

Login Spam :

code à rajouter au début de la page auth_session.php

```
//-----  
// Database connection  
if (!isset($conn)) {  
    $conn = mysqli_connect(DB_HOST, DB_USER, DB_PASS, DB_NAME);  
}  
  
// Détection des tentatives  
$current_time = time();  
$threshold = 10; // Nombre total de tentatives autorisées dans la  
fenêtre de temps  
$block_duration = 30; // Durée de blocage en secondes  
  
// Vérifier les tentatives globales  
$query = "SELECT attempts, last_attempt FROM global_login_attempts  
LIMIT 1";  
$result = mysqli_query($conn, $query);  
  
if ($result && mysqli_num_rows($result) > 0) {  
    $row = mysqli_fetch_assoc($result);  
    $attempts = $row['attempts'];  
    $last_attempt = $row['last_attempt'];  
  
    // Si les tentatives dépassent le seuil et que le blocage est  
encore actif  
    if ($attempts >= $threshold && ($current_time - $last_attempt) <  
$block_duration) {  
        header("HTTP/1.1 429 Too Many Requests");  
        die("<h1>Site sous attaque DDoS. Veuillez réessayer plus  
tard.</h1>");  
    } elseif (($current_time - $last_attempt) >= $block_duration) {  
        // Réinitialiser les tentatives après la durée de blocage  
        $query = "UPDATE global_login_attempts SET attempts = 0,  
last_attempt = ?";  
        $stmt = $conn->prepare($query);  
        $stmt->bind_param("i", $current_time);  
        $stmt->execute();  
    }  
}
```

```

} else {
    // Insérer un nouvel enregistrement si aucune tentative n'a encore
    été enregistrée
    $query = "INSERT INTO global_login_attempts (attempts,
last_attempt) VALUES (0, ?)";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("i", $current_time);
    $stmt->execute();
}

// Incrémenter les tentatives globales en cas d'accès au formulaire
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $query = "UPDATE global_login_attempts SET attempts = attempts + 1,
last_attempt = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("i", $current_time);
    $stmt->execute();
}

//-----

```

ajouter une table sql :

```

CREATE TABLE global_login_attempts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    attempts INT DEFAULT 0,
    last_attempt INT
);

```

Scripte python :

```
import requests
import threading
import time

# Paramètres de la cible
URL = "http://localhost:2024/pages/public/login.php"
DATA = {
    "email": "test@example.com",
    "password": "password123"
}

THREADS = 1 # Nombre de threads
REQUESTS_PER_THREAD = 10 # Nombre de requêtes par thread

def send_requests():
    for _ in range(REQUESTS_PER_THREAD):
        try:
            response = requests.post(URL, data=DATA)
            print(f"Requête envoyée avec statut {response.status_code}")
        except requests.exceptions.RequestException as e:
            print(f"Erreur lors de l'envoi de la requête : {e}")

# Lancer plusieurs threads
threads = []

for i in range(THREADS):
    thread = threading.Thread(target=send_requests)
    threads.append(thread)
    thread.start()

# Attendre que tous les threads soient terminés
for thread in threads:
    thread.join()

print("Simulation terminée.")
```

```
Requête envoyée avec statut 200
Requête envoyée avec statut 200
Requête envoyée avec statut 200
Requête envoyée avec statut 200
Requête envoyée avec statut 200
Requête envoyée avec statut 200
Requête envoyée avec statut 200
Requête envoyée avec statut 429
Requête envoyée avec statut 429
Requête envoyée avec statut 429
Requête envoyée avec statut 429
Requête envoyée avec statut 429
Requête envoyée avec statut 429
```

429 c'est les requetes refusées après le ddos

YouTube Traduire Paraphrasing Tool Q... Microsoft Office Accueil Free Notepad Online ... // Autres marq

Site sous attaque DDoS. Veuillez réessayer plus tard.

