Andrejčák Daniel, ID: 120746

PKS- Zadanie 1: Analyzátor sieťovej komunikácie

Implementačné prostredie

Tento projekt je písaný v jazyku Python verzia 3.11.5

K spusteniu tohto programu sú potrebné knižnice:

- Scapy – otvorenie .pcap súborov a načítanie hexdump jednotlivých rámcov
- Ruamel – vytvorenie a zápis do .yaml súborov, ktoré slúžia ako výstup programu
- Argparse – umožňuje používať argumenty pri spúšťaní programu (zadanie názvu .pcap súboru + prepínač pre úlohu 4)

Používateľské rozhranie

Súbor main.py je "hlavný" súbor, ktorý používateľ spustí. Pri spúšťaní programu je očakávaný argument fileName, ktorý je meno .pcap súboru, pre ktorý sa ma spraviť analýza.

Tento súbor musí byť v rovnakom adresári ako main.py !!!

Používateľ môže zadať argument pre analýzu komunikácií (úloha 4). Ten je voliteľný a jeho použitie sa označuje prepínačom -p

Protokoly, pre ktoré je analýza komunikácií podporovaná sú: ARP, ICMP, TFTP, HTTP, HTTPS, TELNET, SSH, FTP-DATA, FTP-CONTROL

Príklad pre korektné spustenie súboru:

python main.py trace-12.pcap

python main.py -p HTTP trace-12.pcap

Výstup programu je .yaml súbor, ktorý ma názov fileName-protokol-output.yaml , kde fileName je meno .pcap súboru zadaného pri spúšťaní programu a protokol je názov protokolu, ktorý bol zadaný ako argument pre analýzu komunikácií (ak nebol zadaný argument pre analýzu komunikácií, tak názov súboru bude fileName-output.yaml)

Príklad názvu výstupného súboru:

trace-12 -output.yaml

trace-12-HTTP-output.yaml

Andrejčák Daniel, ID: 120746

main.py

Súbor main.py je “hlavný“ súbor, v ktorom sa vykonáva načítanie .pcap súboru, vytvorenie objektov Frame reprezentujúce jednotlivé rámce v .pcap súbore, výpis do .yaml súboru a prípadne analýza komunikácií pre vybrané protokoly.

main

V main sa nachádza kód na parsovanie argumentov z CLI, teda prepínač -p (úloha 4) a fileName. Potom sa volá funkcia loadFrames(), ktorá vráti pole objektov typu Frame, ktoré reprezentujú jednotlivé rámce a nakoniec sa v main určí, či ma nastať štandardná analýza rámcov, alebo analýza komunikácií pre zvolený protokol a podľa toho sa zvolí vhodný výpis, ktorý je zabezpečený funkciami pre výpis.

loadFrames()

Vo funkcii loadFrames() sa najprv načítajú hexdump jednotlivých rámcov z .pcap súboru

```
frames = rdpcap(PCAPFILE)
frameList = [bytes(p) for p in frames]
```

Následne sa vytvoria objekty Frame. Ak je použitý nejaký prepínač, pre analýzu komunikácií, tak sa do formatedFrameList uložia iba rámce, ktoré budú relevantné pre dané komunikácie, táto časť kódu nie je zobrazená v dokumentácii (pre -p HTTP sa uložia iba rámce, ktoré majú aplikačný protokol http)

```
formatedFrameList = []
for i in range(0, len(frameList)):

    ramec = frame.Frame(i+1, frameList[i])

    formatedFrameList.append(ramec)

return formatedFrameList
```

funkcie WriteYaml()

Tieto funkcie slúžia na výpis do .yaml súboru.

Vytvoria hlavičku a vypíšu rámce, alebo komunikácie

Tieto funkcie nijak neovplyvňujú analýzu rámcov a komunikácií a preto sa im dokumentácia nevenuje podrobnejšie

Zoznam všetkých funkcii na výpis

```
#_____funkcie na vypisy do yaml suboru_____
def defaultWriteYaml(frameList):

def arpWriteYaml(completeComms, partialRequestComms, partialReplyComms):

def tftpWriteYaml(comms):

def icmpWriteYaml(comms, partialComms):

def tcpWriteYaml(comms, partialComm, protocol):
```

Andrejčák Daniel, ID: 120746

funkcie pre analýzu komunikácií

Tieto funkcie dostanú ako argument iba pole rámcov, ktoré sú relevantné pre danú komunikáciu (napr. ICMP filter dostane ako argument pole ICMP rámcov). To je zabezpečené vo funkcii loadFrames(). Výstupom funkcií je pole kompletných komunikácií a v niektorých funkciách aj nekompletné komunikácie

arpSwitch()

Funkcia filtruje rámce s ARP etherType do komunikácií. Princíp je, že hľadá páry request – reply a ak ich nájde, tak ich umiestni do kompletnej komunikácia.

