Sommaire Cours

I. Introduction – problématique de la sécurité – sécurité web
II. Présentation des menaces – vulnérabilités
III. Standards de sécurité utiles pour le web
IV. Architecture du web (infrastructure et applicatif)
V. Sécurisation des services
VI. Sécurisation des applications
VII. Sécurisation de l'infrastructure

Sécurisation des applications

1. Injection de commandes
- **a.Débordement de tampon (buffer overflow)**
- **b.Chaîne de format (format string)**
- **c.Injection LDAP**
- **d.Injection de commandes (OS Commanding)**
- **e.Injection SQL**
- **f.Injection SSI**
- **g.Injection XPath**

2: Authentification
- a.Force brute (brute force)
- b.Authentification insuffisante (insufficient authentication)
- c.Mauvais traitement des recouvrements de mot de passe (weak password recovery validation)

3: Révélation d'informations
- a.Listing de répertoires (directory indexing)
- b.Fuite d'informations (information leakage)
- c.Traversée de chemin (path traversal)
- d.Prédiction de localisation de ressources (predictable resource location)

4: Autorisation
- a.Prédiction de session (credential/session prediction)
- b.Autorisation insuffisante (insufficient authaurization)
- c.Expiration de session insuffisante (insufficient session expiration)
- d.Fixation d'identifiant de session (session fixation)

5.Attaques côté client
- **a.Usurpation de contenu (content spoofing)**
- **b.XSS (Cross Site Scripting)**

6.Logiques
- a.Abus de fonctionnalité (abuse of functionality)
- b.Déni de service (denial of service)
- c.Anti-automatisation insuffisante (insufficient anti-automation)
- d.Validation insuffisante du flux logique de l'application

7.Autres
- a.HTTP Response Splitting / CR LF Injection
- b.Prise d'empreinte (Web Server/Application Fingerprinting)

* http://projects.webappsec.org/f/WASC-TC-v2_0.pdf

Serveurs Web, d'Applications, de Données

- Serveur Web a comme rôle :
  - de récolter les requêtes au niveau du serveur http GET/HEAD/POST
  - De faire appel à des services (serveurs d'applications) pour générer des données dynamiquement
  - De renvoyer via le serveur http les pages au client
- Exemple de serveurs d'applications
  - CGI (Common Gateway Interface) qui exécute un programme extérieur dont la sortie standard sera intégrée au client via serveur http.
  - JSP (Java Server page) ou ASP (Active Server Pages) qui insère des blocs de script dans html . Ces pages sont transformées en servlet et sont exécutées par un serveur d'application
  - PHP (Hypertext Preprocessor) langage de script qui produit des données dynamiques en lien avec des serveurs de données .
- Serveurs de données
  - Stockent les informations sur les personnes, les ressources et d'authentification
  - Données générales gérées via langage SQL BD MySQL
  - Authentification LDAP(Lightweight Directory Access Protocol)
  - sont en interaction avec ces langages de scripts via des API
- Serveur web et Client web sont souvent séparés par un proxy qui
  - Intercepte les requêtes
  - Fait du filtrage/traitement (redirection)
  - Les retransmet au serveur

Serveurs Web, d'Applications, de Données

- Risques associés à cette architecture
  - L'injection (client-> Serveur)
    - consiste à envoyer une donnée non fiable à un interpréteur en tant qu'élément d'une commande
    - L'interpréteur est dupé et interprète la commande qui permet l'accès à des données non autorisées
  - Cross-Site Scripting (Serveur ->Client)
    - Données non fiables envoyées à un client web
    - Le navigateur de la victime exécute alors le code frauduleux généré dans la page de résultat. Le navigateur client exécute des scripts qui peuvent conduire à un détournement de session, redirection vers site malveillant défiguration de page …)
  - Défaillance dans la restriction des accès à une URL
    - qui permet un utilisateur d'accéder à des fonctionnalités de l'application, voire des fichiers et répertoires du serveur HTTP sans y être habilité
  - Attaque par traversée de répertoires
    - L'attaque par déduction consiste à deviner l'existence de fichiers http://www.site_vulnerable.fr/admin/admin.php ». Dans ce cas même si l'utilisateur mal intentionné n'y a pas accès au travers de l'application,
  - Faille de redirection
    - se produit quand une application Web réoriente les utilisateurs vers d'autres
    - les victimes sont victimes de phishing

