Intitulé du Master : Génie Informatique
Intitulé de la matière : Sécurité web

Fiche de TD N° 1 :

Répondez à chacune des questions suivantes :

1. Qu'est-ce que le Web 2.0 ?
2. Quels sont les objectifs de la sécurité des systèmes d'information ?
3. Quels sont les modèles architecturaux du Web ?
4. Quels sont les différents composants architecturaux du Web ?
5. Quels sont les différents composantes d'une URL ?

Fiche de TD N° 2 :

Répondez à chacune des questions suivantes :

1. Quelle est la différence entre Dynamic HTML et AJAX ?
2. Le navigateur traite le DOM HTML, est-il vrai pour le DOM CSS ?
3. Qu'offre un plugin et une extension comme Flash pour un navigateur ?
4. Quelle est la différence entre la méthode GET et POST de HTTP ?
5. Quelles sont les principales tâches des serveurs Web ?

Fiche de TD N° 3 :

Répondez à chacune des questions suivantes :

1. Donner quelques standards de la sécurité informatique ?
2. Quelle est la différence entre cryptographie symétrique et asymétrique ?
3. Quelle est la différence entre une fonction de hachage et une fonction cryptographique ?
4. Que signifie certificat électronique ?
5. Que signifie signature électronique ?

Fiche de TD N° 4 :

Exercice #1 :

Donner les différentes étapes afin d'installer SSL pour un site Web dans un serveur Apache ?

Exercice #2 :

1. Donner le schéma XML de
Soit le fichier etudiant.xml
<nom>Abdelkader </nom>
<age>82</age>
<naissance>1879-03-14</naissance>

2. Limiter le contenu d'un élément XML à un ensemble de valeurs acceptables (cas de type de voiture)

3. Limiter le contenu d'un élément XML afin de définir une série de nombres ou de lettres

Exercice #3 :

Que représente l'expression Xpath suivante:
- //message[dest="iut"][@date="2016-01-01"]
- */messages/message/contenu/text()*

Réponse Exercice #1:

Afin de sécuriser son site web en adoptant le protocole SSL, la démarche utilisée dans ce paragraphe s'adresse au serveur Web Apache sous environnement Linux Debian. Les étapes consistent en :

- ✓ Création du certificat serveur
  - ➢ Génération de la clé privée

  openssl genrsa 1024 > serverdah.key
- ✓ Demande de signature de certificat csr (Certificate Signing Request)
  openssl req -new -key serverdah.key > serverdah.csr

Remarque: La commande va vous demander de saisir des champs ; renseignez-les sauf le champ "Common Name" qui doit être identique au nom d'hôte de votre serveur virtuel (www.dah.dz)

- ✓ Envoyer le fichier "serverdah.csr" à un organsime de confiance (Autorité de Confiance) pour signature ou l'auto-signé
- ✓ Création du certificat de l'autorité de certification
  - ➢ Génération de la clé privée

  openssl genrsa -des3 1024 > cadah.key
- ✓ Création du certificat de l'autorité selon la norme X509
  openssl req -new -x509 -days 365 -key cadah.key > cadah.crt

- ✓ La signature du certificat serveur par l'autorité de certification (Certificate Autority)
  openssl x509 -req -in serverdah.csr -out serverdah.crt -CA cadah.crt -CAkey cadah.key -CAcreateserial -CAserial cadah.srl
- ✓ Installation du certificat d'autorité de certification

Installer le certificat de l'autorité de certification dans chaque navigateur client (cadah.crt). Il faut l'installer en tant qu'autorité de certification C'est ce dernier qui va valider le certificat reçu par le client lors de la requête https://

- ✓ Configuration d'Apache
  - ➢ Activer le module SSL (dans/etc/apache2/mods-enabled/, il y a ssl.conf et ssl.load)

  a2enmod ssl
  - ➢ Modifier le fichier configuration /etc/apache2/site-available/default-ssl

  …
  SSLCertificateFile /etc/apache2/ssl/serverdah.crt
  SSLCertificateKeyFile /etc/apache2/ssl/serverdah.key
  …
  </VirtualHost>
  - ➢ Activer le site SSL par défaut (/etc/apache2/site-available/default-ssl)

  a2ensite default-ssl
- ✓ Tester la configuration : A partir de votre navigateur taper
  https://localhost/index.html

Réponse Exercice #2:

1. Soit le schéma XML défini dans le fichier etuduiant.xsd

```
<xs:element name="nom" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="naissance" type="xs:date"/>
```

2. Pour limiter le contenu d'un élément XML à un ensemble de valeurs acceptables, on peut utiliser la contrainte d'énumération.

```
<xs:element name="car" type="carType"/>
<xs:simpleType name="carType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="BMW"/>
  </xs:restriction>
</xs:simpleType>
```

3. Pour limiter le contenu d'un élément XML afin de définir une série de nombres ou de lettres pouvant être utilisés, nous utiliserons la contrainte de modèle.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Réponse Exercice #3 :

Que représente l'expression Xpath suivante:

- //message[dest="iut"][@date="2016-01-01"] signifie usage de prédicat entre crochet
- */messages/message/contenu/text() signifie usage de sélecteur text()*

Fiche de TD N° 5 :

Répondez aux questions suivantes :

1. De quoi est constitué la méthode « roue de Deming » et donner son objectif ?
2. A qoui sert la cartograhie d'un site Web et donner les différents points à inspecter ?
3. De quoi est composé le modèle de menace STRIDE ?
4. De quoi est composé le modèle de risque DREAD ?
5. Quels sont les différents mécanismes d'authentification ?

