Lab 3

Message authentication and integrity

U trećoj laboratorijskoj vježbi cilj nam je bio primjeniti teoretske spoznaje o osnovnim kriptografskih mehanizmima za autentikaciju i zaštitu integriteta poruka u praktičnim primjerima.

U pythonu smo otvorili virtualno okruženje, zatim ušli u Scripts i aktivirali ga

`python -m venv srp`

`cd env`

`cd Scripts`

`activate`

Izazov 1

Implementirati zaštitu integriteta sadržaja dane poruke primjenom odgovarajućeg message authentication code (MAC) algoritma. Koristili smo HMAC mehanizam iz Python biblioteka cryptography.

Napravili smo tekstualnu datoteku "message.txt" u kojoj je sadržana poruka čiji integritet želimo zaštititi

Tada smo u Visual Studio Code pomoću `code .` provjerili čita li ispravno našu poruku iz filea.

```
with open("message.txt", "rb") as file:
  content = file.read()

print(content)
```

Slijedeći korak je bio ubaciti funkciju za izračun MAC vrijednosti za danu poruku te nadodali smo ključ i generirali MAC (pomoću ključa i poruke) pa ga ispisali u heksadekadskom obliku.

```
from cryptography.hazmat.primitives import hashes, hmac

def generate_MAC(key, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
```

```
    h.update(message)
    signature = h.finalize()
    return signature

if __name__ == "__main__":
  key = b"cant guess"
  with open("message.txt", "rb") as file:
    content = file.read()
  mac = generate_MAC (key, content)
  print(mac.hex())
```

Taj MAC smo onda spremili u file "message.mac"

```
with open("message.mac", "wb") as file:
    file.write(mac)
```

Sljedeće smo ubacili funkciju za provjeru validnosti MAC-a za danu poruku pa je naš konačni kod izgledao ovako:

```
from cryptography.hazmat.primitives import hashes, hmac
from cryptography.exceptions import InvalidSignature

def generate_MAC(key, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
    h.update(message)
    signature = h.finalize()
    return signature

def verify_MAC(key, signature, message):
    if not isinstance(message, bytes):
        message = message.encode()

    h = hmac.HMAC(key, hashes.SHA256())
    h.update(message)
    try:
        h.verify(signature)
    except InvalidSignature:
        return False
    else:
        return True

if __name__ == "__main__":
  key = b"cant guess"

  with open("message.txt", "rb") as file:
    content = file.read()

  with open("message.mac", "rb") as file:
    signature = file.read()
```

```
  is_authentic = verify_MAC (key, signature, content)
  print(is_authentic)
```

Sada nam funkcija verify_MAC provjerava da li je MAC koji mu mi prosljedimo iz message.mac jednak kao i onaj koji on generira. Ukoliko su jednaki znači da poruka nije promjenjena i funkcija vraća "True", a ukoliko je izmjenjena ili message.txt ili message.mac onda će nam vratiti "False".

npr. ako u message.txt izmjenimo "Prodaj" u "Kupi" izbacit će nam "False".

Izazov 2

U sljedećem izazovu trebali smo utvrditi vremenski ispravnu sekvencu transakcija sa odgovarajućim dionicama. Na lokalnom serveru u fileu vuko_matej/mac_challenge imali smo parove .txt datoteka i .mac datoteka kojima je trebalo ispitati autentičnost.

Prvo smo trebali preuzeti program wget i pohraniti ga u direktorij gdje se nalazila skripta iz prošlog zadatka.

Onda smo izazove preuzeli lokalno pomoću naredbe:

```
wget.exe -r -nH -np --reject "index.html*" http://a507-server.local/challenges/vuko_matej/mac_challenge
```

Koristili smo iste funkcije kao u prošlom zadatku samo smo izmjenili kod u mainu:

```
if __name__ == "__main__":
  key = "vuko_matej".encode()

  for ctr in range(1,11)
    msg_filename = f"./challenges/vuko_matej/mac_challenge/order_{ctr}.txt"
    sig_filename = f"./challenges/vuko_matej/mac_challenge/order_{ctr}.sig"

    print(msg_filename)
    with open(msg_filename, "rb") as file:
      content = file.read()

    print(sig_filename)
    with open(sig_filename, "rb") as file:
      mac = file.read()

    is_authentic = verify_MAC(key, signature, content)
    print(is_authentic)
```

Za ključ smo koristili naše prezime i ime enkodirano.

U brojaču smo otvarali svaki par .txt i .sig fileova te provjeravali njihovu autentičnost.