

Treća laboratorijska vježba

Na trećoj laboratorijskoj vježbi iz Sigurnosti računala i podataka simulirali smo brute force napad na podatke enkriptirane AES 256 bitnim ključem, a pritom smo morali i dobiti hash value stringa sastavljenog od našeg imena i prezimena.

Nije nam bilo poznato koje podatke dekriptiramo, samo smo znali da je to slika png formata.

Koristili smo se library-ima Fernet i cryptography.

Prvi korak bio je ući u python virtual container u vs code editoru, gdje smo izradili novu mapu.

Kako bismo pronašli naš personalizirani challenge, morali smo otkriti hash value od našeg stringa "prezime_ime" ("cvitanic_vjeran").

Prvi dio koda koji smo napisali odnosio se na hashiranje našeg imena (prezime_ime) te smo rezultat ručno uspoređivali s naslovima challengeova.

```
from cryptography.hazmat.primitives import hashes
from os import path

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

if __name__ == "__main__":
    filename = hash("cvitanic_vjeran") + ".encrypted"
    print(filename)

    if not path.exists(filename):
```

```
with open(filename, "wb") as file:
    file.write(b"") # prazan string
```

Kad smo pronašli naš challenge, izradili bi novi file s tim imenom u istoj mapi te bismo copy-paste-ali sadržaj ciphertexta(challengea) u taj file.

Sada čitamo iz filea vrijednost ciphertexta te ga kao argument šaljemo u funkciju brute_force_attack.

Metodom brute_force_attack ćemo dekriptirati ciphertext.

U while petlji, pomoću Fernet funkcije, neprestano stvaramo nove ključeve(čije vrijednosti rastu od 1 do 2^{22} (to je broj mogućih ključeva jer zadnjih 22 bita ključa poprimaju bilo koje vrijednosti, dok je prvih $256 - 22 = 234$ bitova jednako 0)).

Zatim pomoću ključa dekriptiramo ciphertext.

```
from cryptography.hazmat.primitives import hashes
from os import path
from cryptography.fernet import Fernet

def brute_force_attack(ciphertext):
    # print(ciphertext)

    ctr = 0

    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)

        f = Fernet(key)

        plaintext = f.decrypt(ciphertext)

        ctr += 1

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()
```

```

digest = hashes.Hash(hashes.SHA256())
digest.update(input)
hash = digest.finalize()

return hash.hex()

if __name__ == "__main__":
    filename = hash("cvitanic_vjeran") + ".encrypted"
    print(filename)

    if not path.exists(filename):
        with open(filename, "wb") as file:
            file.write(b"") # prazan string

    with open(filename, "rb") as file:
        encrpyted_challenge = file.read()

    brute_force_attack(encrpyted_challenge)

```

Sad je potrebno javiti računalu kako da prepozna ispravan plaintext među dobivenim plaintextovima.

Znamo da je enkriptiran png file, a znamo i da on ima svoje meta podatke pomoću kojih ga možemo prepoznati(signature). Svaki png file ima istih prvih 8 byte-ova → `b"\211PNG\r\n\032\n"`.

Pomoću ugrađene funkcije `startswith()` provjerimo počinje li dobiveni plaintext s `b"\211PNG\r\n\032\n"`, odnosno je li on zapisan u png formatu.

```

def test_png(header):
    if header.startswith(b"\211PNG\r\n\032\n"):
        return True
    return False

```

Sada ćemo narediti računalu da, kada pronađe odgovarajući plaintext, ispiše u konzolu key te da u BINGO.png file ispiše plaintext.

Donjim kodom želimo omogućiti korisniku praćenje brute force napada tako da u konzoli ispisujemo counter vrijednosti(pomoću kojih generiramo ključ), kako bi korisnik mogao pratiti koliko se ključeva do tog trenutka isprobalo(u prosjeku će se isprobati 1/2 svih mogućih ključeva, tj. u ovom slučaju 2^{21}).

Također, radi preglednosti želimo ispisati vrijednost nakon svakog tisućitog ključa.

```
if not ctr % 1000:
    print(f"[*] Keys tested: {ctr:},", end="\r")
    ctr += 1
```

```
from pydoc import plain
from cryptography.hazmat.primitives import hashes
from os import path
from cryptography.fernet import Fernet
import base64

def test_png(header):
    if header.startswith(b"\211PNG\r\n\032\n"):
        return True
    return False

def brute_force_attack(ciphertext):
    # print(ciphertext)

    ctr = 0

    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)

        f = Fernet(key)

        plaintext = f.decrypt(ciphertext)

        # ima li poruka smisla provjera
        header = plaintext[:32] # uzme prva 32 bajta
        if test_png(header):
            print(f"KEY FOUND: {key}")
            with open("BINGO.png", "wb") as file:
                file.write(plaintext)
            break

    if not ctr % 1000: # 0 -> counter je djeljiv s 1000 ; !0 -> counter nije djeljiv s 1000
```

```

        print(f"[*] Keys tested: {ctr:,}", end="\r")
        ctr += 1

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

if __name__ == "__main__":
    filename = hash("cvitanic_vjeran") + ".encrypted"
    print(filename)

    if not path.exists(filename):
        with open(filename, "wb") as file:
            file.write(b"") # prazan string

    with open(filename, "rb") as file:
        encrpyted_challenge = file.read()

    brute_force_attack(encrpyted_challenge)

```

Library-ji koje koristimo ne mogu dešifrirati ciphertext krivim ključem pa šalju Exception → rješenje je korištenje try except -a.

```

from pydoc import plain
from cryptography.hazmat.primitives import hashes
from os import path
from cryptography.fernet import Fernet, InvalidToken
import base64

def test_png(header):
    if header.startswith(b"\211PNG\r\n\032\n"):
        return True
    return False

def brute_force_attack(ciphertext):

```

```

# print(ciphertext)

ctr = 0

while True:
    key_bytes = ctr.to_bytes(32, "big")
    key = base64.urlsafe_b64encode(key_bytes)

    f = Fernet(key)

    try:
        plaintext = f.decrypt(ciphertext)

        # ima li poruka smisla provjera
        header = plaintext[:32] # uzme prva 32 bajta
        if test_png(header):
            print(f"KEY FOUND: {key}")
            with open("BINGO.png", "wb") as file:
                file.write(plaintext)
            break
    except InvalidToken:
        pass

    if not ctr % 1000: # 0 -> counter je djeljiv s 1000 ; !0 -> counter nije djeljiv s 1000
        print(f"[*] Keys tested: {ctr:},", end="\r")
        ctr += 1

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

if __name__ == "__main__":
    filename = hash("cvitanic_vjeran") + ".encrypted"
    print(filename)

    if not path.exists(filename):
        with open(filename, "wb") as file:
            file.write(b"") # prazan string

    with open(filename, "rb") as file:
        encrpyted_challenge = file.read()


    brute_force_attack(encrpyted_challenge)

```

Pokrenemo kod u odgovarajućem direktoriju u kojem se file nalazi.

- Python brute_force.py

Dobiveni plaintext:



Congratulations Cvitanic Vjeran!
You made it!