Slovenská Technická Univerzita v Bratislave, Fakulta informatiky a informačných technológií

Analyzátor sieťovej komunikácie

Počítačové a komunikačné siete

Norbert Matuška
2023/24

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Contents

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

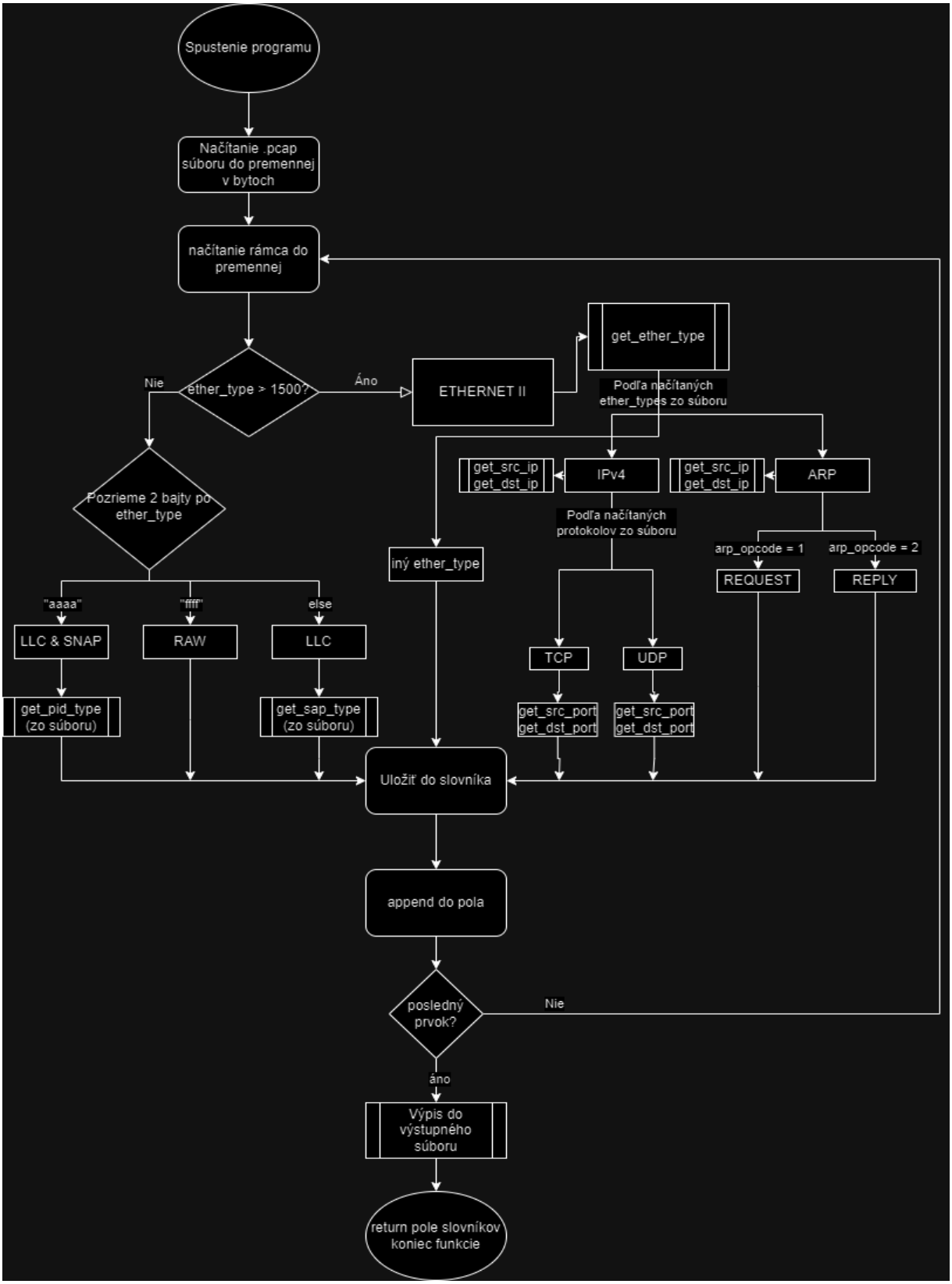
Zadanie úlohy

Navrhnite a implementujte programový analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v .pcap súbore a poskytuje nasledujúce informácie o komunikáciách.

