent la logique applicative.

- **Injection SQL (SQLi)** : Une vulnérabilité classique menant à la modification ou à l'extraction de données. L'attaquant peut obtenir l'ASCII des caractères un par un (*blind SQL injection*).
- **Injection de Commandes OS** (*Command Injection*) :
  - Se produit lorsque l'entrée utilisateur est passée à l'interpréteur de commandes du système hôte.
  - Utilisation de **métacaractères shell** (e.g., `&`, `;`, `|`) pour injecter une commande séparée.
  - L'attaquant peut tenter d'ouvrir un canal hors-bande (*out-of-band*) pour récupérer les résultats (e.g., via `mail` ou `netcat`).
- **Traversée de Chemin** (*Path Traversal*) :
  - Exploite les fonctions qui accèdent aux fichiers sur le système hôte.
  - Utilise la séquence `../` pour remonter dans la structure des répertoires et accéder à des fichiers arbitraires (`/etc/passwd`).
  - Les URL-encodages (`%2e%2e%2f`) ou les **Unicode 16-bit encodings** (`%u002e`) sont souvent tentés pour contourner les filtres.

- **Défauts de Contrôle d'Accès** (*Broken Access Control*) :
  - L'application peut être attaquée en soumettant des requêtes **hors séquence** dans un processus multi-étape.
  - **Accès par Identifiant** : La fonction accédant à une ressource par un identifiant (e.g., `docid=1280149120`) est vulnérable si d'autres utilisateurs peuvent accéder à cet ID.
- **Injection d'Entité Externe XML (XXE)** :
  - Exploite les définitions d'**entités externes** qui utilisent le format URL (via le mot-clé `SYSTEM`) dans l'élément `DOCTYPE`.
  - Permet de forcer le parseur XML à **récupérer le contenu de fichiers locaux** (`<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>`) ou de se connecter à des systèmes *back-end* internes (ex : `http://192.168.1.1:25`).

- **Cross-Site Scripting (XSS)** : Permet d'injecter du code client (JavaScript) dans l'application, exécuté par le navigateur de l'utilisateur ciblé.
- **Vecteurs XSS** : Le code peut être inséré via des **protocoles pseudo-script** (e.g., `javascript:alert(1)`) ou des **gestionnaires d'événements HTML** (e.g., `onerror`).

Divulgation d'Informations (Information Disclosure)

- **Messages d'Erreur Verbeux** : Les erreurs non gérées (*unhandled error*) peuvent retourner des informations sensibles, notamment des **traces de pile** (*stack traces*) qui révèlent des chemins de fichiers ou l'état de l'exécution.
- **Contenu par Défaut** : Les fichiers de configuration, les fichiers sources ou les anciennes versions de fichiers peuvent être laissés sur le serveur, révélant des informations critiques.

Hardening et Sécurité Positive (Défense)

Configuration Sécurisée (Hardening)

- **Principe du Moindre Privilège (PLP)** : Les privilèges du compte OS doivent être au **niveau le moins élevé possible** pour le processus d'application.
- **Hardening du Serveur** :
  - **Supprimer le contenu par défaut** : Exemples d'applications, fonctionnalités de débogage (*debug functionality*) et manuels de serveur.
  - Les outils comme **Nikto** sont efficaces pour localiser ce contenu par défaut.
- **WebDAV** : Les méthodes HTTP WebDAV (`PUT`, `DELETE`, `MOVE`) doivent être restreintes, car elles permettent la manipulation de fichiers sur le serveur.

Hardening et Sécurité Positive (Défense)

Gestion et Validation des Données

- **Validation des Entrées** (*Validate Input*) :
  - Doit être effectuée **côté serveur** (*server-side*). Les contrôles côté client (*Client-Side Controls*) sont facilement contournés (ex : modifier des champs cachés ou désactivés).
  - Préférer l'approche **Accept Known Good** (whitelist) plutôt que Reject Known Bad (blacklist).
- **Traitement Sûr des Données** (Contre l'Injection) :
  - Utiliser des **requêtes paramétrées** (*parameterized queries*) pour les requêtes SQL. L'entrée utilisateur est traitée comme une valeur de donnée et ne peut pas modifier la structure de la requête.
- **Validation des Sorties** (*Validate Output*) :
  - Les données affichées doivent être **encodées en HTML** (*HTML-encoded*) pour neutraliser les caractères potentiellement malveillants. L'encodage garantit que les caractères spéciaux (comme < et >) sont traités comme du contenu et non comme de la structure HTML.

- **Méthodologie d'Attaque** : Comprend l'analyse des requêtes, le test des contrôles d'accès/sessions, et le **fuzzing** des paramètres.
- **Fuzzing** : Utilisation d'outils d'automatisation pour envoyer de grands nombres de chaînes d'attaque (*attack strings*) et identifier des **réponses anormales** (*anomalous responses*).
- **Test de Pénétration** : Le test est une étape critique.
- **Défense en Profondeur** (*Defense-in-Depth*) : Multiplier les couches de protection.

Outils pour les Laboratoires Pratiques

Burp Suite est la suite d'outils intégrée recommandée pour l'analyse des applications Web.

- **Proxy** : Pour intercepter, visualiser et modifier le trafic HTTP/HTTPS.
- **Intruder** : Pour automatiser les attaques personnalisées (énumération d'identifiants, fuzzing).
- **Sequencer** : Pour analyser les propriétés de **randomness** des tokens de session (statistiques).
- **Repeater** : Pour modifier manuellement et réémettre des requêtes pour des tests ad-hoc.

Points Clés à Retenir :

- La sécurité du système (MAC/MLS, PLP) et la sécurité applicative (Validation, Hardening) sont indissociables.
- Les vulnérabilités d'injection et de contrôle d'accès sont les cibles prioritaires.
- La validation doit toujours être effectuée côté serveur.
- Le module pratique approfondira l'exploitation de ces failles.