

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

POČÍTAČOVÉ A KOMUNIKAČNÉ SIETE

Zadanie 1

Analyzátor sieťovej komunikácie

Akademický rok 2022/2023

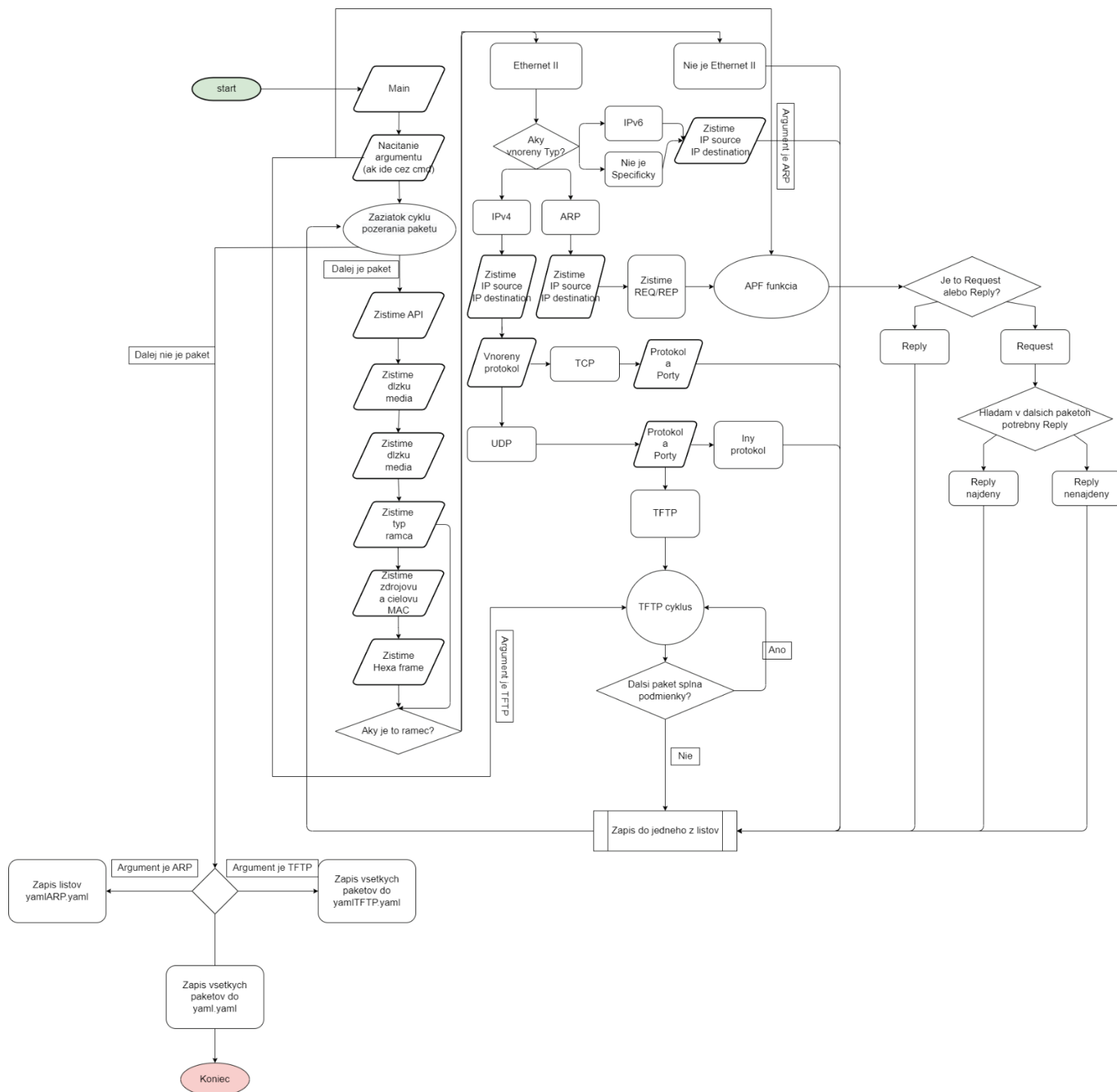
Meno: Rodion Burmistrov

Cvičiaci: Ing. Miroslav Bahleda, PhD.

