HaEpUz gyak2-3: Linux hálózat + szoftver szerszámok és Hálózati funkciók - Gyakorló feladatok

A gyakorló feladatokat (és később majd a kvízt is) a BME Cloud-ban (Smallville) futtatott VM-eken kell megoldani, abban a környezetben, amit a 3. gyakorlaton használtunk. Mindenkinek három saját VM-et kell indítani (gateway: “GW xfce4 2021” template-ből indítva; client és internet: “CLIENT 2021” template-ből indítva). Dolgozni az xfce4 grafikus desktop környezettel “ellátott” GW gépen érdemes, amihez rdp kliens segítségével tudtok kapcsolódni, a gyakorlaton látott módon. Innen külön terminálokból indított ssh kapcsolatokkal tudtok átlépni a másik két gépre.

Az utolsó két feladatot (7-8.) a HaEpUz VM-ben kell megoldani, mert a mininet hálózatemulációs program kell hozzá (az jelenleg még nem elérhető a cloud-os VM-eken).

0. Feladat

Készítsük elő a 3. gyakorlaton látott alap munkakörnyezetet! A hálózati konfigurációknál mindenki használja a neki kiosztott ID-t és címeket, ami a Teams/gyak3 channel/Files alatt érhető el (fájl: HaEpUz-2021-gyak3-IDs.xlsx)! Az alábbi konfigurációs lépésekre lesz szükség:

- eth1 interfészek manuális konfigurálása (GW: 10.”ID”.0.1/24, CLIENT: 10.”ID”.0.2/24)
- CLIENT gépen az eth0 interfész lekapcsolása
- GW gépen forwarding engedélyezése, címfordítás beállítása
- CLIENT gép konfigurálása, hogy a GW-en keresztül kapcsolódjon a külvilághoz

1. Feladat

Installáljuk a `traceroute` programot, ha nem elérhető a VM-en!
(`sudo apt install traceroute`)
Vizsgáljuk meg, hogy a GW VM milyen útvonalon (routereken keresztül) éri el a www.stanford.edu gépet és analizáljuk azt is, hogy ennek hogy történik a felderítése a `traceroute` program segítségével! Ehhez a következő lépéseket hajtsuk végre:

- Találjuk ki a www.stanford.edu névhez tartozó IPv4 címet!
- A `wireshark` program segítségével vizsgáljuk a beazonosított IPv4 címre kimenő és onnan bejövő forgalmakat (a releváns interfészen)!
- A `traceroute` program default beállításával indítsuk a vizsgálatot a www.stanford.edu gépre!

Kérdések:

- Milyen típusú forgalommal történik a tesztelés és milyen lépésekben? Milyen protokoll, IP cím, port szám, TTL érték, stb. van beállítva, hány csomaggal történik a vizsgálat?
- Honnan milyen válaszok érkeznek?
- Ezek alapján hogyan azonosítható az útvonal?
- Hány hop-ból áll az útvonal? Milyen IPv4 című routerekből áll az útvonal?

Alakítsuk át úgy a vizsgálatot, hogy ICMP Echo Request csomagokkal történjen a tesztelés és ne kelljen kivárni a felesleges mérések ereményét! Vizsgáljuk a forgalmat `wireshark` segítségével.

2. Feladat

Keressük meg (készítsünk bash scriptet) a GW VM-ről a legkevesebb hop számmal elérhető root DNS szervert!

Keressük meg (készítsünk bash scriptet) a GW VM-ről a legkisebb átlagos körülfordulási idővel elérhető root DNS szervert!

(root DNS szerver pl.: k.root-servers.net)

3. Feladat

Vizsgáljuk meg, hogy hány hallgató dolgozik a BME cloudban a feladaton. Ehhez egy shell script segítségével számoljuk meg a 10.”ID”.0.1 címen elérhető, futó GW VM-eket! (ID: 50-től 206-ig vizsgálandó)

(Segítség: pl. egy for ciklusban kell ping parancsokat hívogatni.)

4. Feladat

Indítsunk a GW VM-en a netcat (`nc`) program segítségével egy tcp és egy udp szervert (külön terminálban), mindkettő a 80-as porton figyeljen!
Kérdések:

- Mit látunk az IPv4-re vonatkozó socket statisztikákat vizsgálva (`netstat` vagy `ss` programok használhatók)? (Fókuszáljunk az nc programra!)
- Milyen process ID-t használ a tcp és az udp szerver?
- Milyen állapotú a két socket?

