Sigurnost Tjedni Izvještaji

Prvi tjedan:

Cilj-Ubacit evil-station u komunikacijski kanal između station-1 i station-2

WSL (za linux)

mkdir bdimic - Napravio direktorij bdimic

cd bdimic - uški u direktorij

git clone "https://github.com/mcagalj/SRP-2021-22"

cd SRP-2021-22/arp-spoofing/ - ušli u ovaj direktorij (sadrži start.sh i stop.sh)

cd arp-spoofing/ - pristupili tom direktoriju

Otvaramo containere:

docker exec -it station-1 bash

docker exec -it station-2 bash

docker exec -it evil-station bash

Zapravo što radimo jest da naš evil-station imitira station-2 te tako informacije prolaze od station-1 preko evil-stationa do station-2 i tako prisluškujemo razmjenu informacija dvaju stationa

ping- simple utility used to check whether a network is available and if a host is reachable. With this command, you can test if a server is up and running.

tcpdump- dump traffic

arpspoof- intercept packets on a switched LAN

netcat- command-line utility that reads and writes data across network connections, using the TCP or UDP protocols.

Drugi tjedan:

Za enkripciju smo koristili ključeve 22 bita entropije

Koristili smo fernet sustavTrebali smo brute forsati ključ pomoću kojeg smo dobili poruku

Koristili smo fernet sustav:

Fernet koristi sljedeće *low-level* kriptografske mehanizme:

- AES šifru sa 128 bitnim ključem
- CBC enkripcijski način rada
- HMAC sa 256 bitnim ključem za zaštitu integriteta poruka
- Timestamp za osiguravanje svježine (*freshness*) poruka

Ključevi su generirani na sljedeći način:

```
 # Encryption keys are 256 bits long and have the following format:
#
```

```
#             0...000b[1]b[2]...b[22]
#
# where b[i] is a randomly generated bit.
key = int.from_bytes(os.urandom(32), "big") & int('1'*KEY_ENTROPY, 2)

# Initialize Fernet with the given encryption key;
# Fernet expects base64 urlsafe encoded key.
key_base64 = base64.urlsafe_b64encode(key.to_bytes(32, "big"))
fernet = Fernet(key_base64)
```

Učitavanje i spremanje datoteka u Pythonu:

```
# Reading from a file
with open(filename, "rb") as file:
    ciphertext = file.read()
    # Now do something with the ciphertext
```

```
# Writing to a file
with open(filename, "wb") as file:
    file.write("Hello world!")
```

Treći tjedan:

Cilj vježbe je provjeriti integritet poruka

-Pri tome ćemo koristiti simetrične i asimetrične krito mehanizme: *message authentication code (MAC)* i *digitalne potpise* zasnovane na javnim ključevima

Koristimo HMAC za zaštitu integriteta

1. U lokalnom direktoriju kreirajte tekstualnu datoteku odgovarajućeg sadržaja čiji integritet želite zaštititi.
2. Učitavanje sadržaja datoteke u memoriju.

   ```
   # Reading from a file
   with open(filename, "rb") as file:
       content = file.read()
   ```

3. Funkcija za izračun MAC vrijednosti za danu poruku.

   ```
   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
```

4. Funkcija za provjeru validnosti MAC-a za danu poruku.

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


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
```

-Za preuzeti sve izazove sa servera:

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

Otvaranje datoteka:

```
or ctr in range(1, 11):
    msg_filename = f"order_{ctr}.txt"
    sig_filename = f"order_{ctr}.sig"
    print(msg_filename)
    print(sig_filename)

    is_authentic = ...

    print(f'Message {message.decode():>45} {"OK" if is_authentic else "NOK":<6}')
```