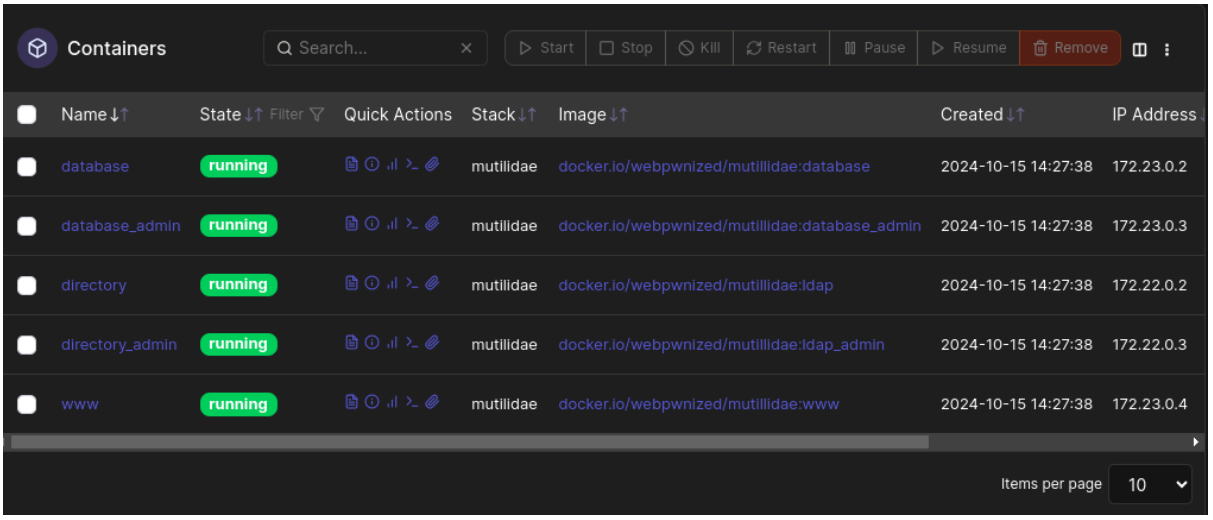
SIO2 – BLOC3 OWASP - ACTIVITÉ 3

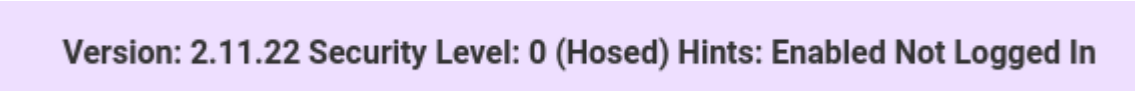
Mohamed
Hafaidh
BTS-SIO-2

VI Premier défi : XSS réfléchi via un contexte HTML

Travail à faire 1

Q1. Commencer par préparer votre environnement de travail en démarrant le serveur Mutillidae ainsi que la machine cliente contenant le proxy BurpSuite. Vérifier que vos deux machines peuvent communiquer via un ping. Puis, positionner le niveau de sécurité de Mutillidae à 0.

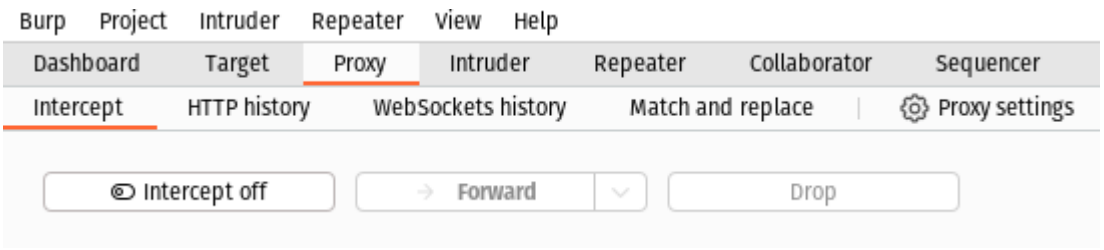




Q2. À l'aide du dossier documentaire n°1 situé en page 10, réaliser le premier défi permettant de capturer le cookie d'identification de la victime. Vous prendrez soin de réaliser vos propres captures d'écrans dans votre compte rendu de TP.

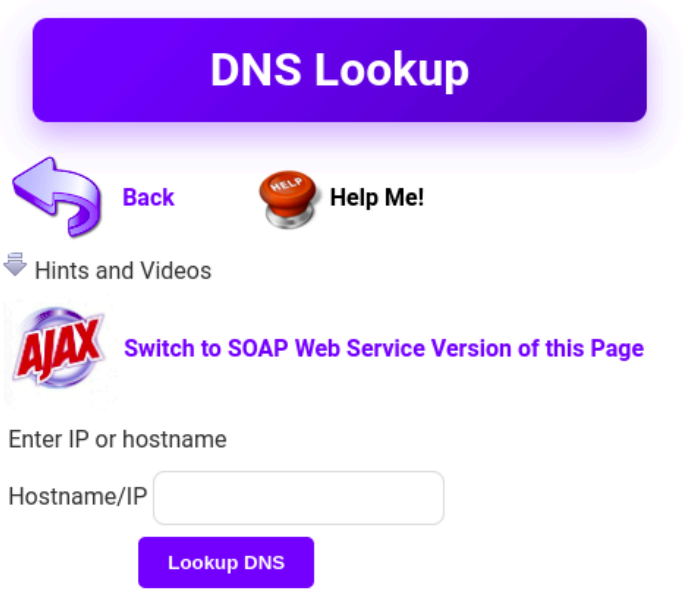
Dossier 1 : XSS réfléchi via un contexte HTML

Positionner le proxy à intercept off puis ouvrir la page suivante :

