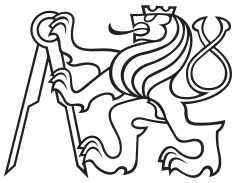
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra telekomunikační techniky

Pasivní analyzátor SIP signalizace

Bc. Vladimir Evseev

Vedoucí: Ing. Ján Kučerák
Obor: Sítě elektronických komunikací
Studijní program: Komunikace, multimédia a elektronika
Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra telekomunikační techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Vladimir Evseev**

Studijní program: Komunikace, multimédia a elektronika
Obor: Sítě elektronických komunikací

Název tématu: **Pasivní analyzátor SIP signalizace**

Pokyny pro vypracování:

Navrhněte a realizujte systém pro pasivní analýzu zachycené SIP signalizace. Předpokládejte otevřený (nešifrovaný) formát záchytu, soustřeďte se na možnosti analýzy větších objemů dat a pokuste se implementovat proces analýzy distribuovaným způsobem.

Seznam odborné literatury:

[1] *RFC 3261 The SIP Protocol.* Dostupné na http://www.rfc-base.org/rfc-3261.html [on-line].
[2] Sisalem, D.: *SIP Security*. Dostupné na http://www.iptel.org/~dor/research.htm [on-line].
[3] Johnston, A.B.: *SIP - Understanding the Session Initiation Protocol*. Artech House; 3 edition (9/2009). 395 pages. ISBN: 978-1-60783-995-8.

Vedoucí: Ing. Ján Kučerák

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Ing. Boris Šimák, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 21. 12. 2015

Poděkování

Děkuji Ing. Jánu Kučerákovi za trpělivost, poskytnuté rady a odborné vedení v průběhu práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně pod vedením vedoucího diplomové práce, a že jsem uvedl veškerou použitou literaturu a další informační zdroje. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.

V Praze, 27. května 2016

Abstrakt

Tato diplomová práce se zabývá návrhem a realizací systému pro pasivní analýzu SIP signalizace a zaměřuje se na možnost zpracování větších objemů dat. V teoretickém rozboru je obsažen popis problematiky protokolů SIP a SDP, a také analýza existujících řešení pro zachycení a zpracování síťové komunikace. Hlavní část práce se zaměřuje na implementaci systému. Zpracování pcap souboru se řeší pomocí parseru, který je napsán v jazyce Python. Uložení výsledků se provádí do SQL databáze, do níž se pak vytváří dotazy pro analýzu stavu hovoru. Pro snadné ovládaní systémem je realizováno webové rozhraní.

V závěru práce jsou provedeny testy zpracování souborů různé velikosti a zhodnocení výsledků měření.

Klíčová slova: SIP, SDP, VoIP, pcap, klient, server, analyzátor

Vedoucí: Ing. Ján Kučerák

Abstract

This thesis describes the design and implementation of the system for passive analyzing of SIP signaling and focuses on processing large volumes of data. The theoretical part contains basic information about SIP and SDP protokols and provides an overview of existing solutions and tools for capturing and processing network communications. The main part focuses on the system implementation. The parser for pcap processing is written in Python. Results are saved into the SQL database. Then queries are performed to analyse the call state. Web-based user interface is implemented for easier interaction with the system.

In the conclusion assessments of the results are given as they pertain to the various file size.

Keywords: SIP, SDP, VoIP, pcap, client, server, parser

Obsah

Obrázky

Tabulky

Seznam zkratek

CLI	Command Line Interface
HTTP	HyperText Transport Protocol
IP	Internet Protocol
RFC	Request for Comments
RTP	Real-time Transport Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
TCP	Transmission Control Protocol
UA	User Agent
UDP	User Datagram Protocol
VoIP	Voice over IP

Kapitola 1
Úvod

VoIP technologie více a více vstupují do našeho života. Vzrůstá počet aplikací, které potřebují ke svému fungování různé druhy spojení mezi účastníky (telefonní hovory, video hovory atd.). Při správě a optimalizaci těchto aplikací jsou kladeny požadavky na analýzu protokolů, které slouží k navazování a ukončování spojení. Nejrozšířenější z nich je protokol SIP, kterým se podrobně zabývá tato práce. Hlavním zdrojem informací o SIP protokolu je RFC 3261 [1].

Pokud jde o analýzu jakéhokoliv síťového provozu, ve většině případu se používá aplikace Wireshark, která nabízí podporu velkého množství protokolů a bohatý výběr nástrojů pro analýzu, více informací lze získat na oficiálních stránkách [2]. Jednou z nevýhod tohoto programu je však to, že v závislosti na technických prostředcích může být obtížné zpracování velkých souborů. To znamená, že při analýze pcap souborů, které jsou větší než 1 GB, může být Wireshark někdy nestabilní. Problém spočívá v tom, že Wireshark nahraje otevřený soubor do operační paměti a poté s ním pracuje. Je si třeba uvědomit, že například ze střední VoIP ústředny může být zachycen mnohem větší objem dat, než zmíněný 1 GB.

Z toho důvodu vznikla tato diplomová práce, jejíž účelem je návrh systému pro pasivní analýzu zachycené SIP signalizace. Cílem je doplnit funkcionalitu aplikace Wireshark, nikoli ji nahradit, a poskytnout uživateli vetší pohodlí při zpracovaní velkého množství SIP signalizace.

Druhá kapitola práce je zaměřena na seznámení z principy fungovaní protokolu SIP. Zahrnuje popis samotného protokolu SIP a také protokolu SDP, o němž lze přečíst v RFC 4566 [3], a příklady sestavení a ukončení spojení.

Předmětem následující kapitoly je analýza existujících řešení pro zpracování zachycené komunikace. V ní jsou rozebrány nejpoužívanější síťové analyzátory, dostupné knihovny a nástroje, které pracují s pcap formátem.

V kapitole 4 je představen návrh možného řešení systému pro analýzu SIP signalizace, podle něhož zpracování probíhá ve dvou dílčích krocích. První se zabývá zpracováním síťových dat a druhý je zaměřen na možnou realizaci dotazů na základě výsledků.

Na implementaci výsledné aplikace je zaměřena pátá kapitola. V té jsou uvedeny nejprve technologie, které byly využity při vývoji aplikace, je navržena struktura celého řešení a jsou rozebrány jednotlivé fáze zpracování.

Na použití výsledné aplikace je soustředěna šestá kapitola. Nejprve v ní jsou rozebrány instalační pokyny a pak následuje krátký úvod do uživatelského rozhraní a představení možností aplikace.

Poslední kapitola se zabývá zhodnocením implementovaného řešení na základě porovnání času zpracování různě velkých souborů. U výsledků testů nejde o získané absolutní hodnoty, ale spíše o relativní rozdíly mezi získanými hodnotami při různých konfiguracích serverů. Kapitola je také zaměřena na možné vylepšení a rozšíření navrženého systému.

Kapitola 2
VoIP technologie

Tento projekt je zaměřen na zpracování protokolu SIP, který slouží k přenosu signalizace mezi účastníky. Tato kapitola je teoretickým úvodem problematiky protokolů SIP, SDP a RTP.

2.1 Session Initiation Protocol

SIP (Session Initiation Protocol) je signalizační protokol, který byl standardizován v roce 2002 dokumentem RFC 3261 [1]. Tento signalizační protokol umožňuje sestavení, modifikaci a ukončení relace s jedním nebo více účastníky.

2.1.1 Vlastnosti

Jedná se o textový protokol kódovaný ve znakové sadě UTF-8, inspirovaný protokoly HTTP (Hypertext Transfer Protocol) a SMTP (Simple Mail Transfer Protocol). Podobně jako v HTTP klient zasílá požadavky na server a server odpovídá klientovi. Na začátku žádosti je umístěna metoda, adresa URI (Uniform Resource Identifier) a verze protokolu. V odpovědi lze najít verzi, návratový kód a jeho slovní popis. Zprávy mají hlavičku a tělo.

Účastníci na sebe odkazují pomocí tzv. SIP URI (Uniform Resource Identifier), který vypadá jako e-mailová adresa: `sip:uzivatel@domena.cz` nebo `sips:uzivatel@domena.cz`. Pro spojení s veřejnou telefonní sítí, používají se telefonní čísla zapsaná následujícím způsobem (tzv. Tel URI): `tel:5-5564-4655`.

SIP URI obsahují adresu domény a jsou propojeny se systémem DNS (Domain Name System). Tel URI jsou mapovány rozšířením ENUM (E.164 Number Mapping), což umožňuje dohledat příslušné domény, k nimž patří dané telefonní číslo.

SIP je nezávislý na transportní vrstvě, však primárním komunikačním protokolem transportní vrstvy je UDP protokol a typický port je 5060 (5061 pro zabezpečený SIP). Dále SIP může využívat spojový protokol TCP (Transmission Control Protocol) a jeho zabezpečenou variantu SCTP (Stream Control Transmission Protocol). Standardem nejsou specifikovány komunikační protokoly, které mají být použité pro vlastní přenos multimediálních dat. Především se používá protokol RTP (Real-time Transport Protocol).

SIP sám o sobě neposkytuje služby IP telefonie, jenom umožňuje tyto služby:

- nalezení volaného uživatele, překlad jmen;
- zjištění, zda je tento uživatel ochoten komunikovat (nenastane odmítnutí nebo obsazení);
- dohodu parametrů spojení;
- navázání spojení;
- řízení spojení, změny parametrů a ukončení spojení.

Pro dohodu parametrů spojení SIP potřebuje další protokol - SDP (Session Description Protocol), jehož zprávy se posílají jako tělo SIP zpráv. Tento protokol slouží k dohodnutí dvou stran o formátu posílaní datových toků a jejich směrů.

2.1.2 Architektura

Prvky SIP architektury:

- User Agent (UA)
- Servery

UA zařízení jsou koncovými zařízeními SIP sítě. Rozdělují se na User Agent Client (UAC) - odesílá žádosti a přijímá odpovědi, a User Agent Server (UAS) - obraceně, odesílá odpovědi a přijímá žádosti [4]. Zvláštním případem je B2BUA (Back to Back User Agent), který jako server přijme žádost a aby ji vyplnil, odešle jako klient novou žádost dále do sítě. V zásadě je to brána mezi dvěma SIP sítěmi. Na rozdíl od proxy serveru musí udržovat stav navázaných SIP dialogů.

Servery se dělí na Proxy, přesměrovací (Redirect) a registrační (Registrar) [4]. Proxy server přijímá požadavek o spojení od UA nebo od jiného proxy serveru a předává tento požadavek dalšímu proxy serveru, pokud volanou stanici nemá ve své správě, nebo přímo volanému UA, pokud je tento účastník v rámci spravované domény. Na proxy serveru také může probíhat autentizace a kontrola administrativních politik, například smí-li účastník vůbec hovor uskutečnit. Proxy servery se dělí na bezestavové a stavové.

Redirect server slouží také ke směrování požadavků k cíli, ale nepřeposílá je dále ve směru volaného. Posílá zpět informaci v odpovědi zařazené do třídy 3xx, komu má UA žádost poslat, aby se dostala k požadovanému cíli.

Registrar server přijímá registrační žádosti od UA a aktualizuje podle nich záznamy v databázi koncových zařízení (location service) pro svou doménu. Z této databáze je pak čtou ostatní druhy SIP serverů obsluhující danou doménu, a to jim umožňuje převádět SIP URI na IP adresy. Všechny výše uvedené druhy prvků nemusí být oddělené.

2.1.3 Druhy zpráv

Pro potřeby komunikace jsou v SIP definovány dva druhy zpráv – žádosti a odpovědi (requests a responses). První řádek zprávy určuje její typ.

Žádosti slouží k posílání požadavků. První řádek má následující strukturu:
`Request-Line: Method Request-URI SIP-Version CRLF`

`Method`: slouží pro identifikaci typu žádosti, existuje 6 typů: `REGISTER`, `INVITE`, `ACK`, `CANCEL`, `BYE`, `OPTIONS`. Význam jednotlivých metod je uveden v tabulce 2.1.

`Request-URI`: identifikuje volaného uživatele nebo zařízení.

`SIP-Version`: obsahuje verzi protokolu.

Method	Význam
`INVITE`	pozvánka k hovoru, jakékoli navazování spojení musí začínat pomocí této metody
`REGISTER`	registrace účastníka hovoru na SIP server
`ACK`	potvrzení přijetí hovoru (potvrzuje, že přišla odpověď OK)
`CANCEL`	zrušení žádosti
`OPTIONS`	dojednání dodatečné informace
`BYE`	ukončení hovoru

Tabulka 2.1: Typy žádostí protokolu SIP dle RFC 3261

Odpovědi protokolu SIP slouží jako zpětná vazba pro žádosti. Zde první řádek začíná slovem Status-Line a má následující strukturu:
`Status-Line: SIP-Version Status-Code Reason-Phrase CRLF`

`Status-Code`: číselná odpověď, která vyjadřuje výsledek žádosti. Existuje 6 možných variant pro odpověď, vždy záleží na první číslici. Typy odpovědí jsou uvedeny v tabulce 2.2.

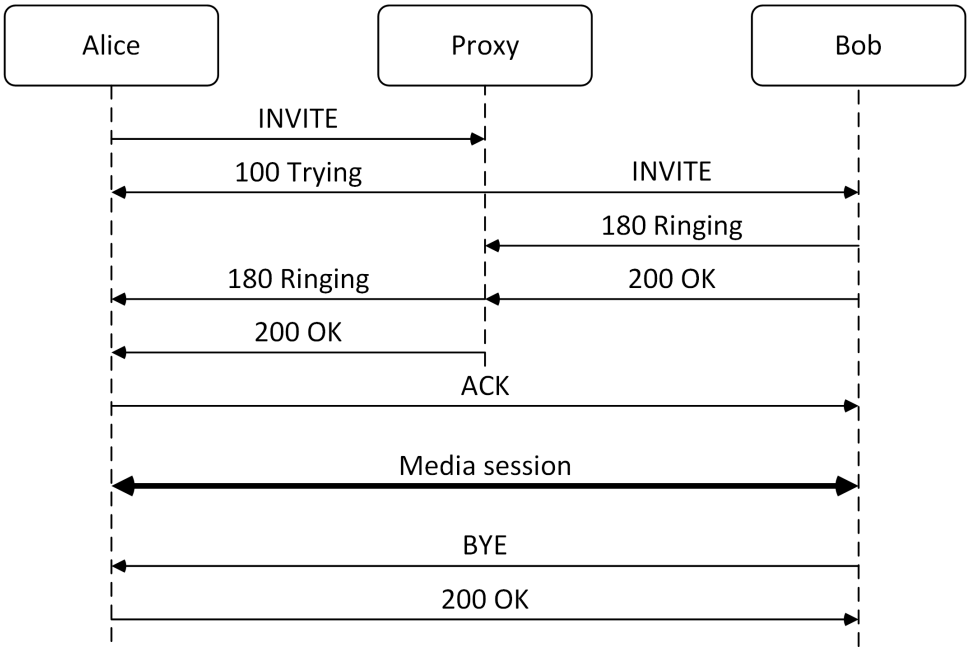
`Reason-Phrase`: textový popis položky Status-Code. Slouží jako překlad hodnoty Status-Code do čitelné podoby.

