Ministèredel’EnseignementSupérieuretdelaRechercheScientifique

**UniversitédesSciencesetdelaTechnologieHouariBoumediene**

Facultéd’Electroniqueetd’Informatique

DépartementInformatique

Filière: Informatique

Spécialité:

**SSI**

*Rapport de réalisation du site web vulnérable et la méthodologie permettant l’exploitation des vulnérabilités*

**Remettre le :23 /02 / 2021**

**Projet Proposé par:**

**M.BERBAR Ahmed**

**Présenté par :**

**OULDECHEFAOUI Ismail**

**MECERHED Ferhat**

Binôme N°  15

Sommaire

[Chapitre 1 : **Réalisation de site web** 3](#_Toc64930615)

[1 Introduction 3](#_Toc64930616)

[2 Structure de site web 3](#_Toc64930617)

[2.1 Structure générale 3](#_Toc64930618)

[3 Définition des vulnérabilités utilisé dans le site 6](#_Toc64930619)

[3.1 INJECTION SQL 6](#_Toc64930620)

[3.2 FORCE BRUTE 7](#_Toc64930621)

[3.3 INCLUSION DES FICHIERS DISTANTS "RFI" 8](#_Toc64930622)

[3.4 AUTHENTICATION BRISÉ " BROKEN AUTHENTICATION " 8](#_Toc64930623)

[3.5 INCLUSION DE FICHIER LOCAL "LFI" 9](#_Toc64930624)

[Chapitre 2: **Exploitation des vulnérabilités** 10](#_Toc64930625)

[1 introduction 10](#_Toc64930626)

[2 Méthodologie d’exploitation des vulnérabilités 10](#_Toc64930627)

[2.1 INJECTION SQL 10](#_Toc64930628)

[2.2 CRAQUAGE DE MOT DE PASSE PAR FORCE BRUTE 10](#_Toc64930629)

[2.3 AUTHENTIFICATION BRISEE 10](#_Toc64930630)

[2.4 TELECHARGEMENT DEFAILLANT 10](#_Toc64930631)

[2.5 INTRUSION DE FICHIER LOCAL 10](#_Toc64930632)

[3 conclusion 10](#_Toc64930633)

**Table des figures**

[Figure 1 index.php (as no admin) 4](#_Toc64930634)

[Figure 2 user.php 4](#_Toc64930635)

[Figure 3 article.php 5](#_Toc64930636)

[Figure 4 login.php 5](#_Toc64930637)

[Figure 5 index.php (as admin) 6](#_Toc64930638)

[Figure 6 SQL Injection 7](#_Toc64930639)

[Figure 7 Brute force 7](#_Toc64930640)

[Figure 8 Inclusion de fichiers distants 8](#_Toc64930641)

[Figure 9 Authentification brisée 9](#_Toc64930642)

[Figure 10 INCLUSION DE FICHIER LOCAL 9](#_Toc64930643)

# Chapitre 1 : **Réalisation de site web**

# Introduction

Dans ce chapitre nous allons définir le processus pour réaliser notre site web . Nous déterminerons les fonctionnalités de notre site , son architecture puis nous définitions les différents vulnérabilités existe dans le site web.

# Structure de site web

## Structure générale

Le site est implémenté sous la pile XAMPP (Apache, MySQL, PHP), trois nivaux d’accès sont disponibles :

* Invité : Etat d’un utilisateur non authentifié
* Compte utilisateur : Etat d’un utilisateur authentifié à un compte non administrateur
* Administrateur : Etat de l’utilisateur authentifié à l’**unique** compte administrateur dans la base de données. Permet le téléchargement de fichier au serveur.

