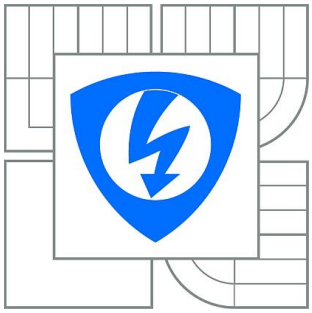
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

TESTOVÁNÍ ODOLNOSTI IP PBX PROTI ÚTOKŮM S VYUŽITÍM TESTERU SPIRENT AVALANCHE

DIPLOMOVÁ PRÁCA

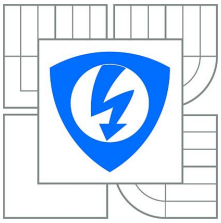
AUTOR PRÁCE — MARTIN ZELENAY

AUTHOR

VEDÚCI PRÁCE — Ing. PAVEL ŠILHAVÝ, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav telekomunikací

Diplomová práca

magisterský študijný obor

Telekomunikační a informační technika

Študent: Bc.Martin Zelenay ***ID*** 136599

Ročník: 2. ***Akademický rok:*** 2014/2015

NÁZOV TÉMY:

Testování odolnosti IP PBX proti útokům s využitím testeru Spirent Avalanche

POKYNY PRE VYPRACOVANIE:

Prostudujte metody zajištění IP PBX proti útokům. Zaměřte se na open source PBX Asterisk, FreeSwitch, YATE, Kamailio a OpenSIPs v posledních LTS verzích. Věnujte se protokolům SIP a IAX2. Útoky generujte s využitím testovacího systému Spirent TestCenter. Navrhněte a v testech ověřte opatření proti jednotlivým typům útoků, které tester zahrnuje. Výsledky testů jednotně vyhodnoťte dle Vámi definovaných kritérií. Ověřte a popište možnosti tvorby vlastních útoků s využitím testeru.

DOPORUČENÁ LITERATÚRA:

[1] Meggelen, J.V, Smith, J., Madsen, L. Asterisk™ The Future of Telephony. Sevastopol: O'Reilly Media, Inc., 2005. ISBN 0-596-00962-3.
[2] Jackson, Benjamin. Asterisk hacking. Burlington: Syngress, 2007. ISBN 978-1-59749-151-8.
[3] Dwivedi, Himanshu. Hacking VoIP :protocols, attacks, and countermeasures / San Francisco : No Starch Press, 2009. ISBN 978-1-59327-163-3.
[4] Endler, David. Hacking exposed VoIP :voice over IP security secrets & solutions / New York : McGraw-Hill, 2007. ISBN 978-0-07-226364-0.

Termín zadania: 23.9.2014 ***Termín odovzdania:*** 29.5.2015

Vedúci práce: Ing. Pavel Šilhavý, Ph.D.

prof. Ing. Kamil Vrba, CSc.

Predseda oborové rady

UPOZORNENIE:

ABSTRAKT

Práca skúma, analyzuje a vyhodnocuje vplyv útokov vo VoIP sieťach na funkčnosť open source ústrední. Rozoberá problematiku útokov cez paketové siete v rámci bezpečnostných noriem a zaužívaných pravidiel. V teoretickej časti sú vysvetlené pojmy a problematika VoIP sietí so zameraním sa na fakty potrebné k plnému pochopeniu skúmaných dejov a meraní. V praktickej časti je zhrnutá realizácia konkrétnych testov útokov podľa zadania práce s prvotnou orientáciou na overenie zariadenia, ústrední a možností realizácie útokov a následné komplexná tvorba a testovanie útokov typu fuzzing a záplavových DoS.

KĽÚČOVÉ SLOVÁ

VoIP, útoky, vplyv, open source, ústredne, bezpečnosť, funknčnosť, stabilita, Asterisk, Freeswitch, Yate, Kamailio, Opensips

ABSTRACT

This work explores, analyzes and rate infuence of VoIP attacks on open source pbx functionality. It describes how voice over IP attacks are achieved according to security standards. There are described concepts and basics of VoIP networks with orientation on facts necessary to understand analyzed actions and measurements in theoretical part. In practical part, there is described realization of attack tests according to instructions with first orientation on initial checking of testing device, pbx's and security attack possibilities and then complex creation and testing of attack scenarios types such as fuzzing and denial of service attacks.

KEYWORDS

VoIP, attacks, influence, open source, pbx, security, functionality, stability, Asterisk, Freeswitch, Yate, Kamailio, Opensips

ZELENAY, M. *Testování odolnosti IP PBX proti útokům s využitím testeru Spirent Avalanche.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 53 s. Vedoucí diplomové práce Ing. Pavel Šilhavý, Ph.D.

PREHLÁSENIE

Prehlasujem, že svoju prácu na tému „*Testování odolnosti IP PBX proti útokům s využitím testeru Spirent Avalanche*“ som vypracoval samostatne pod vedením vedúceho práce a s použitím odborné literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušením ustanovení § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení časti druhej, hlavy VI. diel 4 Trestného zákonníku č. 40/2009 Sb.

V Brne dňa s . . . .

(podpis autora)

POĎAKOVANIE

Rád by som poďakoval vedúcemu práce pánovi Ing. Pavlovi Šilhavému, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci. Zvláštna vďaka patrí mojím najbližším za morálnu a materiálnu podporu počas štúdia.

V Brne dňa . . . . . .

. . . . . .

(podpis autora)

Faculty of Electrical Engineering
and Communication

Brno University of Technology

Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výskum popísaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených z projektu SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

OBSAH

Úvod

Primárnym cieľom tejto práce je zanalyzovať vplyv tzv. fuzzing útokov, zahrnutých v licencií testeru Spirent Avalanche, a záplavových útokov odmietnutia služby na VoIP open source ústredne Asterisk, FreeSwitch, YATE, Kamailio a OpenSIPs. K tomu je potrebné pochopiť problematiku prenosu hlasu cez dátové siete (VoIP) a problematiku v nej používaných signalizačných, transportných a bezpečnostných komunikačných protokolov. Tie sú z hľadiska teoretického rozoberané v prvej časti práce. Ich principiálna znalosť je potrebná k lepšiemu pochopeniu a sledovaniu skúmaných javov.

Ďalej je v práci popísaný charakter, história vzniku a architektúra jednotlivých softwarových a voľne šíriteľných ústrední. Každá z ústrední má iný špecifickú štruktúru a vývojovú komunitu, čo sa odrazí aj na ich neskoršom charaktere a odlišnostiach vo výsledkoch počas testovania.

Nasleduje kapitola venovaná problematike útokov, ktorá z principiálneho hľadiska triedi útoky podľa viacerých kritérii a stručne, vecne a prehľadne útoky popisuje. To je rovnako dôležité k vytvoreniu uceleného pohľadu na rozsah možností zabezpečenia ústrední. Metódam zaistenia ochrany je venovaná samostatná časť.

V praktickej časti práce je detailne rozobornaná inštalácia a konfigurácia jednotlivých ústrední s následným overením ich funkcionality jednak so softwarovými klientmi a jednak so samotným testerom-generátorom Spirent. V ďalšej, hlavnej časti práce je zachytená a rozanalyzovaná realizácia jednotlivých útokov rozdelená do viacerých kapitol podľa typu útoku.

Primárnou bola realizácia útokov s modifikovanými správami (fuzzing), z ktorých časť je zahrnutá v licencii testeru a časť bola vytvorená, aby tak bola demonštrovaná schopnosť Spirentu tvoriť aj vlastné útoky. Výsledky testov sú rozanalyzované a vyhodnotené v samostatnej kapitole, kde je navrhnutých aj overených viacero foriem zabezpečenia. Testovanie v tejto fázi tvorí vyše 150 testov a je podložené výstupmi zo sieťového analyzátoru a doplnené sprievodnými komentármi so zameraním na vysvetlenie charakteru chovania sa jednotlivých ústrední počas útoku.

V sekundárnej časti nasleduje realizácia záplavových útokov odmietnutia služby, ktorých cieľom je preveriť ústredne z výkonnostného hľadiska. Výsledky sa opierajú o grafické porovnanie a samotná analýza a návrh zabezpečenia je obsahom vlastnej kapitoly. Testovanie tohto typu útoku zahrňuje ďalších 50. Posledná kapitola práce sa venuje okrajovo ďalším typom útokov a stručne popisuje možnosti ich tvorby a zabezpečenia voči nim.

1 PBX ústredne

Pobočková telefónna ústredňa (PBX) je zariadenie, ktoré zjednocuje výstupné body všetkých telefónnych koncových zariadení firemnej siete. Medzi jej najväčšie prednosti patrí spoľahlivosť, jednoduchá implementácia do analógových sietí a centralizovaná štruktúra celej jej organizácie. Najrozšírenejšia je v dnešnej dobe realizácia softwarových ústrední, často pomocou tzv. *open source* (voľne modifikovateľných), nakoľko voľnosť modifikácií a prístupu nie sú limitované a sú zadarmo. Existuje pomerne veľké množstvo open source PBX, medzi najznámejšie patria *Asterisk, FreeSWITCH, YATE, Kamailio* a *OpenSIPs,* ktoré boli použité aj v tejto práci.

1.1 Asterisk

Open source ústredňu Asterisk vytvoril Mark Spencer v roku 1999, dôvodom jeho vzniku bola potreba nekomerčného PBX systému. V roku 2004 sa jednalo o prvý plnohodnotný open source PBX systém. V dnešnej dobe existuje Asterisk vo verzii 13, predchádzajúca verzia systému bola 11. Tá je v LTS (Long Term Support), takže sa jedná o verziu kde je zaručená dlhodobá podpora [6]. V tabuľke 1.1 sú znázornené jednotlivé vývojové rady Asterisku.

Tab. 1.1: Vývojové rady ústredne Asterisk [7]

Verzia	Typ	Dátum vydania	Bezpečnostné záplaty do:	Koniec podpory
1.2.X	-	21/11/05	8/7/07	21/11/10
1.4.X	LTS	23/12/06	21/4/11	21/4/12
1.6.0.X	Standard	1/10/08	1/5/10	1/10/10
1.6.1.X	Standard	27/4/09	1/5/10	27/4/11
1.6.2.X	Standard	18/12/09	21/4/11	21/4/12
1.8.X	LTS	21/10/10	21/10/14	21/10/15
10.X	Standard	15/12/11	15/12/12	15/12/13
11.X	LTS	31/10/12	25/12/16	25/10/17
12.X	Standard	20/12/13	20/12/14	20/12/15
13.X	LTS	10/2014	10/2018	10/2019

Jedná sa o *najrozšírenejší* open source systém pre VoIP technológie. Je dostupný na všetkých operačných systémoch od Linux, BSD, Windows, po MacOS X. Poskytuje všetky funkcie, ktoré sú očakávané od pobočkových ústrední. Asterisk je softwarová pobočková ústredňa umožňujúca ako IP telefóniu, tak digitálnu ISDN aj analógovú telefóniu PSTN. Medzi funkcie, ktoré podporuje, patria napríklad IVR (Interactive Voice Response) a ACD (Automatic Call Distribution) [7].

Najznámejšie podporované signalizačné protokoly:

- IAX - Inter-Asterisk Exchange – jedná sa o protokol vytvorený tvorcami Asterisku
- H.323
- SIP - Session Initiation Protocol
- MGCP - Media Gateway Control Protocol
- SCCP - Cisco Skinny.

A najznámejšie podporované kodeky:

• ITU-T G.711 (A-Law a μ-Law), G.726, G.723.1, G.729, GSM a Speex. [8]

1.1.1 Architektúra pobočkovej ústredne Asterisk

Okolo centrálneho jadra PBX sú definované špecifické API (Application Programming Interface). Jadro ovláda vnútorné prepojenie PBX, špecifické protokoly, kodeky a hardwarové rozhrania telefónnych aplikácií.

• **PBX prepojovacie jadro** – základným prvkom systému Asterisk je prepojovací systém pobočkovej ústredne, slúžiaci k spojovaniu rôznorodých užívateľov a automatizovaných úkonov.
• **Spúšťač aplikácií** - slúži k spúšťaniu aplikácii ako je hlasová pošta, prehrávanie súborov a prechádzanie adresárov.
• **Prekladač kodekov** - používa moduly s kodekmi pre kódovanie a dekódovanie rôznych audio formátov, ktoré sa používajú v telefónnom prostredí. Má k dispozícii dostatočne veľkú škálu kodekov, aby vyhovoval rozdielnym potrebám a dosahoval čo najlepších výsledkov v pomere medzi audio kvalitou a využitou šírkou pásma.
• **Plánovač a I/O Manažér** - zaisťuje plánovanie úloh na najnižšej úrovni a zaisťuje optimálne využitie výkonu systému za akýchkoľvek podmienok. [8]

1.2 Freeswitch

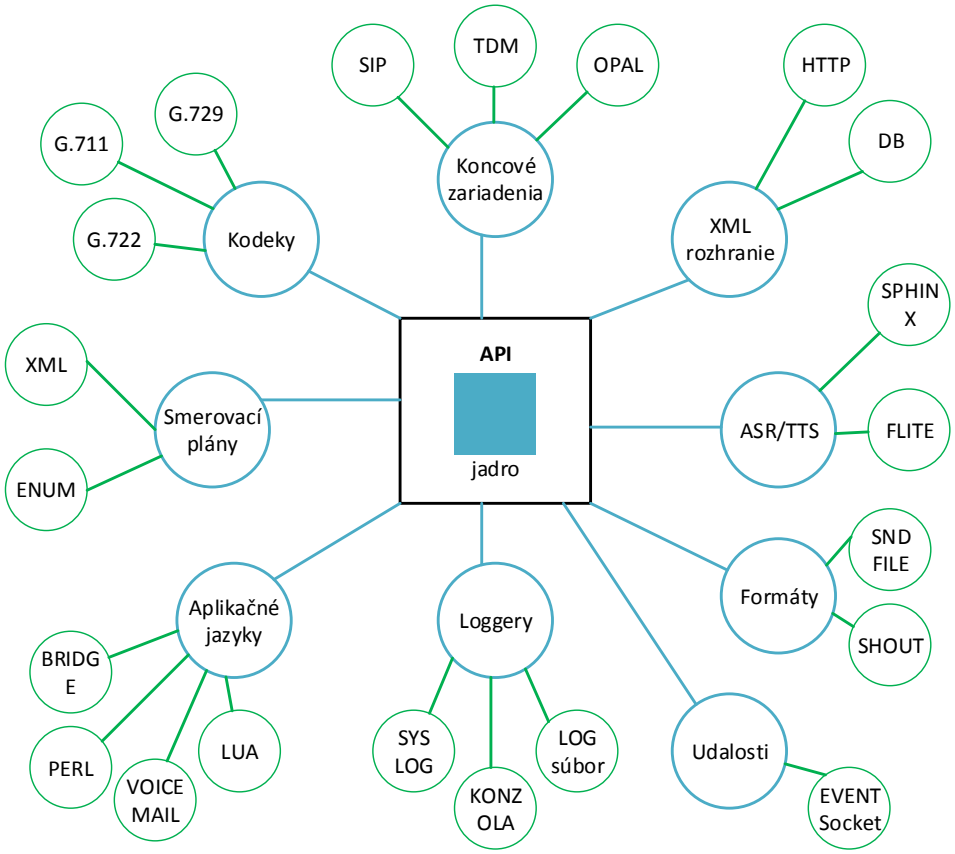
Vznikol v roku 2006 ako konkurenčný open source nástroj ku komerčným riešeniam softwarových ústrední. Od začiatku bol vývojármi Asterisku tvorený ako multiplatformný s podporou rôznych komunikačných protokolov a s možnosťou prenosu popri hlasových, aj video a textových dát. Takisto jeho výhodou sú rozsiahle možnosti modifikovatelnosti a modularity s udržaním si dostatočnej stability. Aby sa vyhlo prehnanej komplexnosti používa známe voľne dostupné softwarové knižnice.

Medzi najznámejšie podporované protokoly patria SIP, SCCP, GoogleTalk. S podporou dodatočných modulov spolahlivo rozchodí aj H.323 alebo XMPP (Extensible Messaging and Presence Protocol). V súčasnej dobe však už nepodporuje IAX2. Je kompatibilný s inými známymi riešeniami open source ústrední, ktoré sú použité aj v tejto práci.

Podporuje zabezpečnie pomocou TLS a SRTP, ktorých princípy sú vysvetlené v kapitole 2.5. Ďalej podporuje širšu škálu úzkopásmových aj širokopásmových hlasových kodekov, vrátane najznámejších G.711, G.726 aj komerčný G.729. [28]

1.2.1 Architektúra pobočkovej ústredne Freeswitch

Jedná sa o jadro – knižnicu so spúšťačom, ku ktorej sú dostupné moduly. API (z ang. application programming interface) podporuje rozšíriteľnosť cez známe jazyky ako Perl, Python alebo C. Implicitne pridružené moduly sú vidieť na obrázku 1.1 modrou, k nim dodatočné moduly sú zelené.



Obr. 1.1: Architektúra Freeswitchu [29]

1.3 Yate

Yet Another Telephony Engine voľne preložené ako „len ďalšia telefonická ústredňa“, pôvodne vyvíjaná rumunskou telekomunikačnou spoločnosťou NULL, je zameraná na hovory z VoIP a PSTN sietí. Je šírená pod GPL (General Public License) a ľahko rozšíriteľná a dokáže, podobne ako ostatné ústredne, prenášať aj video a textové správy. Využíva flexibilný smerovací mechanizmus, ktorý maximalizuje efektívnosť komunikácie a minimalizuje straty. Obsahuje aj klienta na uskutočňovanie hovorov.

Podporuje všetky známe VoIP protokoly počnúc SIP, cez H.323, IAX po MGCP. Medzi ďalšie funkcie, ktoré podporuje, patria napríklad IVR, platobné systémy, záznamník a iné.

Je napísaný v jazyku C++ a podporuje skriptovanie v rôznych iných jazykoch ako PHP, Perl, Python alebo aj Unixový shell.

1.3.1 Architektúra pobočkovej ústredne Yate

Najdôležitejším prvkom je mechanizmus predávania správ. Moduly si vymieňajú správy medzi sebou. Toto umožňuje väčšiu flexibilitu než s jednoduchými funkciami, najmä pretože správy v Yate môžu mať ľubovoľný počet parametrov a môžu byť poslané do viacerých modulov zmenou ich priority. Štyri hlavné komponenty sú:

- **Jadro** – generické triedy ako String, Thread, Socket, Mutex.
- **Mechanizmus správ (Message Engine) –** triedy súvisiace so správami Massage, Engine, Plugin.
- **Mechanizmus telefonovania (Telephony Engine) –** triedy súvisiace s telefonovaním ako Driver a Channel.
- **Moduly –** sú si rovné nezávisle na ich funkcii, vďaka mechanizmu predávania správ. Načítavajú sa spolu s smerovacím plánom po spustení ústredne. [10]

1.4 Kamailio

Pôvodne OpenSER, je voľne dostupný SIP server vydaný pod GPL, schopný obsluhovať tisícky hovorov za sekundu. Vznikol v roku 2001 ako projekt od firmy Fraunhofer, pod názvom SER. Časť vývojárov sa odtrhla a v roku 2004 vytvorila voľne dostupný OpenSER, ktorý s podporou tretích strán rástol a v 2008 bol premenovaný na Kamailio, pôvodne hawaiské slovo označujúce „konverzáciu“. Dnes v sebe zahŕňa aj SIP Express Router (SER), ktorý bol pôvodným projektom od roku 2001. Obidva majú rovnaký zdrojový kód, ale odlišné databázy na ukladanie užívateľských účtov a smerovacích informácii, čo závisí od použitých modulov (napr. na autentizáciu alebo lokalizáciu užívateľa). [25]

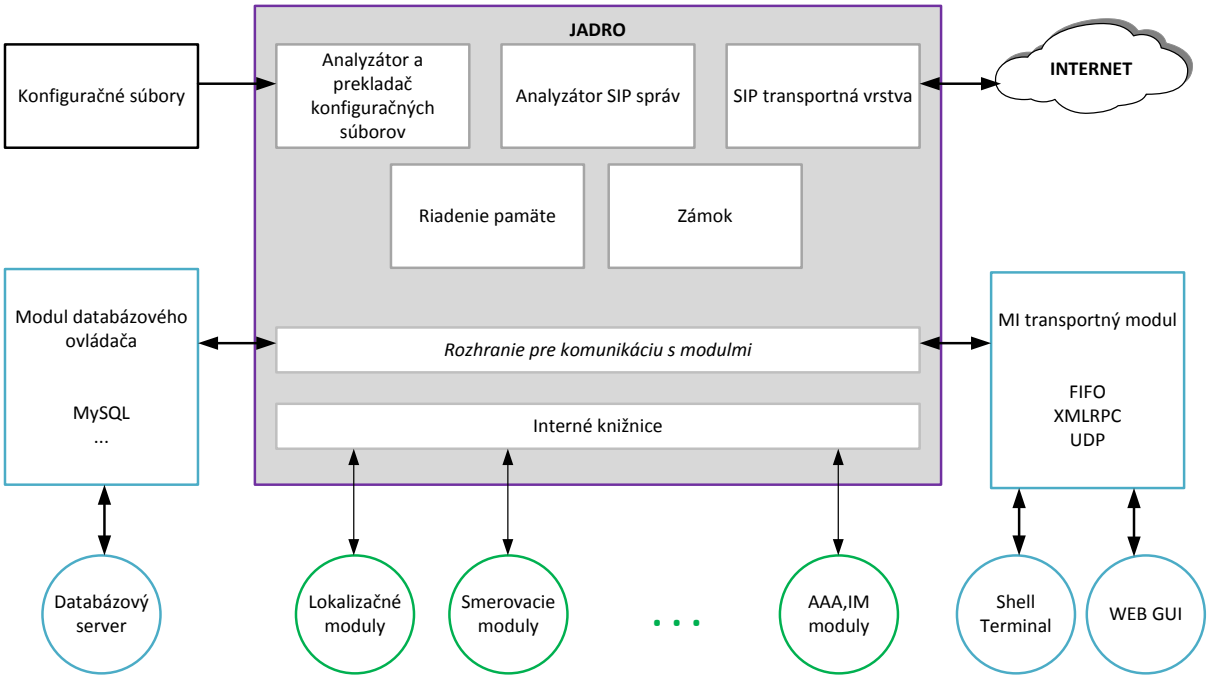
Môže byť použitý ako SIP proxy, registrar, lokalizačný, aplikačný, dispatcher (riadi viacero prichádzajúcich sieťových reláčných požiadaviek do spoločnej fronty) a websocket server (načúva ľubovoľnému portu na serveri, ktorý sleduje špecifický protokol). Vďaka implementácii SER aj ako globálny smerovač alebo ako SBC (Session Border Controller, ktorý sa chová ako VoIP brána). Nedokáže zastupovať back to back user agent, ktorý prijaté požiadavky preformuluje a preposiela ďalej, napr. pre zachovanie anonymity účastníkov.

Jediným podporovaným signalizačným protokolom je SIP. Poslednou stabilnou je verzia 4.1.6. [24]

Kombináciou schopností SIP jadra a prispôsobiteľných API (podpora Java, Perl, Python, C#) je možné dosiahnuť veľkú škálovateľnosť. Z hľadiska bezpečnosti podporuje Digest autentifikáciu, autorizáciu cez ACL (Access Control List), podporu TLS u SIP signalizácie a SRTP pre zabepečenie hlasu. Podporuje takisto známe databázové systémy (MySQL, SQLite, Oracle) a vzájomnú kooperáciu s ostatnými open source PBX.

1.4.1 Architektúra pobočkovej ústredne Kamailio

Využíva modulárnu architektúru pozostávajúcu z jadra s implementovanými knižnicami a modulov. Jadro obsahuje komponenty na riadenie pamäte, analyzátor správ a riadenie prenosových služieb. Ilustrácia architektúry je na nasledujúcom obrázku.



Obr. 1.2: Architektúra Kamailio [24]

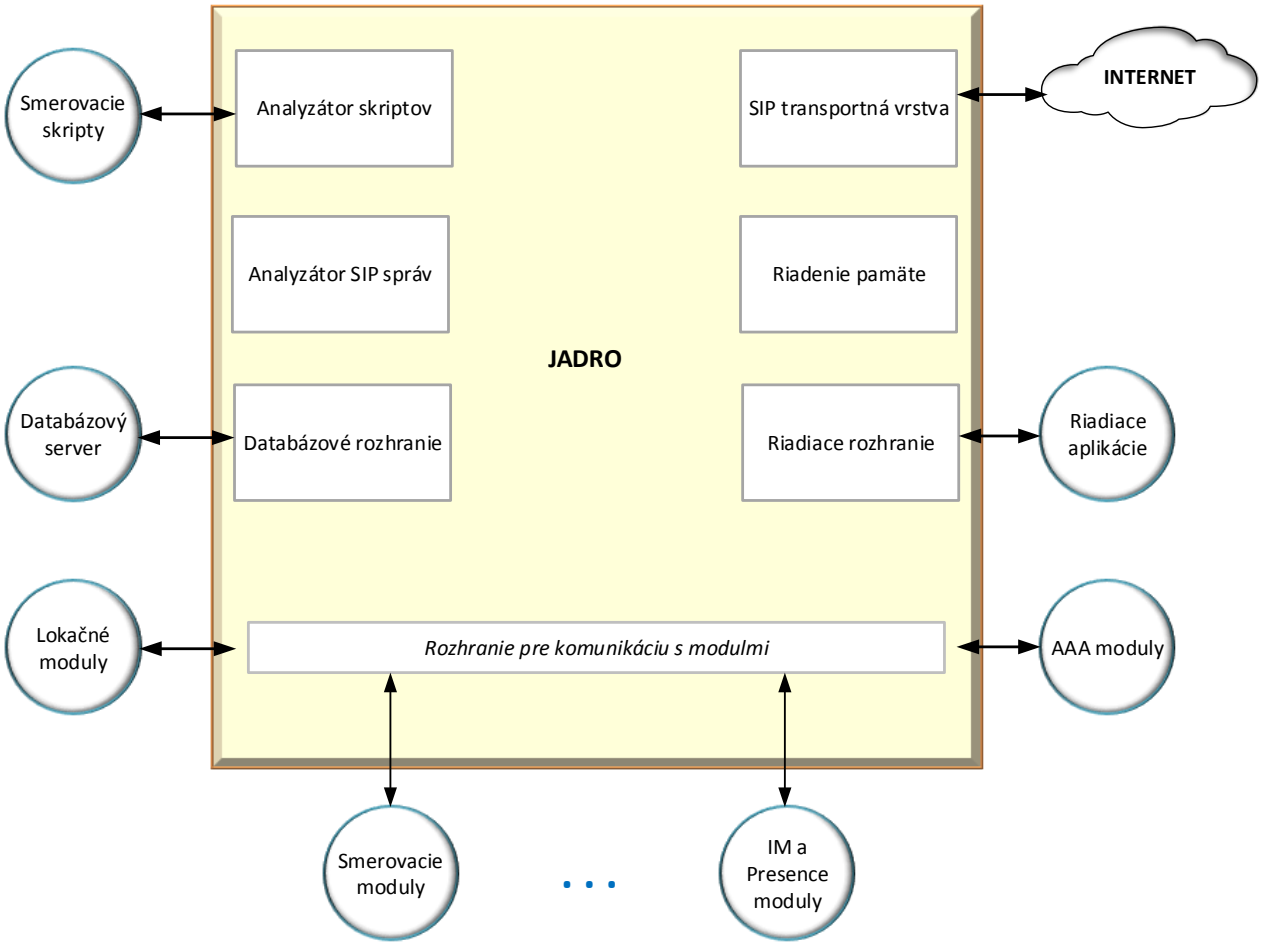
1.5 OpenSIPS

Vznikol v roku 2008, keď sa časť vývojárov OpenSER odtrhla a vytvorila vlastný projekt. Ten je doteraz vyvíjaný nezávisle, aj keď pôvodný OpenSER sa zlúčil s Kamailio. Je šírený pod licenciou GPL a bol napísaný v jazyku C. V súčastnosti je najaktuálnejšia LTS verzia 1.11.3 a v testovacej fázi vývoja je verzia 2.1.1. [30]

Rovnako ako Kamailio je navrhnutý ako signalizačná ústredňa a je zameraná iba na SIP protokol. Z toho vyplýva, že nespracováva tok mediálnych dát a nedokáže transkódovať hlasové dáta medzi rôznymi kodekmi, ani nemá implementované iné hlasové služby ako IVR a záznamník.

1.5.1 Architektúra pobočkovej ústredne OpenSIPS

S Kamailio má podobnú aj architektúru. Takisto bol navrhnutý ako modulárny, pričom jadro je tvorené iba komponentami na správu nižších vrstiev ako riadenie pamäti, analyzátor správ, riadiace i databázové rozhrania a synchronizačné a konfiguračné mechanizmy. Moduly pridávajú ďalšie dodatočné funkcionality ako je vidieť na obrázku. [31]



Obr. 1.3: Architektúra OpenSIPS [31]

2 Protokoly VoIP

VoIP využíva dva druhy protokolov - *signalizačné* (určené na signalizáciu) a *transportné* (na samotný prenos mediálnych dát).

2.1 Signalizačné protokoly

slúžia k naviazaniu spojenia, riadeniu toku a jeho ukončovaniu. Najpoužívanejšie z nich sú porovnané v tabuľke 3.1. Dajú sa rozdeliť na dve kategórie:

- ***Session Control protokoly*** – sú zodpovedné za vytvorenie, udržiavanie a ukončovanie hovorových spojení, sú zodpovedné za vyjednanie parametrov prenosu ako kodeky, tóny, šírky prenosového pásma. Sem patria SIP a H.323.
- ***Media Control protokoly*** – sú zodpovedné za vytvorenie a ukončovanie médiových spojení, komunikáciu s bránami (*media gateways*) medzi IP a PSTN sieťou. Sem patria MGCP a Megaco. [9]

Signalizačné protokoly však samotný prenos mediálnych dát neuskutočňujú, tie prenáša typicky RTP. Zvláštnym prípadom je protokol IAX, ktorý v sebe zahŕňa aj signalizáciu aj prenos audio/video dát.

2.2 Protokol SIP

Signalizačný protokol, ktorý bol prvý raz definovaný štandardom RFC 2543 organizácie IETF. [13] Je založený na princípe HTTP (využíva takisto žiadosti a odpovede) a je alternatívou k H.323. Podobne ako on aj SIP je používaný s ďalšími IETF protokolmi ako SDP, RTSP a SAP. Jeho základné funkcie sú vyhľadanie volaného pri inicializácií hovoru, špecifikovanie vlastností volajúcich zariadení (použiteľné kodeky), prenos informácii o dostupnosti a nastavenie i riadenie relácie.

2.2.1 Adresovanie

Princíp signalizácie spočíva vo výmene správ v relácii (spojení) medzi učastníkmi. Signalizácia prebieha zvlášť od prenosu hlasových dát (sú logicky oddelené) čo je dôležité, pretože signalizačné dáta môžu prechádzať viacerými *proxy* alebo *redirect* servermi a tok hlasových dát ide priamejšou cestou [1]. Na správu multimediálnej relácie je využívaný protokol SDP. Pred zahájením relácie je dôležité najprv účastníka presne určiť, aby ho bolo možné adresovať. Ten je identifikovateľný vďaka *SIP URL* resp. *URI.* Tá je podobná emailovej adrese a je zložená z dvoch častí – užívateľského mena (čísla) a názvu domény, vyzerá takto:

`SIP:meno@domena.com` . [14]

Voliteľnou zložkou je číslo portu (najčastejšie *5060* [16]), ktorý je definovaný za názvom domény, heslom a pred ďalšími parametrami. Adresná hlavička tak môže mať tvar:

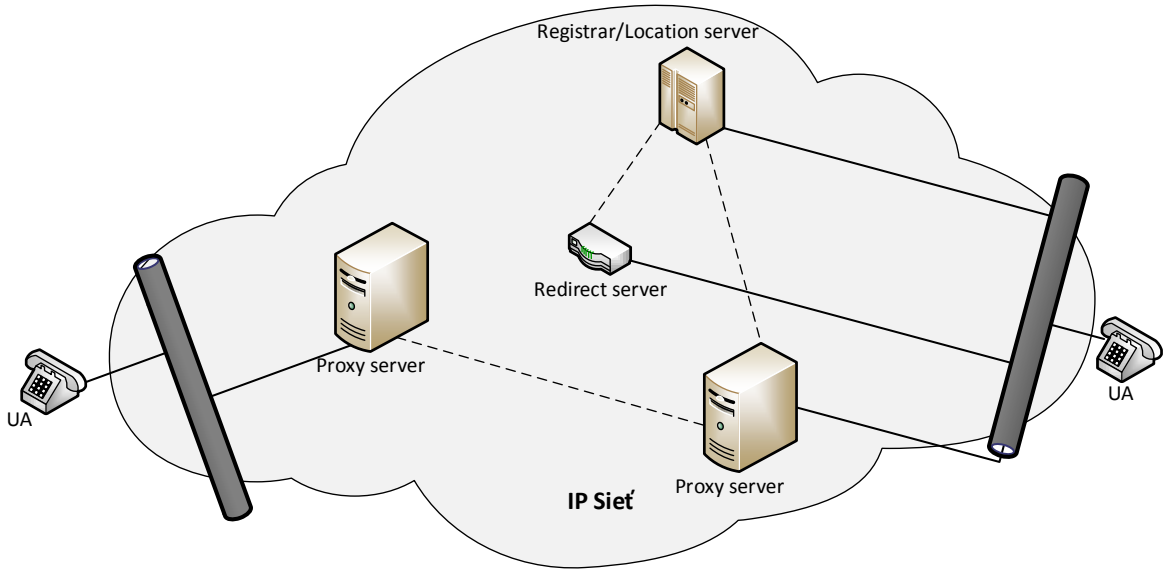
```
sip:<user>[:<password>]@<host>[:<port>][;<uri-parameters>][?<headers>]
```

Podľa RFC 3261 existujú aj zabezpečené adresy, tzv. *SIPS URI.* Tie sú zašifrované pomocou *SSL* resp. *TSL*. Ako predvolený port pre zabezpečenú komunikáciu sa používa *5061*. [15]

2.2.2 Architektúra

Podobne ako štandard *H.323* aj SIP definuje niekoľko elementov v každej VoIP sieti, ktorá využíva tento protokol. V základe sa dajú rozdeliť na *klientov* a *servery,* pričom klient je aplikačný program, ktorý vysiela SIP požiadavky (*request*) a servery na ne reagujú. Z toho vyplýva, že SIP je tzv. klient-server protokol. Príklad architektúry SIP je na obrázku 2.1. Definované sú tieto druhy elementov:

- **User Agent (UA)** – Klient. Generuje a posiela žiadosti, prijíma i spracováva odpovede. Okrem neho existujú aj tzv. *Presence Agent* (prijíma registračné požiadavky a generuje stavové notifikácie, zbiera prezenčné informácie) a *Back-to-back User Agent* (prijaté požiadavky preformuluje a preposiela ďalej, napr. pre zachovanie anonymity účastníkov). [16]
  Klient (UAC) je v praxi obsiahnutý v koncovom zariadení spolu s UA Serverom (UAS), ktorý generuje odpovede na prijaté žiadosti, ktoré posiela späť a udržuje signalizáciu stavu hovorov, v ktorých sa zúčastňuje.
- **Proxy Server** – Jeho primárnou funkciou je smerovanie požiadaviek a odpovedí. Po preposlaní sa správa javí iným serverom akože pochádza od Proxy servera a nie od klienta za ním. Obsahuje databázu, podľa ktorej riadi smerovanie na ďalší skok. Dokáže žiadosť preposlať na viacero zariadení súčasne (*forking*) a zaistiť autorizáciu i autentifikáciu užívateľov. [14]
- **Redirect Server** – Prijíma požiadavky, zmapuje cieľové adresy a pošle ich naspäť klientovi. Ten tak môže priamo odoslať požiadavku na správnu adresu. Redirect Server dokáže na základe priorít vybrať vhodnú adresu. Nie vždy však pošle koncovú adresu, môže poslať adresu najbližšieho servera, ktorý by ju mal poznať. Na rozdiel od *Proxy Servera* teda požiadavky len prijíma a odpovedá na ne. [14]
- **Registrar Server** – Prijíma registračné požiadavky. Keďže SIP uplatňuje princíp registrácie, čím určuje, že registrovaný účastník je dostupný na registrovanej adrese. Prispieva tak k mobilite užívateľov. Typicky býva v kombinácii s iným serverom (*Redirect* alebo *Proxy*). [1]



Obr. 2.1: Príklad SIP architektúry [17]

2.2.3 Správy

Na signalizáciu tento protokol využíva SIP správy, ktoré je možné rozdeliť do dvoch kategórii: žiadosti (*requests* alebo *methods*) a odpovede (*responses*). Správy majú nasledujúcu štruktúru:

- štartovací riadok,
- jeden alebo viac riadkov hlavičky,
- prázdny riadok indikujúci koniec hlavičky,
- voliteľné telo správy.