Status-Code	Význam
1xx	žádost byla přijata, bude se dále zpracovávat
2xx	žádost byla v pořádku přijata a potvrzena
3xx	přesměrování žádosti, je potřeba další informace
4xx	chyba na straně klienta, špatná žádost
5xx	chyba na straně serveru, žádost v pořádku, ale server není ji schopen vyřídit
6xx	globální chyba, žádný server nemůže vyřídit tuto žádost

Tabulka 2.2: Typy odpovědí protokolu SIP dle RFC 3261

2.1.4 Příklad SIP komunikace

Na obrázku 2.1 je vidět proces navazování a ukončování spojení mezi dvěma koncovými uzly.



Obrázek 2.1: Navazování a ukončování spojení

Celá komunikace probíhá tak, že Alice odešle požadavek na spojení s jiným koncovým bodem (v dánem příkladu je to Bob) na proxy server s použitím metody INVITE. Ten jí odpoví zatím zprávou 100 TRYING, která znamená, že server zpracovává požadavek. Proxy server se snaží lokalizovat koncového uživatele Boba pomocí URI. Pokud dorazí požadavek INVITE až k Bobovi, tak jeho koncové zařízení odpovídá zprávou 180 RINGING, čímž informuje server o čekání na odezvu uživatele Boba. Pokud Bob hovor přijme, odesílá jeho koncová stanice zprávu oznamující úspěch požadavku INVITE pomocí zprávy 200 OK. Tato signalizace se pak přeposílá až ke koncové stanici Alice, která se tím dozví o reakci Boba, a potvrdí přijetí zprávy pomocí metody ACK. Pokud se zpráva ACK dostane až k Bobovi, tak je úspěšně navázána jejich multimediální RTP relace [5].

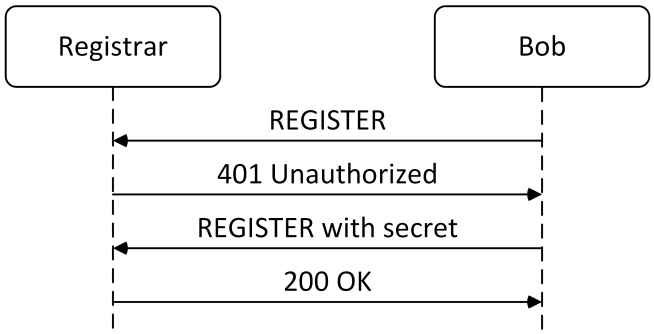
Ukončení hovoru je pak signalizováno zprávou BYE, kterou mohou zaslat oba dva účastníci, a k tomu je ještě potvrzeno kódem 200 OK. V této chvíli dojde k přerušení multimediální RTP relace.

Průběh registrace je popsán na obrázku 2.2. Nejprve klient posílá SIP serveru neautentizovanou zprávu REGISTER, na kterou server odpoví zprávou 401 Unauthorized. Tato žádost si vynucuje autentizaci uživatele a obsahuje výzvu a metodu, kterou se má autentizace provést. Ve výzvě je obsaženo náhodné číslo a metodou se myslí, jaká hashovací funkce má být použita [6]. Tyto udaje jsou obsaženy v poli WWW-Authenticate.

Klient poté na základě znalosti náhodného čísla a svého hesla vypočítá

pomocí dané hashovací funkce hash a zopakuje původní zprávu REGISTER, do které doplní pole Authorization obsahující vypočtenou hash. Tento hash pak server ověří výpočtem na své straně [7]. Pokud výpočet na serveru souhlasí s výpočtem, který poslal klient, je autentizace potvrzena zprávou 200 OK.

Nejdůležitější položky ve zprávě REGISTER jsou Contact a To. Contact obsahuje IP adresu zařízení, které je aktuálně používané uživatelem. Položka To nese SIP adresu uživatele, který se chce zaregistrovat. Klient může specifikovat dobu, po níž má být adresa registrována v poli Expires. Server může provést registraci buď na požadovanou nebo na kratší dobu.



Obrázek 2.2: Registrace

2.2 Session Description Protocol

Detaily jedné relace, jako je typ media, použitý kodek nebo vzorkovací rychlost, nejsou součástí signalizačního protokolu SIP. Tyto parametry se popisují pomocí protokolu SDP (Session Description Protocol), o němž lze více dozvědět více z RFC 4566 [3]. Nepřenáší se pomocí něj vlastní data. SDP zpráva je přenášena v těle SIP zprávy. Obě strany se musí pomocí dvoufázové SDP výměny vzájemně domluvit na parametrech spojení, transportním protokolu, typu kodeku nebo přenosové rychlosti.

Relace je popsána řadou dvojic atribut-hodnota, vždy po jedné na řádku. Názvy atributů jsou jednopísmenné, pak je následuje '=' a hodnota atributu. Nepovinné atributy se označují pomocí '=*'. Hodnota atributu je ASCII řetězec nebo posloupnost určitých značek oddělených mezerou. Atribut může být uveden jen v jedné ze tří částí (Session, Time nebo Media). Syntax SDP je rozšiřitelná. Dále se uvádí jednotlivé položky SDP zprávy.

Session description - Popis relace

v =	verze protokolu
o =	původce a identifikátor relace
s =	jméno relace
i =*	textová zpráva s informací, co se v dané relaci děje
u =*	URI popisu
e =*	e-mailová adresa na osobu zodpovědnou za relaci
p =*	telefonní číslo na osobu zodpovědnou za relaci

c =* informace pro spojení – není vyžadována, je-li zahrnuta v popisu média
b =* 0 či více řádků informací o šířce pásma
1 či více Time descriptions ("t="a "r="řádky; viz níže)
z =* nastavení časového pásma
k =* šifrovací klíč
a =* 0 či více řádků atributů relace
0 či více Media description

Time description - Popis času
t = doba, po kterou je relace aktivní
r =* naplánování automatického zopakování relace

Media description - Popis média
m = název média a transportní adresa
i =* označení média
c =* informace pro spojení
b =* 0 či více řádků informací o šířce pásma
k =* šifrovací klíč
a =* 0 či více řádků atributů relace

2.3 Real-time Transport Protocol

Real-time Transport Protocol (RTP) je protokol, který se používá k samotnému přenášení hlasu a videa ke koncovým uživatelům. Jedná se o real-time protokol a dochází k přenosu dat v reálném čase. Více o RTP protokolu se lze dozvědět z RFC3550 [8].

Přenos dat v reálném čase je zajištěn využitím procedur, které protokol RTP obsahuje. RTP protokol využívá služeb, které nabízejí určování přenášených dat, číslování sekvencí paketů, sledování přenosu a časové známkování (timestamping). RTP protokol navíc podporuje multiplexing a protokol RTCP (RTP Control Protocol), který má zajistit určitou kvalitu poskytovaných služeb, odhadnout velikost relace a poskytnout s tím spojenou kontrolu proti zahlcení. Pomocí podpůrného protokolu RTCP jsou přenášeny statické informace o RTP relace, kde se vyskytují informace o kvalitě, zpoždění přenosu a jitteru.

Nejčastěji používanými protokoly pro přenos informace jsou protokoly UDP a TCP, které nemají pevně definované komunikační porty. Omezující faktor je dynamické přidělování portů, což způsobuje špatnou prostupnost přes firewall nebo NAT.

Kapitola 3

Pasivní síťové analyzátory

Předmětem této kapitoly je představení nejpoužívanějších síťových analyzátorů, dostupných knihoven a nástrojů, které pracují s pcap formátem.

3.1 Definice

Pasivním síťovým analyzátorem je software, který je zaměřen na rozbor síťové komunikace na základě odchycených záznamů paketů ze síťové karty bez jakékoli modifikace přenášených dat. Data jsou ukládána do binárních souborů různých formátů, mezi nejznámější formáty patří: pcap, pcapng či cap (Sun Microsystems, Microsoft Network Monitor) [2].

Za prvé úkolem analyzátoru je načíst odchycená data a dle vrstvy síťového protokolu vyhodnotit hlavičku a odeslat vrstvě vyšší, pokud je v datech zachycena. Za druhé by měl analyzátor zahrnovat pomocné nástroje, aby usnadnil hodnocení zpracovaných dat uživatelem, případně by měl být schopen udělat závěry o zpracovaných datech samostatně.

Při komunikaci se síťová karta chová tím způsobem, že na základě své hardwarové adresy přijímá pouze data, která jsou pro ni určená, a předává je jádru operačního systému. V Promiscuous módu jsou ale přijímána všechna data, bez ohledu na adresu skutečného adresáta. Typicky jsou pro zapnutí Promiscuous módu nutná práva superuživatele v případě operačního systému Linux a tato metoda je základem pro získávání dat pro všechny druhy síťových analyzátorů.

3.2 Libpcap, WinPcap

Na Unixových a Linuxových platformách je pcap implementován knihovnou Libpcap. Byla vyvinuta v kalifornské Lawrence Berkeley National Laboratory a je spravována tvůrci programu tcpdump. Pro Windows je dostupná knihovna WinPcap. Knihovny Libpcap a WinPcap jsou napsány v jazyce C [9].

Aplikační programy můžou používat knihovny Libpcap a WinPcap na odchytávání síťové komunikace - paketů procházejících přes (resp. přicházejících na) dané zařízení prostřednictvím počítačové sítě. Dokáže ze segmentu sítě také odchytávat pakety, které nejsou adresované danému zařízení, síťová

karta však musí být v promiscuous režimu. Novější verze dokážou i vysílat vytvořené, resp. odchycené pakety do počítačové sítě.

Knihovny Libpcap a WinPcap používají mnohé otevřené i komerční síťové aplikace, jde zejména o analyzátory síťových protokolů (protocol analyzers), monitory sítě (network monitors), systémy pro odhalení průniku do sítě (NIDS), odchytávače paketů (packet sniffers), generátory síťového provozu (traffic generators) a testery sítě (network testers).

Tyto systémy dokážou pracovat s formátem souborů pcap. V tomto formátu síťová komunikace jsou ukládaná až od úrovně linkové vrstvy (včetně) ISO/OSI referenčního modelu. Programy založené na Libpcap a WinPcap knihovnách (například tcpdump, Wireshark atd., jsou popsány v dalších kapitolách) dokážou vytvářet i číst zmiňované soubory. Dokážou tedy zachytávat a ukládat sledovanou síťovou komunikaci a později se k ní vrátit za účelem její další analýzy.

3.3 Tcpdump

Tcpdump je jedním z nejstarších analyzátorů využívaných doposud, který byl původně napsán v roce 1987. Tcpdump je typicky součástí každé distribuce systému Linux a funguje pouze v konzolovém režimu. Program umí pracovat s uloženými daty v souboru, umí sám ukládat data do takového souboru (formát pcap) a taký pracovat s daty v reálném čase (Promiscuous mód). Data se dají lehce filtrovat dle třech základních kategorií: typ, směr a protokol [10]. Pro ilustraci filtrování lze se podívat na následující příklady:

```
tcpdump host target.pc and not nontarget.pc
tcpdump tcp port 80 or ssh
tcpdump 'src 10.0.2.4 and (dst port 3389 or 22)'
tcpdump 'tcp[tcpflags] and (tcp-syn or tcp-ack) not 0'
```

Program tcpdump podporuje nativně velkou množinu protokolů, jejichž data dokáže prezentovat v textové podobě. Avšak tcpdump nemá žádné interní moduly pro pomoc uživateli při hodnocení síťového provozu. Typický výpis programu je vidět na obrázku 3.1.

3.4 Tcpflow

Jako o druhém analyzátoru je možné zmínit o programu tcpflow [11]. Tcpflow je program, který zachycuje data přenášená jako součást TCP spojení (toků). Na rozdíl od tcpdumpu a Wiresharku tcpflow rekonstruuje skutečné datové toky a ukládá každý tok do samostatného souboru pro pozdější analýzu nebo ladění. Spojení jsou ukládána do souborů ve formátu, kde figurují zdrojová a cílová IP adresa a taký porty. Tcpflow podporuje čísla TCP sekvence a bude správně rekonstruovat datové toky bez ohledu na opakované přenosy, ale program bohužel nepodporuje IP fragmentaci. Podobně jako v případě předchozího představeného programu umožňuje tcpflow filtrovat data za

```
20:41:17.740174 IP (tos 0x0, ttl 64, id 50628, offset 0, flags [DF], proto UDP (
17), length 92)
    OpenWrt.lan.domain > pc1.lan.41977: [udp sum ok] 33535 q: A? ip5045453.ahcdn
.com. 1/0/1 ip5045453.ahcdn.com. A 88.208.8.161 ar: . OPT UDPsize=4096 (64)
20:41:17.749256 IP (tos 0x0, ttl 64, id 50629, offset 0, flags [DF], proto UDP (
17), length 140)
    OpenWrt.lan.domain > pc1.lan.41977: [udp sum ok] 34385 q: AAAA? ip5045453.ah
cdn.com. 0/1/1 ns: ahcdn.com. SOA ns1.ahcdn.com. admin.ahcdn.com. 1461513600 216
00 3600 691200 38400 ar: . OPT UDPsize=4096 (112)
20:41:21.709636 IP (tos 0x0, ttl 64, id 21017, offset 0, flags [DF], proto TCP (
6), length 52)
    pc1.lan.42144 > ec2-23-23-164-218.compute-1.amazonaws.com.https: Flags [.],
cksum 0x3806 (correct), seq 0, ack 1, win 319, options [nop,nop,TS val 34994656
ecr 3746365642], length 0
20:41:21.824755 IP (tos 0x0, ttl 24, id 16888, offset 0, flags [DF], proto TCP (
6), length 52)
    ec2-23-23-164-218.compute-1.amazonaws.com.https > pc1.lan.42144: Flags [.],
cksum 0xccf7 (correct), seq 1, ack 1, win 80, options [nop,nop,TS val 3746368178
 ecr 34954229], length 0
^C
54 packets captured
54 packets received by filter
0 packets dropped by kernel
```

