VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

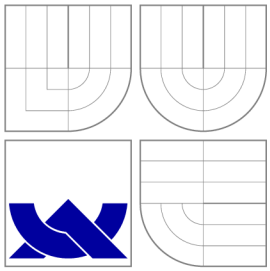
NÁSTROJ PRO SLEDOVÁNÍ RTP STREAMŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

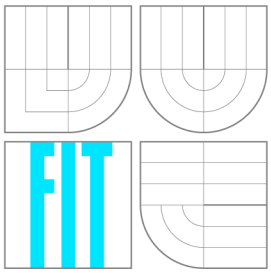
AUTOR PRÁCE
AUTHOR

MARTIN KMEŤ

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO SLEDOVÁNÍ RTP STREAMŮ

A TOOL FOR TRACKING RTP STREAMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KMEŤ

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. JANA SKOKANOVÁ

BRNO 2012

Abstrakt

Tento dokument popisuje vývoj algoritmu a testovací aplikace pro detekci RTP streamů v počítačových sítích. Jedním z výsledků bude nástroj v jazyce Python rozpoznávající RTP streamy v datech odchytávaných ze síťového rozhraní v reálném čase, nebo z dat uložených ve formátu pcap.

Abstract

This document describes the development of the algorithm and test application for detection of RTP streams in computer networks. One of the results will be a tool in Python language recognising RTP streams in data filtered from network interface in real time or from the data in pcap format.

Klíčová slova

stream, multimédiá, RTP, RTCP, real-time přenos

Keywords

stream, multimedia, RTP, RTCP, real-time transfer

Citace

Martin Kmeť: Nástroj pro sledování RTP streamů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Nástroj pro sledování RTP streamů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Jany Skokanové a že jsem uvedl všechny použité literární prameny a publikace.

.....
Martin Kmeť
12. mája 2012

Poděkování

Týmto by som chcel poďakovať Mgr. Jane Skokanovej a Ing. Petrovi Matouškovi, PhD. za pomoc a rady, ktoré viedli ku vzniku tejto práce.

Obsah

Zoznam tabuliek

Kapitola 1

Úvod

Protokol RTP bol navrhnutý pre dáta s potrebou prenosu v reálnom čase, ako napríklad audio alebo video. Jedným z jeho veľkých problémov je obtiažna rozpoznateľnosť, z dôvodu nedostatku stabilných hodnôt v hlavičkách protokolov RTP a ním najpoužívanejšieho transportného protokolu UDP. Tento protokol tak isto nepoužíva štandardizované čísla portov, ale prideľuje ich dynamicky, čiže detekcia na tomto princípe taktiež nie je možná.

Rozpoznať toky dát RTP streamov je dôležité z viacerých dôvodov. Medzi tieto patria napríklad umožniť riadenie kvality služby (QoS), keďže dáta nesené protokolom RTP väčšinou bývajú citlivé na oneskorené doručenie, ktoré môže vznikať najmä v prípade zaťaženia siete. Ďalej môžeme ako príklad uviesť prípad, kedy je potreba vo firemnej sieti povoliť VoIP alebo videokonferenčnú komunikáciu s internetom. Keďže ako je už vyššie uvedené RTP používa dynamické čísla portov, nedá sa toto dosiahnuť pomocou jednoduchého paketového filtra, ale firewall musí byť schopný rozpoznávať RTP streamy.

Väčšina existujúcich nástrojov schopná rozpoznať pakety protokolu RTP na tento účel používa informácie nesené signalizačnými protokolmi (napr. SIP). Táto metóda však zlyháva v prípade, že dáta signalizačného protokolu neboli odchytené alebo rozpoznané (napríklad odchyt dát započal až po začiatku RTP prenosu, alebo bol použitý neznámy signalizačný protokol).

U čitateľa sa predpokladá základná znalosť počítačových sietí a znalosť pojmov s nimi súvisiacimi.

1.1 Ciele práce

Cieľom práce je preskúmať možnosti rozpoznávania RTP streamov bez použitia signalizačných protokolov. Funkčnosť použitých metód bude overovaná na vzorkách reálnych, odchytených dát a výsledky rozpoznávania budú porovnané s výsledkami od existujúcich nástrojov využívajúcich signalizačné protokoly. Cieľom práce nie je vyvinúť absolútne spoľahlivý algoritmus, ale algoritmus schopný rozpoznať reálne toky dát.

1.2 Riešenie

V rámci práce bude vyvinutá aplikácia v jazyku Python s použitím knižnice Scapy. Táto aplikácia bude slúžiť na získanie experimentálnych výsledkov vyvinutých metód a ich následnú demonštráciu. Vyvinutú aplikáciu však bude možné použiť aj ako modul v iných projektoch.

Kapitola 2

Protokoly RTP a RTCP

Oba protokoly boli prvýkrát publikované v Januári 1996 v dokumente RFC 1889[5], ktorý bol v Júli 2003 nahradený dokumentom RFC 3550[6]. Z druhého z uvedených dokumentov je vypracovaná veľká časť tejto kapitoly.

V dokumente sú popisované oba protokoly a to RTP (Real-time Transport Protocol) pre prenos dát v reálnom čase a RTCP (RTP Control Protocol) pre monitorovanie kvality a šírenie informácií o účastníkoch streamu.

RTP streamy väčšinou využívajú služby transportného protokolu UDP[2], ktorý je vhodný najmä vďaka podpore multicastu a malej veľkosti hlavičky. Toto je pri RTP streamoch dôležitá vlastnosť, keďže RTP streamy majú väčšinou charakter veľkého množstva pomerne malých paketov. RTP však môže využiť služby ľubovolného transportného protokolu, ktorý mu umožní oddeliť viac sedení a tok RTP a RTCP dát, čo UDP umožňuje vďaka použitiu portov.

Komunikácia prostredníctvom protokolu RTP prebieha v tzv. sedeniach. V jednom sedení je vždy prenášaný len jeden stream, jedného typu média, ale jeden účastník sa môže v jednom okamihu podieľať na viacerých sedeniach, ktoré sú navzájom nezávislé. Teda napríklad v prípade video-konferencií je video a audio prenášané väčšinou v dvoch samostatných streamoch, ktorý každý vytvorí vlastné sedenie.

V RTP sedení sa môže vyskytovať viac typov zariadení:

Synchronizačný zdroj (SSRC – Synchronization Source) – Zdroj streamu (alebo RTP mixer), ktorý určuje synchronizáciu paketov pre príjemcov streamu. V jednom RTP sedení sa môže vyskytovať len jeden synchronizačný zdroj.

Prispievajúci zdroj (CSRC – Contributing Source) – Jeden zo zdrojov, zasielajúcich dáta mixéru, ktorý z nich skladá výsledný stream. Mixér vkladá jednotlivé identifikátory synchronizačných zdrojov každého streamu, z ktorých skladá výsledný stream a vkladá ich do odosielaných paketov ako zoznam prispievajúcich zdrojov.

Koncový systém – Zariadenie, ktoré odosiela alebo prijíma stream. Do tejto skupiny patria aj SSRC a CSRC.

Mixér – Prechodný systém, ktorý prijíma RTP pakety od viacerých zdrojov a kombinuje ich, pričom môže meniť ich formát. Keďže jednotlivé zdroje nemusia byť synchronizované, stará sa aj o synchronizáciu dát a vytvára vlastné časovanie. Po skombinovaní jednotlivých paketov preposiela výsledný paket ďalej, pričom je uvedený ako synchronizačný zdroj.

Prekladač – Zariadenie, ktoré preposiela RTP pakety bez zmeny synchronizačného zdroja. Môže napríklad meniť formát dát bez kombinovania viacerých zdrojov, prevádzať multicast na unicast atď.

Monitor – Zariadenie prijímajúce RTCP dáta, zisťujúce kvalitu hovoru a zostavujúce štatistiku. Býva vstavané v aplikácii využívajúcej RTP, ale môže byť implementované v samostatnej aplikácii alebo môže existovať aj ako samostatné zariadenie.

RTP podporuje komunikáciu typu „many-to-many“, čo znamená, že v jednom okamihu, môže mať jedno sedenie viac odosielateľov aj príjemcov. O možnosť príjmu sa musí postarať protokol transportnej vrstvy (prevažne UDP pomocou multicastu). Možnosť viacerých odosielateľov je riešená pomocou RTP mixérov.

RTP nebolo navrhnuté pre konkrétny typ dát ako napríklad prenos audia alebo videa, pre ktoré je v súčastnosti najviac využívaný, ale pre prenos všetkých typov dát s potrebou prenosu v reálnom čase. Konkrétny typ použitia špecifikujú RTP profily, ktoré musia byť špecifikované v samostatných dokumentoch. V týchto dokumentoch musia byť uvedené prípadné modifikácie hlavičky, dodefinované ďalšie typy RTP a RTCP správ a mapovanie typov obsahu z hlavičky RTP na konkrétne typy obsahu (napr. kódovanie audia a videa).

Medzi takéto profily patrí napríklad Zabezpečený RTP (SRTP) [1] alebo Profil pre Audio a Video Konferencie s Minimálnou Kontrolou [4], ktorý je dnes suverénne najpoužívanejším RTP profilom a podrobnejšie sa mu budeme venovať v kapitole 2.4.

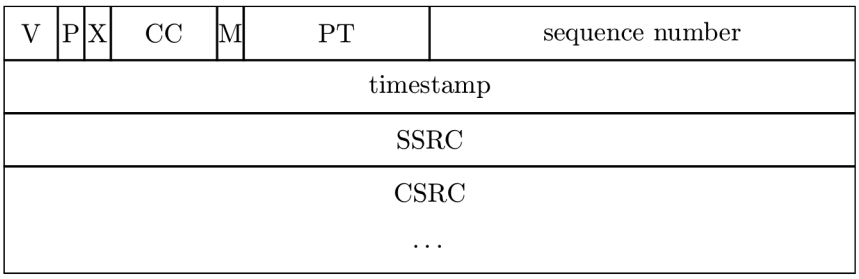
2.1 Protokol RTP

Protokol RTP slúži k samotnému prenosu dát medzi jednotlivými uzlami po sieti.

2.1.1 Formát paketu RTP

Paket protokolu RTP sa skladá z povinnej hlavičky, voliteľného rozšírenia hlavičky a samotných prenášaných dát.

Formát hlavičky, ktorá je v pakete umiestnená bezprostredne na začiatku, je ukázaný na obrázku 2.1.



Obr. 2.1: Hlavička RTP paketu

Popis jednotlivých položiek hlavičky:

V (Version) – Verzia protokolu. U aktuálne využívanej verzie je to vždy 2.

P (Padding) – Ak je tento bit nastavený, dáta nesené paketom obsahuje jeden alebo viacero bajtov, ktoré nie sú užitočnou časťou dát. V tomto prípade posledný bajt nesie počet bajtov pre zahodenie.

X (eXtension) – Ak je nastavené, za fixnou časťou hlavičky bude nasledovať jedno rozšírenie hlavičky.

CC (CSRC Count) – Počet prispievajúcich zdrojov uvedených v hlavičke.

M (Marker) – Interpretácia tohto bitu je ponechaná na jednotlivých profiloch.