Pričom štartovací riadok je u žiadostí nazvaný *Request Line* a má napr. takúto štruktúru: `INVITE sip:9103682854@192.168.16.140 SIP/2.0`. [17]

2.2.4 Žiadosti a odpovede

V doporučení RFC 3261 je definovaných šesť žiadostí: *INVITE*, *ACK*, *OPTIONS*, *BYE*, *CANCEL* a *REGISTER*. Ostatné (*INFO*, *REFER*, a *UPDATE*) sú definované v iných odporučeniach [15]. Sú vysielané klientom. V tabuľke 2.1 je prehľad rôznych druhov SIP žiadostí, popis ich funkcie a odporučenie, v ktorom boli uvedené.

Tab. 2.1: Typy žiadostí SIP [15]

Typ žiadosti	Funkcia	RFC odporučenie
INVITE	určená k nadväzovaniu spojenia, alebo k zmene parametrov existujúceho spojenia	RFC 3261
ACK	slúži ako potvrdenie od užívateľa, že prijal požiadavku	RFC 3261
CANCEL	žiadosť o zrušenie prebiehajúcej žiadosti **INVITE**	RFC 3261
REGISTER	žiadosť o registráciu klienta v *Registrar* serveri	RFC 3261
OPTIONS	slúži k výmene informácii podporovaných funkcií	RFC 3261
BYE	žiadosť o ukončenie spojenia	RFC 3261
REFER	požiadavka iného UA k relácií (napr. inicializácia spojenia cez web)	RFC 3515
SUBSCRIBE	slúži ako požiadavka pre zistenie stavu vzdialeného prvku	RFC 3265
NOTIFY	slúži k informovaniu koncového prvku	RFC 3265
UPDATE	aktualizácia informácie o stave relácie	RFC 3331
INFO	slúži k prenosu informácie počas relácie	RFC 2976

Odpovede sú odpoveďou servera na prijatie žiadosti klienta a sú rozdelené do šiestich kategórii po stovkách označených od 100 do 699. Všetky sú zahrnuté v doporučení RFC 3261. Ich názorne rozdelenie s vysvetlením princípu zobrazuje nasledujúca tabuľka.

Tab. 2.2: SIP odpovede [15]

Označenie	Funkcia	Príklady správ
1xx	Informačná odpoveď, požiadavka bola prijatá a spracováva sa (dočasná odpoveď)	100 Trying 180 Ringing
2xx	Úspech, tj. kladná odpoveď, požiadavka bola úspešne prijatá a spracovaná (finálna odpoveď)	200 OK 202 Accepted
3xx	Presmerovanie	302 Multiple Choices 305 Use Proxy
4xx	Chyba na strane klienta	487 Request Terminated 403 Forbidden
5xx	Chyba na strane servera	501 Not Implemented 503 Service Unavailable

6xx	Globálna chyba	600 Busy Everywhere 603 Decline

2.2.5 Autentizácia

SIP k nej používa princíp žiadosť – odpoveď. Je založený na HTTP Digest Autentifikácii (pozri kap. 3.5.1). Ak User Agent (UA) odošle žiadosť REGISTER alebo INVITE k ústredni, ktorá vyžaduje autentifikáciu, tak tá mu obratom odošle odpoveď 401 alebo 407 oznamujúc, že je vyžadovaná autentizácia. V odpovedi je pole *nonce* (nazývané aj *challenge*), ktoré obsahuje špecifickú hodnotu, ktorú bude UA musieť použiť pri zaslaní údajov na autentifikáciu. [8]

2.3 Protokol IAX

Inter-Asterisk Exchange, je natívny protokol ústredne Asterisk. Bol navrhnutý Mark Spencerom ako alternatívny k SIP a H.323. Ako bolo už spomenuté, má na starosť okrem signalizácie a riadenia aj samotný prenos dát. Okrem hlasových dát dokáže prenášať aj video. Jeho hlavnou výhodou oproti SIP je prenos obidvoch typov dát (signalizačných aj hlasových, resp. video) rovnakým portom 4569. To uľahčuje konfiguráciu a diagnostiku externých pripojení na vnútornej strane za firewallom a NAT. Ďalšou výhodou je menšia náročnosť na šírku prenosového pásma, ktorá je dosiahnutá vďaka zahrnutiu paketov pre viacero hlasových rámcov do jedného datagramu s jednou IAX hlavičkou. [8]

Od roku 2010, sa začala používať jeho druhá štandardizovaná verzia IAX2, ktorá priniesla navyše natívnu podporu šifrovania dát pomocou šifry AES-128. Tá tak umožňuje dáta prenášané protokolom chrániť pred odposluchom a nahradzuje tak TLS+SRTP ochranu používanú na utajenie dát prenášaných SIP protokolom. Navyše druhá verzia je bitovo orientovaná, čo umožňuje efektívnejšie využitie prenosových prostriedkov.[26]

2.3.1 Architektúra

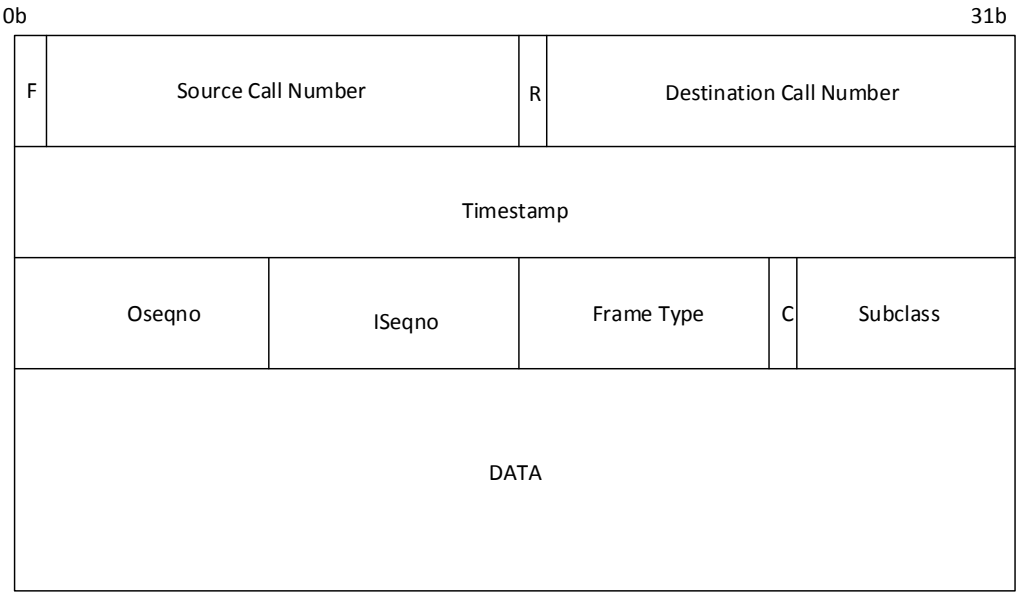
U IAX2 se jedná o typ klient-server. Klientom je softwarový alebo hardwarový telefón nachádzajúci sa vo VoIP sieti, ktorý používa IAX2 pre vytvorenie relácie a prenos dát. Server – ústredňa, má na starosti registráciu a presmerovávanie hovor.

Použitie tohoto protokolu by sa dalo rozdeliť do dvoch prípadov. Prvým je tunelovanie (trunkovanie) medzi ústredňami, ktoré ho podporujú. Druhým je plnohodnotná náhrada SIP a RTP k telefonickej komunikácii.

2.3.2 Rámce

IAX2 využíva viacero typov rámcov, každý ma svoj špecifický účel.

Full rámce – Nesú signalizačné dáta aj multimediálne dáta. Používajú sa pri počiatočnom nastavení hovoru, pri jeho ukončení aj pri synchronizácii časového razítka (timestamp 32B). Jedná sa teda o riadiace rámce, ktoré sú po úspešnom prijme na druhej strane potvrdené (správa ACK). Štruktúru je vidieť na nasledujúcom obrázku. Najbežnejšie sú uvedené v tabuľke 2.3 s príslušnou hodnotou. [27]

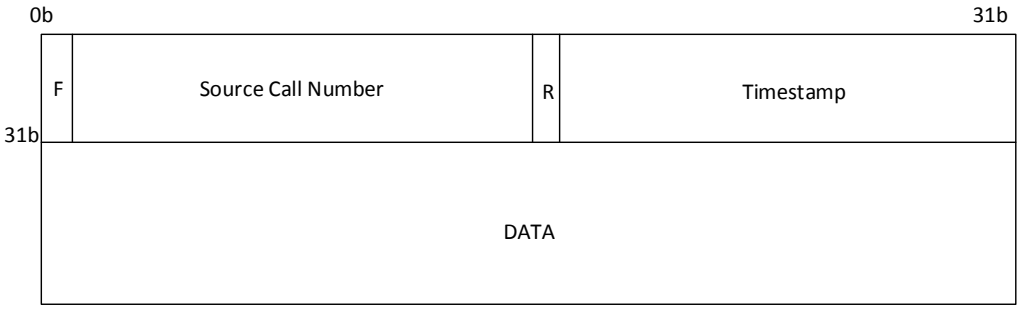


Obr. 2.2: Štruktúra Full rámca [27]

Tab. 2.3: Vybrané typy FULL rámcov u IAX2 [26]

Hodnota	Názov	Funkcia	RFC odporučenie
0x01	**NEW**	Vytvor nový hovor	RFC 5457
0x02	**PING**	Ping žiadosť	RFC 5457
0x04	**ACK**	Potvrdenie	RFC 5457
0x07	**ACCEPT**	Prijatie hovoru	RFC 5457
0x06	**REJECT**	Zamietnutie hovoru	RFC 5457
0x05	**HANGUP**	Zrušenie hovoru	RFC 5457

- **Mini rámec** - Prenáša multimediálne dáta počas spojenia. Časové razítko je polovičné (16B) a chýbajú v ňom bloky na zabezpečenie doručenia tj. neni potvrdzované jeho doručenie. Jeho štruktúra je znázornená na nasledujúcom obrázku.



Obr. 2.3: Štruktúra Mini rámca [27]

- **Meta rámec –** Slúži na dva účely a podľa toho sa aj nazýva – video alebo trunk. Meta video rámce umožňujú prenos video dát s optimalizovanou hlavičkou a Meta trunk rámce slúžia na zlučovanie viacerých dátových tokov do jednej hlavičky, čo prispieva k minimalizácii nárokov na šírku pásma.

Full rámce môžu mať viacero odlišných typov, definovaných v hlavičke v poli Frame Type. Patria medzi ne Voice, Video, Text (nesú rovnaké dáta) alebo riadiaci tzv. Control. Každý riadiaci rámec má svoju podtriedu, uvedenú príslušnou hodnotou v hlavičke rámca, najčastejšie používané sú v tabuľke 2.4.

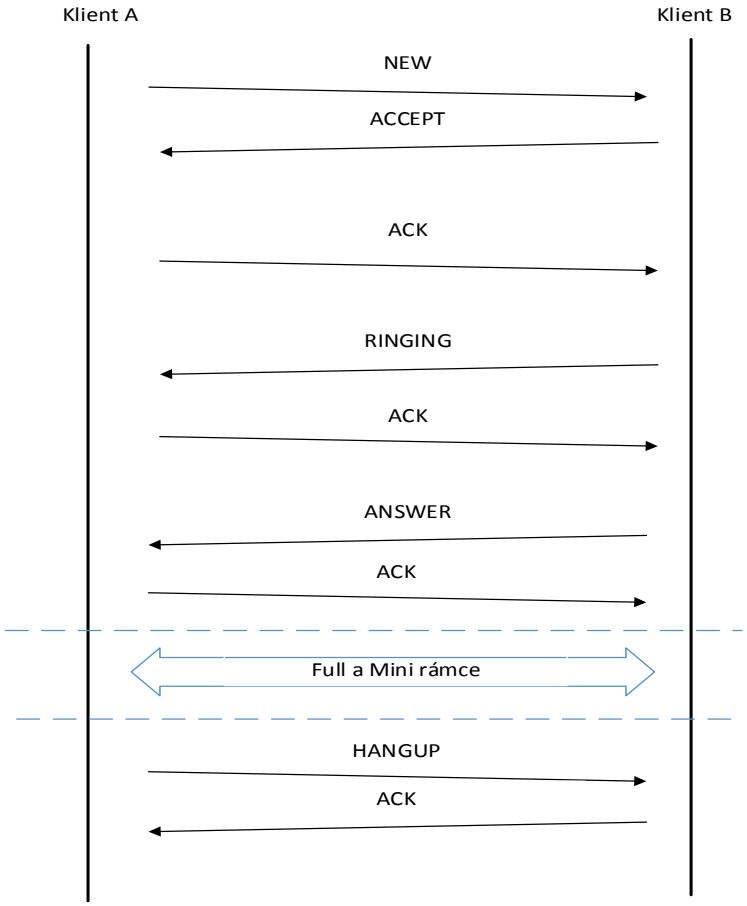
Tab. 2.4: Vybrané typy riadicich FULL rámcov u IAX2 [26]

Hodnota	Názov	Funkcia	RFC odporučenie
0x01	**Hangup**	Hovor bol položený vzdialenou stranou	RFC 5457
0x03	**Ringing**	Vzialený koniec zvoní	RFC 5457
0x04	**Answer**	Vzialený koniec hovor zdvihol	RFC 5457
0x05	**Busy**	Vzialený koniec je obsadený	RFC 5457
0x0E	**Call Progress**	Hovor prebieha	RFC 5457
0x10	**Hold**	Hovor je podržaný	RFC 5457

2.3.3 Priebeh spojenia

Klient, ktorý zahajuje komunikáciu vyšle druhej strane žiadosť NEW o začatie relácie. Druhá strana môže odpovedať ACCEPT (povolí), REJECT (odmietne a uvedie dôvod) alebo AUTHREQ (vyžiadanie autentifikácie).

Typické počiatočné spojenie a vytvorenie relácie u protokolu IAX2 prebieha nasledujúcou výmenou správ (obr. 2.4).



Obr. 2.4: Priebeh spojenia u IAX [32]

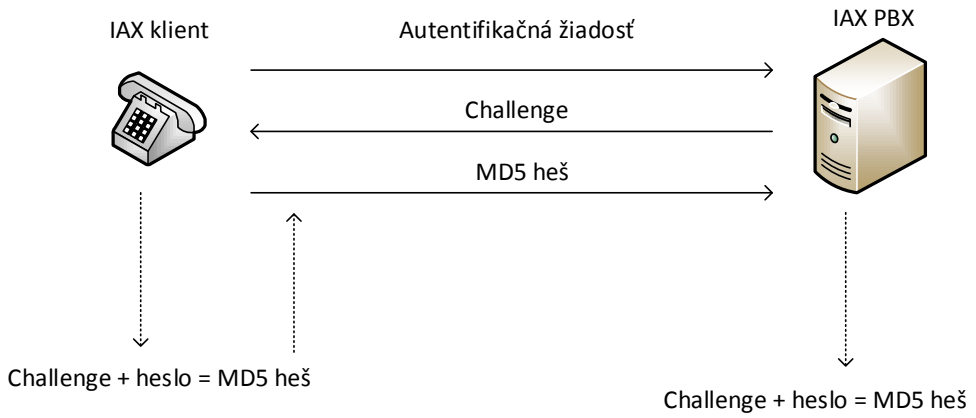
2.3.4 *Autentizácia*

IAX2 podporuje tri druhy autentizačných metód:

- Čistým textom (plaintext)
- MD5
- RSA (pomocou AES šifry)

Prvá uvedená neposkytuje žiadnu ochranu prenášaného užívateľského mena a hesla voči odposluchu, či zneužitiu a robí tak túto možnosť použiteľnú len v krajných prípadoch. Posledná uvedená nie je napriek svojej vysokej bezpečnosti, vďaka používaniu verejných a privátnych klúčov, takmer vôbec implementovaná v súčastných riešeniach.

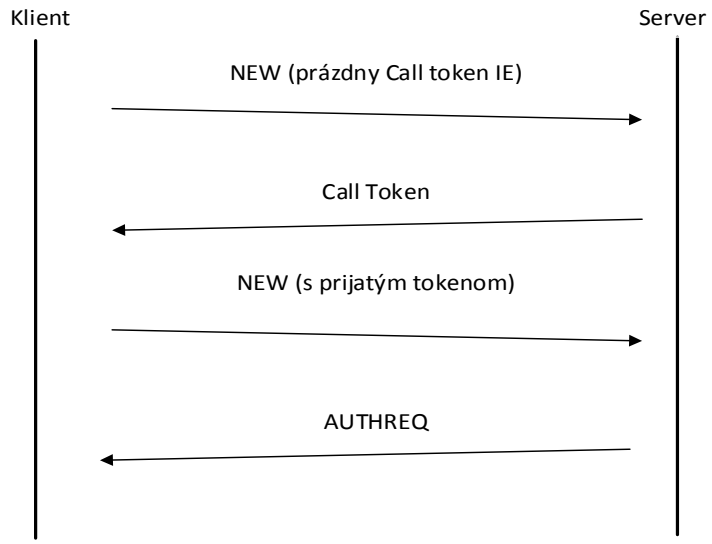
Z ďaleka najpoužívanejšou metódou je hešovacia MD5. Heslo pre reláciu je teda prenášané v zahešovanom tvare a nie je ho vidieť v čistom tvare. Metóda je založena ná požiadavke - odpovedi. Výhodou oproti SIP je, že heš pre rovnaké heslo nie je rovnaký, pretože ústredňa, ktorá autorizuje posiela pri každom autentifikačnom procese klientovi tzv. *challenge* (číslený kód). Ten k nemu pripojí svoje heslo, údaj zahešuje a odošle ústredni. Proces je vidieť na obrázku. [8]



Obr. 2.5: Autentizačný proces u IAX protokolu [8]

2.3.5 *Call token validácia*

Je voliteľná funkcia IAX2 protokolu od tvorcov Asterisku slúžiaca ako ochrana pred útokom odmietnutia služby (DoS). Číslo hovoru tzv. *Call number,* určuje v protokole IAX2 priradenie jednotlivých správ k danému hovoru. V protokole SIP je podobný parameter *Call-ID*. Pomocou nastavenia limitu pre tento parameter sa dosiahne zvýšená ochrana. Priebeh validácie je vidieť na obrázku 2.6. Call Token v sebe nesie zahešované údaje ako IP adresa, port, časové razítko a informácie ústredne. Validácia zaručuje aj ochranu pred komunikáciou zo zneužitej IP adresy. Nakoľko sa jedná o voliteľnú funkcionalitu, nemusia ju klienti podporovať a preto sa pri jej aktivácii na ústredni treba uistiť, že je klientom podporovaná. [34]



Obr. 2.6: Priebeh Call Token validácie [34]

2.4 Transportné protokoly

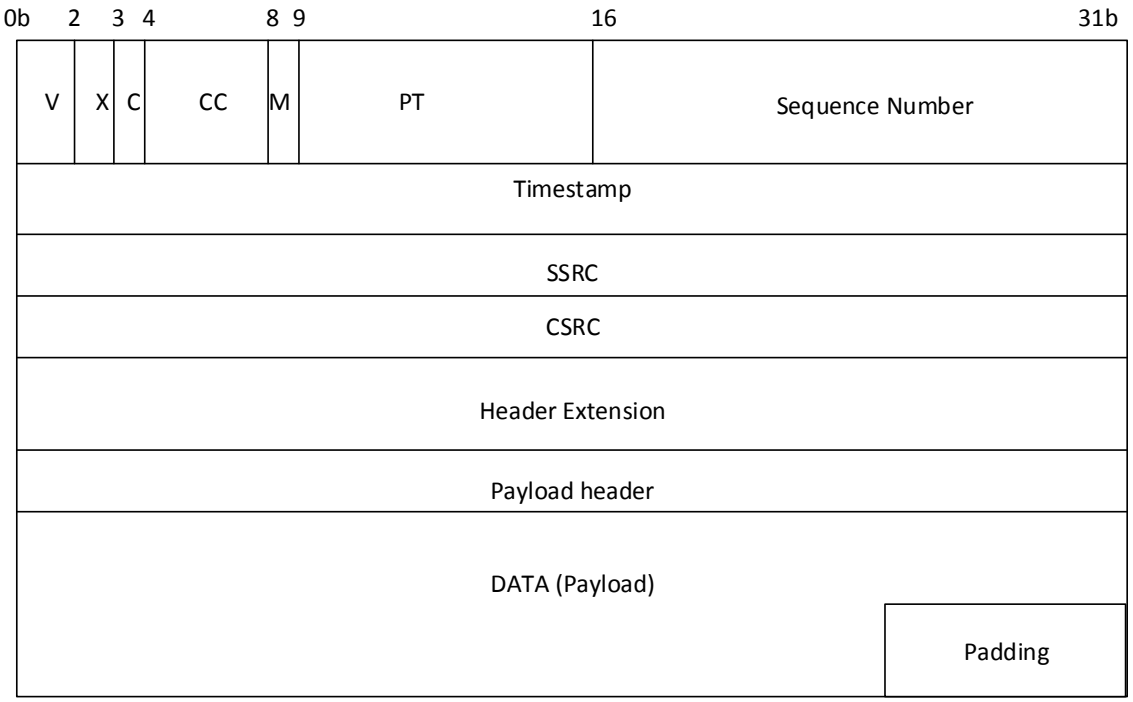
Ako už bolo spomenuté v úvode kapitoly, pre prenos multimediálnych dát je na transportnej vrstve uprednostňovaný UDP nad TCP. Dôvodom je práve to, že UDP protokol ako nespoľahlivý, je oveľa menej náročnejší na prenos a zároveň je rýchlejší, pretože nevyžaduje potvrdenia o doručení. Pretože ako vyplýva z vlastností prenosu hlasu, strata jedného alebo dvoch paketov, ktoré nesú milisekundy reči nespôsobí žiadne výrazné problémy, no na druhej strane práve ich meškanie spôsobuje neschopnosť plynulej komunikácie.

Medzi aplikačné protokoly najčastejšie využívané VoIP sieťami na prenos patria RTP, RTCP a RTSP, ktoré slúžia k vlastnému prenosu multimediálnych dát, tj. audia a videa a ich bezpečnostné alternatívy SRTP a SRTCP, o ktorých pojednávajú nasledujúce podkapitoly.

2.4.1 Protokol RTP

Bol definovaný v odporučení RFC 3550. Nezaručuje včasné doručenie dát, ani negarantuje kvalitu služieb. [19] Ako bolo spomenuté v úvode do kapitoly transportných protokolov, UDP nie je schopné kontrolovať doručenie paketov, ale RTP pomáha pomocou sekvenčného čísla paketu, aby si prijímateľ mohol kontrolovať stratené pakety alebo časovú známku, pomocou ktorej si môže vypočítať aj meškanie doručených paketov.

RTP prenáša digitálne kódovaný hlas tak, že vezme niekoľko digitálnych vzoriek (záťaž – *payload*) a priradí im hlavičku [21]. Hlavička RTP paketu má štruktúru zobrazenú na obr. 2.7. Kde „V“ značí verziu, „X“ prítomnosť rozšírenia hlavičky, „CC“ CSRC (contributing source – id prispievateľa) kontrolný súčet, „M“ značku a „PT“ typ záťaže. Popis funkcionality je nad rámec práce.



Obr. 2.7: RTP hlavička [20]

2.4.2 Protokol RTCP

Je súčasťou celku, ktorý tvorí spolu s RTP protokolom. Na rozdiel od neho sa stará o kontrolu a riadenie spojenia. Takisto je definovaný odporučením RFC 3550 a na prenos používa UDP protokol. Obvykle používa číslo portu o jedno vyššie ako RTP.

Jeho hlavnou úlohou je teda periodická výmena správ medzi účastníkmi s cieľom zabezpečiť uspokojivú kvalitu spojenia. Medzi ďalšie funkcie patrí identifikácia a odhad veľkosti relácie (sleduje počet účastníkov, ktorý periodicky o sebe dávajú vedieť).

RTCP pracuje s viacerými druhmi paketov, ale nikdy neposiela iba jeden druh samostatne, kombinuje viaceré do jedného. Podľa odporučenia RFC 3550 musí každý celkový paket obsahovať SR (Sender Report) alebo RR (Reciever Report) a SDES (Source Description) paket. [23]

2.5 Bezpečnostné protokoly

2.5.1 Protokol SRTP

Je bezpečnostnou nadstavbou protokolu RTP, ktorá poskytuje zaistenie dôveryhodnosti, autentifikácie správ a ochranu odpovedí pre RTP a RTCP. Definuje pre ne kryptografické transformácie a nechavá priestor pre ich rozšírenie do budúcnosti. S vhodnou správou kľúčov predstavuje bezpečné riešenie pre prenos multimediálnych dát v unicast (prenos od jedného k jednému) ale aj multicast prenose (od jedného k viacerým). Dosahuje vysokú priepustnosť a nízky nárast veľkosti paketu. Je vhodným na prenos dát rôznymi technológiami (drôtovými i bezdrôtovými).

SRTP paket obsahuje navyše oproti RTP dve polia. Pole pre voliteľný MKI (Master Key Identifier), ktoré definuje hlavný kľúč tzv. *master key*. Z neho sú následne odvodzované tajné symetrické kľúče tzv. *session keys*, pomocou ktorých sú šiforvané dáta. Na začiatku spojenia si obidve strany externe vymenia zvolený hlavný kľúč pomocou protokolu na bezpečný prenos kryptografických kľúčov sieťou. Najpoužívanejšie sú MIKEY (Multimedia Internet Keying), ZRTP (Z and Real-time Transport Protocol, Z podľa mena jeho tvorcu Philla

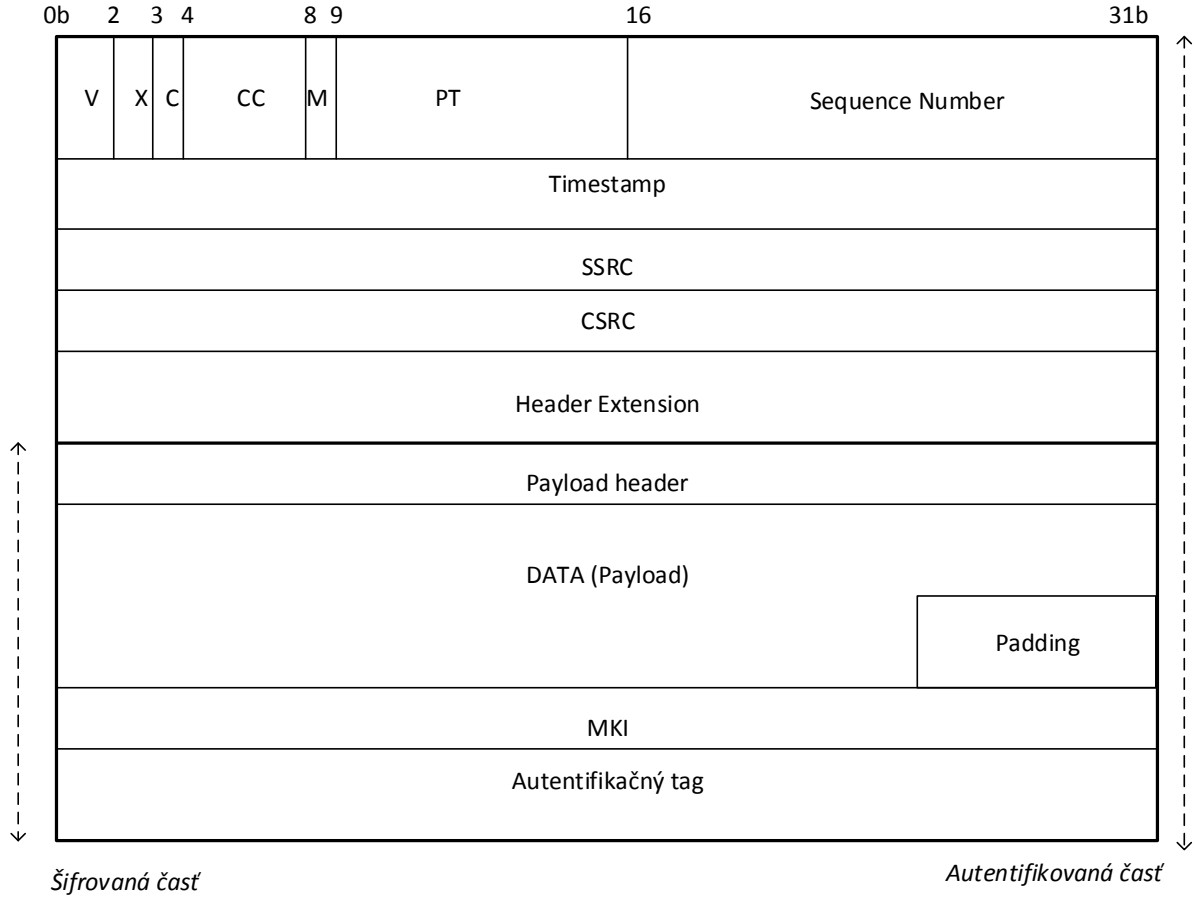
Zimmermanna) alebo DTLS-SRTP (Datagram Transport Layer Security for SRTP). Odvodené šifrovacie a autorizačné kľúče sú platné vždy pre danú reláciu a vďaka mechanizmu na ich periodické odvodzovanie nie je možné pre útočníka, aj keď zachytí jeden z nich, odvodiť klúče pre predošlú alebo nasledujúcu reláciu.

Druhým poľom je Autentication tag**.** Zaisťuje ochranu integrity dát v RTP pakete pomocou zašifrovaného kontrolného súčtu hlavičky a tela paketu. Umožňuje ju algoritmus HMAC – SHA1, ktorý generuje a hešuje bitovú značku, tá je pripojená k paketu a slúži k jeho overeniu.

Šifrovanie v protokole SRTP je realizované pomocou AES šifry v dvoch režimoch pre generovanie prúdového kľúča:

- SICM - Segment Integer Counter Mode, režim čítača umožňujúceho náhodný prístup k všetký blokom, ktoré sú nevyhnutné pre RTP a ktoré sú prepravované cez nespoľahlivé siete s možnou stratovosťou paketov.
- F8 – Jedná sa o upravenú spätnú väzbu so zlepšeným vyhľadávaním a zmenou inicializačnej funkcie. Využíva sa v UMTS 3G sieťach.

Algoritmus umožňuje spracovávať pakety v dopredu neurčenom poradí, čo je nevyhnutné pre dáta prenášané nespoľahlivým UDP. [33]



Obr. 2.8: Hlavička SRTP [33]

2.5.2 Protokol TLS

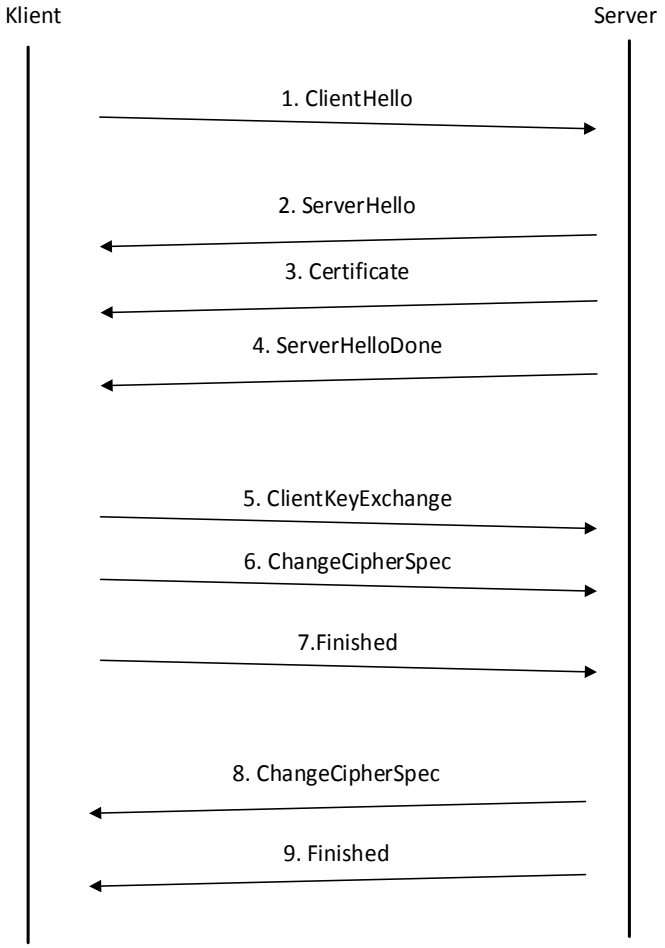
Vychádza z protokolu SSL (Secure Socket Layer), jeho úlohou je zabezpečiť komunikáciu/prenos rovnako ako „konkurenčný“ IPsec. Beží na transportnej vrstve pod aplikačnými protokolmi a je od nich nezávislí. Dokáže pridať zabezpečenie ľubovoľnému aplikačnému protokolu používajúcemu spoľahlivé spojenie. Takisto je použiteľný vo VPN sieťach a pri tunelovaní. Tam je však viac využívaný IPSec. Z hľadiska VoIP technológii sa používa ako ochrana SIP signalizácie. Dokáže jej poskytnúť autentizáciu a šifrovanie. To sa

deje vrátane výmeny kryptografických kľúčov a dohody konkrétneho algoritmu, ktorý bude použitý na šifrovanie, cez tzv. *TLS Handshake* (na obrázku 2.6) pred začiatkom prenosu dát.

TLS obsahuje tri fáze:

- dohoda účastníkov na podporovaných šifrovacích algoritmoch
- výmenú šifrovacích kľúčov a autentizáciu na základe certifikátov
- a šifrovanie provozu.

Na samotné šifrovanie sa používa symetrická šifra (AES, DES), kľúče sú v nej generované unikátne pre každé spojenie. Integrita dát sa overuje pomocou MAC (Message Authetication Code), ktorý je vypočítaný s hešovacou funkciou (SHA-1) a prenesený v poslednej správe TLS Handshake-u (*Finished*). [35]



Obr. 2.9: TLS Handshake [36]

Zahájenie TLS Handshake-u začína klient poslaním správy *Client Hello*. Táto správa obsahuje informácie o verzii TLS, ktorú klient podporuje, náhodné číslo, kompresné metódy a odporučené šifrovacie sady. Na ňu nasleduje odpoveď serveru *Server Hello*. Tá obsahuje informácie, ktoré budú použité na vytvorenie zabezpečeného spojenia. Server následne posiela vygenerovaný certifikát a správu *Server Hello Done*, ktorou klienta informuje o dokončení úvodnej relácie. Klient následne posiela správu *Client Key Exchange,* ktorá obsahuje tajný kľúč zašifrovaný pomocou RSA a verejného kľúča a *Change Cipher Spec,* ktorou oznámi serveru, že ďalšia komunikácia bude už šifrovaná. Následne posiela *Finished* správu. Tá obsahuje heš a MAC predchádzajúcich inicializačných správ. Server po overení následne takisto posiela správy *Change Cipher Spec* a *Finished*. Týmto je nadviazanie spojenia úspešné a je zahájená šifrovaná komunikácia.

V súčasnosti je najmodernejšou verzia 1.2, ktorá je flexibilnejšia a úspornejšia na veľkosť prenášaných dát než prvotná verzia. [36]

2.5.3 Sada IPSec

Definované v RFC 2401, 2406, 2409 a 2411 je protokolová sada postavená na tretej sieťovej vrstve. Je teda aplikačne nezávislá a poskytuje ochranu aj protokolom na transportnej vrstve. Poskytuje utajenie, ochranu integrity, autentizáciu zdroja dát a ochranu pred replay útokmi (pozri kapitola 3.3) pomocou šifrovania a podpisovania každej správy. Keďže je definovaná kombináciou viacerých RFC popisuje dva hlavné podprotokoly: Authentication Header (AH) a Encapsulating Security Payload (ESP). Druhý menovaný je preferovanejší, nakoľko poskytuje autentizáciu aj dôveryhodnosť. AH zaisťuje len autentizáciu.

Aby mohlo byť realizované bezbečné spojenie tj. SA (Security Association) medzi dvoma koncovými účastníkmi, je najprv nutné dynamicky dohodnúť tzv. key management protokol. Bežne je k tomu využívaný IKEv1/v2 (Internet Key Exchange), ktorý využíva Diffie-Hellmanov kryptografický protokol, používajúci symetrickú šifru.

IPsec je v súčastnosti asi najviac využívaný u VPN (Virtual Private Network) spojení a to najmä vo veľkých firmách. Na zabezpečenie SIP sa používa len výnimočne. [11]

3 Problematika útokov a zabezpečenia

3.1 Bezpečnosť a jej aspekty

U všetkých IP technológii zohráva dôležitú úlohu bezpečnosť. Jej hlavné aspekty, počnúc autentizáciou, autorizáciou, cez dostupnosť a utajenie komunikácie až po ochranu integrity je potrebné, viac než kedykoľvek predtým, dosiahnuť najmä v dnešnej dobe. Dobe plnej rozsiahlych útokov hackerských skupín, špionáží konkurenčných spoločností a odpočúvania vládnych agentúr a organizácii. Jednou z najznámejších organizácii, ktoré plnia toto poslanie je VOIPSA - VoIP Security Alliance. Jej náplňou je spolupráca s výrobcami, poskytovateľmi VoIP riešení a výskumnými akademickými tímami k dosiahnutiu vyššej bezpečnosti v tejto oblasti. K tomu je dôležité zohľadnenie nasledujúcich aspektov:

Autentifikácia

Alebo aj autentizácia, prebieha na úrovni riadenia spojenia, keď sa koncový užívateľ registruje a inicializuje hovor. Uskutočňuje sa medzi VoIP telefónom (či už v softwarej alebo hardwarovej forme) a podporným serverom ako SIP Registrar, H.323 Gateway alebo IAX serverom. Protokoly prenosu mediálneho toku ako RTP ho nepožadujú, nakoľko ju používajú práve signalizačné protokoly.

