Izvještaji iz laboratorijskih vježbi

1. Laboratorijska vježba

12.10.20211.

Man-in-the-middle attacks (ARP spoofing)

Realizirati man in the middle napad iskorištavanjem ranjivosti ARP protokola. Student će testirati napad u virtualiziranoj Docker mreži (Docker container networking) koju čine 3 virtualizirana Docker računala (eng. container): dvije žrtve `station-1` i `station-2` te napadač `evil-station`.

U ovoj vježbi koristili smo **Windows** terminal te u istoj smo otvorili **Ubuntu** terminal na wsl sustavu.

Alati koje smo koristili:

Kloniranje repozitorija

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

Promjena direktorija

```
$ cd SRP-2021-22/arp-spoofing/
```

Pokretanje virtualiziranog mrežnog scenarija(Docker)

```
$ chmod +X ./start.sh
$ ./start.sh
```

Zaustavljanje virtualiziranog mrežnog scenarija

```
$ chmod +X ./stop.sh
$ ./stop.sh
```

Ispis s dockera

```
$ docker ps
```

```
uname
```

```
hostname
```

Ispisivanje mrežne konfiguracije

```
$ ifconfig -a
```

Ulaz u station

```
$ docker exec -it station-1 bash
```

Provjera komunikacije između dva stationa

```
$ ping station-2
```

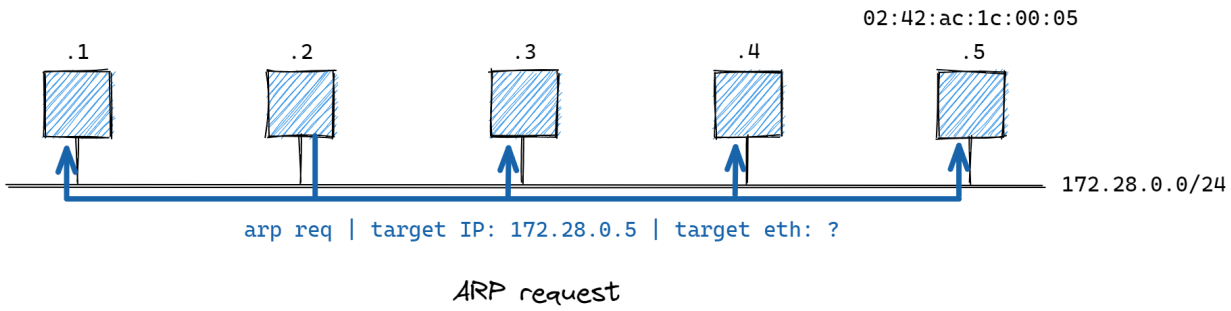
Otvaranje servera TCP na portu 9000 pomoću netcat-a na kontejneru `station-1`

```
$ netcat -lp 9000
```

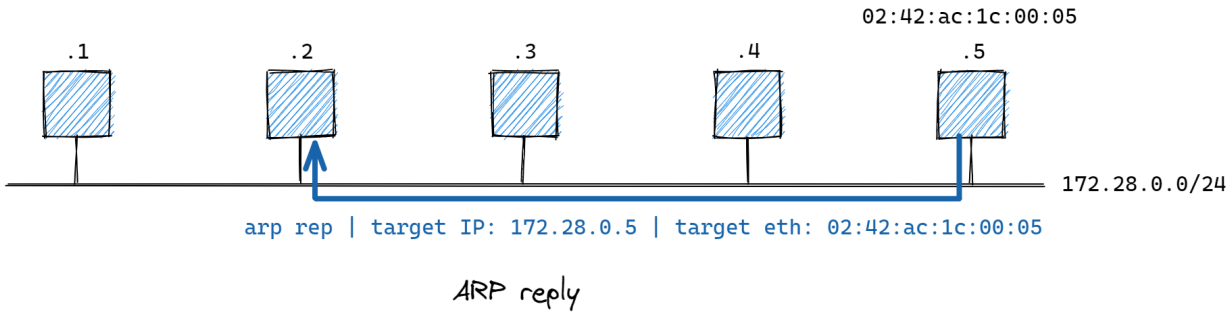
Otvaranje client TCP-a na hostname.u station-1 9000 pomoću netcat-a na kontejneru `station-2`

```
$ netcat station-1 9000
```