PT (Payload Type) – Identifikuje formát obsahu neseného RTP paketom a jeho následnú interpretáciu aplikáciou. Definícia jednotlivých formátov obsahu je ponechaná na jednotlivé profily. Typ môže byť mapovaný na číslo staticky alebo aj dynamicky pomocou iných metód ako RTP (napr. pomocou SDP).

sequence number – Číslo paketu v streame. Čísla jednotlivých paketov sa postupne zvyšujú o jednotku. Počiatočné číslo sekvencie by malo byť volené náhodne.

timestamp – Určuje sa podľa vzorkovania signálu prenášaného pomocou streamu. Musí byť odvodené od hodín, ktoré sa inkrementujú monotónne a lineárne. Dva za sebou vyslané pakety však nemusia mať monotónny priebeh časovej pečiatky. Táto situácia môže nastať ak dáta vznikli v inom poradí ako boli vysielané. Príkladom môže byť video s prekladaným snímkovaním.

SSRC identifier – Identifikačné číslo synchronizujúceho zdroja. Táto 32-bitová hodnota však vôbec nesúvisí s IP adresou zariadenia. Je generovaná spôsobom popísaným v [6] a musí byť v sedení unikátna. V prípade kolízie identifikátorov v sedení, musí jeden zo zdrojov pre seba vygenerovať nové identifikačné číslo.

CSRC identifiers – Identifikačné čísla jednotlivých prispievajúcich zdrojov. Pre hodnoty identifikačného čísla platia rovnaké pravidlá, ako pre synchronizačný zdroj.

V prípade použitia rozšírenia hlavičky, toto musí nasledovať bezprostredne za povinnou časťou. Táto môže obsahovať doplnkové informácie špecifické pre jednotlivé profily. Formát hlavičky nie je jednotný, jedinou spoločnou časťou je druhých 16-bitov rozšírenia, určujúce dĺžku rozšírenia hlavičky v 32-bitových slovách.

Za prípadným rozšírením hlavičky nasledujú samotné dáta. Ich súčasťou možu byť aj informácie potrebné pre spracovanie konkrétneho typu dát aplikáciou. Aj keď tieto informácie môžu byť obsiahnuté v rozšírení hlavičky, doporučuje sa aby boli obsiahnuté až v tejto sekcii.

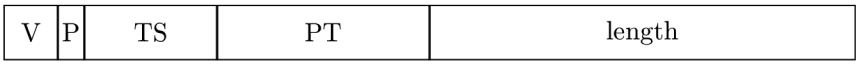
2.2 Protokol RTCP

Protokol RTCP slúži pre obsluhu a podporu služieb poskytovaných protokolom RTP. Medzi jeho hlavné funkcie patrí kontrola kvality služby, pre poskytnutie prostriedkov, pre úpravu rýchlosti odosielania dát zo zdroja a poskytovanie informácií o streame pre príjemcov. RTCP definuje 5 typov správ: SR (Sender Report), RR (Reciever Report), SDES (Source DEScription), BYE a APP, ale norma povoľuje dodefinovanie nových typov profilom v prípade potreby.

RTCP pakety sa prenášajú vo forme zlúčených paketov. Jeden zlúčený paket sa vytvorí spojením viacerých paketov za sebou a až táto výsledná skupina paketov sa zabalí do protokolu nižšej vrstvy. Na začiatku zlúčeného paketu sa musí vždy nachádzať paket typu Sender Report alebo Reciever Report. V prípade, že nie je potrebné preniesť žiadnu

z informácií nesených jedným z týchto paketov na začiatok paketu sa priloží prázdny paket Sender Report. Za týmto úvodným paketom môžu v prípade potreby nasledovať ďalšie pakety týchto typov. Po týchto musí vždy nasledovať paket typu Source Description, ktorý musí minimálne obsahovať položku CNAME a v prípade potreby môže byť nasledovaný ďalšími paketmi tohoto typu. Ako posledné prídu všetky ostatné voliteľné typy paketov, vrátane typov dodefinovaných profilom, ale v prípade potreby paketov typu BYE, tieto musia byť v rámci zloženého paketu umiestnené ako posledné. Všetky pakety nasledujú bezprostredne za sebou V prípade použitia šifrovania sa pred začiatok paketu priloží ešte náhodné 32-bitové číslo. Ak je potrebné pre šifrovanie, prípadne pre potreby protokolov nižších vrstiev zarovnať dáta priložením neužitočných dát, položka padding sa nastaví len v poslednom pakete a nadbytočné dáta sa priložia až za tento paket a teda položka padding nesmie byť v ostatných paketoch nastavená.

Formát jednotlivých paketov je závislý na jeho type. Podrobný výpis a vysvetlenie formátov jednotlivých paketov je nad rámec tohoto dokumentu a je dostupný v [6]. Z tohoto dôvodu bude u jednotlivých paketov uvedený hrubý náčrt štruktúry a niekoľko pre detekciu potencionálne podstatných položiek. Všetky pakety majú však spoločných prvých 32-bitov, ktoré sú vyobrazené na obrázku 2.2. Význam jednotlivých položiek je nasledujúci:



Obr. 2.2: Prvých 32-bitov RTCP paketu

V (Version) – Rovnako ako u hlavičky RTP táto položka určuje verziu použitého protokolu.

P (Padding) – Táto položka je tiež zhodná s protokolom RTP. Určuje, či sa na konci paketu vyskytujú bajty slúžiace pre zarovnanie. Podrobný popis je možné nájsť pri popise hlavičky RTP.

TS (Type Specific) – Význam tejto položky je rozdielny pre rôzne typy paketov a preto bude uvedený pri popise jednotlivých typov.

PT (Packet Type) – Položka určuje, o ktorý z 5 typov RTCP paketu sa jedná.

length – Určuje dĺžku paketu v počte 32-bitových slov mínus 1, vrátane prvých 32-bitov.

2.3 Typy RTCP paketov

2.3.1 SR: Sender Report

Hlásenia zdrojov. Pakety tohoto typu slúžia na prenos štatistík o streame od účastníkov, ktorí zasielajú dáta do sedenia.

V položkách spoločnej časti má `SR` paket v položke `Packet Type` hodnotu 200 a v položke `Type Specific`, je uvedený počet hlásení obsiahnutých v pakete. Za spoločnou časťou nasleduje identifikátor SSRC. Za ňou nasleduje blok informácií od odosielateľa RTCP paketu, nasledovaný niekoľkými blokmi obsahujúcimi samotné hlásenia zdrojov, ktorých počet bol uvedený v spoločnej časti paketu. Ako posledný nasleduje rozšírenie špecifické pre jednotlivé profily.

2.3.2 RR: Reciever Report

Hlásenia príjemcov. Používajú sa pre prenos štatistík od účastníkov, ktorí nezasielajú dáta do streamu. Prípadne sa používa aj pre účastníkov zasielajúcich dáta, ak ich počet presiahne 31.

Formát tohoto typu paketu je veľmi podobný formátu typu `SR`. Jedinými rozdielmi sú: v položke `Packet Type` je hodnota 201 a blok informácií od odosielateľa je vynechaný. Zvyšné položky aj ich poradie je zachované.

2.3.3 SDES: Source DEScription

Popis zdroja. Obsahuje jednotlivé položky pre popis zdroja. Pakety tohoto typu sú informačne najhodnotnejšie, čo sa týka informácií o samotnom streame, preto bude popísaný podrobnejšie.

V spoločnej časti paketu sa v položke `Packet Type` nachádza hodnota 202 a v položke `Type Specific` počet zhlukov informácií. Za spoločnou časťou nasledujú jednotlivé zhluky informácií, ktorých počet bol uvedený.

Jeden zhluk obsahuje v prvých 32 bitoch identifikátor SSRC alebo CSRC, za ktorým nasledujú jednotlivé informačné položky, ktorých formát je veľmi jednoduchý: Prvých 8 bitov tvorí typ položky, nasledovaný 8-bitovou hodnotou dĺžky reťazca v bajtoch, za ktorou nasleduje text v kódovaní UTF-8 s maximálnou dĺžkou 255 bajtov. Začiatok každej položky však musí byť zarovnaný na 32-bitov.

Paket môže obsahovať nasledujúce typy informačných položiek (v zátvorke bude uvedená hodnota v položke `type` na začiatku každej informačnej položky):

CNAME (1) – Zástupné meno zdroja. Identifikuje jednoznačne zdroj streamu a musí byť pre stream jedinečné. Doporučené je plné doménové meno odosielajúceho zariadenia prípadne doplnené o užívateľa, vo formáte `user@host`. Keďže pri kolízií identifikátorov jednotlivých zdrojov dôjde k zmene identifikátora, zástupné meno sa nesmie meniť po celú dobu sedenia.

NAME (2) – Názov reálne opisujúci stream. Môže byť v ľubovolnom formáte podľa volieb používateľa.

EMAIL (3) – Emailová adresa v štandardnom formáte.

PHONE (4) – Telefónne číslo. Je doporučený medzinárodný formát začínajúci znakom „+“.

LOC (5) – Geografická lokácia odosielateľa streamu. V rôznych podmienkach je možné očakávať rôzne stupne detailnosti informácie (napr. číslo izby a poschodie alebo mesto a štát).

TOOL (6) – Názov a prípadne verzia nástroja generujúceho stream.

NOTE (7) – Položka, ktorej obsah môže definovať profil. Nemala by byť obsahom každého paketu RTCP z dôvodu narastajúceho množstva odosielaných dát, ale mala by byť používaná pri výskyte jednorázových udalostí.

PRIV (8) – Položka používaná pre aplikačne špecifické rozšírenia SDES paketov. Jej formát je mierne odlišný: Za 8-bitovou hodnotou dĺžky nasleduje 8-bitová dĺžka prefixu v zhodnom formáte, za ňou nasleduje samotný prefix, nasledovaný hodnotou. Prefix slúži pre označenie určené osobou definujúcou obsah položky.

2.3.4 BYE

Paket zasielaný pri opustení streamu niektorým z účastníkov.

Paket v spoločnej časti v položke `Packet Type` obsahuje hodnotu 203 a v položke `Type Specific` počet identifikátorov SSRC alebo CSRC opúšťajúcich sedenie. Za ním nasledujú samotné identifikátory. Poslednou časťou paketu je voliteľný dôvod odchodu zo sedenia, pred ktorým ešte nasleduje 8-bitová hodnota dĺžky.

2.3.5 APP

Paket prenášajúci informácie špecifické pre aplikáciu. Slúžia pre možnosť implementácie vlastností špecifických pre konkrétne aplikácie.

Jeho hodnota v položke `Packet Type` je 204. V položke `Type Specific` obsahuje podtyp paketu umožňujúci zhlukovanie viacerých typov paketov pod jedno meno. Za povinnou časťou nasleduje identifikátor SSRC alebo CSRC, za ktorým nasleduje meno zvolené autorom aplikácie (32-bitov) a ako posledné sú obsiahnuté samotné aplikačné dáta.

2.4 RTP profil pre audio a video konferencie s minimálnou kontrolou

Tento profil taktiež často označovaný ako RTP Audio Video Profil (RTP/AVP), bol prvý krát definovaný v dokumente [3], ktorý bol nahradený dokumentom [4]. Profil sa v súčastnosti používa na prenos ľubovolného audio a video obsahu, či sa už jedná o použitie v oblasti VoIP, videokonferencií, alebo streamingu videa po počítačovej sieti.

