



**Moje riešenie:**

Program sa spúšťa z **mainOdovzdanie.py**

Pcap súbor som prečítal a uložil si ako string s ktorým som ďalej pracoval

p = bytes\_hex(pkt\_data).decode("utf-8")

Čítal som informácie zatiaľ po dvoch znakoch v stringu a transformoval som si ich podľa potreby na hexadecimálne čísla, decimálne čísla (IP adresa), alebo som ich nechal v stringovom tvare.

Ďalej som si dal do súboru **ethertypes.txt**  tieto základné typy ethernetov a len čítaním prvých 4och znakov som ich porovnával so znakmi v stringu, či sa zhodujú. Nebolo mi zatiaľ potrebné si ukladať tieto informácie z packetu, keďže som výstup rovno zapisoval do výstupného súboru **hexramce.txt.**

0800 IPv4  
0806 ARP  
86dd IPv6

Ak som našiel daný ethertype tak rovnakým štýlom som preň hladal v inom súbore menom **protocols.txt** prislúchajúci protokol.

01 ICMP  
02 IGMP  
06 TCP  
09 IGRP  
11 UDP  
2f GRE  
32 ESP  
33 AH  
39 SKIP  
58 EIGRP  
59 OSPF  
73 L2TP

Ak to nebol Ethernet ale IEEE 802.3 tak tie som rozlišoval v súbore **saps.txt**

ffff IEEE 802.3 - RAW  
e0e0 IEEE 802.3 - IPX  
aaaa IEEE 802.3 - SNAP  
4242 IEEE 802.3 - STP

Celé som to aplikoval v Pythone (pycharme) a konzolu som používal len na pomocné výpisky (rýchly debugger).

Môj výstup:

ramec 2138  
dlzka ramca poskytnuta pcap API - 92 B  
dlzka ramca prenasaneho po mediu - 96 B  
Ethernet II  
Zdrojova MAC adresa: 00 11 2F 67 B7 5F   
Cielova MAC adresa: FF FF FF FF FF FF   
IPv4  
zdrojova IP adresa: 147.175.144.91  
cielova IP adresa: 147.175.145.255  
UDP  
ff ff ff ff ff ff 00 11 2f 67 b7 5f 08 00 45 00  
00 4e 15 a2 00 00 80 11 db 43 93 af 90 5b 93 af  
91 ff 00 89 00 89 00 3a c2 a0 8f 9a 01 10 00 01  
00 00 00 00 00 00 20 45 4f 46 41 45 4a 44 45 44  
46 45 46 45 42 44 44 44 45 43 41 43 41 43 41 43  
41 43 41 43 41 41 41 00 00 20 00 01

2. časť

1. V danom súbore analyzujte komunikácie pre zadané protokoly:
   1. HTTP
   2. HTTPS
   3. TELNET
   4. SSH
   5. FTP riadiace
   6. FTP dátové
   7. TFTP, **uveďte všetky rámce komunikácie**, nielen prvý rámec na UDP port 69
   8. ICMP, uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.
   9. **Všetky** ARP dvojice (request – reply), uveďte aj IP adresu, ku ktorej sa hľadá MAC (fyzická) adresa a pri ARP-Reply uveďte konkrétny pár - IP adresa a nájdená MAC adresa. V prípade, že bolo poslaných viacero rámcov ARP-Request na rovnakú IP adresu, vypíšte všetky. Ak sú v súbore rámce ARP-Request bez korešpondujúceho ARP-Reply (alebo naopak ARPReply bez ARP-Request), vypíšte ich samostatne.

**Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch TCP a UDP aj porty komunikujúcich uzlov.**

V prípadoch komunikácií so spojením vypíšte iba jednu kompletnú komunikáciu - obsahuje otvorenie (SYN) a ukončenie (FIN na oboch stranách alebo ukončenie FIN a RST alebo ukončenie iba s RST) spojenia a aj prvú nekompletnú komunikáciu, ktorá obsahuje iba otvorenie spojenia. Pri výpisoch vyznačte, ktorá komunikácia je kompletná.

Ak počet rámcov komunikácie niektorého z protokolov z bodu 4 je väčší ako 20, vypíšte iba 10 prvých a 10 posledných rámcov tejto komunikácie. **(Pozor: toto sa nevzťahuje na bod 1, program musí byť schopný vypísať všetky rámce zo súboru podľa bodu 1.)** Pri všetkých výpisoch musí byť poradové číslo rámca zhodné s číslom rámca v analyzovanom súbore.