Obrázek 3.1: Výpis analyzátoru tcpdump

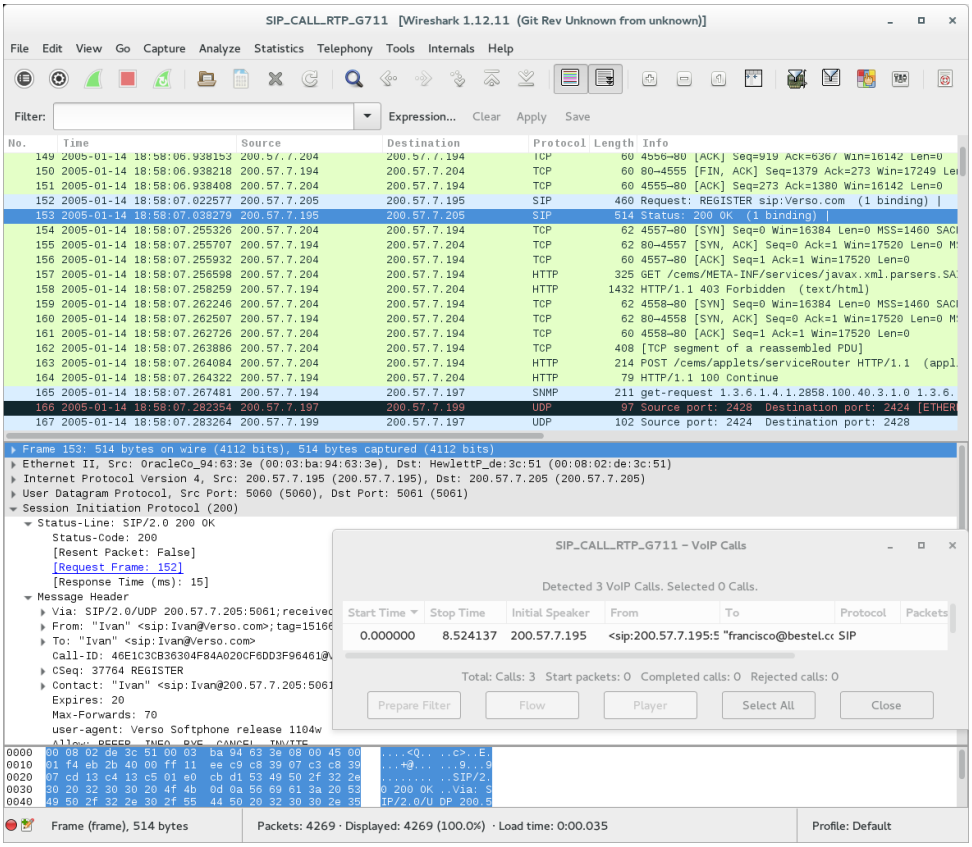
pomoci výrazů. Funkčnost tcpflow je zahrnuta v programu Wireshark, kde má uživatel možnost zobrazit data přenášená po TCP spojení.

3.5 Wireshark

Historie programu se datuje do roku 1997, kdy vznikla první verze pod názvem Ethereal díky Geraldu Combsovi. Po několika přerušeních vývoje v roce 1998 spojil Gerard Combs své síly s Richardem Sharpem a odstartovali vývoj dekódovacích modulů, specifických pro jednotlivé protokoly modelu TCP/IP. V roce 2006 došlo k přejmenování na jméno Wireshark a po deseti letech vývoje byla v roce 2008 uvolněna verze 1.0, která je považována za první kompletní verzi, pokrývající základní množinu protokolů a autoři ji představili téhož roku na konferenci SharkFest.

Wireshark je dnes považován za nejlepší open-source síťový analyzátor s grafickou nadstavbou. Nabízí velice podobné funkce jako tcpdump, navíc však obsahuje mnohem větší počet (stovky) nástrojů pro analýzu komunikačních protokolů a formátů, grafické uživatelské rozhraní a mnoho možností uspořádání a filtrování zobrazených informací [2].

Wireshark využívá knihovny Libpcap a WinPcap, dle zvolené platformy, a GTK+ widget toolkit pro implementaci uživatelského rozhraní. Funguje na různých operačních systémech jako Linux, Mac OS X, BSD, Solaris a Microsoft Windows. Program má i konzolovou (bez GUI) verzi zvanou TShark. Získaná data je možno filtrovat a ukládat do formátu pcap či dalších podporovaných. Za největší přednost programu je považováno přehledné zobrazení vrstev sítě.



Obrázek 3.2: Příklad grafického rozhraní programu Wireshark

3.5.1 Filtrování

Filtrování dat nabízí uživateli jednak hotové filtrovací profily a nebo možnosti jejich implementací. V zásadě lze vytvořit libovolné logické tvrzení pomocí předdefinovaných atributů a základních operandů, jako je porovnání, příslušnost do množiny, atd. Například `TCP only` nebo `no ARP and no DNS`. Kritéria lze skládat z primitivních atributů, jako je IP adresa zdrojového počítače, avšak se dají použit mnohem detailnější atributy, jako například příznaky TCP, call-id, atd. Aplikovaní filtrů je velmi jednoduché, a může být uděláno dvěma způsoby. Za prvé lze napsat kompletní výraz do políčka Filter, a nebo lze otevřít kontextové menu na určitém atributu a aplikovat ten parametr jako filtr. V druhém případě Wireshark napíše výraz do políčka Filter automaticky a dále ho jde upravit, případně doplnit.

3.5.2 Analýza rámců

Analýza přenesených rámců je řízena stromem dekodérů. Dle atributů se vybírá dekodér, který bude užit. Základními atributy pro detekci protokolu jsou pole ethertype na linkové vrstvě, IP protokol na vrstvě síťové a zdrojový a cílový port na vrstvě transportní jak u TCP, tak i UDP. Dekodéry užité na rozklad lze měnit a k tomu Wireshark nabízí i možnost naprogramování

takového modulu. Program v dnešní době je doplněn o velké množství modulu, a má podrobné návody, jak postupovat při vývoji dalších.

3.5.3 Statistické komponenty

K základním jednotkám statistických komponent Wiresharku patří stručný přehled obsahující informace o odchycených datech a informace o hierarchii použitých protokolů během komunikace. Dalším užitečným nástrojem je přehled komunikací, kde na základě protokolů lze naleznout informace o zdrojovém a cílovém uzlech. Mezi grafické nadstavby patří histogram velikostí paketů a velmi názorný Graf toků, jenž v průběhu času vizualizuje toky dat v obou směrech komunikace. Pro protokol TCP jsou v nabídce grafy RTT (Round trip time) a grafy velikostí přenášených dat v průběhu času.

3.6 Knihovna osip2.h

Knihovna osip2.h má za cíl poskytnout vývojářům multimediálního a telekomunikačního softwarů snadné a výkonné programátorské API rozhraní pro inicializaci a řízení spojení na základě protokolu SIP ve svých aplikacích [12]. Ke komponentům této knihovny patří parser samotných SIP zpráv, parser SDP zpráv a knihovna pro vytvoření SIP zpráv a sestavení SIP spojení. Knihovna GNU Osip je napsána v jazyce C a nemá žádné závislosti kromě standardní knihovny C. Tato knihovna může být použita například pro realizaci koncových bodů (IP soft telefony), proxy aneb B2BUA.

Projekt Osip byl zahájen v září 2000. Verze 0.8.0 již odpovídala RFC3261 [1]. Od verze 0.9.7 projekt se přejmenoval na osip2 a měl kompletně přepracované API rozhraní z účelem jeho zjednodušení. Avšak mnozí uživatelé doposud hodnotí ten produkt jako příliš složitý k použití.

Z hlediska této práce je nutné zdůraznit, že projekt osip2.h poskytuje samotný SIP parser, napsaný v jazyce C, který je schopen rozebrat SIP žádosti a odpovědi.

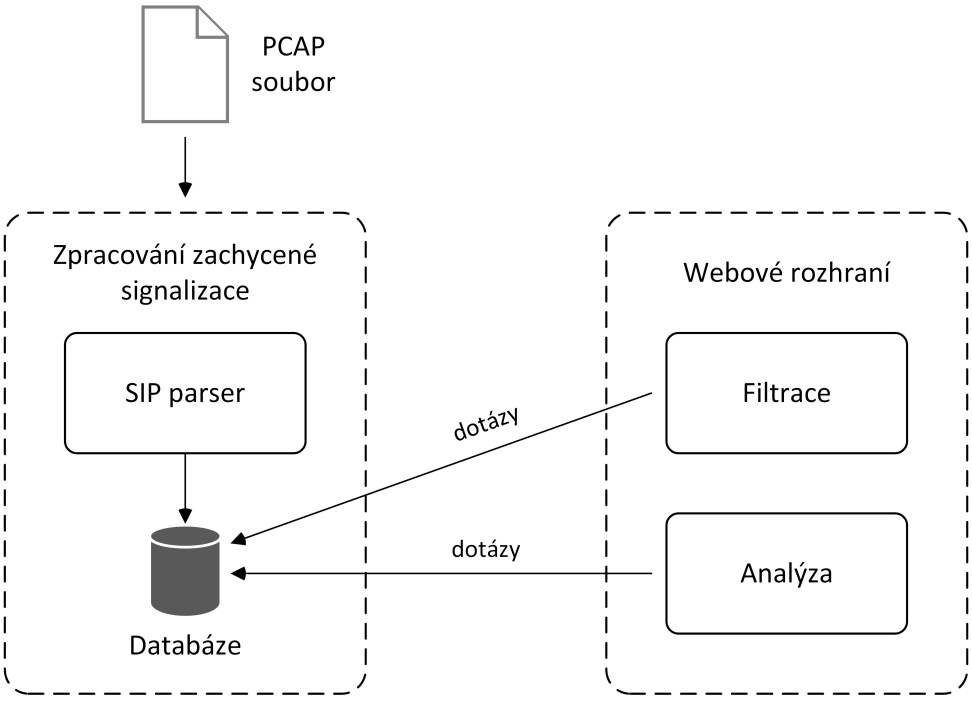
3.7 Nastroj Scapy

Scapy je interaktivní nástroj pro manipulace s pakety, generaci paketů, skenovaní sítě, analytický nástroj, paketový sniffer. Tento modul se velmi často využívá při vývoji softwaru pro analýzu a odposlech síťového provozu a taký útoky na sítě. Sada podporovaných protokolů není obrovská ale v oficiální dokumentaci [13] je velmi detailně popsán postup pro vytváření modulu pro další protokoly. Knihovna Scapy je napsána výhradně pro jazyk Python, a dá se ji tam vyžit bez použiti wrapperu.

Kapitola 4

Návrh systému

Kapitola se zabývá návrhem aplikace, která bude provádět zpracování SIP signalizace a získávání informací z ní. Návrh počítá s rozdělením systému analýzy pcap souboru do dvou funkčních bloků. V prvním bloku se provádí zpracování SIP paketů v nahraném pcap souboru a uložení výsledků do databáze. Druhá část potom využívá informací ze zpracovaných síťových dat v prvním bloku a má možnost nad nimi dotazovat (viz obrázek 4.1). Výsledky dotazů jsou zobrazovány v textové a grafické podobě a poskytují uživateli informace o SIP provozu. Navíc k tomu má uživatel nástroj pro hledání určitých hovorů.

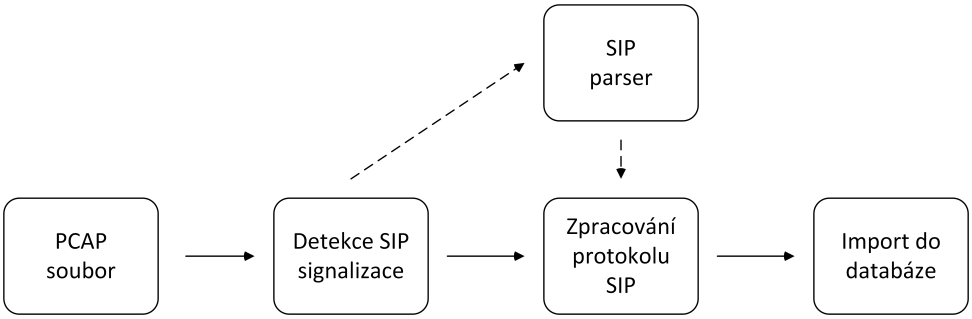


Obrázek 4.1: Funkční bloky navrženého systému

4.1 Zpracování pcap souboru

Jednotlivé kroku zpracování pcap souboru je vidět na obrázku 4.2. Klient nejprve nahraje zvolený pcap soubor na server. Následně spustí úlohu parsování SIP signalizace. Tato úloha na zpracování pcap souboru pracuje na následujícím principu. Ze vstupního pcap souboru uloženého na pevném disku čtou jednotlivé pakety, které se následně zasílají do fáze zpracování.

Zde se provede rozbalení přes jednotlivé vrstvy modelu TCP/IP. Nejdříve se z paketu vytvoří datová jednotka na linkové vrstvě. Z ní se následně získá IP datagram, v němž je uložena zdrojová a cílová IP adresa. V IP datagramu je v datové části uložen TCP nebo UDP paket. Tento paket se na transportní vrstvě identifikuje a získává se z něj zdrojový a cílový port. Dále probíhá zpracování samotného SIP paketu, který se identifikuje pomocí dobře známých a registrovaných portů.



Obrázek 4.2: Architektura zpracování pcap souboru

Takovým způsobem se dostávají záznamy o všech SIP paketech, které se poté importují do SQL databáze.

4.2 Zobrazení výsledků

Druhý funkční blok aplikace poskytuje uživatelské rozhraní, které slouží pro vizualizaci a analýzu dat ze zpracovaného pcap souboru. Vizualizace se provádí pomocí tabulky, která zahrnuje informace o všech SIP pakety z původního pcap souboru. Každý identifikovaný SIP paket je reprezentován jedním záznamem v tabulce. Navíc jsou zde představena analytická data, například počet všech úspěšných hovorů.

Uživatel může vyfiltrovat zpracovaná data podlé takových parametrů jako Call-ID, From a To. Pro jednotlivé hovory (na zakladě parametru Call-ID) se zobrazuje ukázkový diagram a nabízejí se filtry pro získání SDP potoků této konverzace v programu Wireshark.

Kapitola 5

Implementace řešení

Na základě zkoumání problematiky SIP komunikace a možností její zpracování je navrženo řešení pro systém pasivní analýzy. Výsledný systém je implementován jako aplikace s webovým uživatelským rozhraním. Aplikace umožňuje zadávat pcap soubory se zachycenou síťovou komunikací ke zpracování na serveru. Další funkcionalitou je poskytování výsledků tohoto zpracování s možnostmi následné analýzu a získávání dalších informací o zachyceném provozu.

