SRP Vježba 2

Cilj vježbe: koristeći *brute force* metodu doći do enkripcijskog ključa u svrhu pribavljanja *plaintext-a,* u našem slučaju PNG datoteke.

Realizacija: svaki student dobio je vlastiti izazov, ali imena datoteka su enkriptirana. Kako bi pristupio izazovu student mora otkriti svoju datoteku. Ime datoteke je enkriptirano *SHA256* hash kriptografskom funkcijom.

```
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"
```

Kao što vidimo poznat nam je način na koji su imena datoteka enkriptirana. Ako uzmemo u obzir da za dani ulazi hash funkcija uvijek daje isti ali jedinstven izlaz, sve što trebamo napraviti je uvrstiti naše ime i prezime u gornju funkciju, čime dobivamo naše ime u enkriptiranom obliku.

Sada kada znamo koja je naša datoteka, možemo krenuti s "probijanjem" enkripcije same datoteke.

Kod za *brute force*:

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

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

def brute_force():
    filename = "ime_moje_datoteke.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}")
                with open("BINGO.png", "wb") as file:
                    file.write(plaintext)
                break
    except Exception:
            pass
    ctr += 1

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

Objašnjenje: ukratko kod učitava datoteku, ulazi u petlju i *Fernet* rješenju za enkripciju šalje ključeve jedan po jedan, svaki put traži "magic" vrijednost u zaglavlju navodno dekriptirane datoteke da vidi jesmo li ustvari našli traženi *plaintext-a* ili se radi o promašaju. Ako smo našli ključ, slika se sprema u datoteku.

Nakon par minuta dobili smo željeni ključ. Za što je bilo potrebno stotine tisuća iteracija petlje. Ključ naravno neću otkriti iz očitih razloga.

Još jedna stvar koju nisam rekao je da se za enkripciju koristi 22 bitna entropija. Samim time broj mogućih kombinacija bio je 2^22. Iako je to na našem hardveru trajalo par minuta, na specijaliziranim računalima to bi trajalao mnogo kraće. Jedna od solucija bi bila i paralelizacija. Tako da cijelu domenu ključeva podjelimo na više dijelova svaki na svoju jezgru. Što je veća entropija ključa to će više vremena trebati da se pogodi prava vrijednost.