

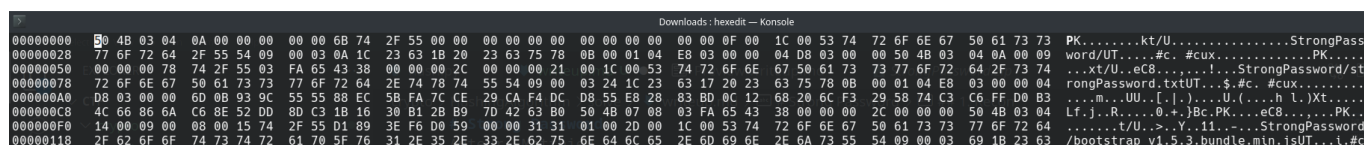
Strong Password

Auteur: bWlrYQ

1. Recon

Nous sommes face à une archive zip chiffrée, un mot de passe fort a été utilisé selon l'énoncé, il est donc inutile d'essayer de le casser à l'aide de méthodes classiques (bruteforce, dictionnaire)...

On essaie quand même de regarder un peu le fichier qui nous est fourni, c'est bien une archive chiffrée, mais on constate quelque-chose de "particulier", un fichier `bootstrap_v5.1.3.bundle.min.js` est présent en plus d'un `flag.txt`.



Le nom du fichier bootstrap a l'air générique, ce qui signifie qu'on doit sûrement pouvoir retrouver son contenu sur Internet et on commence à voir se dessiner une attaque très connue sur les fichiers zip, l'attaque par clair connu.

Cette attaque nous permet de déchiffrer n'importe quelle archive zip si on connaît le contenu d'un des fichiers présent.

2. Solve

La première étape va être de retrouver le clair, en une recherche google on tombe sur la doc de bootstrap [ici](#), puis [ici](#). On peut wget le fichier avec `wget`
<https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js>.

Maintenant qu'on a notre clair il va falloir déchiffrer le ZIP, soit on est excellent en C, en crypto et qu'on maîtrise par coeur l'implémentation de ZIP soit on fait comme moi et on utilise l'excellent `pkcrack` qui fait tout ça pour nous.

```
git clone https://github.com/keyunluo/pkcrack
mkdir pkcrack/build
cd pkcrack/build
cmake ..
make
```

On a tout ce qu'il nous faut pour déchiffrer l'archive, on va donc ziper notre fichier javascript avec `zip unencryptedStrongPassword.zip bootstrap.bundle.min.js`.

Puis avec `pkcrack`, on fait un joli tour de magie:

```

mika@owlryq ~/D/p/bin [master +5] ./rw -c encrypted.zip -p plaintext.txt -p plaintext.zip
$ ./pkcrack -C ../StrongPassword.zip -c StrongPassword/bootstrap_v1.5.3.bundle.min.js -P ../unencryptedStrongPassword.zip -p bootstrap.bundle.min.js -d decrypted.zip -a
Files read. Starting stage 1 on Sat Oct 29 12:12:47 2022
Generating 1st generation of possible key2_22991 values...done.
Found 4194304 possible key2-values.
Now we're trying to reduce these...
Lowest number: 961 values at offset 2047 (their code - the identities are so long they can't afford the disc space" >+)
Lowest number: 954 values at offset 1983
Lowest number: 919 values at offset 1973
Lowest number: 905 values at offset 1778
Lowest number: 868 values at offset 1776
Lowest number: 851 values at offset 1775
Done. Left with 851 possible Values. bestOffset is 1775. (and path) of the ZIP-archive containing the compressed plaintext (see 2. above)
Stage 1 completed. Starting stage 2 on Sat Oct 29 12:13:18 2022
Ta-daaaa! key0=6dd182fe, key1=3d97d840, key2=7bcc2170
Probabilistic test succeeded for 21221 bytes.
Stage 2 completed. Starting zipdecrypt on Sat Oct 29 12:13:43 2022
Decrypting StrongPassword/strongPassword.txt (d589594fa883208ef3a77874)... OK!
Decrypting StrongPassword/bootstrap_v1.5.3.bundle.min.js (4d5dcb8d36e5b7b6c8771574)... OK!
Decrypting StrongPassword/bootstrap_v1.5.3.min.css (f897e17816671ac6cb8e1074)... OK!
Finished on Sat Oct 29 12:13:43 2022
mika@owlryq ~/D/p/bin [master +5] ./rw
$ ls
decrypted.zip  extract*  findkey*  findkey.exe  makekey*  makekey.exe  pkcrack*  pkcrack.exe  zipdecrypt*  zipdecrypt.exe
mika@owlryq ~/D/p/bin [master +5] ./rw
$ unzip decrypted.zip
Archive:  decrypted.zip
  creating: StrongPassword/
    extracting: StrongPassword/strongPassword.txt
    inflating: StrongPassword/bootstrap_v1.5.3.bundle.min.js
    inflating: StrongPassword/bootstrap_v1.5.3.min.css
mika@owlryq ~/D/p/bin [master +5] ./rw
$ ls
StrongPassword/  decrypted.zip  extract*  findkey*  findkey.exe  makekey*  makekey.exe  pkcrack*  pkcrack.exe  zipdecrypt*  zipdecrypt.exe
mika@owlryq ~/D/p/bin [master +5] ./rw
$ cat StrongPassword/strongPassword.txt
FMCTF{Kn0wn_P141nT3xt_@ttacks_4reDAngerous!}
mika@owlryq ~/D/p/bin [master +5] ./rw

```

Et voilà : FMCTF{Kn0wn_P141nT3xt_@ttacks_4reDAngerous!}