Implementace výsledného systému využívá několik technologií. Jejich výběrem a popisem začíná tato kapitola. Dále je popsáno implementované řešení. Zde je vidět rozdělení navrženého systému do logicky souvisejících bloků. Prvním z nich je implementace zpracování pcap souboru se zachyceným SIP provozem. V této části je uvedena implementovaná struktura tohoto zpracování s rozebráním jednotlivých fází. Dále nasleduje popis použité struktury databáze, kam se přímo ukládají výsledky zpracování. Kapitola je ukončena popisem jak bylo vyřešeno uživatelské rozhraní, hledání a analýza výsledků.

5.1 Použité technologie

Významnou částí implementace navrženého systému je realizace samotného SIP parseru. Na základě analýzy technik a prostředků pro zpracování pcap souboru provedené v kapitole 3 a zadaní diplomové práci, které specifikuje navržení systému primárně pro SIP signalizaci, lze stanovit 2 nejvhodnější způsoby implementace:

1. použití knihovny osip2.h;
2. použití nastroje Scapy.

První způsob má výhodu v tom, že obsahuje hotový parser protokolu SIP. Ale, jak bylo zmíněno v třetí kapitole, API rozhraní této knihovny je považováno za velmi složité. Pro využití této knihovny je nutné mít nadstandardní znalosti jazyka C, což je další nevýhodou. Naopak druhý způsob předpokládá využití jazyka Python, programování ve kterém je snadnější pro běžného uživatele. Mimoto nastroj Scapy je uznáván za poměrně jednoduchou manipulaci s pakety. Nevýhodou v tomto případě je to, že SIP protokol není

rozpoznáván pomocí Scapy. Avšak se dá zavést podporu jakéhokoli protokolu v Scapy pomocí napsání rozšiřujících modulů.

Vzhledem k tomu, že SIP je textový protokol, a Python je jeden z pohodlnějších jazyků pro práci s textem, bylo rozhodnuto vyžití nastroje Scapy a napsaní vlastního modulu pro rozeznávání protokolu SIP v jazyce Python.

Pro ukládání zpracovaných dat se v daném případě hodí jakákoli relační databáze. Pro tento účel byla zvolena databáze MySQL. Uživatelské rozhraní bylo rozhodnuto dělat ve webové podobě. Stránky aplikace obsahují HTML a PHP kód, a pro přístup k nim je použit webový server Apache.

5.1.1 Jazyk Python

Jazyk Python začal vyvíjet v roce 1990 Guido van Rossum na Stichting Mathematisch Centrum (CWI) v Nizozemsku. Hlavní platformu autor převzal z jazyka ABC, který dosud vyvíjel. V roce 1995 se vývoj Pythonu přesunul do Corporation for National Research Initiatives (CNRI) v USA. V roce 2000 byl vývojáři Pythonu založen tým BeOpen PythonLabs, který se přesunul pod firmu Zope Corporation. V roce 2001 byla založena nezisková organizace Python Software Foundation (PSF), která tvoří základ vývojářů Python dodnes[14].

Mezi hlavní rysy Pythonu patří to, že je objektově orientovaný, interpretovaný a platformě nezávislý (je vyvinut jako open source projekt, nabízí instalační balíčky pro všechny běžné platformy – Windows, Unix, Mac OS). Mezi další významné vlastnosti patří vysoká modularita, jednoduchá syntaxe a existence velkého množství knihoven a modulů[15]. Python se dále vyznačuje efektivními funkcemi pro zpracování textu, práce se soubory a zpracováním (např. třídění) seznamů. Kromě toho je nutno zmínit o jednoduché a snadně čitelné syntaxi kódu. V porovnání s dalšími jazyky má odlišnou blokovou strukturu kódu – v jiných jazycích se blokové struktury (např. určení těla cyklu, podmínky nebo funkce) definují uzavíráním kódu do závorek, v Pythonu se využívá mezer a odsazování kódu od začátku řádku.

Po instalaci standardních knihoven jazyka, je dostupný IDE (integrated development environment), ve kterém lze spouštět jednotlivé příkazy. Mimo tohoto způsobu lze vytvářet programovací skripty (soubory *.py), které vykonávají soubor příkazů v uvedeném pořadí a tím vytvářejí aplikaci. Tento způsob vývoje nese řadu výhod (okamžité testování funkčnosti, zpracování zdrojového kódu během spuštění, přehlednost apod.).

Python nabízí velké množství rozšíření v podobě modulů, které jsou vytvářeny nezávisle na jádře jazyka. Mezi moduly patří např.: NumPy, SciPy, NumArray, PyWin32, PySerial, MatPlotLib, Pil, Pysk, PySsh, M2Crypto a mnoho dalších. Moduly se instalují podle typu hostujícího operačního systému a ve skriptech se projde jejich import pomocí příkazu import. Tímto postupem se zpřístupní veškeré funkční metody daného modulu. Při realizaci aplikace pro zpracování dat zachyceného síťového provozu se využilo doplňkových modulů, které nejsou součástí programového jádra jazyka. Například pro funkci importu do databáze byl použit modul MySQLdb [16].

5.1.2 MySQL databáze

MySQL patří k nejrozšířenějším relačním databázovým serverům, které jsou založený na dotazovacím jazyku SQL. To hlavně proto, že je distribuován pod opensource licencí a jeho výkon je v porovnání s konkurenčními databázovými servery velmi dobrý. V minulosti MySQL nenabízel takové funkce a možností jako konkurenční systémy, avšak se to změnilo s příchodem verze 5, která přinesla podporu uložených procedur a funkcí a vytváření trigerů[17]. V systému je použita MySQL databáze verze 5.6 která plně dostačuje pro potřeby vývoje tohoto projektu.

Jazyk SQL (Structured Query Language) je dotazovací jazyk standardizovaný pro práci s daty v relačních databázích. SQL je založen na množinových a relačních operacích. Typickým příkladem dotazu je příkaz pro výběr:
`SELECT` atributy `FROM` databáze `WHERE` podmínka

Jedná se o příkaz pro manipulaci s daty, mezi které patří i další dotazy (INSERT, UPDATE, DELETE, atd). Tyto příkazy se označují zkratkou DML – Data Manipulation Language. Další skupina příkazů slouží k tvorbě struktury databáze. Jedná se o příkazy, CREATE, ALTER a DROP. Tato skupina nese zkrácené označení DDL – Data Definition Language. Do poslední skupiny patří příkazy pro řízení přístupových práv a řízení transakcí. Jsou označovány jako DCL – Data Control Language.

5.1.3 Jazyk PHP

PHP (Hypertext Preprocessor) je skriptovací jazyk používaný hlavně pro tvorbu dynamických webových aplikací. Jazyk PHP lze zařadit do struktury jazyka HTML, XHTML, či WML. Procesor jazyka PHP běží na HTTP serveru. Při požadavku klienta na PHP soubor je ten soubor nejprve zpracován PHP procesorem, který vykoná vložený kód a vytvoří HTML dokument. Pak je ten dokument poslán klientovi jako odpověď. Klient tedy dostává jako odpověď pouze stránku s HTML výsledným kódem. Je tím pádem zaručena neviditelnost a bezpečnost kódu skriptu [18]. PHP v sobě kombinuje vlastnosti více programovacích jazyků (Perl, C, Java, Pascal) a nabízí syntaxi a funkce jazyka jednoduché na pochopení. Pro jazyk PHP existuje spousta rozšiřujících knihoven pro práci s různými druhy databází, zpracování obrázků, XML souborů apod.

Základy PHP sahají do roku 1994, kdy jej Rasmus Leordorf vytvořil pro svou osobní potřebu přepsáním z Perlu do jazyka C. Od té doby došlo k dalšímu vývoji, přičemž je nyní k dispozici PHP verze 5, která je postavena na objektově orientovaném přístupu a má velkou podporu ze strany programátorské komunity. V současnosti je jazyk PHP nejvíce využíván v kombinaci s databázovým serverem MySQL a webovým serverem Apache.

5.1.4 Apache

Apache2 je webový server dostupný jako open source pro různé platformy. Komunikace v síti obecně funguje na bázi dotazů a odpovědí. Návštěvníci www stránek jsou vybaveni webovým prohlížečem a připojují se prostřednictvím Internetu k webovým serverům. Server dotaz převezme, vyhodnotí a následně pošle odpověď zpět klientovi. Apache je momentálně nejpoužívanější webový server dnešní doby. Je výkonný a spolehlivý, obsahuje ohromné konfigurační možnosti, umožňuje snadné propojení se zmiňovaným PHP a MySQL databází. Vice o Apache lze dozvědět z webových stránek projektu [19].

5.1.5 phpMyAdmin

Ovládat databázi MySQL a zadávat SQL příkazy lze několika způsoby. Lze využít například přikazovací řádek a nebo grafickou nástavbu. Nejpoužívanější aplikací pro jednoduchou správu databáze MySQL je aplikace phpMyAdmin. Projekt phpMyAdmin [20] je vlastně souborem skriptů PHP umožňující jednoduchou správu obsahu databáze MySQL prostřednictvím webového rozhraní buď z lokální stanice nebo odkudkoliv pomocí sítě Internet. Umožňuje vytvářet/rušit databáze, vytváře/upravovat/rušit tabulky, provádět SQL dotazy a spravovat klíče. Jedná se o jeden z nejpopulárnějších nástrojů pro správu databáze.

5.2 Struktura aplikace

Aby zajistit snadnou správu systému pro analýzu SIP signalizace a možnost rozšířeni funkcionality dílčí části algoritmu jsou vyneseny do samotných souborů. Výsledný systém se pak skládá z několika souborů, jednotlivé funkce kterých jsou popsány v tabulce 5.1.

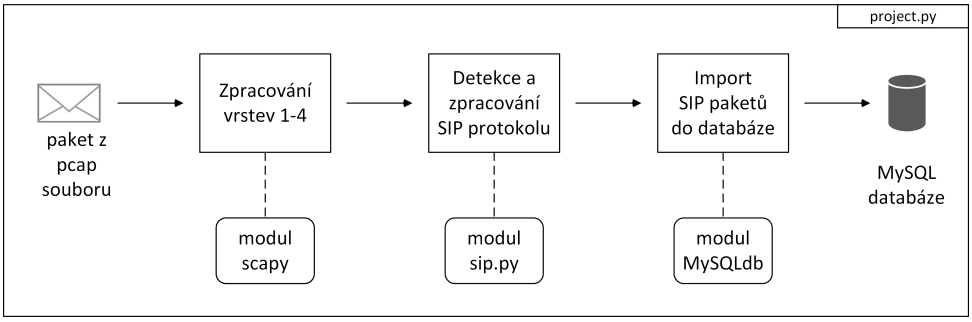
Soubor	Popis
sip.py	modul pro nastroj Scapy, který zavadí podporu SIP protokolu
project.py	hlavní soubor projektu, který analyzuje pcap soubor a ukládá výsledky do databáze
search.php	webové rozhraní, hlavní stránka projektu, zobrazuje zpracované výsledky ve formátu tabulky a dává přístup ke nástrojům hledání a analýzy
results.php	webové rozhraní, tato stránka zobrazuje výsledky analýzy, které uživatel muže následně provést nad zpracovanými daty
config.php	webové rozhraní, skript poskytuje připojení ke databázi se zpracovanými daty
drop_db.php	webové rozhraní, skript umožňuje smazaní databáze se zpracovanými daty

change_db.php	webové rozhraní, skript umožňuje přepnutí na jinou databázu
graph.php	webové rozhraní, tento skript slouží pro kreslení ukázkového diagramu konverzace
run.php	webové rozhraní, tento skript slouží pro spouštění zpracování pcap souboru na lokálním disku serveru.

Tabulka 5.1: Soubory implementovaného systému

5.3 Zpracování pcap souboru

Cílem této části je objasnit, jakým způsobem bylo implementováno zpracování pcap souboru se zachycenou síťovou komunikací. Je popsáno, co se provádí v jednotlivých fázích zpracování a jaké datové jednotky se přenáší mezi jednotlivými fázemi. Celé řešení zpracování, kdy na vstupu máme pcap soubor se zachycenou SIP signalizací a výsledek zpracování toho souboru ukládáme do databáze, ilustruje obrázek 5.1.

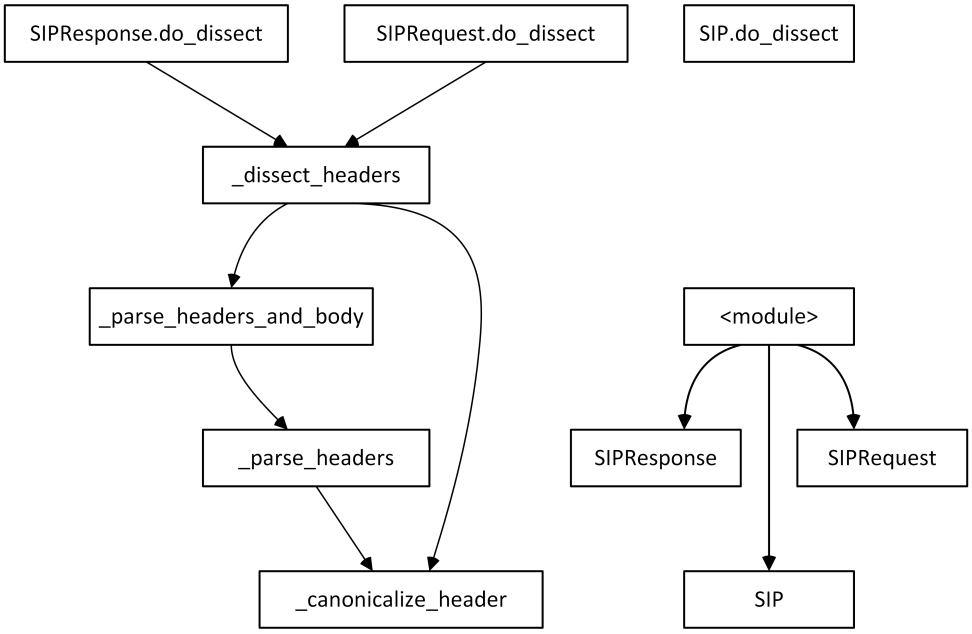


Obrázek 5.1: Schema zpracování pcap souboru

Dále jsou popsány a rozebrány jednotlivé části tohoto zpracování.

5.3.1 Modul sip.py

Modul sip.py je program, který je napsán autorem této práce v jazyce Python, z účelem zavedení do nastroje Scapy podpory SIP protokolu. Při napsaní byla použita on-line dokumentace projektu Scapy pro vyvíjení vlastních modulu [13].