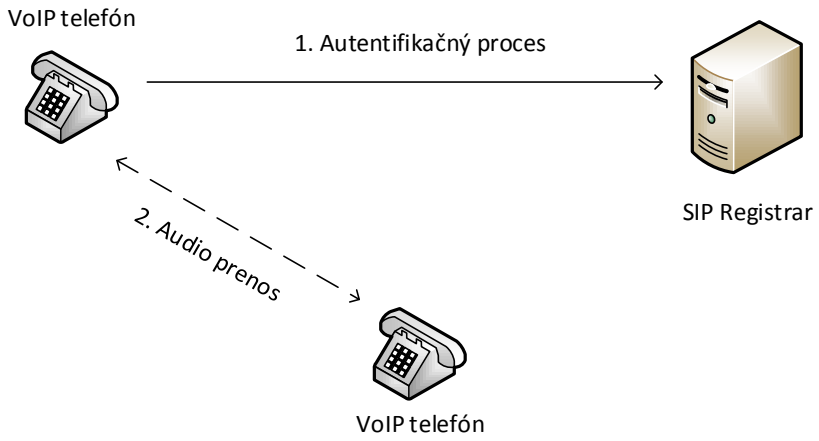
Najpoužívanejšie autentifikačné typy pre konkrétne signalizačné protokoly:

SIP - Digest autentifikácia, ktorá vychádza z HTTP. Využíva MD5 heš.

H.323 - MD5 heš alebo všeobecné ID (používateľské meno), heslo a časové razítko.

IAX - MD5 heš hesla a tzv. challenge (informačný element).

Medzi dvoma účastníkmi autentizácia priamo neprebieha, ale so serverom pozri obr. 3.1.



Obr. 3.1: Autentifikačný proces vo VoIP

Autorizácia

Jej využitie v zabezpečení spočíva napríklad v limitovaní možností vytáčania čísel alebo obmedzeniu povolení na pripojenie do určitej VoIP siete. V podnikových VoIP sieťach býva zriedka využívaná a je jednoducho prekonateľná. Entity, ktoré môžu byť použité na autorizáciu sú:

E.164 Alias – Medzinárodné číslo zkladajúce sa z kódu krajiny CC (Country Code), národného cieľového kódu NDC (National Destination Code) a čísla účastníka SN (Subscriber Number). Môže nadobúdať 15 znakov a byť nastavené dynamicky gatekeeperom alebo staticky koncovým užívateľom.

MAC adresa – Machine Access Control adresy sú na každom ethernetovskom zariadení na 2.vrstve OSI modelu.

URI – Identifikátor používaný SIP protokolom na rozpoznanie agenta. IAX ho môže tiež použiť.

Dostupnosť

VoIP siete musia byť dostupné neustále, pretože ľudia si už zvykli, že telefonická dostupnosť je stopercentná, narozdiel od iných dátových služieb. V súčasnej dobe je nevyhnutné, aby poskytovateľ zabezpečil spoľahlivé telefonické pripojenie. K tomu je potrebné správne využívať QoS (napríklad uprednostniť hlasové pakety pred dátovými) a oddelovať dátové siete od hlasových (typicky na inej VLAN) a predchádzať tak problémom, kedy pri výpadku prepínača alebo smerovača dojde k nedostupnosti telefonických prípojok.

Šifrovanie

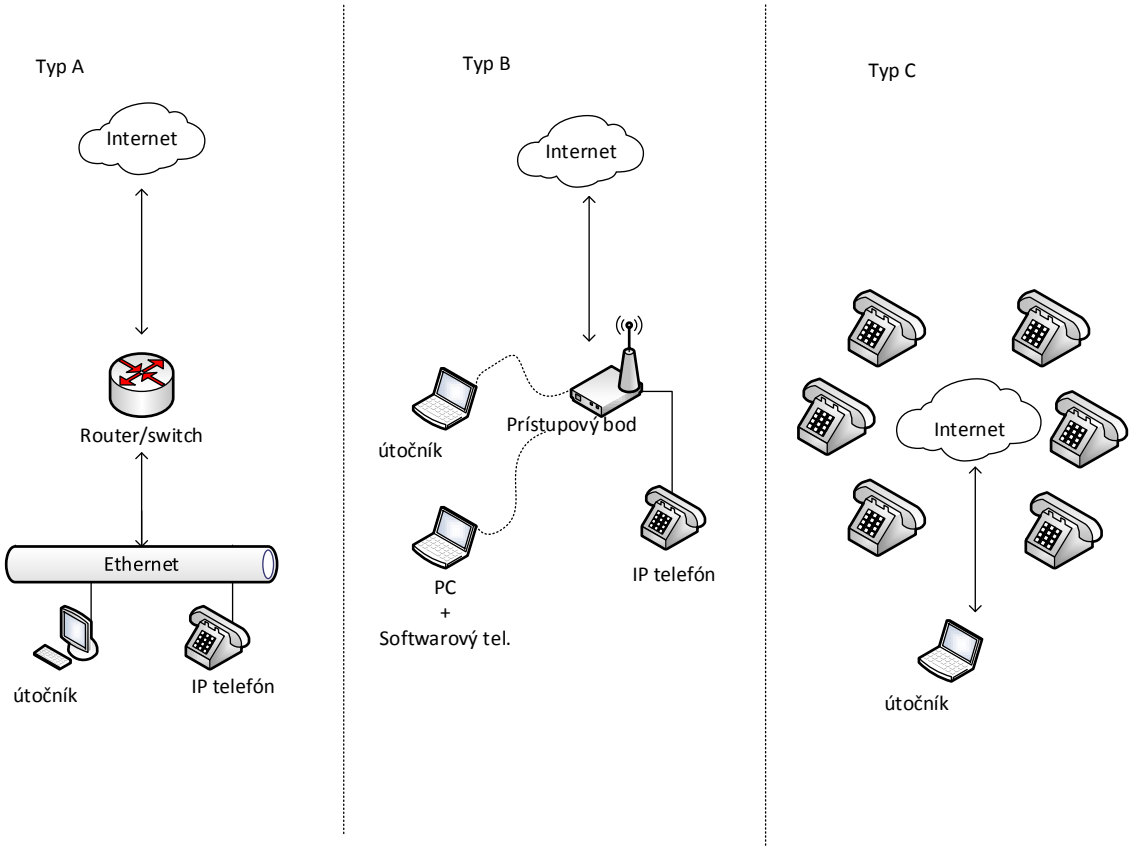
K utajeniu komunikácie u VoIP technológie dochádza na viacerých miestach vrátane signalizačných a transportných protokolov. Nakoľko autentizácia sa objavuje na úrovni signalizácie a pakety nesúce hlas sú prenášané na úrovni prenosu multimédii transportnými protokolmi, je často nevyhnutné šifrovať dáta v obidvoch úrovniach. To je dosiahnuteľné pomocou IPSec, SRTP a SSL protokolmi, ktoré boli podrobne popísané v kapitole 2.5. [23]

3.2 Teória útokov

Všetky súčasné informačné technológie majú bezpečnostné problémy , VoIP nie je výnimkou. Pre nezainteresovaných je často mätúce správne vnímať zložitosť v realizovaní útoku. Pravdou ostáva, že s dostatočnou motiváciou, vrátane zbohatnutia alebo pomsty, môže byť napadnutá ľubovoľná bezpečnostná zložka. Útočné vektory (rozumieme nimi spôsoby,

ktorými útočník môže získať prístup do siete a realizovať v nej útoky) vo VoIP sú podobné tým z oblasti sietí. Napríklad k realizovaniu útoku, nie je potrebné mať fyzický prístup k telefónu alebo k serveru. V závislosti od prostredia, v ktorom sú VoIP služby realizované závisia aj možnosti uskutočnenia útokov:

A) V lokálnej káblovej sieti pripojením sa a/alebo zdielaním ethernetového pripojenia pre telefón.
B) V lokálnej sieti, ktorá používa bezdrôtové pripojenie pre rôznych užívateľov ako sú siete v kaviarňach, obchodoch alebo hoteloch sa môže útočník jednoducho bezdrôtovo napojiť do siete a zachytávať VoIP dáta resp. hovory.
C) Verejná sieť (Internet) alebo inak nedôveryhodná, kde sa používa VoIP komunikácia je pre útočníka najľahším cieľom. [23]



Obr. 3.2: Útočné vektory vo VoIP [23]

Všeobecné úrovne, ktoré môžu byť napadnuté vo VoIP infraštruktúre, by sa dali rozdeliť na:

- **IP infraštruktúru –** slabo zabezpečené miesta a prvky v VoIP sieti.
- **Operačný systém –** VoIP zariadenia (ústredne, softwaroví klienti) dedia rovnaké bezpečnostné nedostatky ako OS alebo firmware, na ktorom sú spustené.
- **Konfigurácia –** V predvolenóm nastavení väčšinou majú zariadenia prebytok bežiacich služieb. Tie sú spustené na otvorených portoch a môžu byť bezpečnostnými dierami na realizáciu útokov, pretečenie zásobníkov alebo prekonanie autentifikácie.
- **Účel –** Zastaralé alebo nekvalitne vyvinuté technológie môžu byť napadnuté a využité k zneužitiu, napr. legacy DNS. [37]

Typy konkrétnych útokov vo VoIP sieťach s využitím dvoch skúmaných protokolov SIP a IAX2, by sa dali rozdeliť do nasledujúcich dvoch kategórii - s cieľom zneužiť službu alebo ju znefunkčniť, a teda:

- **Útoky na reláciu** – odposluch, odcudzenie hesla, muž v strede - služba funguje.

- **Útoky na funkcionalitu služby** – DoS, záplavové útoky, s násilným ukončením, registráciou alebo odregistrovaním, znemožnenie dostupnosti - služba nefunguje.

Okrem tohto sa vyskytujú techniky ako získavanie citlivých údajov (voice phising), ktorá využíva princíp, kedy užívateľ zavolá na podvrhnuté číslo od útočníka, ktoré sa napríklad tvári dôveryhodne v prišlom emaili. Ďalej spam cez internetovú telefóniu, ktorý často využíva nahrávky obsahujúce telefónne číslo, na ktoré treba zavolať, ak užívateľ chce niečo získať. V neposlednom rade existujú aplikácie ako VoIPBuster, ktoré sú schopné umožniť volania z PC na mobily, či pevné linky bezplatne.

3.3 Útoky na prebiehajúcu reláciu

Pred útokom na reláciu je potrebné zachytiť jej parametre a to sa deje odposluchom (angl. *sniffing* alebo *eavesdropping*). Odposluch vo VoIP sieti v podstate nie je odlišný od odposluchu bežného sieťového provozu. Avšak pripojenie do VoIP siete býva často odlišné a to najmä vo firemnom prostredí. VoIP siete bežne bývajú v iných VLAN ako iné dátové služby prístupné z úžívateľských počítačov. Odlišné VLAN siete majú viacero výhod, napríklad k zaisteniu rôznych stupňov bezpečnosti na každej z nich, iných QoS parametrov alebo k odlíšeniu priorít. Spôsoby akými útočník môže pristúpiť k odposluchu boli popísané útočnými vektormi v predchádzajúcej podkapitole.

Ak sa v praxi útočníkovi podarí kompromitovať telefón má možnosť zachytiť všetku z neho alebo na neho uskutočnenú komunikáciu. Veľa VoIP telefónov obsahuje rozšírené možnosti nastavenia cez webové rozhranie. Niektoré z nich napríklad umožňujú zachytávať všetku komunikáciu každému, kto má prístup k administrátorskému rozhraniu.

Ak sa útočníkovi podarí získať administrátorský prístup k prepínaču cez webové rozhranie alebo telnet môže využiť službu RSPAN (Remote Switched Port Analyzer), ktorú dnes veľa prepínačov obsahuje v nastaveniach a kopírovať tak všetku komunikáciu zo zvolených portov na ich monitoring na špecifickej VLAN.

Pri útokoch na softwarovú ústredňu alebo klienta závisí ich ochrana od iných podporných vrstiev systému, na ktorom bežia. Typicky je ním Windows alebo Linux. Dnes už však existuje dostatok exploitačných nástrojov na prenikntuie do slabo zabezpečených strojov. Realizácia odposluchu pomocou sieťového analyzátora je podrobne rozanalyzovaná v kapitole 4.6. V SIP komunikácii je potrebné poznať heslo pre registrovanie klienta k ústredni, po zachytení nezašifrovaného zahešovaného hesla sieťovým analyzátorom je ľahké dané heslo určiť. K tomu sa dajú použiť na internete bežne dostupné utility a skripty ako napríklad SipCrack, ktorý hrubou silou slovníkovým útokom láme heslo, ktoré je v hešovanom tvare. U IAX, kde je prenášané zahešované heslo s unikátnym tzv. *challenge*, to je napríklad IAX.Brute. Problémom však je, že slovníkový útok potrebuje pre najrýchlejšie určenie hesla silný výkon a trvá určitý, v závislosti od zložistosti hesla nemalý čas.

Ďalším typom útoku je často používaná technika muža v strede (Man-in-the-Middle). Je dosiahnutá vďaka tzv. ARP Poisoningu alebo DNS spoofingu. Funguje na princípe, že niektoré systémy nahradia alebo akceptujú zápis do ich ARP tabuľky resp. DNS cache aj napriek tomu, že predtým neposlali ARP alebo DNS žiadosť. Útočník tak oklame úžívateľské stroje, ktoré si myslia, že útočníkova MAC adresa je adresou druhého PC, zatiaľ čo on potajme preposiela komunikáciu na cieľový stroj [39]. Vo VoIP u IAX protokolu teda útočník zachytí autentifikačnú žiadosť od klienta, zabráni aby pokračovala k serveru a namiesto toho odošle klientovi vlastný challenge. Následne realizuje s klientovou odpoveďou autentifikáciu u ústredne. U SIP je proces podobný. Akonáhle útočník od UA zachytí žiadosť smerom k ústredni, nedovolí jej prejsť. Sám jej odošle žiadosť. Od ústredne obdrží tzv. *nonce* údaj, ktorý pošle UA. Ten mu odpovie s hešom hesla a nonce hodnoty, ktorý útočník použije na autentifikáciu u ústredne. Na internete je znova dostupných viacero nástrojov na realizáciu tohto typu útoku. Známe sú napr. Cain&Abel alebo Ettercap. [8]

Ďalším typom je replay útok, spočíva v zachytení a znovu odoslaní správy. Teda ak útočník zkopíruje správu alebo sled správ medzi dvoma užívateľmi a cieľu útoku tým spôsobí komplikácie napr. s redundatnými správami. Alebo útočník môže útok využiť aj k odregistrácii od ústredne (nemusí poznať heš hesla, keďže správu len kopíruje) vydávajúc sa za jedného z užívateľov, ktorého ústredňa pozná. [41]

To vedie k tzv. útokom odstránenia registrácie (registration removal) alebo odcudzeniu hovoru (call/registration hijack). Odregistrovanie klienta od ústredne vedie k neschopnosti prijímať hovory. Klienti majú väčšinou implicitne nastavené časové intervaly v akých sa môžu najskôr pokúsiť znova registrovať. Ak sa teda klient registruje v ústredni a následne je útočníkom odregistorvaný, nie je schopný do uplynutia intervalu sa znova registrovať. Samotné odregistrovanie využíva v hlavičke pole *Expires* nastavené na nulovú hodnotu. Druhým útokom sa dosiahne zmena legitímnej registrácie za falošnú, spôsobujúc tak, že prichádzajúce hovory sú smerované na neexistujúci alebo iný cieľ. Dosahuje sa to v hlavičke pomocou zmeny poľa *Contact* za útočníkovu. K obidvom typom týchto útokov existuje viacero dostupných nástrojov, medzi najznámejšie patrí SiVuS.

Nakoniec by sa do tejto podkapitoly dali zaradiť útoky typu fuzzing. Sú založené na procese odosielania náhodných dát v rámci komunikačného protokolu s cieľom spôsobiť chybu ústredne. Inokedy je ich cieľom skúšanie vnútorných procesov zodpovedných za vysporiadavanie sa s chybami. Jedná sa teda aj o metódu na hľadanie bugov a zranitelností využívajúc neštandardné časti v protokoloch. Najčastejšie sa takýto útok realizuje pomocou skriptu alebo aplikácie, ktorá podľa definovaných scenárov generuje rôzne hodnoty pre rôzne typy správ a častí hlavičiek protokolu. Sem teda môžme zaradiť aj útoky, ktorým sa primárne venuje výskumná, praktická časť práce a teda útoky na ústredne pomocou modifikovaných hlavičiek správ. V súčasnosti je v tejto oblasti výnimočný fínsky univerzitný projekt PROTOS, ktorého cieľom je práve takto založené testovanie protokolov. [39] Z pohľadu cieľa práce je zase účelom testovanie schopností a reakcií samotných ústrední.

3.4 Útoky na poskytovanie služby

Útokom na poskytnutie služby sa najčastejšie rozumie zamedzenie jej primárnemu účelu – funknčnosti. To sa realizuje typicky pomocou tzv. DoS útokov, útokov zamietnutia služby. U VoIP tento typ útoku môže konkrétne spočívať v generovaní a následnom posielaní veľkého množstva žiadostí, aby tak došlo na strane servera (ústredne) k zahlteniu a následnemu znemožneniu funkcionality. DoS útok je možné považovať za slabší, ak je ústredňa schopná naďalej fungovať, aj keď dochádza k nadmernému zaťaženiu procesora a pamäte. To sa začne prejavovať nárastom meškania paketov alebo pri silnejšom útoku ich zahadzovaním a v horšom prípade zamrznutím alebo násilným ukončením ústredne systémom, na ktorom je spustená.

Realizácia tohto typu útoku je v protokole SIP realizovaná najčastejšie cez zasielanie správ náročnejších na oblužné prostriedky ako REGISTER, INVITE, OPTIONS a v protokole IAX cez správy NEW, ACK alebo PING.

Rozšírenou variantou tohto typu útoku je DDoS (distribuovaný). Ten ako to vyplýva z názvu spočíva v distribuovanom, resp. koordinovanom útoku mnohých koncových staníc. Tie sú typicky súčasťou tzv. botnetu, ktorý je tvorený množstvom nezabezpečených počítačov nainfikovaných malwarom. Ten je typicky proces bežiaci na pozadí spusteného systému a na príkaz prevádzkovateľa botnetu využíva systémové zdroje počitača k realizovaniu útokov. Úžívateľská koncová stanica je označovaná príznačným názvom bot a je teda súčasťou „otrockej“ siete (botnet). [23]

Ďalším typom v tejto oblasti je útok ukončenia relácie cez správy BYE u SIP alebo HangUP u IAX. Užívateľ pri ukončení hovoru odošle BYE žiadosť ústredni. Ak však útočník predtým z relácie zachytí tzv.Call-ID, môže odoslať fiktívnu žiadosť o ukončenie hovoru a znemožniť tak funkcionalitu. Pre reláciu v IAX musí zachytiť Source Call ID (SCID),

Destination Call ID (DCID), Inbound Sequence Number (iseq), a Outbound Sequence Number (oseq). [23]

Dôležité je ešte dodať, že útoky na funkcionalitu zamerané na softwarové ústredne, ktoré komunikujú cez TCP/IP sadu, nie je niekedy potrebné zamerať na aplikačnú úroveň, ale je to možné dosiahnuť aj útokom na ICMP (útoky *smurf*, *ping of death* a *ping flood*), UDP (*DNS amplification*) alebo TCP (*SYN flood*) protokoly. Práca neni zameraná na tieto typy a preto sú len stručne uvedené.

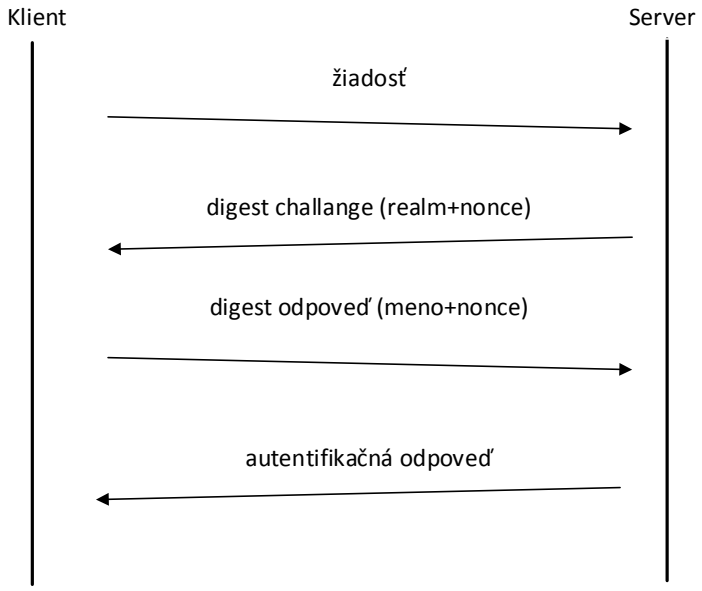
Zariadenie Spirent Testcenter obsahuje balík útokov pozostávajúcich z modifikovaných základných typov správ SIP protokolu. Z toho vyplýva primárne zamerenia praktickej časti na odtestovanie schopností ústrední voči takýmto neštandardným správam. Tester takisto obsahuje možnosti realizácie DoS útokov, ktorému bude venovaná sekundárna pozornosť, aj pretože testovanie výkonovej schopností voči tomuto typu útoku už bolo náplňou iných prác.

3.5 Metódy zaistenia ochrany PBX

Okrem bezpečnostných protokolov spomínaných v kapitole 2.4 existujú aj ďalšie štandardizované mechanizmy pre zaistenie autenticity a integrity dát. Tie budú popísané v nasledujúcich podkapitolách. Následne budú v ďalších vysvetlené možnosti ochrán proti útokom ako sú firewally a detekčné systémy.

3.5.1 HTTP Digest autentizácia

SIP poskytuje autentifikačný mechanizmus založený na autentifikácii z HTTP. Tento mechanizmus sa používa v prostrediach, kde sa vyžaduje od klientov, aby sa pred výmenou dát (zahájením komunikácie) najprv autentifikovali serveru. Hovorí sa jej Digest a využíva MD5 hešovaciu funkciu na zahešovanie užívateľského mena a hesla. Jej všeobecný princíp je uvedený na obrázku 3.3.



Obr. 3.3: Princíp HTTP Digest autentizácie [18]

U protokolu SIP mechanizmus funguje tak, že ak sa klient snaží naviazať spojenie s UAS, registračným serverom, server pošle *401 Unauthorized* odpoveď a tak vyzve UAC. Ak navezuje spojenie s proxy serverom, ten odpovedá 407 *Proxy Authentication Required*. Existujú dve variácie a to autentifikácia klienta a autentifikácia servera.

Schéma autentifikácie klienta slúži k overeniu identity klienta, aby bola legitímna, keď pristupuje k ústredni. Poskytuje ochranu voči niektorým typom útokov popísaných v predchádzajúcej podkapitole. Jej princíp spočíva v nasledujúcich krokoch. SIP protokol

definuje tzv. *nonce* hodnotu na autentifikáciu klienta k serveru. Jej hodnota sa líši od konkrétnej implementácie, mala by však vychádzať zo zhrnutia (v ang. digest) IP adresy a časového razítka. Rovnaká hodnota môže použitá dovtedy pokiaľ nevyprší časové razítko. Toto znižuje možnosť útoku typu replay, nakoľko útočník musí stihnúť použiť hodnotu *nonce* pred vypršaním časového razítka.

Schéma autentifikácie servera zaručuje serveru, že server má legitímnu identitu a tak sa autentifikuje voči klientovi. Zaistí sa tak ochrana pred odcudzením servera. [18]

Proces tejto metódy autentizácie spočíva v realizácii nasledujúcich krokov:

1. UAC odošle žiadosť REGISTER alebo INVITE k zahájeniu komunikácie
2. UAS odpovie správou 401 alebo 407, ktorá v sebe nesie hodnotu *nonce*
3. UAC následne vytvorí heš z kombinácie užívateľského mena, realm a hesla v tvare: *HA1=MD5(username:realm:password)*, druhý heš z názvu použitej metódy a z adresy v tvare: *HA2=MD5(method:URI)* a z nich dvoch nakoniec odpoveď tj. finálny heš v tvare: *odpoveď=MD5(HA1:nonce:HA2)*
4. UAC odošle vytvorenú odpoveď na UAS
5. UAS realizuje tie isté operácie a výsledné hodnoty porovná. Ak sú zhodné odošle klientovi kladnú odpoveď a ten bude autentizovaný.

Obdobný mechanizmus používa aj IAX, ten bol popísaný v samostnatnej podkapitole 2.3.4.

3.5.2 Secure Multipurpose Internet Mail Exchange (S/MIME)

Šifrovanie celých SIP správ v end-to-end komunikácii s odôvodnením, aby bolo zaistené utajenie nie je správne, pretože smerovacie prvky ako SIP proxy servery potrebujú vidieť určité polia v hlavičke, aby mohli smerovať správy korektne. Ak by sa na toto nemyslelo, správy by nebolo možné smerovať. Preto existuje mechanizmus S/MIME (vychádzajúci z nezabezpečeného MIME), ktorý povoluje UA šifrovať MIME telo v rámci SIP. Zabezpečuje teda tieto telá správ bez toho, aby zasahoval do hlavičiek. Takisto poskytuje vzájomnú autentizáciu a je ho možné využiť aj k zabezpečeniu integrity a utajenia hlavičiek pomocou tunelovania.

SIP správy teda nesú MIME telá a tento štandard obsahuje mechanizmy na zaistenie obsahu, aby tak boli dosiahnuté podporované typy tiel (v poli *Content-Type* v tvare *type/názov*, tj. napríklad *application/sdp*). Šifrovanie tela sa realizuje pomocou verejného kľúča od druhého koncového užívateľa. Ten vlastní aj súkromný kľúč, ktorým prijatú správu rozšifruje. Tento kľúč je jediný použiteľný na rozšifrovanie správy. Pre zaobstaranie kľúčov je nutné, aby bola prítomná certifikačná autorita a zaisťovala vydávanie certifikátov (obsahujúcich verejný a súkromný kľúč) a bola dôveryhodná pre všetkých účastníkov. [23]

Certifikáty, ktoré sa používajú k účelom identifikácie koncového užívateľa sú odlišné od tých, ktoré používajú servery v jednom hlavnom princípe. Spoliehajú sa na to, že ich držiteľ je jednoznačne identifikovaný adresou. Tá je zložená z častí užívateľského mena, symbolu „@“ a názvu domény, najčastejšie odpovedá užívateľovej URI adrese.

Ak UA použije v žiadosti S/MIME telo, ale obdrží odpoveď, ktorá obsahuje MIME telo (tj. nezabezpečené) mal by notifikovať užívateľa, že relácia nemôže byť zabezpečená. Avšak, ak UA, ktorý podporuje S/MIME obdrží žiadosť s nezabezpečeným telom, nemal by odpovedať so zabezpečeným telom. Ak ale očakáva použitie S/MIME od žiadateľa (napríklad, ak v hlavičke v poli *From* adresa odpovedá identite kľúča), mal by takisto upozorniť užívateľa, že relácia nemôže byť zabezpečená. [12]

3.5.3 Firewall

VoIP firewall je zariadenie alebo aplikácia, ktorá podľa bezpečnostných nastavení definuje či budú povolené alebo zamietnuté určité hovory, resp. správy. Aby poskytoval

nadradenú bezpečnosť nad iné mechanizmy, je potrebné zaistiť jeho vysokú spoľahlivosť a minimálne oneskorenie dát. Existuje viacero firewallových techník, niektoré bývajú využívané súčastne.

- **Filter paketov –** prezerá každý paket na sieťovej vrstve, ktorý do neho vchádza alebo vychádza a akceptuje alebo zamieta jeho priechod ďalej podľa nakonfigurovaných pravidiel (zdrojová a cieľová IP adresa, typ protokolu, zdrojový a cieľový port).
- **Aplikačná brána -** bezpečnostná politika sa nastavuje na aplikačnej vrstve, čo zaisťuje vyššiu efektivitu filtrovania, avšak môže viesť k zníženiu výkonu.
- **Transportná brána –** pred ustanovením spojenia prezerá iba TCP a UDP relácie.
- **Proxy server –** efektívne ukrýva pravú sieťovú zdrojovú a cieľovú adresu.
- **Stavová inšpekcia –** sleduje transakcie, aby zaistil, že príchodzie pakety boli vyžiadané užívateľom. Všeobecne môže bežať cez všetky vrstvy.

V súčasnosti sa často firewally tvoria aj ako samotné distribúcie bežiace na vlastnom operačnom systéme. Najvýkonejšie sú však vyrábané aj ako samostatné hardwarové moduly s vlastnými konfiguračnými rozhraniami. V nasledújucej tabuľke je stručný prehľad najznámejších súčastných firewallových riešení.

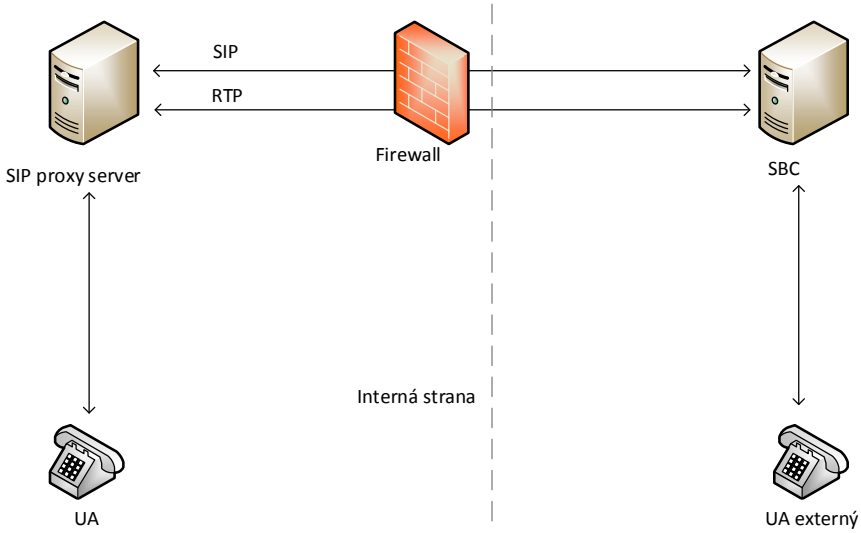
Tab. 3.1: Prehľad súčasných firewallových riešení

Názov	OS	Typ
Monowall	Unix	software
Netfilter/iptables	Unix	software
Turtle	Unix	software
Firestarter	Unix	software
UWF	Unix	software
IPCop	proprietárny	software
pfSense	proprietárny	software
OpenBSD	proprietárny	software
Cisco ASA/PIX	proprietárny	hardware
CheckPoint	proprietárny	hardware
Securepoint	proprietárny	hardware
Juniper SSG/SRX	proprietárny	hardware
Intego VirusBarrier	Mac	software
Antivírusové FW	Windows	software

Veľa unixových firewallových riešení v podstate vychádza z *iptables*, ktorý ma jednu z najväčších komunít a je úspešne tvorený od roku 1998. Na filtráciu rozdielnych protokolov pomocou tzv. reťazových pravidiel využíva odlišné kernelovské moduly linuxového jadra. [5]

VoIP siete používajú rôzne transportné porty a veľa z nich nie je statických. Jedná sa najmä o tie, ktoré používa RTP protokol (pozri 2.4.1). To limituje schopnosť firewallu určiť presné porty, ktoré majú byť otvorené. Ďalším problémovým aspektom býva použitie prekladu adries (NAT). Koncové body, ktoré chcú kontaktovať externé entity môžu mať problém, pretože RTP porty používajú UDP s reálnymi zdrojovými a cieľovými hodnotami v hlavičke. Riešení je viacero, od použitia statických portov pre RTP po použitie SBC (Session Border Controller) a brán. Väčšina výrobcov dnes podporuje statické porty a tak je v praxi RTP tok dát limitovaný na jeden, dva porty, drasticky znižujúc potrebu otvorených portov na firewally. Ďalšia možnosť, využitie SBC, čo sú zariadenia spravujúce signalizáciu a prenos hlasových dát medzi koncovými bodmi s využitím NAT. Typicky bývajú umiestnené mimo firewallu v demilitarizovanej zóne (DMZ) alebo v externej sieti, tak aby mohli vytvárať, komunikovať

a ukončovať spojenia namiesto koncových entít. V internej sieti za firewallom komunikujú s proxy serverom (u SIP). Na firewalli je povolené len spojenie medzi týmito dvoma prvkami. Vo vnútri sa teda všetci užívatelia kontaktujú s interným proxy a ten komunikuje s SBC, ktorý vonku vystupuje ako užívateľ z vnútra. Príklad takejto architektúry je na obrázku. [23]



Obr. 3.4: SBC v SIP architektúre

3.5.4 IDPS

Skratka predstavuje pojem z ang. Intrusion Detection/Prevention Systems a teda systémy s detekciou/prevenciou vniknutia. Ako z názvu vyplýva jedná sa o systémy alebo zariadenia, ktorých cieľom je monitoring sieťových aktivít s ohľadom na podozrivú činnosť. Ich hlavnými funkciami sú identifikácia podozrivých činností, zaznamenanie informácii (logovanie) o tom, pokus o zamedzenie resp. zakázanie takejto aktivity a jej nahlásenie. Detekčné systémy majú schopnosť len detekovať, pričom prevenčné aj aktivite zabrániť a do budúcna jej predísť. Vo všeobecnosti existujú nasledujúce tri druhy detekčných metód:

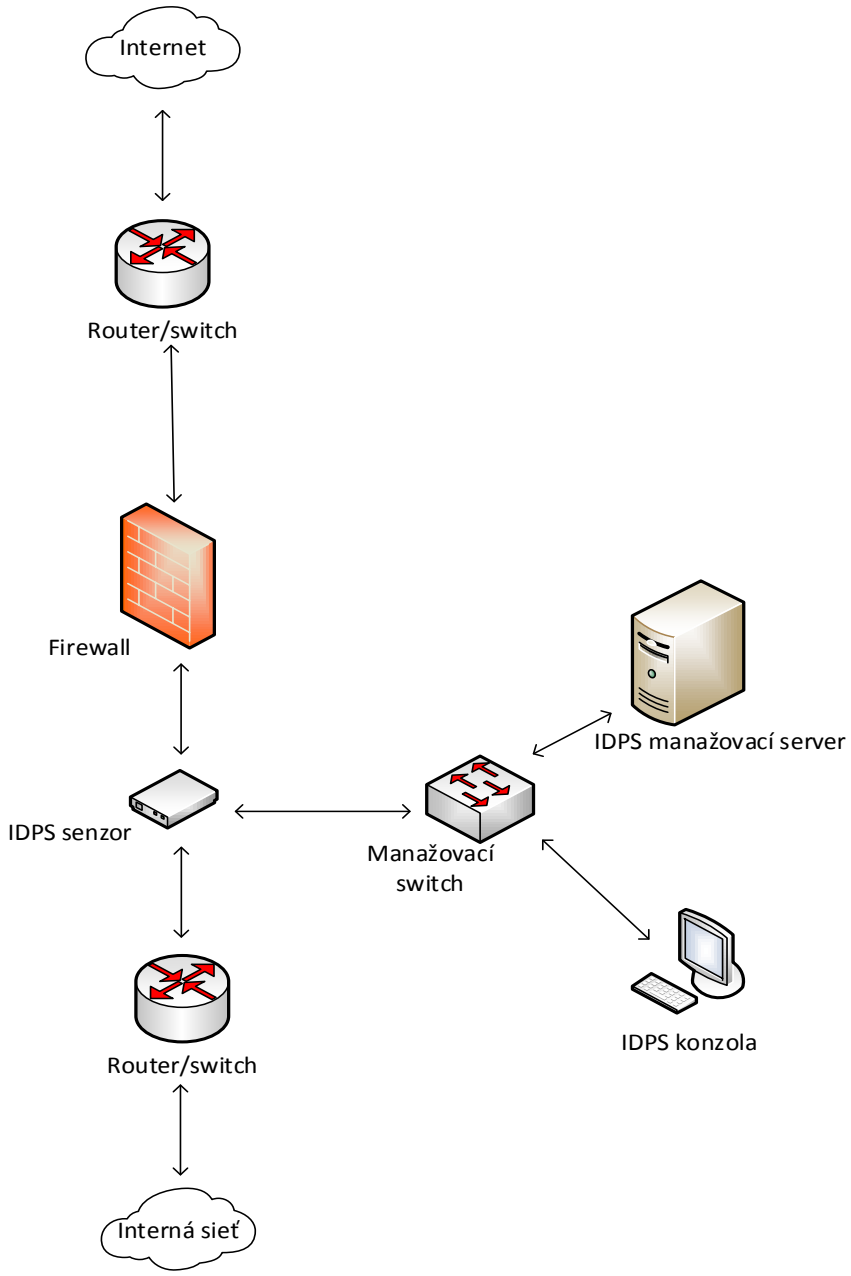
- Vzorová – založená na porovnávaní príchodzích paketov podľa prednastavených útočných vzorov – podpisov (ang. signatures)
- Anomálie – založená na poznaní normálnej sieťovej aktivity ako: aká šírka pásma je pri bežnej premávke používaná, aké sú bežne komunikujúce protokoly, aké porty i zariadenia sú prepojené a alarmuje administrátora pri detekovaní anomálii v týchto stavoch.
- Stavová – je založená na identifikácii deviácii protokolových stavov, porovnávaním sledovaných javov s predvolenými profilmi všeobecne akceptovateľných definícii miernej aktivity. [1]

Prevenčné systémy dokážu pracovať v rôznych konfiguráciach, typicky však podľa svojich pravidiel spolupracujú s firewallom. Dajú sa rozdeliť do štyroch kategórii:

- Sieťové (Network IPS) – monitorujú podozrivé spojenia analýzou protokolových aktivít v celej sieti.
- Bezdrôtové (Wireless IPS) - sledujú iba bezdrôtovú komunikáciu monitoringom bezdrôtových protokolov.
- Analyzátory správania (Network behavior analysis NBA) – sledujú sieťový prenos a identifukujú hrozby, ktoré generujú nezvyčajné toky dát ako sú napr. DDoS útoky, určité formy malwaru a porušenia politík.
- Užívateľské (Host-based IPS) – majú typicky formu nainštalovaného softwaru, ktorý monitoruje iba daný užívateľský stroj. Okrem monitoringu operačného

systému dokážu kontrolovať integritu dát, logy, registre a detekujú tzv. *rootkity* (podporné sady nástrojov na maskovanie vírusov).

