SVEUČILIŠTE U RIJECI
FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA

Preddiplomski studij informatike

Projektni zadatak iz kolegija
SIGURNOST INFORMACIJSKIH I KOMUNIKACIJSKIH SUSTAVA

Aplikacija za siguran razgovor u programskom jeziku Python

Autori: Bojan Radulović, Matija Prpić, Roko Peruško

Mentori: prof. dr. sc. Božidar Kovačić
Milan Petrović, mag. inf.

U Rijeci, lipanj 2022.

Sadržaj

Uvod

Tema projektonog zadatka i ovog seminara izrada je aplikacije za sigurno sljanje i primanje poruka u programskom jeziku Python.

Kako bi se ostvarila sigurna komunikacija, koriste se protokoli za autentifikaciju, razmjenu ključeva, šifriranje poruka te autentifikaciju poruka.

Za autentifakicaju servera i klijenta koristi se ECDSA protokol, za razmjenu ključeva koristi se x25519, za šifriranje poruka koristi se Fernet, a za njihovu autentifikaciju koristi se Poly1305 protokol.

Program je izrađen za operacijski sustav Linux (Ubuntu) te je prilagođen za pokretanje i uredan ispis u terminalu.

Kod programa podijeljen je u dva dijela:

- Serverska strana – datoteka server.py
- Klijentska strana – datoteka client.py

Korišteni Python moduli

Za realizaciju projekta korišteno je nekoliko Python modula:

- socket – modul korišten za slanje podataka između servera i klijenta
- threading – modul korišten za paralelno izvođenje dijelova programskog koda
- sys i os – moduli korišteni za zaustavljanje i izlazak iz procesa/programa
- ecdsa – modul korišten za implementaciju ECDSA autentifikacije. Prije pokretanja programa potrebno ga je preuzeti naredbom "pip install ecdsa" u terminalu
- base64 – modul korišten za kodiranje koristeći base64
- pyca/cryptography – modul korišten za sve ostale kriptografske potrebe programa (Fernet šifriranje, x25519 razmjena ključeva, Poly1305 autentifikacija poruka…)

```
import socket
import threading
import sys
import os
from ecdsa import SigningKey, VerifyingKey, BadSignatureError

from cryptography.hazmat.primitives import hashes, poly1305, serialization
from cryptography.hazmat.primitives.asymmetric.x25519 import X25519PrivateKey, X25519PublicKey
from cryptography.hazmat.primitives.kdf.hkdf import HKDF

from cryptography.fernet import Fernet
import base64

from cryptography.hazmat.backends import default_backend
```

Slika 1, Korišteni moduli

Opis funkcija programa

recv_msg() - Funkcija koja neprekidno čeka pridolazeće poruke. Nakon primanja poruke, ona se najprije dešifrira pomoću Fernet-a. Zatim se iz poruke odvaja sam sadržaj poruke od autentifikacijskog koda koji joj je dodao pošiljatelj. Pomoću tog autentifikacijskog koda i Poly1305 protokola, poruka se zatim autentificira. Nakon uspješne autentifikacije poruke, njen sadržaj se prikazuje u terminalu. U slučaju da je sadržaj pristigle poruka "exit", prekida se rad programa. Funkcija nema ulazne argumente i ne vraća nikakve vrijednosti.

```
def recv_msg():
    while True:
        #cekanje poruke
        recved_msg = conn.recv(1024)
        if not recved_msg:
            sys.exit(0)
        try:
            #fernet dekripcija
            recved_msg = f.decrypt(recved_msg)
        except:
            print("Fernet invalid token!")
            sys.exit(0)
        #razdvajanje poruke od autentifikacijskog kod
        msg = recved_msg[:-16]
        recved_hash = recved_msg[-16:]

        #autentifikacija poruke
        p = poly1305.Poly1305(derived_key)
        p.update(msg)
        try:
            p.verify(recved_hash)
            #ispis poruke
            print("Client: " + msg.decode())
            #gasenje programa ako je poruka exit
            if(msg.decode() == "exit"):
                s.shutdown(socket.SHUT_RDWR)
                s.close()
                os._exit(0)
        except:
            print("Message authentication failed!")
```

Slika 2, Kod funkcije recv_msg()

send_msg() – Ova funkcija najprije čeka unos korisnika koji će se koristi kao sadržaj porruke koja će biti poslana drugoj strani. Nakon unosa sadržaja porukea, najprije se za tu poruku generira autentifikacijski kod pomoću Poly1305 protokola. Taj autentifikacijski kod se zatim dodaje na kraj poruke. Sadržaj poruke se zatim šifrira pomoću Fernet-a i šalje drugoj strani. Zatim se sadržaj poruke prikazuje u terminalu. U slučaju da je sadržaj poruka "exit", prekida se rad programa. U slučaju da sadržaj poruke nije "exit", funkcija se ponavlja. Funkcija nema ulazne argumente i ne vraća nikakve vrijednosti.

```
def send_msg():
    while True:
        #unos poruke
        sent_msg = input()
        #ovaj kod ukljanja liniju s unosom iz terminala radi vece preglednosti
        print ("\033[A                                \033[A")
        #dodavanje autentifikacijskog koda poruci
        p = poly1305.Poly1305(derived_key)
        p.update(sent_msg.encode())
        sent_hash = p.finalize()
        #slanje poruke i autentifikacijskog koda
        conn.send(f.encrypt(sent_msg.encode() + sent_hash))
        #ispis poruke
        print("Server: " + sent_msg)
        #gasenje programa ako je poruka exit
        if(sent_msg == "exit"):
            s.shutdown(socket.SHUT_RDWR)
            s.close()
            os._exit(0)
```