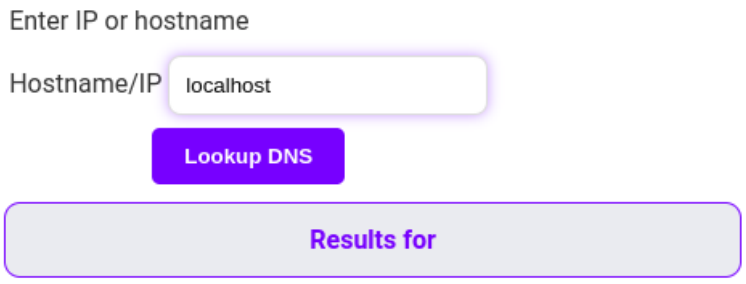


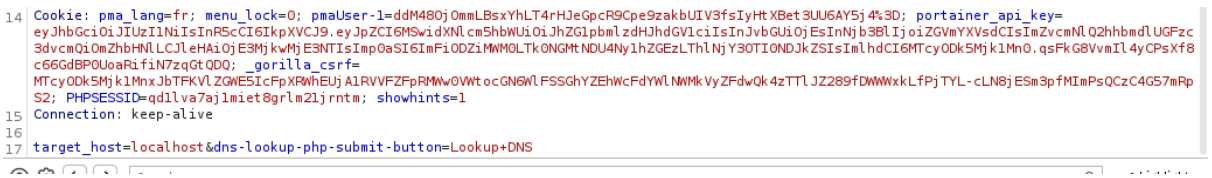
Aller dans OWASP 2017 => A7 : Cross Site Scripting (XSS) => Reflected (First Order) => DNS Lookup

---



Taper localhost dans le Hostname/IP puis cliquer sur Lookup DNS, n'oubliez pas d'activer le intercept en on :