Príkladom takýchto systémov sú open-source riešenia fail2ban, Snort (sieťový) alebo OSSEC (užívateľský), ktoré sú použité aj ako navrhnuté opatrenia pred skúmanými útokmi v praktickej časti práce. Architektúra sytému v rámci siete je na nasledujúcom obrázku.



Obr. 3.5: Sieťový IDPS v architektúre siete

Typickými komponentami IDPS systémov sú senzory alebo agenti, manažovací server, databázový server a konzola. Senzory monitorujú siete a agenti monitorujú a analyzujú stanice. Manažovací server od nich dostáva informácie a riadi ich. Databázový server je úložiskom pre tieto informácie. Konzola je program, ktorý poskytuje vstupno-výstupné rozhranie pre administrátora systému. Tieto komponenty môžu byť prepojené štandardnou firemnou sieťou alebo sú v nezávislej bezpečnostnej obslužnej sieti. [1]

4 Inštalácia a konfigurácia ústrední

Ústredne boli nainštalované na systém Ubuntu v LTS verzii 12.04. Nasledujúce podkapitoly popisujú realizované a potrebné kroky k správnemu nainštalovaniu a konfigurovaniu ústrední. Príkazy boli zadávané do unix terminálu v prihlásení pod *sudo* účtom. Časť z nich bola prebratá z oficiálnych stránok tvorcov ústrední a z [38]. Konfigurácie ústrední obsahujú dva účty, medzi ktorými boli realizované hovory a následne implementované útoky a nastavenie smerovania medzi nimi v smerovacích plánoch. Zabezpečenie bolo realizované na overenie jeho funkčnosti, ale útoky boli realizované na nezabezpečené ústredne.

4.1 Asterisk

Bola nainštalovaná posledná aktuálna LTS verzia 11.6 certified. V nasledujúcich riadkoch je postup inštalácie a konfigurácie ústredne pre realizovanie testov.

V prvom kroku skontrolujeme a nainštalujeme aktualizácie systému:

```
apt-get update
apt-get upgrade
```

Inštalácia balíčkov nutných pre kompiláciu.

```
apt-get install -y build-essential linux-headers-‘uname -r‘
libxml2-dev ncurses-dev libsqlite3-dev sqlite3
```

Inštalácia balíčkov potrebných pre SIP SSL, SRTP, podpory speex kodeku a formátu ogg vorbis:

```
apt-get install libssl-dev
apt-get install libsrtp1-dev srtp-utils srtp-docs
apt-get install libvorbis-dev
apt-get install libspeex-dev libspeexdsp-dev
```

Vytvorenie a presun do aktuálne vytvoreného pracovného adresára:

```
mkdir /usr/src/asterisk
cd /usr/src/asterisk
```

Stiahnutie zdrojových kódov, extrakcia tar archívu a následné presunutie rozbalenej zložky so zdrojovými kódmi.

```
wget http://downloads.asterisk.org/pub/telephony/certified-
asterisk/
releases/certified-asterisk-11.6-cert2.tar.gz
tar -xvzf certified-asterisk-11.6-cert2.tar.gz
cd certified-asterisk-11.6-cert2
```

Spustenie konfiguračného skriptu s výberom inštalačnej cesty pomocou prefixu.

```
./configure –prefix=/opt/asterisk-11.6
```

Kontrola modulu res_srtp v konfiguračnom menu:

```
make menuselect
```

Komplilácia a inštalácia:

```
make
make install
make samples
```

Ústredňa je nainštalovaná, spustí sa cez:

```
asterisk –rvvvv
```

kde počet parametrov „v“ označuje početnosť výpisov. Automaticky sa spustí aj riadiaca konzola. Vypne sa cez príkaz: `stop`.

Nasleduje základná konfigurácia účtov a smerovacieho plánu. Tie sú v Asterisku v súboroch *sip.conf* pre účty protokolu SIP, *iax.conf* pre účty protokolu IAX2 a *extensions.conf* pre smerovací plán hovorov. Konfiguračné súbory sú súčasťou prílohy.

Zabezpečenie ústredne pomocou TLS a SRTP sa realizuje cez skript, ktorý je súčasťou zdrojového adresáru a ktorý generuje certifikáty a cez povolenie šifovania v konfiguračných súboroch.

Vytvorenie adresára pre kľúče/certifikáty.

```
mkdir opt/asterisk-1.8/etc/asterisk/keys
```

Spustíme skript ast_tls_cert skript, ktorý sa nachádza v zdrojovom adresári Asterisku a preklikáme sprievodcu na heslá.

```
./ast_tls_cert -C pbx.test.com -O "PBX Test" -d opt/asterisk-
1.8/etc/asterisk/keys
```

Následne vytvoríme certifikát pre klienta, kde za parametrom -o nasleduje jeho názov:

```
./ast_tls_cert -m client -c /opt/asterisk-
1.8/etc/asterisk/keys/ca.crt –k /opt/asterisk-
1.8/etc/asterisk/keys/ca.key -C 1000.test.com –O "PBX Test" -d
/opt/asterisk-1.8/etc/asterisk/keys -o 1000
```

Vytvorený certifikát je nutné implementovať do klienta.

Povolenie šifrovania sa realizuje pridaním aktivačných príkazov do súboru *sip.conf*:

```
[general]
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlscafile=/etc/asterisk/keys/ca.crt
tlscipher=ALL
tlsclientmethod=tlsv1
tlsdontverifyserver=no
[1000] ;treba pre každý účet
transport=tls ; nastavenie typu prenosu TLS
encryption=yes ; povolenie SRTP pre daný účet
```

4.2 Freeswitch

Bola nainštalovaná posledná aktuálna LTS verzia 1.5. Nasleduje postup inštalácie a konfigurácie ústredne pre realizáciu testov.

Inštalácia základných balíčkov:

```
sudo apt-get install git-core build-essential autoconf automake
libtool libncurses5 libncurses5-dev make libjpeg-dev pkg-config
unixodbc unixodbc-dev zlib1g-dev
```

Následne inštalácia voliteľných balíčkov pre podporu SIP SSL a TLS, ZRTP a podobne:

```
sudo apt-get install libcurl4-openssl-dev libexpat1-dev libssl-
dev libtiff4-dev libx11-dev unixodbc-dev libssl-dev python2.6-
dev zlib1g-dev libzrtpcpp-dev libasound2-dev libogg-dev
```

```
libvorbis-dev libperl-dev libgdbm-dev libdb-dev python-dev uuid-
dev bison
```

Zmena pracovného adresára, kde budú uložené zdrojové kódy:

```
cd /usr/local/src
```

Stiahnutie zdrojových súborov:

```
git clone git://git.freeswitch.org/freeswitch.git
```

Zmena aktuálneho adresára na adresár so zdrojovými súbormi:

```
cd freeswitch
```

Vytvorenie konfiguračných súborov pred kompiláciou:

```
./bootstrap.sh
```

Spustenie konfiguračného skriptu:

```
./configure
```

Kompilácia a inštalácia:

```
make

make install
```

Vytvorenie symbolických odkazov pre zjednodušené spúšťanie:

```
ln -s /usr/local/freeswitch/bin/freeswitch
/usr/local/bin/freeswitch

ln -s /usr/local/freeswitch/bin/fs_cli /usr/local/bin/fs_cli
```

Ústredňa sa zapne a inicializuje skriptom:

```
/usr/local/freeswitch/bin/freeswitch
```

do riadiacej konzole sa prihlási cez:

```
/usr/local/freeswitch/bin/fs_cli
```

opustenie konzole cez: `/exit`
zastavit ústredňu v termináli cez: `shutdown`.

Po inštalácii nasleduje základná konfigurácia účtov a smerovacieho plánu. Tie sú vo Freeswitchi v xml súboroch v zložke */usr/local/freeswitch/conf/directory/default/*, účty maju každý vlastný xml súbor. Predvolená konfigurácia obsahuje 20 účtov od 1000 do 1019. Predvolený smerovací plán hovorov je v súbore *default.xml*. Pre testovanie boli použité predvolené konfiguračné súbory.

Zabezpečenie pomocou TLS a SRTP sa dosiahne cez vygenerovanie certifikátov pomocou skriptu umiesteného v */freeswitch/bin/gentls_cert* a následnou implementáciou certifikátu klienta do softwarového klienta. Okrem toho je potrebné upraviť konfiguráciu súboru vars.xml zmenou

```
<X-PRE-PROCESS cmd="set" data="internal_ssl_enable=true"/>
```

A v súbore conf/directory/default.xml, zmeniť:

```
<param name="dial-string"
value="{sip_secure_media=${regex(${sofia_contact(${dialed_user}@
${dialed_domain})}|transport=tls)},presence_id=${dialed_user}@${
dialed_domain}}${sofia_contact(${dialed_user}@${dialed_domain})}
" />
```

Upravené konfiguračné súbory pre zabezpečenú komunikáciu sú súčasťou prílohy.

4.3 Yate

Bola nainštalovaná v aktuálnej stabilnej verzii 5.0. Nasleduje postup inštalácie a konfigurácie ústredne pre realizáciu testov.

Vytvorenie a presun do aktuálne vytvoreného pracovného adresára:

```
mkdir /usr/src/yate
cd /usr/src/yate
```

Stiahnutie zdrojových kódov:

```
wget http://voip.null.ro/tarballs/yate5/yate-5.0.0.tar.gz
```

Rozbalenie tar archívu a presun do zložky so zdrojovými súbormi.

```
tar -xvzf yate-5.0.0.tar.gz
cd yate-5.0.0
```

Spustenie konfiguračného skriptu so zmenou inštalačného adresára:

```
./configure
```

Kompilácia a inštalácia zo zdrojových kódov:

```
make
make install
```

Nainštalovanú ústredňu pustíme cez:

```
yate –vvvvDof ,
```

kde počet parametrov „v“ učuje podobne ako u Asterisku početnosť vypisovaných logov. Do riadiacej konzole sa dostaneme pomocou telnetu, cez príkaz:

```
telnet 127.0.0.1 5038
```

Konzolu opustíme cez: `quit`
a zastavíme cez: `ctrl+c` .

Po inštalácii nasleduje základná konfigurácia. Konfiguračné súbory sú v zložke /usr/local/etc/yate. Tie v Yate tvoria súbory *ysipchan.conf* pre parametre účtov a *regfile.conf* pre účty a ich smerovanie. Nastavené konfiguračné súbory sú súčasťou prílohy.

Zabezpečenie pomocou TLS a SRTP sa dosiahne cez vygenerovanie certifikátov (yate neobsahuje autogenerovací skript, ale je možné použiť už vytvorené certifikáty), následnou implementáciou certifikátu klienta do softwarového klienta a úpravou konfigurácie súborov *ysipchan.conf* doplnením:

```
[general]
type=tls
addr=0.0.0.0
port=5061
sslcontext=server_context
[default]
secure=enable
```

A v súbore *openssl.conf*, nastaviť:

```
[server_context]
enable=yes
certificate=name.crt ;meno certifikátu a kľúča
key=name.key
```

4.4 Kamailio

Táto ústredňa bola nainštalovaná v aktuálnej stabilnej verzii 4.2.0, spolu s OpenSIPS vyžaduje komplikovanejšiu inštaláciu kvôli nastaveniu databázového systému na ukladanie

informácii. U obidvoch bola použitá MySQL. Nasleduje postup inštalácie a konfigurácie pre testovanie.

Inštalácia základných balíčkov:

```
apt-get install mysql-server git gcc flex bison libmysqlclient-dev make libssl-dev libcurl4-openssl-dev libxml2-dev libpcre3-dev
```

Vytvorenie zložky pre inštaláciu a zmena adresára:

```
mkdir –p /usr/local/src/kamailio
cd /usr/local/src/kamailio
```

Stiahnutie inštalačných súborov:

```
git clone git://git.sip-router.org/kamailio kamailio
cd kamailio
git checkout –b 4.2 origin/4.2
```

Generácia konfiguračných súborov:

```
make cfg
```

Editácia súboru modules.lst a povolenie modulu TLS, ktorý sa inak nekompiluje :

```
vi modules.lst
include_modules = db_mysql tls
```

Kompilácia a inštalácia programu:

```
make all
make install
```

Nasleduje vytvorenie MySQL databázy.
Pred vytvorením databázy je potreba v súbore kamctlrc

```
vi /usr/local/etc/kamailio/kamctlrc
```

povoliť (odkomentovaním):

```
DBENGINE=MYSQL
DBHOST=localhost
DBNAME=kamailio
DBRWUSER="kamailio"
DBRWPW="kamailiorw"
DBROUSER="kamailioro"
DBROPW="kamailioro"
DBACCESSHOST=127.0.0.1
DBROOTUSER="root"
```

Vytvorenie databáze cez:

```
/usr/local/sbin/kamdbctl create
```

Konfigurácia nainštalovaného programu v súbore kamailio.cfg, kde povolíme prácu s MySQL databázou:

```
vim /usr/local/etc/kamailio/kamailio.cfg
```

Na začiatok pridáme riadky:

```
#!define WITH_MYSQL
#!define WITH_AUTH
```

```
#!define WITH_USRLOCDB
```

Prekopírovanie inicializačného súboru, a zmena práv

```
cp usr/local/src/kamailio/kamailio/pkg/kamailio/deb/debian/
kamailio.init /etc/init.d/kamailio
chmod 755 /etc/init.d/kamailio
```

Editujeme skript:

```
vi /etc/init.d/kamailio
```

Upravíme riadky odkazujúce na cestu k spusteniu skriptu a konfiguračnému súboru:

```
DAEMON=/usr/local/sbin/kamailio
CFGFILE=/usr/local/etc/kamailio/kamailio.cfg
```

Prekopírovanie súboru kamailio.default:

```
cp      /usr/local/src/kamailio/kamailio/pkg/kamailio/deb/debian/
kamailio.default /etc/default/kamailio
```

Povolíme spoustenia programu zmenou riadku cez vi editor:

```
RUN_KAMAILIO=yes
```

Vytvorenie zložky pre ukládanie dočasných súborov:

```
mkdir –p /var/run/kamailio
```

Vytvorenie užívateľa a skupiny aplikácie cez terminál. Predvolený je názov kamailio.

```
adduser --quiet --system --group --disabled-password \
--shell /bin/false --gecos "Kamailio" \
--home /var/run/kamailio kamailio
```

Zmena vlastníka súboru na:

```
chown kamailio:kamailio /var/run/kamailio
```

Inštalácia a základné nastavenie je hotové, ústredňa sa pustí cez:

```
 kamailio start
```

a do riadiacej konzole sa dostaneme cez príkaz: `kamcmd .`

Nasleduje základná konfigurácia účtov. Tá sa realizuje jednoducho vďaka implementovanej databázi pomocou riadiacich príkazov, na pridanie účtu postačí:

```
kamtcl add <meno-účtu> <heslo>
```

Pre výpis účtov stačí príkaz `kamctl db ul show.`

Konfiguračné súbory sú zložke `/etc/kamailio/etc/kamailio/.`

Zabezpečenie pomocou TLS a SRTP sa dosiahne cez vygenerovanie certifikátov (kamailio neobsahuje autogenerovací skript, ale je možné použiť už vytvorené certifikáty), následnou implementáciou certifikátu klienta do softwarového klienta a úpravou konfigurácie súboru v *kamailio.cfg* doplnením:

```
#!define WITH_TLS
listen=tls:127.0.0.1:5061
```

a v súbore *tls.cfg* úpravou:

```
[server:default]
method = TLSv1
verify_certificate = yes
require_certificate = no
private_key = /etc/certs/pbx/key.pem ; cesta ku klúču
```

```
certificate = /etc/certs/pbx/cert.pem ; a  certifikátu
ca_list = /etc/certs/CA/cert.pem
 [client:default]
verify_certificate = yes
require_certificate = yes
```

4.5 OpenSIPS

Nainštalovaná bola posledná aktuálna stabilná verzia 1.11. Nasleduje postup inštalácie a konfigurácie pre testovanie.

Inštalácia nutných balíkov:

```
apt-get install subversion flex bison libncurses-dev
libmysqlclient-dev libsctp-dev m4 mysql-server
```

Stiahnutie instalačných súborov:

```
svn co https://svn.code.sf.net/p/opensips/svn/branches/1.11
opensips_1_11
```

Prepnutie do adresáru a spustenie konfiguračného súboru:

```
cd opensips_1_11/
make menuconfig
```

Povolenie ukladania do MySQL databázy:

```
Configure Compile Options → Configure Exclude Modules → db_mysql
```

Zadáme inštalačnú cestu:

```
Configure Compile Options → Configure Install Prefix →
/usr/local/opensips.
```

a nainštalujeme:

```
Compile And Install OpenSIPS
```

Zeditujeme súbor opensipsctlrc:

```
cd /usr/local/opensips/etc/opensips
vi opensipsctlrc
```

Povolenie databáze MySQL odkomenotovaním riadkov:

```
DBENGINE=MYSQL
DBHOST=localhost
DBNAME=opensips
DBRWUSER=opensips
DBRWPW="opensipsrw"
DBROOTUSER="root"
```

Vytvorenie databáze MySQL

```
cd /usr/local/opensips/sbin/
./opensipsdbctl create
```

Vytvorenie konfiguračného súboru:

```
./osipsconfig
```

Vygenerujeme konfikuračný súbor:

```
Generate OpenSIPS Script → Residential Script → Generate
Residential Script
```

Editácia vytvoreného konfiguračného súboru:

```
cd /usr/local/opensips/etc/opensips

vi opensips_residential_[čas].cfg
```

Upravíme cestu k modulom programu:

```
mpath="/usr/local/opensips/lib/opensips/modules/"
```

A v sekcii URI module doplníme:

```
modparam("uri","db_url",
"mysql://opensips:opensipsrw@localhost/opensips")
```

Kopírovanie, úprava práv a editácia inicializačného súboru:

```
cd /usr/local/src/opensips_1_11/packaging/debian

cp opensips.init /etc/init.d/opensips

chmod 755 /etc/init.d/opensips

vi /etc/init.d/opensips
```

Zmena cesty k deamonu:

```
DAEMON=/usr/local/opensips/sbin/opensips
```

Zadáme cestu ke konfiguračnímu súboru:

```
OPTIONS="-P $PIDFILE -m $S_MEMORY -M $P_MEMORY -u $USER -g
$GROUP -f
/usr/local/opensips/etc/opensips/opensips_residential_[čas].cfg"
```

Vymazanie možnosti debugovania:

```
if [ "$1" != "debug" ]; then

check_fork

fi
```

Kopírovanie, úprava práv a editácia default súboru:

```
cp opensips.default /etc/default/opensips

cd /etc/default/

vi opensips
```

Zmeníme položky:

```
RUN_OPENSIPS=yes

USER=root

GROUP=root
```

Spustenie ústredne cez:

```
 /etc/init.d/opensips start
```

Prístup do konzole (spustenie a potom pripojenie):

```
/usr/local/opensips/sbin/opensipsctl start

/usr/local/opensips/sbin/osipsconsole
```

Nasleduje základná konfigurácia účtov. Tá sa realizuje podobne ako u Kamailio vďaka implementovanej databázi pomocou riadiacich príkazov, na pridanie účtu postačí:

```
opensipsctl add <meno-účtu> <heslo>
```

Konfiguračné súbory sú v zložke /usr/local/opensips/etc/opensips/.

Zabezpečenie sa dosiahne cez implementáciou certifikátu klienta do softwarového klienta (Opensips automaticky po inštalácii vytvorí certifikáty), a úpravou konfigurácie súboru *opensips.cfg* doplnením na začiatok:

```
listen = tls:127.0.0.1:5061

tls_verify_server = 0

tls_verify_client = 0

tls_require_client_certificate = 0

tls_handshake_timeout=30

tls_send_timeout=30

tls_method= TLSv1

tls_ciphers_list="NULL"

tls_certificate=
"/usr/local/opensips/etc/opensips//tls/user/usercert.pem"

tls_private_key=
"/usr/local/opensips/etc/opensips//tls/user/userprivkey.pem"

tls_ca_list=
"/usr/local/opensips/etc/opensips//tls/user/usercalist.pem"

tls_server_domain [127.0.0.1:5061]

{

tls_method = SSLv23

tls_certificate="/usr/local/etc/opensips//tls/user/user-
cert.pem"

tls_private_key="/usr/local/etc/opensips//tls/user/userprivkey.p
em"

tls_ca_list="/usr/local/etc/opensips/tls//user/user-calist.pem"

}
```

4.6 Test funkcionality – odposluch relácie

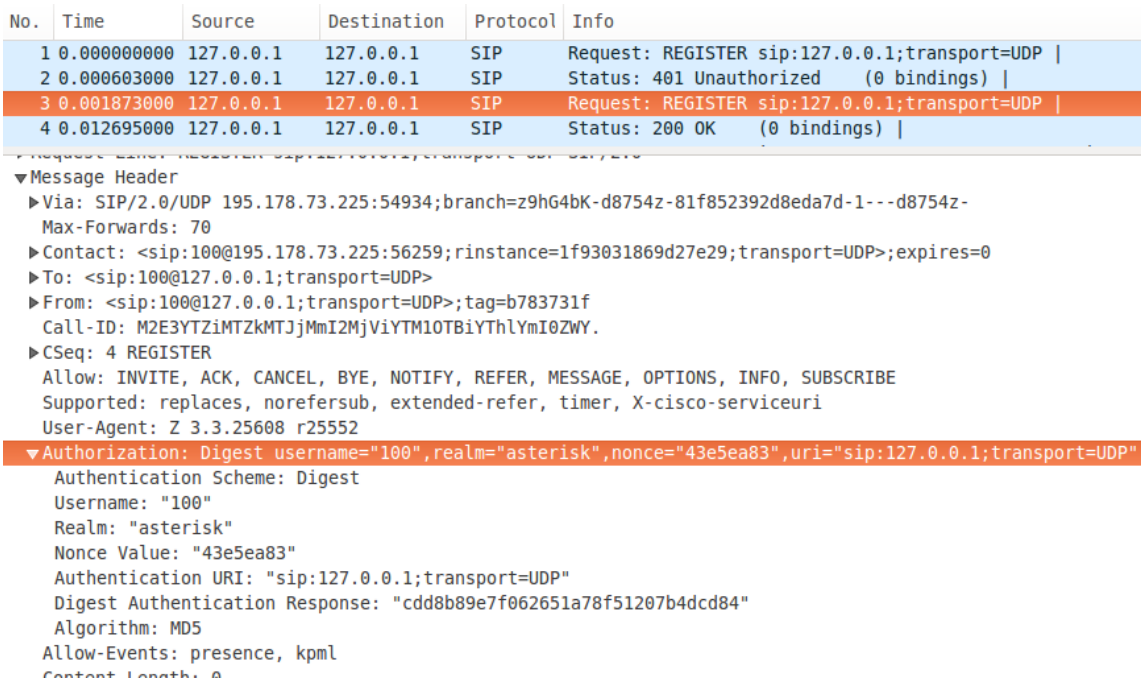
Po nainštalovaní a konfigurácii ústrední do virtuálneho Ubuntu systému boli nainštalovaný dvaja koncoví klienti vo forme softwarového telefónu (PhonerLite a Blink pre komunikáciu využívajúc SIP protokol), ktorí obidvaja podporujú zabezpečenie signalizácie TLS aj hovorových dát pomocou SRTP. Po nastavení klientských účtov a naimplementovaní vygenerovaných certifikátov boli odtestované možnosti zabezpečenia dát.

Pre overenie funkcionality ústrední s IAX2 protkolom bol využitý softwarový telefón Zoiper. IAX2 však natívne podporujú iba ústredne Asterisk a Yate. Freeswitch podporoval staršiu verziu protokolu v beta verzii modulu *mod_opal* (primárne pre H.323 protokol), v novej podporu vypustil. Kamailio a OpenSIPS IAX2 nepodporujú vôbec.

V základnom nastavení ani jedna z ústrední nepodporuje šifrovanie riadiacich alebo multimediálnych dát pomocou bezpečnostných protokolov. Na zachytenie a odposluch dát v takýchto paketoch postačí ľubovoľný sieťový analyzátor, napr. Wireshark.

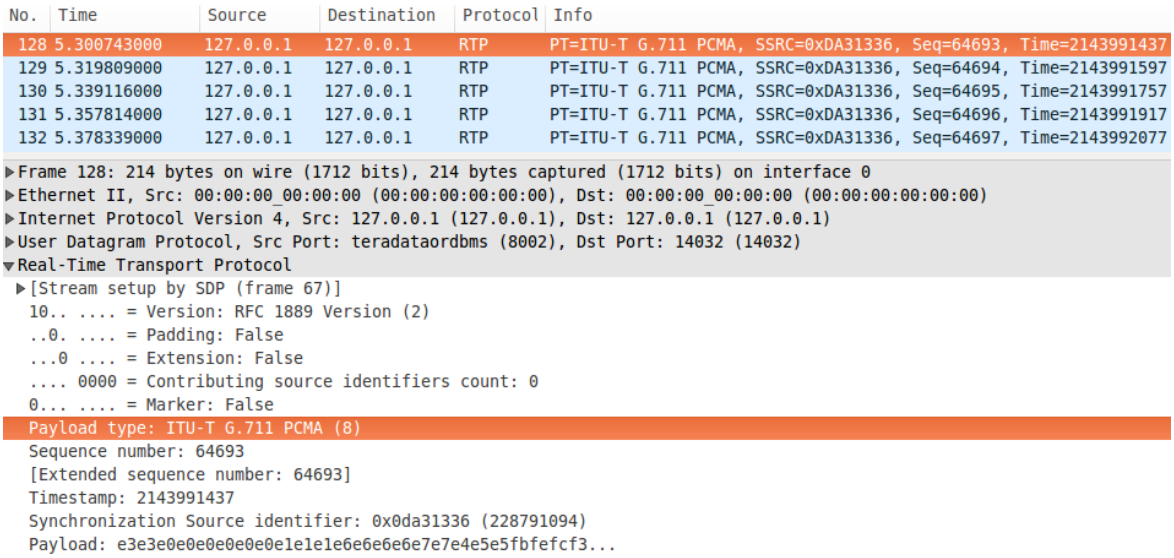
Signalizácia je štandardne prenášaná nešifrovane cez SIP, pri registrácii účtu klienta k ústredni dochádza k výmene správ *Register*, kde je najprv posielaná klientom bez autentizácie a po výzve ústredne správou *Unauthorized* je poslaná s poľom *Authentication*, viď obr. 4.1. Prenos citlivých dát ako názov účtu, adresa ústredne, názov použitého VoIP klienta, náhodné číslo (*Nonce Value*), použitá hešovacia funkcia (*MD5*) a samotný heš hesla s náhodným číslom (*Digest Authentication Response*) sú prenášané nešifrovane. Pre útočníka je teda po zachytení týchto údajov možné vstúpiť do komunikácie.

Správou *200 OK* ústredňa potvrdí úspešnú registráciu klienta po porovnaní hešovaných hodnôt.

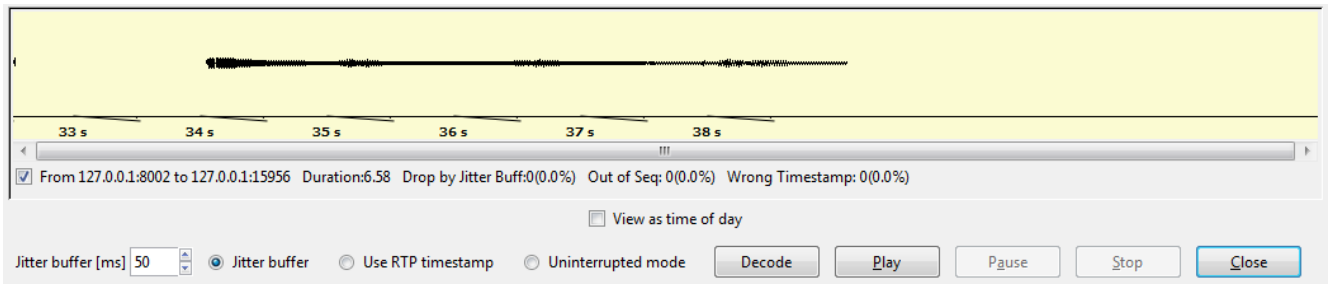


Obr. 4.1: Zachytená registrácia SIP klienta k ústredni

Pre prenos hlasových dát je štandardne používaný nešifrovaný RTP, časť zachytených paketov je na obrázku. Je možné vidieť použitý zvukový kodek, sekvenčné číslo a časové razítko. Samotné multimediálne dáta sú zobrazené v záťaži v hexa tvare a s pomocou RTP prehrávača, ktorý má Wireshark implicitne vstavaný je možné ich dekódovať a prehrať viď obrázok 4.3.

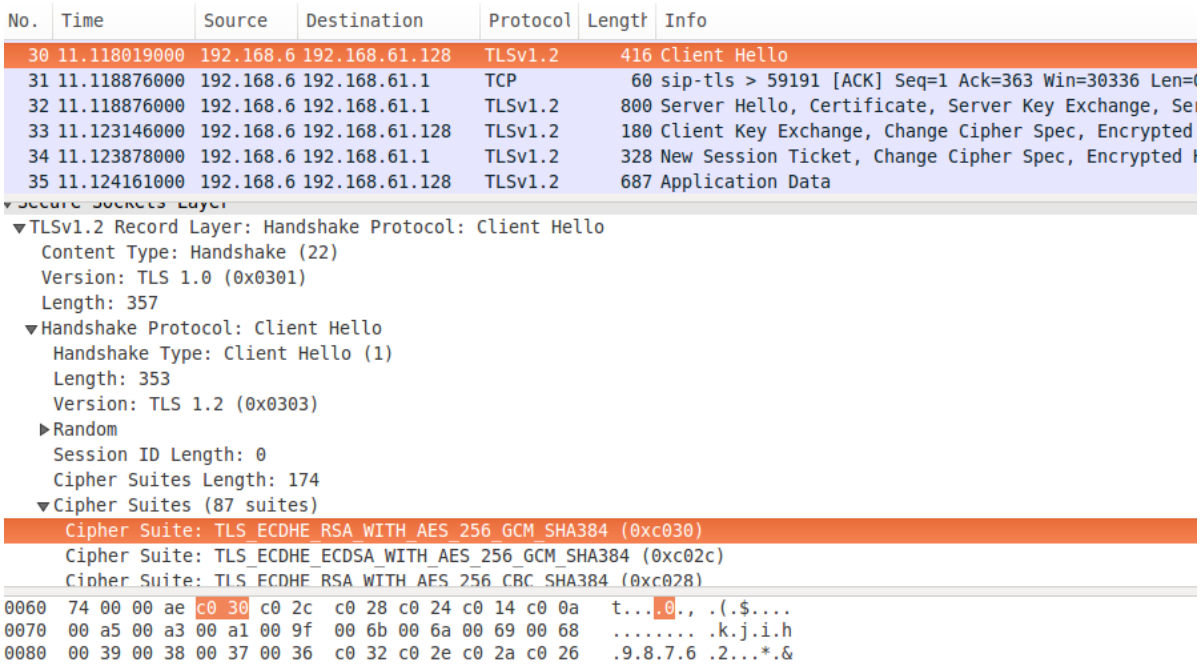


Obr. 4.2: Zachytené RTP pakety



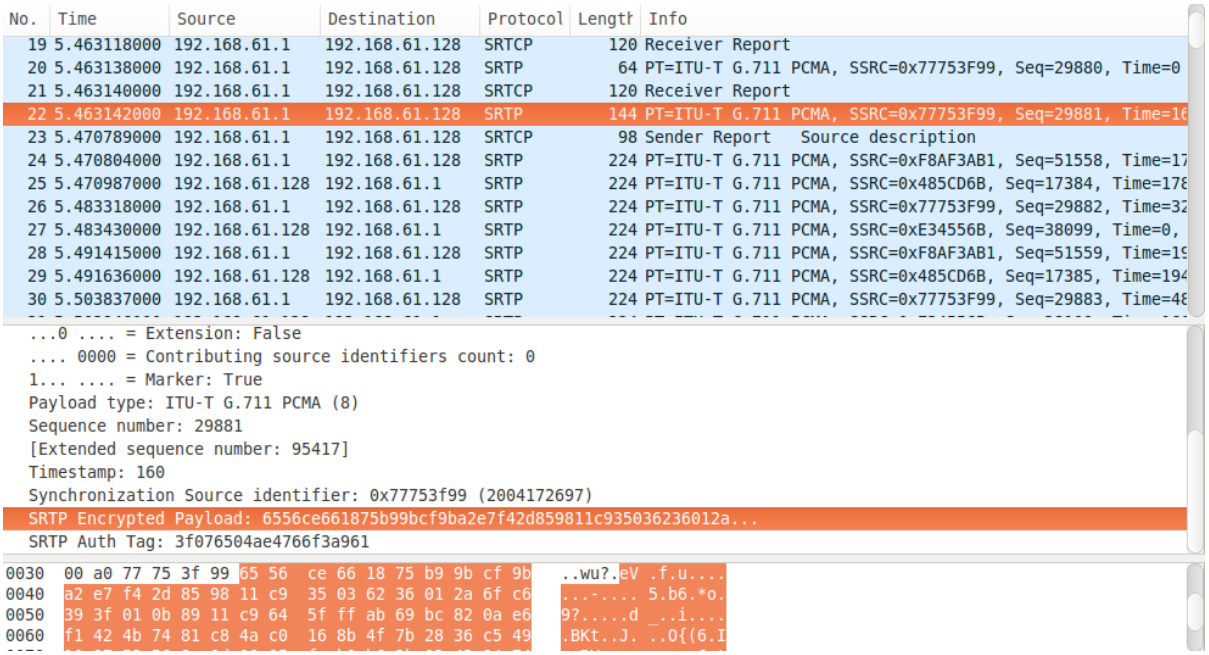
Obr. 4.3: Rekonštrukcia hovoru vo Wiresharku

Prevenciou proti odposluchu relácie a zároveň odcudzeniu údajov je použitie bezpečnostných protokolov. To je možné pri naimplementovaní certifikátov do ústredne a volajúcich klientov. Popis generácie a implementácie certifikátov pre jednotlivé ústredne bol spomenutý v podkapitolách inštalácie ústrední. Samotná zachytená komunikácia s nasadením TLS protokolom začína tzv. *handshake-om*, ktorý bol už bližšie vysvetlený v samostatnej kapitole 2.5.2. Štruktúru jeho inicializačnej správy *Client Hello* je vidieť na obrázku detailnejšie aj s použitou šifrovacou sadou RSA s AES. Celý sled správ, vrátane dohody účastníkov na podporovaných šifrovacích algoritmoch, výmene šifrovacích kľúčov, autentizácie na základe certifikátov a následnom šifrovaní je zachytený takisto na obrázku nižšie.



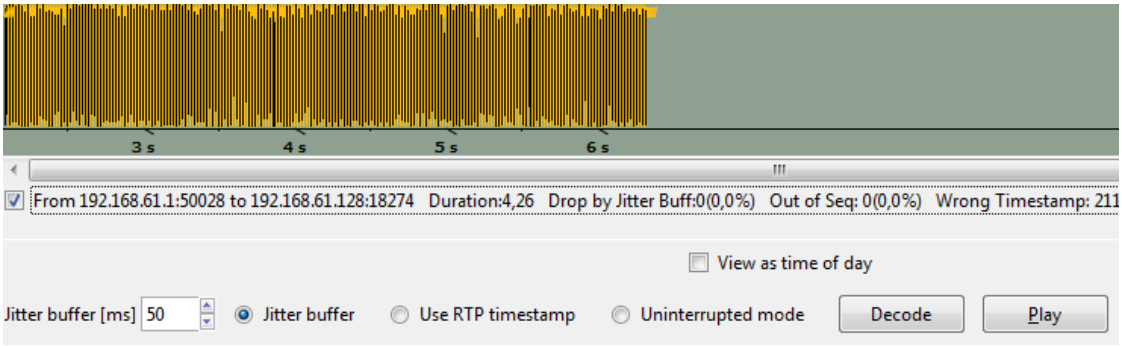
Obr. 4.4: Zachytený TLS handshake.

Pri nastavení podpory SRTP dochádza k šifrovaniu záťaže (payload) tj. zvukových dát a pripojeniu zašifrovaného kontrolného súčtu v poli Auth Tag (pozri obrázok 4.5).