Profil žiadnym spôsobom neovplyvňuje stavbu RTP hlavičky a samotný nepoužíva jej rozšírenie. Aplikácie však nesmú ignorovať bit `extension` a musia byť prípadne pripravené rozšírenie hlavičky ignorovať.

Tento profil podporuje viacero druhov kódovaní samostatného audia, videa, alebo kombinácie oboch. Ich mapovanie na položku `payload type (PT)` v hlavičke RTP paketu môže byť buď statické alebo dynamické. Statické čísla `PT` pre audio kódeky sú uvedené v tabuľke 2.1 a pre video a kombinované kódeky v tabuľke 2.2. Každému kódeku je pridelené aj krátke meno (encoding name), ktoré môže byť použité v signalizačných protokoloch (napr. SDP). Ďalšie kódeky môžu byť staticky mapované po registrácii v organizácií IANA (Internet Assigned Numbers Authority), z dôvodu zachovania unikátneho mapovania pre jednotlivé kódeky. Pri použití dynamického mapovania kódeku sa použije jedna z hodnôt v rozsahu 96 – 127 a o mapovanie konkrétneho kódeku sa postará jeden z použitých signalizačných protokolov.

Pre potreby overenia súčasnej reálnej situácie bol pre potreby tejto práce vypracovaný výskum, ktorý skúmal podporu kódekov v niekoľkých aplikáciách používajúcich prenos cez RTP. Ako vzorka boli vybraté softvérové telefóny Ekiga, SJphone a X-Lite, ktoré sú dostupné zadarmo, hardwarového telefónu Well SIP-T20, domáceho smerovača s VoIP bránou Linksys WRP400-G2 a video-konferenčného softvéru Polycom PVX.

Výsledné zistenia týkajúce sa softvéru Ekiga sú zaznamenané v tabuľke 2.3. Je v nich možné zaznamenať, že v rámci tohoto softvéru sa hodnoty `payload type`, ktoré sú prideľované dynamicky rozdielne aj medzi verziou pre operačný systém Windows a Linux. Dodržuje však všetky hodnoty statického mapovania v oboch týchto verziách.

Z výsledkov získaných z programu X-Lite a zaznamenaných v tabuľke 2.4, je vidieť, že napriek väčšiemu množstvu kódekov zhodných s predchádzajúcim programom, nie je

zhodná ani jedna hodnota pridelená kódekom používajúcim dynamické mapovanie. Avšak aj tento program dodržuje všetky hodnoty používané pri statickom mapovaní.

Výsledky získané z programu SJphone sú zaznamenané v tabuľke 2.5. V tomto prípade sa jedna hodnota `payload type` pre kódek iLBC zhodovala s hodnotou, ktorú používal softvér X-Lite. Keďže SJphone používa aj iné čísla pre dynamicky mapované typy obsahov, ktoré sa opäť nezhodujú so žiadnym z predchádzajúcich nástrojov môžeme predpokladať, že toto je spôsobené náhodou a teda toto nie je pravidlom. Všetky hodnoty použité pre kódeky používajúce statické mapovanie ostali opäť zachované.

Hardvérový telefón Well SIP-T20, používal iba kódeky so statickým mapovaním. Ich číslovanie ostalo zachované. Výsledky sú zaznamenané v tabuľke 2.6.

Domáci smerovač s integrovanou VoIP bránou Linksys WRP400-G2, bol v tomto teste zvláštnosťou. V menu pre konfiguráciu umožňoval zmenu čísel používaných pre kódeky s dynamickým mapovaním. V tabuľke 2.7 sú uvedené preto hodnoty zistené zo signalizačného protokolu SIP, pri východzích nastaveniach zariadenia a kódeky ktoré používali statické mapovanie. V prípade týchto sa číslovanie v menu meniť nedalo a ostalo zachované v porovnaní s ostatnými nástrojmi. Toto zariadenie taktiež podporovalo ako jediné ďalší protokol pre doručovanie udalostí a to Cisco NSE a zapuzdrené RTP.

Videokonferenčný softvér Polycom PVX rovnako ako predchádzajúce nástroje dodržiava hodnoty pre statické mapovanie, avšak opäť nenastala žiadna zhoda, týkajúca sa čísel používaných pre dynamické mapovanie kódekov. Výsledky sú zaznamenané v tabuľke 2.8.

Podľa výsledkov je zjavné, že všetky aplikácie striktne dodržujú statické mapovanie. U dynamického mapovania sa nemožno spoliehať na žiadne ustálené hodnoty. Tieto sa značne líšili nielen medzi aplikáciami samotnými, ale napríklad aj medzi vydaniami pre operačný systém Linux a Windows s rovnakým číslom verzie (Ekiga). Jedinou výnimkou bol typ obsahu označovaný ako `telephone-event`, slúžiaci na prenos tónovej voľby využívanej analógovými telefónmi, podľa normy DTMF. Tento typ bol vo všetkých programoch mapovaný na číslo 101.

Pri samotnej detekcii tieto zistenia môžu byť užitočné, pretože môžeme prehlásiť, že statické mapovanie sa v praxi skutočne dodržuje, ale u dynamického mapovania sa nemôžeme spoľahnúť na žiadne konkrétne kódeky u niektorých hodnôt. Tieto zistenia sa použijú pre mapovanie hodnôt pre určenie kódeku u detekovaných streamov.

2.5 Použitie

Dnes sa RTP a RTCP používa primárne na prenos audia a videa cez siete postavené na modeli TCP/IP a to pomocou profilu RTP/AVP. V oblasti VoIP je to primárne využívaný prostriedok na prenos samotného hlasu cez sieť. Tak isto je využívaný pri potrebe real-time streamovania videa najmä v oblasti videokonferencií, kde momentálne najviac využívaný štandard H.323 využíva RTP na prenos audio a video dát.

Existuje však aj mnoho aplikácií prenášajúcich audio a video cez sieť inými protokolmi. Medzi takéto patrí napríklad program Skype, ktorý využíva vlastný, uzavrený protokol. Je to hlavne z toho dôvodu, že Skype nemusí dodržiavať kompatibilitu medzi viacerými aplikáciami a vývojári mohli použiť protokol „ušitý na mieru“ ich potrebám.

Ďalším príkladom môžu byť v dnešnej dobe veľmi populárne a využívané video-servery typu `www.youtube.com`. Tie väčšinou používajú na prenos protokol HTTP. Tento bol zvolený z dvoch hlavných dôvodov. Prvým z nich je, že dáta nevznikajú v reálnom čase a môžu byť čo najrýchlejšie a v čo najväčšej kvalite prenesené k príjemcovi, kde sú po dobu prehrávania uložené v cache pamäti. Druhým z nich je potreba dostupnosti služby čo najväčšej

skupine ľudí. Ak sa užívatelia dostanú k stránke znamená to, že bezpečnostná politika siete povoľuje komunikáciu protokolom HTTP a teda aplikácia nebude mať problém preniesť video.

2.6 Rozpoznávanie

Keďže RTP väčšinou využíva služby transportného protokolu UDP, ktorý neposkytuje informácie o protokole vyššej vrstvy, ktorého dáta nesie, nedajú sa RTP dáta triviálne odhaliť. RTP hlavička neposkytuje ani dostatok stabilných hodnôt aby sa o pakete mohlo prehlásiť, že nesie RTP dáta jednoduchou analýzou hlavičky. RTP tak isto nepoužíva pevne definované čísla portov, čiže táto metóda je rovnako nepoužiteľná.

Pre toto väčšina aplikácií rozpoznáva RTP pomocou informácií nesených v signalizačných protokoloch (napríklad program Wireshark) alebo nie sú schopné RTP rozpoznať vôbec (väčšina jednoduchých firewallov).

Ďalšou možnosťou rozpoznania RTP, pomocou pokročilej analýzy sa zaoberá táto práca. Existujú však už aj nástroje, ktoré toto dokážu. Príkladom takýchto aplikácií môže byť program Microsoft Network Monitor (dostupný na: http://www.microsoft.com/en-us/download/details.aspx?id=4865), ktorý ale funguje na uzavretej platforme a teda sa nedá využiť v ďalších projektoch. Tento pracuje na princípe postupného filtrovania na základe vlastností jednotlivých paketov. Pri RTP tak kontroluje napríklad verziu protokolu alebo čísla používaných portov. Niektoré z vlastností kontrolovaných týmto programom budú kontrolované aj v algoritme ktorým sa zoberá táto práca. Avšak niektoré ako napríklad kontrola či sa pri RTP používajú len párne porty z dôvodu, že reálne streamy zachytených dát toto v niektorých prípadoch porušovali. Výsledky od tohoto nástroja sa často krát výrazne líšia od informácií získaných pomocou signalizačných protokolov. Toto bolo spôsobené pravdepodobne práve princípom detekcie, ktorý skúmal len pakety jednotlivo, bez prihliadnutia na charakter dát, teda väčšieho množstva paketov s opakujúcimi sa niektorými vlastnosťami (synchronizačný zdroj, adresy, porty).

V novších verziách programu Wireshark sa objavila pri nastaveniach protokolu RTP možnosť detekcie tohoto protokolu aj mimo konverzácie indikovanej niektorým zo signalizačných protokolov. Po zapnutí tejto možnosti sa však môže objaviť pomerne veľké množstvo falošne identifikovaných paketov, ktoré budú označené ako poškodené. Tento program tak isto nebral do úvahy samotný charakter dát.

PT	encoding name	clock rate (Hz)
0	PCMA	8 000
1	reserved	
2	reserved	
3	GSM	8 000
4	G723	8 000
5	DVI4	8 000
6	DVI4	16 000
7	LPC	8 000
8	PCMA	8 000
9	G722	8 000
10	L16	44 100
11	L16	44 100
12	QCELP	8 000
13	CN	8 000
14	MPA	90 000
15	G728	8 000
16	DVI4	11 025
17	DVI4	22 050
18	G729	8 000
19	reserved	
20	unassigned	
21	unassigned	
22	unassigned	
23	unassigned	
dyn	G726-40	8 000
dyn	G726-32	8 000
dyn	G726-24	8 000
dyn	G726-16	8 000
dyn	G729D	8 000
dyn	G729E	8 000
dyn	GSM-EFR	8 000
dyn	L8	var.
dyn	RED	
dyn	VDVI	var.