Fiche de TD N° 6 :

Répondez aux questions suivantes :

1. C'est quoi un service d'annuaire et en quoi il est différent d'une base de données ?
2. C'est quoi le protocole LDAP ?
3. Que permet un serveur LDAP ?
4. Quel est le modèle de fonctionnement de LDAP ?
5. Quel est le modèle de sécurité de LDAP ?

Fiche de TD N° 7 :

Répondez aux questions suivantes :

1. En quoi consiste la sécurité des systèmes d'exploitations ?
2. Quels sont les droits sur les fichier Unix ?
3. Que signifie la commande chmod 777 inex.html ?
4. Que représente la base SAM (Security Account Manager) ?
5. Que représente la base de registre dans le système d'exploitation Windows ?

Fiche de TD N° 8 :

Répondez aux questions suivantes :

1. Quels sont les différents types de comptes dans le SE Windows ?
2. Que reprséente la fonctionnalité User Account Control dans Windows ?
3. Quels sont les différents mécanismes d'authentification dans Windows ?
4. Citer quelques exemples d'attaques réseau ?
5. Peut-on dire que le protocole DNS est sûr ?

Fiche de TD N° 9 :

Répondez aux questions suivantes :

1. Quels sont les objectifs des framework en environnement J2EE ?
2. Que permet l'API JAAS (Java Authentication and Authorisation Service) ?
3. Quels sont les aspects de l'exécution d'une application dans un environnement d'exécution dit « managé » ?
4. Que représente l'environnemt .NET ?
5. Que représente le composant CLR (Common Language Runtime) ?

Fiche de TD N° 10 :

Répondez aux questions suivantes :

1. Selon le top 10 des attaques Web les plus pertinentes du site OWASP pour l'année 2017, citer les et donner une explicaction pour chacune d'elle ?

2. Donner la classification des types de menaces définie par le WASC (Web Application Security Consortium) ?

Réponse aux questions suivantes :

1. Top 10 2017 des attaques selon OWASP (Open Web Application Security Project)

Attaque	Description
A1. Injection	Des failles d'injection, telles que l'injection SQL, OS, XXE et LDAP, se produisent lorsque des données non approuvées sont envoyées à un interpréteur dans le cadre d'une commande ou d'une requête. Les données hostiles de l'attaquant peuvent tromper l'interprète à exécuter des commandes involontaires ou à accéder aux données sans autorisation appropriée.
A2. Authentification et gestion de session interrompues	Les fonctions applicatives liées à l'authentification et à la gestion de session sont souvent implémentées incorrectement, permettant aux attaquants de compromettre les mots de passe, les clés ou les jetons de session, ou d'exploiter d'autres failles de mise en œuvre pour assumer l'identité d'autres utilisateurs (temporairement ou définitivement).
A3. Cross-Site Scripting (XSS)	Des failles XSS se produisent chaque fois qu'une application inclut des données non fiables dans une nouvelle page Web sans validation correcte ou échappement, ou met à jour une page Web existante avec les données fournies par l'utilisateur à l'aide d'une API de navigateur qui peut créer du

	JavaScript. XSS permet aux attaquants d’exécuter des scripts dans le navigateur qui peut détourner les sessions des utilisateurs, altérer les sites Web ou rediriger l'utilisateur vers des sites malveillants.
A4. Contrôle d'accès cassé	Les restrictions sur ce que les utilisateurs authentifiés sont autorisés à faire ne sont pas correctement appliquées. Les attaquants peuvent exploiter ces failles pour accéder à des fonctionnalités et / ou des données non autorisées, telles que l'accès à d'autres comptes des utilisateurs, afficher les fichiers sensibles, modifier les données d'autres utilisateurs, modifier les droits d'accès, etc.
A5. Mauvaise configuration de la sécurité	Une bonne sécurité nécessite d'avoir une configuration sécurisée définie et déployée pour l'application, les frameworks, le serveur d'applications, le serveur Web, le serveur de base de données, la plate-forme, etc. Les paramètres sécurisés doivent être définis, implémentés et maintenus, car les valeurs par défaut sont souvent non sécurisées. De plus, les logiciels doivent être tenus à jour.
A6. Exposition des données sensibles	De nombreuses applications Web et API ne protègent pas correctement les données sensibles, telles que les données financières, les soins de santé et les informations personnelles. Les attaquants peuvent voler ou modifier ces données faiblement protégées pour mener une fraude par carte de crédit, un vol d'identité ou d'autres crimes. Les données sensibles méritent une protection supplémentaire telle que le cryptage au repos ou en transit, ainsi que des précautions particulières lors de l'échange avec le navigateur.
A7. Protection contre les attaques insuffisante	La majorité des applications et des API n'ont pas la capacité de base de détecter, de prévenir et de répondre aux attaques manuelles et automatisées. La protection contre les attaques va bien au-delà de la validation d'entrée de base et implique automatiquement la détection, la journalisation, la réponse et même le blocage des tentatives d'exploitation. Les propriétaires d'applications doivent également pouvoir déployer rapidement des correctifs pour se protéger contre les attaques.
A8. Cross-Site Request Forgery (CSRF)	Une attaque CSRF force le navigateur d'une victime connectée à envoyer une requête HTTP falsifiée, y compris le cookie de session de la victime et toute autre information d'authentification automatiquement incluse, à une application Web vulnérable. Une telle attaque permet à l’attaquant de forcer le navigateur d’une victime à générer des requêtes que l’application vulnérable considère comme des requêtes légitimes de la victime.
A9. Utilisation de composants avec des vulnérabilités connues	Les composants, tels que les bibliothèques, les frameworks et d'autres modules logiciels, s'exécutent avec les mêmes privilèges que l'application. Si un composant vulnérable est exploité, une telle attaque peut faciliter de graves pertes de données ou une prise de contrôle du serveur. Les applications et les API utilisant des composants avec des vulnérabilités connues peuvent saper les défenses des applications et permettre diverses attaques et impacts.
A10. API sous-protégées	Les applications modernes impliquent souvent des applications clientes riches et des API, telles que JavaScript dans le navigateur et les applications mobiles, qui se connectent à une API quelconque (SOAP / XML, REST / JSON, RPC, GWT, etc.). Ces API sont souvent non protégées et contiennent de nombreuses vulnérabilités.