Obrázek 5.2: Funkce a třídy modulu sip.py

Modul sip.py obsahuje 3 třídy: `SIP`, `SIPRequest` a `SIPResponce` (obr. 5.2). Paket nejprve postupuje do třídy `SIP`, která pomocí funkce `guess_payload_class` rozhoduje buď je to žádost a nebo odpověď. Uvnitř funkce `guess_payload_class` se provádí klasické hledaní pomocí Python modulu `re` s využitím speciálních masek.

Pro hledání žádosti byl požit výraz, který lze vidět na výpisu 5.1. Tento vyraz odpovídá takové struktuře řádku Start-Line, když na začátku jde metoda, pak nasleduje URI a na konci řádku je umístěná verze protokolu.

```
re.compile(
    r"^(?:INVITE|ACK|BYE|CANCEL|OPTIONS|REGISTER|PRACK|
        SUBSCRIBE|NOTIFY|PUBLISH|INFO|REFER|MESSAGE|UPDATE) "
    r"(?:.+?) "
    r"SIP/\d\.\d$"
    )
```

Výpis 5.1: Hledání SIP žádosti

Pro hledání odpovědi je použit výraz z výpisu 5.2, který odpovídá takové struktuře řádku Start-Line, když na začátku jde verze protokolu, pak nasleduje číselná odpověď a na konci řádku je umístěn její textový popis.

```
re.compile(r"^SIP/\d\.\d \d\d\d .*$")
```

Výpis 5.2: Hledání SIP odpovědi

Po rozhodnutí paket se předává do odpovídající třídy (SIPRequest anebo SIPResponce). Třídy obsahují popis všech možných položek dle RFC 3261[1], které se můžou objevit v hlavičce SIP paketu (výpis 5.3).

```
fields_desc = [StrField("StatusLine", "", fmt="H"),
    StrField("Accept", "", fmt="H"),
    StrField("Accept-encoding", "", fmt="H"),
    StrField("Accept-language", "", fmt="H"),
    StrField("Alert-info", "", fmt="H"),
    StrField("Allow", "", fmt="H"),
    StrField("Authentication-info", "", fmt="H"),
    StrField("Authorization", "", fmt="H"),
    StrField("Call-id", "", fmt="H"),
    StrField("Call-info", "", fmt="H"),
    StrField("Contact", "", fmt="H"),
    StrField("Content-disposition", "", fmt="H"),
    StrField("Content-encoding", "", fmt="H"),
    StrField("Content-language", "", fmt="H"),
    StrField("Content-length", "", fmt="H"),
    StrField("Content-type", "", fmt="H"),
    StrField("Cseq", "", fmt="H"),
    StrField("Date", "", fmt="H"),
    StrField("Error-info", "", fmt="H"),
    StrField("Expires", "", fmt="H"),
    StrField("From", "", fmt="H"),
    StrField("In-reply-to", "", fmt="H"),
    StrField("Max-forwards", "", fmt="H"),
    StrField("Mime-version", "", fmt="H"),
    StrField("Min-expires", "", fmt="H"),
    StrField("Organization", "", fmt="H"),
    StrField("Priority", "", fmt="H"),
    StrField("Proxy-authenticate", "", fmt="H"),
    StrField("Proxy-authorization", "", fmt="H"),
    StrField("Proxy-require", "", fmt="H"),
    StrField("Record-route", "", fmt="H"),
    StrField("Reply-to", "", fmt="H"),
    StrField("Require", "", fmt="H"),
    StrField("Retry-after", "", fmt="H"),
    StrField("Route", "", fmt="H"),
    StrField("Server", "", fmt="H"),
    StrField("Subject", "", fmt="H"),
    StrField("Supported", "", fmt="H"),
    StrField("Timestamp", "", fmt="H"),
    StrField("To", "", fmt="H"),
    StrField("Unsupported", "", fmt="H"),
    StrField("User-agent", "", fmt="H"),
    StrField("Via", "", fmt="H"),
    StrField("Warning", "", fmt="H"),
    StrField("Www-authenticate", "", fmt="H")]
```

Výpis 5.3: Všechny možné začatky řadku SIP hlavičky

Jak je vidět z obrázku 5.2, poté se zavolá funkce `_dissect_headers`, spolu se vnořenými funkcemi `_parse_headers_and_body`, `_parse_headers` a `_canonicalize_header`.

Pomocí funkce `_parse_headers_and_body` se provádí rozdělení paketu na hlavičku a tělo. Realizace této funkce využívá vlastnosti SIP paketu, že hlavička má být oddělená od těla zprávy jedním prázdným řádkem. Odpovídající kód v jazyce Python lze vidět z výpisu 5.4, kde `\x0d\x0a\x0d\x0a` je hexadecimální ekvivalent `\r\n\r\n`, což jsou dva po sobe jdoucí konce řádků, tedy prázdný řádek.

```
pattern = b"\x0d\x0a\x0d\x0a"
patternIndex = s.find(pattern)
headers = s[:patternIndex + len(pattern)].decode("utf-8")
body = s[patternIndex + len(pattern):]
```

Výpis 5.4: Detekce těla SIP zprávy

Pro snadné pochopení logiky funkce `_parse_headers` je uveden pseudokód 1. Na začátku text, který patří do hlavičky SIP zprávy, je rozdělen na řetězce pomocí příkazu `s.split("\r\n")`. Poté je každý řetězec následně rozdělen na dvě části podle symbolu dvojtečka. První část (označená v kódu jako `key`) se porovnává s názvy položek v odpovídající třídě matice fields_desc (výpis 5.3). Pokud proměna `key` je nalezena, zapíše se do vytvořeného na začátku seznamu `headers_found` spolu s odpovídající hodnotou - druhou částí (označenou v kódu jako `value`). Realizace se počítá s tím, že například položka `Via` se může v hlavičce objevit několik krát. Proto v kódu je zavedená kontrola, jestli proměna `key` již je v seznamu `headers_found`. Pokud ano, tak se tato položka doplní jen o další data.

Data: Hlavička SIP
Result: Rozebraná SIP hlavička v souladu s fields_desc (výpis 5.3)

Inicializace matice *záhlaví*
záhlaví ← rozdělená vstupní data podle symbolu konec řádku (\r\n)
Inicializace asociativního pole (dictionary) *nalezená_záhlaví*
for *každý element v matice záhlaví* **do**
  [*key*,*value*] ← rozdělení *elemet*u podle symbolu dvojtečka (:)
  if *klíč je v nalezená_záhlaví* **then**
    nalezená_záhlaví[*klíč*] doplníme o proměnou *hodnota*
  else
    nalezená_záhlaví ← [*key*,*value*]
  end
end

Algoritmus 1: Parsování SIP hlavičky - funkce _parse_headers

Funkce `_canonicalize_header` se používá pro převod na velká písmena. Tím pádem všechny porovnání dvou řetězců probíhají ve velkých písmenech, aby zabránit nesouladu písmen. Funkce `_dissect_headers` nahrává data ze seznamu `headers_found` do položek odpovídající třídy matice fields_desc (výpis 5.3).

5.4 Ukládání výsledků do databáze

Způsob zpracování výstupu z modulu sip.py je ovlivněn výstupním formátem představení paketů v programním prostředí jazyka Python. Díky tomu že ve třídách SIPRequest a SIPResponce jsou definovány všechny možné parametry záhlaví SIP paketu, lze ke každému z nich obrátit pomocí příkazu `<packet>.sprintf("{SIP:%SIP.<jméno_parametru>%}")`. Ve výsledné aplikaci je potřeba ukládat výstup úlohy do databáze MySQL. Využívá se přitom modul MySQLdb, který obsahuje funkce `sql_connection` pro připojení k databázi. Na vstupu aplikace je jako atribut požadován soubor s údaji pro připojení - jméno, heslo, název databáze a její adresa v sítě (používá se loopback adresa 127.0.0.1 pro lokální uložení dat). Až se sestaví spojení, proběhne naplnění tabulky SQL příkazem INSERT.

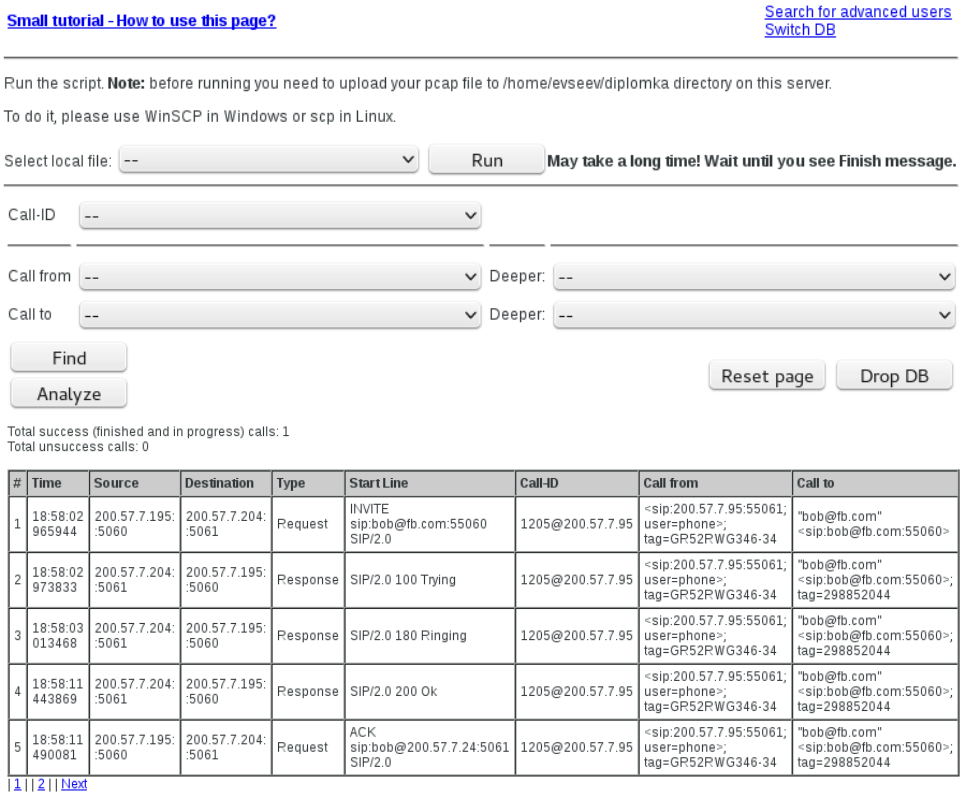
5.4.1 Struktura databáze

Zpracovaná data síťové komunikace se na konci zpracování SIP paketu ukládají do databáze MySQL. Výsledná databáze tedy obsahuje pouze jednu tabulku, která se nazývá SIPheaders (obr. 5.3). Dále se v aplikaci tato databáze využívá webovým rozhraním pro zobrazovaní výsledků, jejich filtraci a analýzu. Návrh této databáze se prováděl souběžně s návrhem entit, jejíchž data se do databáze budou ukládat. Entity totiž představují definované položky tříd SIPResponce a SIPRequest.

Obrázek 5.3: Struktura databáze v aplikaci

5.5 Webové rozhraní

Webové rozhraní poskytuje ovládací prvky, které slouží k vytváření uživatelem úkolů analýzy a zobrazování výsledků zpracování. Základním jazykem pro jeho realizaci byl zvolen PHP. Dále jsou rozebrány významné částí webového rozhraní.



Obrázek 5.4: Domovská stránka systému pro analýzu SIP signalizace

Hlavní elementem na domovské stránce aplikace (obr. 5.4) je tabulka výsledků. Tabulka dává přehled zpracovaných paketů, kde každý řádek obsahují informace o SIP paketů. Tato tabulka je vytvářena pomocí následujícího kódu:

```
<?php
$sql = "SELECT id,time,src_ip,dest_ip,src_port,dest_port,
    sip_type,start_line,callid,from1,to1,from2,to2 FROM ".
    $SETTINGS["data_table"]." WHERE id>0 LIMIT $start_from,
    $num_on_page ";
$sql_result = mysqli_query ($connection, $sql) or die ('
    request "Could not execute SQL query" '.$sql);
if (mysqli_num_rows($sql_result)>0) {
    while ($row = mysqli_fetch_assoc($sql_result)) {
        ?>
        <tr>
        <td><?php echo $row["id"]; ?></td>
```

```
        <td><?php echo $row["time"]; ?></td>
        <td><?php echo $row["src_ip"].':'.$row["src_port"];
            ?></td>
        <td><?php echo $row["dest_ip"].':'.$row["dest_port"
            ]; ?></td>
        <td><?php echo $row["sip_type"]; ?></td>
        <td><?php echo $row["start_line"]; ?></td>
        <td><?php echo $row["callid"]; ?></td>
        <td><?php echo $row["from1"]; ?></td>
        <td><?php echo $row["to1"]; ?></td>
        </tr>
        <?php
    }
}
?>
```

Výpis 5.5: Tabulka domovské stránky aplikace

Z výpisu 5.5 je vidět, že nejsou-li aplikované žádné filtry, tak se zobrazuje základní tabulka obsahující záznamy databáze, zkrácené na sloupečky ID, čas, IP adresa a port odesilatele a cíle, druh zprávy, Start Line, Call-ID, URI volajícího a volaného. Tabulka je zobrazována po stránkách.

Aplikováním různých filtrů lze zobrazit jenom data, o nichž má zájem uživatel. Realizované způsoby filtrace zpracovaných dat:

- Filtrace podle call-id. SQL dotaz:

```
$sql = "SELECT callid FROM ".$SETTINGS["data_table"]."
   WHERE id>0 GROUP BY callid ORDER BY callid";
```

- Filtrace podle volajícího (poličko From). SQL dotaz:

```
$sql = "SELECT from1 FROM ".$SETTINGS["data_table"]."
   WHERE id>0 GROUP BY from1 ORDER BY from1";
```

- Filtrace podle volaného (poličko To). SQL dotaz:

```
$sql = "SELECT to1 FROM ".$SETTINGS["data_table"]."
   WHERE id>0 GROUP BY to1 ORDER BY to1";
```

Jak lze vidět z obrázku 5.3 jsou v databázi založeny dva sloupce From - from1 a from2. Oba dva obsahují SIP URI volajícího, avšak rozdíl spočívá v tom, že from1 nese k tomu unikátní označení (tag) hovorů. Například jestli pcap soubor obsahuje jenom dva hovory, které byly inicializovaný Bobem, všechny hodnoty sloupce from2 budou stejné na rozdíl od sloupce from1, který bude mít dvě různé hodnoty - `Bob <sip:bob@domena.com:5060>;tag=XXXXXXXXXX` a `Bob <sip:bob@domena.com:5060>;tag=YYYYYYYYYY`. Tím pádem, v závislosti na tom, který sloupec je zvolen pro SQL dotaz, lze dostat různou hloubku hledání. Výšeuvedené platí i pro sloupec To.