1. Výpis všetkých rámcov v hexadecimálnom tvare postupne tak, ako boli zaznamenané v súbore.
2. Výpis IP adries a vnorených protokol na 2-4 vrstve pre rámce Ethernet II.
3. Na konci výpisu z úlohy 2 uveďte pre IPv4 packety nasledujúcu štatistiku:
    a. Zoznam IP adries všetkých odosielajúcich uzlov a koľko paketov odoslali.
    b. IP adresu uzla, ktorý sumárne odoslal (bez ohľadu na prijímateľa) najväčší počet paketov a koľko paketov odoslal, ak ich je viac, tak uviesť všetky uzly.
4. Váš program rozšírte o analýzu komunikácie pre vybrané protokoly:
    a. TCP
    b. TFTP
    c. ICMP
    d. ARP
    e. Fragmentované ICMP
5. Súčasťou riešenia je aj dokumentácia

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Diagram pre čítanie zo súboru



Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách

Úloha 1 až 3

Po spustení programu zavolám funkciu, v ktorej sa vykonáva úloha 1 až 3. V tejto funkcii si na začiatku zavolám funkciu read_file(), ktorá mi načíta obsah .pcap súboru do premennej packet_bytes. Ďalej inicializujem objekt YAML pre neskoršie použitie, packets_array čo bude pole slovníkov do ktorého budem ukladať spracované packety a získam plnú cestu k súboru (to je iba kvôli tomu aby som mohol vypísať názov testovacieho súboru aj do konzoly a aj do výstupného súboru). Vytvorím si taktiež inštanciu objektu IP_Address_Counter, ktorá počíta a ukladá ip adresy pre štatistiku z 3. úlohy. Ďalej iterujem cez každý jeden packet a vyťahujem potrebné informácie. Ak je ether_type väčší ako 1500 viem, že to je ETHERNET II a môžem ďalej skúmať ether_type. Ak je ether_type IPv4, určím IPv4 protokol, a ak je to UDP alebo TCP, určím pri nich src_port a dst_port samozrejme s tým, že počítam dĺžku ip headeru. Ak je ether_type ARP, určím či sa jedná o REQUEST alebo REPLY a taktiež určím src_port a dst_port. Ak ale ether_type nieje väčší ako 1500 tak to bude IEEE 802.3. Tým pádom sa pozerám DSAP a SSAP a ak ich hexadecimálny tvar je „aaaa“, je to LLC & SNAP, alebo ak tam nájdem „ffff“, bude to RAW. Ak to nie je ani jedno z toho, bude to LLC. Nakoniec si celý packet uložím do slovníka a slovník appendnem do listu už analyzovaných packets. Po skončení iterovania vypíšem všetky zanalyzované packety do výstupného .yaml súboru spolu aj so štatistikou z 3. úlohy. Funkcia nakoniec vráti pole spracovaných packetov pre ďalšie použitie pri filtroch z úlohy 4.