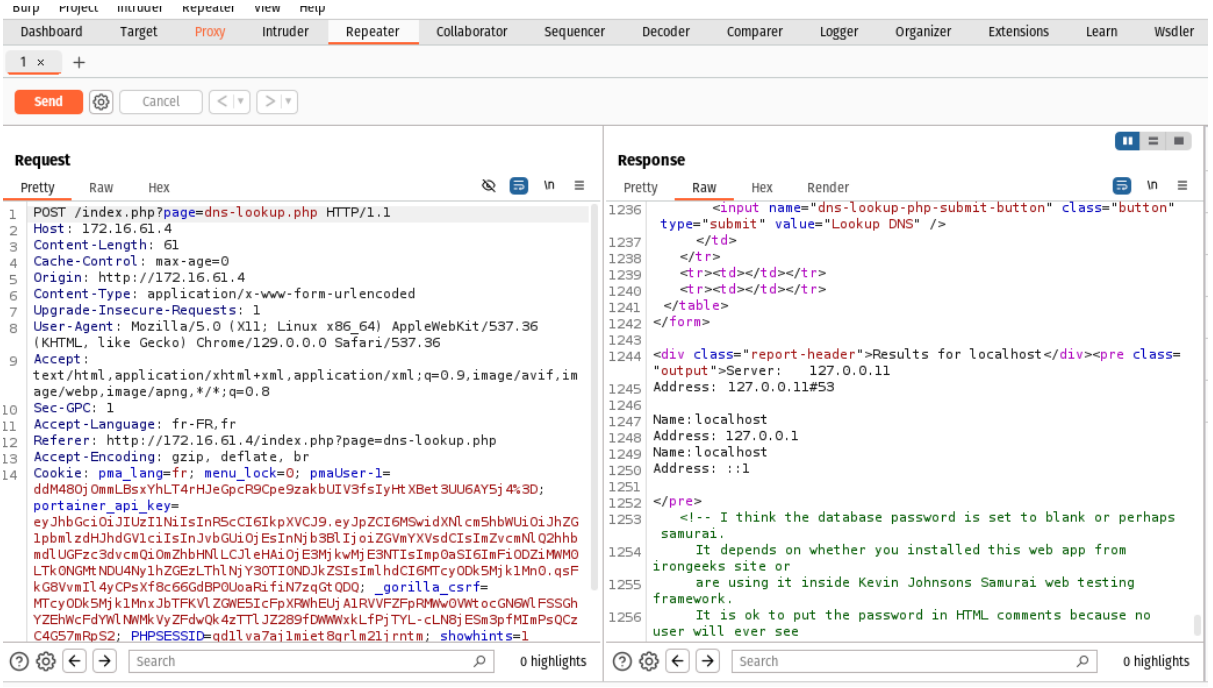


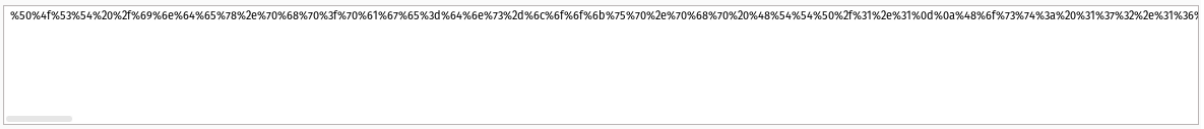


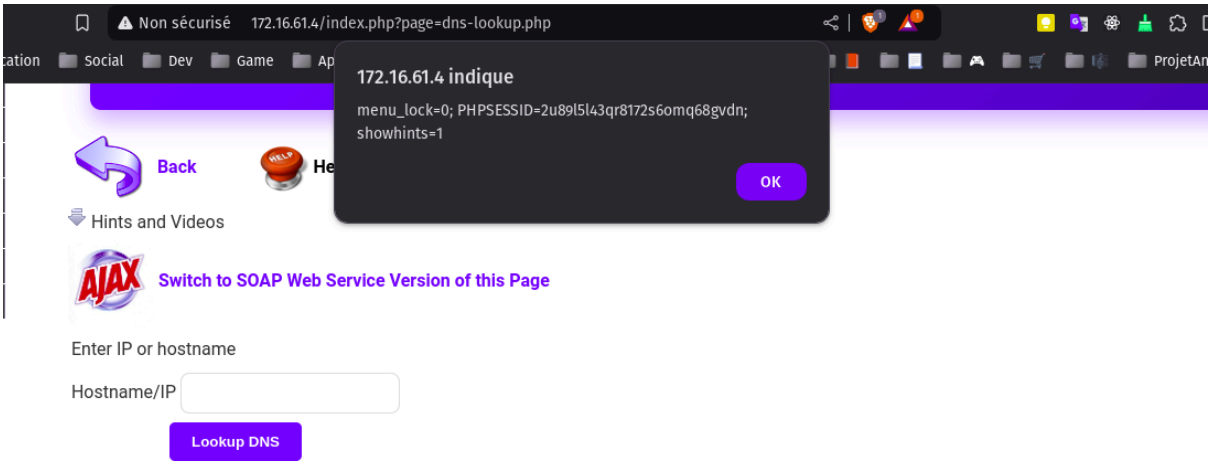
```
Content-Length: 61
Cache-Control: max-age=0
Origin: http://172.16.61.4
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
Sec-GPC: 1
Accept-Language: fr-FR,fr
Referer: http://172.16.61.4/index.php?page=dns-lookup.php
Accept-Encoding: gzip, deflate, br
Cookie: pma_lang=fr; menu_lock=0; pmaUser-1=ddM48OjOmmLBsxYhLT4rHJeGpcR9Cpe9zakbUIV3fsIyHtXBet3UU6AY5j4%3D; portainer_api_key=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJhZG1pbmlzdHJhdGVlciIsInJvbGUiOjEsInNjb3BlIjoiZGVmYXVsdCIsImZvcmNlQ2hhbmdlUGFzc3dvcmQiOmZhbHNlLCJleHAiOjE3MjkwMjE3NTIsImp0aSI6ImFiODZiMWM0LTk0NGMtNDU4Ny1hZGEzLThlNjY3OTI0NDJkZSIsImlhdCI6MTcyODk5Mjk1Mn0.qsFkG8VvmIl4yCPsXf8c66GdBP0UoaRifiN7zqGtQDQ; _gorilla_csrf=
MTcyODk5Mjk1MnxJbTFKVlZGWE5IcFpXRWhEUjA1RVVFZFpRMWw0VWtocGN6WlFSSGhYZEhWcFdYWlNWMkVyZFdwQk4zTTlJZ289fDWWWxkLfPjTYL-cLN8jESm3pfMImPsQCzC4G57mRpS2; PHPSESSID=qdllva7ajlmiet8grlm21jrntm; showhints=1
Connection: keep-alive

target_host=<script>alert(document.cookie);</script>localhost&dns-lookup-php-submit-button=Lookup+DNS
```

Étape n°3 : Création et exploitation du payload



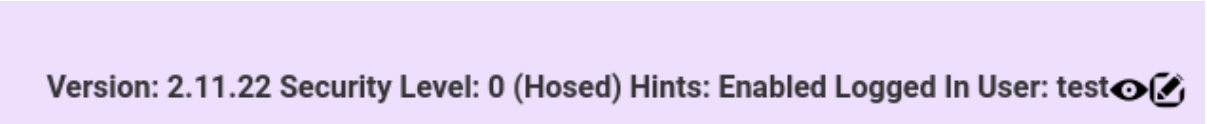




Dossier 2 : XSS permanent via une page affichant des logs

Positionner le proxy BurpSuite à intercept off.

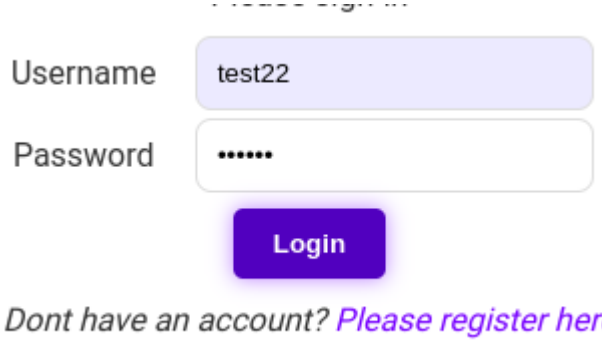
S’authentifier avec un compte inexistant login/register (utilisateur test par exemple).



Ouvrir la page suivante : Others => Denial of service => Show Web Log.

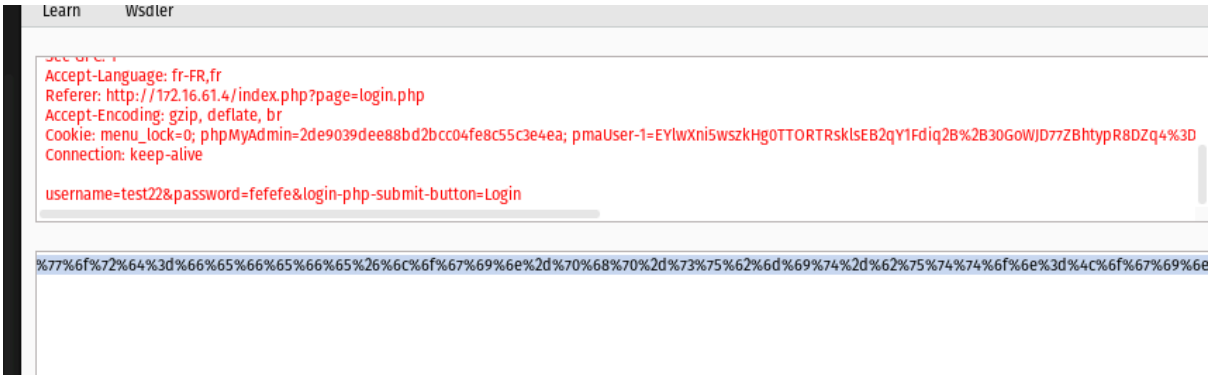
La page qui s'ouvre permet d'afficher les traces (logs) des tentatives d'authentification.