**Mappage du site** **:**

/

|\_ index.php

|\_article.php

|\_user.php

|\_login.php

|\_users\_document

|\_ *Documents utilisateurs*

|\_database\_scripts

|\_createDB.php

|\_createTableUser.php

|\_createTableArt.php

|\_createTableToken.php

|\_insertDBA.php

**Description des différentes vues :**

* index.php : Contient un formulaire d’envoi d’emails (fictifs). Contient également un formulaire pour télécharger des fichiers vers le serveur à condition d’être administrateur.
* article.php : Imprimes les articles de la BDD avec un formulaire de filtre de recherche (point d’entrée de l’injection SQL).
* user.php : Imprimes les utilisateurs de la BDD avec un formulaire de filtre de recherche (insensible à l’injection)
* login.php : Permet de s’authentifier avec l’obtention d’un cookie (en local) contenant un jeton d’authentification.
* database\_scripts : Contient des scripts pour l’initialisation/population de la base de données (absents dans la version production)

Vue sur le premier plan des pages :



Figure index.php (as no admin)



Figure user.php

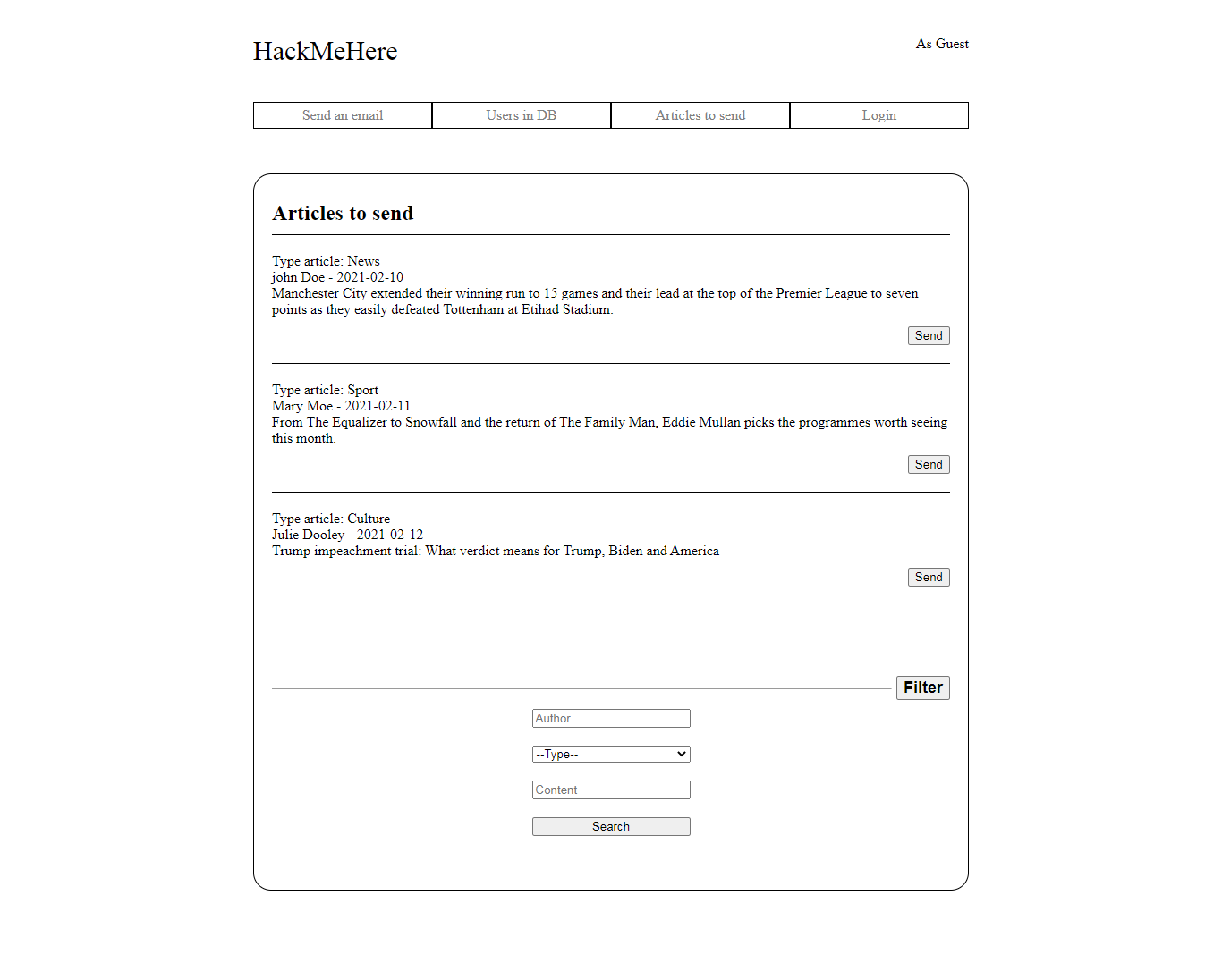


Figure article.php



Figure login.php



Figure index.php (as admin)

# Définition des vulnérabilités utilisé dans le site

Dans cette partie, nous allons donner une brève définition des vulnérabilités sur le site, et est de faciliter la question dans le chapitre 2, où nous procéderons directement sur la façon d'exploiter les vulnérabilités.