Úloha 4

TFTP

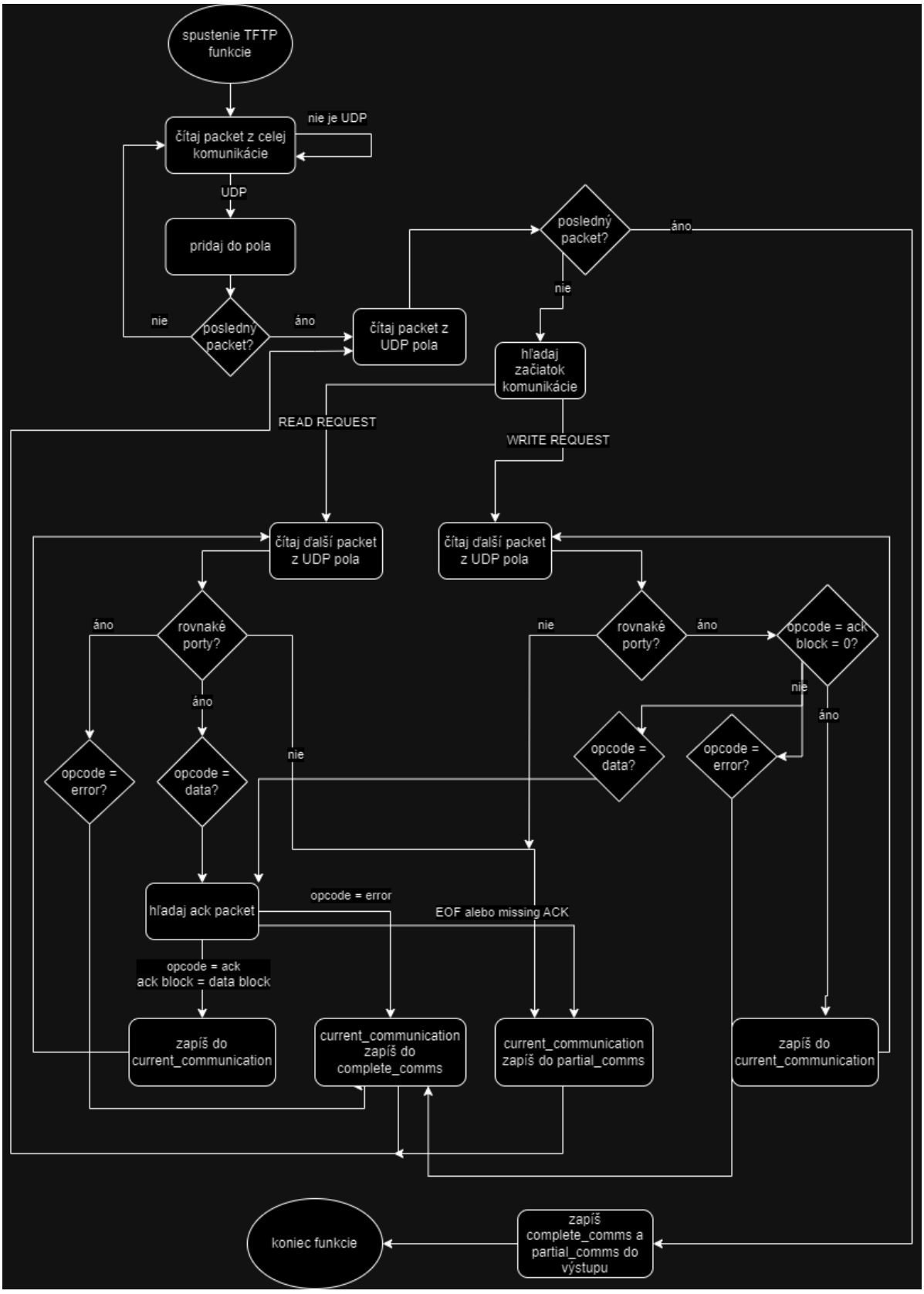
Na začiatku vyberiem všetky packety z pola analyzovaných packetov, ktoré majú protokol UDP. Ďalej iteratívne prechádzam cez pole a porovnávam tftp_opcode.

Ak nájdem 1, znamená to READ REQUEST a teda začiatok komunikácie, ktorú priradím do pola current_communication.

> Následne sa pozerám na ďalšie packety v UDP poli, kontrolujem src a dst ip, ak je zhoda pozriem či packeta má data opcode a hľadám k nej ACK.
>
> > Ak sa nájde ACK k dátovej packete, pridám ju do current_communication. Ak je zároveň posledná dátová packeta menšia ako predošlá dátova packeta, znamená to ukončenie komunikácie a priraďujem current communication do complete communications
> >
> > Ak sa nájde packeta s error opcode, znamená to ukončenie komunikácie a priraďujem current communication do complete communications
> >
> > Ak sa nenájde ani ACK packeta, ani error packeta, priraďujem current communication do partial communications.

Ak nájdem 2, znamená to WRITE REQUEST a teda začiatok komunikácie, ktorú priradím do pola current_communication. Pri WRITE REQUEST to funguje podobne až na to že úplne na začiatku hľadám najprv ACK na WRITE REQUEST a potom to funguje na rovnakom princípe ako pri READ REQUEST.