2. Types d’attaques (classification WASC)

Nous reprenons la classification des types de menaces définie par le WASC (Web Application Security Consortium) :

1. Authentification
a. Force brute (brute force)
b. Authentification insuffisante (insufficient authentication)
c. Mauvais traitement des recouvrements de mot de passe (weak password recovery validation)
2. 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)
3. Attaques côté client
a. Usurpation de contenu (content spoofing)
b. XSS (Cross Site Scripting)
4. Exécution 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
5. Révélation d’informations
a. Listing de repertoires (directory indexing)
b. Fuite d’informations (information leakage)
c. Traversée de chemin (path traversal)
d. Prédiction de localisation de ressources (predictable resource location)
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 / CRLF Injection
b. Prise d’empreinte (Web Server/Application Fingerprinting)

Fiche de TD N° 11 :

Exercice #1 :

Donner un exemple d'attaque par injection SQL et comment s'en prémunir ?

Exercice #2 :

Donner un exemple d'injection Xpath ?

Exercice #3 :

Donner un exemple d'injection de commande ?

Exercice #4 :

Donner un exemple d'attaque par traversée de répertoire et comment s'en prémunir ?

Exercice #5 :

Donner un exemple d'attque XXE (XML eXternal Entity) ?

Réponse Exercice #1 :

1. Exemple d'attaque par injection SQL :

a. Injection SQL
Dans cet exemple, nous avons utilisé des machines virtuelles. L'hyperviseur utilisé est Virtual Box d'Oracle, version 5.0.4. La machine hôte est un Linux Debian de nom de code Jessie.

Les commandes suivantes pour connaître la version du système d'exploitation utilisé.
cat /etc/os-release ou encore *lsb_release -a*

Le fichier utilisé est : virtualbox-5.0_5.0.4-102546~Debian~jessie_i386.deb
Avant d'installer Virtual Box, j'ai dû faire quelques changements.
le fichier */etc/apt/sources.list*, ajouter
deb http://http.debian.net/debian/ jessie main contrib
- apt-get update
- dpkg -i virtualbox-5.0_5.0.4-102546~Debian~jessie_i386.deb
- apt-get install dkms (construction de modules)

Pour sécuriser l'installation de Virtual Box
apt-key add oracle_vbox.asc
Le fichier *oracle_vbox.asc* est un fichier qui contient la clé publique téléchargé du site virtualbox.org.

Remarque :
Le fichier headers est également important, il suffit de l'installer par la commande
- apt-get install linux-headers-$(uname -r)

L'installation d'une machine virtuelle Debian est simple, il suffit de téléchargé « .iso » correspondant et de l'installer. J'ai également téléchargé la version Kali de linux pour effectuer quelques tests. Une fois les deux machines virtuelles installées, j'ai lancé sur ma machine hôte la commande suivante :

VBoxManage dhcpserver add --netname intnet --ip 192.168.1.1 --netmask 255.255.255.0 --lowerip 192.168.1.20 --upperip 192.168.1.250 --enable

Pour les machines virtuelles, au niveau *Settings-Network*, choisir *Attached to Internal Network* et comme nom de réseau *intnet*.