Obr. 4.5: Zachytené SRTP pakety so zašifrovanou záťažou

Rekonštrukcia hovoru pomocou vstavaného prehrávača už nie je možné rozpoznať hlas. Celý zvuk znie ako silne zašumený a tým pádom je plne nezrozumiteľný (porov. obr. 4.3 a 4.6).



Obr. 4.6: Rekonštrukcia zašifrovaného hovoru - silný šum

Pri spoločnej implementácii TLS a SRTP Wireshark SRTP pakety zobrazuje len ako UDP datagramy (SRTP tvorí telo) [33], ktoré je možné analyzovať a prehrať aj zašifrovaný zvuk. [40] Pakety SRTP ako sú zachytené na obrázku 4.6 zobrazuje len pri implementácii bez TLS šifrovania signalizácie.

Zachytené toky dát (hovor., tls. a srtp.pcapng) a rekonštruované hovory (hovor_raw a srtp_raw) sú súčasťou prílohy. Týmto testom sa overila správna funkcionalita jednotlivých ústrední a demonštrovalo sa ako jednoduché je odpočúvať nezabezpečený hovor.

5 Spirent TestCenter C1

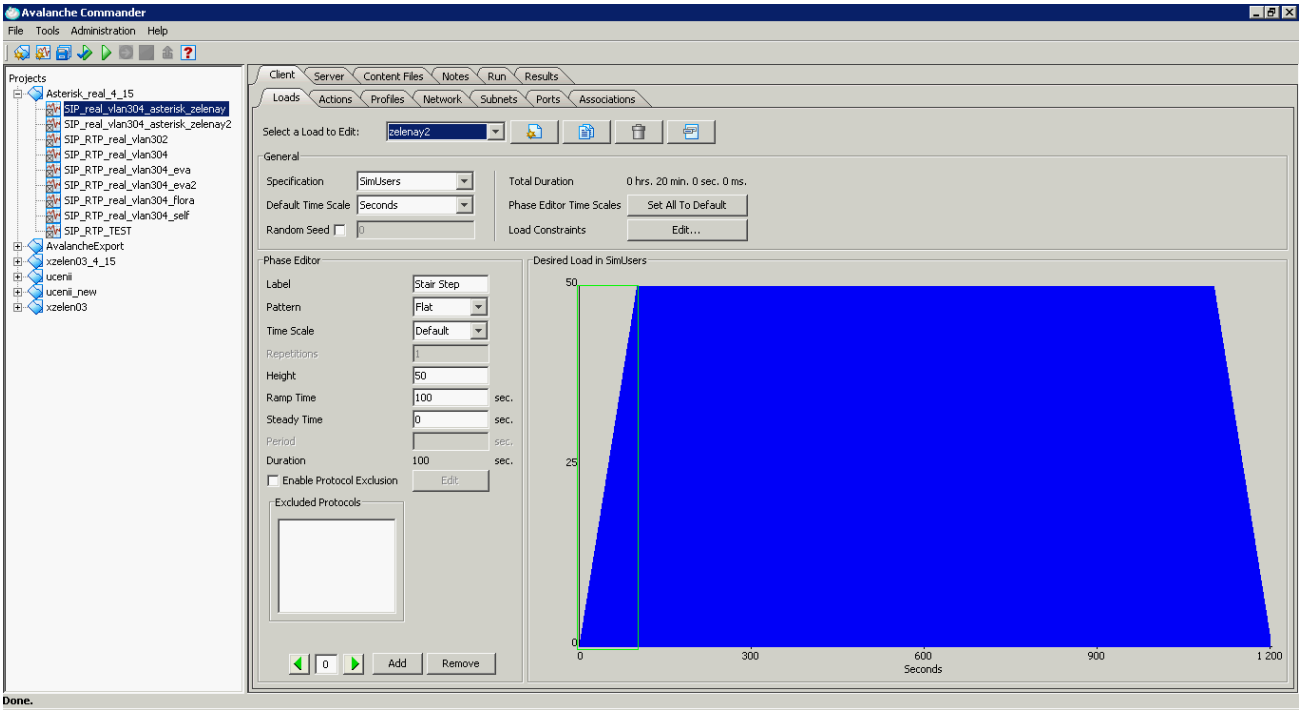
Umožňuje komplexné testovanie sieťovej infraštruktúry na vrstvách L2 až L7 vrátane generovania reálneho prenosu a simulácie chovania tisícov užívateľov a takisto aj aplikačných serverov. Simuluje teda stranu klienta (klientov) ale aj servera. V našom prípade obidve strany tvoria VoIP klienty, ktorý realizujú volania medzi sebou. Pre každú stranu sú v grafickom rozhraní programu samostatné konfiguračné karty. Tento program na obsluhu je Avalanche

Commander a je pravidelne aktualizovaný firmou Spirent. Testy boli realizované s verziou 4.33, ktorá je posledná aktuálna pre tento model Spirent generátoru. Treba však podotknúť, že sa jedná o verziu z roku 2013 a nakoľko sa tento model nahradil novším, ďalšia softwarová podpora pre tento model už nie je odvtedy k dispozícii. Okrem Avalanche Commanderu je k zariadeniu priložená sada podporných aplikácii ako Attack Designer, Results Reporter a iné. Prvý spomínaný bol využívaný na tvorbu a úpravu útokov.

TestCenter C1 obsahuje štyri porty, ktoré je možné použiť k simulácii i generovaniu prenosu a ktoré sú ovládané pomocou grafického užívateľského prostredia. Dokáže generovať hlasové (VoIP), video (streaming, VoD) a dátové záťaže. Podporuje protokoly IPv4 a IPv6, HTTP, HTTPS, FTP, SIP, RTSP/RTP, RTMP, SMTP/POP3/MAP4, DNS, Telnet, PPPoX, 802.1Q VLAN tagging, Q-in-Q tagging, NAC, IPSEC, RADIUS a ďalšie. K tomu, aby bolo možné konfigurovať test pre jeden z protokolov, ktoré nie sú v základnej licencii, je potrebné vlastniť na daný protokol licenciu, ktorá musí byť dodatočne dokúpená k zariadeniu.

Súčasťou sú aj detailné štatistiky, či už grafické alebo tabuľkové a takisto aj kompatibilita zobrazenia generovaných dát s programom Wireshark.

Testovacia záťaž je nastaviteľná podľa požiadaviek, vrátane jej priebehu a názorne zobrazená v grafe viz obrázok 5.1, kde sa v ľavej časti nachádza menu s vytvorenými profilmi a v centrálnej časti hore jednotlivé karty na konfiguráciu strany klienta, servera, beh testu, výsledky a poznámky.



Obr. 5.1: Rozloženie záťaže v karte klienta

Počas simulácie v čase vykresľuje množstvo priebehov a závislostí podľa voľby užívateľa (napr. priebehy počtu spojení, prenosu RTP dát alebo počet relácii za čas).

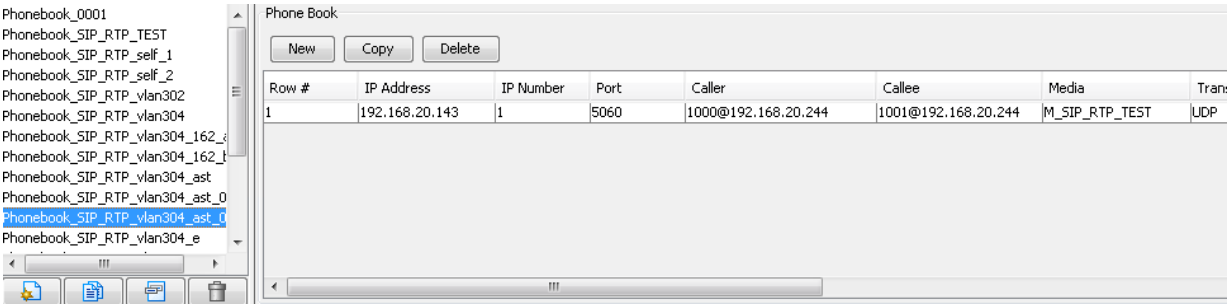
Obr. 5.2: Spirent TestCenter C1

Jeho obsluha a nastavovanie, sa po fyzickom prepojení portov a oboznámení sa s obsluhou, bola realizovaná cez obslužný PC na lokálnej školskej sieti a obsluha ústrední cez virtuálny stroj spustený na tomto PC.

5.1 Test funkcionality

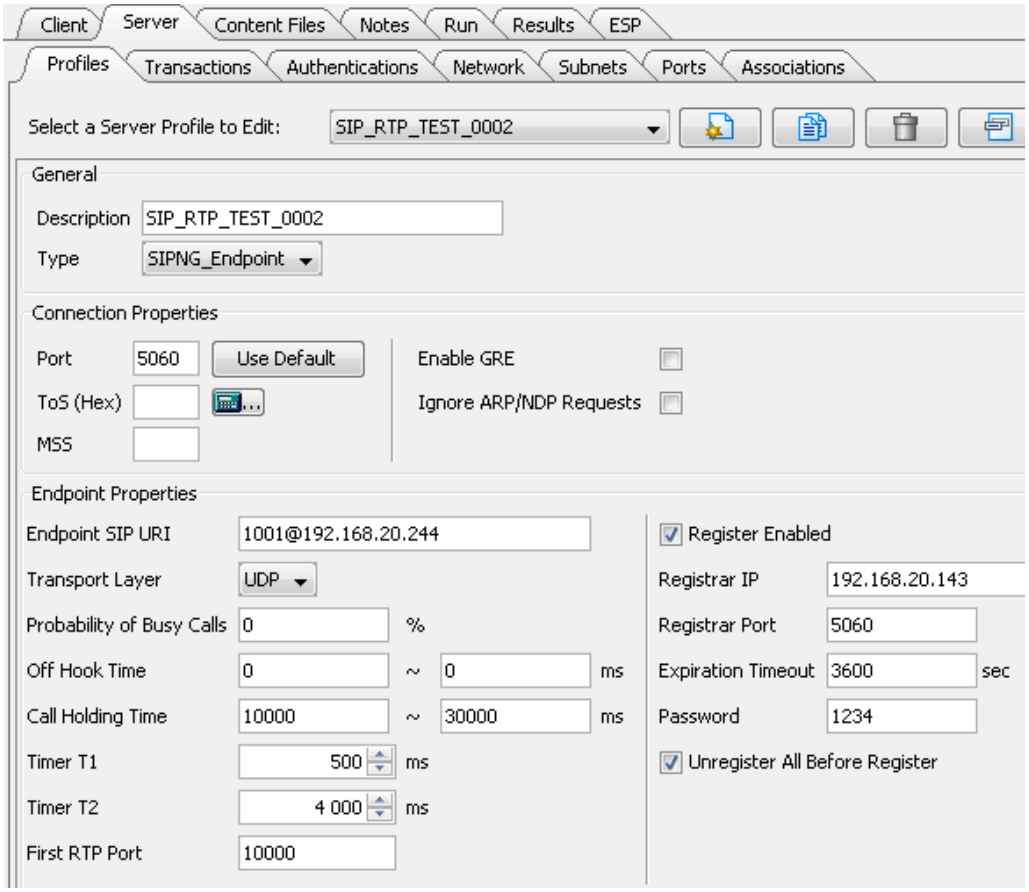
Po prvotnom zoznamení s TestCentrom boli realizované prvé testy zamerané na overenie funkcionality generátora spolu s inštalovanými ústredňami. Ako bolo spomenuté tie boli inštalované a konfigurované na virtuálny systém Ubuntu na stolný PC v laboratórii.

Na TestCentri bol nastavený rovnomerný priebeh záťaže, 10 účastníckych spojení a pracovalo sa s profilmi klienta aj servera (zastupujú dve koncové zariadenia - softwarových klientov medzi ktorými boli generované hovory). Pre obidvoch bol nakonfigurovaný príslušný protokol k VoIP signalizácii – SIP, ktorý má výrobca, tj. firma Spirent v aplikácii v podobe rozšírených nastavení nazvaný SIPng. V profile klienta bol vytvorený tzv. *Phonebook*, kde bolo nastavené číslo a klapky pre volajúceho a volaného a príslušné parametre pre zrealizovanie hovoru (port, protokol). Tu bola nadefinovaná aj dĺžka hovoru a prenášaná zvuková stopa (*Media*) pozri nasledujúci obrázok.



Obr. 5.3: Phonebook v konfigurácii klienta

Pre simuláciu bolo ďalej potrebné nastaviť príslušnú podsieť a asociovať vytvorené profily s portami. Následne bola spustená simulácia, ktorá podľa definovaných profilov generovala hovory medzi dvoma koncovými stranami, účty klientov boli registrované (nastavenie pomocou IP ústredne, ktorá sa v tomto prípade chová ako Registrar server) na ústredni, ktorá hovory riadila. Test bol zopakovaný pre všetky ústredne a potvrdil, že TestCenter dokáže plnohodnotne nahradiť koncové strany a poskytnúť tak ucelené možnosti na testovanie reálneho VoIP charakteru v sieti.



Obr. 5.4: Karta profilu Servera (druhý koncový užívateľ) s povolenou registráciou na PC s ústredňou

Nevýhodou testera je absencia obdobnej podpory protokolu IAX resp. IAX2. Tento protokol je podporovaný iba v rámci možnosti tvorby útokov (pozri ďalšie kapitoly). Tester ho teda k simulácii a generovaniu hovorov nemá v možnostiach profilov zahrnutý.

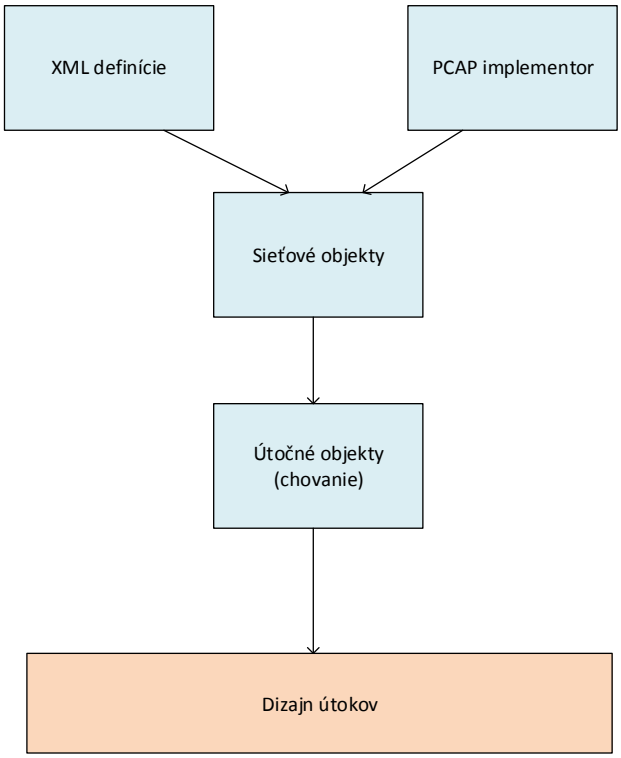
5.2 Spirent Attack Designer

Je aplikácia, ktorá je súčasťou balíka obslužných programov pre generátor a slúži na vytváranie jednoduchých útokov modifikáciou konkétneho sieťového protokolu. Útoky sú definované ako súbory v XML jazyku. Tieto súbory sú do testov nakonfigurované cez voľbu Editora v karte *Attack Lists* v Avalanche Commanderi (pozri nasledujúci obr.).

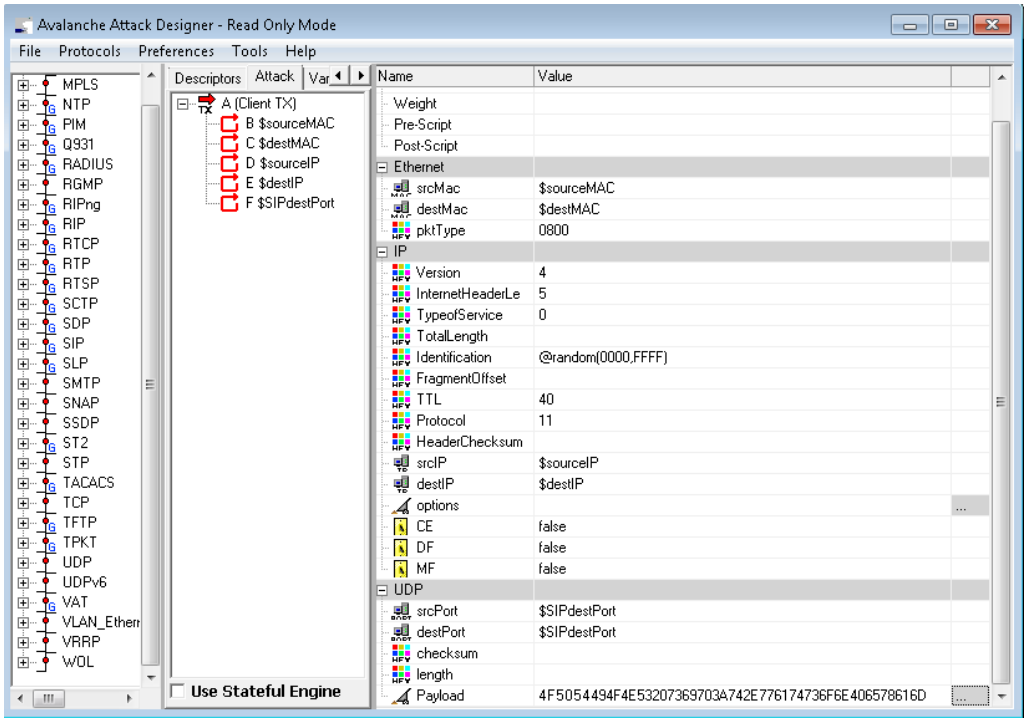
Obr. 5.5: Umiestnenie Editoru na pridanie útoku do testu

Schematický princíp tvorby útoku je nasledovný. Najprv je vybraný špecifický sieťový objekt ako cieľ útoku. Potom je vložený útočný charakter pre konkrétny sieťový protokol. Nakoniec, po nastavení hodnôt premenných, ktoré sú požadované pre konkrétny dizajn útoku sa návrh uloží do XML súboru. Z toho je pri generovaní komunikácie tvorený konkrétny sled paketov, ktoré odsimulujú útok na cieľové zariadenie. Attack Designer dokáže implementovať aj *.pcap* súbory z Wiresharku na importovanie zachytenej komunikácie do dizajnu útoku. Táto

funkcia je však čiastočne zastaralá, nakoľko podporuje iba priamo *.pcap* súbory a nie novšie *.pcapng*. Takisto je aj nanešťastie dosť nespoľahlivá, nakoľko aplikácia často po importe padá.



Obr. 5.6: Funkcia Attack Designeru



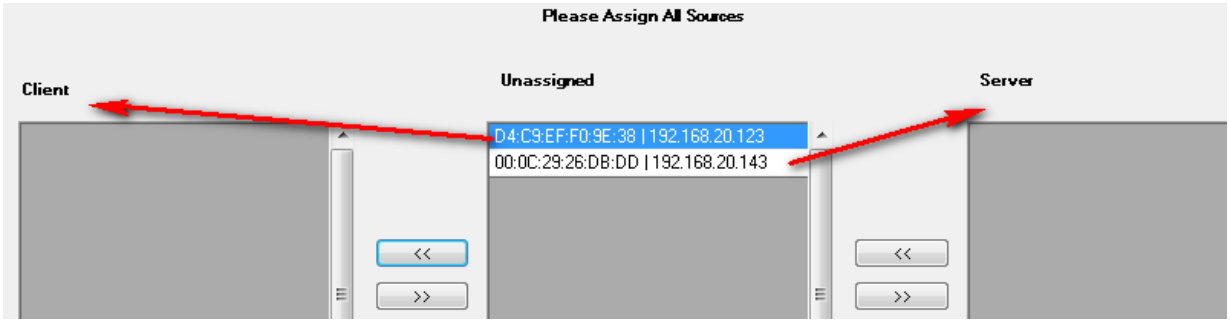
Obr. 5.7: Vzhľad Attack Designeru

5.2.1 Tvorba útokov

Vytvorenie útoku by sa dalo popísať v nasledujúcich krokoch. Po zvolení možnosti *New* v hornom menu *File* si je následne nutné vybrať konkrétny aplikačný protokol, ktorý bude použitý pre jednu z komunikujúcich strán (definované v Avalanche Commanderi ako klient a

server). Na výber je sada takmer všetkých známych protokolov definovaných v nejakom RFC. Následne je možné modifikovať jednotlivé položky v hlavičkách počnúc rámcom a pokračujúc cez paket a datagram resp. segment až k aplikačnému protokolu a neseným dátam v záťaži (*Payload*).

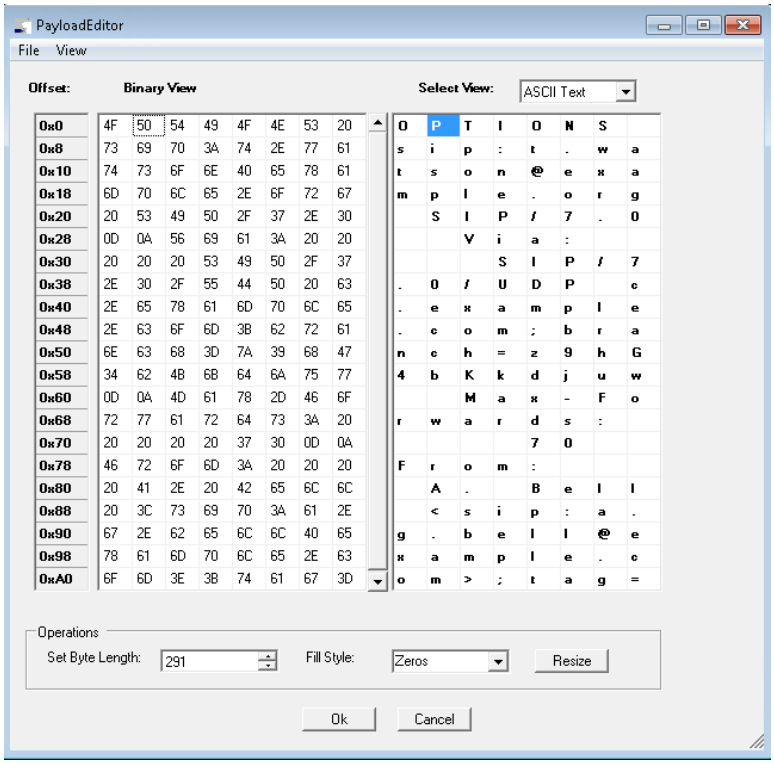
Attack Designer podporuje ako už bolo spomenuté aj možnosť importovať útoky zo zachytených paketov v *pcap* súboroch. To je možné cez menu *File* a následne *Import*. V nastaveniach je možné definovať, či majú byť importované všetky pakety alebo či len záťaže paketov a či chceme odstrániť prázdne pakety. Po načítaní súboru *pcap* je nutné vybrať, ktoré pakety z danej IP pochádzajú od servera a ktoré od klienta (pozri nasledujúci obr.)



Obr. 5.8: Voľba strán pri importovaní pcap súboru

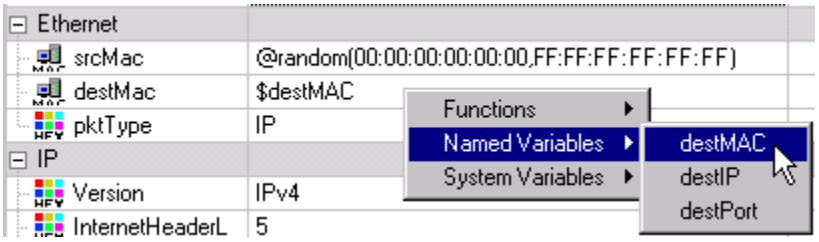
Menším problémom je, že po importe aplikácia nerozpozná aplikačný protokol a jeho obsah vloží do záťaže transportného protokolu (stalo sa to pri všetkých pokusoch importovať SIP alebo IAX dáta).

Pre testy s modifikovaným SIP záhlavím realizované v nasledujúcej kapitole je dôležitá hodnota UDP záťaže Payload, tá býva v hexa tvare a je možné ju editovať pomocou jednoduchého vstavaného editoru. Ten je nanešťastie pre úžívateľa nepriateľsky optimalizovaný a tak jednoduchá editácia niekoľkých predvolených hodnôt trvá neprimeranú dobu.



Obr. 5.9: Editor záťaže Payload

Aplikácia ďalej ponúka možnosť definovať si vlastné premenné, ktoré bude nutné pri spúšťaní testu v Avalanche Commanderi nastaviť na požadované hodnoty. Je tak zjednodušená úprava základných parametrov ako zdrojová, cieľová MAC alebo IP adresa a porty, počet opakovaní a prestávka medzi opakovaniami, ktoré neni nutné pri ich potencionálnej zmene v iných podmienakch meniť. Každá premenná sa definuje začínajúcim znakom „$". Použitie je na obrázku.



Obr. 5.10: Použitie premenných na vyplnenie položiek hlavičky

Okrem premenných má aplikácia vstavaných aj niekoľko systémových funkcií, ktoré začínajú znakom „@" a priraďujú sa cez rovnakú ponuku kliknutím pravého tlačidla myši do požadovaného poľa hlavičky. Ich využitie je vhodné pri opakovanom testovaní v rámci jedného scenáru. Na výber sú funkcie ako *@random* (vyberie náhodnú hodnotu z rozsahu) a *@range* (vyberie hodnotu z rozsahu, iteratívne).

Poslednou vlastnosťou programu je tvorba novej protokolovej sady, ktorú je možné aj použiť k tvorbe vlastného útoku. Tá zahŕňa kompletné definovanie polí hlavičky vlastného protokolu.

V rámci školou vlastnenej licencie, TestCenter obsahuje približne tri desiatky SIP útokov s modifikovaným správami (fuzzing), ktoré sú testované v ďalšej kapitole.

6 Útoky s modifikovanými správami

Správy s modifikovaným záhlavým predstavujú bezpečnostné riziko. Analyzátory syntaxu (z ang. parser) u ústrední musia dôsledne zvažovať hraničné podmienky a podvrhnuté vložené časti ako súčasť ich dizajnu. Tzv. *fuzzing* útoky rôznych internetových systémov a aplikácii používajú vytvorený obsah, aby ústredniam spôsobili nežiadúce problémy. Veľa z nasledujúcich správ je konštruovaných tak, aby ústredne testovali v bodoch, ktoré takéto útoky používajú k nalomeniu bezpečnosti. [41]

Testy s modifikovaným záhlavým sú teda zamerané na overenie chovania sa ústrední pri prijímaní resp. spracovávaní správ s neobvyklou a štandardu (pre SIP - RFC 3261 a pre IAX2 – RFC 5456) neodopovedajúcou štruktúrou. Väčšina z modifikovaných správ pre SIP bola súčasťou vývojovej štúdie RFC 4475 - SIP Torture Test Messages zameranej na testovanie SIP implementácii. Jedná sa o základné a najnáročnejšie správy na obsluhu i prostriedky (z hľadiska nároku na systém), a teda OPTIONS, INVITE, REGISTER a pre IAX2 správy NEW a REGREQ. Jednotlivé konfiguračné súbory útokov pre Spirent Attack Designer sú súčasťou prílohy.

Pre softwarové ústredne alebo iné implementácie je pre takéto modifikované správy akceptovateľné, ak ich zamietnu, avšak mali by to robiť iba vtedy, ak hodnota daného poľa je dôležitá pre ďalšie spracovanie [41]. Ako bude vidieť, niektoré z ústrední sú veľmi konzervatívne, správy zamietajú aj keď nemusia, iné ich zas liberálne niekedy modifikujú do správneho tvaru, alebo naopak sa s nimi nedokážu vysporiadať a znefunkčnia resp. zacyklia sa.

Útoky boli realizované pomocou Spirent TestCenter C1 a zadaných ústrední. Testovacia záťaž bola nastavená v testoch na simuláciu účastníckeho spojenia, do ktorého bol počas testu následne pustený nakonfigurovaný (nastavené potrebné adresné parametre) útok. Na každej z ústrední bolo počas testov zapnuté logovanie a debug komunikácie, čo umožnilo vidieť chybové hlášky a upozornenia, ale aj sled spracovávaných správ. Takisto bolo na rozhraní sieťovej karty aktivované zachytávanie pomocou analyzátoru Wireshark, ktorý slúžil ku kontrole a analýze vymieňanej komunikácie medzi jednotlivými ústredňami a TestCentrom.

6.1 Modifikované SIP OPTIONS žiadosti

Tento typ žiadosti slúži na overenie podporovaných vlastností a metód prenosu u SIP implementácie, čo môže byť indikáciou či ústredňa bude schopná prijať INVITE žiadosť k zahájeniu relácie. Pri štandardnej komunikácii koncová strana (ústredňa) na OPTIONS žiadosť odpovedá správou 200 (OK) ak je schopná hovor prijať, 4xx ak nie je, napr. 486 (BUSY HERE), tj. ak je zaneprázdnená a pod. Útoky týmto typom správy boli generované zo Spirentu z rozsahov adries 192.168.20.242-245. IP adresa PC s ústredňou, ktorej chovanie sa pri prijímaní týchto správ je predmetom skúmania, je v testoch 192.168.20.143, ak nie je napísané inak.

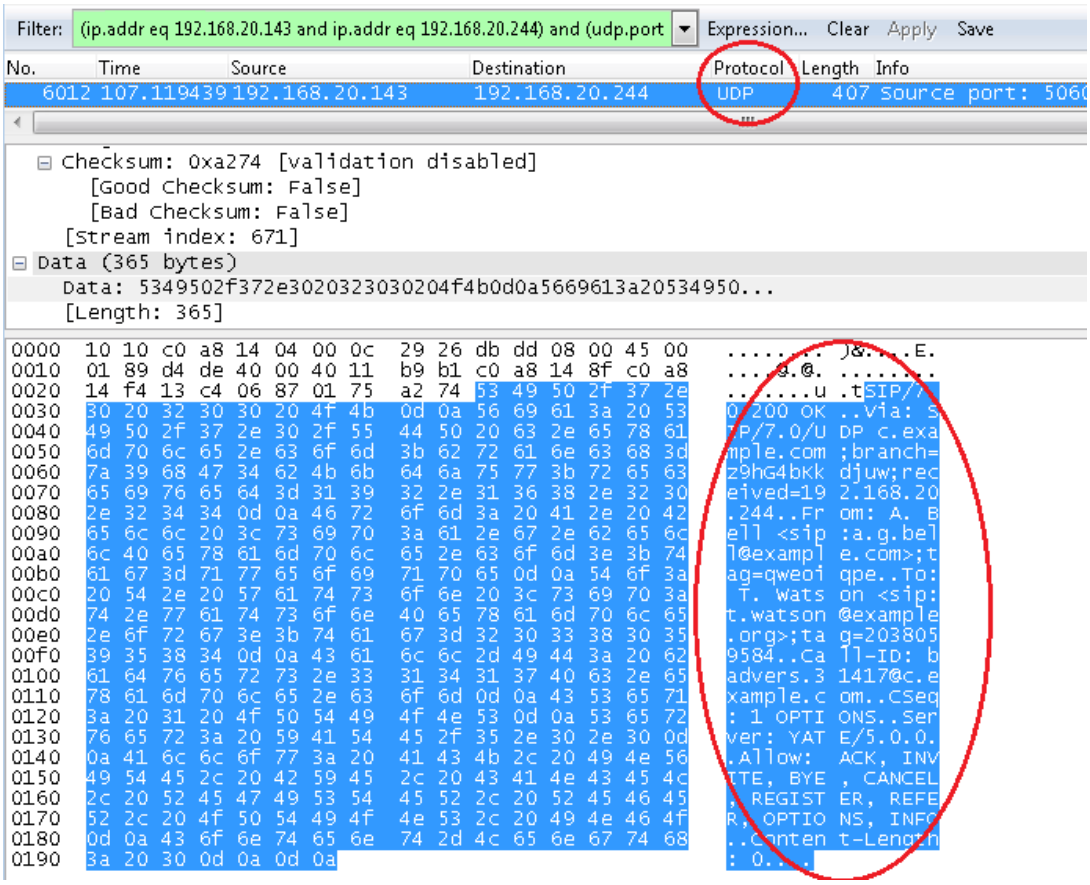
Príkladom štandardnej OPTIONS žiadosti je:

```
OPTIONS sip:1001@192.168.20.244 SIP/2.0
To: sip:1001@192.168.20.244
From: sip:1000@192.168.20.244;tag=33242
Max-Forwards: 7
Via: SIP/2.0/UDP 192.168.20.143;branch=z9hG4bKhjhs8ass877
Call-ID: badbranch@192.168.20.143
CSeq: 8 OPTIONS
Accept: application/sdp
Content-Length: 0
```

6.1.1 *Nesprávna verzia SIP záhlavia*

Na tento typ útoku bola použitá generácia SIP OPTIONS správ s verziou 7.0 v záhlaví. Podľa RFC 3261 je toto nesprávne (aktuálna verzia je 2.0) [15], takže správa by mala byť v predpokladanom scenári zamietnutá a pretože je to neočakávané, mala by zmiasť alebo znefunkčniť SIP implementáciu.

Samotný analyzátor Wireshark nedokázal neštandardnú správu rozpoznať ako správu protokolu SIP a eviduje ju ako UDP datagram, kde je až v jeho tele (Data) vidieť štruktúru neštandardnej SIP správy (obr. 5.15).

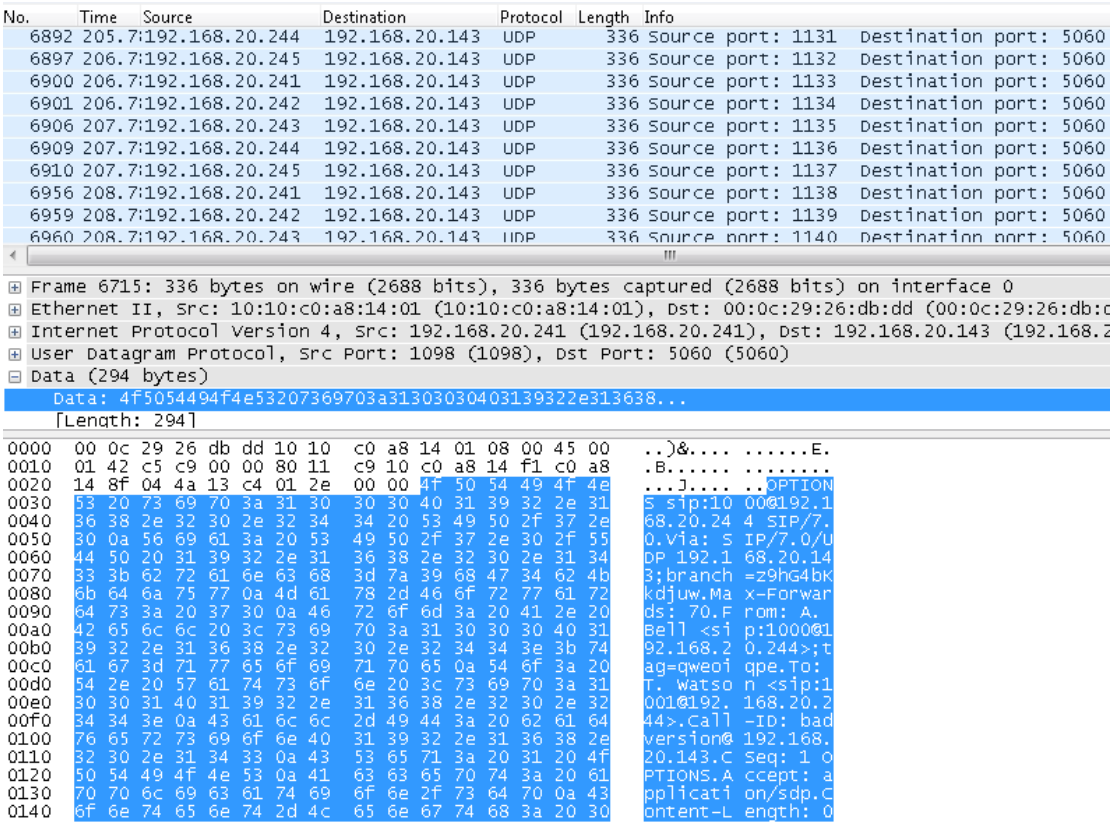


Obr. 6.1: V UDP datagrame sa prenáša neštandardná SIP OPTIONS správa.

