Fakulta informatiky a informačných technológií STU v Bratislave

Ilkovičova 2, 842 16 Bratislava 4

Počítačové a komunikačné siete

Analyzátor sieťovej komunikácie

Ondrej Špánik

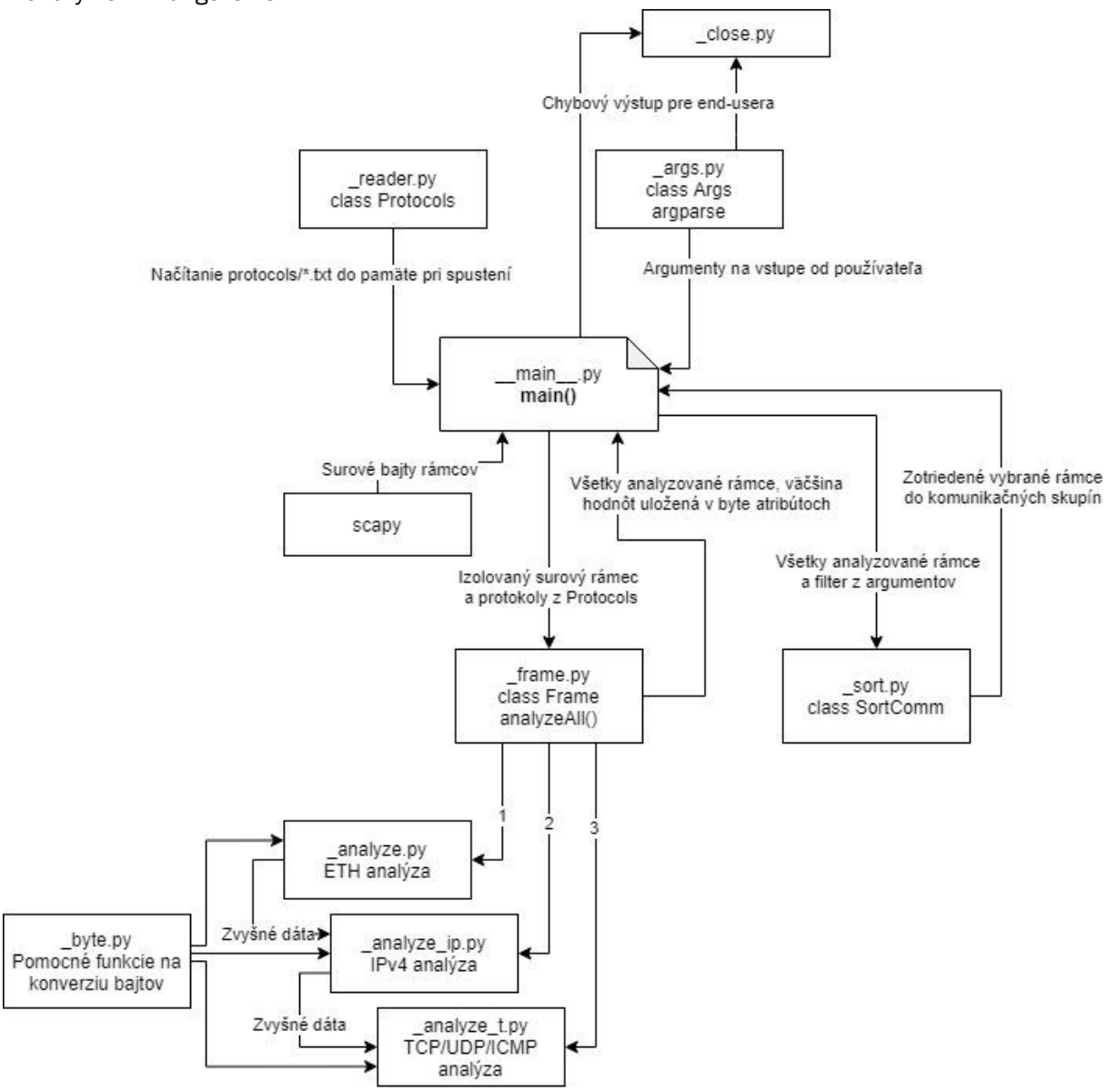
ID: 103151

Meno cvičiaceho:	Ing. Lukáš Mastiľak
Časy cvičení:	Štvrtok 18:00
Akademický rok:	2022/23 ZS

Obsah

1. Koncepcia riešenia

Blokový návrh fungovania



2. Mechanizmus analyzovania protokolov

Podľa jednotlivých vrstiev:

Vrstva sieťového rozhrania (ETH)

```
# DATA LINK HEADER
# \_ MAC
self.eth_dst = _bytes[0:6] # 0-5 bajt (6B)
self.eth_src = _bytes[6:12] # 6-11 bajt (6B)

# \_ LENGTH + STANDARD || ETHERTYPE
len_or_type = btoi(_bytes[12:14]) # 12-13 bajt (2B)
if (len_or_type <= 1500):
    self.eth_len = _bytes[12:14]

    # STANDARD
    dsap_or_raw = _bytes[14:16].hex() # 14-15 bajt (2B)
    # 802.3 RAW                    0xFFFF
    # 802.2 SNAP (== LLC + SNAP)   0xAAAA
    # 802.2 LLC                    0x____
    if (dsap_or_raw == "ffff"):
        self.eth_std = self.Eth_stds.IEEE_802_3_RAW
    elif (dsap_or_raw == "aaaa"):
        self.eth_std = self.Eth_stds.IEEE_802_3_LLCSNAP
    else:
        self.eth_std = self.Eth_stds.IEEE_802_3_LLC

elif (len_or_type >= 1536):
    self.eth_std = self.Eth_stds.ETHERNET2
    self.eth_type = _bytes[12:14]
    self.has_eth_type = True
    self.data = _bytes[14:] # 14 bajt po koniec
    return
else:
    self.eth_std = self.Eth_stds.UNKNOWN
    self.data = _bytes[14:] # assumed
    return
```

Dôjde k rozpoznaniu druhu Ethernetového spojenia na základe štyroch podporovaných štandardov:

- 802.3 RAW
- 802.3 LLC+SNAP
- 802.3 LLC
- ETHERNET 2

V niekorých daných štandardoch sú definované najpodstatnejšie zložky pre ďalšie pokračovanie analýzy:

- EtherType: nachádza sa na rôznych miestach ETH2 a LLC+SNAP, v ostatných nie je prítomný. Je potrebný pre identifikáciu IPv4
- Data: Zvyšné dáta, ktoré sa posúvajú do analýzy na ďalšej vrstve

V tomto kroku okrem iného sú uložené source a destination MAC adresy a v prípade RAW identifikovaný IPX.

Sieťová vrstva (IP, ICMP)

```
self.protocol = _data[9:10] # 6.bajt od zaciatku IPv4 (1B)
self.protocol_str = self.str_ip(btoi(self.protocol))

self.ip_src = _data[12:16] # 12-15 bajt (4B)
self.ip_dst = _data[16:20] # 16-19 bajt (4B)
self.data = _data[20:] # 20+ bajt
return
```

Keďže na základe EtherType je známe IPv4, stačí vytiahnúť potrebné dáta:

- Protokol TCP/UDP/ICMP
- Source IP adresa
- Destination IP adresa

A pokračovať ďalej so zvyšnými dátami na TCP/UDP analýzu.

ICMP je riešený spoločne s TCP/UDP vzhľadom na jednoduchosť riešenia.

Transportná vrstva (TCP, UDP)

```
if (_protocol_str in ["TCP","UDP"]): # TCP, UDP
    self.port_src = btoi(_data[0:2])
    self.port_dst = btoi(_data[2:4])

    if (_protocol_str == "TCP"): # TCP
        # first half of the 12th byte, byte has 8 bits,
        # the actual length of the header is "the first
        self.header_len = (_data[12] >> 4) * 4 # single
        self.data = _data[self.header_len:]

        nonce_mask = 15 # 00001111
        self.flags = bytes([
            (nonce_mask & _data[12]), # last 4 bits of
            _data[13] # 13th byte for CWR,ECN,Urgent,AC
            ])

        self.flags_str = zeroPrefix(itobin(btoi(self.fl
    elif (_protocol_str == "UDP"): # UDP
        self.header_len = 8 # should always be 8
        self.data = _data[8:]

elif (_protocol_str == "ICMP"): # ICMP
    self.type = _data[0]
    self.code = _data[1]
else:
    self.unsupported = True
```

Na základe protokolu obdržaného z hlavičky IPv4 ďalej pokračuje analýza.

V prípade TCP sú identifikované aj jednotlivé flagy, ktoré sú pomocou slovníka aj neskôr ľudsky popísané.

TCP flagy zdieľajú polovicu prvého bajtu s dĺžkou TCP hlavičky, ktorá na rozdiel od UDP hlavičky môže mať variabilnú dĺžku, minimálne avšak 20B. V kóde je možné vidieť bitové operácie pre vytiahnutie podstatných častí v oboch prípadoch.