Tabuľka 2.1: Zodpovedajúce hodnoty `payload type` pre audio kódeky. [4]

PT	encoding name	media type	clock rate (Hz)
24	unassigned	V	
25	CelB	V	90 000
26	JPEG	V	90 000
27	unassigned	V	
28	nv	V	90 000
29	unassigned	V	
30	unassigned	V	
31	H261	V	90 000
32	MPV	V	90 000
33	MP2T	AV	90 000
34	H263	V	90 000
35 – 71	unassigned	?	
72 – 76	reserved	N/A	N/A
77 – 95	unassigned	?	
96 – 127	dynamic	?	
dyn	H263-1998	V	90 000

Tabuľka 2.2: Zodpovedajúce hodnoty `payload type` pre video a kombinované kódeky. [4]

encoding name	PT (Windows)	PT (Linux)	media type	clock rate (Hz)
Speex	125	113	A	16 000
iLBC	110	116	A	8 000
PCMU	0	0	A	8 000
PCMA	8	8	A	8 000
GSM	3	3	A	8 000
CELT	95	115	A	48 000
CELT	99	114	A	32 000
G722	9	9	A	16 000
Speex	124	112	A	8 000
G726-16	105	122	A	8 000
G726-24	104	121	A	8 000
G726-32	103	120	A	8 000
G726-40	102	119	A	8 000
MS-GSM	106	123	A	8 000
h264	109	N/A	V	90 000
theora	126	99	V	90 000
h261	31	31	V	90 000
h263	34	N/A	V	90 000
h263-1998	108	N/A	V	90 000
MP4V-ES	114	N/A	V	90 000
telephone-event	101	101	-	8 000

Tabuľka 2.3: Kódeky podporované programom Ekiga a zodpovedajúce čísla pre `payload type`

encoding name	PT	media type	clock rate (Hz)
BV32	107	A	16 000
BV32-FEC	119	A	16 000
DVI4	5	A	8 000
DVI4	6	A	16 000
PCMA	8	A	8 000
PCMU	0	A	8 000
GSM	3	A	8 000
iLBC	98	A	8 000
L16	102	A	16 000
SPEEX	97	A	8 000
SPEEX-FEC	105	A	8 000
SPEEX	100	A	16 000
SPEEX-FEC	106	A	16 000
h263	34	V	90 000
h263-1998	115	V	90 000
telephone-event	101	-	8 000

Tabuľka 2.4: Kódeky podporované programom X-Lite a zodpovedajúce čísla pre `payload type`

encoding name	PT	media type	clock rate (Hz)
GSM	3	A	8 000
iLBC	97	A	8 000
iLBC	98	A	8 000
speex	110	A	8 000
PCMA	8	A	8 000
PCMU	0	A	8 000
telephone-event	101	-	8 000

Tabuľka 2.5: Kódeky podporované programom SJphone a zodpovedajúce čísla pre `payload type`

encoding name	PT	media type	clock rate (Hz)
PCMA	8	A	8 000
PCMU	0	A	8 000
G729	18	A	8 000
G722	9	A	8 000
telephone-event	101	-	8 000

Tabuľka 2.6: Kódeky podporované hardvérovým telefónom WELL SIP-T20 a zodpovedajúce čísla pre `payload type`

encoding name	PT	media type	clock rate (Hz)
G726-32	98	A	8 000
G729ab	99	A	8 000
PCMA	8	A	8 000
PCMU	0	A	8 000
telephone-event	113	-	8 000
encaprtp	100	-	8 000
telephone-event	101	-	8 000

Tabuľka 2.7: Kódeky podporované routerom Linksys WRP400-G2 a zodpovedajúce čísla pre `payload type`

encoding name	PT	media type	clock rate (Hz)
SIREN14	99	A	16 000
SIREN14	98	A	16 000
SIREN14	97	A	16 000
G7221	102	A	16 000
G7221	101	A	16 000
G7221	103	A	16 000
G722	9	A	8 000
G728	15	A	8 000
G729	18	A	8 000
PCMU	0	A	8 000
PCMA	8	A	8 000
h264	109	V	90 000
h263	34	V	90 000
h263-1998	96	V	90 000
h261	31	V	90 000
telephone-event	101	-	8 000

Tabuľka 2.8: Kódeky podporované programom Polycom PVX a zodpovedajúce čísla pre `payload type`

Kapitola 3

Riešenie detekcie streamov

3.1 Detekcia RTP

Pre RTP pakety existuje viacero kritérií, ktoré musia spĺňať. Samotná detekcia bude prebiehať na princípe postupného filtrovania nevyhovujúcej komunikácie. Filtrovať sa bude na základe niektorých hodnôt a tým, či súhlasia so skutočným stavom paketu. Pri filtrácií sa využijú všetky položky z RTP hlavičky okrem položiek `sequence number` a `timestamp`.

V prvom kroku bude vylúčená komunikácia, ktorá podľa nasledujúcich kritérií s najväčšou pravdepodobnosťou nenesie RTP dáta.

- Ako nosný protokol na transportnej vrstve sa v prevažnej väčšine prípadov používa protokol UDP, teda všetky pakety používajúce ostatné transportné protokoly budú z ďalšieho spracovania vylúčené.
- Keďže porty používané pri RTP komunikácii sú prideľované dynamicky, môžeme predpokladať, že komunikácia nebude používať tzv. „well-known“ porty, teda porty v rozsahu $0-1023$. Ak bude komunikácia prebiehať na niektorom z týchto portov, môže byť taktiež vylúčená.

Následne sa dáta nesené protokolom UDP analyzujú ako RTP, pričom sa skontroluje dĺžka paketov a vylúčia sa tie, ktorých dĺžka je nedostačujúca na vystavanie RTP hlavičky, vrátane identifikátorov CSRC, ktorých počet je uvedený v položke `CC`.

V treťom kroku sa postupne vylúčia pakety, ktoré nevyhovujú nasledujúcim hodnotám v RTP hlavičke a skutočnému stavu paketu:

- V položke `version` v hlavičke RTP sa musí nachádzať hodnota 2. V opačnom prípade je isté, že pakety nepatria k RTP komunikácií verzie 2 podľa [6].
- Následne sa kontroluje hodnota v položke `payload_type`. Pri tejto kontrole môžu nastať dva prípady:
  - Ak užívateľ požaduje detekciu len niektorých typov obsahu, porovná sa hodnota v pakete so zoznamom požadovaných typov a to na základe číselnej hodnoty, ale aj reťazca s názvom. V prípade, že sa typ obsahu zhoduje s jedným z požadovaných typov, ktoré zadal užívateľ, pokračuje sa v spracovávaní paketu. V opačnom prípade sa paket vylúči.
  - V prípade že nebol zadaný žiadny špecifický typ obsahu pre detekciu, skontroluje sa či typ obsahu nezodpovedá jednej z rezervovaných hodnôt podľa [4]. Ak typ obsahu zodpovedá jednej z týchto hodnôt, bude z ďalšieho spracovania vylúčený.

- Ak je nastavený bit `extension`, z nesených dát sa vystavia rozšírená hlavička, pričom sa skontroluje, či je dĺžka dostatočná pre vystavanie podla položky `length`. Ak je bit nastavený, ale hlavičku sa nepodarí vystavať, paket sa vylúči zo spracovania.
- Ak je nastavený bit `padding`, skontroluje sa posledný bajt dát. V prípade, že hodnota nesená týmto bajtom je vyššia ako dĺžka nesených dát, paket sa opäť vylúči zo spracovania.

Ak paket vyhovuje všetkým vyššie zmieneným kritériám, môžeme prehlásiť, že sa pravdepodobne jedná o RTP paket a teda ho môžeme zaradiť do kolekcie paketov, ktoré budú tvoriť stream. Tento algoritmus je vyobrazený na obrázku 3.1. Stream bude identifikovaný zdrojovou a cieľovou IP adresou, zdrojovým a cieľovým portom, spolu s identifikátorom synchronizačného zdroja a typom obsahu. Týmto sa zabezpečí spoľahlivá separácia jednotlivých streamov. Stream, ktorý bude počas detekcie meniť typ obsahu, bude rozdelený na viacero streamov, každý s jedným typom obsahu. Toto umožní taktiež sledovať jednotlivé zmeny.

Po ukončení spracovávania jednotlivých paketov sa zo zoznamu detekovaných streamov odstránia tie, ktoré obsahovali príliš malé množstvá paketov (tento parameter bude nastaviteľný užívateľom). U tohto kroku sa predpokladá, že na jeho základe bude možné odfiltrovať veľké množstvo falošne rozpoznaných paketov.

V priebehu testovania, bol identifikovaný problém s použitím kontroly bajtov slúžiacich pre zarovnanie v prípade, že je nastavený bit `padding`. Problém je popísaný v kapitole 5.2.1. Ako riešenie bolo zvolené pridanie možnosti vypnutia tejto kontroly. V prípade jej vypnutia budú správne detekované všetky pakety, avšak zvyšuje sa aj riziko falošnej detekcie RTP paketov.

3.2 Detekcia RTCP

Pri detekcii RTCP paketov môžeme vychádzať z viacerých skutočností a teda je možné očakávať, že detekcia bude dosahovať oveľa vyššej úspešnosti ako detekcia RTP. Pri detekcii môžme zanedbať možnosť výskytu 32-bitového čísla na začiatku zloženého paketu, keďže toto sa môže vyskytovať len pri použití šifrovania a tým pádom je detekcia znemožnená.

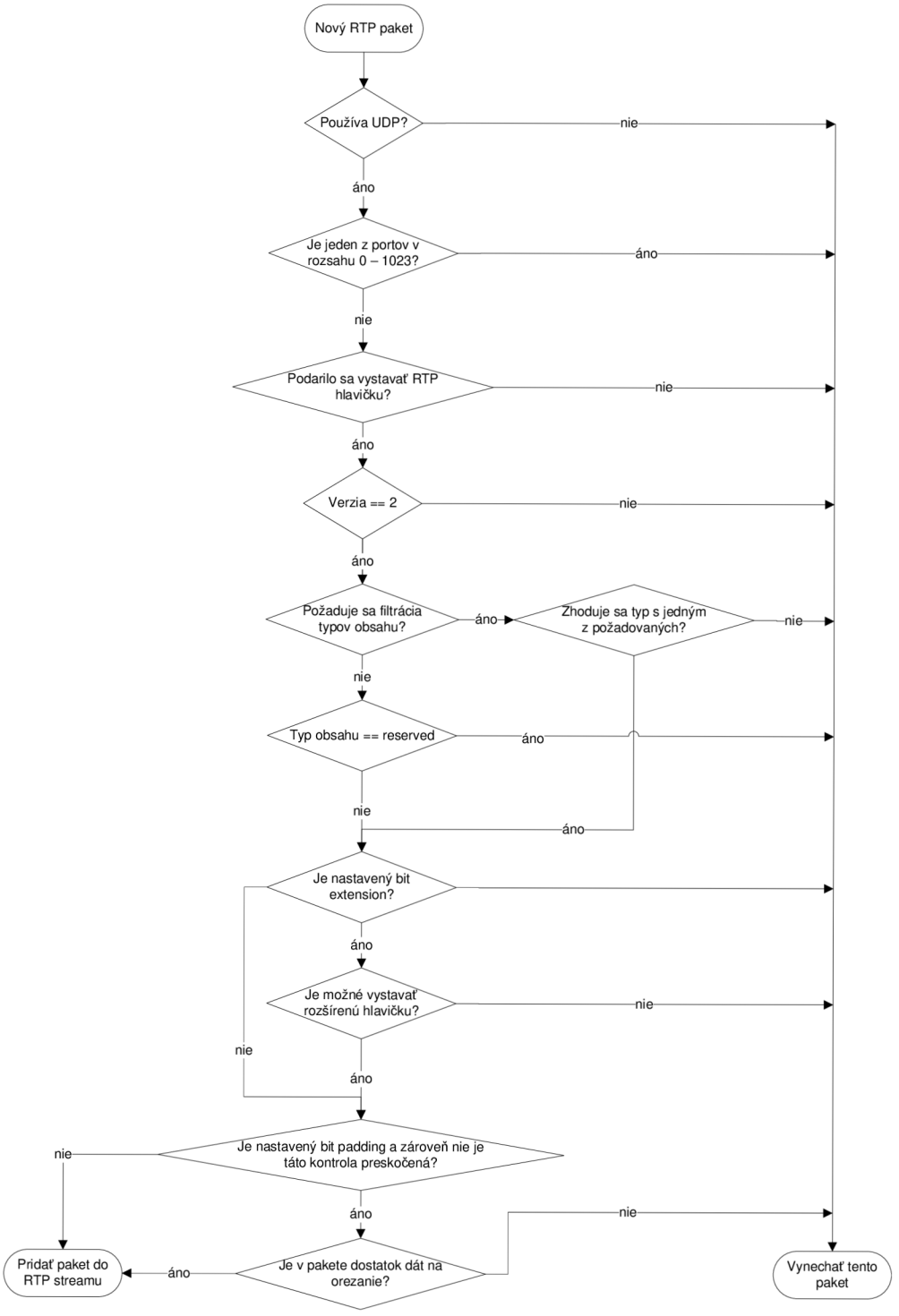
Detekcia samotná je založená na rovnakom princípe, teda postupnom filtrovaní paketov, ako detekcia RTP paketov.