## INJECTION SQL

**Une injection SQL** est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données. L'attaquant détourne les requêtes en y injectant une chaîne non prévue par le développeur et pouvant compromettre la sécurité du système .

C'est un cas particulier d'un type de vulnérabilité plus général qui peut se manifester lorsqu'un langage est imbriqué dans un autre.

Lorsqu'on évoque l'injection SQL, on parle tout le temps de faille dans un site Web, mais il n'y a pas que les sites qui sont touchés par cette faille: n'importe quelle application dialoguant avec une base de données en utilisant des requêtes sur lesquelles l'utilisateur a une influence peut être vulnérable aux injections SQL.

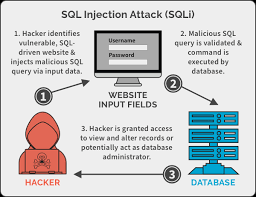


Figure SQL Injection

## FORCE BRUTE

L'**attaque par force brute** est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Il s'agit de tester, une à une, toutes les combinaisons possibles.

Cette méthode est en général considérée comme la plus simple concevable. Elle permet de casser tout mot de passe en un temps fini indépendamment de la protection utilisée, mais le temps augmente avec la longueur du mot de passe. En théorie la complexité d'une attaque par force brute est une fonction exponentielle de la longueur du mot de passe, la rendant en principe impossible pour des mots de passe de longueur moyenne; En pratique des optimisations heuristiques peuvent donner des résultats dans des délais beaucoup plus courts.

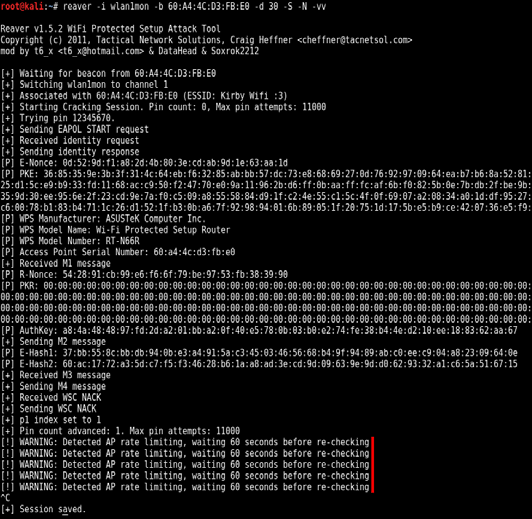


Figure Brute force

## INCLUSION DES FICHIERS DISTANTS "RFI"

**RFI**  (pour Remote File Inclusion) Cette attaque vise à inclure des fichiers externes au moment de l'exécution du code. Comme pour les autres attaques, la vulnérabilité est créée à cause de l'imprudence du développeur qui ne vérifie pas assez les entrées du site web, Si l'attaque RFI réussit alors elle peut causer plein d'ennuies comme par exemple:

* Exécution d'un code externe malveillant sur le site victime.
* Affichage du contenu d'un fichier local non autorisé en temps
* Affichage du code source PHP.