Dátum: 20.10.2022

Počet strán: 10

- Diagram spracovávania a fungovania riešenia:



- Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách

Vo svojom projekte sa rozoberajú hlavne protokoly na type rámca Ethernet II. Najprv sa zisťuje vnorený protokol v hlavičke rámca. V špeciálne uvedenom textovom súbore je ich predpripravené veľké množstvo. My ale rozoberáme najmä IPv4, IPv6 a ARP.

```
#zistim vnoreny protokol v hlavickke ramca patri do nejakeho znameho. Uvediem mu meno
vnorenyTYP = (''.join('{:02X}'.format(r) for r in raw(packet)[12:14]))
with open('VnorenyTyp.txt') as VT:
    for line in VT:
        line = line.strip()
        splitline = line.split(':')
        if splitline[0] == vnorenyTYP:
            vnorenyT = splitline[1]

#ak je to IPv4 tak este raz pozriem ip-cka (pre istotu som to spravil este raz) a pozriem na vnoreny protokol
if vnorenyT == ('IPv4'):

# ak je to ARP tak este raz pozriem ip-cka, lebo tu sa nachadzaju na inom mieste a pozriem sa tam, kde sa nachaza
#informacia o tom, ci je request alebo reply, to budem potrebovat dalej, ak mame filter na ARP
if vnorenyT == ('ARP'):

# ak je to IPv6 tak prepisem ip-ka, lebo IPv6 ma uplne iny format
if vnorenyT == ('IPv6'):
```

Ďalej paket ide po jednej z troch cest (ak je to Ethernet II).

Ak je to IPv4 tak po zistení zdrojovej a cieľovej IP adresy pozerám na to, aký ma vnorený protokol. Znovu pomocou textového súboru nájdem ci paket spada pod nejaký zo známych prípadov. V projekte si hlavne vyznačím prípad ak je to TCP, alebo UDP.

```
vnorenyPROTOCOL = (''.join('{:02X}'.format(r) for r in raw(packet)[23:24]))
ulohati(ipsource)
#porovnavam s textakom
with open('VnorenyPtorokol.txt') as VP:
    for line in VP:
        line = line.strip()
        splitline = line.split(':')
        if splitline[0] == vnorenyPROTOCOL:
            vnorenyP = splitline[1]

#ak je to TCP tak pozriem si na porty a zistim ci ich poznám
if vnorenyP == ('TCP'):

# ak je to UDP tak pozriem si na porty a zistim ci ich poznám
if vnorenyP == ('UDP'):
```

Na ďalšej vrstve si potrebujem zistiť ci na 4-tej vrstve sa nachádza nejaký port, ktorý by spadal pod známy a uviesť názov každého takého. Tak isto si pomôžem textovými súborami.

```
# porovnavam s textakmi
with open('UDP.txt') as UDPtxt:
    for line1 in UDPtxt:
        line1 = line1.strip()
        splitline = line1.split(':')
        if splitline[0] == src:
            appProtocol = splitline[1]
            src = int(src)
            detector = 1
        elif splitline[0] == dst:
            src = int(src)
            appProtocol = splitline[1]
            dst = int(dst)
            detector = 1
```

```
#porovnavam s textakmi
with open('TCP.txt') as TCPtxt:
    for line1 in TCPtxt:
        line1 = line1.strip()
        splitline = line1.split(':')
        if splitline[0] == src:
            appProtocol = splitline[1]
            src = int(src)
            detector = 1
        elif splitline[0] == dst:
            src = int(src)
            appProtocol = splitline[1]
            dst = int(dst)
            detector = 1
```

V projekte rozoberám prípad použitia špeciálneho porta, ktorý patri protokolu UDP – TFTP.

