Informacje

- **Kontakt:** katarzyna.mazur@umcs.pl
- **Konsultacje:** pokój 412 na 4 piętrze, przed konsultacjami proszę o wiadomość mailową
- **Zasady zaliczenia:**
- **Materiały, aktualności, zmiany terminów zajęć:** `https://kampus.umcs.pl/course/view.php?id=15080`

Zadania

2.1 Wykonaj szyfrowanie ciągu znaków `elJSK3h3aC8vTHZDSU4ydFpSdGNjMUZkZjhpckxEYm1yMm5uVW53OHQxND0K` za pomocą algorytmu AES-256-ECB z użyciem podanego klucza. Klucz znajduje się w pliku `ex2.1.key`. Odpowiedź (zaszyfrowany tekst) zakoduj kodowaniem Base64. Klucz użyty podczas szyfrowania powinien być podawany z linii komend. Prawidłowa odpowiedź do zadania to:
`O4nGnWb8DPrNGEUISBwKmOJ5b/0H58kBFmowcpfOjFaE7+KiQi9vW/su9xHD/w4HQdM4CjOvQ1vlGsjuaCqTRA==`.

2.2 Wykonaj deszyfrowanie pliku `enc2.2.txt` za pomocą algorytmu AES-256-ECB z użyciem podanego klucza: `01ec3b5108aada670ed14d03e926fe94`. Klucz powinien być podawany w linii komend. Wynikiem powinien być zrozumiały tekst.

2.3 Wykonaj deszyfrowanie pliku `enc2.3.txt` za pomocą algorytmu CAMELLIA-128-ECB z użyciem podanego hasła: `g4X6ZiYokdM=`. Hasło powinno być podawane z pliku.

2.4 Wykonaj deszyfrowanie pliku `enc2.4.txt` za pomocą algorytmu AES-256-CBC z użyciem podanego hasła: `WiWaTmfhq1FkEsU5ae+Ywoes8Ng8HIFstPZ38ktZmEI=`, wiedząc, że funkcja generowania klucza to PBKDF2.

2.5 Wykonaj deszyfrowanie pliku `enc2.5.txt` za pomocą algorytmu 3DES z użyciem podanego klucza: `b8a6c5dab155f2120629d0fc68f15ff9`, wiedząc, że funkcja generowania klucza to PBKDF2.

2.6 Wykonaj szyfrowanie pliku `ex.2.6.txt` za pomocą algorytmu BLOWFISH-ECB z użyciem klucza, który wygenerujesz za pomocą `OpenSSL rand`. Następnie wykonaj deszyfrowanie pliku, zapisując wynik deszyfrowania do pliku `dec2.6.txt`. Za pomocą polecenia `diff` lub `md5sum` sprawdź, czy pliki `ex.2.6.txt` oraz `dec2.6.txt` są identyczne.

2.7 Wykonaj deszyfrowanie pliku `enc.2.7.txt` za pomocą algorytmu AES-256-ECB z użyciem podanego klucza: `0311fab728530ebe`, algorytmu generowania klucza PBKDF1 oraz wskazanej ilości iteracji algorytmu równej 356.

2.8 Wykonaj deszyfrowanie pliku `enc.2.8.txt` za pomocą algorytmu AES-256-CBC z użyciem podanego hasła: `0311fab728530ebe`, algorytmu generowania klucza PBKDF2 oraz wskazanej ilości iteracji algorytmu równej 41331.

2.9 Ze strony kursu pobierz plik `secured.zip`. Plik ten jest zabezpieczony hasłem. Jest to jedno z najczęściej używanych przez użytkowników haseł. Za pomocą programu JohnTheRipper spróbuj złamać hasło, którym zaszyfrowany jest plik. Możesz skorzystać z listy najpopularniejszych haseł dostępnej na githubie: `https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10k-most-common.txt`.

2.10 Ze strony kursu pobierz plik `secret.zip`. Plik ten jest zabezpieczony hasłem. Wiedząc, że plik ten jest zabezpieczony hasłem o długości pomiędzy 5-6 znaków, i zawiera jedynie cyfry, za pomocą programu JohnTheRipper spróbuj złamać hasło, którym zaszyfrowany jest plik. Wygeneruj listę możliwych haseł za pomocą programu `crunch`.

2.11 Wykonaj zadanie **2.9** za pomocą narzędzia `fcrackzip`.

2.12 Zidentyfikuj, jaki algorytm szyfrujący został wykorzystany do zaszyfrowania tekstu: `Z8CerTOLe1JlDKWfvDeifw==` przy pomocy klucza `a35febba42490abe`.

2.13 W pliku `enc2.13.txt` znajduje się zaszyforwany za pomocą klucza `7bb464cff6b5c03335b4a248cef76582` obrazek w formacie `*.png`. Odszyfruj obrazek. Rozwiązaniem zadania powinien być plik `*.png`.

2.14 Ze strony kursu pobierz plik `secdir.zip`. Plik ten jest zabezpieczony hasłem. Spróbuj złamać hasło za pomocą narzędzia `hashcat`.