V prvom kroku sa rovnako ako pri RTP skontroluje či sa používa protokol UDP a čísla používaných portov. Následne sa prvých pár bajtov paketu analyzuje ako spoločná časť hlavičky paketu, skontroluje sa dĺžka paketu a verzia udávaná v RTCP hlavičke. Ako posledná sa pred pokusom o dekompozíciu paketu skontroluje typ paketu v prípade, že sa jedná o typ Sender Report alebo Reciever Report, nasleduje dekompozícia zloženého paketu v prípade, že je typ iný dáta sa zahodia, keďže ako prvý sa v zloženom pakete musí nachádzať jeden z týchto typov.

Počas samotnej dekompozície paketu sa prevedú ešte kontroly na nenastavený bit `padding` v hlavičkách jednotlivých paketov, ktorý smie mať nastavený len posledný paket. Ak je tento u posledného paketu nastavený, skontroluje sa, či má paket dostatok dát na orezanie.

Ak po orezaní niektorého z paketov nastane situácia, že za paketom nasledujú ešte dáta, ktoré sú ale nedostatočné pre zostavenie spoločnej hlavičky, alebo po zostavení hlavičky nie je dostatok dát na zostavenie celého paketu, celý zložený paket sa zahodí.

Ak všetky kontroly prebehli v poriadku, môžeme paket prehlásiť za RTCP a jednotlivé RTCP pakety zo zloženého paketu sa môžu odovzdať k ďalšiemu spracovaniu.



Obr. 3.1: Algoritmus pre detekciu jednotlivých RTP paketov

Kapitola 4

Testovacia aplikácia

Aplikácia je implementovaná v jazyku Python 2.7, s použitím open-source knižnice Scapy (dostupné na: http://www.secdev.org/projects/scapy/). Skladá sa z dvoch modulov: `rtpdetector.py` a `rtcpdetector.py`, ktoré sú použiteľné ako samostatné aplikácie ale aj ako moduly pre využitie v ďalších aplikáciach. Ako príklad bude slúžiť aplikácia `rtpinfo.py`, ktorá bude využívať funkčnosť oboch modulov. Týmto riešením sa síce zvýši celková náročnosť aplikácie, keďže časť detekcie je založená na kontrole rovnakých vlastností paketu (kontrola UDP, číslo verzie v hlavičke), ale získame tým možnosť použiť detekciu oboch protokolov nezávisle na sebe. Naviac primárne určenie aplikácie bude skenovanie odchytených dát vo formáte pcap a teda výkon nie je podstatný pre túto aplikáciu. Oba moduly aj výsledná aplikácia budú schopné detekcie RTP a RTCP paketov používajúcich ako protokol IP verzie 4, tak aj novší IP verzie 6.

4.1 Modul rtpdetector.py

Modul implementuje detekciu založenú na princípe popísanom v kapitole 3.1. V module sú implementované triedy `RTPdetector` a `RTPstream`. Modul je možné použiť pre využitie v ďalších aplikáciách pomocou objektu triedy `RTPdetector`, alebo po spustení pracuje ako samostatný detektor RTP streamov.

4.1.1 Trieda RTPStreamProperties

Trieda `RTPStreamProperties` je len kontajnerom pre údaje o RTP streame, ale pre možnosť použiť ju ako kľúč pre slovník v jazyku Python. Je upravená ako nemeniteľná, a teda údaje o streame je možné zadať len pri vzniku objektu tejto triedy a neskôr ich nie je možné meniť.

Ďalej táto trieda implementuje len povinné súkromné metódy pre použitie ako kľúč v slovníku.

4.1.2 Trieda RTPStream

Táto trieda slúži pre ukladanie informácií o jednotlivých streamoch ako napríklad: IP adresy, porty, ID synchronizačného zdroja ale aj počet a čísla paketov, ktoré niesli dáta tohto streamu. Stará sa však aj o ukladanie obsahu neseného jednotlivými RTP paketami a poskytuje rozhranie pre neskoršie získanie všetkých týchto údajov o streame.

Pri vzniku objektu sa konštruktoru zadáva iba parameter `store`, ktorý označuje či bude počas detekcie ukladaný aj obsah nesený jednotlivými paketmi.

Rozhranie tejto triedy poskytuje prostriedky pre postupné pridávanie jednotlivých paketov do streamu pomocou metódy `add_packet(packet)` a získanie jednotlivých informácií o streame pomocou metód `properties()` pre získanie údajov o streame uchovaných v objekte triedy `RTPStreamProperties`, `packet_nums()` pre získanie čísel paketov, ktoré niesli dáta streamu a `payload()` pre získanie dát nesených RTP streamom.

4.1.3 Trieda RTPdetector

Samotná detekcia RTP streamov je implementovaná v tejto triede. Postup použitý pri detekcii je popísaný v kapitole 3.1.

Pri vytváraní inštancie objektu je možné konštruktoru triedy poskytnúť viacero parametrov: `min_packet` pre nastavenie minimálneho počtu paketov v streame, `num_profile` pre nastavenie filtrovania pomocou čísel typov obsahu uvedenych v tabuľkách 2.1 a 2.2, `str_profile` pre nastavenie filtrovania pomocou krátkych názvov typov obsahu definovaných v tabuľkách 2.1 a 2.2 a `store_payload` pre nastavenie ukladania obsahu neseného jednotlivými streamami. V prípade, že niektoré z týchto nastavení nebude konštruktoru predané, použije sa východzia hodnota. V prípade potreby je možné kedykoľvek v priebehu detekcie toto nastavenie zmeniť pomocou metódy `set_params(min_packet, num_profile, str_profile, store_payload)`. Ak nebude niektoré z nastavení funkcii predané, ostane toto na predchádzajúcej hodnote.

Po vytvorení inštancie objektu tejto triedy, je možné začať so spracovávaním jednotlivých paketov. Toto sa prevádza pomocou metódy `process_packet(packet)`, ktorej sa ako argument predá paket získaný, alebo vytvorený pomocou knižnice Scapy. Po vykonaní samotnej detekcie je potrebné zavolať metódu `clean_streams()`, ktorá odstráni streamy, ktoré obsahujú menší počet paketov ako ten, ktorý bol zadaný pomocou predchádzajúcej metódy (prípadne predvolenej hodnoty 100 paketov).

Výsledky detekcie (aj priebežné) je možné získať pomocou metódy `detected_streams()`, ktorá vracia list objektov triedy `RTPStream`, s informáciami o všetkých detekovaných streamoch. V prípade potreby získania počtu všetkých spracovaných paketov je možné použiť metódu `packet_processed()`.

4.2 Modul RTCPdetector.py

Tento modul implementuje detekciu RTCP založenú na princípoch uvedených v kapitole 3.2. Modul je tak isto ako predchádzajúci možné použiť v ďalších aplikáciách, avšak po spustení funguje aj ako samostatná aplikácia. Tento modul implementuje triedy `RTPCDetector` a `RTPCInfo`.

4.2.1 Trieda RTCPInfo

Táto trieda slúži pre ukladanie informácií zistených o jednotlivých subjektoch komunikujúcich pomocou protokolu RTP, zistených z paketov protokolu RTCP.

Po vytvorení objektu tejto triedy, sa údaje môžu kedykoľvek aktualizovať pomocou metód `act_info(cname, name, email, phone, loc, tool, note)`, ktorej sa predávajú jednotlivé položky zasielané v paketoch typu `source description` a metódy `act_send(packets,`

`bytes)`, ktorá aktualizuje počítadlá zaslaných paketov a bajtov daným odosielateľom. V prípade, že sa ktorejkoľvek z metód nepredajú všetky parametre, vynechané nebudú aktualizované, ale budú ponechané na predchádzajúcej hodnote.

4.2.2 Trieda RTCPDetector

Táto trieda implementuje samotnú funkčnosť detekcie paketov a ich následnú dekompozíciu zo zložených paketov na jednotlivé RTCP pakety.

4.3 Aplikácia rtpinfo.py

Táto aplikácia demonštruje kombinované použitie oboch týchto modulov. Väčšina volieb fungujúca pri spúštaní (rtpinfo.py), je rovnaká aj pri spúštaní jedného z modulov. Popis jednotlivých volieb je možné zobraziť aj pomocou voľby (–help) alebo (-h).

Aplikácia skúma postupne jednotlivé pakety pomocou vyššie spomínaných modulov. Najprv použije modul `RTPDetector` a v prípade, že ten vyhlási že paket nepatrí do RTP streamu, predá sa modulu `RTCP detector`. Na konci detekcie sa mapujú jednotlivé získané informácie z RTCP paketov na RTP streamy podľa identifikačného čísla synchronizačného zdroja.

4.3.1 Použitie aplikácie

Požadované parametre filtrovania a ďalšie potrebné nastavenia sa budú aplikácii predávať pomocou parametrov príkazového riadku.

Možné nastavenia behu aplikácie budú nasledovné:

--help (-h) – Pomocou tejto voľby je možné zobraziť užívateľskú nápovedu.

--file (-f) – Voľba pre filtrovanie paketov zo súboru so zachytenou sieťovou prevádzkou vo formáte pcap. Názov súboru musí nasledovať bezprostredne za voľbou. Nemôže sa kombinovať s voľbou (--interface).

--interface (-i) – Voľba slúži pre nastavenie režimu odchytávania paketov zo sieťového rozhrania. Názov rozhrania musí nasledovať bezprostredne za voľbou. Nemôže sa kombinovať s voľbou (--file).

--min_packets (-m) – Minimálny počet paketov v jednom RTP streame. Pakety budú jednoznačne identifikované ako RTP, až keď ich počet v jednom streame prekročí zadanú hodnotu. Hodnota musí nasledovať bezprostredne za voľbou. Ak sa voľba nepoužije, bude použitá východzia hodnota 100 paketov.

--num_profile (-n) – Číslo typu obsahu pre RTP/AVP podľa [4], ktoré majú byť detekované. Táto voľba môže byť použitá viacnásobne. Číslo musí nasledovať bezprostredne za voľbou.