Kapcsolódjunk a CLIENT VM-ről mindkét szerverhez (külön terminálból) és teszteljük a kapcsolatokat!
Kérdések:

- A socket statisztikákat vizsgálva hány bejegyzésünk van az `nc` programra vonatkozóan?
- Milyen állapotúak a socketek?

Kapcsolódjunk a GW VM-ről a lokálisan futó szerverhez és teszteljük a kapcsolatot!
Kérdések:

- A socket statisztikákat vizsgálva hány bejegyzésünk van az `nc` programra vonatkozóan?
- Ezek melyik interfészekre vonatkoznak és milyen állapotúak a socketek?

5. Feladat

Állítsuk be az órán és gyakorlaton látott DNAT konfigurációt a GW gépen! (A GW 8080-as portja legyen forwardolva a CLIENT 80-as portjára.) Ismételjük meg az előző feladatot úgy, hogy a CLIENT és GW VM-eken indítunk egy-egy `nc` szervert, amik a 80-as tcp porton figyelnek! A kapcsolatok tesztelését az INTERNET VM-ről végezzük, közben vizsgáljuk a socket statisztikákat mindhárom gépen! A GW VM-en vizsgáljuk az `iptables` nat tábláját is!
Kérdések:

- A socket statisztikákat vizsgálva hány bejegyzésünk van az `nc` programra vonatkozóan a három gépen az INTERNET VM-ről való csatlakozás előtt?
- Milyen bejegyzések és milyen státuszú socketek vannak a csatlakozások után?
- Hány illeszkedés volt közben a nat tábla vonatkozó bejegyzésére?

6. Feladat

Töltsük be a `wireshark`ba az alábbi pcap fájlt, ami egy IP alapú telefonbeszélgetés logja:
http://ib213.tmit.bme.hu:8888/haepuz/RTP-Protocol.pcap

Kérdések:

- Az első SIP csomag alapján mi a hívott fél SIP címe?
- Az első RTP csomag alapján milyen codec-kel van a beszélgetés kódolva (payload type)?
- A wikipédia alapján, aminek a hitelességében ne bízzunk vakon, milyen átviteli sebességet ér el ez a codec?
- a `tcpdump` program kimenetét feldolgozva számoljuk meg összesen hány byte-nyi RTP csomagot küldött a hívott fél a hívónak!

7. Feladat

Az utolsó két feladatot (7-8.) a HaEpUz VM-ben kell megoldani, mert a mininet hálózatemulációs program kell hozzá.

Az alábbi `topo.py` python scriptet hozzuk létre a home könyvtárunkban (/home/vagrant/topo.py):

```
import random
from mininet.topo import Topo

class PingTopo( Topo ):
      def build( self ):
      nb_host = random.randint(10,20)
      addrs = range(2,255)
      random.shuffle(addrs)
      addrs = [1] + addrs[:nb_host]

      switch = self.addSwitch( 's0' )
      for i, addr in enumerate( addrs, start=1 ):
            h = self.addHost( 'h%d' % i, ip='10.10.10.%d/24' % addr)
            self.addLink( h, switch)

topos = { 'ping': ( lambda: PingTopo() ) }
```

Ezután indítsunk egy mininet hálózatot a fájlban leírt topológia szerint:

```
sudo mn --custom ~/topo.py --topo=ping
```

Nézzük meg a h1-es hoszt ip-címét és alhálózatát, majd írjunk egy bash scriptet, amit lefuttatva a h1-es hoszton kiírja azoknak a hosztoknak az IP-címet, amik h1-gyel közös alhálózatban vannak.

(Segítség: pl. egy for ciklusban kell ping parancsokat hívogatni.)

8. Feladat

Az utolsó két feladatot (7-8.) a HaEpUz VM-ben kell megoldani, mert a mininet hálózatemulációs program kell hozzá.

Keressük meg a home könyvtárunk alatt az összes `m` nevű fájt. Az egyik ezek közül arra használható, hogy a hoszt gépről egy futó mininetes hálózatemuláció során egy emulált gépen futtassunk le egy parancsot. Ha argumentum nélkül indítjuk el, egy tömör helpet ad.

Indítsuk el a mininetet az alábbi paranccsal:
```
sudo mn --custom ~/topo.py --topo=ping
```

Írjunk egy bash szkriptet az (`m` parancs felhasználásával), amit ha a hoszt képen futtatunk, akkor megcseréli a h2 és a h3 gépek ip-címeit és kiírja a h1-es gép mac címét.