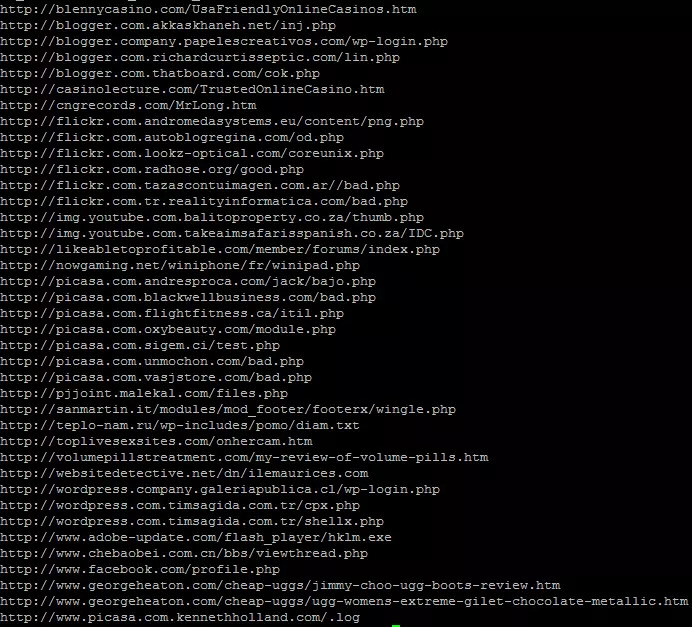


Figure Inclusion de fichiers distants

## AUTHENTICATION BRIS**É "** BROKEN AUTHENTICATION **"**

L'**authentification brisée** est un terme générique désignant plusieurs vulnérabilités que les attaquants exploitent pour se faire passer pour des utilisateurs légitimes en ligne. D'une manière générale, l'authentification brisée fait référence à des faiblesses dans deux domaines: la gestion de session et la gestion des informations d'identification. Les deux sont classés comme authentification brisée car les attaquants peuvent utiliser l'une ou l'autre des voies pour se faire passer pour un utilisateur: identifiants de session piratés ou identifiants de connexion volés.

Les attaquants utilisent une grande variété de stratégies pour tirer parti de ces faiblesses, allant des énormes attaques de bourrage d'informations d'identification aux stratagèmes hautement ciblés visant à accéder aux informations d'identification d'une personne spécifique.

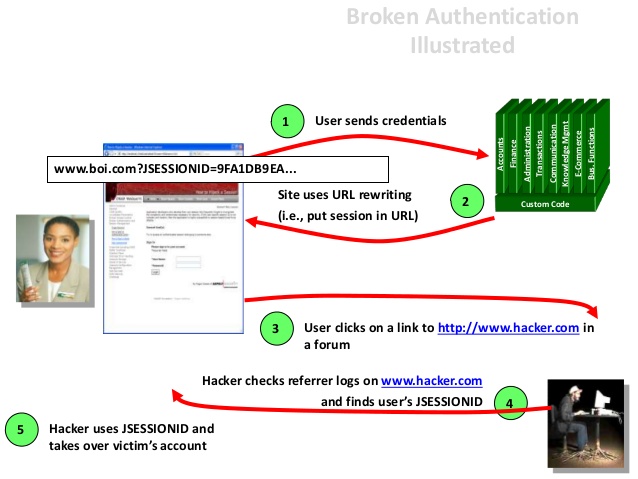


Figure Authentification brisée

## INCLUSION DE FICHIER LOCAL "LFI"

Lorsqu'une application utilise un chemin de fichier comme entrée, l'application traite cette entrée comme fiable et sûre. Un fichier local peut ensuite être injecté dans l'instruction incluse. Cela se produit lorsque votre code est vulnérable. Dans ce cas, un pirate fait une requête qui trompe l'application en exécutant un script PHP malveillant (web Shell par exemple). Le problème principal se produit lorsque l'utilisateur exécute ensuite l'application Web qui permet à son tour au pirate d'exécuter tout code malveillant côté serveur de son choix. Il n’existe pas non plus de moyen sûr de garantir que l’application enregistre le fichier sur le serveur sur lequel se trouve la vulnérabilité **LFI.**