Je-li aplikován jakýkoliv filtr, lze vyfiltrovaná data analyzovat. Analýza určující stav hovoru je založena na postupných SQL dotazech, během kterých

se kontrolují sloupce Start-line, CSeq atd. Její logika je rozebrána v další částí této kapitoly. Analýza hovoru se zobrazuje na zvláštní stránce results.php.

Skripty spuštění zpracováni pcap souboru, připojení k databázi a její vymazaní a taky kreslení grafu jsou realizovány ve zvláštních souborech - run.php, config.php, drop_db.php a graph.php a pak jsou připojeny jako moduly příkazem `include`, například `include("config.php")`.

5.5.1 Logika analýzy hovorů

Jak již bylo řečeno, analýza stavu hovoru je realizována pomocí postupných SQL dotazů. Implementovaný algoritmus se při zpracovávání vstupních dat řídí logikou uvedenou v pseudokódu 2.

Tento algoritmus se začíná kontrolou existence INVITE zprávy. V kladném případě budou provedeny dotazy na každý stav obyčejného stanovení spojení (tedy stavy 180 Ringing, 200 OK a ACK) a jeho zrušení (BYE a 200 OK). Jakmile předpokládaný stav při stanovení spojení bude chybět, algoritmus ověří, jestli nebyly obdrženy chybné zprávy (tedy zprávy s kódy 4XX, 5XX nebo 6XX). V případě že INVITE zpráva nebyla nalezena, je nutné zkontrolovat jestli nejde o registraci terminálu. Pokud ne, se kontrolují chybné hlášky.

Každá podmínka je samotný SQL dotaz. Příklad dotazu v jazyce PHP pro kontrolu paketu 200 OK je vidět z výpisu 5.6. Ostatní dotazy jsou realizovány stejným způsobem.

```
$sql = "SELECT id FROM ".$SETTINGS["data_table"]." WHERE
    callid = '".mysqli_real_escape_string($connection,
    $callid)."' AND start_line LIKE '%200 O%' AND cseq LIKE
    '%INVITE%'"
```

Výpis 5.6: Hledaní odpovědi 200 OK na žádost INVITE

Podstatné je, že při kontrole takových paketů, jako například 200 OK, je nutné aby parametr CSeq měl hodnotu paketu na který tento 200 OK paket odpovídá. Jinak řečeno, pro jednotlivý Call-ID může byt několik 200 OK paketů, které lze rozlišit podle políčka CSeq.

Realizovaný algoritmus detailní pokrývá stanovení spojení. Kód může být snadné rozšířen zavedením dalších podmínek pro hledaní jednotlivých chybových hlášek.

```
Data: Call-ID
Result: Záver o stavu hovoru
if existuje INVITE zpráva pro tento Call-id then
    RESULT ← "jenom INVITE zpráva";
    if existuje 180 Ringing zpráva then
        RESULT ← "obdržena 180 Ringing zpráva";
        if existuje 200 OK zpráva s CSeq = INVITE then
            RESULT ← "obdržena 200 OK zpráva";
            if existuje ACK zpráva then
                RESULT ← "Hovor právě probíhá";
                if existuje BYE zpráva then
                    RESULT ← "Hovor ve stavu ukončení";
                    if existuje 200 OK zpráva s CSeq = BYE then
                        RESULT ← "Hovor je ukončen";
                    end
                end
            else
                if existuje CANCEL zpráva then
                    RESULT ← "Hovor je zrušen";
                else
                    if existuje 4XX nebo 5XX nebo 6XX zpráva then
                        RESULT ← "chyba, obdržena _kód_chyby_ zpráva";
                    end
                end
            end
        else
            if existuje CANCEL zpráva then
                RESULT ← "Hovor je zrušen";
            else
                if existuje 4XX nebo 5XX nebo 6XX zpráva then
                    RESULT ← "chyba, obdržena _kód_chyby_ zpráva";
                end
            end
        end
    else
        if existuje 100 Trying zpráva then
            RESULT ← "obdržena 100 Trying zpráva";
        else
            if existuje CANCEL zpráva then
                RESULT ← "Hovor je zrušen";
            else
                if existuje 4XX nebo 5XX nebo 6XX zpráva then
                    RESULT ← "chyba, obdržena _kód_chyby_ zpráva";
                end
            end
        end
    end
else
    if existuje REGISTER zpráva then
        RESULT ← "probiha registrace";
    else
        if existuje 4XX nebo 5XX nebo 6XX zpráva then
            RESULT ← "chyba, obdržena _kód_chyby_ zpráva";
        else
            RESULT ← "Nejde analyzovat hovor, pro další informace se podívejte na
              graf";
        end
    end
end
```

Algoritmus 2: Pseudokód analýzy hovoru

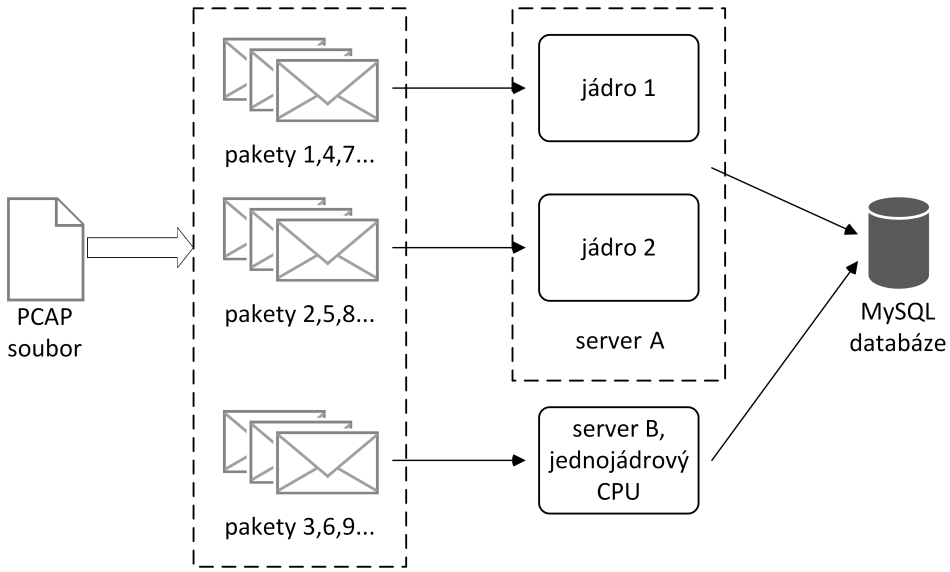
5.6 Distribuce zpracovávaní

V průběhu řešení diplomové práce byla prozkoumaná problematika distribuovaného způsobu zpracovávaní. Aplikace pracující s vetším objemem dat často vyžadují podporu vícejádrových procesoru. Tím lze zajistit paralelní zpracovaní menších bloků dat.

Jedna iterace algoritmu zpracování pcap souboru navrženým systémem je tvořena následujícími kroky:

1. vyčlenění k-ho paketu z pcap souboru;
2. dekódování hlaviček paketu;
3. v případě přítomnosti SIP hlavičky bude připraven a spouštěn SQL dotaz;
4. vyčlenění *(k+1)*-ho paketu ...

Zpracování jednotlivých SQL dotazů je vzájemně nezavilé, tím pádem lze využít paralelní výpočty za účelem zmenšení časové náročnosti algoritmu. Pro to je možné navrhnout systém s n jádry, který zajistí zpracování bloku n po sobe jdoucích paketů během jedné iterace. Jednotlivé jádra můžou být dokonce fyzicky rozdělená, viz obrázek 5.5. Na tom obrázku je představen systém rozdělení úlohy zpracování na dva servery, jeden z kterých má dvoujádrový procesor. Výsledná data se uchovají do jedné databáze, která může být dokonce i na zvláštním serveru.



Obrázek 5.5: Distribuované zpracování

K realizaci popsané metody v jazyce Python slouží modul Multiprocessing. Logika realizace je uvedena v pseudokódu 3. Příkaz `itertools` umožňuje prohledávat pcap soubor s požadovaným krokem N, který je určen počtem jader.

Pomocí funkce `ThreadPool()` se vytvoří množina vláken, do níž postupně zařadíme potoky pro každé jádro pomocí funkcí `map()` a `join()`.

Data: PCAP soubor
Result: Záznamy SQL databáze

Inicializace modulu *Multiprocessing*
`sip_analyzer(`*i*`)` **begin**
  for *each_packet itertools(PcapReader(Soubor), i, None, N)* **do**
    process_paket(each_packet)
  end
end
N ← počet jader
[*pool*] ← ThreadPool(N)
for *i in range(N)* **do**
  pool.map(sip_analyzer, *i*)
  pool.join()
end

Algoritmus 3: Vícejádrové zpracování v jazyce Python

Další možností realizace paralelního zpracování spočívá v rozdělení vstupního pcap souboru na části. Pak dostupná jádra systému budou zpracovávat jen tyto jednotlivé části. K realizaci metody je navržen skript v jazyce Bash. Pro rozdělení pcap souboru je využit příkaz `tcpdump`, kde klič `-C` specifikuje velikost menších souboru. Pro paralelní spuštění několika pcap parseru je vhodné použit paket GNU Parallel [21]. Navržený skript je uveden ve výpisu 5.7. Vstupními parametry uvedeného skriptu jsou pcap soubor pro analýzu a počet jader v systému.

```
#!/bin/bash

TMPPCAP='/tmp/tmppcap'

#Get the size of the input pcap file
FILESIZE=$(stat -c%s "$1")
#Get the size of temporary files
NEWSIZE=$(($FILESIZE / $2))
NEWSIZE=`echo "($NEWSIZE + 1000000)/1000000" | bc`

#Split the input pcap file to the small pcap files
tcpdump -r $1 -w $TMPPCAP -C $NEWSIZE

#Start processing, argument -P means use multicores
ls ${TMPPCAP}* | parallel -n 1 -P $2 ./project.py {}

#Remove temporary files
rm -f /tmp/tmppcap*
```

Výpis 5.7: Paralelní zpracování pomoci Bash skriptu

Kapitola 6

Instalace systému pro pasivní analýzu SIP signalizace a jeho použití

Jak je popsáno v předchozích kapitolách, navržený systém se skládá ze dvou funkčních bloků - Python skript pro zpracování pcap souboru a uložení výsledků do databáze a webového uživatelského rozhraní, napsaného v jazyce PHP. V této kapitole se rozebírá postup pro zprovoznění tohoto systému na čerstvé instalované Linux distribuce. Pro ilustraci byla použita distribuce Debian 7.10. Instalace pro jiné Linux distribuce jsou až na drobné výjimky zcela totožné. Součástí této kapitoly je krátký úvod do uživatelského rozhraní a představení možností systému pro analýzu SIP signalizace.

Zde je nutné uvést zkratku LAMP, která je velmi známá pro správce linuxových serverů. LAMP je zkratka, která v informatice označuje sadu svobodného softwaru používaného jako platforma pro implementaci dynamických webových stránek. Zahrnuje tyto technologie: Linux, Apache, MySQL a PHP; jednotlivé technologie mohou být nahrazeny alternativami. Protože uživatelské rozhraní vyvinutého systému představuje webovou stránku, dále uvedená instalace bude obsahovat instalaci LAMP[22].

6.1 Instalace

Celý postup je třeba začít instalací balíčků, které jsou nutnou prerekvizitou pro spuštění vytvořeného kódu. Za prvé k překladu souboru project.py a sip.py se potřebuje samotný baliček python, moduly Scapu a MySQLdb. Tedy v konzoli serveru se spustí následující příkaz jako root:

```
apt-get install python python-scapy python-mysqldb
```

Dalším komponentem je databáze. Samotná instalace MySQL serveru je zajištěna následujícím příkazem:

```
apt-get install mysql-server
```

Během instalace uživatel je požádán o zadání hesla pro administrátorský účet databáze. Po instalaci dojde ke spuštění MySQL serveru. Na konci je třeba splnit požadavky pro webové rozhraní. Jde o instalaci PHP a serveru Apache:

```
apt-get install php5 apache2
```

Po instalaci dojde ke spuštění Apache serveru. Dále je třeba nějakým způsobem PHP, MySQL a Apache propojit mezi sebou. O to se postará tyto balíčky:

```
apt-get install php5-mysql libapache2-mod-php5
```

Dalším krokem je vytváření uživatele a databáze pro aplikace.

```
mysql -u root -p
CREATE DATABASE SIPparser;
CREATE USER evseev;
GRANT ALL ON SIPparser.* TO evseev@localhost;
SET PASSWORD FOR evseev@localhost=PASSWORD('python');
exit;
```

Strukturu databáze SIPparser lze zavést ručně nebo pomocí souboru schema.sql, který předtím se nahraje na server programem scp.

```
mysql SIPparser < schema.sql -u evseev -p
```

Poslední změny v nastavení MySQL se tykají globální proměnné `sql_mode`. Nastavení této proměnné určuje případy, když při zápisu do políčka tabulky data přesahuje objem políčka. Protože někdy v políčku Via se můžou objevit relativně dlouhá data, je nutné stanovit, aby MySQL zachoval tam všechno, co mu podaří nahrát. Poté je nutné restartovat MySQL server.

```
echo "sql_mode=NO_ENGINE_SUBSTITUTION" >> /etc/mysql/my.cnf
service mysql restart
```

Poslední potřebný software je aplikace phpMyAdmin, kterou jako ostatní lze nainstalovat pomocí příkazu `apt-get install`:

```
apt-get install phpmyadmin
```

Nyní jsou všechny pomocné baličky instalovány a lze přenést soubory samotné aplikace na server. Proto se vytváří v domovském adresáře složka `diplomka`, a do ní se nahrává soubory `project.py`, `sip.py` a `sql_conn.txt`, poslední má v sobě údaje potřebné pro připojení k databázi. Do adresáře `/tmp/pcap` je potřeba nahrát pcap soubory pro další zpracovaní. Pro vzdálené nahrávaní se používá program SCP, nebo, v případě operačního systému Windows na straně uživatele, WinSCP.

```
mkdir diplomka
cd diplomka
scp evseev@78.128.195.234:/home/evseev/diplomka/project.py .
scp evseev@78.128.195.234:/home/evseev/diplomka/sip.py .
scp evseev@78.128.195.234:/home/evseev/diplomka/sql_conn.txt
    .
mkdir /tmp/pcap
```