Sur la machine virtuelle Debian, j'ai installé Apache, MySql et Php
- *apt-get install apache2*
- *apt-get install mysql-server mysql-client*
- *apt-get install php5*

Dans le répertoire */var/www/html*, j'ai créé un répertoire nommé « exemple1 » dans lequel j'ai écrit les exemples suivants :

Le fichier exemple1.html

```
<html>
<body>
<form action='exemple1.php' method = GET'>
<p> Votre pseudo : <input type='text' name='spseudo'/></p>
<p> Votre mot de passe : <input type='text' name='spass'/></p>
<p>  <input type='Submit' value='OK'/></p>
</form>
</body>
</html>
```

Dans le fichier *exemple1.php*

```
<?php
$username = 'root' ;
$password = 'mysql' ;
$hostname='localhost' ;
$db_name='dahmani' ;
$table_name='users_dahmani' ;
$db_connect= mysql_connect($hostname,$username,$password) or die('Accès refusé sur MySQL') ;
$db_selected=mysql_select_db($db_name,$db_connect) or die('Accès refusé sur la base de données') ;


$qry= « Select * From $table_name Where username= ' ».$_GET['spseudo']. « ' and
                                password = password(' »._GET[spass]. »') » ;
$result=mysql_query($qry) ;
$num_rows=mysql_num_rows($result) ;
if($num_rows==0){echo 'Cet identifiant n'existe pas' ; mysql_error() ; exit}
$row = mysql_fetch_row( $result);
echo 'User identifier : '.$row[0].'<br>' ;
```

```
echo 'Username : '.$row[1].'<br>' ;
echo 'Password  : '.$row[2].'<br>' ;
mysql_close('$db_connect') ;
?>
```

Dans cet exemple, un développeur pourra entamer les instructions suivantes avant de lancer le programme.

Dans la machine virtuelle contenant Apache et mySQL et en ouvrant un terminal, on tape les commandes suivantes :

```
#mysql  --user=root  --password=mysql
```

mysql>create database dahmani ;
mysql>show databases ; (pour confirmer)
mysql>use dahmani ;

Un développeur pourra créer cette liste à l'aide de l'instruction SQL suivante :

```
create table users_dahmani (id INTEGER PRIMARY KEY,
                            username VARCHAR(32),
                            password VARCHAR(41)     );
```

mysql>show tables ; (pour visualiser les tables)
mysql>describe users_dahmani ; (visualiser la structure)
mysql>insert into users_dahmani values(9,'ydahmani9',password('ydahmani9')) ;
mysql>exit

Que se passe-t-il si on saisit :
Votre pseudo : *9' or '1'='1*
Votre mot de passe : Peu importe
Select * From users_dahmani Where id = '9' or '1'='1' and password = password('Peu importe') ; qui est toujours vraie peu importe le mot de passe saisi du moment que le pseudo « 9 » est déjà saisi dans la table users_dahmani (valeur de vérité est « Vrai or Vrai and Faux » qui est toujours Vrai) .

La chaine suivante peut également donner le 1er enregistrement peu importe le mot de passe:
Votre pseudo: dah' or '1'='1' or ' ' =' Votre mot de passe : Peu importe

b. Prévention contre l'injection SQL

Le principal problème demeure : les chaînes ne sont pas correctement échappées ou que les types de données ne sont pas contraints. Il faudrait utiliser certaines fonctions comme : `mysqli_real_escape_string; is_numeric() ; ctype_digit().`

La meilleure solution consiste à faire appel à des requêtes préparées. Initialement destinées à optimiser les connecteurs de base de données, elles séparent les instructions SQL des données issues des utilisateurs.

Exemple :

```
$stmt = $dbh->prepare("INSERT INTO Customers (CustomerName,Address,City)
VALUES (:nam, :add, :cit)");
$stmt->bindParam(':nam', $txtNam);
$stmt->bindParam(':add', $txtAdd);
```

```
$stmt->bindParam(':cit', $txtCit);
$stmt->execute();

$mysqli = new mysqli('hostname', 'db_username', 'db_password', 'db_name');
$query = sprintf("SELECT * FROM `Users` WHERE UserName='%s' AND Password='%s'", $
$mysqli->real_escape_string($username),
$$mysqli->real_escape_string($password));
$mysqli->query($query);
```

Réponse Exercice #2 :

Injection de XPath

Lorsque l'on choisit de stocker des données sensibles en XML plutôt que dans une base de données SQL, les attaquants peuvent s'appuyer sur une injection de XPath pour contourner une authentification. Xpath est un langage de requêtes spécialisé permettant de parcourir des documents XML complexes.

Soit le fichier « utilisateursxml » un document XML renfermant les données suivantes :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE utilisateurs SYSTEM "utilisateurs.dtd">
<?xml-stylesheet type="text/css" href="utilisateurs.css" ?>
<utilisateurs>
        <utilisateurr>
        <numero>  0 </numero>
        <pseudo> admin </pseudo>
        <mpass> admin </mpass>
        </utilisateur>
        <utilisateurr>
        <numero>  1 </numero>
        <pseudo> ydahmani1 </pseudo>
        <mpass> ydahmani1 </mpass>
        </utilisateur>
        <utilisateurr>
        <numero>  2 </numero>
        <pseudo> ydahmani2 </pseudo>
        <mpass> ydahmani2 </mpass>
        </utilisateur>
</utilisateurs>
```

Le fichier "utilisateurs.dtd" se compose de :

```
<!ELEMENT utilisateurs (utilisateur)>
<!ELEMENT utilisateur (numero, pseudo, mpass)+>
<!ELEMENT numero (#PCDATA)>
<!ELEMENT pseudo (#PCDATA)>
<!ELEMENT mpass (#PCDATA)>
```

Le fichier "utilisateurs.css" se compose de :