--str_profile (-s) – Názov typu obsahu pre RTP/AVP podľa [4], ktoré majú byť detekované. Táto voľba môže byť použitá viacnásobne. Názov musí nasledovať bezprostredne za voľbou. Pri zadávaní záleží na veľkosti písmen.

--save_payload (-p) – Uloží binárnu kópiu obsahu neseného RTP streamami do súborov. Súbory sa umiestnia do priečinku, ktorého názov musí nasledovať bezprostredne za voľbou. V prípade kontroly zarovnávajúcich bajtov sa tieto do súboru neukladajú.

--packets_list (-l) – Vypíše do súboru čísla paketov a číslo streamu, ku ktorému paket patrí. Názov súboru, do ktorého sa majú tieto informácie vypísať, musí nasledovať bezprostredne za voľbou.

--no_padding_check (-c) – Vynechá kontrolu zarovnávajúcich dát. Potreba pridania tejto voľby bola zistená pri testoch a je podrobnejšie popísaná v kapitole 5.2.1.

Jediným povinným parametrom bude voľba zdroju paketov, tj. jeden z parametrov (--interface) alebo (--file).

Voľby `--num_profile` a `--str_profile` sa môžu ľubovolne kombinovať a zadávať viacnásobne. Pri zadaní viacerých typov obsahov sa zobrazia všetky zadané profily. Ak sa nepoužije ani jedna z volieb, budú zobrazované všetky streamy.

Beh aplikácie bude ukončený v prípade použitia voľby (--file) automaticky po prečítaní všetkých paketov zo vstupného súboru, alebo v prípade použitia voľby (--interface) po prijatí signálu (SIGINT).

Príklady spustenia:

./rtpinfo.py –interface eth0 -m 20 -s PCMA -n 3 – detekcia na rozhraní s názvom eth0, minimálny počet paketov v streame je 20, detekujú sa iba kódeky PCMA a GSM

./rtpinfo.py -i wlan0 -s PCMA -s PCMU – detekcia na rozhraní s názvom wlan0, detekujú sa iba kódeky PCMA a PCMU

./rtpinfo.py -f subor.pcap –no_padding_check – detekcia zo súboru subor.pcap, vynecháva sa kontrola zarovnávajúcich bajtov

./rtpinfo.py -f subor.pcap -l list.txt – detekcia zo súboru subor.pcap, do súboru list.txt sa uloží zoznam RTP paketov

4.3.2 Výstup aplikácie

Aplikácia bude všetky varovania a chyby vypisovať na štandardný chybový výstup. Počas prebiehajúcej detekcie sa na štandardný výstup nebudú vypisovať žiadne údaje. Po ukončení detekcie sa na štandardný výstup vypíšu údaje o identifikovaných streamoch v nasledujúcom formáte:

```
==== Stream 1 ====
Src IP:   147.229.14.24
Dst IP:   147.229.252.55
Src port: 49154
Dst port: 18256
SSRC:     0x38f0af78
Payload:  PCMA (8)
Packets:  2534
==== Stream 2 ====
Src IP:   147.229.252.55
```

```
Dst IP:   147.229.14.24
Src port: 18256
Dst port: 49154
SSRC:     0x1137fc37
Payload:  PCMA (8)
Packets:  2531
RTCP Info:
    CNAME: 0.0.0@147.229.252.55
    NAME:  Cisco IOS, VoIP Gateway
    TOOL:  Cisco IOS, VoIP Gateway
    Packets send:  2330
    Bytes send:    372800

===== RTP =====
RTP:        5065
Other:      236
===== RTCP ======
SR:    10
RR:    1
SDES:  11
BYE:   1
APP:   0
```

Na začiatku sa vypisuje označenie začiatku informácií o streame spolu s poradovým číslom streamu. Streamy sa zoraďujú podľa poradia zachytenia prvého RTP paketu patriaceho tomuto streamu. Nasledujú informácie o zdrojovej a cieľovej IP adrese, o zdrojovom a cieľovom porte, identifikačnom čísle synchronizačného zdroja, použitom kódeku (podľa čísla uvedenom v položke `payload type`) a počtu paketov, ktoré patria do tohto streamu.

V prípade dostupnosti RTCP informácií pre daný synchronizačný zdroj, následne sa vypíšu informácie zozbierané z RTCP paketov. Sú to informácie z paketov typu `source description`, okrem typu `APP` a informácií o počte zaslaných paketov a zaslaných bajtov hlásených zdrojom.

Po výpise všetkých detekovaných paketov, nasleduje zhrnutie výsledkov pre RTP, kde je uvedený počet paketov identifikovaných ako RTP a počet všetkých ostatných paketov (vrátane RTCP). Pre RTCP sú to počty jednotlivých typov paketov po rozdelení zložených paketov.

V prípade použitia voľby `--packets_list` sa do zvoleného súboru zapíše zoznam paketov vo formáte:

```
198,2
199,1
200,2
201,0
202,1
203,2
```

Prvé číslo zodpovedá poradovému číslu paketu v súbore pcap, prípadne poradové číslo zachyteného paketu a rovná sa číslu zobrazovanému programom Wireshark. Druhé číslo zodpovedá číslu streamu, tak ako je priradené streamu v informáciách, ktoré sa vytlačia na

štandardný výstup. V prípade, že je na tomto mieste uvedená 0, paket nebol detekovaný ako RTP a teda nepatrí k žiadnemu streamu. Do tohto zoznamu sa zapisujú všetky spracované pakety.

Kapitola 5

Testovanie na dátach

5.1 Metodika testovania

Pre testovanie aplikácie sa použije notebook s procesorom Intel Core 2 Duo P8700 (2,53GHz), s nainštalovaným operačným systémom Gentoo Linux (jadro 3.3.3), interpretom Python 2.7.2 a knižnicou Scapy 2.2.0.

Testy sa budú prevádzať na aplikácii `rtpinfo.py`, ktorá využíva funkčnosť oboch modulov. V rámci testovania sa budú porovnávať výsledky detekcie nástrojov používajúcich signalizáciu zastúpených programom Wireshark, na dátach odchytených pri použití rôznych RTP/AVP dátových tokoch s použitím rôznych signalizačných protokolov. Aplikácia sa bude testovať s rôznym nastavením minimálneho počtu paketov na stream a to s pôvodnou hodnotou 100, 10 a 1. Týmto sa bude overovať schopnosť odhaliť aj minimálne časti tokov dát a vplyv týchto nastavení na falošnú detekciu. Ako testovacie sady budú použité rôzne vzorky odchytenej komunikácie vo formáte pcap, za použitia rôznych dát, prenášaných pomocou rôznych kódekov a signalizačných protokolov. Po zachytení s dátami nebude už ďalej manipulované.

Testy budú postupne spúšťané nasledujúcimi príkazmi:

- So zapnutou kontrolou zarovnávajúcich bajtov:
  - `./rtpinfo.py -f [súbor]`
  - `./rtpinfo.py -f [súbor] -m 10`
  - `./rtpinfo.py -f [súbor] -m 1`
- S vypnutou kontrolou zarovnávajúcich bajtov:
  - `./rtpinfo.py -f [súbor] -c`
  - `./rtpinfo.py -f [súbor] -c -m 10`
  - `./rtpinfo.py -f [súbor] -c -m 1`

Položka `[súbor]` sa bude vždy nahradzovať konkrétnym názvom súboru so zachytenými dátami.

Porovnávané nebudú len celkové čísla detekovaných paketov, ale aj správne priradenie jednotlivých paketov k streamom. V prípade, že paket bol priradený k inému streamu, bude tento odrátaný z celkového počtu správne detekovaných paketov a bude prirátaný

k počtu nesprávne detekovaných paketov. Táto skutočnosť naviac bude spomenutá vo vyhodnotení testu. Porovnávať sa bude výstupný list paketov vygenerovaný pomocou voľby `--packets_list`, oproti upravenému súboru CSV, ktorý bol exportovaný z programu Wireshark. Chybné priradenie je však veľmi nepravdepodobné na základe návrhu ukladania jednotlivých streamov a pravdepodobne by poukazovalo na chybu v implementácii.

Pri každom teste budú popísané dáta použité pri detekcii, tabuľka výsledkov detekcie a slovné vyhodnotenie výsledných dát a zistených skutočností. V tabuľke výsledkov budú vždy uvedené:

- Počet detekovaných streamov.
- Počet správne detekovaných paketov a percentuálne vyjadrenie vzhľadom na skutočný počet RTP alebo RTCP paketov.
- Počet falošne detekovaných paketov a percentuálne vyjadrenie vzhľadom k celkovému počtu spracovávaných paketov.
- Priemerný čas behu aplikácie v priebehu 5 pokusov v tvare minút:sekúnd.

5.2 Testy

5.2.1 Test č. 1

V rámci prvej vzorky dát prebiehala komunikácia medzi videokonferenčnou jednotkou Tandberg C60 a videokonferenčným softvérom Polycom PVX. Výsledný súbor so zachytenými dátami má veľkosť 20,8 MB a obsahuje celkovo 38193 paketov. Signalizácia medzi komunikujúcimi subjektami prebiehala pomocou štandardu H.323. Program Wireshark pomocou informácií nesených signalizačnými protokolmi identifikoval celkovo 36603 RTP paketov v siedmich streamoch.

V prvom teste bola použitá verzia programu bez možnosti vynechať kontrolu a orezanie doplňujúcich bajtov slúžiacich pre zarovnanie. Problém, ktorý sa rieši pomocou vynechania tejto kontroly bol totiž zistený, až pri testovaní na použitej vzorke dát.

Aplikácia sa spúšťala postupne postupne príkazmi:

```
./rtpinfo.py -f h323.pcap
./rtpinfo.py -f h323.pcap -m 10
./rtpinfo.py -f h323.pcap -m 1
```

Výsledky testu sú zaznamenané v tabuľke 5.1.

Min. paketov	**100**	**10**	**1**
Detekovaných streamov	5 (71,43%)	6 (85,71%)	7 (100%)
Správne detekovaných paketov	35998 (98,35%)	36010 (98,38%)	36015 (90,39%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	0:56	0:55	0:55

Tabuľka 5.1: Výsledky testu č. 1

Pri tomto teste nedošlo k žiadnym falošným detekciám, avšak neboli správne detekované všetky RTP pakety. Jeden priechod testu trval približne 55 sekúnd a teda rýchlosť spracovania dát bola približne 0,378 MB/s.

Po bližšom preskúmaní paketov, ktoré neboli správne detekované ako RTP, bolo ako príčina identifikované používanie bitu `padding` aj v prípade, že posledný bajt nenesie počet bajtov, ktoré môžu byť orezané. Toto sa prejavilo aj u iných dát používajúcich signalizáciu štandardom H.323. Toto môže spôsobovať napríklad šifrovanie dát, kde spôsob spracovávania zarovnávajúcich bajtov musí špecifikovať dokument, ktorý špecifikuje aj samotné šifrovanie [6].

Keďže spracovávanie takýchto dát je mimo rozsah tejto práce, ako najvhodnejšie riešenie bolo zvolené doplnenie možnosti túto kontrolu vynechať. Týmto sa síce zvýši potencionálne množstvo falošne detekovaných paketov, ale bude umožnená správna detekcia aj takýchto paketov.

