Técnicas de Perceção de Redes
Network Awareness

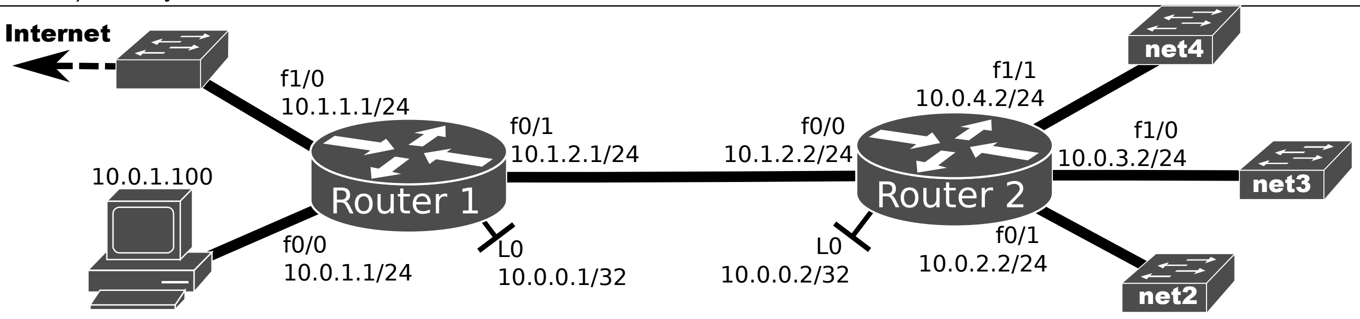
Data Acquisition

Objectives

- Nodes, links and terminals data acquisition with SNMP
- Traffic flows data acquisition with Netflow/IPFIX
- Raw traffic acquisition with libpcap
- Equipment/Servers Logs acquisition via SSH

Data acquisition with SNMP

1. Configure a network (in GNS3) according to the following figure. The PC can be a VM or the host PC, with Linux (Debian) with Python, SNMP tools, network MIBs and CISCO MIBs.



MIB references: SNMP Object Navigator, http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en
MIBS: IF-MIB, IP-MIB, and CISCO-QUEUE-MIB

Python references: *Snimpy* – API reference, https://snimpy.readthedocs.org/en/latest/api.html
argparse - Parser for command-line options, https://docs.python.org/3/library/argparse.html
matplotlib.pyplot - http://matplotlib.org/api/pyplot_api.html

2. In both routers, configure a SNMP version 3 community (using the name "private") with Read-Only permissions, and access with authentication (MD5, password authpass) and encryption (AES128, password: privpass), for user uDDR from group gDDR:

```
Router(config)# snmp-server user uDDR gDDR v3 auth md5 authpass priv aes 128 privpass
Router(config)# snmp-server group gDDR v3 priv
Router(config)# snmp-server community private RO
```

3. Download and test the baseSNMP.py script, and understand how different MIB objects can be accessed.

```
python baseSNMP.py -r 10.0.0.2
```

4. Use the following MIB objects to access relevant interface traffic statistics:
- *ifHCOutUcastPkts*, *ifHCInUcastPkts*, *ifHCOutOctets*, *ifHCInOctets* from IF-MIB.

5. Create a time loop, with periodicity given by argument, and retrieve interface statistics. Display byte and packet increments (in both directions).

6. Periodically save data in text or JSON files.
Note: Close all files and present final report upon script termination (e.g., catch CTRL+C):

```
  try:


  .
..
  except KeyboardInterrupt:
  …
```

7 (Extra, requires X). Plot interface usage (one per interface). You may use python package *matplotlib.pyplot*, *see: `ion(), plot(), draw(), show()`.*

Data acquisition with NetFlow/IPFIX

NetFlow references: NetFlow Overview

NetFlow Export Datagram Format (version 1 and 5 formats)

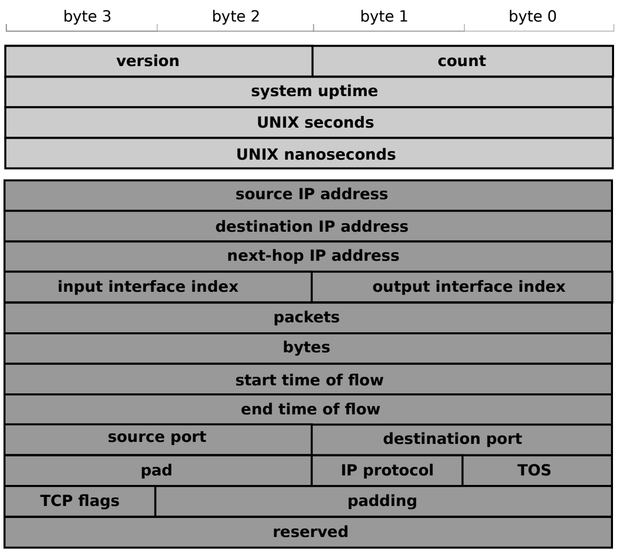
Python references: *socket* - Low-level networking interface, https://docs.python.org/2/library/socket.html