Spam mail :

à modifier la fonction send_email

```
//-----
function send_email($email_cont) {
    global $conn;

    // Détection des tentatives globales d'envoi d'e-mails
    $current_time = time();
    $threshold = 2; // Nombre total d'e-mails autorisés par minute
    $block_duration = 3000; // Durée de blocage en secondes

    // Vérifier les tentatives globales
    $query = "SELECT attempts, last_attempt FROM global_email_attempts
LIMIT 1";
    $result = mysqli_query($conn, $query);

    if ($result && mysqli_num_rows($result) > 0) {
        $row = mysqli_fetch_assoc($result);
        $attempts = $row['attempts'];
        $last_attempt = $row['last_attempt'];

        if ($attempts >= $threshold && ($current_time - $last_attempt)
< $block_duration) {
            // Retourne un code HTTP 429 au lieu d'exécuter le reste de
la fonction
            header("HTTP/1.1 429 Too Many Requests");
            echo "<html>
                <head>
                    <title>Service temporairement
indisponible</title>
                </head>
                <body style='font-family: Arial, sans-serif;
text-align: center; margin-top: 20%;'>
                    <h1 style='color: red;'>Le système d'envoi
d'e-mails est temporairement bloqué</h1>
                    <p>Nous avons détecté un nombre élevé de
tentatives d'envoi. Veuillez réessayer dans quelques minutes.</p>
```

```

        <p><strong>Temps restant estimé :</strong> " .
ceil(($block_duration - ($current_time - $last_attempt)) / 60) . "
minute(s).</p>

        </body>
    </html>";

    exit; // Empêche toute exécution supplémentaire
} elseif (($current_time - $last_attempt) >= $block_duration) {
    // Réinitialiser les tentatives après la durée de blocage
    $query = "UPDATE global_email_attempts SET attempts = 0,
last_attempt = ?";

    $stmt = $conn->prepare($query);
    $stmt->bind_param("i", $current_time);
    $stmt->execute();
}

} else {
    // Insérer un nouvel enregistrement si aucune tentative n'a
encore été enregistrée
    $query = "INSERT INTO global_email_attempts (attempts,
last_attempt) VALUES (0, ?)";

    $stmt = $conn->prepare($query);
    $stmt->bind_param("i", $current_time);
    $stmt->execute();
}

// Incrémenter les tentatives globales
$query = "UPDATE global_email_attempts SET attempts = attempts + 1,
last_attempt = ?";
$stmt = $conn->prepare($query);
$stmt->bind_param("i", $current_time);
$stmt->execute();

// Envoi des e-mails
$sender_id = $email_cont['sender_id'];
$receiver_ids = get_id_by_email($email_cont['emails']);
$title = $email_cont['title'];
$content = $email_cont['content'];
$date = $email_cont['date'];
$error_not_sent = [];

for ($i = 0; $i < count($receiver_ids); $i++) {
    $receiver_id = $receiver_ids[$i];

```

```

        $query = "INSERT INTO Mails (sender_id, receiver_id, title,
content, date) VALUES ('$sender_id', '$receiver_id', '$title',
'$content', '$date')";
        $result = mysqli_query($conn, $query);
        if (!$result) {
            array_push($error_not_sent, $receiver_id);
        }
    }
    return $error_not_sent;
}

//-----

```

base de données à rajouter :

```

CREATE TABLE global_email_attempts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    attempts INT DEFAULT 0,
    last_attempt INT
);

```

Le système d'envoi d'e-mails est temporairement bloqué

Nous avons détecté un nombre élevé de tentatives d'envoi. Veuillez réessayer dans quelques minutes.

Temps restant estimé : 45 minute(s).

Scripte Python

```

import requests
import threading
import time

# Configuration
BASE_URL = "http://localhost:2024"

```

```

LOGIN_URL = f"{BASE_URL}/pages/public/login.php"
MAIL_URL = f"{BASE_URL}/pages/user/mail.php"
THREADS = 1 # Nombre de threads concurrents
REQUESTS_PER_THREAD = 5 # Nombre de requêtes par thread

# Informations de connexion
LOGIN_DATA = {
    "email": "yoann.constans@insa-cvl.fr",
    "password": "123456"
}

# Données pour envoyer un mail
MAIL_DATA = {
    "sender_id": 2, # ID valide de l'expéditeur
    "emails[]": ["xuan_phuc.pham@insa-cvl.fr"], # Doit correspondre à
l'attente de send_email
    "title": "Test DDoS",
    "content": "This is a test email.",
    "date": "2025-01-01 00:00:00"
}

# Fonction pour se connecter et envoyer des mails
def send_mails():
    with requests.Session() as session:
        # Étape 1 : Se connecter
        login_response = session.post(LOGIN_URL, data=LOGIN_DATA)
        if login_response.status_code != 200:
            print(f"Échec de la connexion :
{login_response.status_code}")
            return

        print("Connecté avec succès.")

        # Étape 2 : Envoyer plusieurs mails
        for _ in range(REQUESTS_PER_THREAD):
            try:
                mail_response = session.post(MAIL_URL, data=MAIL_DATA)
                if mail_response.status_code == 200:
                    print("Mail envoyé avec succès.")
                else:
                    print(f"Échec de l'envoi du mail, statut :
{mail_response.status_code}")
            except requests.RequestException as e:

```



```

        print(f"Erreur lors de l'envoi du mail : {e}")

# Lancer plusieurs threads pour simuler un DDoS
threads = []

for i in range(THREADS):
    thread = threading.Thread(target=send_mails)
    threads.append(thread)
    thread.start()

# Attendre que tous les threads soient terminés
for thread in threads:
    thread.join()

print("Simulation terminée.")

```

Scripte fonctionne pas encore mais c'est pas grave car logiquement l'envoi se fait depuis la page de l'envoi

Register Spam :

ajouter ce code au début dans la page register.php

```

<?php
require "../controller/config.php";
require ROOT_PATH . DS . "controller" . DS . "auth_session.php";

// Détection des tentatives globales d'enregistrement
$current_time = time();
$threshold = 3; // Nombre d'enregistrements autorisés par minute
$block_duration = 300; // Durée de blocage en secondes

// Connexion à la base de données
if (!isset($conn)) {
    $conn = mysqli_connect(DB_HOST, DB_USER, DB_PASS, DB_NAME);
    if (!$conn) {
        die("Erreur de connexion à la base de données : " .
mysqli_connect_error());

```

```

    }
}

// Vérifier les tentatives globales
$query = "SELECT attempts, last_attempt FROM global_register_attempts
LIMIT 1";
$result = mysqli_query($conn, $query);

if ($result && mysqli_num_rows($result) > 0) {
    $row = mysqli_fetch_assoc($result);
    $attempts = $row['attempts'];
    $last_attempt = $row['last_attempt'];

    // Bloquer si le seuil est atteint
    if ($attempts >= $threshold && ($current_time - $last_attempt) <
$block_duration) {
        echo "<html>
            <head>
                <title>Service temporairement indisponible</title>
            </head>
            <body style='font-family: Arial, sans-serif;
text-align: center; margin-top: 20%;'>
                <h1 style='color: red;'>Le service d'inscription
est temporairement bloqué</h1>
                <p>Nous avons détecté un nombre élevé de tentatives
d'enregistrement. Veuillez réessayer dans quelques minutes.</p>
                <p><strong>Temps restant estimé :</strong> " .
ceil(($block_duration - ($current_time - $last_attempt)) / 60) . "
minute(s).</p>
            </body>
        </html>";
        exit; // Empêche toute exécution supplémentaire
    } elseif (($current_time - $last_attempt) >= $block_duration) {
        // Réinitialiser les tentatives après la durée de blocage
        $query = "UPDATE global_register_attempts SET attempts = 0,
last_attempt = ?";
        $stmt = $conn->prepare($query);
        $stmt->bind_param("i", $current_time);
        $stmt->execute();
    }
} else {
    // Insérer un nouvel enregistrement si aucune tentative n'a encore
été enregistrée

```

```

        $query = "INSERT INTO global_register_attempts (attempts,
last_attempt) VALUES (0, ?)";
        $stmt = $conn->prepare($query);
        $stmt->bind_param("i", $current_time);
        $stmt->execute();
    }

    // Incrémenter les tentatives globales
    $query = "UPDATE global_register_attempts SET attempts = attempts + 1,
last_attempt = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("i", $current_time);
    $stmt->execute();

include ROOT_PATH . DS . "components" . DS . "header.php";
require ROOT_PATH . DS . "components" . DS . "nav_bar.php";
?>

```

Le service d'inscription est temporairement bloqué

Nous avons détecté un nombre élevé de tentatives d'enregistrement. Veuillez réessayer dans quelques minutes.