```
utilisateurs{display : block}
utilisateur{display : block}
numero{
display : block ;
font-family:arial ;
color:red ;
text-align:center
}
pseudo{
display : block ;
font-family:arial ;
color:yellow ;
text-align:center
}

numero{
display : block ;
font-family:arial ;
color : #800000 ;
text-align:center
}
```

A présent on code le fichier exemple2.html comme suit :

```
<html>
<body>
<form action='exemple2.php' method = GET'>
<p> Votre pseudo : <input type='text' name='spseudo'/></p>
<p> Votre mot de passe : <input type='text' name='spass'/></p>
<p>  <input type='Submit' value='OK'/></p>
</form>
</body>
</html>
```

Le fichier exemple2.php comprend :

```
<?php
$pseu = $_GET['spseudo'];
$mp   = $_GET['spass'];
$sxml = simplexml_lod_file('utilisateurs.xml');
$path = “/utilisateurs/utilisateur[pseudo = '$pseu' and mpass = '$mp']”;
$result =  $sxml->xpath($path);
foreach($result as $node){
echo “Numero Trouve : ”{$node->numero} <br />”;
echo “Pseudo Trouve : ”{$node->pseudo} <br />”;
echo “Mot de passe Trouve : ”{$node->mpasse} <br />”;
}
?>
```

Si on injecte dans la zone pseudo la chaine : *admin' or '1'='1* alors on peut accéder à cet enregistrement peu importe le mot de passe saisi.

Réponse Exercice #3 :

Injection de commande

Une injection de commande réussie donne à l'attaquant certains privilèges sur la machine distante.

Soit l'exemple suivant. On veut envoyer localement un email à un utilisateur. On crée le fichier à envoyer

Soit le fichier youcef.txt contenant:
Bonjour
Cette lettre vous ai adressée par soucis de test.

Le fichier exemple3.html composé du code suivant :

```
<html>
<body>
<form action='exemple3.php' method = GET'>
<p> Sujet de l'email : <input type='text' name='semail'/></p>
<p> Destination de l'email : <input type='text' name='demail'/></p>
<p> Fichier attaché à l'email : <input type='text' name='femail'/></p>
<p>  <input type='Submit' value='OK'/></p>
</form>
</body>
</html>
```

Le fichier exemple3.php comprenant le code suivant:

```
<?php
$s_email = $_GET['semail'];
$d_email = $_GET['demail'];
$f_email = $_GET['femail'];

$p_email   = " {$d_email} -s  \"{$s_email}\" < ./{$s_email} ";
$cmd = exec('mail $p_email ');
?>
```

Si on saisi les chaines suivantes, l'email parviendra à son destinataire malgré la chaine saisie.
Sujet de l'email : Sougueur
Destination de l'email : *-v ; mail youcef@localhost*
Fichier attaché à l'email : youcef.txt

En verifiant via la commande Linux
cat /var/mail/youcef

On va trouver que l'email est envoyé malgré la chaine forgée au niveau de la zone texte Destination de l'email

Réponse Exercice #4 :

Attaque par traversée de répertoires

Les attaquants utilisent les attaques par traversée de répertoires pour accéder à des fichiers critiques sur les serveurs web pour obtenir des données confidentielles telles les clés privées SSL, des mots de passe, etc.

Un exemple simple est celui de la lecture d'un fichier utilisateur. Soit alors les codes suivants:
On crée le fichier exemple4.html

```
<html>
<body>
<form action='exemple4.php' method = GET'>
<p> Fichier a visualiser : <input type='text' name='sfile'/></p>
<p>  <input type='Submit' value='OK'/></p>
</form>
</body>
</html>
```

Le fichier exemple4.php comprend :