Pri každom teste bude taktiež preverené korektné zozbieranie údajov dostupných z RTCP dát a korektné priradenie týchto údajov jednotlivým RTP streamom.

5.2.2 Test č. 2

Po doplnení možnosti vynechať kontrolu zarovnávajúcich dát, bol na sade dát z testu č. 1 vykonaný nový pokus, s použitím oboch volieb.

Aplikácia sa spúšťala postupne postupne príkazmi:

```
./rtpinfo.py -f h323.pcap
./rtpinfo.py -f h323.pcap -m 10
./rtpinfo.py -f h323.pcap -m 1
./rtpinfo.py -f h323.pcap -c
./rtpinfo.py -f h323.pcap -c -m 10
./rtpinfo.py -f h323.pcap -c -m 1
```

Výsledky testu sú zaznamenané v tabuľke 5.2.

Min. paketov (bez -c)	**100**	**10**	**1**
Detekovaných streamov	5 (71,43%)	6 (85,71%)	7 (100%)
Správne detekovaných paketov	35998 (98,35%)	36010 (98,38%)	36015 (90,39%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	0:55	0:55	0:55
Min. paketov (s -c)	**100**	**10**	**1**
Detekovaných streamov	5 (71,43%)	7 (100%)	7 (100%)
Správne detekovaných paketov	36490 (99,69%)	36603 (100%)	36603 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	0:46	0:47	0:46

Tabuľka 5.2: Výsledky testu č. 2

Z výsledkov môžeme vidieť, že vypnutie kontroly umožnilo správnu detekciu všetkých RTP paketov bez zavedenia falošnej detekcie na tejto sade dát. Neúplná detekcia v prípade nastavenia minimálneho počtu paketov na hodnotu 100 bola spôsobená malým počtom paketov v dvoch streamoch, kde ich bolo v prvom 56 a v druhom 55.

V prípade vynechania kontroly zarovnávajúcich bajtov stúpla rýchlosť spracovania dát z pôvodných 0,378 MB/s na približne 0,452 MB/s. Toto bolo spôsobené veľkým počtom paketov s nastaveným bitom `padding`, u ktorých vynechanie tejto kontroly spôsobilo celkové zrýchlenie procesu.

Informácie zozbierané z RTCP paketov boli správne priradené RTP streamom.

5.2.3 Test č. 3

Test bol vykonaný na sade dát zachytávajúcej prerušovanú komunikáciu medzi tromi počítačmi s videokonferenčným programom Polycom PVX a videokonferenčnou jednotkou Tandberg C60, ktorá zároveň poskytovala služby MCU. Výsledná veľkosť zachytených dát vo formáte pcap je 59,9 MB. Program Wireshark rozoznal ako RTP 99646 paketov z celkového počtu 101169 paketov. Tieto boli rozdelené medzi 30 streamov.

Aplikácia sa spúšťala postupne postupne príkazmi:

```
./rtpinfo.py -f h323-3PCMCU.pcap
./rtpinfo.py -f h323-3PCMCU.pcap -m 10
./rtpinfo.py -f h323-3PCMCU.pcap -m 1
./rtpinfo.py -f h323-3PCMCU.pcap -c
./rtpinfo.py -f h323-3PCMCU.pcap -c -m 10
./rtpinfo.py -f h323-3PCMCU.pcap -c -m 1
```

Výsledky testu sú zaznamenané v tabuľke 5.3.

Min. paketov (bez -c)	**100**	**10**	**1**
Detekovaných streamov	20 (66,67%)	20 (66,67%)	30 (100%)
Správne detekovaných paketov	97955 (98,30%)	97955 (98,30%)	97981 (98,33%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	2:22	2:20	2:19
Min. paketov (s -c)	**100**	**10**	**1**
Detekovaných streamov	20 (66,67%)	26 (86,67%)	30 (100%)
Správne detekovaných paketov	99478 (99,83%)	99638 (99,99%)	99646 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	2:01	2:01	2:01

Tabuľka 5.3: Výsledky testu č. 3

Z výsledkov je opäť poznať vplyv vynechania kontroly zarovnania, opäť umožnilo detekciu všetkých streamov. Vypnutím tejto kontroly opäť vzrástla aj priemerná rýchlosť spracovania dát z 0,427 MB/s na 0,495 MB/s.

Sto percentná úspešnosť však bola dosiahnutá až po znížení hodnoty minimálneho počtu paketov v streame na 1. Táto skutočnosť bola opäť spôsobená streamami s nízkym počtom paketov, ktoré niesli dáta udalostí pre zmenu polohy kamery na videokonferenčnej jednotke.

Informácie o z RTCP paketov neboli priradené k dvom zo streamov, po preskúmaní dát sa však ukázalo, že tieto neboli vôbec zachytené alebo odoslané z neznámeho dôvodu. Tým pádom môžeme povedať, že aplikácia z tohoto ohľadu funguje správne.

5.2.4 Test č. 4

Ďalšia sada dát bola zachytená pri krátkom telefonickom hovore medzi softvérovým telefónom SJphone a VoIP ústredňou AVAYA, ktoré používajú signalizačný protokol SIP. Veľkosť zachytených dát je 1,2 MB. Bolo zachytených celkovo 5301 paketov, z ktorých Wireshark rozpoznal ako RTP 5065 paketov v dvoch streamoch.

Aplikácia sa spúšťala postupne postupne príkazmi:

```
./rtpinfo.py -f sip.pcap
```

```
./rtpinfo.py -f sip.pcap -m 10
./rtpinfo.py -f sip.pcap -m 1
./rtpinfo.py -f sip.pcap -c
./rtpinfo.py -f sip.pcap -c -m 10
./rtpinfo.py -f sip.pcap -c -m 1
```

Výsledky testu sú zaznamenané v tabuľke 5.4.

Min. paketov (bez -c)	**100**	**10**	**1**
Detekovaných streamov	2 (100%)	2 (100%)	2 (100%)
Správne detekovaných paketov	5065 (100%)	5065 (100%)	5056 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	0:06	0:06	0:06
Min. paketov (s -c)	**100**	**10**	**1**
Detekovaných streamov	2 (100%)	2 (100%)	2 (100%)
Správne detekovaných paketov	5065 (100%)	5065 (100%)	5056 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	0 (0%)
Priemerný čas detekcie	0:06	0:06	0:06

Tabuľka 5.4: Výsledky testu č. 4

V tomto teste sa možnosť voľby vypnutia kontroly zarovnania absolútne neprejavila, a to ani v počte detekovaných paketov, ani v rýchlosti detekcie, ktorá bola približne 0,2 MB/s. Toto bolo spôsobené malým množstvom dát, pri ktorom nebolo možné, aby sa rozdiely prejavili. Ten istý dôvod malo aj výrazné zníženie rýchlosti, keď sa prejavila výraznejšie réžia programu pri inicializácii a výpise výsledkov.

Program v tomto teste správne detekoval všetky pakety, bez toho aby falošne detekoval nadbytočné RTP pakety.

Informácie z RTCP paketov boli priradené len jednému streamu. Toto bolo spôsobené použitím programu SJphone, ktorý nerešpektuje dokument [6] a RTCP pakety vôbec neodosiela.

5.2.5 Test č. 5

Tento test bol vykonaný na sade dát zachytávajúcej, audio a video stream na požiadanie. Ako server slúžil program VLC. Najprv bol spustený stream jedného videa, ktorý bol po chvíli zastavený. Následne boli spustené dva nezávislé streamy, z dvoch samostatných serverov. Vo všetkých prípadoch bola signalizácia prenášaná pomocou protokolu RTSP. Výsledná veľkosť súboru so zachytenými dátami je 180,8 MB. Wireshark rozpoznal ako RTP 184107 paketov, zaradených do šiestich streamov, z celkového počtu 197165 zachytených paketov.

Aplikácia sa spúšťala postupne postupne príkazmi:

```
./rtpinfo.py -f rtsp.pcap
./rtpinfo.py -f rtsp.pcap -m 10
./rtpinfo.py -f rtsp.pcap -m 1
./rtpinfo.py -f rtsp.pcap -c
./rtpinfo.py -f rtsp.pcap -c -m 10
./rtpinfo.py -f rtsp.pcap -c -m 1
```

Min. paketov (bez -c)	**100**	**10**	**1**
Detekovaných streamov	6 (100%)	6 (100%)	8 (133,33%)
Správne detekovaných paketov	184107 (100%)	184107 (100%)	184107 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	4 (0,002%)
Priemerný čas detekcie	3:49	3:45	3:47
Min. paketov (s -c)	**100**	**10**	**1**
Detekovaných streamov	6 (100%)	6 (100%)	8 (133,33%)
Správne detekovaných paketov	184107 (100%)	148107 (100%)	184107 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	4 (0,002%)
Priemerný čas detekcie	3:47	3:47	3:46

Tabuľka 5.5: Výsledky testu č. 5

Výsledky testu sú zaznamenané v tabuľke 5.5.

V teste bolo vo všetkých pokusoch správne detekovaných všetkých 184107 paketov, ktoré boli v každom prípade zaradené do správnych streamov. V prípade zníženia minimálneho počtu paketov v streame na jeden, boli falošne detekované 4 pakety. Tieto pakety boli zaradené do dvoch nových streamov a neovplyvnili teda správne detekované streamy. Tieto pakety v skutočnosti patrili komunikácii protokolu LLMNR.

Na výsledkoch sa vypnutie kontroly zarovnávajúcich bajtov vôbec neprejavilo, toto bolo spôsobené tým, že RTP pakety v tomto streame nepoužívali zarovnanie. Táto voľba sa tak isto neprejavila na falošnej detekcii, ktorá bola absolútne rovnaká v oboch prípadoch.

Priemerná rýchlosť spracovania dát bola 0,7965 MB/s. Tak isto sa na nej neprejavilo vypnutie kontroly zarovnávajúcich paketov z vyššie zmienených dôvodov.

Informácie získané pomocou RTCP boli správne priradené ku všetkým trom streamom.

5.2.6 Test č. 6

Všetky predchádzajúce testy boli primárne mierené na overenie správnej detekcie streamov. Neoverovali však dostatočne odolnosť voči falošnej detekcii. A práve na túto vlastnosť bol zameraný tento posledný test.

Sada dát zachytáva komunikáciu medzi dvoma softvérovými telefónmi Ekiga využívajúcich SIP, na rôznych operačných systémoch. Okrem samotného hovoru však na oboch systémoch, bolo aj množstvo inej komunikácie. Pre pridanie potencionálnych zdrojov falošnej detekcie, medzi týmito systémami prebiehala aj výmena súboru protokolom `tftp`, a pomocou nástroja netcat boli prenášané náhodné sekvencie, pomocou protokolu UDP.

Výsledná veľkosť súboru je 107,3 MB. Zachytených bolo tentokrát celkovo 126257 paketov, z ktorých iba 19313 paketov vyhodnotil program Wireshark ako RTP. Z týchto však prvé dva pakety, z neznámych dôvodov, obsahovali iba samé nuly, vrátane položiek v hlavičke. Z tohoto dôvodu budeme počítať, že súbor obsahuje 19311 RTP paketov, rozdelených do 4 streamov.