Linki

- `https://www.openssl.org/docs/man1.0.2/man1/openssl-enc.html`
- `https://wiki.openssl.org/index.php/Enc`

Odpowiedzi

2.1 Polecenia (robią to samo):

```
openssl enc -aes-256-ecb -in ex2.1.txt -K 241e0a0b6afe4503b359a15e34c0d8b7 | base64
openssl enc -aes-256-ecb -in ex2.1.txt -a -K 241e0a0b6afe4503b359a15e34c0d8b7
```

Wynik:
`O4nGnWb8DPrNGEUISBwKmOJ5b/0H58kBFmowcpfOjFaE7+KiQi9vW/su9xHD/w4HQdM4CjOvQ1vlGsjuaCqTRA==`

2.2 Plik został utworzony za pomocą polecenia:
`openssl enc -aes-256-ecb -in ex2.2.txt -out enc2.2.txt -a -K 01ec3b5108aada670ed14d03e926fe94`

Deszyfrowanie:
`openssl enc -d -aes-256-ecb -in enc2.2.txt -out dec2.2.txt -a -K 01ec3b5108aada670ed14d03e926fe94`

Odkodowanie Base64:
`cat dec2.2.txt | base64 -d`

2.3 Plik przygotowany został za pomocą polecenia:
`openssl enc -CAMELLIA-128-ECB -in ex2.3.txt -out enc2.3.txt -a -kfile ex2.3.key`

Polecenia (rozwiązanie, zamienne):
```
openssl enc -d -CAMELLIA-128-ECB -in enc2.3.txt -out dec2.3.txt -a -kfile ex2.3.key
openssl enc -d -CAMELLIA-128-ECB -in enc2.3.txt -out dec2.3.txt -a -pass file:ex2.3.key
```

2.4 Plik przygotowany został za pomocą polecenia:
`openssl enc -aes-256-cbc -in ex2.4.txt -out enc2.4.txt -a -kfile ex2.4.pass -pbkdf2`

Polecenie (rozwiązanie):
`openssl enc -d -aes-256-cbc -in enc2.4.txt -out dec2.4.txt -a -kfile ex2.4.pass -pbkdf2`

2.5 Plik przygotowany został za pomocą polecenia (`-des-ede3-ecb` to `-des-ede3`):
```
openssl enc -des-ede3 -a -in ex2.5.txt -out enc.2.5.txt -pbkdf2
-K b8a6c5dab155f2120629d0fc68f15ff9
```

```
openssl enc -des-ede3-ecb -a -in ex2.5.txt -out enc.2.5.txt -pbkdf2
-K b8a6c5dab155f2120629d0fc68f15ff9
```

Polecenie (rozwiązanie):

```
openssl enc -d -des-ede3 -a -in enc2.5.txt -out dec2.5.txt -pbkdf2
-K b8a6c5dab155f2120629d0fc68f15ff9
openssl enc -d -des-ede3-ecb -a -in enc2.5.txt -out dec2.5.txt -pbkdf2
-K b8a6c5dab155f2120629d0fc68f15ff9
```

2.6 Plik przygotowany został za pomocą polecenia:

```
openssl rand -hex 16
openssl rand -base64 32
openssl enc -a -BF-ECB -in ex2.6.txt -out enc2.6.txt -K b8a6c5dab155f2120629d0fc68f15ff9
```

Polecenie (rozwiązanie):

```
openssl enc -d -a -BF-ECB -in enc2.6.txt -out dec2.6.txt -K b8a6c5dab155f2120629d0fc68f15ff9
diff ex2.6.txt dec2.6.txt
```

2.7 Plik przygotowany został za pomocą polecenia:

```
openssl enc -aes-256-ecb -a -in ex2.7.txt -out enc2.7.txt -K 0311fab728530ebe -iter 356
```

Polecenie (rozwiązanie):

```
openssl enc -d -aes-256-ecb -a -in enc2.7.txt -out dec2.7.txt -K 0311fab728530ebe -iter 356
```

2.8 Plik przygotowany został za pomocą polecenia:

```
openssl enc -a -AES-256-CBC -in ex2.8.txt -out enc2.8.txt
-kfile ex2.8.pass -pbkdf2 -iter 41331
```

Polecenie (rozwiązanie):

```
openssl enc -d -a -AES-256-CBC -in enc2.8.txt -out dec2.8.txt
-kfile ex2.8.pass -pbkdf2 -iter 41331
```

2.9 Konwersja zipa do formatu dla Johna:

```
zip2john secured.zip > secured.hash
```

Złamanie hasha:

```
john secured.hash --wordlist=10k-most-common.txt
john secured.hash --show
```

2.10 Generowanie haseł:

```
crunch 5 6 0123456789 -o wordlist.txt
```

2.11 Polecenie:

```
fcrackzip -v -u -D -p 10k-most-common.txt secured.zip
```

2.12 Polecenia pomocnicze:

```
openssl list -cipher-commands > ciphers.txt
xargs -n1 < ciphers.txt > cipherscommands.txt
```

Polecenia:

```
openssl list -cipher-commands | xargs -n1 > ciphers.txt
```

Skrypt:

```
#!/bin/bash
fname=$1
while read cipher; do
    echo "-----------------"
    echo $cipher
    openssl enc -d -$cipher -in enc2.12.txt -K a35febba42490abe -a
    echo "-----------------"
done < $fname
```

2.13 Plik przygotowany został za pomocą polecenia:

```
cat 61542ca02c1e9.png | base64 > ex2.13.txt
openssl enc -seed-ecb -a -in ex2.13.txt -out enc2.13.txt -K 7bb464cff6b5c03335b4a248cef76582
```

Polecenia:

```
openssl enc -d -seed-ecb -a -in enc2.13.txt -out dec2.13.txt -K 7bb464cff6b5c03335b4a248cef76582
cat dec2.13.txt | base64 -d > tux2.13.png
```

2.14 Wydobycie hasha z zipa:

```
zip2john secdir.zip | cut -d ':' -f 2 > secdir.hash
```

Spr typu hasha i znalezienie odpowiedniego mode:

```
hashcat -a 0 -m 13600 secdir.hash /usr/share/wordlists/rockyou.txt
hashcat -a 0 -m 13600 secdir.hash /usr/share/wordlists/rockyou.txt --show
```

Hasło:

```
$zip2$*0*1*0*0c049efd2d817dcf*b075*8*4311b5cefc19be03*66bc702d92e1fdac3d69*$/zip2$:soccer
```