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40
Diagram pre fungovanie TFTP

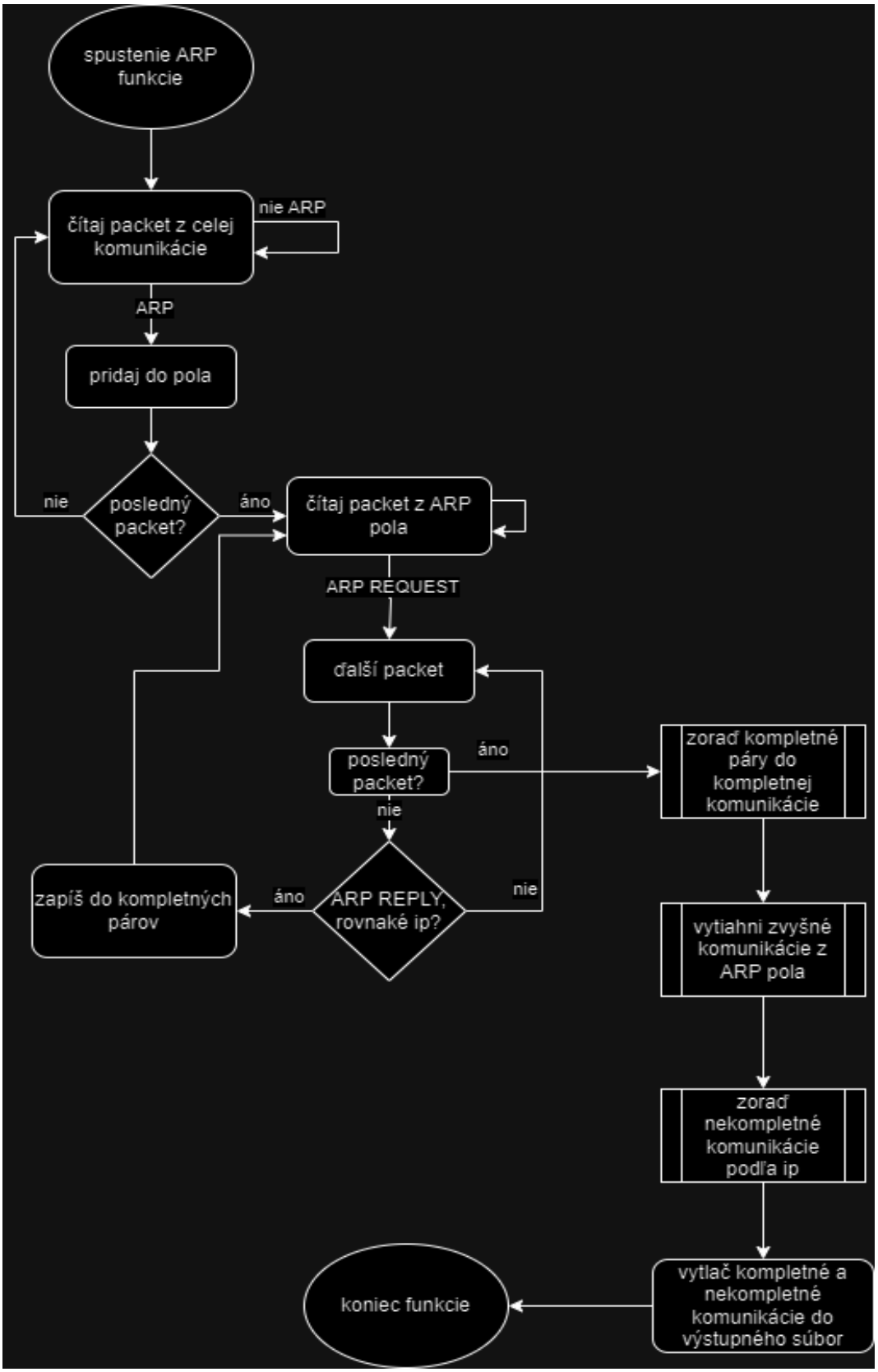


Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

ARP

Na začiatku vyberiem všetky packety z pola analyzovaných packetov, ktoré majú protokol ARP. Následne spárujem všetky ARP requesty s ARP reply podľa ip komunikácie. Ďalej tieto páry spájam do kompletnej komunikácie taktiež podľa ip. Potom si vyberiem všetky zvyšné ARP komunikácie, vložím ich do pola nezoradených nekompletných komunikácii a prechádzam týmto polom a zoraďujem podľa ip do nekompletných komunikácií. Na konci vytlačím kompletné a nekompletné komunikácie do výstupného yaml súboru.