Vďaka dĺžke TCP hlavičky je možné ďalej pokračovať v analýze dát, tam už na rad prichádza aplikačná vrstva s protokolmi ako HTTP, DHCP, FTP, atď.

Zoskupovanie podľa komunikácií

- Špecificky implementované a testované pre:
    - TFTP
    - ICMP
    - ARP
- Niektoré ďalšie protokoly tiež podporované
- Cieľom zoskupovania je informovať používateľa o všetkej komunikácií medzi istým klientom a serverom na danom protokole. Pre zoskupovanie treba použiť špeciálny argument pri spúšťaní programu.

3.Štruktúra externých súborov pre protokoly/porty

Externé súbory je všetky možné nájsť v zložke protocols.

Pre správne fungovanie programu je potrebné aby všetky existovali pri spustení. Môžu byť aj prázdne.

V prípade chýbajúceho páru v danom súbore je predvolené informovať používateľa, že sa jedná o neznámy EtherType, TCP port a pod.

Súbory sú načítané hneď po spustení programu a následne sa k nim už nepristupuje v ciele zrýchlenia programu ako aj zamedzenia možnosti akéhokoľvek pádu z dôvodu spamového prístupu.

eth_types.txt	0x???? Meno - 0x???? je hexadecimálne znázornený EtherType - Meno je ľubovoľné pomenovanie pre daný EtherType - Zobrazené pre koncového používateľa v bežnom výpise analýzy z programu
icmp_types.txt	n Popis - n je dekadické číslo daného ICMP typu od ktorého sa odvodí popis - Popis je ľubovoľný zrozumiteľný popis pre koncového používateľa
ip_protocols.txt	0x?? Meno - 0x?? je hexadecimálne znázornený protokol pod IP - Najpoužívanejšie a ďalej analyzované sú ICMP, TCP, UDP, ktoré sú prvotne identifikované na základe 0x?? čísla - Meno je používané pre koncového používateľa a v prípade ICMP aj pri argumente na zoskupovanie
saps.txt	0x?? Meno - 0x???? je hexadecimálne znázornený SAP - Meno je ľubovoľné pomenovanie pre daný SAP - Zobrazené pre koncového používateľa v bežnom výpise analýzy z programu
tcp_ports.txt	n Skratka - n je číslo portu - Skratka je jednoslovný názov, ktorý možno ľahko vložiť do argumentu pre zoskupovanie (-s Skratka), na základe ktorého sa spätne určí „n“, teda číslo portu z tohto súboru a vykoná zoskupovanie
udp_ports.txt	Rovnako ako pri tcp_ports.txt

4. Používateľské rozhranie

Jedná sa o CLI rozhranie, ku ktorému je prístup cez argumenty programu.

Najpodstatnejší argument je cesta k PCAP súboru, ktorá má byť uvedená ako prvá.

Zvyšné argumenty a popis k nim je možné pozrieť pomocou flagu „-h“, ktorý otvorí vždy pomocníka:

```
usage: . [-h] [-f FIRST] [-c COUNT] [-o OUTPUT] [--print] [-p PROTOCOL] path

PCAP Network Analyzer (PKS) - Ondrej Spanik 2022 - github.com/iairu

positional arguments:
  path                  Path to a *.pcap file, from which to read network packets

options:
  -h, --help            show this help message and exit
  -f FIRST, --first FIRST
                        Which frame to start from (default:1 = first)
  -c COUNT, --count COUNT
                        Number of packets to read (default:-1 = all)
  -o OUTPUT, --output OUTPUT
                        Path to output YAML file - if omitted 'out.yaml' will be
                        used
  --print               Print YAML output to STDOUT instead of saving to file
  -p PROTOCOL, --protocol PROTOCOL
                        Filter by a supported protocol
```