Serveurs Web, d'Applications, de Données

- L'injection (client-> Serveur)
  - Attaque réside dans la détection des technologies utilisées pour formuler le code d'attaque adéquat
  - La plus utilisée des attaques selon l'OWASP
  - Objectifs ; de la simple récupération des données à la prise de contrôle totale des serveurs
  - Utilisation des zones de saisie proposée à l'utilisateur pour injecter des données malicieuses
  - L'interpréteur les interprètent en tant qu'élément d'une commande
  - L'interpréteur est dupé et interprète la commande qui permet l'accès à des données non autorisées
  - **Exemple** : Injection code PHP , script CGI, LDAP,
- Parade :
  - Paramétrage des serveurs pour masquer les informations de technologie utilisée
  - Vérification systématique et filtrage des données saisies avant interprétation par le serveur

Serveurs Web, d'Applications, de Données

Cross-Site Scripting basée sur le fait que les données reçues par une application web ne peuvent pas être considérées comme sûres.

- XSS stockée (Serveur ->Client)

:

  - le pirate va envoyer un **contenu malicieux dans un application web**, qui va le stocker (par exemple dans une base de données).
  - le contenu malicieux sera **retourné dans le navigateur des autres utilisateurs** lorsqu'ils iront sur le site.
  - **Exemple: L'a**ttaquant envoi un message ou un commentaire contenant le contenu malicieux. Lorsque les autres utilisateurs vont se rendre sur le forum ou le blog, ce contenu sera là, à chaque fois qu'ils afficheront la page.

Figure 13 - Principe d'une attaque XSS stockée

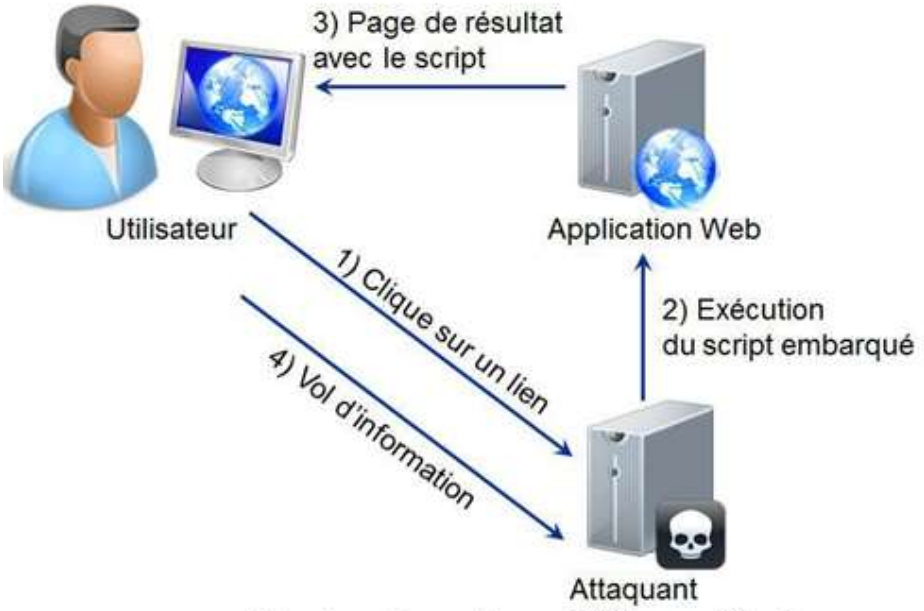
Serveurs Web, d'Applications, de Données

- XSS par réflexion
  - Le contenu malicieux est **livré** (envoyée par email ou par un autre moyen) **via une URL** qui le contient
  - **Exemple www.victim-website-example.com/previsionsmeteo?ville=Lyon**
  - Le pirate pourra utiliser cette URL pour fournir un contenu malicieux comme ceci:

:

```
http://www.forum-vulnérable.com/recherche.php?parametre=<script>alert('attaque XSS')</script>
```

Exemple: Pour consulter son courrier, l'utilisateur va préalablement s'authentifier et ses informations d'identification seront stockées dans des cookies. Un courrier malveillant peut intégrer du code JavaScript qui sera interprété par le navigateur. Ce code sera capable de récupérer les cookies et envoyer les informations à l'attaquant.