Aplikácia sa spúšťala postupne postupne príkazmi:

```
./rtpinfo.py -f sip-falsedetect.pcap
./rtpinfo.py -f sip-falsedetect.pcap -m 10
./rtpinfo.py -f sip-falsedetect.pcap -m 1
./rtpinfo.py -f sip-falsedetect.pcap -c
./rtpinfo.py -f sip-falsedetect.pcap -c -m 10
```

```
./rtpinfo.py -f sip-falsedetect.pcap -c -m 1
```

Výsledky testu sú zaznamenané v tabuľke 5.6.

Min. paketov (bez -c)	**100**	**10**	**1**
Detekovaných streamov	2 (50%)	2 (50%)	11056 (276400%)
Správne detekovaných paketov	19300 (99,94%)	19300 (99,94%)	19311 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	11052 (8,75%)
Priemerný čas detekcie	3:41	3:40	3:42
Min. paketov (s -c)	**100**	**10**	**1**
Detekovaných streamov	2 (50%)	2 (50%)	11060 (276500%)
Správne detekovaných paketov	19300 (99,94%)	19300 (99,94%)	19311 (100%)
Nesprávne detekovaných paketov	0 (0%)	0 (0%)	11056 (8,76%)
Priemerný čas detekcie	3:37	3:38	3:38

Tabuľka 5.6: Výsledky testu č. 6

V testoch s minimálnym počtom paketov nastavených na hodnoty 100 a 10 neboli detekované dva streamy. Tieto obsahovali v prvom prípade päť a v druhom šesť paketov. Tieto streamy obsahovali video, ktoré malo malý počet zmien a kódek umožňoval variabilný dátový tok. Z tohto dôvodu bolo odoslané len minimálne množstvo RTP paketov v týchto streamoch. V tomto prípade sa ukazuje dôležitosť správneho nastavenia minimálneho počtu paketov pre zachytenie všetkých streamov.

V prípade nastavenia minimálneho počtu paketov v streame na hodnotu jedna boli už správne detekované všetky streamy, ale bolo taktiež falošne detekované veľké množstvo paketov. Z výsledkov je však vidno, že každý falošne detekovaný paket je zaradený do samostatného streamu. Toto bolo overené spustením detekcie s nastavením minimálneho počtu paketov v streame na hodnotu dva. V tomto prípade boli správne detekované všetky RTP pakety a streamy bez falošne detekovaných paketov.

Potlačenie kontroly zarovnávajúcich bajtov nemalo na výsledky žiadny výraznejší vplyv. Iba v poslednom prípade, bol počet falošne detekovaných paketov zvýšený o štyri. Taktiež rýchlosť spracovania dát stúpla len zanedbateľne a priemerne sa po celú dobu pohybovala na hodnote 0,4877 MB/s.

Ku všetkým streamom boli správne zozbierané a priradené dáta získané z RTCP paketov.

5.2.7 Celkové vyhodnotenie

Na začiatku testovania bola odhalená potreba vypnutia jednej kontroly pre spoľahlivú detekciu všetkých reálne prenášaných RTP streamov. Táto voľba zároveň nespôsobila výrazne zvýšenie falošnej detekcie paketov a preto by som ju doporučoval v praxi vypínať. Ako pozitívny, vedľajší efekt mala voľba aj mierny nárast rýchlosti spracovania dát, ktorý ale môže pri väčších dátových sadách spôsobovať zrýchlenie o desiatky sekúnd, až minút.

V priebehu testovania sa zároveň potvrdil predpoklad, že nesprávne priradenie paketu k inému streamu nie je možné, keďže takáto situácia nenastala ani pri jednom pakete v žiadnom teste.

Presnosť detekcie streamov sa ukázala dostatočná aj pri štandardnom nastavení minimálneho počtu paketov v streame na hodnotu 100. Streamy, ktoré neboli detekované z dôvodu priveľmi vysokej hodnoty, niesli väčšinou málo vysielané signalizačné dáta, a teda

neboli z pohľadu sledovania audio a video dát podstatné. V reálnom nasadení môžem teda doporučiť ponechanie pôvodnej hodnoty, toto však značne závisí na konkrétnom účele detekcie, použitých kódekoch a dobe po ktorú sú dáta odchytávané.

Pri všetkých testoch boli korektne zozbierané dostupné dáta o jednotlivých synchronizačných zdrojoch z RTCP paketov a zároveň boli správne priradené jednotlivým RTP streamom.

Kapitola 6

Záver

Cieľom tejto práce bolo vyvinúť algoritmus a nástroj pre detekciu RTP streamov a experimentálne overiť jeho funkčnosť.

Na základe porovnania s existujúcimi nástrojmi je úspešnosť detekcie vyvinutého nástroja dostatočne presná. Nástroj dokázal správne identifikovať všetky streamy bez nadmernej falošnej detekcie. Tá nastáva len v prípade nastavenia nízkej hodnoty minimálneho počtu paketov v streame, pretože väčšina falošne detekovaných streamov nedosiahne viac ako jeden až dva pakety.

Nástroj je tak isto schopný správneho zberu informácií o streame zo samotných RTP a RTCP paketov a ich následného poskytnutia k ďalšiemu spracovaniu.

Spracovanie paketov je možné zo zachytených dát uložených vo formáte pcap, nástroja tcpdump, alebo pomocou odchytávania v reálnom čase na niektorom zo sieťových rozhraní.

Vyvinutý algoritmus nie je schopný detekcie vysoko neštandardných streamov iných profilov ako RTP/AVP, alebo nesených iným transportným protokolom ako UDP. Toto však nebolo cieľom tejto práce a bez bližšej špecifikácie pre potreby detekcie týchto streamov, by túto funkčnosť pravdepodobne nebolo možné s dostatočnou účinnosťou dosiahnuť.

Keďže je nástroj implementovaný v interpretovanom jazyku Python, je pomerne pomalý, a preto nie je vhodný pre spracovanie veľkého množstva dát v reálnom čase. Rýchlosť detekcie by však bolo možné zásadne zvýšiť implementáciou vyvinutého postupu v jazyku prekladanom do strojového kódu (napr. C/C++).

Medzi ďalšie možné zlepšenia patria napríklad vylepšenie mapovania informácií zistených z RTCP paketov na dané RTP streamy, kde môže momentálne vzniknúť konflikt v prípade, že by na sieti existovali dva streamy medzi navzájom rôznymi subjektami, ktoré by vygenerovali zhodné identifikácie synchronizačného zdroja. Keďže sa táto hodnota generuje z veľkej množiny čísel a samotný proces generovania popísaný v [6] hovorí, že samotné generovanie úplne náhodného čísla nie je dostatočné, ale treba ho založiť aj na hodnotách špecifických pre daný subjekt, pravdepodobnosť kolízie je minimálna. Vylepšením spôsobu mapovania by mohol byť rovnako umožnený zber informácií odosielaných príjemcom streamu.

Ďalším možným rozšírením by bola možnosť prechádzať samotné dáta nesené RTP streamom a zisťovať konkrétne o aký kódek sa jedná, prípadne umožniť uloženie prehrateľného súboru.

Literatúra

[1] BAUGHER, M., MCGREW, D., NASLUND, M. et al. *The Secure Real-time Transport Protocol (SRTP), RFC 3711* [online]. March 2004. Dostupné na: <http://tools.ietf.org/html/rfc3711>.

[2] POSTEL, J. *User Datagram Protocol, RFC 768* [online]. August 1980. Dostupné na: <http://tools.ietf.org/html/rfc768>.

[3] SCHULZRINNE, H. *RTP Profile for Audio and Video Conferences with Minimal Control, RFC 1890* [online]. Január 1996. Dostupné na: <http://tools.ietf.org/html/rfc1890>.

[4] SCHULZRINNE, H., CASNER, S. *RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551* [online]. Júl 2003. Dostupné na: <http://tools.ietf.org/html/rfc3551>.

[5] SCHULZRINNE, H., CASNER, S., FREDERICK, R. et al. *RTP: A Transport Protocol for Real-Time Applications, RFC 1889* [online]. Január 1996. Dostupné na: <http://tools.ietf.org/html/rfc1889>.

[6] SCHULZRINNE, H., CASNER, S., FREDERICK, R. et al. *RTP: A Transport Protocol for Real-Time Applications, RFC 3550* [online]. Júl 2003. Dostupné na: <http://tools.ietf.org/html/rfc3550>.

Dodatok A

Obsah DVD

Priložený dátový nosič DVD obsahuje:

- text bakalárskej práce vo formáte PDF
- zdrojové súbory bakalárskej práce pre systém LaTeX
- vyvinuté moduly rtpdetector.py a rtcpdetector.py
- testovaciu aplikáciu rtpinfo.py
- dokumentáciu vygenerovanú pomocou nástroja pydoc
- knižnica Scapy
- súbory s odchytenou komunikáciou použitou pri testovaní
  - Test č. 1 – h323.pcap
  - Test č. 2 – h323.pcap
  - Test č. 3 – h323-3PCMCU.pcap
  - Test č. 4 – sip.pcap
  - Test č. 5 – rtsp.pcap
  - Test č. 6 – sip-falsedetect.pcap

Dodatok B

Prehľad niektorých multimediálnych kódekov prenášaných pomocou RTP

encoding name	Názov	Ďalšie informácie
PCMU	G.711 PCM μ-Law	RFC 3551
GSM	GSM	RFC 3551
G723	G.723.1	RFC 3551
DVI4	IMA ADPCM	RFC 3551
LPC	Linear Predictive Coding	RFC 3551
PCMA PCMU	G.711 PCM *A*-Law	RFC 3551
G722	G.722	RFC 3551
L16	Linear PCM 16-bit	RFC 3551
QCELP	Qualcomm Code Excited Linear Prediction	RFC 2658
MPA	MPEG-1 or MPEG-2 Audio	RFC 2250
G728	G.728	RFC 3551
G729	G.729	RFC 3551
G726	G.726	RFC 3551
GSM-EFR	GSM-EFR	RFC 3551
L8	Linear PCM 8-bit audio	RFC 3551
RED	Redundant audio payload format	RFC 2198
VDVI	Variable rate DVI4	RFC 3551
CelB	Sun's CellB Video Encoding	RFC 2029
JPEG	JPEG Video	RFC 2435
nv	Xerox PARC's Network Video	RFC 3551
H261	H.261	RFC 4587
MPV	MPEG-1 or MPEG-2 Video	RFC 2250
MP2T	MPEG-2 transport stream	RFC 2250
H263	H.263	RFC 2190
H263-1998	H.263, second version	RFC 4629
speex	Speex Codec	www.speex.org/
iLBC	Internet low Bitrate Codec	RFC 3951
CELT	CELT	www.celt-codec.org/
H264	H.264	RFC 3984
theora	Theora	www.theora.org/
MP4V-ES	MPEG-4 Visual	RFC 3016
BV32	BroadVoice Speech Codec	www.broadcom.com/support/broadvoice/
SIREN14	Polycom Siren 14/G 722.1C	http://www.polycom.com/company/about_us/technology/siren14_g7221c/
G7221	G.722.1	RFC 5577

Tabuľka B.1: Prehľad niektorých multimediálnych kódekov prenášaných pomocou RTP