```
<?php
echo “Attaque par traversée de répertoire <br />”
$s_fichier = $_GET['sfile'];
include ($s_fichier);
?>
```

Si on injecte dans la zone Fichier à ouvrir la chaine : *../../../../../etc/passwd* alors on peut ouvrir le fichier de tous les utilisateurs de notre serveur.

Prévention contre l’attaque par traversée de répertoire
PHP active par défaut le réglage *magic_quotes_gpc* , qui échappe les caractères douteux dans les requêtes GET , POST et les cookies en les préfixant de l'anti-slash.

Parfois c'est à la responsabilité du programmeur d'etablir des règles pour ce protéger contre ce genre d'attaque.

Réponse Exercice #5 :

Attaque XXE (XML eXternal Entity)

Les attaques par entités XML externes permettent à l’attaquant de visualiser des fichiers de son choix sur le serveur.

Dans cet exemple, on permet à chaque utilisateur d'avoir son propre site Web
On crée tout d'abord un prototype pour chaque utilisateur

mkdir /etc/skel/public_html
mkdir /etc/skel/logs
echo “<h1> Nouvel espace Web cree </h1> ” > /etc/skel/public_html/index.html
useradd -g www-data -m younes

Créer un fichier de configuration virtual host de l'utilisateur younes
contenu du fichier /etc/apache2/sites-available/younes.conf
<VirtualHost *:80>
ServerName localhost
DocumentRoot /home/younes/public_html
<Directory /home/younes/public_html/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
 </Directory>
ErrorLog /home/younes/logs/error.log
LogLevel warn
CustomLog /home/younes/logs/access.log combined
ServerSignature Off
</VirtualHost >

Désactiver dans /etc/apache2/mods-enabled/php5.conf la partie
#<IfModule mod_userdir.c>
jusqu'à
#<IfModule>
Cette désactivation va permettre de lancer les pages php plutôt de les ouvrir.

Ajouter dans le fichier /etc/apache2/apache2.conf
ServerName localhost

Activer ensuite le module userdir
a2enmod rewrite
a2enmod userdir

relancer apache
service apache2 restart

Vérifier la configuration
apache2ctl -t

Si le message est Syntax OK, on teste le site de l'utilisateur younes par
links http://localhost/~younes **Attaque XXE (XML eXternal Entity)**

Les attaques par entités XML externes permettent à l’attaquant de visualiser des fichiers de son choix sur le serveur.

Dans cet exemple, on permet à chaque utilisateur d'avoir son propre site Web
On crée tout d'abord un prototype pour chaque utilisateur

mkdir /etc/skel/public_html

mkdir /etc/skel/logs
echo “<h1> Nouvel espace Web cree </h1> ” > /etc/skel/public_html/index.html
useradd -g www-data -m younes

Créer un fichier de configuration virtual host de l'utilisateur younes
contenu du fichier /etc/apache2/sites-available/younes.conf
<VirtualHost *:80>
ServerName localhost
DocumentRoot /home/younes/public_html
<Directory /home/younes/public_html/>
        Options Indexes FollowSymLinks MultiViews
        AllowOverride All
 </Directory>
ErrorLog /home/younes/logs/error.log
LogLevel warn
CustomLog /home/younes/logs/access.log combined
ServerSignature Off
</VirtualHost >

Désactiver dans /etc/apache2/mods-enabled/php5.conf la partie
#<IfModule mod_userdir.c>
jusqu'à
#<IfModule>
Cette désactivation va permettre de lancer les pages php plutôt de les ouvrir.

Ajouter dans le fichier /etc/apache2/apache2.conf
ServerName localhost

Activer ensuite le module userdir
a2enmod rewrite
a2enmod userdir

relancer apache
service apache2 restart

Vérifier la configuration
apache2ctl -t

La page s'affichera.

Maintenant on passe à l'exemple 5 qui traite de l'attaque XXE.
Dans cet exemple on crée un fichier exemple5.html dans le répertoire /home/younes/public_html
<html>
<body>
<a link rel="relate" type="application/rss+xml" title="Flux Younes"
href="http://localhost/~younes/exmple5RSS.xml"
<img src=exemple5.png>

Flux RSS Younes

<form action='exemple5.php' method = GET'>
 Fichier a visualiser : <input type='text' name='sfile'/>
  <input type='Submit' value='OK'/>
</form>
</body>
</html>

Le fichier de flux exemple5RSS.xml contient

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE testRSS [
<!ENTITY passwd SYSTEM "/etc/passwd">
]>
<rss version="2.0">
<channel>
<title>Mon flux RSS d’attaque présentant /etc/passwd</title>
<description>ceci est file:/etc/passwd: &passwd;</description>
        <item>
        <title>/etc/passwd</title>
        <description>Nouvelle RSS Younes Dec. 2015 &passwd; </description>
        <link>http://localhost/~younes/exmple5RSS.php</link>
        </item>
</channel>
</rss>
```

Le fichier exemple5RSS.php contient

```
<?php
$buffer=<<<XML
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE commande [
<!ELEMENT commande ANY>
<!ENTITY test SYSTEM "/etc/passwd">
]>

<commande> &test; </commande>
XML ;

libxml_disable_entity_loader(false) ;// se protéger contre XXE
$ch=simplexml_load_string($buffer, 'SimpleXMLElement',LIBXML_NOENT) ;
$ch->asXML('php://output') ;
?>
```

Maintenant toute personne voulant s'abonner à notre site doit installer à titre d'exemple avec le navigateur FireFox le plugin SimpleRSS Reader. Une fois installé, toute modification dans le fichier exemple5RSS.xml sera notifiée dans le navigateur de l'abonné.

Fiche de TD N° 12 :

Exercice #1 :

Donner un exemple d'attaque par injection de données arbitraires dans un site web (XSS) ?

Exercice #2 :

Donner un exemple d'attaques inter-domaines (CSRF) ?

Exercice #3 :

Quelles sont les contre mesures pour se protéger des attques dans Apache-PHP ?

Réponse Exercice #1 :
Injection de données arbitraires dans un site web (XSS)

Cet exemple présente une injection de HTML reflété

```
<html>
<body>
<?php
if (isset($_GET['UserInput'])){
$out = 'vous avez saisi: "' . $_GET['UserInput'] . '".';
} else {
$out = '<form method="GET">saisissez quelque chose ici: ';
$out .= '<input name="UserInput" size="50">';
$out .= '<input type="submit">';
$out .= '</form>';
}
print $out;
?>
</body>
</html>
```

Si on saisit dans l'adresse URL
http://localhost/partie2/XSS/exemple6.php?UserInput=<script>alert(1)</script>

Cette commande permet d'injecter du code et d'afficher une boite de dialogue.