Principe d'une attaque XSS par réflexion

Serveurs Web, d'Applications, de Données

- DOM based XSS
  - Attaque qui n'utilise pas le serveur web
  - **les attaques DOM XSS se passent directement dans le navigateur de la victime**.
  - Attaques présentes dans les application « web 2.0", où une bonne quantité de code Javascript est exécutée dans le navigateur (vol de vos cookies, ou encore une redirection vers un autre site)

:

  - **Exemple :** www.victim-website-example.com/anagram_app/input/myword

Donne les anagrammes de myword

  - Un code xss malicieux envoyé par un pirate pourra vous envoyer l'URL suivante:

www.victim.website-example.com/anagram_app/input/myword[code malicieux]

- Parade :

Stockée, reflétée :tout le contenu doit être nettoyé et "validé" afin de pouvoir être utilisé de manière sûre sur la page.
DOM, le contenu doit également être encodé, avant d'être utilisé par l'application.

  - & par &amp ;
  - < et > par &lt ; et &gt ;
  - « par &quot ; 'par &#27 ; et / par &#x2F ;

Serveurs Web, d'Applications, de Données

- Défaillance dans la restriction des accès à une URL
  - Autre attaque consiste a deviner l'existence de fichiers ou répertoires

  Exemple : http://site.vulnerable.com/admin/admin.php
  - Ou a ne pas spécifier de nom de fichier

  Exemple : http://site.vulnerable.com/admin/
- Parade :
  - les répertoires doivent contenir un fichier index.html, ce qui évite de pouvoir accéder au répertoire lui-même
  - Vérification systématique et filtrage des données saisies avant interprétation par le serveur (interdire les caractères douteux comme '/ 'ou '\')

Serveurs Web, d'Applications, de Données

– Attaque par traversée de répertoires
  - permet d'accéder à des fichiers du serveur http notamment ceux contenant les clés privées de chiffrement.
  - Les applications vulnérables ouvrent des fichiers dont le nom est donné en paramètre de la requête http.

  Exemple : application qui inclut du texte en fonction de la langue du navigateur

```
<?php
$language="entete-en";
if (isset($_GET['lang'])) {
  $language=$_GET['lang'];
}
include ("/usr/local/webapp/template/" . $language . ".php")
?>
```

```
<?php
$languages=array("entete-en","entete-fr","entete-es");
$language=$languages[1];
if (isset($_GET['lang'])) {
  $tmp=$_GET['lang'];
  if (array_search($tmp, $languages)) {
    $language=$tmp;
  }
}
include ("/usr/local/webapp/template/" . $language . ".php")
?>
```

  - le langue est donnée en paramètre de la requête http,

    http://sitevulnerable.com?lang=../../../../etc/passwd%00

– Parade :
  - Vérification systématique et filtrage des données saisies avant interprétation par le serveur (interdire les caractères douteux comme '/ 'ou '\')
  - établir une liste de fichiers utilisables et refuser tout autre fichier.

Serveurs Web, d'Applications, de Données

- Faille de redirection
  - nombreux sites implémentent des mécanismes de redirections. :
  - redirection d'un utilisateur vers la page demandé une fois connecté
- **Exemple d'URL de redirection :**
  *http://www.sitepiraté.com/**login.php?goto=evil.com/login***
  - Le site piraté (site victime) redirige vers une page spécifiée via le paramètre "goto ».
  - Ce paramètre n'est pas vérifié par l'application web, qui redirige l'utilisateur victime.
- Parade :
  - supprimer les redirections ou désactiver les redirections hors des domaines de vos applications web
  - vérifier le "referrer" durant le process de redirection,