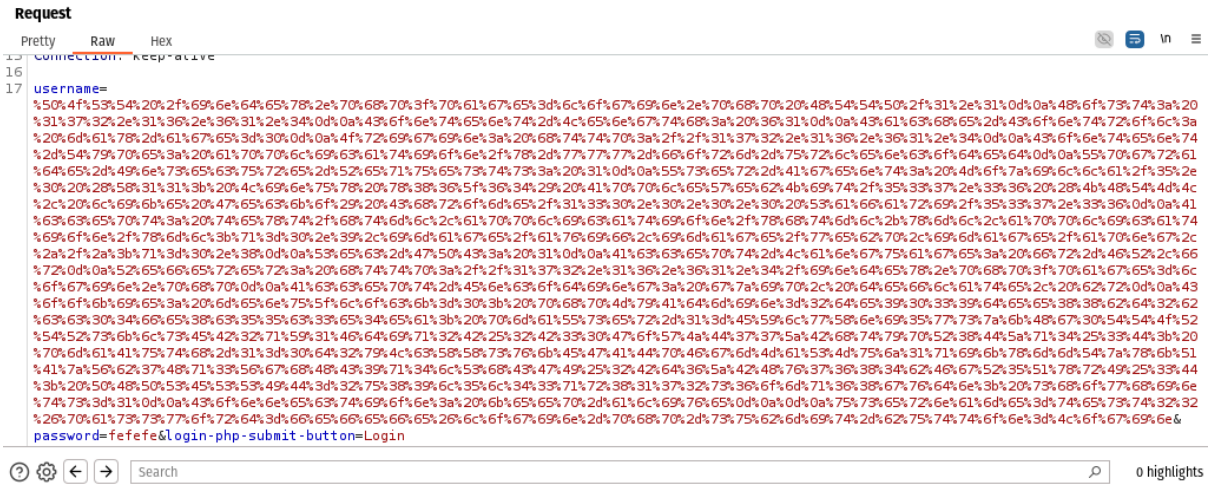
Hostname	IP	Browser Agent	Message
172.16.5.110	172.16.5.110	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36	User test attempting to authenticate
172.16.5.110	172.16.5.110	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36	Login Succeeded: Logged in user: test (39)

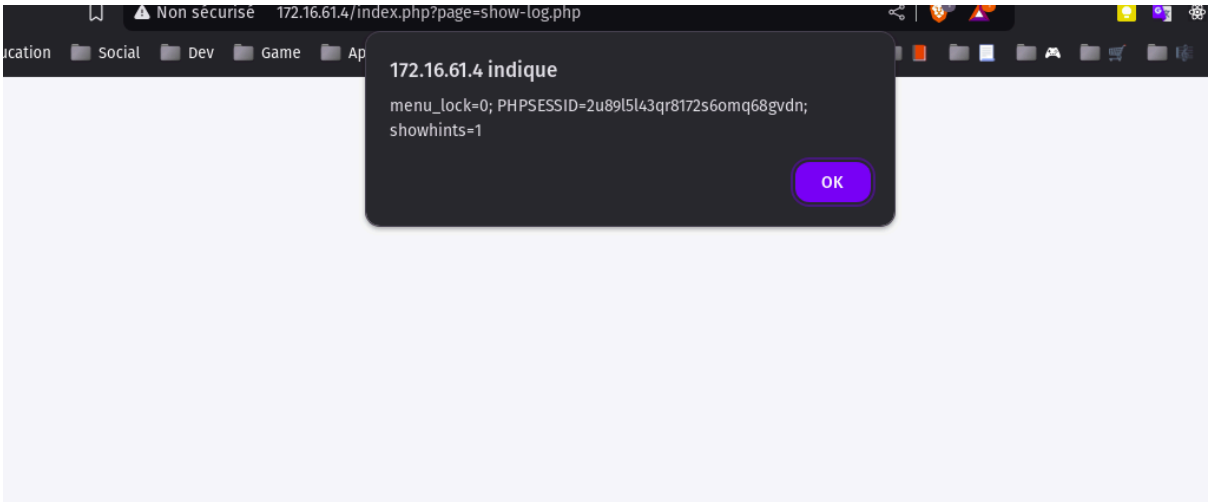


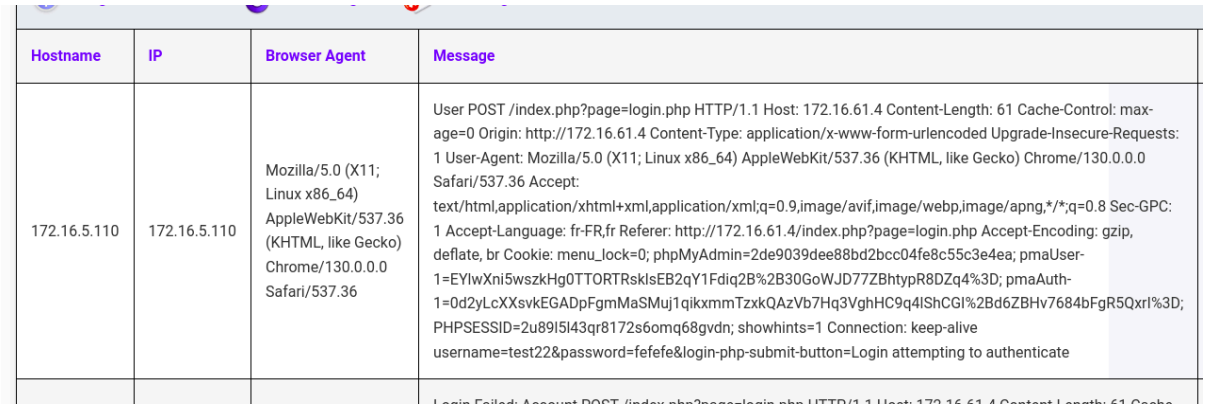

```
Sec-GPC: 1
Accept-Language: fr-FR,fr
Referer: http://172.16.61.4/index.php?page=login.php
Accept-Encoding: gzip, deflate, br
Cookie: menu_lock=0; phpMyAdmin=2de9039dee88bd2bcc04fe8c55c3e4ea
 pmaAuth-1=0d2yLcXXsvkEGADpFgmMaSMujlqikxmmTzxkQAzVb7Hq3VghHC9q4
showhints=1
Connection: keep-alive

username=test22&password=fefefe&login-php-submit-button=Login
```