Ukážkový štandardný výpis

```
name: PKS2022/23
pcap_name: _Samples\trace-26.pcap
packets:
- frame_number: 1
  len_frame_pcap: 352
  len_frame_medium: 356
  frame_type: Ethernet II
  src_mac: 00:16:47:02:24:1a
  dst_mac: 01:80:c2:00:00:0e
  ether_type: LLDP
  hexa_frame: |
    01 80 c2 00 00 0e 00 16 47 02 24 1a 88 cc 02 07
    04 00 16 47 02 24 00 04 07 05 46 61 30 2f 32 34
    06 02 00 78 0a 06 53 77 69 74 63 68 0c f7 43 69
    73 63 6f 20 49 4f 53 20 53 6f 66 74 77 61 72 65
    2c 20 43 33 35 36 30 20 53 6f 66 74 77 61 72 65
    20 28 43 33 35 36 30 2d 49 50 53 45 52 56 49 43
    45 53 4b 39 2d 4d 29 2c 20 56 65 72 73 69 6f 6e
    20 31 32 2e 32 28 35 35 29 53 45 37 2c 20 52 45
    4c 45 41 53 45 20 53 4f 46 54 57 41 52 45 20 28
    66 63 31 29 0a 54 65 63 68 6e 69 63 61 6c 20 53
    75 70 70 6f 72 74 3a 20 68 74 74 70 3a 2f 2f 77
    77 77 2e 63 69 73 63 6f 2e 63 6f 6d 2f 74 65 63
    68 73 75 70 70 6f 72 74 0a 43 6f 70 79 72 69 67
    68 74 20 28 63 29 20 31 39 38 36 2d 32 30 31 33
    20 62 79 20 43 69 73 63 6f 20 53 79 73 74 65 6d
    73 2c 20 49 6e 63 2e 0a 43 6f 6d 70 69 6c 65 64
    20 4d 6f 6e 20 32 38 2d 4a 61 6e 2d 31 33 20 31
    30 3a 31 30 20 62 79 20 70 72 6f 64 5f 72 65 6c
    5f 74 65 61 6d 08 10 46 61 73 74 45 74 68 65 72
    6e 65 74 30 2f 32 34 0e 04 00 14 00 04 10 0c 05
    01 0a 14 1e fe 03 00 00 00 01 00 fe 06 00 80 c2
    01 00 01 fe 09 00 12 0f 01 03 6c 00 00 10 00 00
```

Naľavo je štandardný výstup programu do YAML súboru.

```
- frame_number: 2
  len_frame_pcap: 60
  len_frame_medium: 64
  frame_type: IEEE 802.3 LLC
  src_mac: 00:16:47:02:24:1a
  dst_mac: 01:80:c2:00:00:00
  sap: STP
  hexa_frame: |
    01 80 c2 00 00 00 00 16 47 02 24 1a 00 26 42 42
    03 00 00 00 00 00 80 01 00 16 47 02 24 00 00 00
    00 00 80 01 00 16 47 02 24 00 80 1a 00 00 14 00
    02 00 0f 00 00 00 00 00 00 00 00 00
- frame_number: 3
  len_frame_pcap: 60
  len_frame_medium: 64
  frame_type: Ethernet II
  src_mac: 00:16:47:02:24:1a
  dst_mac: 00:16:47:02:24:1a
  ether_type: ECTP
  hexa_frame: |
    00 16 47 02 24 1a 00 16 47 02 24 1a 90 00 00 00
    01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00

...

ipv4_senders:
- node: 10.20.30.1
  number_of_sent_packets: 91
- node: 10.20.30.254
  number_of_sent_packets: 20
- node: 0.0.0.0
  number_of_sent_packets: 3
max_send_packets_by:
- 10.20.30.1
```

...

Na spodnej časti súboru je možné vidieť leaderboard odosielateľov.

5. Implementačné prostredie

Implementácia bola vykonaná v jazyku Python 3.10 pomocou VSCode na operačnom systéme Windows 11.

Medzi použité externé knižnice patrí scapy, ktoré je použité čiste za účelom získania surových bajtov jednotlivých rámcov pre vlastnú analýzu.

Zvyšok programu je programovaný pomocou Python tried, teda objektovo-orientovane.

Prvé spustenie

Pre spustenie je ideálne sprevádzkovať virtuálny environment (venv) pomocou návodu priloženého v README.md, následne venv aktivovať a nainštalovať scapy[complete] pomocou pip.

6. Zhodnotenie a možnosti rozšírenia

V momentálnom stave program disponuje dostatočne robustným riešením pre analýzu jednotlivých častí väčšiny rámcov, avšak pri rozširovaní za to platím najmä časom.

Prvoradý by mal byť refaktoring: Presun nového TFTP kódu pod Filters. Následne úprava spôsobu, akým Filters funguje, keďže mi príde zbytočne abstraktný a je časovo namáhavé s ním manipulovať.

Program sa dá ďalej rozšíriť čo sa týka nielen výpisov, ale aj jednotlivých filterov: Chýba napríklad TCP filter.