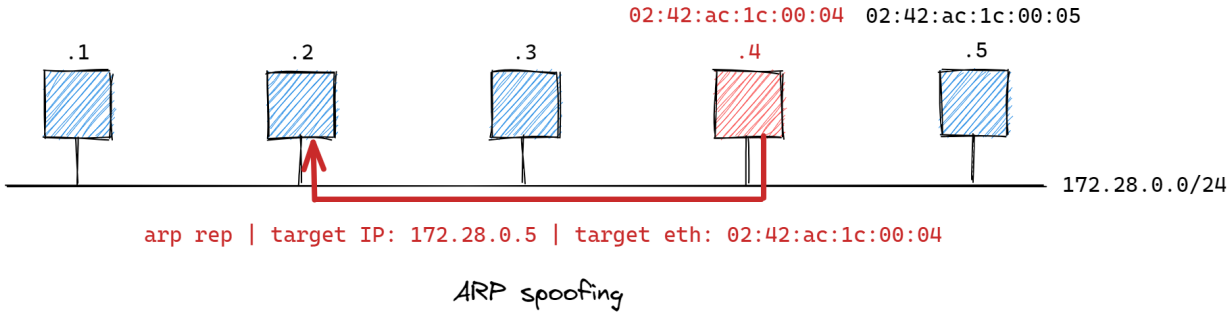
Arpspoof



ARP request



ARP reply



ARP spoofing

Pokretanje napada u evil-station-u

```
$ arpspoof -t station-1 station-2
```

Pokretanje tcpdump-a u drugom evil-station-u (docker-u) i praćenje prometa

```
$ tcpdump
```

Onemogućavanje slanja poruka iz station-1 u station-2

```
$ echo 0 < /proc/sys/net/ipv4/ip_forward
```

Još alata:

Čišćenje screen-a

//ctrl+L

Ispis direktrorija

```
$ ls -
```

Brži ispis komande

//tab

Prekid programa

//ctrl+C

Razdjela ekrana

//shift+alt

HUB - omogućuje povezivanje više računala u jednu mrežu

SWITCH - povezuje različite uređaje zajedno na jednoj računalnoj mreži

2. Laboratorijska vježba

26.10.2021.

Symmetric key cryptography - a crypto challenge

Riješiti odgovarajući crypto izazov, odnosno dešifrirati odgovarajući *ciphertext* u kontekstu simetrične kriptografije. Izazov počiva na činjenici da student nema pristup enkripcijskom ključu.

Za pripremu crypto izazova, odnosno enkripciju korištena je Python biblioteka `cryptography`. *Plaintext* koji student treba otkriti enkriptiran je korištenjem high-level sustava za simetričnu enkripciju iz navedene biblioteke - `Fernet`.

Pripremljeni su nam personalizirani izazovi na internom serveru.

Pokretanje virtual python okruženja

```
$ python -m venv srp
```

```
$ activate
```

$ pip install cryptography

$ from cryptography.fernet import Fernet

Funkcija koja generira ključ koji dekriptira podatke

$ key = Fernet.generate_key

$ python

$ f = Fernet(key)

Primjer korištenja varijabli u pythonu

$ plaintext = b"hello world"

$ print("helo world")

//b označava byte-e u kojima treba biti enkriptirana riječ

Enkriptiranje poruke

$ ciphertext = F.encrypt(b"hello world")

Dekriptiranje poruke

$ F.decrypt(ciphertext)

Izlazak iz pythona

$ exit()

Odlazak u VS → kodiranje

$ code brute_force.py

U VS Brute-force attack code

#datoteke koje importamo kako bismo ih mogli koristiti

import base64

from cryptography.fernet import Fernet

from cryptography.hazmat.primitives import hashes

#hash f-ja za generiranje naziva datoteke koju trebamo dekriptirati (Secure Hash Algortihm 256)

```
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()
#f-ja koja provjerava je li ono što je enkriptirano slika png formata

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

#f-ja u kojoj izvršavamo napad

def brute_force():
    filename = "ime_datoteke.encrypted"
    #Reading from a file
    with open(filename, "rb") as file:
        ciphertext = file.read()
    #brojač ključeva
    ctr = 0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)
        #ispis svakog provjerenog 1000tog ključa
```

```
        if not(ctr + 1) % 1000:
            printf(f"[*] Keys tested: {ctr + 1:,}", end = "\r")
        #izlazak iz infinite loop-a
        try:
            plaintext = Fernet(key).decrypt(ciphertext)
            header = plaintext[:32]
            #je li slika?
            if test_png(header):
                printf(g"[+] KEY FOUND: {key}")
                #Writing to file ono što smo dekriptirali
                with open("lol.png", "wb") as file:
                    file.write(plaintext)
                #izlazimo iz while-a jer smo pronašli ključ
                break
            except Exception:
                pass
            ctr += 1


    if __name__=="__main__":
        brute_force()
        #h = hash('mamic_rosana')
        #print(h)
```

Dekriptirana slika

Congratulations Mamic Rosana!

You made it!