Apache očekává webovou prezentaci v adresáři `/var/www`. Vytvoří se tam složka `dipl-sip`, a do ní se nahrají webové stránky aplikace.

```
mkdir /var/www/dipl-sip.com/
cd /var/www/dipl-sip.com/
scp evseev@78.128.195.234:/home/evseev/diplomka/web/* .
```

Nakonec se v adresáři `/etc/apache2/sites-available/` vytvoří soubor `dipl-sip.conf`, jehož obsah je představen výpisem 6.1.

```
<VirtualHost *:80>
    ServerAdmin evseev@dipl-sip
    ServerName dipl-sip
    ServerAlias www.dipl-sip
    DocumentRoot /var/www/dipl-sip/
    ErrorLog /var/www/dipl-sip/logs/error.log
    CustomLog /var/www/dipl-sip/logs/access.log combined
</VirtualHost>
```

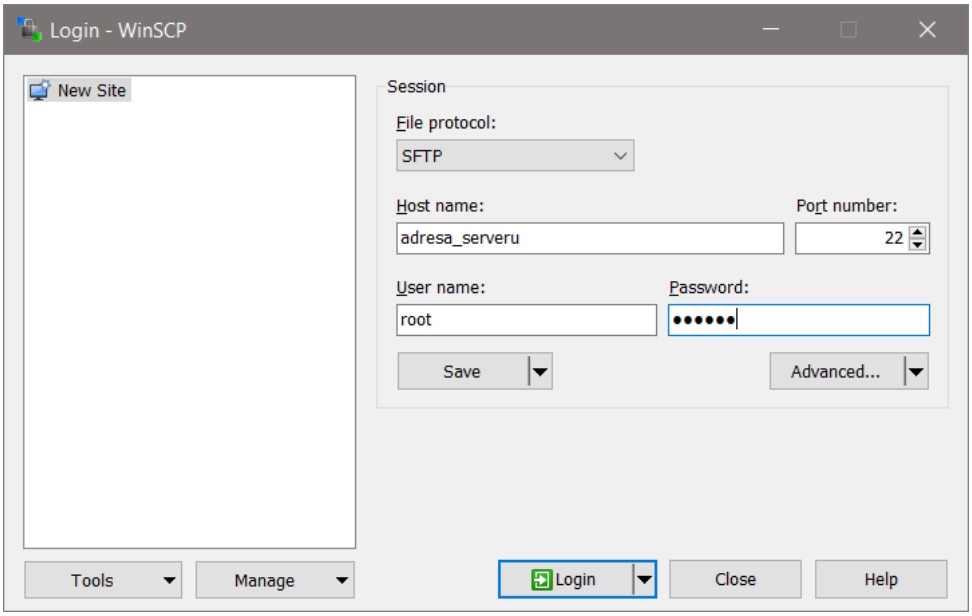
Výpis 6.1: Soubor dipl-sip.conf

Po restartu webserveru Apache systém je hotový k použití.

6.2 Použití systému

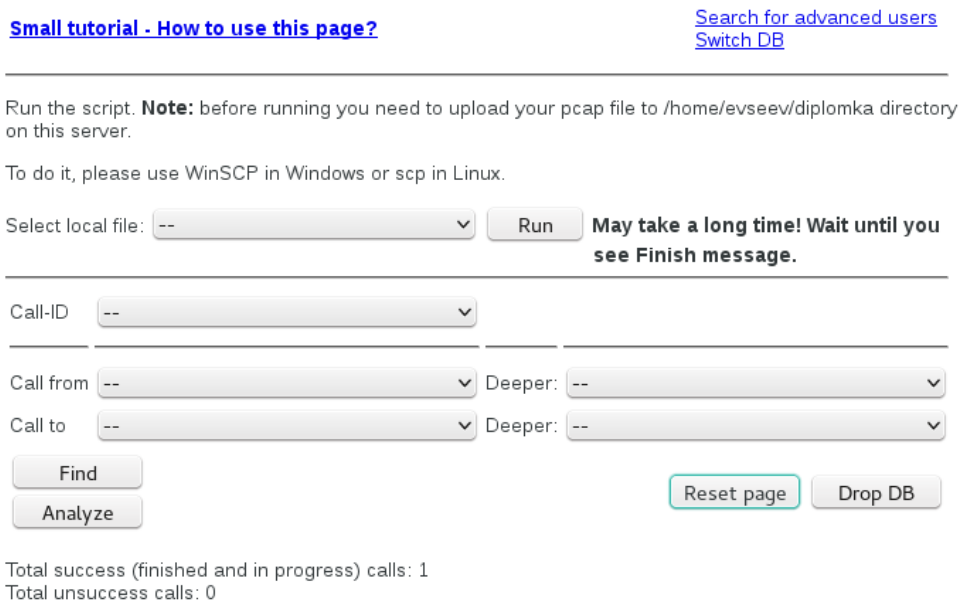
Na začátku je nutné nahrát pcap soubor na server pomocí scp klientu. Pro Windows uživatele lze použit WinSCP program. Po spuštění je třeba zadat IP adresu serveru, login a heslo (obr. 6.1). Po připojení se zobrazí standardní okno správce souborů, ve kterém je třeba zvolit pcap soubor a zmáčknout tlačítko Nahrát. Pro Linux uživatele stačí použit následující příkaz:

```
scp <cesta_do_pcap_souboru> root@<adresa_serveru>:/tmp/pcap
```



Obrázek 6.1: WinSCP

Grafické uživatelské rozhraní aplikace se skládá ze dvou hlavních webových stránek - domovské a stránky pro zobrazení výsledků analýzy. Pro přístup k uživatelskému rozhraní je nutno spustit webový prohlížeč a zadat adresu ve formátu http://<adresa_serveru>/dipl-sip/search.php. Hlavní ovládací prvky jsou rozděleny do tří částí: manipulace se soubory - zpracovávání, hledaní a statistická data (obr. 6.2).



Obrázek 6.2: Ovládací prvky

Prvním krokem je spuštění zpracování pcap souboru, který byl vzdáleně nahrán na server. Proto je třeba zvolit tento soubor z položky Select local file a potvrdit zpracovaní tlačítkem Run. Po zpracováni se zobrazí zpráva Finish a statistická data - časová náročnost zpracování. Teď obsah pcap souboru je dostupny v databázi a lze začít jeho analýzu.

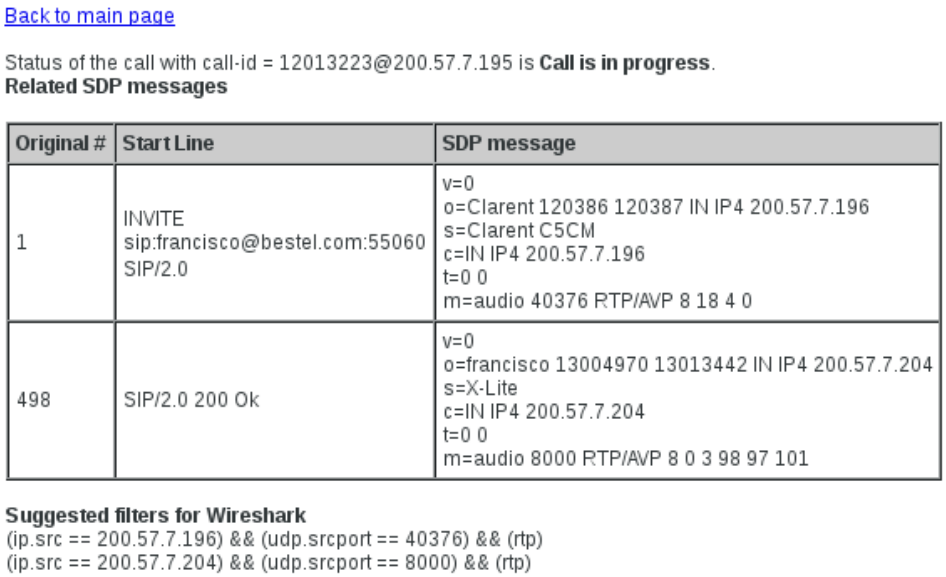
Na domovské stránce se zobrazí tabulka s informací o SIP zprávách. Tabulka dává přehled zpracovaných paketů, každý řádek zahrnuje popis jednoho SIP paketu. Dalším krokem je aplikace filtrů. Při filtraci zpracovaných dat jsou možné dvě varianty:

1. uživatel zná Call-ID hovoru, který chce analyzovat;
2. uživatel chce najít hovory (jeden nebo víc), které patří jednomu volajícímu nebo volanému.

První případ je zcela jednoduchý a předpokládá využití seznamu Call-ID, ve kterém uživatel najde patřičný hovor a poté potvrdí to tlačítkem Find. Následně by se měly zobrazit jenom řádky, odpovídající tomuto Call-ID. V tuto chvíli se lze tlačítkem Analyze pokusit analyzovat nalezené pakety. Ve případě scénáře číslo 2 uživatel může najít známý jemu SIP URI ve seznamech Call From nebo Call To. Jestli k tomu zná jednotlivý tag hovoru, může využit

seznamy Deeper a hned najít signalizaci k jednomu spojení. Zase tlačítkem Find potvrdí své rozhodnutí.

V případě hledání podle URI účastníka, seznam dostupných Call-ID bude taky vyfiltrován. To je uděláno pro případ, když počet hodnot ve seznamu Call-ID přesahuje jednotky tisíc a uživatel chce pochopitelně provést předběžnou filtraci. Každopádně po filtraci a zobrazení výsledků vyhledávání (paketů jednoho nebo více spojení) lze přejít na stránku analýzy, kde bude dostupna podrobnější informace. Lépe o ní napoví obrázek 6.3.


Back to main page

Status of the call with call-id = 12013223@200.57.7.195 is **Call is in progress**.
Related SDP messages

Original #	Start Line	SDP message
1	INVITE sip:francisco@bestel.com:55060 SIP/2.0	v=0 o=Clarent 120386 120387 IN IP4 200.57.7.196 s=Clarent C5CM c=IN IP4 200.57.7.196 t=0 0 m=audio 40376 RTP/AVP 8 18 4 0
498	SIP/2.0 200 Ok	v=0 o=francisco 13004970 13013442 IN IP4 200.57.7.204 s=X-Lite c=IN IP4 200.57.7.204 t=0 0 m=audio 8000 RTP/AVP 8 0 3 98 97 101

Suggested filters for Wireshark
(ip.src == 200.57.7.196) && (udp.srcport == 40376) && (rtp)
(ip.src == 200.57.7.204) && (udp.srcport == 8000) && (rtp)

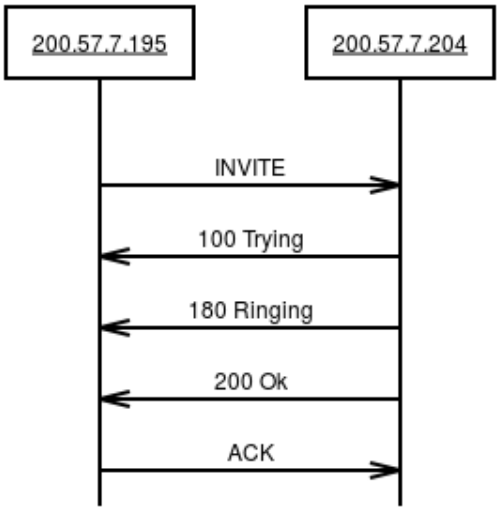
Obrázek 6.3: Stránka z výsledky analýzy

Z obrázku 6.3 je vidět, že první řádek dává závěr o stavu spojení, přičemž v případě analýzy několika Call-ID tento závěr bude zobrazen pro každý z nich. Poté uživatelovi bude představena tabulka existujících SDP zprávy. Ve prvním sloupce této tabulky je sekvenční číslo paketu v původním pcap souboru, jinými slovy, pokud se podíváme ve Wiresharku na pcap soubor tento paket tam bude mít toto pořadí. Další sloupec dává přehled SIP paketů, které nesou v sobě SDP zprávy. Poslední sloupec nese jednotlivé SDP zprávy.

Po tabulce program vygeneruje filtry, pomocí kterých lze ve programu Wireshark vyfiltrovat příslušné RTP potoky.

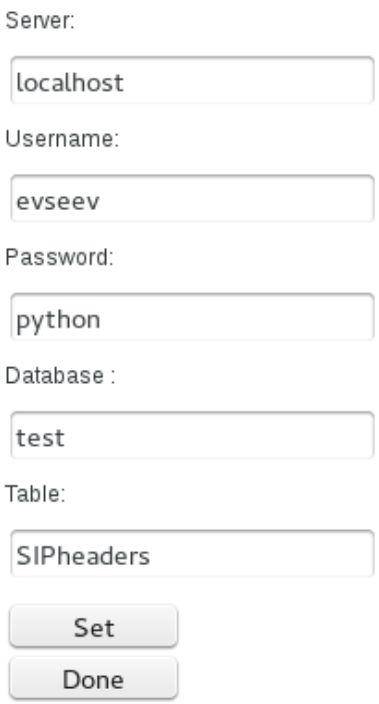
Pokud uživatel analyzuje podle jednotlivého Call-ID, na stránce výsledků takže bude nakreslen diagram stanovení spojení (obr. 6.4).

Pro potřeby pokročilejších uživatelů je systém propojen s aplikací phpMyAdmin. Pro přechod do této aplikace je zaveden odkaz "Search for advansed users"na domovské stránce (viz obrázek 6.2). Po kliknutí na tento odkaz uživatel je přesměrován do aplikace phpMyAdmin. Má možnost pracovat s originální databází, jejíž struktura je uvedena na obrázku 5.3. Tuto možnost lze využít zejména pro vytváření vlastních SQL dotazů v případě, že poskytnuté funkce jsou pro uživatele nedostačující.



Obrázek 6.4: Příklad SIP grafu

Výsledky zpracování můžou být uloženy do různých databází, a proto ve webovém rozhraní realizována možnost jejich připojení. Pro změnu nastavení připojení je dostupen odkaz Switch DB (viz obrázek 6.2), po kliknutí na který se zobrazí parametry pro přístup (viz obrázek 6.5). Tyto parametry určují databázi, k niž bude uživatelské rozhraní připojeno. V políčkách jsou vždy uvedeny současné hodnoty. Po upravení hodnoty je třeba potvrdit změny tlačítkem Set a poté se pomocí tlačítka Done vrátit na domovskou stránku.



Obrázek 6.5: Nastavení parametrů pro přípojení k databázi

Kapitola 7

Zhodnocení dosažených výsledků

Pro zjištění schopností navrženého řešení bylo spuštěno zpracovaní několika velkých pcap souboru. Soubory byly vygenerovány pomocí testovacího prostředku SIPp.