Hostname	IP	Browser Agent	Message
172.16.5.110	172.16.5.110	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36	User POST /index.php?page=login.php HTTP/1.1 Host: 172.16.61.4 Content-Length: 61 Cache-Control: max-age=0 Origin: http://172.16.61.4 Content-Type: application/x-www-form-urlencoded Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8 Sec-GPC: 1 Accept-Language: fr-FR,fr Referer: http://172.16.61.4/index.php?page=login.php Accept-Encoding: gzip, deflate, br Cookie: menu_lock=0; phpMyAdmin=2de9039dee88bd2bcc04fe8c55c3e4ea; pmaUser-1=EYlwXni5wszkHg0TTORTRskIsEB2qY1Fdiq2B%2B30GoWJD77ZBhtypR8DZq4%3D; pmaAuth-1=0d2yLcXXsvkEGADpFgmMaSMuj1qikxmmTzxkQAzVb7Hq3VghHC9q4IShCGI%2Bd6ZBHv7684bFgR5Qxrl%3D; PHPSESSID=2u89l5l43qr8172s6omq68gvdn; showhints=1 Connection: keep-alive username=test22&password=fefefe&login-php-submit-button=Login attempting to authenticate

L’attaquant dispose du cookie d’identification de la victime.

Travail à faire 2 : Nouvelle tentative en mode sécurisé et analyse du code source :

Test du niveau 1 de sécurité :

Q1. Est-ce que le niveau de sécurité 1 permet d'éviter l'attaque avec Burpsuite ?

Non car le niveau 1 ne soit pas suffisant pour empêcher une attaque XSS avec BurpSuite. En cas de non-conformité de la validation des entrées ou des protections XSS de base, BurpSuite peut toujours être employé pour injecter un script. Toutefois, il est possible qu'il existe un certain code d'émission qui empêche les attaques basiques.

Q2. Est-il possible d'écrire le code malicieux directement dans le formulaire ?

Oui, il est envisageable de générer du code malveillant dans un formulaire en utilisant le niveau de sécurité 1 si les données ne sont pas filtrées de manière adéquate. Par exemple, dans un formulaire vulnérable, un champ texte peut être ouvert à un script JavaScript sans être encodé ou validé.

Q3. En observant le code de la page dns-lookup.php, repérer les sécurités activées à ce niveau.

Avant de procéder au traitement, il est nécessaire de vérifier le format des données saisies par l'utilisateur.

Pour encoder les données avant de les afficher dans la page HTML, il est possible d'utiliser des fonctions telles que htmlspecialchars() ou htmlentities(), ce qui permet d'éviter l'exécution de scripts malveillants.

Le filtrage ou l'échappement des caractères spéciaux (tels que <, >, ", etc.) sont employés dans les attaques XSS.

Avant de procéder au traitement, il est nécessaire de vérifier le format des données saisies par l'utilisateur.

Pour encoder les données avant de les afficher dans la page HTML, il est possible d'utiliser des fonctions telles que htmlspecialchars() ou htmlentities(), ce qui permet d'éviter l'exécution de scripts malveillants.

Le filtrage ou l'échappement des caractères spéciaux (tels que <, >, ", etc.) sont employés dans les attaques XSS.

Q4. Quels sont les caractères typiques utilisés lors d'une attaque XSS ?

Les attaques **XSS** exploitent les entrées utilisateur pour injecter du code JavaScript malicieux dans une page web. Les caractères typiques utilisés dans ces attaques sont :

- `<`
- `>`
- `"` **et** `'`
- `(` **et** `)`
- `/`
- `script` : La balise `<script>` est la principale pour injecter du JavaScript.
- `javascript:` : Utilisé dans les attributs ou URL pour exécuter du code JavaScript.

Par exemple : `<script>alert('XSS')</script>`

Test du niveau 5 de sécurité :

Q5. Est-ce que le niveau de sécurité 5 permet d'éviter l'attaque avec BurpSuite ?

Effectivement, le niveau de sécurité 5 devrait prévenir les attaques XSS via BurpSuite en mettant en place un codage de sortie rigoureux, des vérifications d'entrée et éventuellement l'évasion des balises HTML et JavaScript.

Q6. En observant le fichier dns-lookup.php, repérer les variables spécifiques associées à ce niveau de protection.

Il est possible d'utiliser des fonctions telles que **filter_var()** afin de vérifier les entrées des utilisateurs.

Pour encoder les données avant leur affichage, il est recommandé d'utiliser **htmlspecialchars()** ou d'autres méthodes similaires.

La désinfection des variables : On peut utiliser la fonction strip_tags() ou des méthodes similaires afin de supprimer les balises HTML.

Il serait préférable que ces mécanismes évitent les injections malveillantes, y compris les attaques **XSS**.

Q7. Expliquer le rôle de l'instruction suivante dans le fichier `dns-lookup.php` (ligne n° 44) :

$html = htmlspecialchars($input, ENT_QUOTES, 'UTF-8');

Cette fonction est utilisée pour encoder le contenu de la variable $input en substituant certains caractères spéciaux par leurs entités HTML correspondantes. Cela bloque toute injection de code HTML ou JavaScript. À titre d'exemple, les < et > se transforment en < et >, ce qui fait que le contenu est affiché comme du texte brut plutôt que comme du code HTML exécutable.

Q8. Que vérifie la protection contre les injections de commandes ?

Les mesures de protection peuvent comprendre :

- **Nettoyage** des entrées afin d'éliminer les caractères potentiellement dangereux (tels que `;`, `|`, `&`).
- **Emploi de fonctions sécurisées** comme `escapeshellarg()` ou `escapeshellcmd()` pour s'assurer que les commandes transmises aux systèmes externes sont correctement échappées et ne peuvent pas être interprétées de manière malveillante.

Q9. Quelle fonction permet d'éviter spécifiquement les attaques de type XSS ?