Temps restant estimé : 5 minute(s).

ajouter la base de donnée suivante

```

CREATE TABLE global_register_attempts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    attempts INT DEFAULT 0,
    last_attempt INT
);

```

Scripte python :

```
import requests
import threading

# Configuration
BASE_URL = "http://localhost:2024"
REGISTER_URL = f"{BASE_URL}/pages/public/register.php"
THREADS = 1 # Nombre de threads concurrents
REQUESTS_PER_THREAD = 2 # Nombre de requêtes par thread

# Données pour l'inscription
REGISTER_DATA = {
    "nom": "TestNom",
    "prenom": "TestPrenom",
    "email": "testt{counter}@example.com", # Ajout d'un compteur pour
    éviter les doublons
    "departement": "STI",
    "sexe": "M",
    "date_naissance": "2000-01-01",
    "adresse": "12ff, rue de la rue",
    "telephone": "1234567890",
    "annee": "1A",
    "password1": "Password123",
    "password2": "Password123",
    "register_btn": "Register"
}

# Fonction pour envoyer les tentatives d'inscription
def register_attempts(thread_id):
    with requests.Session() as session:
        for i in range(REQUESTS_PER_THREAD):
            try:
                # Mettre à jour l'adresse e-mail pour chaque tentative
                data = REGISTER_DATA.copy()
                data["email"] = f"test{thread_id * REQUESTS_PER_THREAD
+ i}@example.com"

                response = session.post(REGISTER_URL, data=data)
```

```
        print(f"Tentative {i + 1} (Thread {thread_id}) : Statut  
{response.status_code} - Réponse {response.text[:100]}")  
    except requests.RequestException as e:  
        print(f"Erreur lors de la tentative d'inscription :  
{e}")  
  
# Lancer plusieurs threads pour simuler des requêtes simultanées  
threads = []  
  
for thread_id in range(THREADS):  
    thread = threading.Thread(target=register_attempts,  
args=(thread_id,))  
    threads.append(thread)  
    thread.start()  
  
# Attendre que tous les threads soient terminés  
for thread in threads:  
    thread.join()  
  
print("Simulation terminée.")
```

Maintenance mode :

rajouter ce code dans les pages suivantes :

dans functions/public : contact_info.php / forum.php

dans pages/public : register.php/cours.php (ca doit etre mis avant `include ROOT_PATH . DS . "components" . DS . "header.php";`)

dans controller/api : auth_sessions.php

dans pages/user : mail.php / send_email.php / notes.php / new_post.php / annonce_add.php / annonce.php tous le reste des pages

```
require "../..controller/config.php";
require ROOT_PATH . DS . "functions" . DS . "admin" . DS .
"maintenance_status.php";

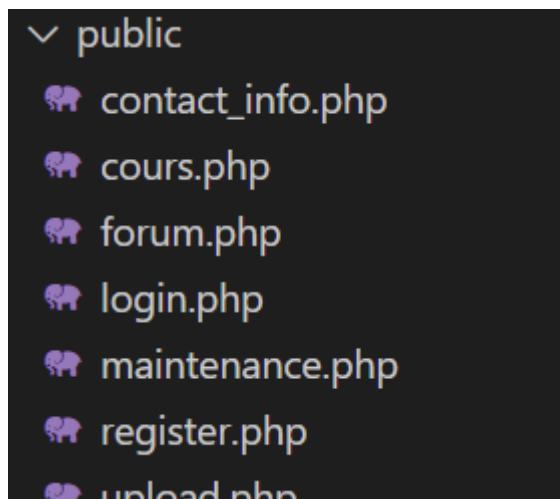
if ($maintenance == 1) {
    header("Location: /pages/public/maintenance.php");
    exit;
}
```

ce que ça donne quand on ouvre une page :

Site en Maintenance

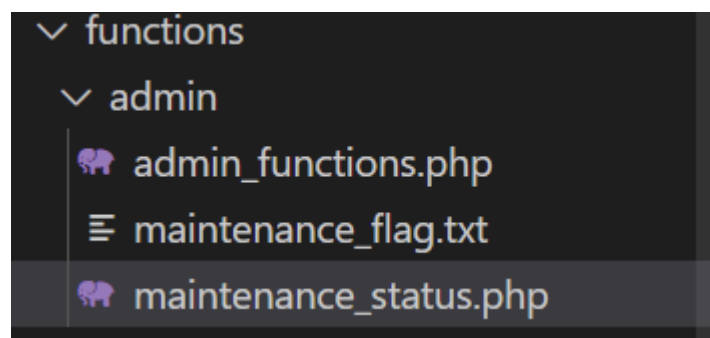
Le site est actuellement en maintenance. Veuillez réessayer plus tard.

il faut rajouter la page maintenance.php dans pages/public :