Si la méthode POST est utilisée, l'attaquant devra saisir le code suivant :

```
<html>
<body>
```

```
<form name="formulaireMalveillant"method="POST" action="http://localhost/partie2/XSS/exemple6.php">
<input type="hidden" name="UserInput" value="<script>alert(1)</script>">
</form>
<script>
document.formulaireMalveillant.submit()
</script>
</body>
</html>
```

Ce code produira le même effet que le précédent, une boite de dialogue s'affichera.

Réponse Exercice #2 :

Attaques inter-domaines (CSRF)

Ces exemples permettent de présenter quelques usages légitimes des interactions inter-domaines.
On crée un fichier exemple7a.html

```
<html>
<a href="http://localhost"> Ceci est un exemple </a>
</html>
```

Les images peuvent également servir de liens :

```
<a href="http://localhost">
<img src="/images/exemple7.png">
</a>
```

On peut ajouter à ce fichier le code javascript suivant

```
<html>
<a href="http://192.168.1.20"> Ceci est un exemple </a>
<script>
window.open('http://192.168.1.21/Partie2/XSS/exemple6s.php','exemple','width=400,height=300');
</script>
</html>
```

Avec cet exemple, le navigateur iceweasel vous averti d'une fenêtre pop-up. C'est l'utilisateur qui décide de l'accéptation de cette redirection.

Un autre exemple utilise les iFrame. On crée le fichier exemple7b.html

```
<html>
<a href="http://192.168.1.20"> Ceci est un exemple </a>
<iframe src ="http://192.168.1.21/Partie2/XSS/exemple6s.php" width="100%">
</iframe>
</html>
```

Dans cet exemple l'utilisateur ne se doutera pas qu'il utilise 2 domaines dans sa page Web.
En fait c'est le navigateur qui se charge de la sécurité se basant sur la politique de même origine, c'est à dire un document d'un domaine ne peut envoyer des données à un autre domaine.
Nous voyons que l'inclusion d'une seule balise d'image peut servir à détourner une application web vulnérable. Ce genre d'attaque est appelé "forgement de requêtes
sur d'autres sites" (Cross-Site Request Forgery ou CSRF, parfois XSRF), attaque par

commande d’URL, CSRF (URL Command Attack) ou encore passager clandestin d’une session (Session Riding).

Dans exemple7c.html, on ecrit le code suivant :

```
<html>
<a href="http://192.168.1.20"> Ceci est un exemple </a>
<img
style = "display:none"
src ="http://192.168.1.21/Partie2/XSS/exemple6s.php" width="100%">
</a>
</html>
```

Dans cet exemple l'utilisateur ne verra pas qu'il utilise le code de l'autre domaine.

Une manière de se protéger contre de telles attaques consiste à utiliser les cookies. Cependant il existe des utilitaires qui prévoit ou devine leurs valeurs.

Réponse Exercice #3 :

- Installation du module *mod_evasive*, ce module assure une défense contre les attaques par force brute.
Il faut l'installer et faire quelques modofocations dans le fichier httpd.conf du serveur Apache.
Exemple

```
 <IfModule mod_evasive20.c>
DOSHashTableSize 3097
DOSPageCount 2
DOSSiteCount 50
DOSPageInterval 1
DOSSiteInterval 1
DOSBlockingPeriod 10
DOSWhitelist 139.124.35.1
</IfModule>
```

- utiliser un .htaccess (protection du répertoire et de ses sous-répertoires) ou des sessions.

D'une manière générale, Utiliser un outil pour mesurer les performances lors de la montée en charge permet de donner une idée du
nombre de requêtes qu’un pirate aura à générer pour obtenir un déni de service (JMeter http://jakarta.apache.org/jmeter).
- Utiliser mod_evasive
- Régler les directives du httpd.conf qui limitent les ressources utilisées et le nombre de clients et de connexions persistantes simultanés (MaxClients, MaxRequestsPerChild, KeepAlive, MaxKeepAliveRequests, Timeout, KeepAliveTimout, RLimitMEM, RLimitCPU, RLimitNPROC, LimitRequestBody, LimitRequestFields, LimitRequestFieldSize, LimitRequestLine).
- Régler les directives du php.ini memory_limit, post_max_size, max_input_time, max_execution_time.

Sécurité par l'obscurité :

C'est une des formes de sécurité les plus faibles. *Il faut :*
- paramétrer correctement les fichiers de configuration php.ini et httpd.conf ;
- éventuellement recompiler le serveur web.

1. Informations données par Apache
- Mettre ServerSignature à Off dans le httpd.conf
- Mettre ServerTokens à Prod (ProductOnly) dans le httpd.conf
2. Informations données par PHP
- Mettre ServerTokens à Full l'information PHP disparaitra
- Mettre expose_php à Off dans le php.ini
3. Informations données par les extensions PHP
- Mettre dans le fichier httpd.conf
AddType application/x-httpd-php .asp .htm .html
- configurer le type d'application par défaut
DefaultType application/x-httpd-php
- localement dans un .htaccess (règle de réécriture)
rewriteEngine on
rewriteRule (.+).html$ $1.php

- Changer le nom par défaut donné au cookie de session dans la directive session.name du php.ini
4. Informations données par l'affichage des erreurs (Information leakage)

php.ini
- display_errors = off
- display_startup_errors = off
- log_errors = on
- error_log=chemin
- error_reporting = E_ALL

5. Informations données par les briques logicielles ou par les programmeurs

interdire l'utilisation de fonctions php qui fournissent des informations sur la version ou le système avec la directive du php.ini
- disable_functions = "php_uname, phpversion, show_source, highlight_file, mysql_get_server_info "

- open_basedir = "/web/:/tmp/"

6. Interdire l'indexation
On peut écrire un fichier robots.txt à la racine du site web pour indiquer tout ce qui ne doit pas être indexé (ce fichier ne doit pas contenir de ligne vide).