Pour prévenir les attaques **XSS**, la fonction essentielle est :

- `htmlspecialchars()` : Cette fonction transforme les caractères spéciaux (tels que `<`, `>`, `&`) en leurs entités HTML respectives (comme `<`, `>`), ce qui empêche l'exécution de balises HTML ou de scripts JavaScript dans le navigateur.

Q10. Modifier le code source de la page `dns-lookup.php` afin d'isoler l'effet de cette protection.

Tu peux isoler l'effet de la protection en modifiant le code de `dns-lookup.php` pour activer ou désactiver cette fonction d'encodage par exemple :

Exemple avant protection :

echo $input; // Affiche directement l'entrée de l'utilisateur (potentiellement vulnérable).

Exemple après application de la protection XSS avec `htmlspecialchars()` :

echo htmlspecialchars($input, ENT_QUOTES, 'UTF-8'); // Protège contre XSS.

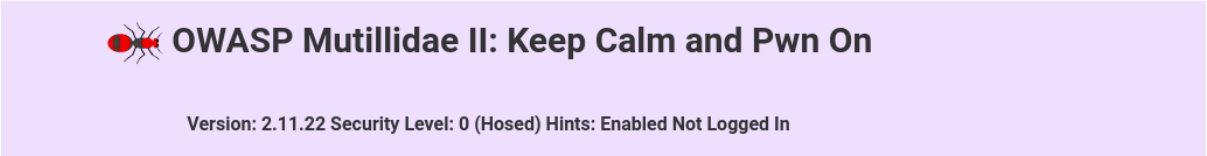
Q11. Résumer les protections mises en œuvre par le niveau de protection n°5.

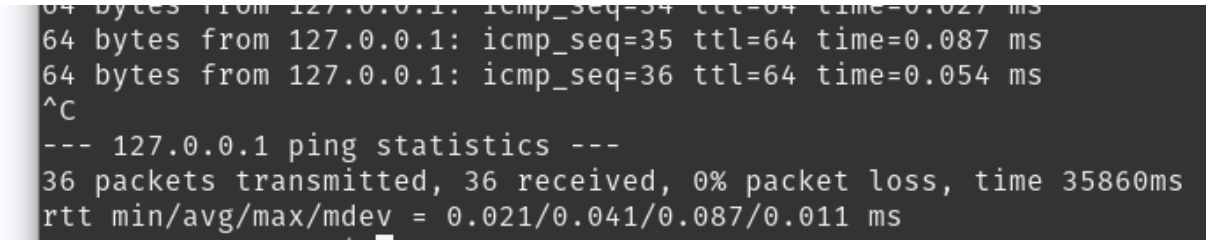
Le **niveau 5 de sécurité 5** de Mutillidae met en œuvre des mesures rigoureuses pour se prémunir contre les attaques, telles que :

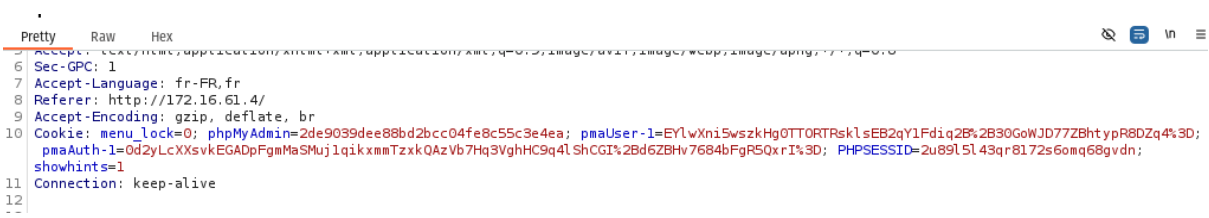
1. **Validation et filtrage des données** : Contrôle des saisies des utilisateurs afin d'éviter les injections néfastes.
2. **Encodage des sorties** : Application de `htmlspecialchars()` ou de fonctions similaires pour empêcher l'exécution de scripts XSS.
3. **Protection contre les injections SQL et les commandes système** : Adoption de requêtes préparées et de filtres pour contrer les injections.
4. **Filtrage des caractères spéciaux** : Échappement des caractères spéciaux afin d'éviter leur interprétation par le navigateur.

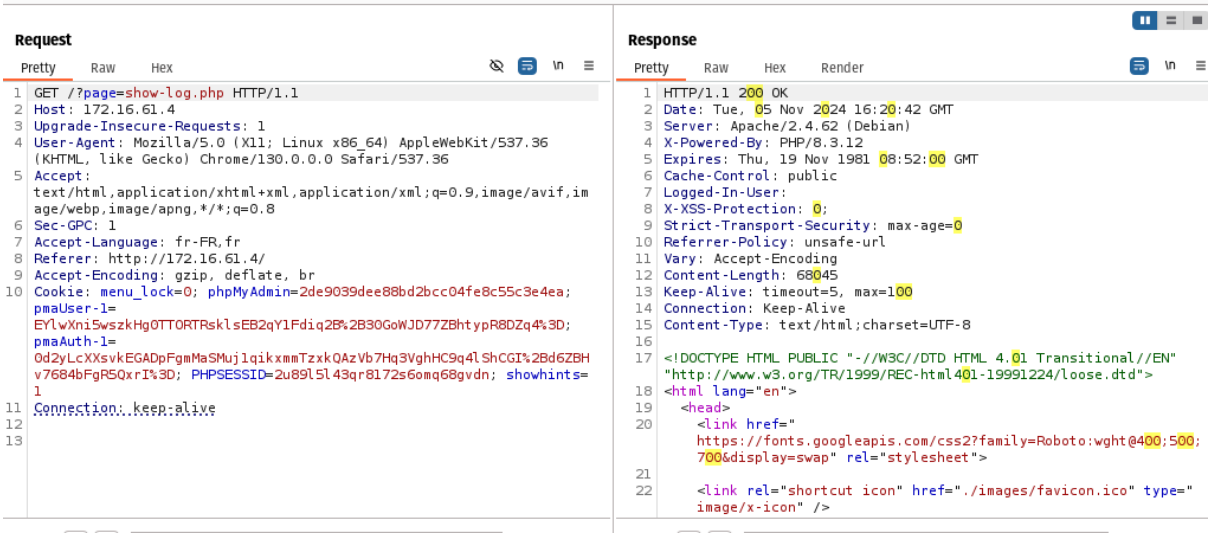
Travail à faire 3 :