Úlohou je vypísať a zistiť všetky rámce, s ktorými port TFTP ma komunikáciu. Najprv si zapíšem údaje samotného rámca, ktorý ma TFTP port 69.

```
#ak protokol je TFTP tak si zapisem jeho udaje do "Globalnych" premennych, aby mi pomohli dalej
    if appProtocol == 'TFTP':
        tftpnumber = tftpnumber + 1
        GLOBALsrc = src
        GLOBALipd = ipdestination
        GLOBALips = ipsource
        GLOBALa = 1
```

U ďalšieho paketa pozriem si, že ci spadá pod podmieni komunikácie. Ak áno, tak rámce si pokračujem zapisovať do listu, ktorý sa nasledovne zapíše do špeciálne uvedeného súboru, dokým komunikácia sa neukončí.

```
    if GLOBALa == 1: #zaciatok nie je tu, lebo premenna GLOBALa bude == 1 az ked sa stretne TFTP protokol prvý krat
        if (GLOBALsrc) == dst:
            if (str(GLOBALipd) == ipsource) or (str(GLOBALipd) == ipdestination):
                if (str(GLOBALips) == ipdestination) or (str(GLOBALips) == ipsource):
#vyssie som porovnaval ze ci ten dalsi a dalsi paket splna podmienky komunikacii
#ak ano, tak sa zapise do samotneho listu
#ked komunikacia sa skonci, pomocne premenne sa vynuluju
        else:
            GLOBALa = 0
            GLOBALsrc = 0
            GLOBALipd = ''
            GLOBALips = ''
```

Vo výslednom dokumente yaml komunikácia sa vyznačuje nasledovne.

- NOVA KOMUNIKACIA: 1

- NOVA KOMUNIKACIA: 3

- NOVA KOMUNIKACIA: 4

- NOVA KOMUNIKACIA: 2

Ďalším vnoreným protokolom v hlavičke rámca je ARP. Tým sa veľmi do hĺbky v projekte nezaobráame, ale ak máme nastavený filter na ARP, tak program musí nájsť všetky dvojice request-reply, alebo, ak dvojica nie je – zapísať rámec ako nekompletnú komunikáciu. Vo výsledku sa to celé musí zapísať do zvláštného yaml súboru.

Najprv si nájdeme informáciu o tom, či ARP rámec je reply, alebo request.

```
requestorreply = ('.'.join('{:02X}'.format(r) for r in raw(packet)[21:22]))
```

```
if requestorreply == ('01'):
    REQUEST = 'REQUEST'
```

```
if requestorreply == ('02'):
    if ramec not in GLOBALARP:
        REPLY = 'REPLY'
```

Ak rámec je request, tak prejdeme všetky rámce, kým nenajdeme mu reply, ak najdeme, zapíšeme oba rámce ako kompletnú komunikáciu do listu, ak nenajdeme – zapíšeme rámec ako nekompletnú komunikáciu do iného listu.

```
#ak ARP je Request, tak rýchlo prejdeme všetky pakety, zistím všetky informácie o pakete a hlavne či je to ARP
if requestorreply == ('01'):
```

```
#ak je to Reply, ktorý ešte nebol nájdený tak zistím či spĺňa podmienky
#a ak ano, tak zapíšeme si obidva pakety do listu kompletných komunikácií
```

```
if requestorreplylooking == ('02'):
    if rameclooking not in GLOBALARP:
```

```
#ak sa par nenasiel, tak rámec sa zapíše do listu nekompletných komunikácií
if ramec not in GLOBALARP:
```

Ak na vstupe je rámec reply, tak automaticky ide do listu nekompletnej komunikácie, lebo nemôže ísť reply pred requestom

```
if requestorreply == ('02'):
    if ramec not in GLOBALARP:
        REPLY = 'REPLY'
        ARPlistfailed.append({
            'number_comm': new,
            'packets': ''
        })
        ARPlistfailed.append({
            'frame_number': ramec,
            'len_frame_pcap': API,
            'len_frame_medium': Media,
            'frame_type': outputtype,
            'src_mac': outputzdroj,
            'dst_mac': outputciel,
            'ether_type': vnorenyT,
            'arp_opcode': REPLY,
            'src_ip': ipsource,
            'dst_ip': ipdestination,
            'hexa_frame': ruamel.yaml.scalarstring.LiteralScalarString(fullPacket)
        })
```