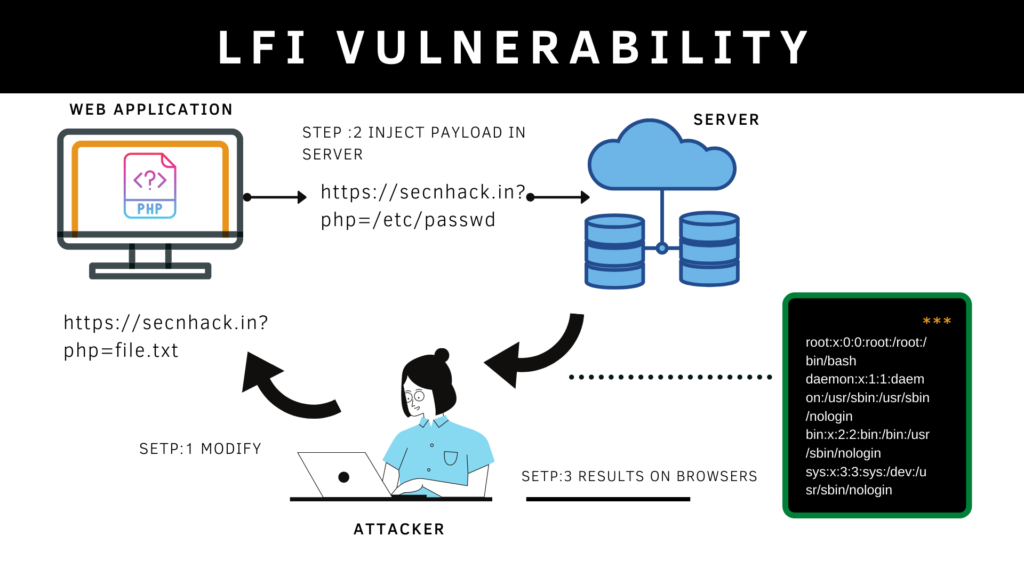


Figure INCLUSION DE FICHIER LOCAL

# Chapitre 2: **Exploitation des vulnérabilités**

# Introduction

Dans ce chapitre nous allons définir le processus qui sera suivi pour exploiter les vulnérabilités du site web. et prend le contrôle de la machine vulnérable.

# Méthodologie d’exploitation des vulnérabilités

# 2.1 Point d’entrée : injection SQL

Dans la page /article (qui affiches les articles de la base de données), un formulaire pour filtrer la recherche est rajouté. L’un des champs de ce formulaire (ici une balise <select>) n’est pas désinfecter pas le serveur, cela simule de véritable faille où le développeur assume que les entrées possibles sont restreints aux options données, or l’attribut *option* peut très bien être modifié en modifiant directement le DOM.

Le but de cette manœuvre est d’extraire les enregistrements de la table utilisateur, une commande similaire à celle décrite ci-dessous le permet : (évidement l’attaquant doit consulter en premier lieu les méta-tables pour connaître les noms des tables/colonnes)

' OR 1=1) UNION (SELECT 1, firstname, lastname, password, 5, password\_quality FROM MyUser); #

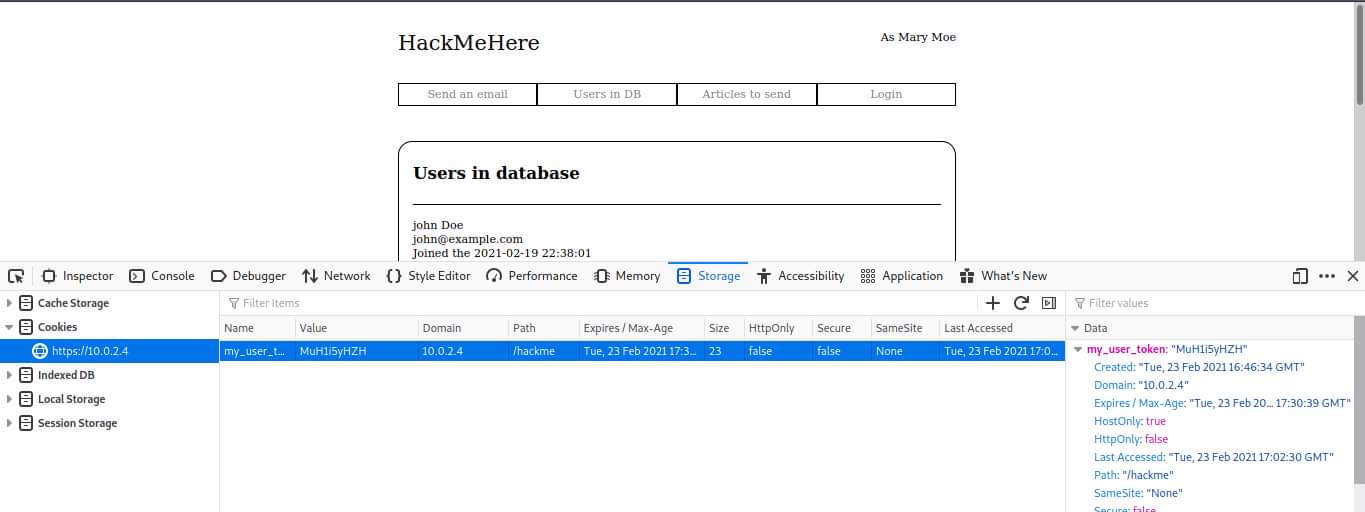
# E:\Xampp\htdocs\Projet_Security\screenshots\sql_injection.jpg