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Maintenance</title>
</head>
<body style="font-family: Arial, sans-serif; text-align: center;
margin-top: 20%;">
  <h1 style="color: red;">Site en Maintenance</h1>
  <p>Le site est actuellement en maintenance. Veuillez réessayer plus
tard.</p>
</body>
</html>
```


dans functions/admin rajouter maintenance_status.php :



```
<?php
// Fichier où sera sauvegardé l'état de maintenance
$maintenance_file = __DIR__ . '/maintenance_flag.txt';

// Lire l'état actuel de la maintenance
if (file_exists($maintenance_file)) {
    $maintenance = file_get_contents($maintenance_file);
} else {
    $maintenance = 0; // Par défaut, pas de maintenance
}

if (isset($_GET['update']) && isset($_GET['value'])) {
    $maintenance = intval($_GET['value']);
    file_put_contents($maintenance_file, $maintenance);
}
?>
```

et maintenance_flag.txt :

mettre juste 0 ou 1

<http://localhost:2024/functions/public/forum.php?update=1&value=1>

Remarques générales :

Les modes anti spam sont à notre avantage et c'est à eux de notifier les problèmes suivant et les corriger

- Les mode maintenance bloquent les pages pour tous le monde et il faut qu'ils corrigent soit pour bloquer un compte ou une ip spécifique
- Le anti spam pour la page register compte l'actualisation de la page comme une requête et ca compte pour le span
- Le mode spam de register bloque toute la page au lieu de ca il doit bloquer le compte visé à la place

Pour le mode maintenance :

Le mode maintenance est vulnérable au injection xss avec la fonction update qui fonctionne avec la requête Guet .

Bloquer d'ip :

rajouter le code suivant dans :

auth_session.php (juste après `if (isset($_POST['login_btn']))`) {
et dans la page : register.php

```
// Chemin vers le fichier contenant les IP bloquées
$blocked_ips_file = ROOT_PATH . "/functions/admin/blocked_ips.txt";

// Fonction pour vérifier si une IP est bloquée
function isIpBlocked($ip, $blocked_ips_file) {
    if (!file_exists($blocked_ips_file)) {
        echo "<html>
            <head>
                <title>fichier non trouvé</title>
            </head>
        </html>";
    }

    // Lire toutes les IP bloquées depuis le fichier
    $blocked_ips = file($blocked_ips_file, FILE_IGNORE_NEW_LINES |
FILE_SKIP_EMPTY_LINES);
    return in_array($ip, $blocked_ips); // Vérifie si l'IP actuelle est
dans la liste
}

// Obtenir l'adresse IP de l'utilisateur
$user_ip = $_SERVER['REMOTE_ADDR']; // Récupère l'IP de l'utilisateur
```

```

echo $user_ip;

// Vérifier si l'IP est bloquée
if (isIpBlocked($user_ip, $blocked_ips_file)) {
    // Afficher une page HTML si l'IP est bloquée
    echo "<html>
        <head>
            <title>IP Bloquée</title>
        </head>
        <body style='font-family: Arial, sans-serif; text-align:
center; margin-top: 20%;'>
            <h1 style='color: red;'>Ton IP est bloquée</h1>
            <p>Reviens plus tard.</p>
        </body>
    </html>";
    exit; // Empêche toute exécution supplémentaire
}

//-----

```

dans fonctions public rajouter le code suivant dans functions/public/ blocked_ips.php

```

<?php

// Inclure les fonctions
require "../controller/config.php";

// Chemin vers le fichier contenant les IP bloquées
$blocked_ips_file = ROOT_PATH . "/functions/admin/blocked_ips.txt";

// Vérifier les paramètres dans l'URL
if (isset($_GET['action']) && isset($_GET['ip'])) {
    $ip = $_GET['ip']; // Vulnérable à une injection XSS pour les tests

    if ($_GET['action'] === 'block') {
        blockIp($ip, $blocked_ips_file);
    }
}

```

```

        echo "<script>alert('IP bloquée : {$ip}')

```

```
// Réécrire le fichier avec les IP restantes
file_put_contents($blocked_ips_file, implode(PHP_EOL, $updated_ips)
. PHP_EOL);
}
?>
```

http://localhost:2024/functions/public/blocked_ips.php?action=block&ip=172.18.0.1

ici le problème et que la fonction utilise un fichier txt au lieu d'une base de donnée sécurisée et le fichier qui contient les fonctions est dans les fonctions publiques au lieu de celle de admin .