Posledným vnoreným protokolom v hlavičke rámca bol IPv6. Ten mnou v projekte sa nerozoberal, ale pri ňom bolo dôležité upresniť že zistenie jeho zdrojovej a cieľovej IP adresy bolo úplne odlišné od ostatných.

```
# ak je to IPv6 tak prepisem ip-ka, lebo IPv6 ma uplne iny format
    if vnorenyT == ('IPv6'):
        ipsource = (''.join('{:02X}'.format(r) for r in (raw(packet)[22:38])))
        ipsource = (':'.join(ipsource[i:i + 4] for i in range(0, len(ipsource), 4)))

        ipdestination = (''.join('{:02X}'.format(r) for r in (raw(packet)[38:54])))
        ipdestination = (':'.join(ipdestination[i:i + 4] for i in range(0, len(ipdestination), 4)))
```

Takto vyzerá výpis IP pre IPv6:

```
ether_type: IPv6
src_ip: FE80:0000:0000:0000:3D03:3447:1C53:6B5F
dst_ip: FF02:0000:0000:0000:0000:0000:0001:0003
```

Takto vyzerá výpis IP pre ostatne protokoly:

ether_type: IPv4	ether_type: ARP
src_ip: 12.0.0.1	src_ip: 12.0.0.1
dst_ip: 12.0.0.5	dst_ip: 12.0.0.1

- Príklad štruktúry externých súborov pre určenie protokolov a portov

Najprv sa používa textový súbor ProtocolsStvorka.txt, ten používam keď prijímam argument pre prepínač „-p“, ten slúži na porovnanie vstupu argumentu, či je zmysluplný, alebo nie (ale popravde ten mi z nejakého dôvodu nefunguje, a prepínač vypíše chybu len keď argument na vstupe bude žiadny). Vyzerá nasledovne:

DNS	SSH
TIME	TELNET
DHCP	SMTP
BOOTPC	DOMAIN
TFTP	FINGER
NETBIOS-NS	HTTP
NETBIOS-DGM	POP3
SNMP	SUNRPC
SNMP-TRAP	NNTP
ISAKMP	NETBIOS-SSN
SYSLOG	IMAP
RIP	BGP
TRACEROUTE	LDAP
ECHO	HTTPS
Chargen	MICROSOFT-DS
FTP-DATA	SOCKS
FTP-CONTROL	

Ďalší textový súbor ktorý používam je VnorennyTyp.txt. Ten slúži na určenie známeho protokolu v hlavičke rámca. Obsahuje názov protokolu a jeho číslo, oddelene dvojbodkou „:“. Vyzerá nasledovne:

```
0800:IPv4
86DD:IPv6
0806:ARP
0842:Wake-on-LAN
22F0:Audio Video Transport Protocol (AVTP)
22F3:IETF TRILL Protocol
22EA:Stream Reservation Protocol
6002:DEC MOP RC
6003:DECnet Phase IV, DNA Routing
6004:DEC LAT
8035:Reverse Address Resolution Protocol (RARP)
809B:AppleTalk (Ethertalk)
80F3:AppleTalk Address Resolution Protocol (AARP)
```

Ďalší textový súbor ktorý používam je VnorenyProtokol.txt. Ten slúži na určenie známeho vnoreneho protokolu. Obsahuje názov protokolu a jeho číslo, oddelene dvojbodkou „:“. Vyzerá nasledovne:

```
00:HOPOPT
01:ICMP
02:IGMP
03:GGP
04:IP-in-IP
05:ST
06:TCP
07:CBT
08:EGP
09:IGP
0A:BBN-RCC-MON
0B:NVP-II
0C:PUP
0D:ARGUS
```