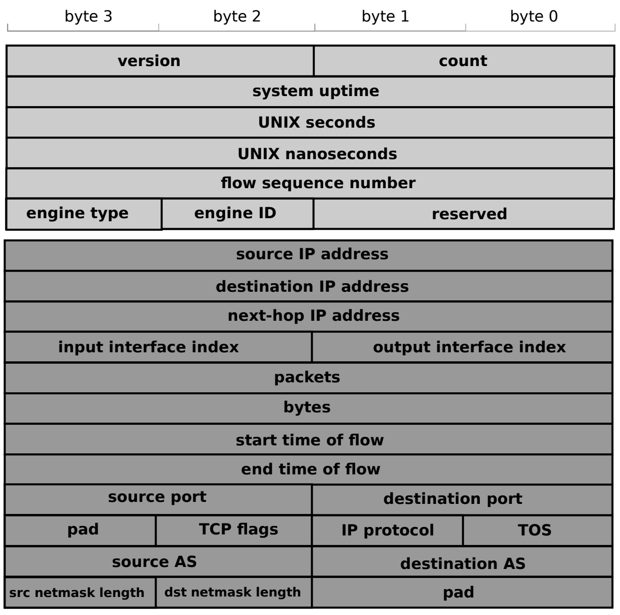
struct - Interpret strings as packed binary data, https://docs.python.org/2/library/struct.html

netaddr IPAddress and IPNetwork - https://netaddr.readthedocs.org/en/latest/tutorial_01.html

netaddr IPSet - https://netaddr.readthedocs.org/en/latest/tutorial_03.html

NetFlow v1 and v5 header and body formats





9. Configure Router2 to export (to PC VM) the flow statistics using NetFlow version 1 for all traffic egressing interface f0/1.

```
Router2(config)# interface FastEthernet0/1
Router2(config-if)# ip flow egress
Router2(config)# ip flow-export destination 10.0.1.100 9996
Router2(config)# ip flow-export source Loopback 0
Router2(config)# ip flow-export version 1
```

10. Download and test the baseNetFlow.py script, generating traffic to and from terminals (VPCS) in networks net2, net3, and net4. Understand how the NetFlow packet is received and how data fields can be accessed.

```
python baseNetFlow.py -r 10.0.0.2 -n 10.0.2.0/24 10.0.3.0/24 10.0.4.0/24
```

11. Complete the code to retrieve the relevant NetFlow version 1 data and construct the traffic matrix. Configure the required additional NetFlow export commands in Router1 and Router2.
Note: consider using Python library *netaddr* classes IPAddress, IPNetwork, and IPSet.

12. Include support to NetFlow version 5.
Change routers' configurations to export flow data using NetFlow version 5:

```
Router(config)# ip flow-export version 5
```

Data acquisition with pcap

Python references: pyshark - Python packet parser using wireshark's tshark, http://kiminewt.github.io/pyshark/
- https://github.com/KimiNewt/pyshark

13. Download and test the basePCap.py script, by capturing the TCP/IP packets flows between your PC and all external networks,

```
    python basePCap.py -i eth0 -c <pc_ipaddr> -s 0.0.0.0/0
```

14. Test the basePCap.py script, by capturing the TCP/IP packets flows between your PC and YouTube servers during the visualization of videos. Discover your machine IP address (*pc_ipaddr*) and the range of IPv4 addresses for your location (*yt_net*). From UA network, *yt_net* should be (extending the network, and by approximation) 194.210.238.0/24.

```
    python basePCap.py -i eth0 -c <pc_ipaddr> -s <yt_net>
```

15. Complete the code to store on a text file the number of packets and bytes (download and upload) per sampling interval (passed as argument, default 1 second) in observation windows of 300 seconds (5 minutes).

Data Acquisition via SSH

16. To the network constructed in 1., add a SWL3 device and activate the Routers' and SWL3 remote console via SSH.

```
 Router1(config)# aaa new-model
 Router1(config)# enable password labcom
 Router1(config)# username labcom secret 0 labcom
 Router1(config)# ip domain-name con.ara.com
 Router1(config)# ip ssh rsa keypair-name sshkey
 Router1(config)# crypto key generate rsa usage-keys label sshkey modulus 2048
 Router1(config)# ip ssh version 2
 Router1(config)# ip ssh time-out 60
 Router1(config)# ip ssh authentication-retries 2
```

Download, test and improve the baseSSHConsole.py by retrieving the ARP tables from all devices, and the Forwarding Table from the SWL3. Construct a basic rule to detect MAC and IP Spoofing attacks.

17. To the network constructed in 1., add a VM with a Linux server with Apache and Samba services. Using as base the provided script baseSSHConsole.py retrieve the log files from both services via SSH. Construct a basic rule to detect high rates of requests/accesses.