Je cyklus, ktorý prechádza všetkými rámcami

- ak narazí na request, tak ho priradí do poľa partialRequest – nekompletnej komunikácie určenej pre request

- ak narazí na reply, ku ktorému nájde request, tak request vyberie z partialRequest a spolu s reply ich umiestni do complete

- ak narazí na reply, ku ktorému nenájde request tak ho umiestní do partialReply - nekompletnej komunikácie určenej pre reply bez request.

Po prejdení všetkých rámcov budú v 3 poliach: páry, request bez reply a reply bez request. Potom sa volá funkcia arpWriteYaml(), ktorá ich vypíše do .yaml súboru.

tftpSwitch()

Funkcia filtruje rámce s TFTP protokolom do komunikácií. Princíp je, že jednotlivé rámce sa ukladajú do dictionary, kde key je source IP, destination IP, source port a destination port.

Je cyklus, ktorý prechadza všetkými rámcami

- ak narazí na read alebo write request, vytvorí v dictionary pole pre novú komunikáciu

- ak narazí na data a existuje otvorená komunikácia do ktorej rámec patrí, tak ho pridá do poľa, ktoré reprezentuje danú komunikáciu

- ak narazí na acknowledgement a existuje otvorená komunikácia do ktorej rámec patrí, tak ho pridá do poľa, ktoré reprezentuje danú komunikáciu

    - ak bola veľkosť posledného datagramu menšia ako veľkosť prvého poslaného datagramu, tak komunikáciu presunie do dictionary pre kompletné komunikácie

- ak narazí na error a existuje komunikácia do ktorej rámec patrí, tak ho pridá do poľa, ktoré reprezentuje danú komunikáciu a presunie ho do dictionary pre kompletné komunikácie

Po skončení tohto cyklu je ešte jeden cyklus, ktorý prejde všetkými komunikáciami, ktoré nie sú ešte v kompletných komunikáciách. Ak narazí na komunikáciu v ktorej sa poslal iba 1 datagram a po ňom acknowledgement, tak ju presunie do kompletných komunikácií.

Andrejčák Daniel, ID: 120746

icmpSwitch()

Funkcia filtruje rámce s ICMP protokolom do komunikácií. Princíp je, že jednotlivé rámce sa ukládajú do dictionary, kde key je source IP, destination IP a ICMP identifier. Jednotlivé páry ECHO REQUEST a ECHO REPLY (prípadne Time exceeded) ukláda do komunikácií ako páry. Pretože niektoré rámce môžu byť fragmentované, tak páry môžu byť teoretický aj štvorice, ale kvôli jednoduchosti ich budeme nazývať páry.

Je cyklus, ktorý prechádza všetkými rámcami

- ak narazí na ECHO REQUEST a neexistuje komunikácia, do ktorej rámec patrí, tak vytvorí v dictionary pole pre novú komunikáciu, ak existuje komunikácia, tak rámec pridá do poľa, ktoré reprezentuje danú komunikáciu

- ak narazí na ECHO REPLY, alebo Time exceeded a existuje komunikácia do ktorej rámec patrí, tak ho pridá do poľa, ktoré danú komunikáciu reprezentuje. Ak neexistuje, tak ho rámec priradí do nekompletných komunikácií.

- ak narazí na iný type (Destination unreachable), tak ho daj do nekompletných komunikácií

Po skončení tohto cyklu je ešte jeden cyklus, ktorý rozbalí páry ECHO REQUEST – ECHO REPLY (alebo Time exceeded) a rozdelí komunikácie na kompletné a nekompletné podľa toho, či každý ECHO REQUEST má ECHO REPLY pár. Tie, ktoré majú idú do kompletných a tie, ktoré nie, idú do nekompletných.

tcpSwitch()

Funkcia dostane na vstupe aplikačný protokol. Podľa neho filtruje rámce do komunikácií. Princíp je, že jednotlivé rámce sa ukladajú do dictionary, kde key je source IP, destination IP, source port a destination port. Každej komunikácii vytvorí jedinečné pole flagov nastavených na False, ktoré sa budú meniť na True ak komunikácia bude korektne otvorená a uzavretá. Ak budú všetky flagy True, komunikácia je považovaná za kompletnú

Je cyklus, ktorý prechádza všetkými rámcami

- ak neexistuje komunikácia, do ktorej by rámec patril, tak vytvorí pole, ktoré bude danú komunikáciu reprezentovať a pridá do nej rámec. Zároveň vytvorí pole flagov, ktoré bude jedinečné pre túto komunikáciu.