Asterisk na prijaté SIP OPTIONS správy podľa očakávaného chovania nereagoval. Správy síce boli zachytené v logu ústredňe ako prijaté, ale žiadna reakcia zo strany ústredne na ne nenasledovala, čo bolo overené aj zachytenou komunikáciu pomocou analyzátoru Wireshark. Z nej je zrejmé, že ani na jednu správu z TestCentra, generovaných na rôznych portoch a v úvode spomínanom rozsahu IP, a následne doručených ústredni (cieľová IP 192.168.20.143) nebola odoslaná odpoveď (obr. 5.16).

```
<--- SIP read from UDP:192.168.20.244:1156 --->
OPTIONS sip:1000@192.168.20.244 SIP/7.0
Via: SIP/7.0/UDP 192.168.20.143;branch=z9hG4bKkdjuw
Max-Forwards: 70
From: A. Bell <sip:1000@192.168.20.244>;tag=qweoiqpe
To: T. Watson <sip:1001@192.168.20.244>
Call-ID: badversion@192.168.20.143
CSeq: 1 OPTIONS
Accept: application/sdp
Content-Length: 0
<------------->
```

Obr. 6.2: Prijatá SIP OPTIONS v logu Asterisku



Obr. 6.3: Zachytená komunikácia z TestCentru do ústredne Asterisk

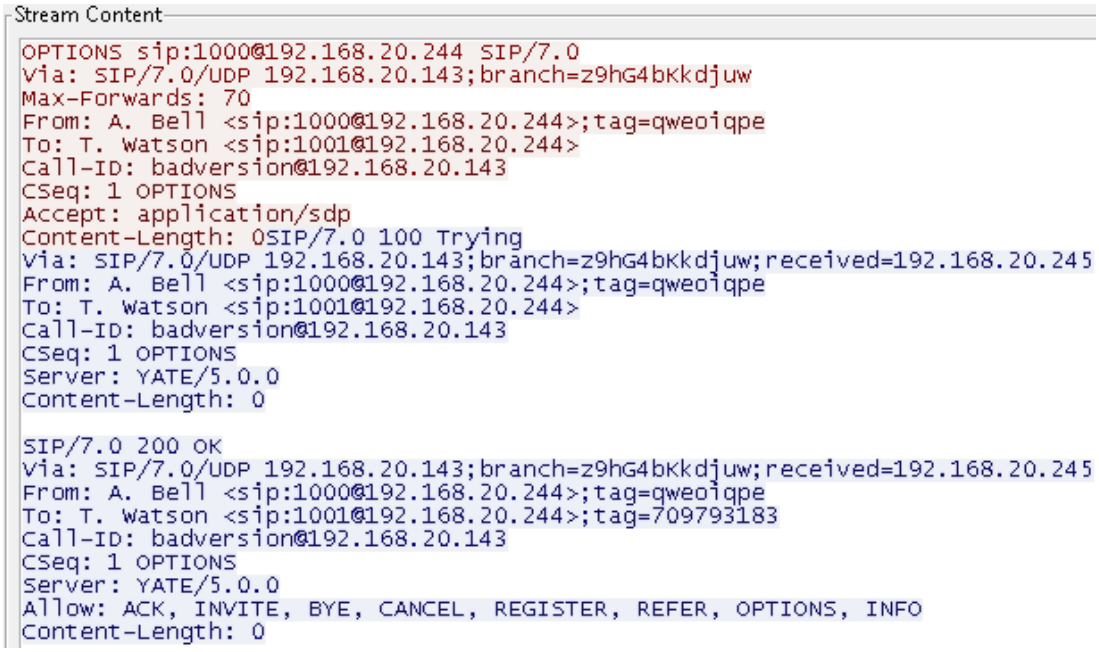
Freeswitch sa rovnako podľa očakávaní nedokáže vysporiadať s týmto typom správy. Na prichádzajúce SIP OPTIONS správy nereaguje takisto ako Asterisk.

Ústredňa YATE sa zachovala neočakávane a na správu reagovala potvrdením 200 (OK) odosieľatelovi (IP Spirentu 192.168.20.244) so zachovanou neštandardnou hodnotou verzie tj. 7.0.

```
------
20141119142103.703687 <sip:INFO> 'udp:0.0.0.0:5060' sending code 200 0x7f5b380017
20 to 192.168.20.243:1145 [0x7f5b634296b0]
------
SIP/7.0 200 OK
Via: SIP/7.0/UDP 192.168.20.143;branch=z9hG4bKkdjuw;received=192.168.20.243
From: A. Bell <sip:1000@192.168.20.244>;tag=qweoiqpe
To: T. Watson <sip:1001@192.168.20.244>;tag=95382560
Call-ID: badversion@192.168.20.143
CSeq: 1 OPTIONS
Server: YATE/5.0.0
Allow: ACK, INVITE, BYE, CANCEL, REGISTER, REFER, OPTIONS, INFO
Content-Length: 0
```

Obr. 6.4: Odpoveď odoslaná ústredňou YATE ako reakcia

To potvrdzuje aj zachytený obsah výmeny správ vo Wiresharku, kde je vidieť ako sa ústredňa nenechá pomýliť neštandardným číslom verzie a posiela odpovede (najprv 100 Trying a následne 200 OK) so zachovanou verziou v záhlaví.



Obr. 6.5: Zachytený obsah jedného toku správ medzi TestCentrom a YATE

Kamailio rovnako ako Asterisk a Freeswitch na neštandardné správy nereaguje ale v pokročilejšom výpise (oproti predchádzajúcim ústredniam) zachyteného logu je vidieť konkrétne chybové oznámenie na špecifickú položku hlavičky tzv. *Via*.

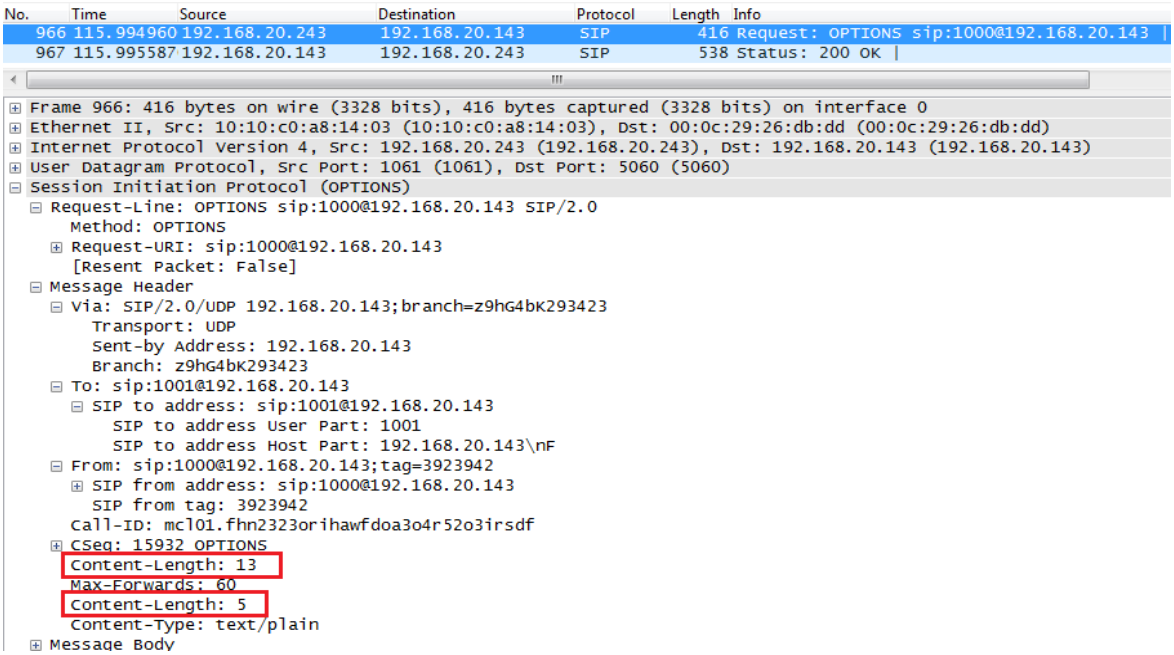
Obr. 6.6: Chybové onámenie v logu Kamailio ako reakcia na neštandardnú verziu

Opensips podobne ako Kamailio reagoval sledom chybových oznámení odkazujúcich na neštandardnú verziu v hlavičke v poli *Via*.

6.1.2 Viacnásobné pole Content-Length

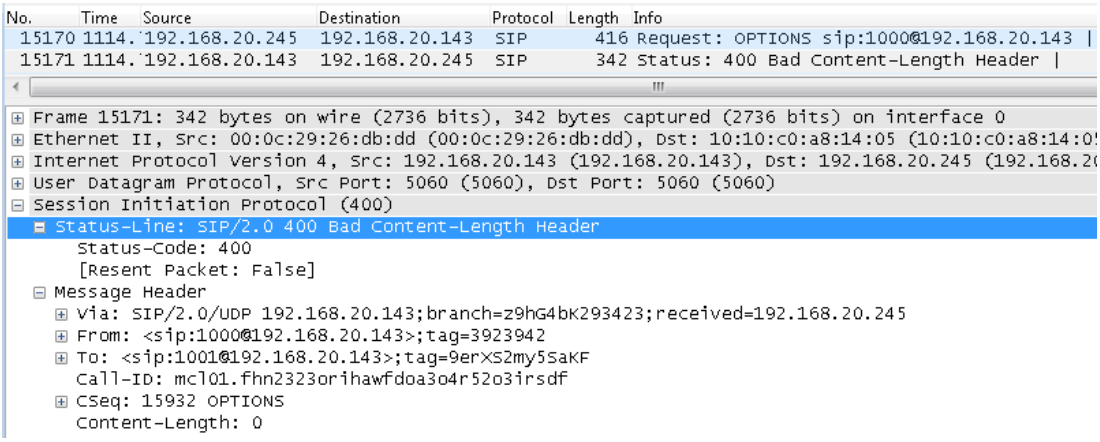
Tento typ útoku bol realizovaný pomocou SIP žiadosti OPTIONS. V nej sa pole *Content-Length* určujúce dĺžku (v dekadickom čísle oktetov) tela správy nesúceho obsah správy vyskytovalo dvakrát, čo je neštandardné. Ústredna nebude poznať dĺžku tela čo by malo spôsobiť nepredvítelné chovanie. Ak správa neobsahuje telo, štandardne je veľkosť nastavená na 0. [15]

Asterisk po prijatí tejto neštandardnej žiadosti dokázal zaslať štandardnú odpoveď 200 OK, ktorou značí schopnosť v komunikácii pokračovať.



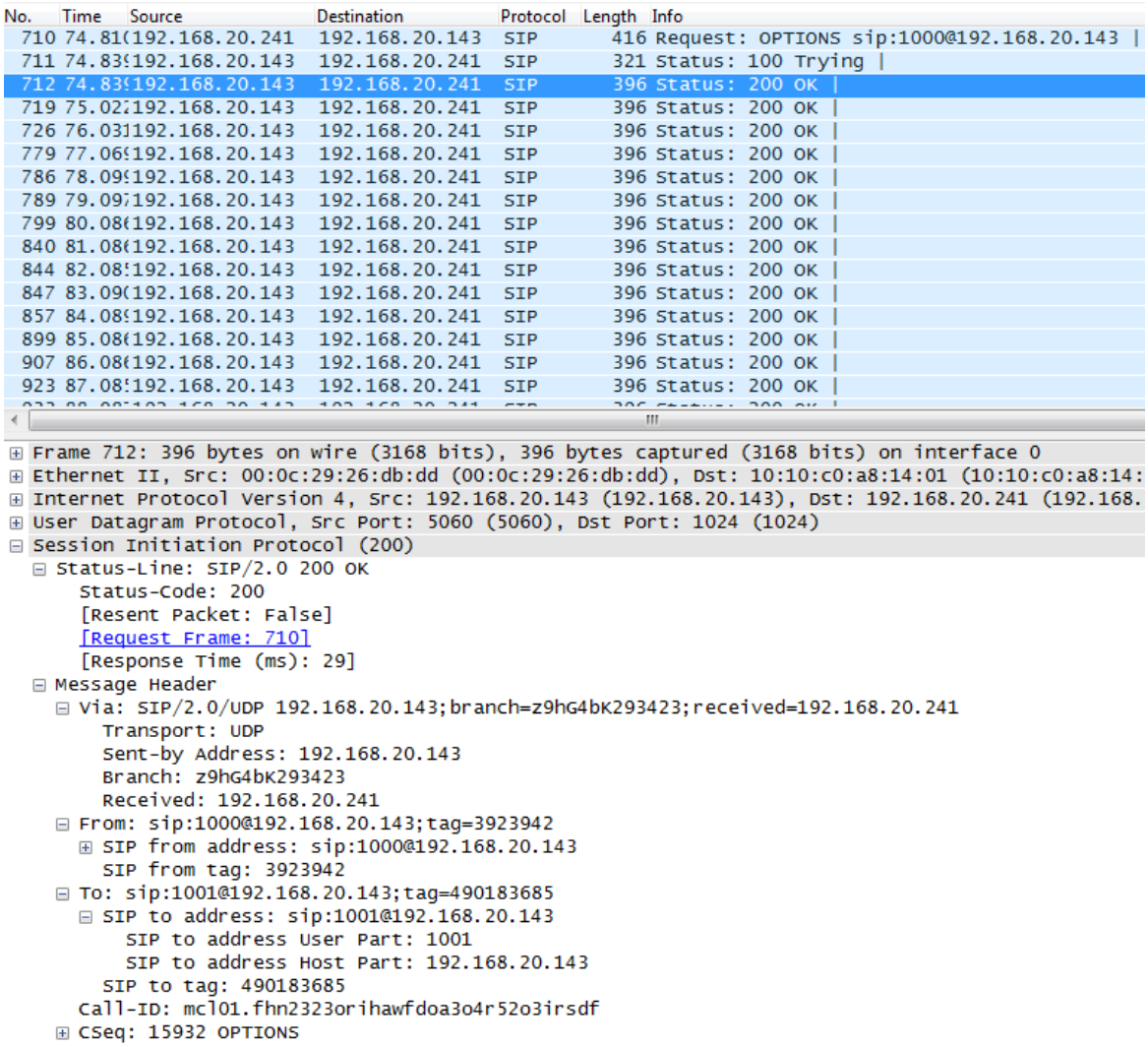
Obr. 6.7: Žiadosť Asterisku s dvojitým poľom dĺžky a následná odpoveď

Freeswitch zareagoval odpoveďou 400. Tá značí tzv. *Bad Request*, teda je odpoveďou na nezrozumiteľný syntax žiadosti. Mala by obsahovať aj dôvod. [15] Ten ústredňa poskytuje ako nesprávnu dĺžku hlavičky (Bad Content-Length Header) viď obrázok.



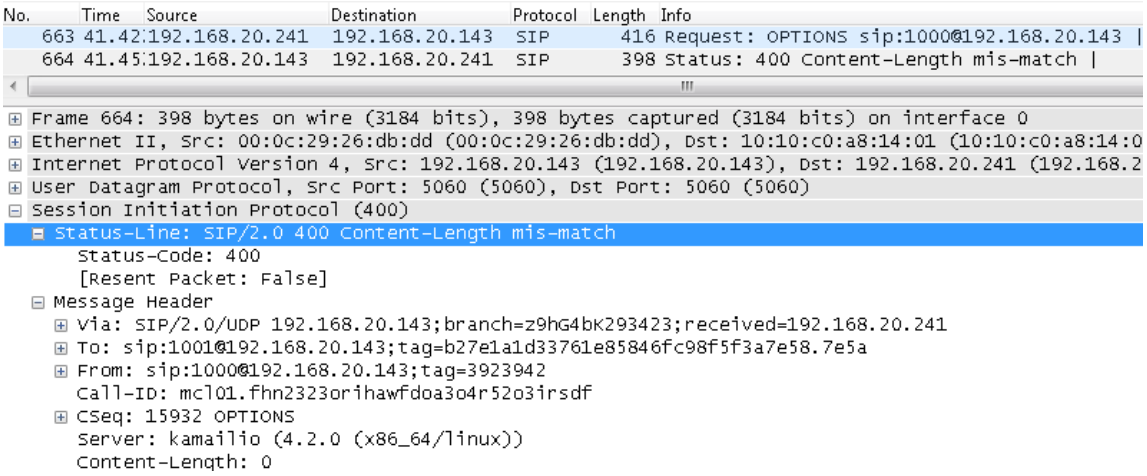
Obr. 6.8: Odpoveď 400 zaslaná Freeswitchom

Yate na prijatú neštandratnú žiadosť reagovala odpoveďou 100 Trying a následnym periodickým posielaním odpovede 200 OK až do ukončenia zachytávania. Z tohto netypického cyklického chovania sa dá usúdiť, že ústredňa sa nedokázala zkonsolidovať.



Obr. 6.9: Cyklický sled odpovedí od Yate

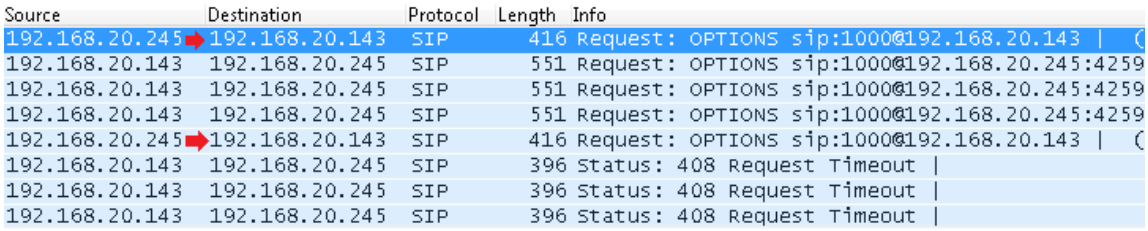
Podobne ako Freeswitch reagovala aj ústredňa Kamailio, zaslaním odpovede 400. Tentoraz s odôvodnením Content-Length mis-match (nezhoda v dĺžke tela).



Obr. 6.10: Odpoveď Kamailio na dvojitú dĺžku tela

Opensips ústredňa na žiadosť s viacerými poľami dĺžky cyklicky odpovedala zasielaním rovnakej OPTIONS žiadosti alebo odpoveďou 408 Request Timeout, ktorá sa používa ak server nedokáže vytvoriť odpoveď v rozumnom čase [15]. Z tohto chovania sa dá

posúdiť, že ju takáto žiadosť úplne zmätie a nedokáže na ňu reagovať ani zrozumiteľnou odpoveďou (napríklad ako Kamailio).

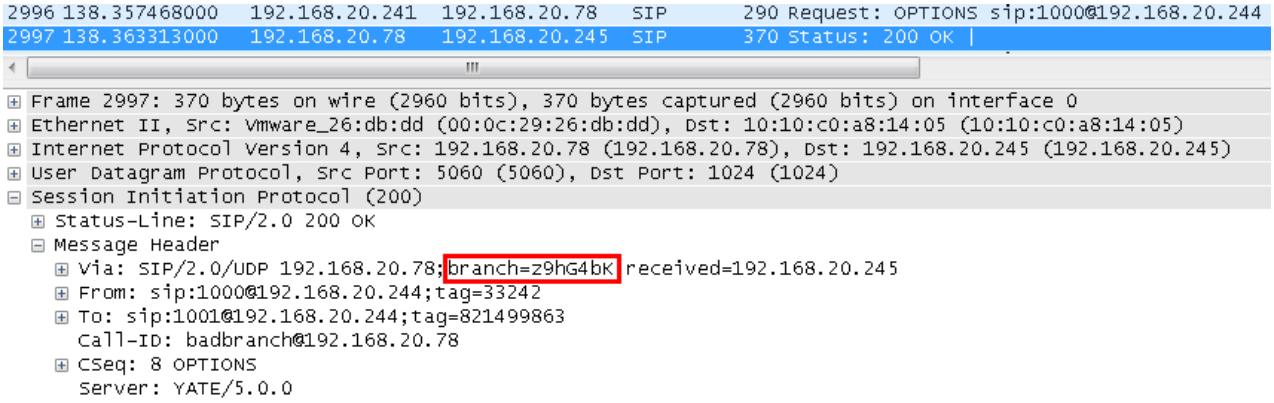


Obr. 6.11: Zmätené chovanie ústredne (modifikovaná OPTIONS vyslaná od TestCentra označ. šípkou)

6.1.3 Chýbajúca časť branch v položke Via

Tento útok posiela SIP OPTIONS správu s chýbajúcim dodatočným branch identifikátorom, ktorý nasleduje po povinnej časti „**z9hG4bK**“, podľa novšieho štandardu RFC 3261. Musí byť teda unikátny, zložený z povinnej časti, za ktorou nasleduje náhodne vygenerovaná dodatočná časť, napr. **z9hG4bK***asdFr*. Takáto správa bola ešte prípustná v starej verzii RFC 2543, ale moderné ústredne, ktoré tento typ nemajú v podpore môže pomýliť alebo zhodiť.

S týmto typom správy si poradia všetky ústredne a reagujú na ňu očakávanou odpoveďou 200 OK. V tomto teste bola IP adresa PC s ústredňou výnimočne 192.168.20.78.



Obr. 6.12: Odpoveď 200 OK odoslaná ústreňami a neštandardný branch údaj

6.1.4 Medzery v adrese poľa To:

Tento útok pošle SIP OPTIONS správu s medzerou v adresnom poli To hlavičky SIP. Konkrétne v testovanom prípade to je

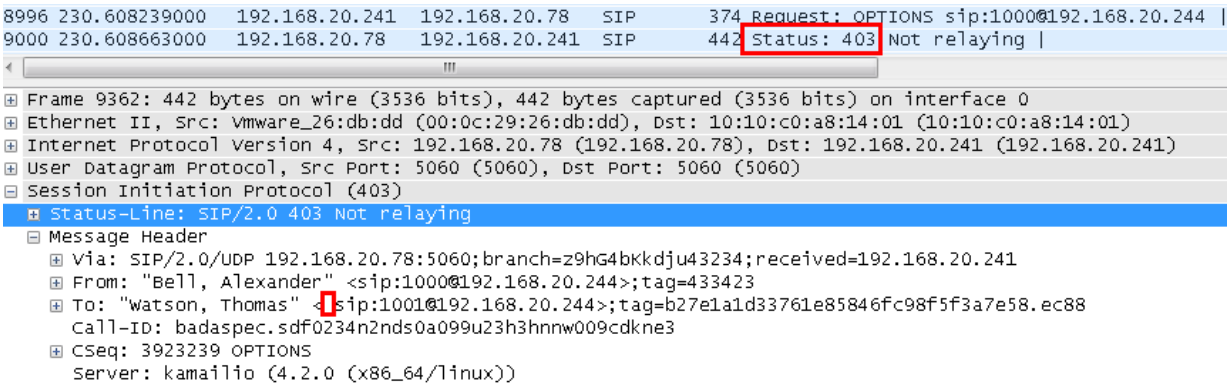
```
To: "Watson, Thomas" < sip:1001@192.168.20.244>
```

kde sa pred sip adresou nachádza medzera. Znova sa jedná o modifikáciu štandardu. V tomto teste bola znova IP adresa PC s ústredňou 192.168.20.78 ako aj v predchádzajúcom.

Asterisk aj Freeswitch odpovedajú štandardne 200 OK. Ústredňa YATE na príchodziu správu s týmto poľom odpovie síce takisto kladne, ale podobne ako v iných prípadoch ju začne cyklicky zasielať dokola až do skončenia testu.

Kamailio odpovedá prekvapujúco hláškou 403, ktorá sa používa v zmysle Forbidden tj. Zakázaný, teda ak server žiadosti porozumel, ale odmieta ju vykonať. A žiadosť by nemala byť opakovaná. [15] Tu má tvar *Not relaying* (pozri obr. 6.13), čo sa u tejto ústredne vyskytuje, ak žiadosť vyhodnotí ako bezpečnostné riziko, napr. účastníka z mimo lokálnej domény. Tento účet však na ústredni je nakonfigurovaný a tak sa dá predpokladať, že pridaná medzera k číslu účtu (vnímaná pravdepodobene ako ďalší znak) sa ústredni javí ako cudzí účastník.

Opensips podobne ako Kamailio reaguje odpoveďou 403, ale u neho je to *Rely forbidden*, ktoré má identický význam.



Obr. 6.13: Odpoveď 403 od Kamailio na žiadosť s medzerou v adrese

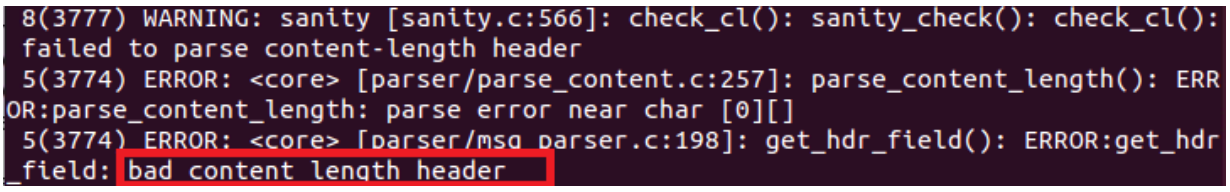
6.1.5 *Chýbajúca medzera v poli From:*

Tento útok posiela SIP OPTIONS žiadosti s chýbajúcou medzerou medzi zobrazovaným menom volajúceho (v tomto prípade "*caller*") a začiatkom jeho klapky *sip:1000@192.168.20.244* v hlavičke poľa From. V aktuálnej RFC to nie je povolené, avšak predpokladá sa, že v novších už áno [41].

Asterisk zareagoval odpoveďou 200 OK. Freeswitch odpovedal rovnako ako Asterisk, ale odpoveď cyklicky odosielal, z čoho sa dá predpokladať, že žiadosť ho zmiatla.

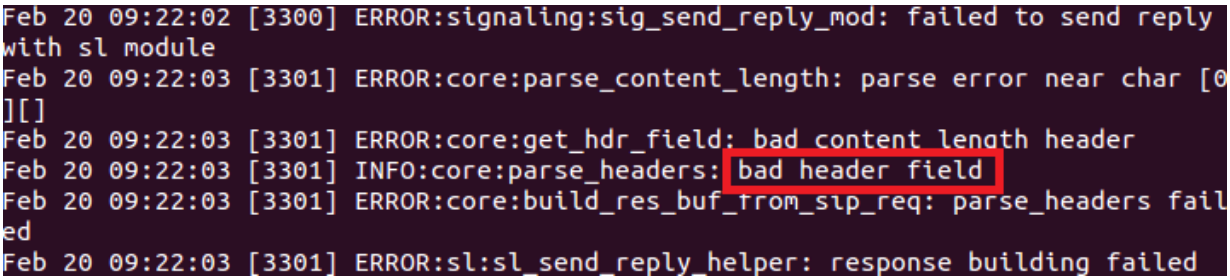
Yate znova odpovedal na prvú žiadosť správou 100 Trying a následne cyklicky zasielal 200 OK ako Freeswitch.

Kamailio na tento typ žiadosti vôbec nereagoval odpoveďou, vo výpise logu je vidieť chybové logy o nesprávnej hlavičke, konkrétne časť *Content length*, ktorá ako už bolo spomenuté v útoku 6.1.2 určuje dĺžku tela správy. Tá tu však bola v poriadku a tak sa dá predpokladať, že táto neštandardná správa ústredňu zmiatla a chybu nedokázala presne definovať.



Obr. 6.14: Výpis logu Kamailio na útok s chýbajúcou medzerou v poli From

Opensips na tento typ žiadosti vôbec nereagoval odpoveďou, vo výpise logu je vidieť okrem chybových hlásení o neschopnosti zostaviť odpoveď aj informáciu o nesprávnej hlavičke, ktorá však nie je bližšie špecifikovaná.



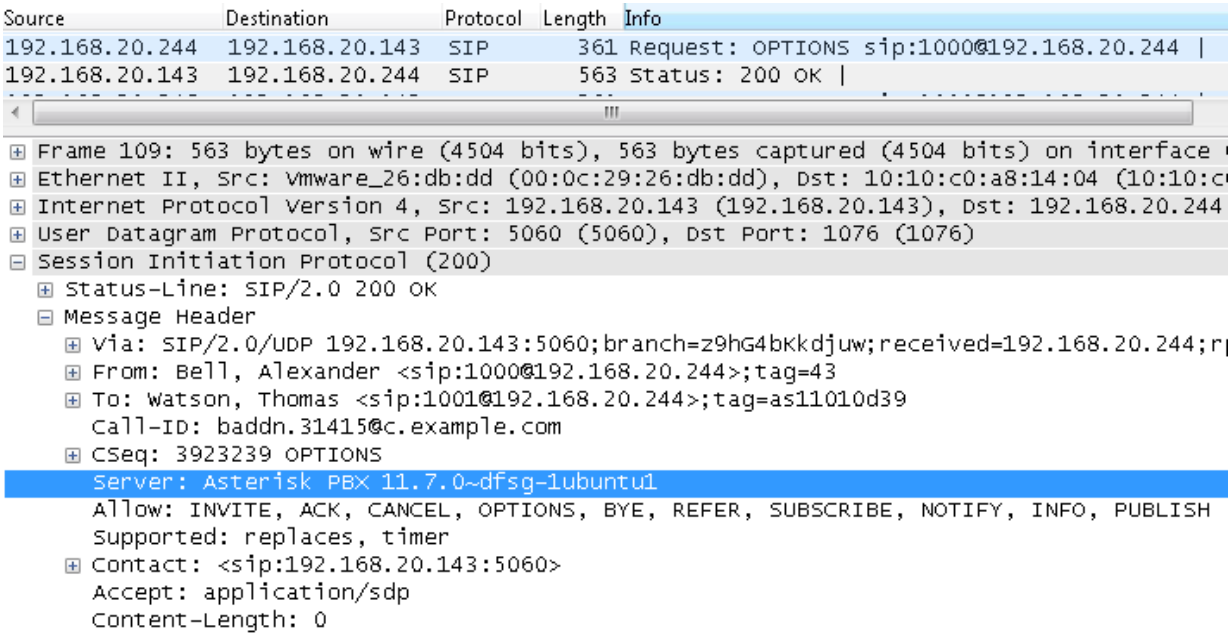
Obr. 6.15: Výpis logu Opensips na útok s chýbajúcou medzerou v poli From

6.1.6 *Aliasy bez úvodzoviek*

Tento útok posiela SIP OPTIONS správu so zobrazovacím menom (aliasom) volajúceho a volaného bez úvodzoviek, a obsahujúc nepísmenový tzv. non-token znak (čiarku). Teda tvar:

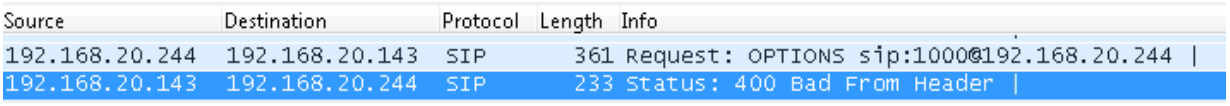
```
From: Bell, Alexander <sip:1000@192.168.20.244>;tag=43
To: Watson, Thomas <sip:1001@192.168.20.244>
```

Takýto tvar je považovaný za invalidný. Asterisku tento nesprávny tvar nespôsobuje problémy a odpovedá 200 OK. Pozri obr.



Obr. 6.16: Odpoveď 200 OK od Asterisku na nesprávne aliasy

Freeswitch odpovedá správou 400 Bad From Header, čo je definované číslo odpovede na zdeformované tvary žiadostí [15]. Na pole *To:* nereaguje, čo by sa dalo vysvetliť tak, že akonáhle rozpozná neznámy tvar v poli *From:* generuje odpoveď, a tak nereaguje už ani na pole *To*:, ktoré nasleduje v hlavičke OPTIONS žiadosti po ňom.



Obr. 6.17: Odpoveď Kamailio 400 Bad From Header

Yate prekvapujúco odpovedá po prvotnom 100 Trying, zaslaním odpovede 200 OK a teda podobne ako Asterisku, mu tento neštandardný tvar nespôsobuje komplikácie ako v iných testoch.

Kamailio rovnako ako v predchádzajúcom type útoku neodpovedá a v logu vypisuje chybové hlášky zamerané na chybnú hlavičku, presnejšie časť *Content length*, čo je však nesprávne diagnostikovaná časť. Opensips reaguje znova odpoveďou 403 Rely forbiden.

6.1.7 *Medzery na konci tzv. Request-Line*

Request-Line riadok začína s tokenom označujúcim metódu (v tomto prípade OPTIONS), za ním nasleduje tzv. Request-URI (teda *sip:1000@192.168.20.244*) a nakoniec verzia protokolu, za ktorou má nasledovať zalomenie [15]. V tomto útoku však nasledujú tri medzery a až potom zalomenie. Je možné, že ústredne sa budú snažiť kompenzovať tento nedostatok automaticky.

U Tohto typu modifikácie už Wireshark nerozpoznal, že sa jedná o protokol SIP ale zachytil záťaž protokolu UDP, čiže následne bolo možné dáta konvertovať do SIP tvaru.

Asterisk s touto žiadosťou už stabilne nemá problémy a odpovedá 200 OK ako v predchádzajúcich testoch.

Freeswitch nedokázal odpovedať a aj v logoch je vidieť len prijatú žiadosť, no žiadnu chybovú hlášku či odpoveď.

Yate takisto odpovedal hláškou 200 OK. Pozri obr.

```
20150221113735.997550 <sip:INFO> 'udp:0.0.0.0:5060' sending code 200 0x7f11a4001
450 to 192.168.20.245:1024 [0x7f11d40426b0]
------
SIP/2.0 200 OK
Via: SIP/2.0/TCP 192.168.20.143:5060;branch=z9hG4bK299342093;received=192.168.20
.245
From: <sip:1000@192.168.20.244>;tag=329429089
To: <sip:1001@192.168.20.244>;tag=1535322033
Call-ID: trws.oicu34958239neffasdhr2345r
CSeq: 238923 OPTIONS
Server: YATE/5.0.0
Allow: ACK, INVITE, BYE, CANCEL, REGISTER, REFER, OPTIONS, INFO
Content-Length: 0
```

Obr. 6.18: Odpoveď 200 OK od ústredne Yate

Kamailio znova neodosiela žiadnu odpoveď. V logu chybovým hlásením oznamuje neschopnosť vytvoriť odpoveď a všeobecne rozpoznájú chybnú hlavičku (*bad header*).

Opensips odpovedá klientovi správou 403 Rely forbiden, ale bližšie chybu nedefinuje.