Posledné dva testové súbory UDP.txt a TCP.txt slúžia na určenie známych portov pre vnorene protokolu UDP a TCP. Majú číslo portu a známy názov portu, oddelený dvojbodkou. Vyzerajú nasledovne:







UDP:

```
35:DNS
37:TIME
67:DHCP
68:BOOTPC
69:TFTP
137:NETBIOS-NS
138:NETBIOS-DGM
161:SNMP
162:SNMP-TRAP
500:ISAKMP
514:SYSLOG
520:RIP
33434:TRACEROUTE
```

TCP:

```
7:ECHO
19:Chargen
20:FTP-DATA
21:FTP-CONTROL
22:SSH
23:TELNET
25:SMTP
53:DOMAIN
79:FINGER
80:HTTP
110:POP3
111:SUNRPC
119:NNTP
139:NETBIOS-SSN
143:IMAP
179:BGP
389:LDAP
443:HTTPS
445:MICROSOFT-DS
1080:SOCKS
```


- Opísané používateľské rozhranie

	.idea	20.10.2022 22:45	Папка с файлами
	Dokumentation	20.10.2022 22:48	Папка с файлами
	Protocols	20.10.2022 22:49	Папка с файлами
	Wireshark	13.10.2022 17:12	Папка с файлами
	Yaml	20.10.2022 22:46	Папка с файлами
	main	20.10.2022 22:51	JetBrains PyCharm ...

Odovzdaný súbor obsahuje priečinok Dokumentation, v ktorom sa nachádza tato PDF dokumentácia. Priečinok Protocols s textovými súbormi, s ktorými pracuje program. Priečinok Wireshark, s pcap súbormi, aby práca s programom bola jednoduchšia. A s toho istého dôvodu je tu aj priečinok Yaml, kde sa nachádzajú výsledne Yaml súbory.

Aby pustiť program s filtrami, treba odtiaľto ísť z priečinku do comandlinu, kde treba napísať kománd „python main.py -p argument“, kde namiesto argumentu treba napísať filter. V mojom prípade sú to filtre ARP a TFTP.

Aby zmeniť skúmaný .pcap subor – treba zmeniť adresar na začiatku funkciu main()

```
#main
def main():
    pcap = rdpcap("Wireshark/trace-15.pcap") #cesta na pcap
    ramec = 1
    file = open('Yaml/yaml.yaml', 'w') #cesta na yaml
    file.write("name: PKS2022/23\npcap_name: all.pcap\n") #napisem zaciatok yamlu
```

A na začiatku funkciu arpfunckia() (ak bude použitý filter ARP)

```
#funkcia na filter ARP, dostane ARP paket, ku ktorému bude hľadať dvojicu
def arpfunckia(vnorenýT, ramec, API, Media, outputtype, outputzdroj, outputciel, ipsource, ipdestination, fullPacket, requestorreply):
    if vnorenýT == ('ARP'):
        new = 1

    #ak ARP je Request, tak rýchlo prejdem všetky pakety, zistím všetky informácie o pakete a hlavne či je to ARP
    if requestorreply == ('01'):
        REQUEST = 'REQUEST'
        pcap = rdpcap("Wireshark/trace-15.pcap") # cesta na pcap
        rameclooking = 1
```

Výsledok bude preukázaný v .yaml súboroch.

- Voľbu implementačného prostredia

Ako implementačne prostredie som si zvolil programovací jazyk Python 3 a ako programovacie prostredie program PyCharm.

- Zhodnotenie a prípadné možnosti rozšírenia.

Myslím si že celkom mne sa podarilo spraviť projekt Analyzátor sieťovej komunikácii, boli mnou úspešne spravené úlohy 1) 2) 3) a z úlohy 4) spravený prepínač, protokol TFTP a protokol ARP (f, g, j). Nepodarilo sa spraviť len TCP protokol, a ICMP protokol, dalo by sa to rozšíriť program o tieto funkcie.