Izvještaj za 2. laboratorijske vježbe

Za ovaj zadatak koristili smo se programskim jezikom Python i njegovom bibliotekom crypthography.

Pokrećemo virtualno okruženje u Pythonu kako ne bismo smetali globalnom okruženju naredbom python -m venv grupa3srp (kasnije virtualno okruženje samo izbrišemo)

Virtualno okruženje pokrećemo naredbama:

cd mirtakardum

cd grupa3srp

cd Scripts

activate

cd.. 2x

Instaliramo biblioteku crypthography koja je prisutna samo u ovom virtualnom okruženju - pip install cryptography

Importamo sustav za simetričnu enkripciju Fernet - ** `from** **cryptography.fernet** **import** Fernet`

Primjer:

key = Fernet.generate_key() - generiranje ključa i spremanje u varijablu key

f = Fernet(key)

plaintext = b"Hello world" - binarni format, podatak koji želimo enkriptirati

ciphertext = f.encrypt(plaintext) - algoritam za enkriptiranje plaintexta

f.decrypt(ciphertext) - algoritam za dekriptiranje ciphertexta

b'Hello world' - ispis

Challenge:

Za otkrivanje osobnog file-a sa local servera a507 koristimo kod:

```
from cryptography.hazmat.primitives import hashes

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()

filename = hash('prezime_ime') + ".encrypted"

if __name__ == "__main__":
    h = hash('kardum_mirta')
    print(h)
```

python brute_force.py - izbacuje ime odgovarajućeg osobnog file-a na a507 serveru

```
 key = int.from_bytes(os.urandom(32), "big") & int('1'*KEY_ENTROPY, 2)
 key_base64 = base64.urlsafe_b64encode(key.to_bytes(32, "big"))
 fernet = Fernet(key_base64)

def brute_force():
  ctr = 0
  while True:
    key_bytes = ctr.to_bytes(32, "big")
    key = base64.urlsafe_b64encode(key_bytes)
    if not (ctr+1) % 1000:
      print(f"[*] Keys tested: {ctr+1:,}", end="\r")
    ctr += 1
```

Pseudokod izazova:

```
C = E_K(P)
for k=1,...,infinity
P = D_k(C)
if test_png(P)
print("KEY FOUND:", k)
break
```

Budući da znamo da je rješenje png formata, početak svakog dekriptiranog plaintexta testiramo tako da vidimo radi li se o png headeru.

```
from cryptography.fernet import Fernet
import base64
from cryptography.hazmat.primitives import hashes

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()

filename = hash('prezime_ime') + ".encrypted"

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

def brute_force():
    # Reading from a file
    filename = "ime_osobnog_filea.encrypted"
    with open(filename, "rb") as file:
        ciphertext = file.read()

    ctr = 0
    while True:
            key_bytes = ctr.to_bytes(32, "big")
            key = base64.urlsafe_b64encode(key_bytes)
            if not (ctr+1) % 1000:
                print(f"[*] Keys tested: {ctr+1:,}", end="\r")

            try:
                plaintext = Fernet(key).decrypt(ciphertext)
                header = plaintext[:32]
                if test_png(header):
                    print(f"[+] KEY FOUND: {key}")
                    # Writing to a file
                    with open("BINGO.png", "wb") as file:
                        file.write(plaintext)
                    break
            except Exception:
                pass
            ctr += 1

if __name__ == "__main__":
    brute_force()
```