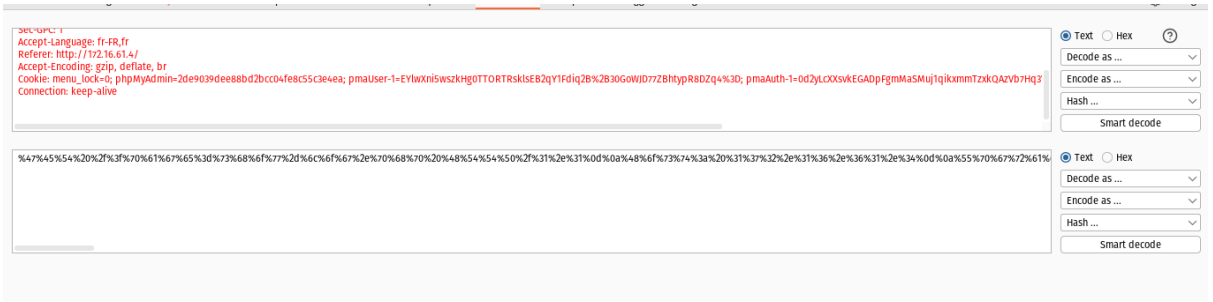
Q1. Commencer par préparer votre environnement de travail en démarrant le serveur Mutillidae ainsi que la machine cliente contenant le proxy BurpSuite. Vérifier que vos deux machines peuvent communiquer via un ping. Puis, positionner le niveau de sécurité de Mutillidae à 0.

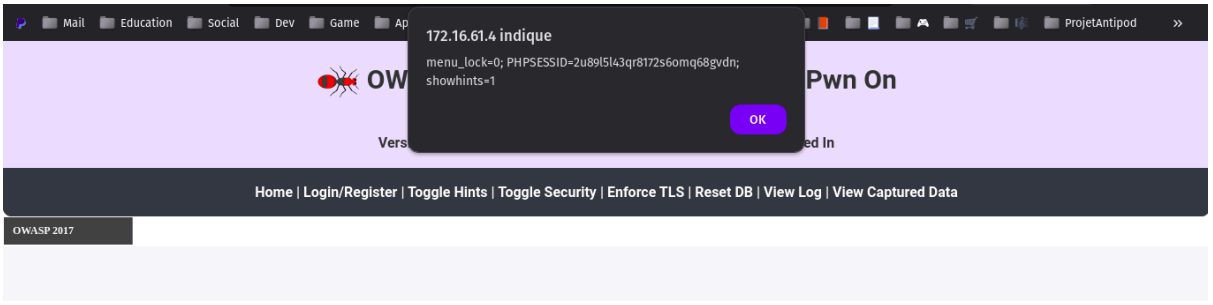


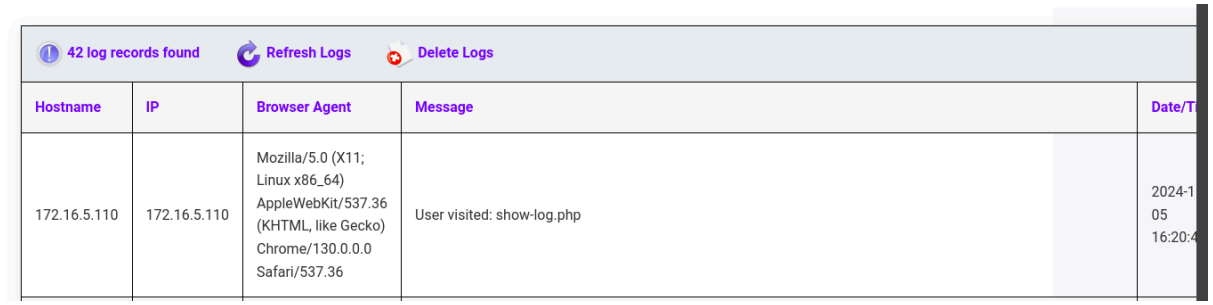




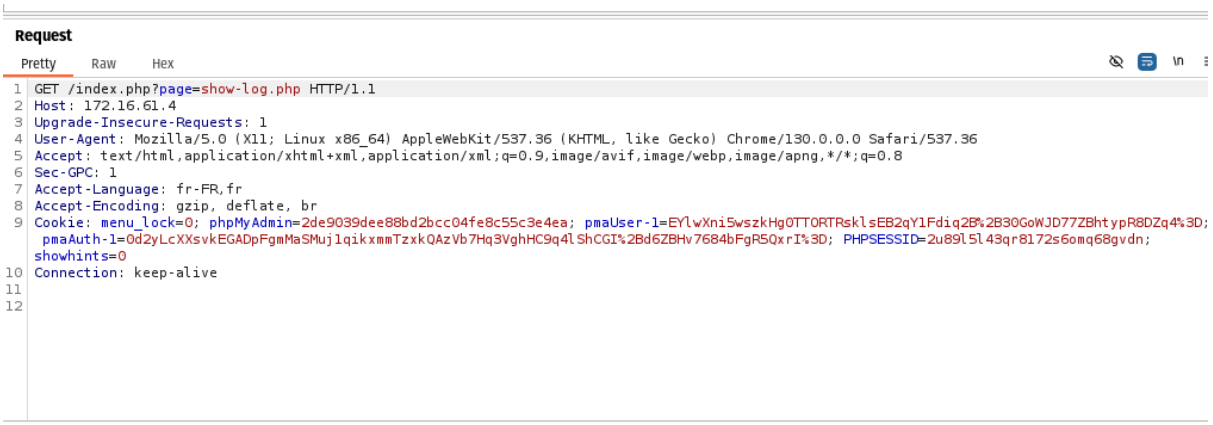








Travail à faire 4 :