*User-agent: **
Disallow: /cgi-bin/
Disallow: /docs/

7. Filtrer les entrées et protéger les sorties

Programmeur PHP

Filtrer les entrées de l'application (vérifier toute donnée reçue avant de l'utiliser) par une approche de type liste
blanche :
- données à vérifier avant utilisation : $_GET, $_POST, $_COOKIE, $_REQUEST, $_FILES, $_SERVER, $_ENV et $_SESSION (si on a stocké des données non vérifiées en session).
- contrôles à effectuer :
type : vérifier que le type correspond à celui attendu : intval, ctype, … (voir plus bas « possibilités offertes par PHP pour vérifier les données »)
cast : pour plus de sécurité caster les données avant de les mettre dans des variables : $prix = (float) $_POST['prix'] ;
présence de tous les arguments attendus ;
pour les nombres : la valeur est comprise entre deux bornes ;
pour les listes : la valeur appartient à la liste des valeurs autorisées (select, radio, checkbox, …) ;
jeu de caractères ;
taille de la donnée : tailleMin < tailleDonnee < tailleMax ou tailleDonnee = N (utiliser strlen) ;
valeur nulle autorisée ou non ;
la valeur suit une expression régulière.

Protéger les sorties vers le client
- coder les caractères spéciaux de préférence avec htmlentities (ne pas oublier de protéger les messages d'erreur car ils contiennent souvent les données entrées) :
*** htmlspecialchars(chaine, ENT_QUOTES) retourne la chaîne avec les caractères spéciaux &, ", ', <, > transformés en entités html. L'argument permet de créer des entités aussi pour le caractère ' ;*
*** htmlentities(chaine[, ENT_QUOTES, encodage])retourne la chaîne avec des entités html pour tous les caractères qui ont une entité définie dans la DTD, l'encodage peut être défini (par défaut utilise iso-8859-1) ;*
- définir le jeu de caractères de la page (ISO-8859-1, UTF-8, etc) :
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

Administrateur

Réglage de php.ini

- error_reporting = E_ALL
- register_globals = off
- open_basedir = arborescence_autorisée;
- allow_url_fopen = off
- disable_functions = "liste _fonctions_non_autorisées" ; (extract, system, exec, …)
- magic_quotes_gpc = off

- Utiliser mod_security (filtrage des données)
- mettre en place safe_mode pour empêcher l'accès depuis un script à un autre script du serveur
- Interdire l'ouverture d'URL avec allow_url_fopen = off

8.Injection de codes et commandes

Pour se protéger – côté développeur :
- Filtrer les données de l'utilisateur avant de les utiliser par une approche liste blanche (cf. chapitre 5).

- Protéger les données utilisateur des méta-caractères (attention si magic_quotes_gpc = on il faudra enlever
les \ avant d'appliquer cette protection) :
o utiliser escapeshellcmd pour le nom de la commande,
o utiliser escapeshellarg pour chaque argument.
-
- Supprimer toute commande système inutile (s'il existe une fonction interne l'utiliser), par ex. en php un system("sendmail…") devrait être remplacé par un appel à la fonction mail.
- Protéger un nom de fichier dynamique avec basename pour inclure le fichier dans le cas où il est dans le
même répertoire ou dans un sous-répertoire (protection contre le directory traversal) ;
- Utiliser realpath qui donne le chemin absolu (développe les liens symboliques, change le . et le .., enlève les
séparateurs de répertoires lorsqu'ils sont doublés //) et vérifier ensuite que le propriétaire du script est autorisé à
accéder à la ressource.
- Initialiser les variables (le niveau d'erreur maximal affichera des alertes pour toute variable non initialisée).

Pour se protéger – côté administrateur :
- Interdire si possible les fonctions eval, system, exec, shell_exec, passthru, popen, … avec la directive disable_functions.
- register_globals = off
- Préciser le ou les répertoires d'inclusion autorisés pour le site avec la directive open_basedir.
- Effectuer les mises à jour des briques logicielles et les correctifs de sécurité.
- Mettre en place des règles de filtrage sur le pare-feu du serveur web pour limiter les connexions vers d'autres
sites web.
- Eventuellement mettre en place safe_mode pour empêcher l'accès depuis un script à un autre script du serveur
web dont l'utilisateur n'est pas propriétaire (directory traversal). Il faut cependant noter que la directive safe_mode est abandonnée dans PHP6) ;
- Ne jamais montrer les erreurs.
- Interdire l'ouverture d'URL avec allow_url_fopen = off (allow_url_include depuis PHP 5.2.0),
utiliser curl s'il faut permettre à des utilisateurs d'accéder à des URL.