Prostredie som vzhľadom na nasledujúci filter urobil interaktívne, čiže používateľ si v konzole sadá meno pcap súboru, čo chce analyzovať, a potom nasledujúce možnosti:

*Zadaj cislo pre akciu:*

*'1' - pre vypis vsetkych ramcov*

*'2' - pre specificke komunikacie*

*2*

*Zadaj ktoru komunikaciu chces vyfiltrovat:*

*'1' - HTTP*

*'2' - HTTPS*

*'3' - TELNET*

*'4' - SSH*

*'5' - FTP-CONTROL*

*'6' - FTP-DATA*

*'7' - TFTP*

*'8' - ICMP*

*'9' – ARP*

Program som rozdelil do viacerých funkcí:

def filter\_pcap\_tcp(file\_name, filter\_name):

def filter\_pcap\_udp(file\_name, filter\_name):

def filter\_pcap\_icmp(file\_name, filter\_name):

def filter\_pcap\_arp(file\_name, filter\_name):

Každá slúži na filtrovanie daného TCP/UDP/ICMP alebo ARP protokolu, a následné vypísanie daných rámcov spadajúcich do týchto ‘filtrov’.

Postup som zvolil jednoduchý a podobný v každej funkcii.

Najprv som overil, že či pri TCP a UDP daný rámec spadá pod Ethernet II a IPv4 hlavičku a ďalej som už mohol hľadať jednotlivé protokoly.

if(filter\_name == "HTTP" or filter\_name == "HTTPS" or filter\_name == "TELNET" or filter\_name == "SSH" or filter\_name == "FTP-DATA" or filter\_name == "FTP-CONTROL"):

Riadil som sa na základe TCP/UDP server portov, ktoré som si uložil do **tcpports.txt** a**udpports.txt** na základe ich hexadecimálneho tvaru

0007 ECHO  
0013 CHARGEN  
0014 FTP-DATA  
0015 FTP-CONTROL

...

Pri ICMP som sa pozeral už na informácie z hlavičky type a code

type = p[68:70]  
code = p[70:72]

Tieto v nasledujúcom poradí som si uložil do súboru **icmptypes.txt**

0000 Echo reply  
0300 Destination network unreachable  
0301 Destination host unreachable  
0302 Destination protocol unreachable

...

Nakoniec som riešil ARP pri ktorom komunikujú packety medzi sebou – Request a Respond

Jeden packet si pýta MAC adresu druhého len na základe informácie o jeho IP adrese a Respondent mu ju má za úlohu poslať.

Operation 0001 – request

Operation 0002 - respond

Komunikacia c.1  
ARP-Request, IP adresa: 10.20.30.254, MAC adresa: ???  
Zdrojova IP: 10.20.30.1, Cielova IP: 10.20.30.254  
ramec 41  
dlzka ramca poskytnuta pcap API - 42 B  
dlzka ramca prenasaneho po mediu - 64 B  
Ethernet II  
ARP  
Zdrojova MAC adresa: 00 10 18 33 CF 44   
Cielova MAC adresa: FF FF FF FF FF FF   
ff ff ff ff ff ff 00 10 18 33 cf 44 08 06 00 01  
08 00 06 04 00 01 00 10 18 33 cf 44 0a 14 1e 01  
00 00 00 00 00 00 0a 14 1e fe

Komunikacia c.1  
ARP-Reply, IP adresa: 10.20.30.1, MAC adresa: 00 16 47 02 24 40   
Zdrojova IP: 10.20.30.254, Cielova IP: 10.20.30.1  
ramec 42  
dlzka ramca poskytnuta pcap API - 60 B  
dlzka ramca prenasaneho po mediu - 64 B  
Ethernet II  
ARP  
Zdrojova MAC adresa: 00 16 47 02 24 40   
Cielova MAC adresa: 00 10 18 33 CF 44   
00 10 18 33 cf 44 00 16 47 02 24 40 08 06 00 01  
08 00 06 04 00 02 00 16 47 02 24 40 0a 14 1e fe  
00 10 18 33 cf 44 0a 14 1e 01 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00

Úspešné nadviazanie spojenia – komunikácie!

**Zhodnotenie:**

Pri riešení programu som sa snažil vyhýbať neefektívnemu ukladaní MAC či IP adries, čo sa mi aj podarilo, lebo som ich rovno zapisoval zo stringovej podoby do výstupného súboru **hexaramce.txt.** Nepodarilo sa mi implementovať počítanie source a destination adries pri IPv4 ale mal som rozrobené riešenie, pri ktorom by som si použitú adresu uložil ako strig (hexadecimálne hodnoty) do výstupného súboru aby pre každú IP bola dĺžka rovnaká a potom na ďalšom rámci by som buď za adresu napísal počet výskytov. Nové IP by som zapisoval na samostatný riadok a pri opakovaní by som číslo za IP adresou navyšoval

1833CFFF 55

1921DAFF 21