Les Caractéristiques Typiques d'un Niveau 5

- **Authentification multifactorielle renforcée:** Au-delà du simple mot de passe, l'authentification peut nécessiter l'utilisation d'un token matériel, d'une application d'authentification, de la biométrie (empreintes digitales, reconnaissance faciale) ou d'une combinaison de ces éléments.
- **Chiffrement de bout en bout:** Toutes les données, qu'elles soient en transit ou au repos, sont cryptées avec des algorithmes de chiffrement robustes, rendant leur contenu illisible en cas d'interception.
- **Surveillance et analyse de sécurité en temps réel:** Des systèmes de surveillance sophistiqués détectent les anomalies et les tentatives d'intrusion en temps réel, permettant une réponse rapide aux menaces.
- **Mise à jour régulière des logiciels et des correctifs de sécurité:** Les vulnérabilités sont rapidement corrigées pour empêcher les attaquants d'exploiter les failles connues.
- **Contrôles d'accès stricts:** Les accès aux systèmes et aux données sont limités aux personnes autorisées, et les privilèges sont accordés sur une base de besoin en savoir.

Pourquoi l'encodage est-il important ?

- **Représentation des caractères:** Chaque caractère (lettre, chiffre, symbole) est représenté par un nombre binaire spécifique. L'encodage définit la correspondance entre ces caractères et ces nombres.
- **Universalité:** Il permet aux ordinateurs de différents pays et systèmes d'interpréter les mêmes textes de manière cohérente.
- **Problèmes d'encodage:** Les problèmes d'encodage sont une source fréquente de bugs, notamment lorsqu'on manipule des textes provenant de sources différentes.

Exemples d'encodages associés à différents langages :

- **Python:**
    - **UTF-8:** L'encodage le plus couramment utilisé, supportant une large gamme de caractères et flexible en termes de taille d'octet par caractère.
    - **Latin-1:** Encodage plus ancien, limité aux caractères de l'alphabet latin.

    - **ASCII:** Encodage de base pour les caractères ASCII, utilisé pour les textes en anglais.
- **Java:**
    - **UTF-8:** Comme en Python, UTF-8 est très répandu.
    - **Unicode:** Java utilise intrinsèquement l'Unicode, qui est un standard international pour représenter tous les caractères.
- **C++:**
    - **ASCII:** Souvent utilisé par défaut, mais les compilateurs C++ modernes supportent généralement UTF-8 et d'autres encodages.
- **JavaScript:**
    - **UTF-16:** L'encodage par défaut, bien que UTF-8 soit également supporté.
- **PHP:**
    - **UTF-8:** L'encodage recommandé pour les applications web modernes.
- **Ruby:**
    - **UTF-8:** L'encodage par défaut, mais d'autres encodages sont possibles.

Autres exemples d'encodages:

- **ISO-8859-1:** Encodage latin-1, couramment utilisé pour les langues européennes occidentales.
- **Shift-JIS:** Encodage japonais.
- **GB2312:** Encodage chinois simplifié.
- **EUC-KR:** Encodage coréen.

Facteurs influençant le choix de l'encodage:

- **Langues supportées:** L'encodage doit couvrir tous les caractères utilisés dans les textes.
- **Performance:** Certains encodages sont plus efficaces que d'autres en termes de mémoire et de vitesse de traitement.
- **Compatibilité:** L'encodage choisi doit être compatible avec les systèmes et les applications avec lesquels on interagit.

Principales Bonnes Pratiques

- **Validation et Filtrage des Entrées Utilisateur:**
    - **Tout ce qui provient de l'utilisateur doit être considéré comme potentiellement dangereux.**
    - **Validez la longueur, le format et le contenu des entrées.**
    - **Utilisez des listes blanches pour autoriser uniquement les caractères et les tags HTML sûrs.**
- **Encodage (Échappement) des Sorties:**
    - **Avant d'afficher des données provenant de l'utilisateur dans une page HTML, encodez-les pour les rendre inoffensives.**
    - **Utilisez les fonctions d'encodage appropriées selon le contexte (HTML, URL, attributs, etc.).**
- **Utilisation de Frameworks et Bibliothèques Sécurisées:**
    - **Les frameworks web modernes (e.g., React, Angular, Vue.js) incluent des mécanismes de protection contre le XSS.**
    - **Utilisez des bibliothèques spécialisées pour l'encodage et la validation des données.**
- **Mise en place d'une Politique de Sécurité de Contenu (CSP):**
    - **La CSP permet de restreindre les sources à partir desquelles le navigateur peut charger des scripts.**
    - **Configurez une CSP stricte pour limiter les risques d'exécution de code malveillant.**
- **Sécurisation des Cookies:**
    - **Utilisez le drapeau `HttpOnly` pour empêcher les scripts client d'accéder aux cookies.**
    - **Utilisez le drapeau `Secure` pour transmettre les cookies uniquement sur des connexions HTTPS.**
- **Mises à Jour Régulières:**
    - **Maintenez vos applications et les bibliothèques tierces à jour pour corriger les vulnérabilités connues.**
- **Tests de Pénétration Réguliers:**
    - **Simulez des attaques pour identifier les failles restantes.**

à Retenir

- **La prévention du XSS est une responsabilité de tous les développeurs web.**
- **Il n'existe pas de solution miracle.** Une combinaison de différentes mesures est nécessaire pour une protection efficace.

- **La sécurité est un processus continu.** Les menaces évoluent constamment, il est donc important de rester vigilant et de mettre à jour ses connaissances.

En conclusion, en adoptant ces bonnes pratiques et en restant informé des dernières menaces, vous pouvez considérablement réduire les risques d'attaques XSS et protéger vos applications web ainsi que les données de vos utilisateurs.