# 2.2 Craquage de mot de passe en force brute

Les mots de passe (chiffré en MD5) extrait pendant la phase précédente. Pour respecter la limite de temps, certain compte sont doté d’un mot de passe faible (de longueur X, sans caractère spéciaux). Une colonne décrivant la qualité du mot de passe est ajouté à chaque compte, cela permet à l’attaquant de savoir quel mot de passe peut être craqué dans un temps raisonnable.

# 2.3 Authentification brisée

Une fois le mot de passe déchiffré et l’email du compte en question est connu (phase 1), l’attaquant peut s’authentifier au compte utilisateur régulier. L’authentification se fait via un cookie contenant un jeton généré par le serveur à la création du serveur. Ce cookie étant stocké en local (machine de l’attaquant) il peut être modifié par un jeton d’un compte administrateur, jeton qui est récupérer durant la 1ère phase.

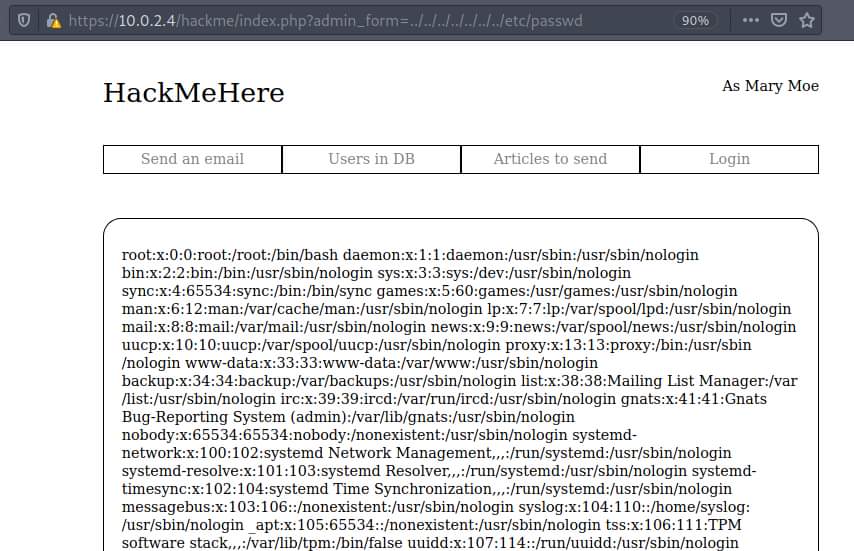


# 2.4 Téléchargement défaillant

Le compte administrateur donne accès à un nouveau formulaire permettant de joindre un fichier à la requête POST, ce dit formulaire ne vérifie que l’extension du fichier envoyé mais accepte TOUT fichier d’extension .txt, ce qui permet la mise en place de n’importe quel programme côté serveur

# 2.5 Inclusion de fichier local (LFI)

Le fichier étant téléchargé sur le serveur, il n’est cependant pas directement exécutable en vue du changement d’extension. Pour cela, le formulaire administrateur décrit plus haut est affiché en passant en paramètre un fichier .php à la page index.php, paramètre qui est naïvement exécuté. Cela permet l’exécution de n’importe quel script, celui téléchargé entre autres.



# Conclusion

En conclusion, nous avons appris comment créer des vulnérabilités dans les sites web et comment les exploiter pour prend la contrôle de site web puis obtenir l'accès au machine vulnérable et prendre le contrôle totale de la machine.