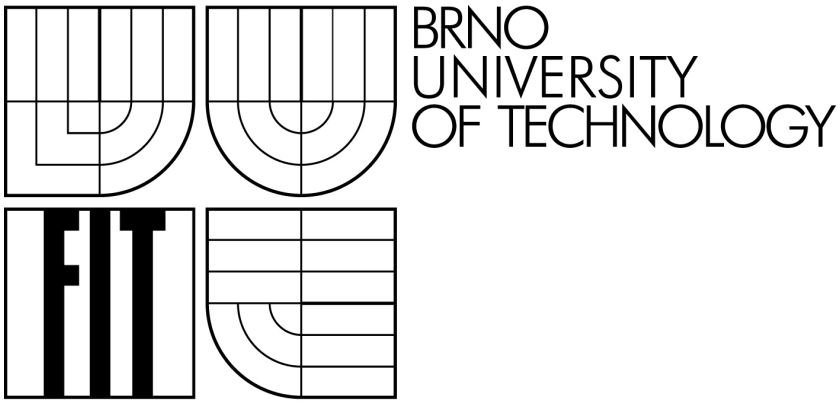

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Analyzátor VoIP signalizace

Autor: Tomáš Slunský
Editor: Patrície Mahdalová
Login: xsluns01
Datum: 29.11. 2014

Obsah

Úvod

Cílem projektu bylo naimplementovat program, který bude analyzovat VOIP signalizaci. Program bude fungovat obdobným způsobem jako sipscanner s tím rozdílem, že výstupy budou zapsány v XML.

Cílem analýzy bude konkrétně protokol SIP a jeho součástí, jako protokol SDP pro identifikaci sezení, nebo RTP protokol pro samotný přenos dat. Úkolem programu tedy bude analyzovat jednotlivé zprávy mezi klientem a serverem. Zprávy SIP protokolu se identifikují velkými písmeny, jelikož samotný SIP protokol vychází z HTTP protokolu a také díky tomu je celé zpracování programu značně jednodušší. Dále je třeba zmínit, že nejčastějším protokolem nad kterým probíhá signalizace SIP je protokol UDP, ovšem komunikace může probíhat i nad TCP nebo SCTP, což je potřeba v implementaci programu zohlednit.

Klient, který iniciuje spojení se ve většině případů registruje přes registrační server. Na každou žádost mu server odpovídá stovkovými hláškami, které jsou rozděleny do 6 skupin. Na základě odpovědí se dále vyhodnocuje další signalizace a postupy zpracování signalizace.

Návrh programu

Program je rozdělen na několik základních funkcí, které vykonávají operace nad protokolem SIP, tyto funkce dále používají další menší funkce, užité pro rychlejší zpracování cele problematiky. Tímto způsobem se dosáhne vyšší přehlednosti a efektivity při zpracování.

Popis některých funkcí:

- **pktParser()**:
  - slouží pro načtení základních informací z vrstvy Raw, TCP/UDP/SCTP a IP vrstvy paketu

- **parseAMCOfSDP()**:
  - stará se o načtení konkrétních dat z protokolu SDP

- **pktSdpParser()**:
  - zpracování protokolu SDP

- **executePkts()**:
  - v této funkci je implementován celý algoritmus zpracování SIP komunikace

- **pktsToXML()**:
  - převod dat z datové struktury do XML

- **argsExecute():**
  - ověření platnosti zadaných parametrů

Popis implementace

Pro implementaci VOIP analyzátoru SIP jsem zvolil jazyk Python s knihovnou scapy pro zpracování síťové komunikace.