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40
Diagram pre fungovanie ARP

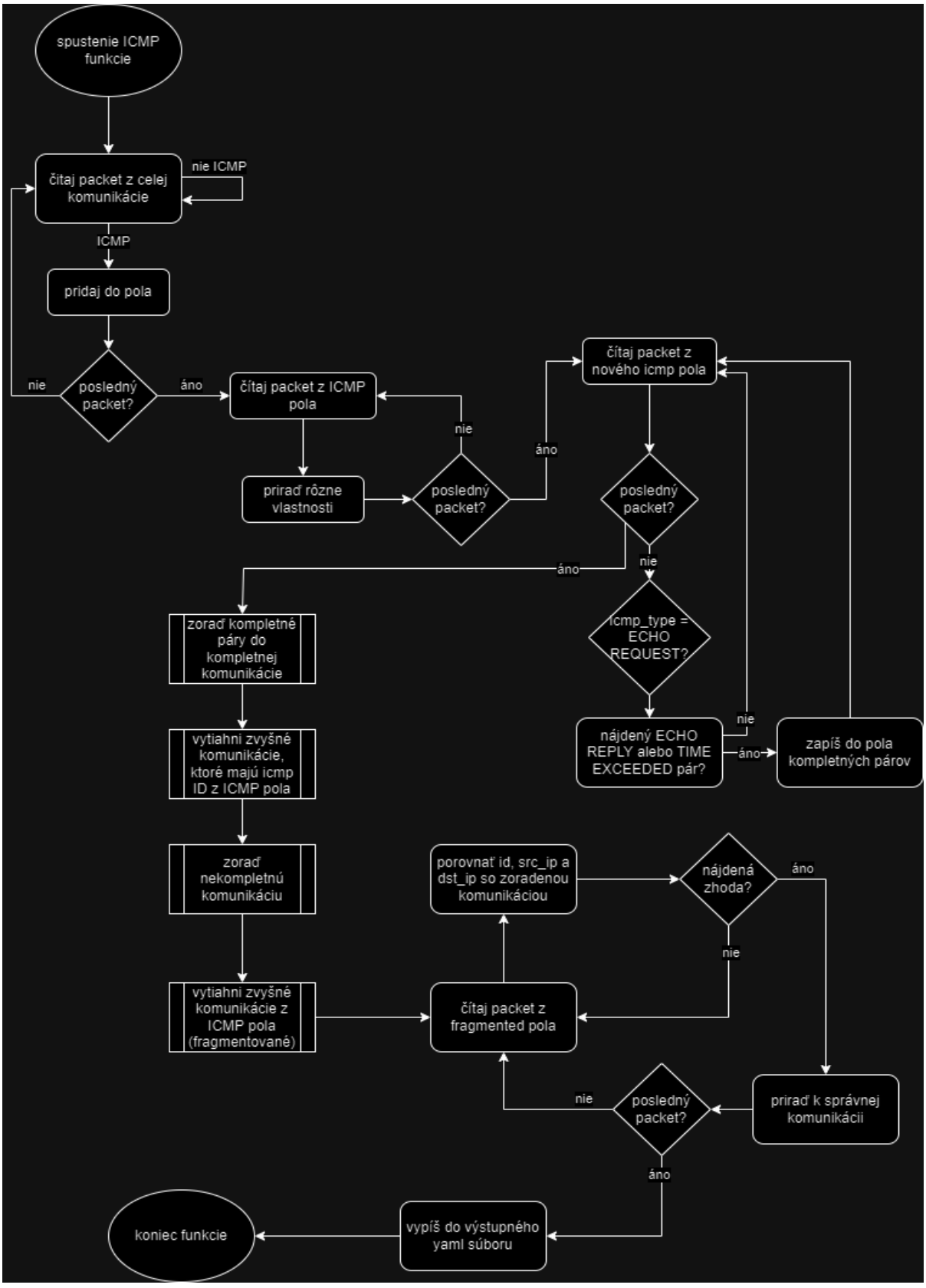


Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

ICMP + frag

Na začiatku vyberiem všetky packety z pola analyzovaných packetov, ktoré majú protokol ICMP. Prejdem celým ICMP polom a určím tam rôzne potrebné dáta ako napr. icmp_id, icmp_seq, flag, frag_offset atď. Následne prechádzam celým polom a hľadám páry ECHO REQUEST a ECHO REPLY alebo ECHO REQUEST a TIME EXCEEDED podľa id, seqn. Ďalej tieto páry spojím do kompletných komunikácií podľa ip. Potom už len vytiahnem zvyšné komunikácie, ktoré nie sú v kompletných a zároveň majú nejaké icmp id a zoradím ich do nekompletných komunikácií. Komunikácie, ktoré ostali v ICMP poli môžu byť jedine fragmentované packety bez icmp id a tie ďalej porovnávam s ip id kompletných a nekompletných komunikácií a pripájam ich k jednotlivým packetom kde patria. Na konci už len vypíšem všetko zaradom.

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40
Diagram pre fungovanie ICMP + frag



Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Príklad štruktúry externých súborov pre určenie protokolov a portov

Takto vyzerá môj súbor pre určenie TCP protokolu:

```
20 FTP-DATA
21 FTP-CONTROL
22 SSH
23 TELNET
25 SMTP
53 DNS
80 HTTP
110 POP3
119 NNTP
137 NETBIOS-NS
139 NETBIOS-SSN
143 IMAP
162 SNMP-TRAP
179 BGP
389 LDAP
443 HTTPS
514 SYSLOG
17500 DB-LSP-DISC
```

Pre ICMP vyzerá zase troška inak ale na podobnom princípe

```
0:ECHO REPLY
3:DESTINATION UNREACHABLE
4:SOURCE QUENCH
5:REDIRECT
8:ECHO REQUEST