7.1 SIPp

SIPp je open source generátor provozu, který je navržen převážně pro testování SIP aplikace. SIPp umožňuje simulovat chování jako UAC i UAS a generovat signalizaci a přenos multimediálních dat. Program se používá pro generování provozu a simulování zátěžových modelových situací pro ověření zabezpečení ústředny a jejího provozu v různých situacích. Původní zdrojový kód programu SIPp byl napsán Richardem Gayraudem a upraven Olivierem Jacquesem. V současné době je program více zdokonalován a upravován komunitou developerů. Program si získal popularitu díky jednoduchému použití. Ovládání programu je skrze příkazový řádek, přes který se dají ovládat všechny funkce. Postupem vývoje byla implementována funkce importování XML souborů, pomocí kterých se dají vytvořit složitější komunikační schémata, a CSV souborů obsahujících rozšiřující data použitelných při psaní skriptů. Kód programu je otevřen pro opravení a případně vylepšení. Více informací o programu jsou dostupné na stránkách programu [23].

7.2 Způsob testování a dosažené výsledky

Pro účely testování realizovaný systém byl instalován na dvou různých počítačích. Apache a MySQL servisy jsou provozovány vždy v rámci jednoho serveru. Konfigurace serverů je následující:

1. Server 1 - Laptop, 4 jádra, 2.40 GHz, 6 GB RAM, SSD disk
2. Server 2 - Virtuální počítač, 2 jádra, 3 GHz, 1.5 GB RAM, HDD 7200 otáček

Testování probíhalo se třemi pcap soubory o různých velikostech. Cílem této volby bylo zjistit chování s ohledem na velikost souboru. Soubory byly

vytvořeny podle stejného scénáře a liší se počtem zachycených hovorů (viz tabulka 7.1). Každý hovor trvá 8 vteřin a používá se kodek G.711. Tento způsob zvolení testovacích souborů dovoluje dostat porovnatelné výsledky. Pro každý soubor se provádělo 5 měření.

Název souboru	**Velikost**	**Počet hovorů**	**Počet paketů**
10000.pcap	753 MB	10000	2520112
15000.pcap	1095 MB	15000	3780058
20000.pcap	1493 MB	20000	5040156

Tabulka 7.1: Testovací soubory

Výsledky testů je vidět v tabulce 7.2. V tabulce 7.3 je uvedena velikost databáze po zpracování souboru.

Pokus	**Doba zpracování, server 1**	**Doba zpracování, server 2**
Soubor 10000.pcap		
1	776.0 s	1564.8 s
2	777.5 s	1561.9 s
3	778.1 s	1563.6 s
4	773.2 s	1560.7 s
5	774.9 s	1562.5 s
Soubor 15000.pcap		
1	1173.0 s	2337.2 s
2	1174.1 s	2345.8 s
3	1167.8 s	2339.3 s
4	1166.9 s	2340.1 s
5	1172.5 s	2344.6 s
Soubor 20000.pcap		
1	1536.9 s	3112.1 s
2	1537.2 s	3114.6 s
3	1538.1 s	3109.5 s
4	1540.4 s	3107.8 s
5	1543.5 s	3113.0 s

Tabulka 7.2: Výsledky testování zpracování souboru

Název souboru	**Velikost databáze**	**Počet záznamů**
10000.pcap	52.3 MB	119720
15000.pcap	79.6 MB	179580
20000.pcap	107.9 MB	239440

Tabulka 7.3: Obsah databáze po zpracování pcap souboru

Bylo provedeno hodnocení časové náročností při využití paralelních výpočtů (výpis 5.7). Testy se prováděly za vyžitím 4 jader serveru 1, výsledky jsou v tabulce 7.4.

Název souboru	Doba zpracování, server 1
10000.pcap	252.7 s
15000.pcap	382.7 s
20000.pcap	501.3 s

Tabulka 7.4: Paralelní výpočty

Pomocí Linux příkazu `time` lze spočítat metriku Resident set size (RSS), která zachycuje paměť skutečně využitou procesem. Jak je vidět z tabulky 7.5 tato hodnota ve všech testech nepřesahovala 32 MB. Dále pomocí nastroje `top`, který slouží pro monitorování právě běžících procesů, jsou zaznamenány okamžité hodnoty RSS po otevření testovacích souborů programem Wireshark. Lze všimnout, že Wireshark nahrává do operační paměti počítače celý soubor a poté s ním pracuje, což může způsobit problémy s nedostatkem paměti na některých počítačích.

Název souboru	Navržený systém		Aplikace Wireshark, RSS
	Max RSS, server 1	Max RSS, server 2	
10000.pcap	30.2 MB	20.2 MB	943 MB
15000.pcap	30.2 MB	20.5 MB	1403 MB
20000.pcap	30.1 MB	20.6 MB	1895 MB

Tabulka 7.5: Využití paměti

Pokus	Doba zobrazení domovské stránky	Doba filtrace podle Call-ID	Doba analýzy
Server 1			
1	5.6 s	5.3 s	0.11 s
2	5.8 s	6.3 s	0.09 s
3	5.1 s	5.6 s	0.09 s
4	5.9 s	5.9 s	0.09 s
5	6.1 s	5.2 s	0.08 s
Server 2			
1	7.0 s	6.2 s	0.11 s
2	6.4 s	6.6 s	0.12 s
2	6.9 s	6.3 s	0.11 s
2	6.0 s	6.1 s	0.11 s
2	6.1 s	5.4 s	0.11 s

Tabulka 7.6: Výsledky testování webového rozhraní

Dalším krokem je testování odezvy uživatelského rozhraní po zpracování pcap souboru. Časová náročnost zpracování SQL dotazů záleží na velikosti databáze. Z kapitoly 5 je vidět, že webové rozhraní navržené aplikace využívá několik SQL dotazů, a proto je třeba otestovat, jestli se ji dá používat s plnou databází. Databáze v průběhu testů obsahovala pakety 15000 hovorů

(179580 záznamů). Testy byly provedeny zase pro systémy instalované na dvou různých serverech. Pro testování byl použit plugin Page Load Time pro prohlížeč Google Chrome, který zobrazuje jak rychle se nahraje webová stránka.

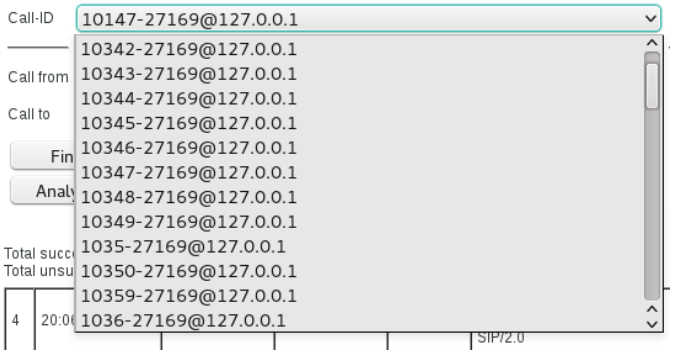
Z naměřených hodnot je vidět, že doba zpracování pcap souboru je lineární závislá na jeho velikosti, a pro vetší soubory může dosahovat 25 minut. Na druhou stranu, soubor 15000.pcap, obsahující 15000 hovorů délkou 8 vteřin každý, byl vygenerován během 50 minut (5 spojení za vteřinu). To zmámená, že lze uskutečnit zpracování v dobu zachycení pcap souborů. Dále je třeba všimnout, že po jednom dlouhém zpracování nasleduje velice rychle využití uživatelského prostředí. Filtrace výsledků sice zobrazuje s malým zdržením, avšak analýza paketů a kreslení grafů probíhají pro uživatele zcela nevnímatelně. Nakonec tabulka 7.3 ukazuje, že výsledná velikost databáze je skoro 10 krát menší než původní pcap soubor.

Analýza navrženého řešení a jeho testování ukazuje, že nejnáročnější operací je samotné parsování SIP protokolu. Určité zlepšení přináší zavedení vícejádrové podpory popsané v kapitole 5.6.

7.3 Možná vylepšení systému

Z pohledu návaznosti na tuto práci by bylo užitečné zavedení funkce zpracovaní v reálném čase. Zachycená data by se přímo převáděla na vstupní data ke zpracování a byla by ukládána do databáze. Až by se dosáhlo některého omezení, tak by se zachycená data ukládala do souboru nebo pamětí následně by se zpracovávala.

Další zlepšení se týká webového rozhraní. Při zobrazení domovské stránky nejvíc času zabírá zpracování dotazu, který naplňuje položky výběru Call-ID, From a To (viz obrázek 7.1)). Problém je v tom, že dotaz se vyplňuje při každém obnovení stránky a filtraci. Proto lze zamyslit na zavedení možnosti obnovení jenom tabulky výsledků.



Obrázek 7.1: Položka výběru Call-ID

Z hlediska vývoje analýzy je možné přidání dalších funkcí, například zavedení do algoritmu analýzy hovorů hledaní nějakých určitých chybných hlášek.

Kapitola 8

Závěr

Cílem této diplomové práce byl návrh a realizace systému pro pasivní analýzu zachycené SIP signalizace se zaměřením na možnost analýzy většího objemu dat. K tomu jsem využil vlastnost SIP protokolu, kterou je kódování ve znakové sadě UTF-8, a proto je snadno čitelný.

Na počátku bylo nejprve nutné prozkoumat formát SIP zpráv, tzn. jaká data budou na vstupu zpracovávána. Dále bylo nutné rozebrat existující řešení a knihovny, případně i moduly pro vývoj takových aplikací. Na základě těchto dat bylo rozhodnuto, že se pro zpracování zachycené síťové komunikaci implementuje vlastní parser v jazyce Python. S ohledem na zpracovaní většího počtu dat byla pro uložení výsledků zvolena MySQL databáze.

V dalším kroku bylo navrženo a implementováno webové rozhraní, které slouží k analýze výsledků. Pomocí něho uživatel může najít hovor podle známého Call-ID, URI volajícího nebo volaného. Při implementaci analýzy byly využity SQl dotazy, které jsou vykonávány PHP skripty. Navíc bylo provedeno doplnění analýzy informací o SDP pakety a vytvoření filtrů pro hledaní příslušných RTP potoků. Kromě toho byl systém propojen s nástrojem phpMyAdmin, což umožňuje pokročilejším uživatelům vytvářet vlastní SQL dotazy nad zpracovanými daty. Taky byly vyzkoušeny metody distribuovaného zpracování.

Po implementaci výsledného řešení, byly provedeny testy s různými velikostmi zpracovávaných vstupních souborů. Tyto testy ukázaly možnost použití systému pro větší objemy dat. Velice důležitým jsou poznatky z testů, které mohou sloužit k dalšímu vylepšení a zrychlení navrženého systému.

Hlavním přínosem této práce je napsaní modulu, který zavadí podporu SIP protokolu do nastroje Scapy. Protože tento nástroj slouží k manipulaci a generaci paketů, po začlenění podpory SIP protokolu se otevírají další možnosti využití. Na této bázi se lze například pokusit o realizaci User Agent, nebo o testování útoků na SIP ústředny.

Celkově si z této práce odnáším mnoho zkušeností. Zejména hlubší pochopení protokolu SIP a práce s SQL databází.

Příloha A

Literatura

[1] RFC 3261. SIP: Session Initiation Protocol.

Dostupné na https://www.ietf.org/rfc/rfc3261.txt [on-line] (cit. 24.5.2016)

[2] Wireshark - About.

Dostupné na https://www.wireshark.org/about.html [on-line] (cit. 24.5.2016)

[3] RFC 4566. SDP: Session Description Protocol.

Dostupné na https://www.ietf.org/rfc/rfc4566.txt [on-line] (cit. 24.5.2016)

[4] Johnston, A.B: SIP: Understanding the Session Initiation Protocol.

Artech House; 3rd edition, 2009. 395 pages. ISBN: 978-1-607-83995-8

[5] Ahson, S.A., Ilyas, M.: SIP handbook: services, technologies, and security of Session Initiation Protocol.

CRC Press; 2009. 599 pages. ISBN: 978-1-4200-6603-6

[6] Sisalem, D.: SIP Security.

John Wiley & Sons; 2009. 350 pages. ISBN: 978-0-470-51636-2

[7] Zabezpečení multimediálního přenosu v reálném čase

Dostupné na http://realtimesecure.asp2.cz/Default.aspx [on-line] (cit. 24.5.2016)

[8] RFC 3550. RTP: A Transport Protocol for Real-Time Applications.

Dostupné na https://www.ietf.org/rfc/rfc3550.txt [on-line] (cit. 24.5.2016)

[9] Tcpdump/libpcap public repository.

Dostupné na http://www.tcpdump.org/ [on-line] (cit. 24.5.2016)

[10] A tcpdump with examples.

Dostupné na https://danielmiessler.com/study/tcpdump/ [on-line] (cit. 24.5.2016)

[11] tcpflow - Linux man page.

Dostupné na http://linux.die.net/man/1/tcpflow [on-line] (cit. 24.5.2016)

[12] libosip Documentation.

Dostupné na http://www.gnu.org/software/osip/doc/html/ [on-line] (cit. 24.5.2016)

[13] Scapy Documentation.

Dostupné na http://www.secdev.org/projects/scapy/doc/ [on-line] (cit. 24.5.2016)

[14] Programovací jazyk Python.

Dostupné na https://www.python.org/ [on-line] (cit. 24.5.2016)

[15] Downey, A.: How to Think Like a Computer Scientist - Learning with Python.

Wellesley, MA: Green TeaPress; 1st edition, 2008. 262 pages. ISBN: 860-1-234-62098-3

[16] MySQLdb User's Guide.

Dostupné na http://mysql-python.sourceforge.net/MySQLdb.html [on-line] (cit. 24.5.2016)

[17] MySQL Development History.

Dostupné na http://dev.mysql.com/doc/refman/5.1/en/development-history.html [on-line] (cit. 24.5.2016)

[18] Naramore, E.: Vytváříme webové aplikace v PHP5, MySQL a Apache.

Brno: Computer Press; 1st edition, 2006. 813 pages. ISBN 80-251-1073-7

[19] The Apache HTTP server project.

Dostupné na https://httpd.apache.org/ [on-line] (cit. 24.5.2016)

[20] phpMyAdmin - Documentation.

Dostupné na https://www.phpmyadmin.net/ [on-line] (cit. 24.5.2016)

[21] O. Tange: GNU Parallel - The Command-Line Power Tool.

The USENIX Magazine, February 2011:42-47.

[22] Správa linuxového serveru: Instalace LAMP.

Dostupné na http://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-instalace-lamp [on-line] (cit. 24.5.2016)

[23] Welcome to SIPp.

Dostupné na http://sipp.sourceforge.net/ [on-line] (cit. 24.5.2016)