6.1.8 Neznáme Require a Proxy-Require hodnoty

Tento útok posiela SIP OPTIONS žiadosť s neznámymi hodnotami v *Require:* a *Proxy-Require:* hlavičke SIP. Technicky je to valídne, ale keďže to nie je predpokladané môže dôjsť k neštandardnému chovaniu ústrední. [41] Polia sú používané klientom, keď oznamuje serveru, aké možnosti predpokladá, že server podporuje, aby mohlo dôjsť k spracovaniu správy. Nie sú povinné, ak sú však prítomné, nesmú byť odignorované. Server by mal odpovedať hláškou 420 Bad Extension, kde v poliach v týchto poliach pošle zoznam nepodporovaných metód. [15]

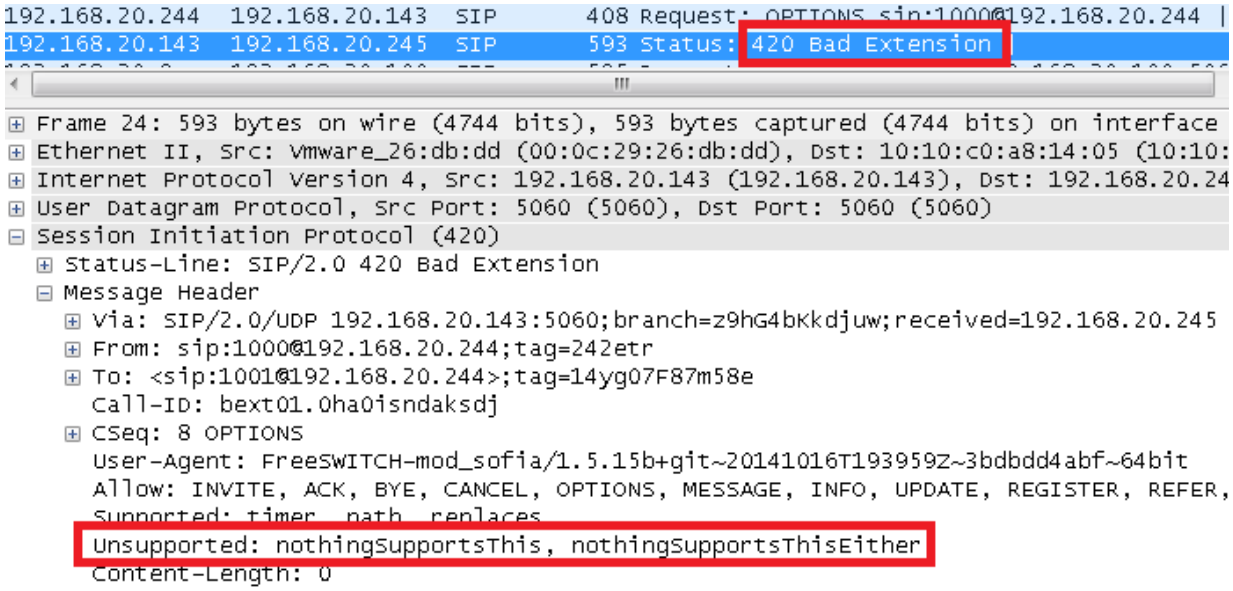
Asterisk však odpovedá správou 200 OK. V odpovedi ale uvádza podporované metódy. pozri obr.

```
Source          Destination     Protocol  Length  Info
192.168.20.244  192.168.20.143  SIP          408  Request: OPTIONS sip:1000@192.168.20.244 |
192.168.20.143  192.168.20.244  SIP          521  Status: 200 OK |

⊞ Frame 98: 521 bytes on wire (4168 bits), 521 bytes captured (4168 bits) on interface (
⊞ Ethernet II, Src: Vmware_26:db:dd (00:0c:29:26:db:dd), Dst: 10:10:c0:a8:14:04 (10:10:
⊞ Internet Protocol Version 4, Src: 192.168.20.143 (192.168.20.143), Dst: 192.168.20.24
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 1071 (1071)
⊟ Session Initiation Protocol (200)
  ⊞ Status-Line: SIP/2.0 200 OK
  ⊟ Message Header
    ⊞ Via: SIP/2.0/UDP 192.168.20.143:5060;branch=z9hG4bKkdjuw;received=192.168.20.244;r
    ⊞ From: sip:1000@192.168.20.244;tag=242etr
    ⊞ To: sip:1001@192.168.20.244;tag=as0457f919
      Call-ID: bext01.0ha0isndaksdj
    ⊞ CSeq: 8 OPTIONS
      Server: Asterisk PBX 11.7.0~dfsg-1ubuntu1
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO, PUBLISH
      Supported: replaces, timer
    ⊞ Contact: <sip:192.168.20.143:5060>
```

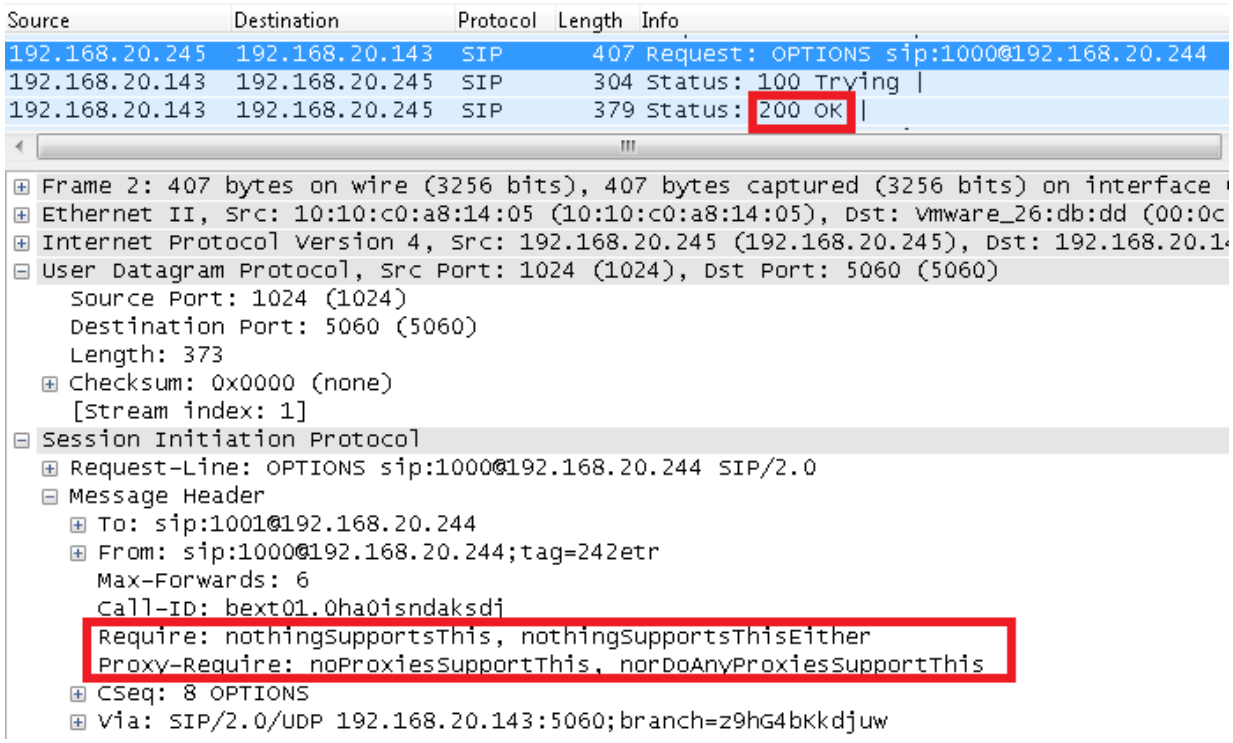
Obr. 6.19: Odpoveď Asterisku 200 OK s podporvanými hodnotami pre Require polia

Freeswitch odpovedá podľa štandardu odpoveďou 420 Bad Extension a zasiela aj zoznam podporvaných a nepodporovaných metód, pozri obr.



Obr. 6.20: Zoznam nepodporovaných metód zaslaných v odpovedi 420 od Freeswitchu

Yate naopak nemá problém a odpovedá prvotným 100 Trying a následným 200 OK, kde však hodnoty obidvoch polí zkopíruje, čo nie valídna odpoveď podľa [15].



Obr. 6.21: Odpoveď Yate na neznáme Require a Proxy-Require hodnoty

Kamailio znova neodpovedá, v logoch poskytuje iba informácie o rozpoznanej chybnej hlavičke. Opensips opakovane odpovedá 403 Rely forbiden.

6.1.9 Nesprávna adresná URI schéma

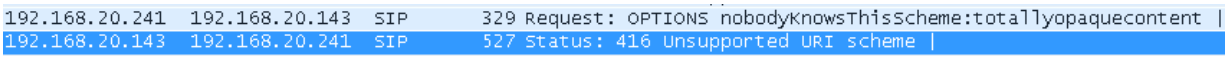
Tj. žiadosť OPTIONS s tvarom:

```
OPTIONS nobodyKnowsThisScheme:totallyopaquecontent SIP/2.0
```

namiesto sip:URI tvaru, teda:

```
OPTIONS sip:1000@192.168.20.244 SIP/2.0
```

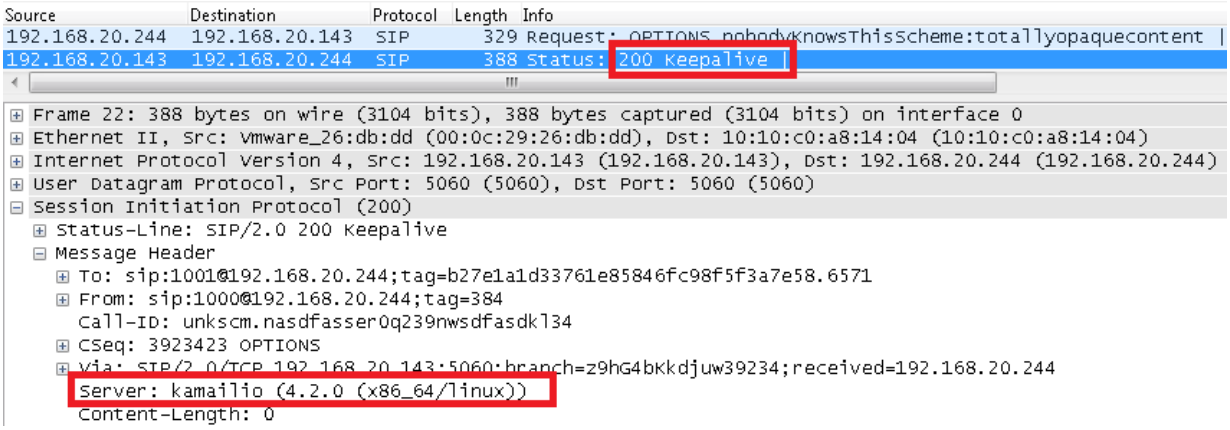
Asterisk odpovedá hláškou 416 Unsupported URI Scheme, ktorá vyjadruje, že server nemôže spracovať žiadosť pretože používa jemu neznámu adresnú schému URI. [15]



Obr. 6.22: Odpoveď Asterisku 416 Unsupported URI Scheme

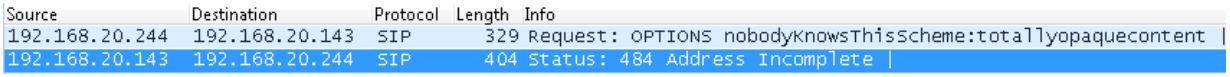
Freeswitch neodpovedá. V logu je vidieť iba prijaté žiadosti. U Yate dochádza znova k cyklickému zasielaniu kladnej odpovede.

Kamailio odpovedá správou 200 Keepalive. V logu však vypíše chybovú hlášku: „*pv_get_ruri_attr(): failed to parse the R-URI*“, ktorá je súčasťou výstupu kontroly tvaru URI adresy. Z odpovede Keepalive môžme súdiť, že chce byť informovaný o stave napr. kvôli predpokladu, že dôjde k zmene adresnej schémy na správny tvar.



Obr. 6.23: Odpoveď Keepalive od Kamailio

Opensips na tento typ správy reagoval odpoveďou 484 Address Incomplete, v logu podrobne definoval chybnú adresu tj. *bad uri.* Dá sa konštatovať, že ústredňa má takýto typ modifikácie hlavičky detailne ošetrený.



Obr. 6.24: Odpoveď 484 Address Incomplete od Opensips

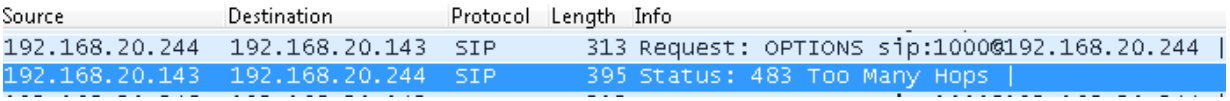
6.1.10 Nulová hodnota v poli Max-Forwards

Útok posiela OPTIONS žiadosť s hodnotou 0 v hlavičke v poli *Max-Forwards:*. Táto hodnota je povolená, ale niekedy neočakávaná. Chovanie ústrední môže byť rôzne. [41] Hodnota by mala odpovedať maximálnemu počtu skokov tj. proxy alebo brán na ceste k cieľu. Na nulovú hodnotu, by mala byť zaslaná odpoveď 483 Too Many Hops [15].

Asterisk aj Freeswitch odpovedajú kladne zaslaním 200 OK. Z čoho vyplýva, že nulová hodnota je pre nich prijateľná, aj keď fyzicky by mali byť posledné v preposielaní takejto žiadosti.

Ústredňa Yate sa znova zacykluje odpoveďou 200 OK. Potvrdzuje tak znova svoju neodladenosť a funkčné nedostatky, najmä v porovnaní s ostatnými testovanými ústredňami.

Kamailio a Opensips odpovedajú podľa odporučenia RFC 3261 odpoveďou 483.



Obr. 6.25: Kamailio odpovedá podľa odporučenia RFC správou 483

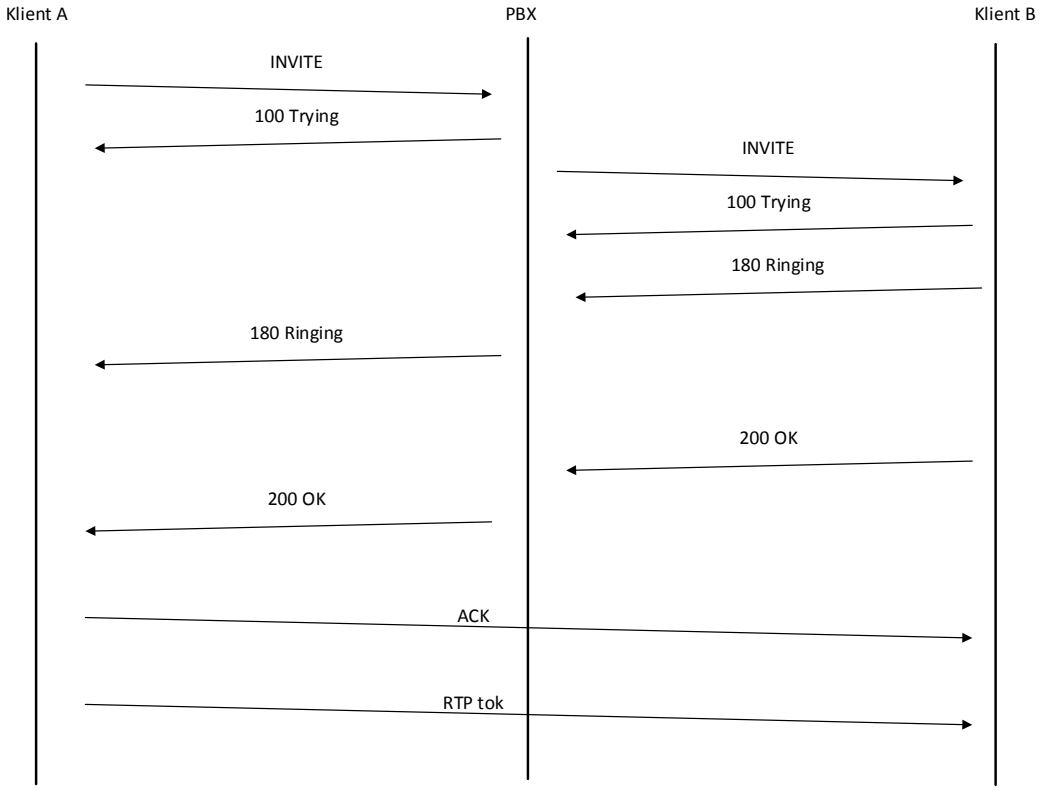
6.2 Modifikované SIP INVITE žiadosti

V nasledujúcich podkapitolách sú rozanalyzované realizované útoky s modifikovanými INVITE žiadosťami. Tie slúžia k inicializácii relácie. Klient pošle tento typ žiadosti k ústredni s parametrami v hlavičke, ktorými indikuje s kým chce nadviazať spojenie a aké vlastnosti od spojenia očakáva. Následne obdrží odpoveď typu 1xx. Typicky 100 Trying, ktorou ústredňa indikuje, že žiadosť preposlala druhej strane. Tento scénar sa zmení, ak účet nie je registrovaný a ústredňa vyžaduje autentifikáciu (implicitne aktívne nastavenie u všetkých testovaných ústrední). Vtedy klient obdrží odpoveď radu 4xx, typicky 401 Unauthorized a následne sa bude musieť autentifikovať. Tento proces bol popísaný v kapitole 3.5.1.

Testy boli realizované medzi dvoma klientami, ktorých charakter bol generovaný Spirentom. Prvý klient mal pridelenú adresu z rozsahu 192.168.20.242-5 a odosiela žiadosť klientovi s adresou 192.168.20.244. Medzi nimi bola ústredňa s IP 192.168.20.143, ktorej reakcie na modifikované správy boli predmetom skúmania. Na rozdiel od predchádzajúcich testov na správy OPTIONS, analyzované budú aj reakcie ústredne smerom k druhému užívateľovi, nakoľko práve jemu sú žiadosti adresované. Príkladom štandardnej INVITE žiadosti je:

```
INVITE sip:1001@192.168.20.244 SIP/2.0
To: sip:1001@192.168.20.244
Contact: <sip:1000@192.168.20.244>
From: sip:1000@192.168.20.244;tag=234
Max-Forwards: 5
Call-ID: sdp01.ndaksdj9342dasdd
CSeq: 8 INVITE
Via: SIP/2.0/UDP 192.168.20.143;branch=z9hG4bKkdjuw
Content-Length: 150
Content-Type: application/sdp

v=0
o=mhandley 29739 7272939 IN IP4 192.168.20.244
s=-
c=IN IP4 192.168.20.244
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC
```



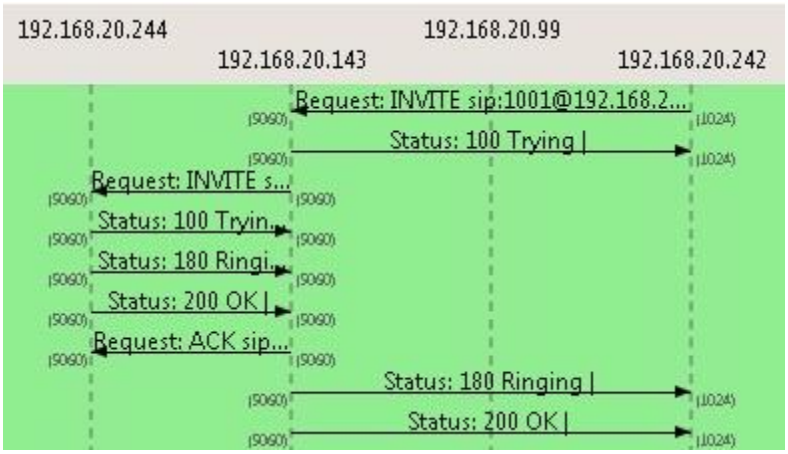
Obr. 6.26: Tzv. flow (tok dát) medzi klientami [9]

6.2.1 Neštandardné pole Accept

Útok posiela SIP INVITE správu, ktorá požaduje neznámy typ v poli Accept, konkrétne hodnotu: `text/nobodyKnowsThis` .

Ústredňa môže takýto formát odignorovať, ale môže ju aj pomýliť. Pole Accept nie je povinnou súčasťou hlavičky správy INVITE. Indikuje serveru, ktoré Content-Type typy správ môže klient akceptovať v odpovedi a vo všetkých nasledujúcich odpovediach v rámci vytvorenej relácie. Jej využitie je vhodné na indikáciu podporovaných SDP formátov. [15]

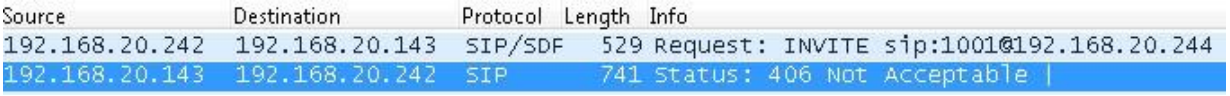
Takáto správa Asterisku prekvapujúco nespôsobila problémy a reagoval klasickým 100 Trying, pričom žiadosť preposlal koncovému užívateľovi. Výsledný tok dát zodpovedá štandardu, pozri obr.



Obr. 6.27: Tok dát Asterisku po prijatí modifikovanej INVITE

Freeswitch podľa predpokladu reaguje ošetrujúcou hláškou 406 Not Acceptable, ktorá sa používa na indikáciu neschopnosti ústredne generovať odpovede, ktoré majú tvar zodpovedajúci požiadavkám klienta naznačeným v poli Accept. [15] Prvotne osloví druhého

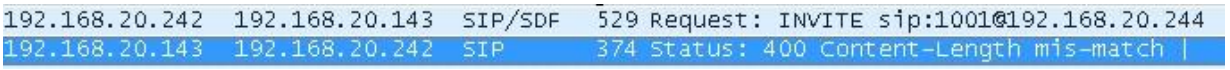
klienta metódou NOTIFY, cez ktorú informuje o požiadavke v poli Accept v žiadosti INVITE a následne odpovedá iniciátorovi spomínanou hláškou (pozri nasledujúci obrázok).



Obr. 6.28: Reakcia Freeswitchu na neštandardnú hodnotu Accept

Yate reaguje klasickou 100 Trying, zatiaľ čo žiadosť upraví podľa neznámeho vlastného vzorca (vynechá pole *Accept*, zmení telo) a prepošle druhému užívateľovi. Spojenie však nedokáže uskutočniť, pretože po preposlaní upravenej žiadosti nedostáva odpoveď a posiela príjemcovi druhú a to typ CANCEL, ktorou indikuje ukončenie. Iniciátora však dodatočne neinformuje. Takéto chovanie je netypické a opakuje sa u nej vo viacerých testoch.

Kamailio odpovedá hláškou 400 Content-Length mis-match (pozri obr. 6.29) a žiadosť ani nepreposiela ďalej druhej strane. Takúto správu má teda ošetrenú len minimálne, jeho definícia chyby je nesprávne určená. Po kratšom hľadaní na developerskom fóre sa dá zistiť, že sa jedná o chybu (bug) v aktuálnej verzii, ktorá sa niekedy vyskytuje pri spracovávaní INVITE žiadostí, aj keď pole Content-Length je v poriadku.



Obr. 6.29: Chybná odpoveď Kamailio po prijatí žiadosti s nesprávnou Accept

Opensips reaguje ako aj v niektorých iných testoch zákazom odpoveďou 403 Rely forbiden a chybnú žiadosť nepreposiela k cieľu.

6.2.2 Nesprávna hodnota časovej zóny v Date poli

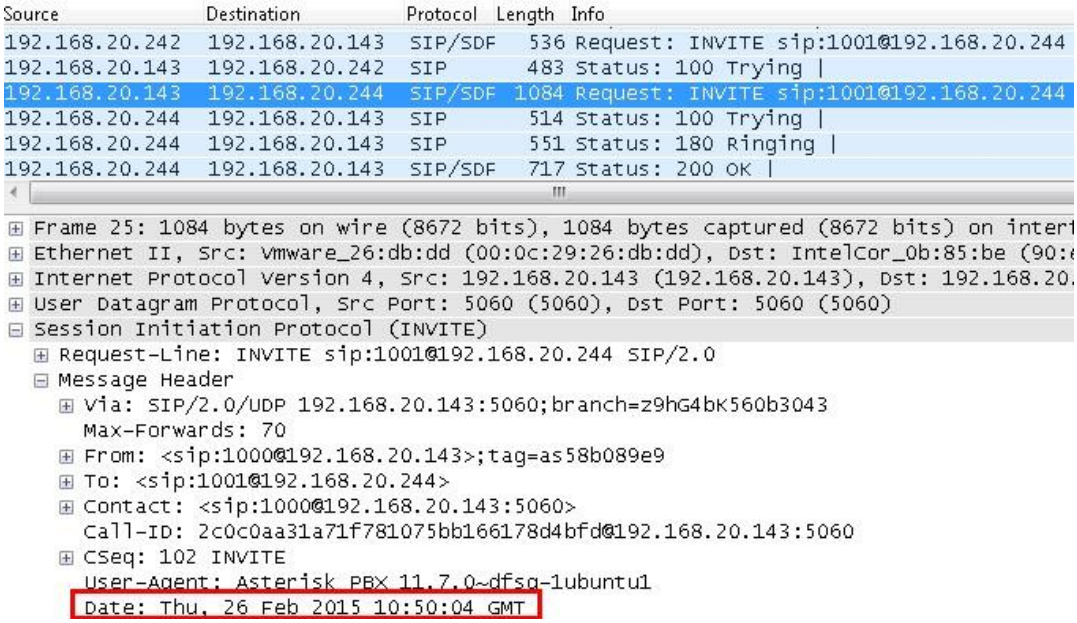
Tento útok posiela modifikovanú INVITE správu s údajom časovej zóny inej ako GMT, ktorá je jediná povolená [15]. Tento údaj sa nachádza v nepovinnom poli *Date* a indikuje kedy bola žiadosť odoslaná. Zložený je z dátumu a času, príklad formátu:

```
Date: Sat, 13 Nov 2010 23:29:00 GMT
```

zmenená žiadosť nesie údaj:

```
Date: Fri, 1 Jan 2015 16:00:00 EST
```

Asterisk s údajom nemá problém a správu nezahadzuje, práve naopak, reaguje štandardným 100 Trying. Následne žiadosť upraví o správny časový údaj a prepošle druhému koncovému užívateľovi. Potom nasleduje štadnartná realizácia spojenia (pozri nasledujúci obr.)



Obr. 6.30: Upravená žiadosť INVITE Asteriskom preposlaná druhému klientovi

Freeswitch naopak odpovedá 400 Bad Session Description (pozri nasledujúci obr.), ktorá sa používa k naznačeniu iniciátorovi, že nebolo možné žiadosť spracovať kvôli nesprávnemu syntaxu. Môžeme konštatovať napriek tomu, že nedokáže žiadosť spracovať má ošetrený takýto tvar a nespôsobuje mu problémy.

Obr. 6.31: Odpoveď Freeswitchu na INVITE s nesprávnou časovou zónou

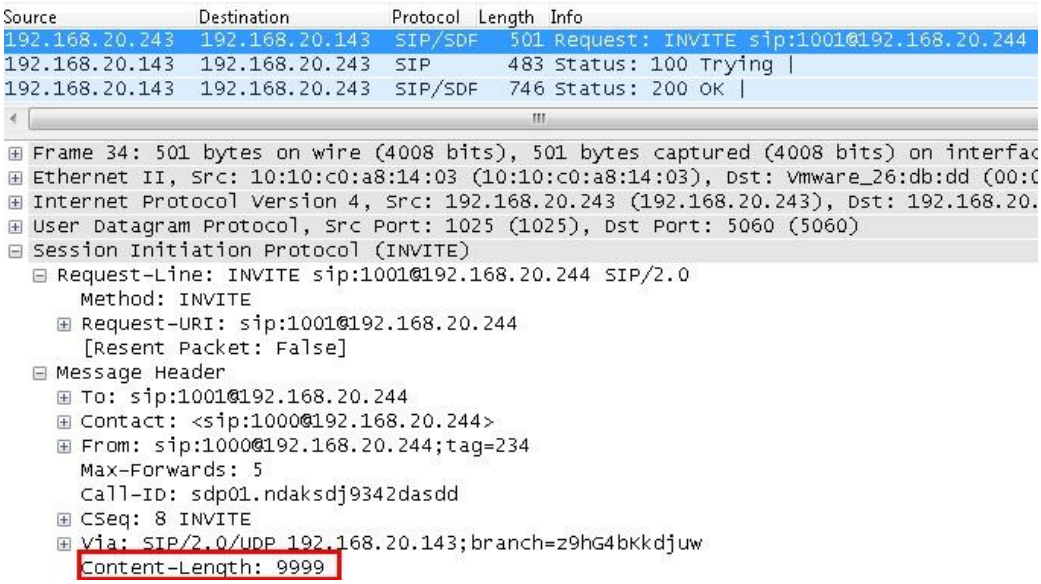
Yate rovnako ako v predchádzajúcom prípade odpovie 100 Trying, no neuskutoční spojenie. Prvotnú žiadosť upraví do podoby bez poľa *Date*, odpoveď na vlastnú modifikovanú žiadosť nedostáva a zruší ju teda metódou CANCEL.

Kamailio žiadosť vyhodnocuje znova bezpečnostné riziko a nepreposileja ju k cieľu. Iniciátora informuje správou 403 Not relaying, čím mu značí aby takúto žiadosť neopakoval, pretože ju neprepošle. Opensips reaguje takisto, odpoveď má však u neho tvar 403 Rely forbiden.

6.2.3 Hodnota Content Length väčšia ako telo správy

Útok posiela žiadosť INVITE s poľom Content-Length omnoho väčším než je skutočné telo správy. Nie je to valídne a v závislosti na transportnom protokole by mali byť očakávané rôzne reakcie. Pre UDP by mala byť takáto žiadosť zamietnutá, ale môže spôsobiť aj iné komplikácie [41]. Pole Content-Length určuje dĺžku (v dekadickom čísle oktetov) poľa Massage-Body (telo správy). V tomto útoku má hodnotu 9999 oproti bežne pohybújucej sa okolo 150.

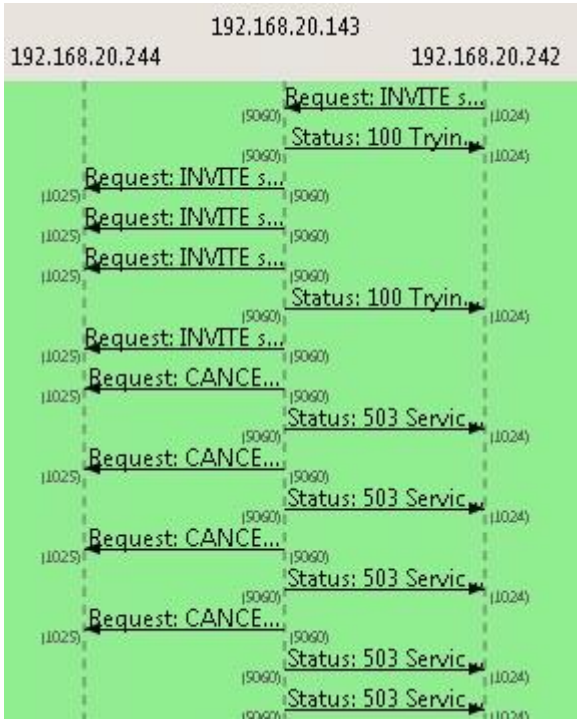
Asterisk sa prekvapujúco nenechal zmiasť a odpovedá typickou 100 Trying za ktorou nasleduje 200 OK (pozri nasledujúci obr.) a realizácia spojenia. Znova môžeme konštatovať, že takýto tvar žiadosti má ošetrený a sám ho opraví pri preposielaní druhému užívateľovi.



Obr. 6.32: Odpovede Asterisku na žiadosť s veľkou hodnotou v Content-Length

Freeswitch danú žiadosť zahodí, tj. nepreposiela ju druhej strane a iniciátorovi odošle odpovď 400 Bad Request.

Yate po prvotnej odpovedi 100 Trying a preposlaní žiadosti koncovému klientovi viackrát za sebou, následne posiela koncovej strane žiadosť CANCEL a iniciátorovi odpoveď 503 Service Unavailable, ktorá mu značí, že server je momentálne preťažený a nedokáže spracovať žiadosť (pozri nasledujúci obr.), v odpovedi v poli Retry-After môže uviesť, kedy klient môže žiadosť skúsiť poslať znova, avšak toto pole Yate neposiela. Vtedy to pre klienta znamená, akokeby dostal odpoveď 500 (Server Internal Error) a nesmie žiadosť posielať tomuto severu znova. [15] Následne celý proces cyklicky opakuje do skončenia testu. Z tohto chovania môžeme konštatovať, že takýto typ správy ústredňu znefunkční a nie je schopná ju spracovať, je však aspoň schopná klienta o tom informovať.



Obr. 6.33: Znefunkčnenie Yate po prijatí modifikovanej INVITE

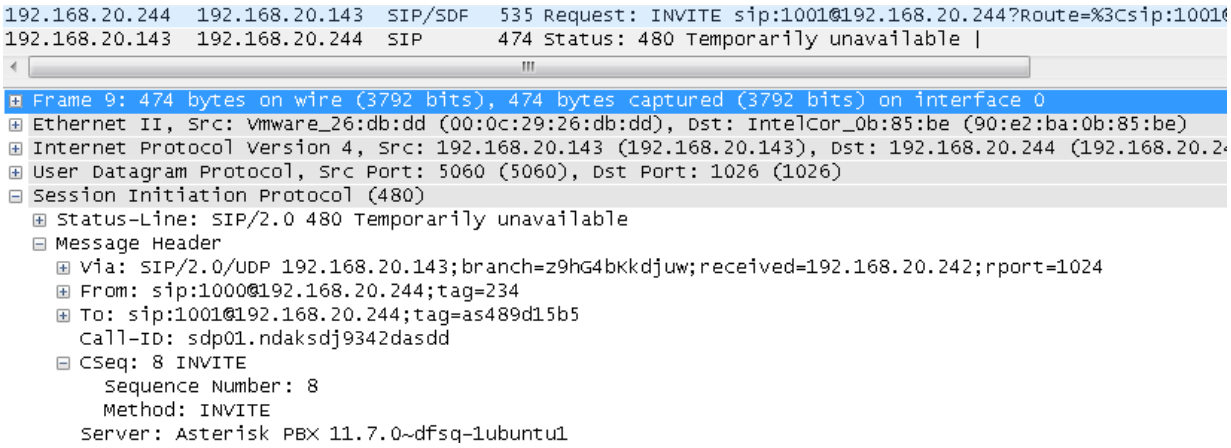
Kamailio žiadosť neprepošle, avšak tentoraz správne použije všeobecnú informatívnu odpoveď 400, so špecifikovaním problému Content-Length mis-match. Opensips aj tento typ žiadosti vyhodnocuje nebezpečne a ďalšie spracovávanie komunikácie odmieta odpoveďou 403.

6.2.4 Zdeformovaná URI adresa (uniklé znaky)

Útok posiela SIP INVITE správu s deformovanou časťou URI adresy v hlavičke. Jedná sa o tzv. escaped headers tj. uniklé časti hlavičky. Konkrétne má tvar:

```
sip:1001@192.168.20.244?Route=%3Csip:1001@192.168.20.244%3E
```

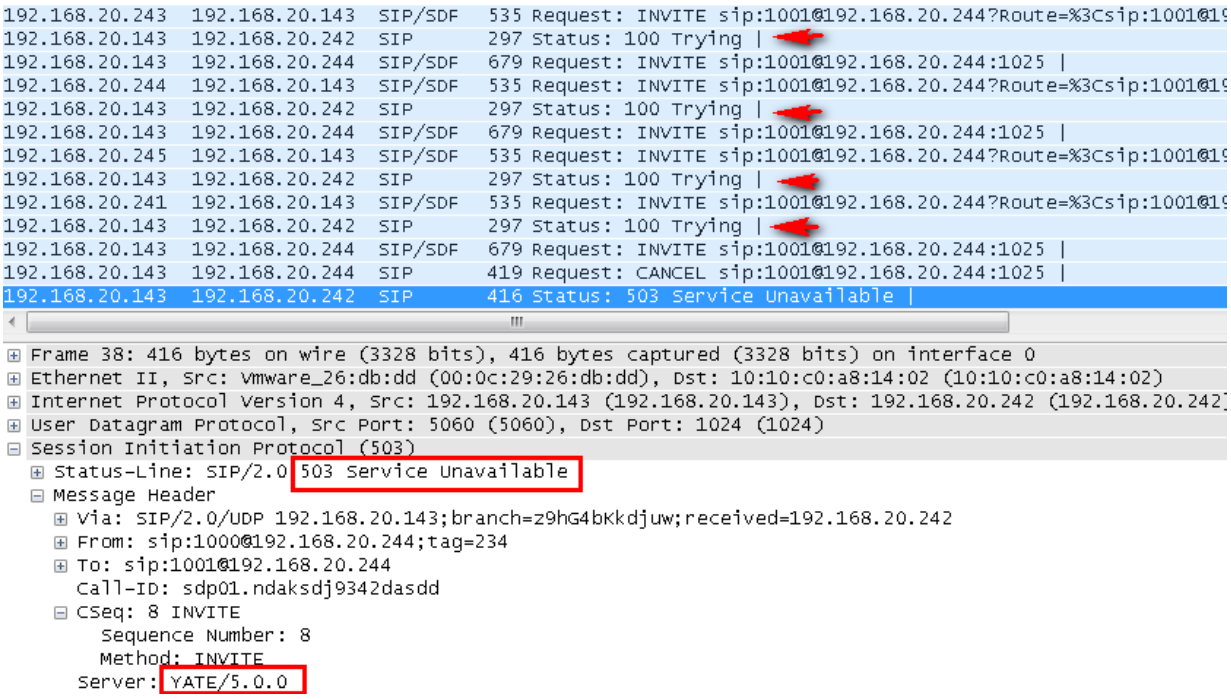
Asterisk reaguje odpoveďou 480 Temporarily unavailable, ktorá sa používa, ak server nevie kam by mal požiadavku smerovať [15]. Takáto odpoveď je rozumná, nakoľko takýto adresát neexistuje.



Obr. 6.34: Asterisk na žiadosť s uniknutými znakmi v URI reaguje odpoveďou 480

Freswitch odpovedá ako aj v útoku s nesprávnou hodnotou poľa *Date*, hláškou 400 Bad Session Description, ktorou značí, že nebolo možné žiadosť spracovať kvôli nesprávnemu syntaxu.

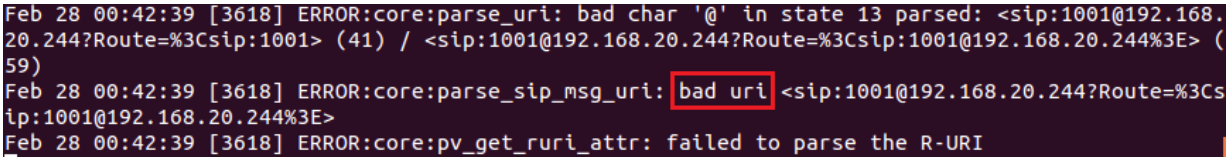
Ústredňa Yate sa snaží kontaktovať cyklicky druhú stranu, avšak pochopiteľne na neexistujúci URI adresu nemôže a tak to po čase bez odpovede ukončí správou CANCEL a iniciátorovi odpovedá cyklickým zaslielaním odpovede 503 ako aj v iných testoch.



Obr. 6.35: Pokusy Yate doručiť žiadosť na neexistujúcu URI adresu

Kamailio rozpozná chybnú hlavičku, žiadosť ďalej neposiela, ale odpovedá 400 Bad Request URI.

Ústredňa Opensips tajúto žiadosť očakávane nepreposiela ďalej, odpovedá 484 Address Incomplete, podobne ako v útoku s modifikovanou URI adresou u žiadosti OPTIONS. V logoch potvrdzuje viacerými chybovými hláškami, že takúto hlavičku nespracováva.



Obr. 6.36: Výpis logu Opensips v reakcii na žiadosť s uniknutými znakmi v URI

Obr. 6.37: Odpoveď 484 ústredňe Opensips na uniknuté znaky v URI

6.2.5 Zdeformovaná URI adresa (ohraničená < >)

Útok posiela SIP INVITE žiadosť s URI adresou ohraničenou znakmi < >, čo je nepovolené. Jej tvar je teda:

```
INVITE <sip:1001@192.168.20.244> SIP/2.0
```

Asterisk odpovedá rovnako ako v predchádzjúcom prípade a teda správou 480. Nedokáže tak cieľ rozpoznať, aby mu mohol požiadavku o zahájenie relácie smerovať, aj keď tentoraz tvar deformácie pôsobí zanedbateľne.

Freeswitch odpovedá takisto chybovou hláškou, u neho to je ale 400 Bad Request, čo je všeobecné označenie, presnejší dôvod chyby v odpovedi teda neuvádza.

```
192.168.20.244  192.168.20.143  SIP/SDF   513 Request: INVITE <sip:1001@192.168.20.244> |
192.168.20.143  192.168.20.244  SIP       306 Status: 400 Bad Request |
```

Obr. 6.38: Odpoveď Freeswitchu 400 Bad Request na nesprávnu adresu v žiadosti

Yate reaguje rovnako ako v predchádzajúcom teste. Po neúspešnom kontaktovaní druhej strany, posiela cyklicky odpoveď 503.