- ak existuje komunikácia do ktorej rámec patrí, tak rámec pridá do poľa, ktoré reprezentuje danú komunikáciu

- ak má rámec flag SYN, zistí, či to je prvý, alebo druhý SYN, ktorý sa posiela pri otváraní komunikácie a príslušný flag v poli flagov nastaví na True

- ak má rámec flag ACK, zistí, či je to ACK pre prvý SYN, druhý SYN, prvý FIN, alebo druhý FIN a príslušný flag v poli flagov nastaví na True. Ak to bol ACK pre druhý FIN a komunikácia bola korektne otvorená, tak ju presunie do kompletných komunikácií

- ak má rámec flag FIN, zistí či je to prvý, alebo druhý FIN a príslušný flag v poli flagov nastaví na True

- ak má rámec flag RST a komunikácia bola korektne otvorená, tak ju presunie do kompletných komunikácií. Ak nebola korektne otvorená, tak je braná iba ako korektne uzavretá a príslušne flagy nastaví na True

Po skončení tohto cyklu ešte prebehne jeden cyklus, ktorý sa pokúsi nájsť nekompletnú komunikáciu, ktorá iba otvorená, alebo iba uzavretá.

frame.py

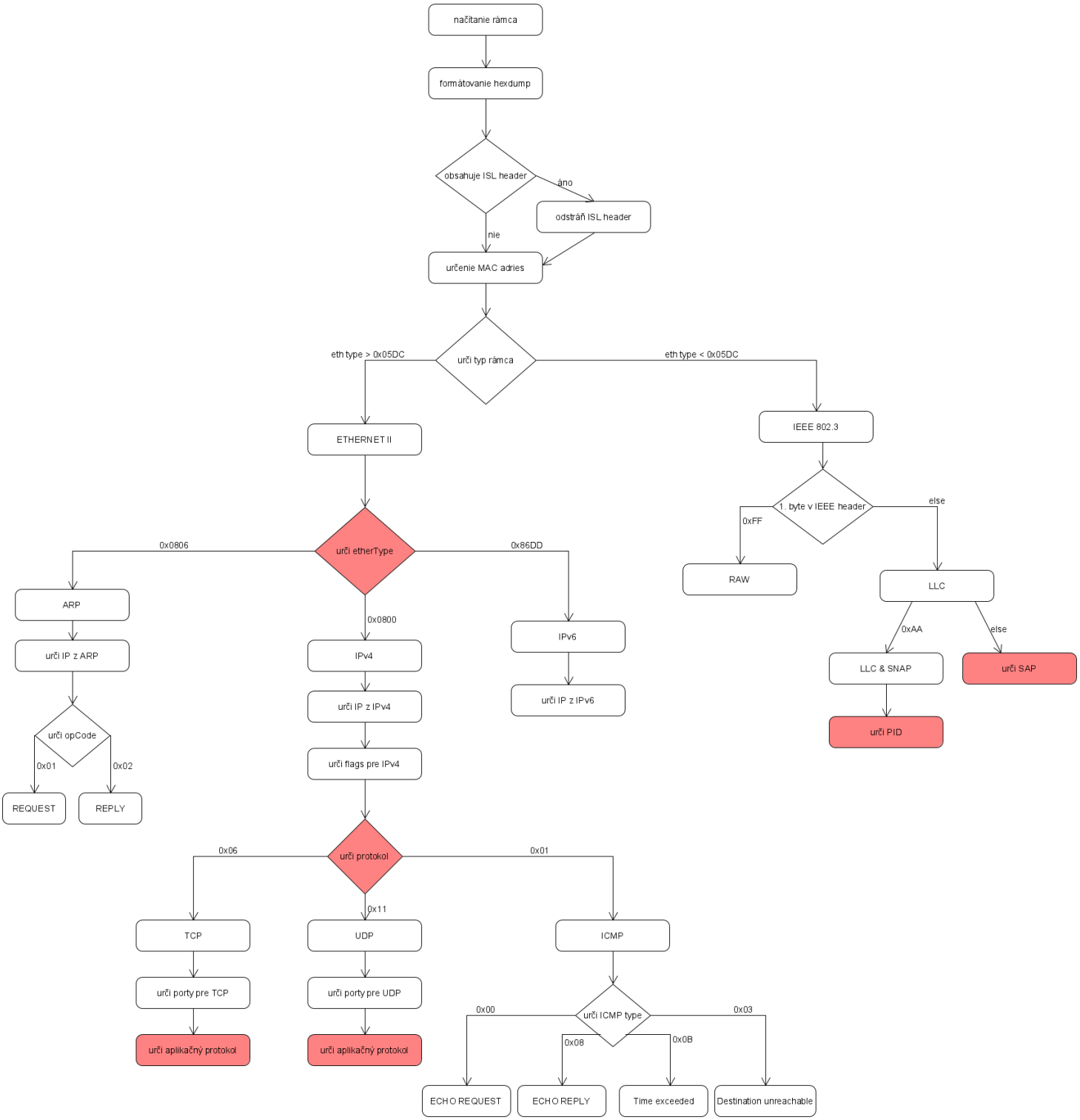
Súbor frame.py je program, v ktorom sa nachádza trieda Frame, ktorá slúži na vytvorenie objektov, ktoré reprezentujú jednotlivé rámce v .pcap súbore.