9:ROUTER ADVERTISEMENT
10:ROUTER SELECTION
11:TIME EXCEEDED
12:PARAMETER PROBLEM
13:TIMESTAMP
14:TIMESTAMP REPLY
15:INFORMATION REQUEST
16:INFORMATION REPLY
17:ADDRESS MASK REQUEST
18:ADDRESS MASK REPLY
30:TRACEROUTE
```

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Opísanie používateľského prostredia

Funkcia na začiatku vykoná úlohy 1 až 3 a následne sa spýta usera na výber filtra

```
Testing file: trace-26.pcap
MENU
Type one of the following:

ARP
ICMP
TFTP
FRAG

input: TFTP
```

Testovací súbor sa dá vybrať hneď na začiatku kódu vo funkcii:

```
def testing_file():
    file_path = "vzorky_pcap_na_analyzu/"
    file_name = "trace-26.pcap"
    full_file_path = file_path + file_name

    return full_file_path
```

Do premennej file_name napíšeme súbor, ktorý chceme testovať. Musí existovať.

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Voľba implementačného prostredia

Môj program som písal v jazyku Python v PyCharm IDE od JetBrains. Tento jazyk som si vybral kvôli zjednodušeniu práce s bajtmi a aby som využil jednoduché vstavané funkcie v Pythone ako je hexlify, int a str funkcie. Následne som využil knižnice ruamel.yaml a scapy kvôli rdpcap.

Zhodnotenie a možnosti rozšírenia

Môj program nie je úplne najefektívnejšie spracovaný, ako možnosť rozšírenia by som celý program zefektívnil, pretože niektoré funkcie sa vykonávajú zbytočne/neefektívne. Taktiež by som program rozšíril o TCP filter, keby som mal viac času na pracovaní.

Norbert Matuška AIS_ID: 110849
xmatuskan@stuba.sk
Štvrtok 18:00-19:40

Zdroje

Diagramy spracované na stránke:

https://app.diagrams.net/