Filtrace paketů

Ihned po spuštění programu a načtení vstupních dat dochází k odfiltrování paketů, které se netýkají SIP protokolu, filtrace probíhá dle protokolu nad kterými SIP protokol může být realizován a dále dle portu, přičemž se upřednostňuje ten implicitní pro sip (v případě neuvedení v argumentu)

Vyhodnocení komunikace

Komunikace je vyhodnocována algoritmem ve funkci **executePkts()** uvedené již výše. Na vstup algoritmu vždy přicházejí již vyfiltrované pakety pomoci funkce **filter()** a následně algoritmus očekává, že dojde k registraci klienta, v případě že k registraci nedojde, pokračuje ve výkonu dál. Zjištění konkretních dat z paketů, jsou zajištěny pomoci vrstvy paketu **Raw (load)**, kde se dané data o SIP nacházejí. Jelikož všechny zprávy SIP jsou uvedeny hned na začátku bloku **load** tak za pomocí funkce **pktSearch()**, vracející SIP odpověď na požadavek/žádost klienta může docházet k odchytu adekvátních paketů.

V případě že byl odchycen paket s žádostí register, algoritmus „skočí“ do bloku, zpracovávající tuto žádost, a je ukončen až jakmile přijde zpráva 4xx/5xx/6xx – tedy když se jedná o chybu. Pokud ale přijde odpověď na žádost 1xx - načítá další paket a v případě, že přijde 2xx začne s parsováním dat o registraci a přidáním je do datové struktury.

Jádrem zpracování je blok INVITE, kde dochází k vyhodnocení celé komunikace. Jakmile tento typ paketu zachycen, algoritmus skočí do bloku, kde se vše zpracovává a po odeslání žádostí začne načítat další pakety, kde se snaží odchytit odpověď a podle odpovědí řídí další zpracování. V případě, že přijde odpověď na register:

- **1xx/3xx:**

  - dojde k načtení dalšího paketu

- **2xx**:

  - vykonávání bloku končí, parsují se data ze SIP/SDP a ukládají do datové struktury

- **4xx/5xx/6xx:**

  - vykonávání se přeruší – vyskočí se z bloku zpracování a očekává se další paket

Převod do XML

Jakmile funkce **executePkts()** dokončí svoji práci a uloží data do datové struktury, předají se do funkce pro zpracování XML. Ta probíhá v cyklu, kdy načítá pouze data ze struktury a podle vyžadovaného výstupu je formátuje do proměnné, jakmile je cyklus dokončen, data se přepíši do výstupního souboru. Zajímavostí je, že až ve výpisu dochází k ověřování, kolik media description bloků daný hovor obsahuje, nebo kolik kodeků každé media mají odsouhlasená oběma stranami.

Problémy při implementaci

- Při zpracování časů z časových značek paketu SIP, kdy v případě, že dochází například ukončení pomocí CANCEL mi nebylo úplně jasné, jakou značku použít

- odchyt dat z rozhraní, do poslední chvíle mi moc nešlo zprovoznit odposlech pomocí funkce sniff

- některé věci v RFC a v testech mi neseděli, tudíž jsem řešil nejednoznačnost při načítaní například identifikace z paketu SIP klienta

Navod k použití

Použití: **./sipscan {-f|-i} name -o file [-p number]**

-f [--file] name – vstupní soubor s daty pro analýzu
-i [--interface] name – odchyt dat pro analýzu z rozhraní
-o [--output] file – zápis výsledných XML do výstupního souboru
-p [--port] num -- číslo portu, kde probíhala SIP

Závěr

Projekt byl pro mě další výzvou, naučil jsem se dost věcí ohledně VOIP a telefonie obecně. Dále například práci s pakety na úrovni jednotlivých síťových vrstev a prakticky si vše naživo vyzkoušel. Projekty tedy hodnotím jak přínos a implementačně jsem se snažil splnit všechny požadavky, což se mi také povedlo. Projekt byl náležitě otestován a ve všech testech prošel, tak jak jsem předpokládal.

Literatura

[1] Session Initiation Protocol (SIP) Basic Call Flow Examples, December 2003, RFC3665

[2] SDP: Session Description Protocol, July 2006, RFC4566

[3] RTP: A Transport Protocol for Real-Time Applications, July 2003, RFC3550