V tejto triede sa nachádzajú metódy, ktoré z hexdumpu rámca priradia objektu všetky informácie o rámci, ako napríklad typ rámca, MAC adresy, IP adresy, určia vnorený protokol, ...

Princíp získavania týchto informácii je taký, že metódy postupne prechádzajú hexdump, dekódujú jednotlivé byty a priraďujú hodnoty atribútom, ktorým jednotlivé byty zodpovedajú. Napr. načíta obsah prvých 6 bytov a ich hodnotu priradí atribútu srcMac (zdrojová MAC adresa).

Andrejčák Daniel, ID: 120746

Diagram spracovania jednotlivých packetov.



Tento diagram reprezentuje postup spracovania údajov o rámci z jeho hexdump.

Formátovanie hexdump znamená, že sa hexdump usporiada do riadkov po 16 bytov, aby potom jeho následný výpis spĺňal úlohu 1.g. Ešte predtým sa spraví kópia hexdump s ktorou ostatné funkcie pracujú.

Andrejčák Daniel, ID: 120746

Funkcie, ktoré pracujú s informáciami z linkovej a sieťovej vrstve po získaní informácii zmažú hlavičku protokolu s ktorým pracovali, aby bola uľahčená implementácia funkcii, ktoré pracujú s transportnou vrstvou (neplatí pre funkciu, ktorá spracúva ARP etherType).

Červenou farbou sú označené funkcie, ktoré využívajú externý súbor na určenie portov a protokolov.

Externý súbor na určenie protokolov a portov

V priečinku Protocols sa nachádza súbor protocols.yaml, v ktorom sú vypísane známe protokoly a porty. Nachádzajú sa tam vnorené protokoly pre ETHERNET II, SAP, PID, vnorené protokoly pre IPv4 a známe porty pre TCP a UDP.

Program protocols.py slúži na načítanie tohto súboru do formy, s ktorou dokáže Python jednoducho pracovať – dictionary. Tento dictionary sa využíva pri vytváraní objektov Frame, ktoré reprezentujú zachytené rámce v .pcap súboroch.

```
PROTOCOLSFILE = "./Protocols/protocols.yaml"

protocolsFile = open(PROTOCOLSFILE, "r")

protocols = dict(yaml.load(protocolsFile))
```

Výhodou tejto implementácie je jednoduché pridanie napr. ďalšieho známeho portu pre TCP. Stačí ho pridať do protocols.yaml, protocols.py ho automaticky načíta a program ho bude vedieť používať, bez akejkoľvek zmeny kódu

Príklad formátu protocols.yaml

```
ipv4_protocol:
    0x01: ICMP
    0x02: IGMP
    0x06: TCP
    0x11: UDP
    0x67: PIM

tcp_protocol:
    20: FTP-DATA
    21: FTP-CONTROL
    22: SSH
    23: TELNET
```