Slika 3, Kod funkcije send_msg()

generate_keys() – Funckija koja generira ECDSA i x25519 ključeve te ih pohranjuje u vanjske .pem datoteke. Funkcija nema ulazne argumente i ne vraća nikakve vrijednosti.

```
def generate_keys():
    #generiranje i pohrana ecdsa kljuceva
    sk = SigningKey.generate()
    vk = sk.verifying_key
    with open("server_private.pem", "wb") as f:
        f.write(sk.to_pem())
    with open("server_public.pem", "wb") as f:
        f.write(vk.to_pem())

    #generiranje i pohrana x25519 privatnog kljuca
    server_private_key2 = X25519PrivateKey.generate()
    with open("server_private_x25519.pem", "wb") as f:
        f.write(server_private_key2.private_bytes(
            encoding=serialization.Encoding.PEM,
            format=serialization.PrivateFormat.PKCS8,
            encryption_algorithm=serialization.NoEncryption()))
```

Slika 4, Kod funkcije generate_keys()

start_talk() – Funkcija koja pokreće paralelno izvršavanje funkcija recv_msg() i send_msg(). Funkcija nema ulazne argumente i ne vraća nikakve vrijednosti.

```
def start_talk():
    #pokretanje procesa primanja i slanja poruka
    t = threading.Thread(target=recv_msg)
    t.start()
    send_msg()
```

Slika 5, Kod funkcije start_talk()

Opis rada programa

Za uspješno izvršavanje programa, potrebno ga je pokrenuti unutar teminala u operacijskom sustavu Linux (Ubuntu). Prvo je potrebno pokrenuti serversku stranu (server.py) pa zatim klijentsku (client.py).

Prvi korak rada programa generiranje je ključeva i njihovo spremanje u vanjske .pem datoteke pozivom funkcije generate_key() na serverskoj i klijentskoj strani. Ako su ključevi već generirani, ovu funkciju nije potrebno pozivati. Zatim se ti isti ključevi učitavaju iz nastalih .pem i pretvaraju u bajtove (radi slanja javnih ključeva putem socket-a).

Server zatim stvara novi socket na zadanoj adresi i vratima (localhost:8080) te čeka povezivanje klijenta.

Nakon što se klijent uspješno poveže, server mu šalje svoj javni ECDSA verifikacijski ključ i poruku potpisanu njime. Kada klijent primi ključ i poruku, pomoću njih vrši autentifikaciju servera. Nakon uspješne autentifikacije servera, na analogni način i server autentificira klijenta.

U sljedećem koraku rada programa, klijent i server razmijenjuju svoje javne x25519 ključeve. Pomoću svog privatnog x25519 ključa i javnog x25519 ključa druge strane, i server i klijent stvaraju zajedničke ključeve koji su jednaki jedan drugome. Radi dodatne sigurnosti, i server i klijent vrše dderivaciju svog zajedničkog ključa. Ovime je završen proces razmjene ključeva x25519 protokolom.

Nakon toga, i server i klijent inicijaliziraju Fernet šifriranje koristeći zajednički ključ.

Sljedeći korak je početak razmjene poruka unutar aplikacije za razgovor. Obje strane pozivaju funkciju start_talk() koja na prethodno opisan način pokreće procese primanja i slanja poruka.

Zaključak

Programski jezik python sadrži veliki broj odličnih modula za razvoj aplikacija za sigurnu razmijenu poruka.

Koristeći ECDSA protokol za autentifikaciju klijenta i servera, x25519 protokola za razmjenu ključeva, Fernet šifriranja i autentifikaciju poruka pomoću Poly1305, moguće je u programskom jeziku Python razviti aplikaciju za sigurnu komunikaciju.

Izvori

1) socket — Low-level networking interface: https://docs.python.org/3/library/socket.html

2) ecdsa 0.17.0: https://pypi.org/project/ecdsa/

3) X25519 key exchange: https://cryptography.io/en/latest/hazmat/primitives/asymmetric/x25519/

4) Fernet (symmetric encryption): https://cryptography.io/en/latest/fernet/

5) Poly1305: https://pycryptodome.readthedocs.io/en/latest/src/hash/poly1305.html

6) sys — System-specific parameters and functions: https://docs.python.org/3/library/sys.html

7) os — Miscellaneous operating system interfaces: https://docs.python.org/3/library/os.html

8) Vedran Miletić, Rad s Python modulom pyca/cryptography: https://gaseri.org/hr/nastava/izvedbeni/2021-2022/SIKS/