Ústredňa Kamailio odpovedá rovnako ako v predchádzajúcom teste. Na rozdiel od Freeswitchu už poskytuje presnejšie určenie chyby v hlavičke. Dokáže teda aj v debugovacom výpise upozorniť na nesprávnu adresu a odoslať inicátorovi žiadosti špecifickú odpoveď, ktorá bližšie určuje dôvod, tj. že sa jedná o chybnú URI adresu, pozri nasledujúce obrázky.

```
 7(3486) WARNING: sanity [sanity.c:243]: check_ruri_scheme(): failed to parse re
quest uri [<sip:1001@192.168.20.244>]
```

Obr. 6.39: Upozornenie v logu Kamailio na nesprávnu URI adresu v žiadosti

```
192.168.20.244  192.168.20.143  SIP/SDF   513 Request: INVITE <sip:1001@192.168.20.244>
192.168.20.143  192.168.20.244  SIP       367 Status: 400 Bad Request URI |
⊞ Frame 23: 367 bytes on wire (2936 bits), 367 bytes captured (2936 bits) on interface
⊞ Ethernet II, Src: Vmware_26:db:dd (00:0c:29:26:db:dd), Dst: 10:10:c0:a8:14:03 (10:10:
⊞ Internet Protocol Version 4, Src: 192.168.20.143 (192.168.20.143), Dst: 192.168.20.24
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol (400)
  ⊞ Status-Line: SIP/2.0 400 Bad Request URI
  ⊟ Message Header
    ⊞ To: sip:1001@192.168.20.244;tag=b27e1a1d33761e85846fc98f5f3a7e58.16d0
    ⊞ From: sip:1000@192.168.20.244;tag=39291
      Call-ID: sdp01.ndaksdj9342dasdd
    ⊟ CSeq: 8 INVITE
        Sequence Number: 8
        Method: INVITE
    ⊞ Via: SIP/2.0/UDP 192.168.20.143;branch=z9hG4bKkdjuw;received=192.168.20.243
      Server: kamailio (4.2.0 (x86_64/linux))
```

Obr. 6.40: Odpoveď Kamailio na nesprávnu adresu v žiadosti

Posledná ústredňa reagovala odpoveďou 484 Address Incomplete, čím potvrdzuje, že typy útokov s modifikovanou adresou v žiadosti má ošetrené a nedovolí, aby jej spôsobili komplikácie.

```
192.168.20.244  192.168.20.143  SIP/SDF   513 Request: INVITE <sip:1001@192.168.20.244> |
192.168.20.143  192.168.20.244  SIP       377 Status: 484 Address Incomplete |
```

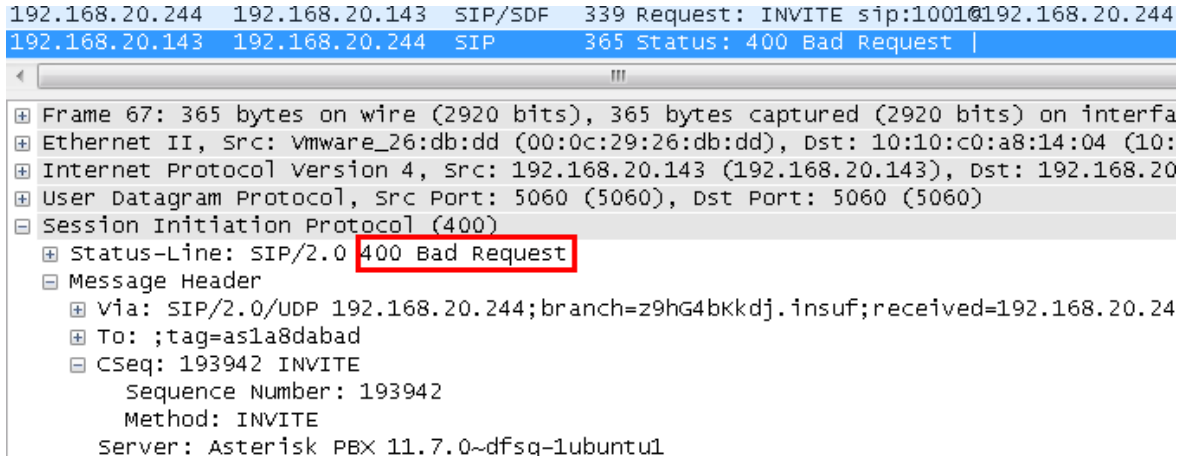
Obr. 6.41: Odpoveď 484 Opensips na modifikovanú adresu

6.2.6 Chýbajúce povinné polia v hlavičke žiadosti

Tento útok posiela INVITE žiadosť s chýbajúcimi poliami *To:* (na logickú identifikáciu cieľa), *From*:, (ktoré má informovať druhú stranu o zdroji, odkiaľ žiadosť o začatie relácie prišla), *Max-Forwards:* (maximálny počet skokov) a *Contact:* (adresa odosielateľa žiadosti z pohľadu ústredne). Tieto polia sú podľa RFC povinné. Tvar tejto žiadosti je teda v našom testovaní nasledovný:

```
INVITE sip:1001@192.168.20.244 SIP/2.0
CSeq: 193942 INVITE
Via: SIP/2.0/UDP 192.168.20.244;branch=z9hG4bKkdj.insuf
Content-Type: application/sdp
Content-Length: 152
```

Ústredňa Asterisk žiadosť očakávane nedokáže preposlať, v logu to značí chybovými hláškami ako *No field 'From'*. Iniciátorovi, (ktorého poznačí na koniec riadku pola *Via:*) odpovedá všeobecnou chybovou odpoveďou 400 Bad Request a bližšie podrobnosti neurčuje.



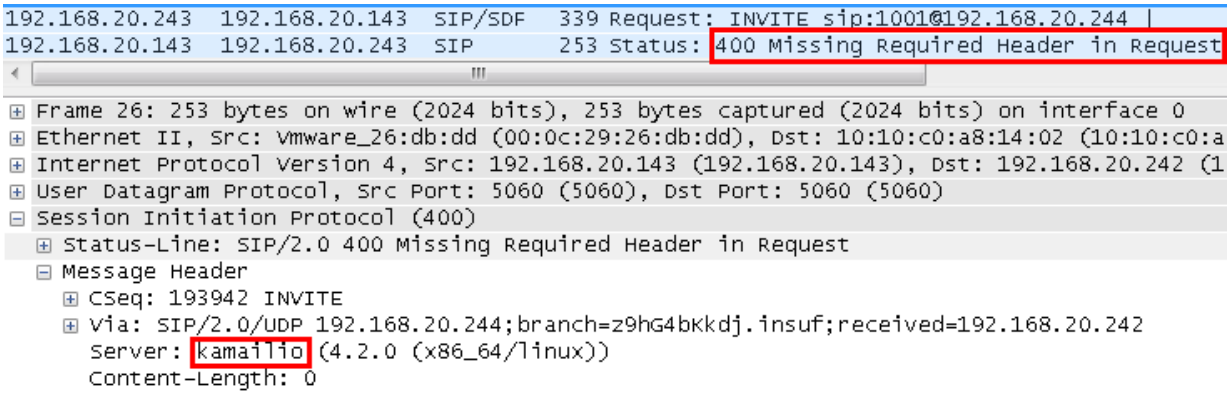
Obr. 6.42: Odpoveď 400 od Asterisku na žiadosť bez povinných polí

Freeswitch a Yate odpovedajú rovnako ako Asterisk, správou 400 Bad Request (pozri obr.).

```
------
20150228150749.122720 <sip:INFO> 'udp:0.0.0.0:5060' sending code 400 0x7ff2d80008c0 to 192.168.
20.245:1042 [0x7ff30a4d77b0]
------
SIP/2.0 400 Bad Request
Via: SIP/2.0/UDP 192.168.20.244;branch=z9hG4bKkdj.insuf;received=192.168.20.245
CSeq: 193942 INVITE
Server: YATE/5.0.0
Contact: <sip:1001@192.168.20.143:5060>
Allow: ACK, INVITE, BYE, CANCEL, REGISTER, REFER, OPTIONS, INFO
Content-Length: 0
```

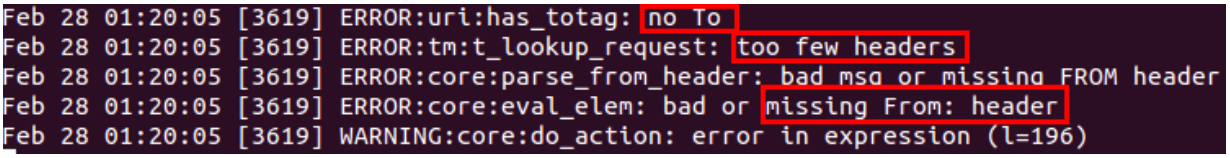
Obr. 6.43: Odpoveď 400 od Yate na žiadosť bez povinných polí

Kamailio na túto žiadosť reaguje chybovou hláškou 400, konkrétne Missing Required Header in Request, ktorou pomerne presne určuje odosielateľovi dôvod zamietnutia. Aj keď v tomto prípade chýba viacero povinných polí, odpoveď neurčuje ktoré presne.



Obr. 6.44: Odpoveď 400 od Kamailio na žiadosť bez povinných polí

Opensips reaguje znova zákazom 403. V debugovacom režime sú podrobne diagnostikované chybné časti, pozri obr.



Obr. 6.45: Výpis z debugovacieho režimu Opensips na žiadosť bez povinných polí

6.2.7 Plne zdeformovaná žiadosť

Útok posiela úplne zdeformovanú "trýzniacu" žiadosť podľa tzv. „torture“ testu z IETF konceptu RFC 4475. Jedná sa o žiadosť INVITE, ktorá je maximálne neštandardná. Obsahuje nesprávne riadkovanie, uniklé znaky, prázdne a neznáme polia v hlavičke, neznáme parametre i poradie a viacero iných modifikácii, ktoré sú však technicky legálne. [41] Jej forma je nasledovná:

```
INVITE sip:1001@192.168.20.244;unknownparam SIP/2.0
TO :
 sip:1001@192.168.20.244;   tag    = 1918181833n
from   : "J Rosenberg \\\""       <sip:1000@192.168.20.244>
  ;
  tag = 98asjd8
MaX-fOrWaRdS: 0068
Call-ID: wsinv.ndaksdj@192.168.20.244
Content-Length   : 150
cseq: 0009
  INVITE
Via  : SIP  /   2.0
 /UDP
    192.168.20.143;branch=390skdjuw
s :
NewFangledHeader:   newfangled value
 continued newfangled value
UnknownHeaderWithUnusualValue: ;;,,;;,;
Content-Type: application/sdp
Route:
 <sip:192.168.20.143;lr;unknownwith=value;unknown-no-value>
v:  SIP  / 2.0  / TCP     192.168.20.143   ;
  branch  =   z9hG4bK9ikj8  ,
 SIP  /    2.0   / UDP  192.168.20.143   ; branch=
 z9hG4bK30239
m:"Quoted string \"\"" <sip:192.168.20.244> ; newparam =
      newvalue ;
  secondparam ; q = 0.33
```

Asterisk na takúto žiadosť vôbec nereagoval. Neodoslal teda žiadnu odpoveď. V jeho logu je vidieť, že žiadosť prijal, no zároveň odignoroval.

Freeswitch prekvapujúco na takúto žiadosť reaguje znova klasickým 400 Bad Request. Nepracuje s ňou ďalej, ale dokáže teda ešte aj odosielateľa informovať, že takúto žiadosť neakceptuje. Dokonca rozpozná zo žiadosti odosielateľa, zachová Call-ID a tri polia Via, ktoré sú v žiadosti v rozbitom tvare (pozri obr. z logu).

```
   ------------------------------------------------------------------------
send 392 bytes to udp/[192.168.20.243]:5060 at 05:50:11.865412:
   ------------------------------------------------------------------------
   SIP/2.0 400 Bad Request
   Via: SIP/2.0/UDP 192.168.20.143;branch=390skdjuw;received=192.168.20.243
   Via: SIP/2.0/TCP 192.168.20.143;branch=z9hG4bK9ikj8
   Via: SIP/2.0/UDP 192.168.20.143;branch=z9hG4bK30239
   From: "J Rosenberg \\\"" <sip:1000@192.168.20.244>;tag=98asjd8
   To: <sip:1001@192.168.20.244>;tag=1918181833n
   Call-ID: wsinv.ndaksdj@192.168.20.244
   CSeq: 9 INVITE
   Content-Length: 0
```

Obr. 6.46: Odpoveď vytvorená Freeswtichom na zdeformovanú INVITE

Ústredňa Yate najprv zareaguje bežnou odpoveďou 100 Trying, ktorou značí odosielateľovi žiadosti, že bude kontaktovať cieľ, avšak následne žiadosť vyhodnotí ako nesprávnu a zasiela odpoveď 481 Call/Transaction Does Not Exist. Tá sa používa, aby UAS mohol informovať klienta o obdržaní žiadosti, na ktorú nepozná existujúcu reláciu. Keďže relácia ešte nebola vytvorená, tak pochopiteľne ani nemôže existovať. Dá sa konštatovať, že vhodnejšia by bola iná informatívna správa, ale už len samotný fakt, že odpoveď dokáže vytvoriť a odoslať, je pri takto deformovanej žiadosti pozitívny.

```
192.168.20.242  192.168.20.143  SIP/SDP   978 Request: INVITE sip:1001@192.168.20.244;unknownparam |
192.168.20.143  192.168.20.242  SIP       478 Status: 100 Trying |
192.168.20.143  192.168.20.242  SIP       609 Status: 481 Call/Transaction Does Not Exist |
⊞ Frame 24: 609 bytes on wire (4872 bits), 609 bytes captured (4872 bits) on interface 0
⊞ Ethernet II, Src: Vmware_26:db:dd (00:0c:29:26:db:dd), Dst: 10:10:c0:a8:14:02 (10:10:c0:a8:14:02)
⊞ Internet Protocol Version 4, Src: 192.168.20.143 (192.168.20.143), Dst: 192.168.20.242 (192.168.2
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 1024 (1024)
⊟ Session Initiation Protocol (481)
  ⊟ Status-Line: SIP/2.0 481 Call/Transaction Does Not Exist
      Status-Code: 481
      [Resent Packet: False]
      [Request Frame: 22]
      [Response Time (ms): 7]
  ⊟ Message Header
    ⊞ Via: SIP  /   2.0/UDP192.168.20.143;branch=390skdjuw;received=192.168.20.242
    ⊞ Via: SIP  / 2.0  / TCP     192.168.20.143;branch=z9hG4bK9ikj8  ,SIP  /    2.0   / UDP  192.1
    ⊞ from: "J Rosenberg \\\""       <sip:1000@192.168.20.244>;tag = 98asjd8
    ⊞ TO: sip:1001@192.168.20.244;tag=1918181833n
      Call-ID: wsinv.ndaksdj@192.168.20.244
    ⊞ cseq: 0009INVITE
      Server: YATE/5.0.0
    ⊞ Contact: <sip:1001@192.168.20.143:5060>
```

Obr. 6.47: Odpoveď Yate na zdeformovanú INVITE

Kamailio na takúto žiadosť nereaguje. V logu je však vidieť, že registruje chybné časti v hlavičke a v nich je vidieť, že nerozpozná ani použitú metódu (pozri obr.).

```
 7(3272) ERROR: <core> [receive.c:149]: receive_msg(): core parsing of SIP messa
ge failed (192.168.20.243:1025/1)
 5(3270) ERROR: <core> [parser/parse_cseq.c:65]: parse_cseq(): ERROR:parse_cseq:
 no method found
 5(3270) ERROR: <core> [parser/parse_cseq.c:98]: parse_cseq(): ERROR: parse_cseq
: bad cseq
 5(3270) ERROR: <core> [parser/msg_parser.c:161]: get_hdr_field(): ERROR: get_hd
r_field: bad cseq
 5(3270) ERROR: <core> [parser/msg_parser.c:705]: parse_msg(): ERROR: parse_msg:
 message=<INVITE sip:1001@192.168.20.244;unknownparam SIP/2.0
TO :
```

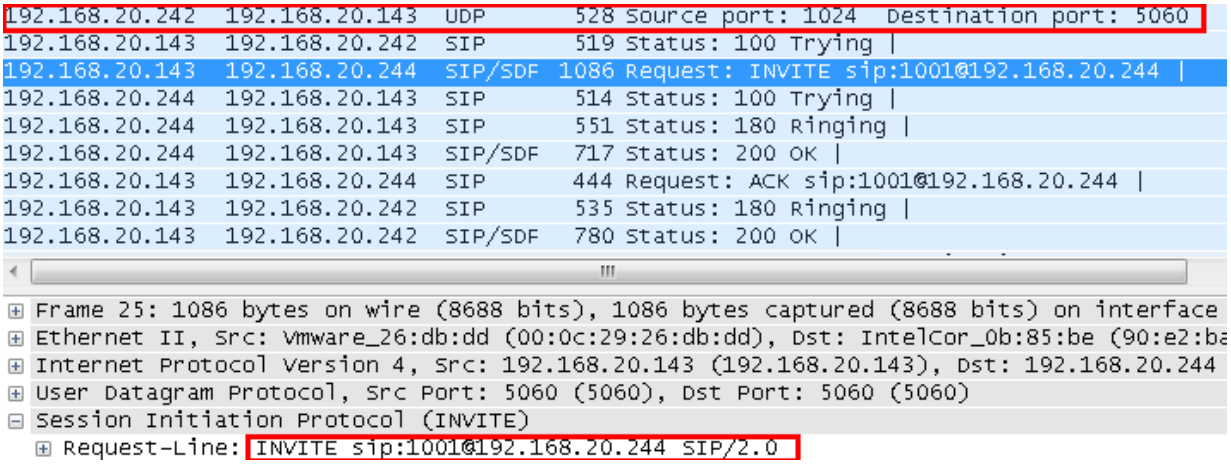
Obr. 6.48: Výpis logu Kamailio ako reakcia na zdeformovanú INVITE, ktorú ani nerozpozná

Opensips znova vo výpise podrobne rozpoznáva chybné časti hlavičky a sadu chybových hlásení ukončuje hláškou *Unable to parse msg received*, čo značí neschopnosť žiadosť rozanalyzovať. Iniciátorovi však nedokáže poslať odpoveď.

6.2.8 *Medzery v Request-Line*

Útok je podobný ako s OPTIONS v podkapitole 6.1.7, ale tu sa jedná o medzery medzi elementami, tj. konkrétne pred a po adrese URI. Toto pole je prvé v celej hlavičke žiadosti a adresa v ňom býva zhodná s cieľom v poli *To*. [15]

Asterisk reaguje už typicky liberálne. Žiadosť upraví a prepošle koncovému užívateľovi. Nasleduje výmena správ k zahájeniu relácie. Wireshark žiadosť nedokáže analyzovať ako správu INVITE protokolu SIP, zobrazí ju v podobe UDP. Výmenu správ je vidieť na obrázku.

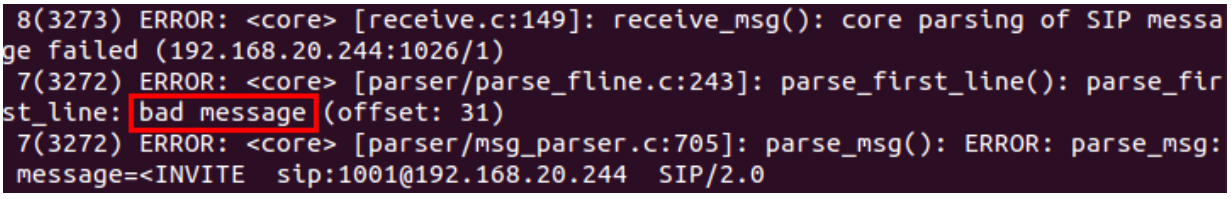


Obr. 6.49: Zahájenie relácie u Asterisku, ktorý pôvodnú žiadosť upraví do správneho tvaru

Ústredňa Freeswitch odpovedá chybovou odpoveďou 400 Bad Request, bližší dôvod chyby však neuvádza.

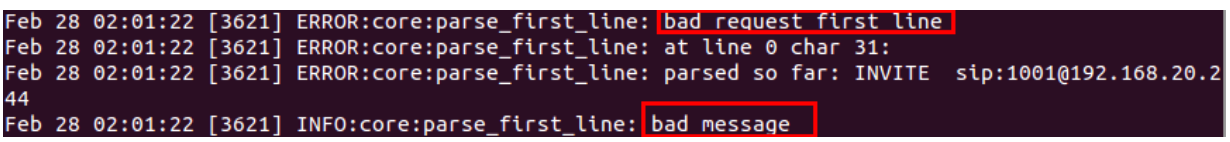
Yate odpovie prvotným 100 Trying, žiadosť prepošle s opraveným tvarom Request-Line a je schopná smerovať reláciu. Môžeme konštatovať, že reaguje štandardne ako v podobnom teste so správou OPTIONS, pozri kapitola 6.1.7, a že takáto správa jej teda žiaden problém nespôsobuje ako v niektorých predchádzajúcich testoch.

Kamailio na žiadosť nereaguje, avšak chybné správy z logu diagnostikujú problém, kde mu vadí prvý riadok žiadosti ako je vidieť na obrázku.



Obr. 6.50: Kamailio chybný prvý riadok rozpozná

Opensips na takúto žiadosť vôbec neodpovedá. V logu je vidieť, že chybnú Request-Line rozpozná, ale nedokáže ju rozanalyzovať a pracovať s ňou ďalej.



Obr. 6.51: Opensips takisto chybný prvý riadok rozpozná

6.2.9 *Neznámy Content-Type*

Útok posiela modifikovanú SIP INVITE s neznámym typom tela správy, tzv. Content-Type. Toto pole udáva akú formu majú dáta nesené v záťaži. Typicky to je typ:

```
application/sdp
```

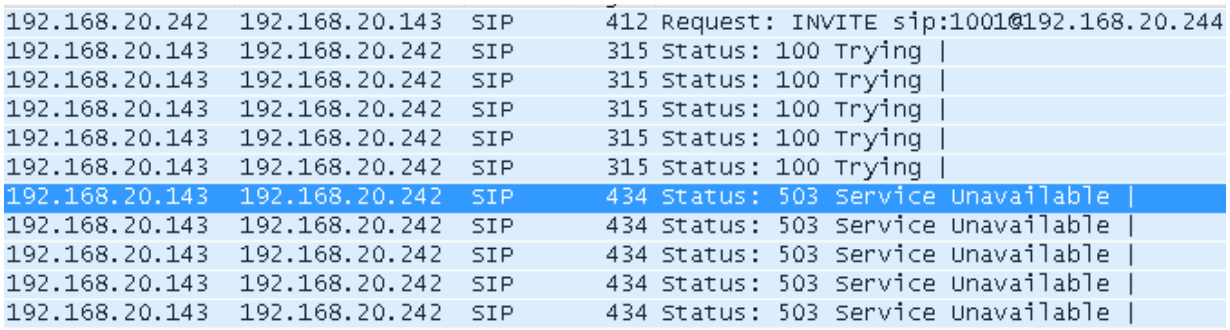
V tomto prípade to bude:

```
application/unknownformat
```

Ústredňa Asterisk takýto formát sama pri preposlaní druhej strane upraví na známy tvar a následne realizuje počiatočnú výmenu správ k tvorbe relácie.

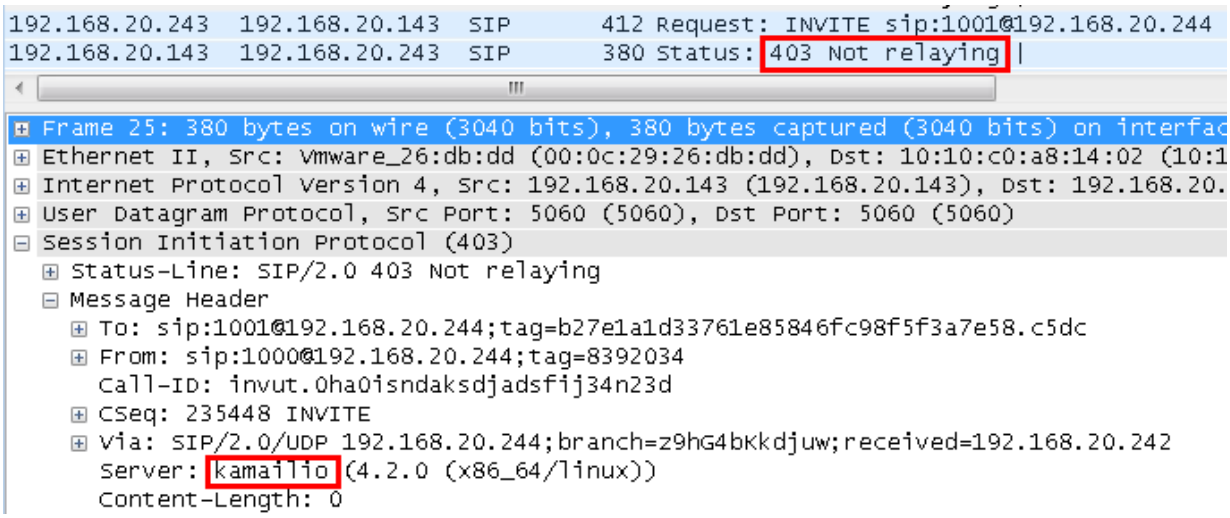
Freeswitch reaguje všeobecnou chybovou odpoveďou 400 Bad Request, bližší dôvod chyby neuvádza.

Yate žiadosť prijme, obratom odpovie 100 Trying. Následne pošle druhému klientovi žiadosť, v ktorej upraví *Content-Type* na „application/sdp“ a zmení aj telo hlavičky. Vytvorí vlastnú modifkáciu podobne ako v testoch na začiatku kapitoly. Na takúto žiadosť však nedostane odpoveď a celý proces cyklicky opakuje až do ukončenia útoku. Sled správ medzi ústredňou a odosielateľom modifikovanej žiadosti je na obrázku.



Obr. 6.52: Cyklická reakcia Yate v teste s neznámym Content-Type

Kamailio reaguje odpoveďou 403 Not relaying, čím dáva jasne najavo, že takúto žiadosť nespracováva ďalej. Opensips rovnako takúto žiadosť ignoruje a reaguje znova odpoveďou - zákazom 403 Rely forbiden.



Obr. 6.53: Reakcia Kamailio na žiadosť s neznámym Content-Type

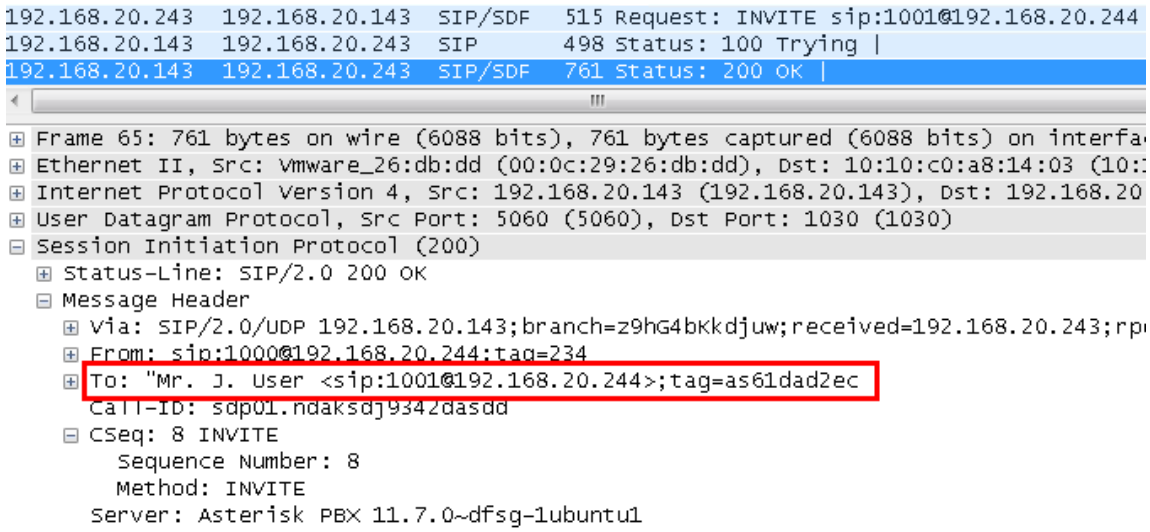
6.2.10 *Chýbajúce úvodzovky v mene volajúceho*

Útok posiela SIP INVITE žiadosť s menom volajúceho (display name) v poli *To*, ktoré obsahuje úvodzovky na začiatku mena, ale chýbajú na konci. Tvar je teda:

```
To: "Mr. J. User <sip:1001@192.168.20.244>
```

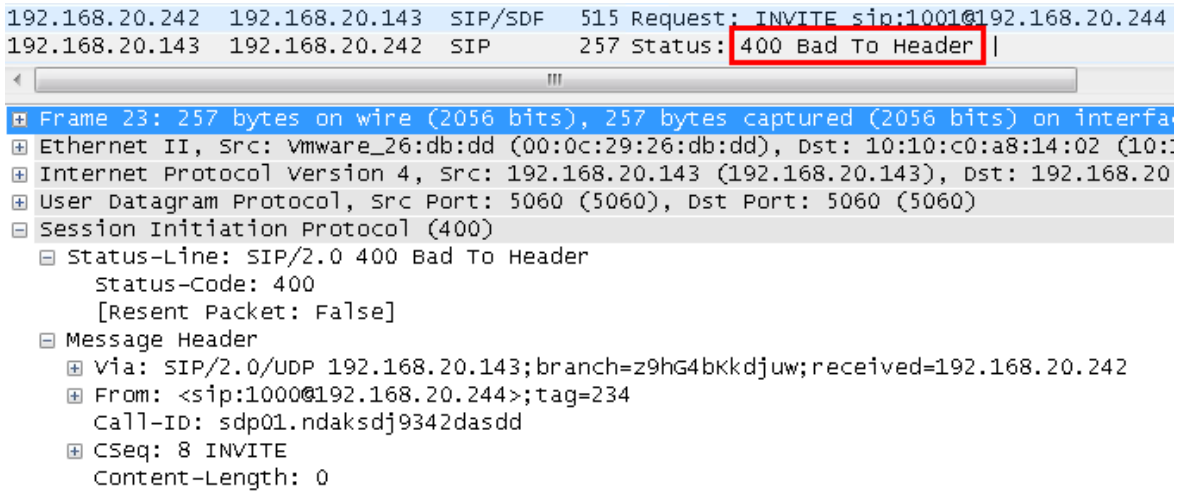
Nie je to validne, ale ústredňa môže tento tvar rozpoznať alebo aj upraviť. Pretože to nie je bežné, môže ju to aj pomýliť alebo zhodiť. [41]

Asterisk na túto žiadosť reaguje bez problémov. Preposiela ju druhej strane a odpovede preposiela inicátorovi. Následuje štandardná realizácia spojenia medzi obidvoma klapkami.



Obr. 6.54: Kladná odpoveď Asterisku na žiadosť s chýbajúcimi úvodzovkami v mene

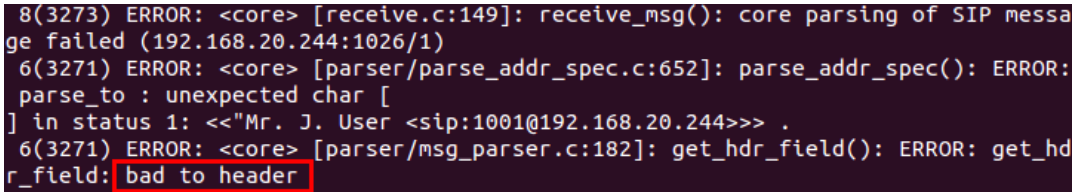
Ústredňa Freeswitch modifikovanú žiadosť rozpozná a odpovedá špecifickou chybovou správou 400 Bad To Header, ktorou značí, že pole *To* má neakceptovateľný tvar.



Obr. 6.55: Odpoveď Freeswitchu typu 400 na žiadosť s chýbajúcimi úvodzovkami v mene

Yate sa reaguje ako v predošlom teste, kde dôjde k modifkácii žiadosti, kde síce doručovateľa rozpozná avšak jeho alias nepoužije, zmení aj telo hlavičky. Celý proces u nej končí znova cyklickým odosielaním správy 503 iniciátorovi relácie.

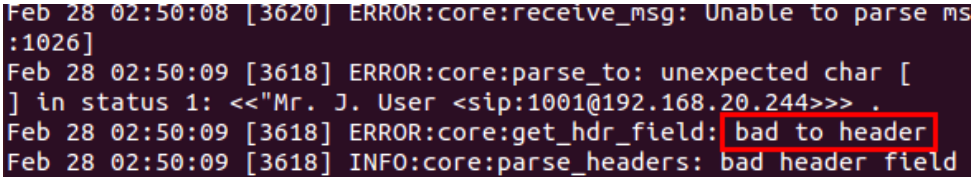
Kamailio nereaguje, v logu ale vypisuje chybové hlásenie, kde mu vadí pole *To* (pozri nasledujúci obr.)



Obr. 6.56: Výpis logu v Kamailio na žiadosť s chýbajúcimi úvodzovkami v mene

Opensips na takúto žiadosť takisto vôbec nereagoval. V logu je však opakovane vidieť sériu chybových hlásení (pozri nasledujúci obr.), kde mu vadí chybné pole *To*.

Z logovacíh správ u obidvoch týchto ústrední je vidieť ich veľká podobnosť aj v syntaxe (pravdepodobne vyplývajúca z identickej minulosti celého projektu, pozri kapitoly 1.4 a 1.5).



Obr. 6.57: Výpis logu v Opensips na žiadosť s chýbajúcimi úvodzovkami v mene

6.3 Modifikované SIP REGISTER žiadosti

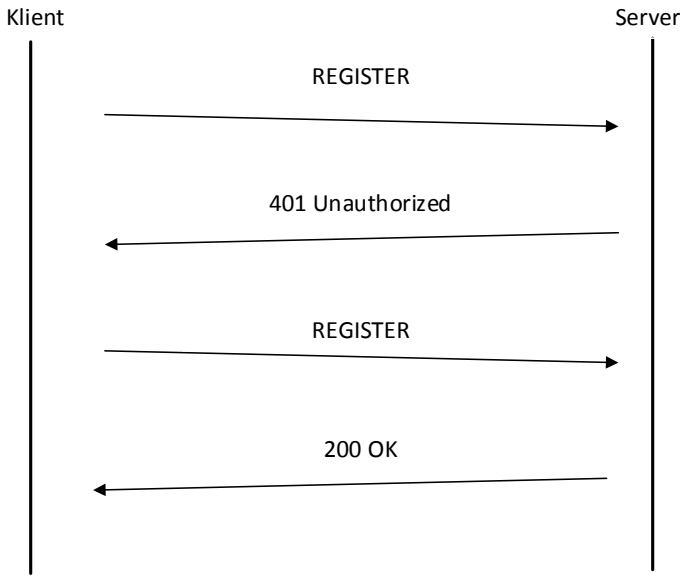
Táto časť testovania sa zameriava na útoky s využitím žiadosti REGISTER. Registrácia vytvára vzťahy v lokačnej službe pre konkrétnu doménu, ktorá asociuje adresu daného záznamu v URI tvare s jednou alebo viacerými kontaktnými adresami (v poli *Contact*). To znamená, že ak proxy server obdrží žiadosť pre určitú doménu, kde sa adresa žiadateľa (*Request-URI*) zhoduje s adresou v záznamoch, server ju bude smerovať na kontaktnú adresu registrovanú na daný záznam.

Registrácia zahrňuje posielanie tohto typu žiadosti tzv. registrar serveru (pozri kap. 2.2.2). Ten vystupuje v prvej línii pre lokačnú službu danej domény, čítajúc a zapisujúc záznamy podľa obsahu príchodzích REGISTER správ. Lokačná služba spolupracuje s proxy serverom, ktorý zodpovedá za smerovanie v danej doméne. Proxy aj registrar sú logické role, ktoré sú typicky implementované v ústredni, spolu ešte s ďalšími. [15]

Žiadosť musí obsahovať tieto polia v hlavičke: Request-URI, To, From, Call-ID, Cseq, Contact. Príkladom takejto žiadosti je :

```
REGISTER sip:192.168.20.143 SIP/2.0
To: sip:1000@192.168.20.143
From: sip:1000@192.168.20.143;tag=43251j3j324
Call-ID: dblreq.0ha0isndaksdj99
Contact: sip:1000@192.168.20.244
CSeq: 8 REGISTER
Max-Forwards: 8
Via: SIP/2.0/UDP 192.168.20.244;branch=z9hG4bKkdjuw23492
```

Proces registrácie prebieha cez autentizáciu. Odpoveďou na žiadosť býva správa 401 Unauthorized, pokiaľ server nemá vypnutú autentifikáciu, (vtedy klienta zaregistruje do databázy a zašle potvrdenie 200 OK). V odpovedi 401 zasiela údaje potrebné pre klienta k autentizácii. Klient ich následne podľa HTTP Digest autentizačného princípu (kapitola 3.5.1) použije k autentizácii. Schéma priebehu registrácie je na obrázku nižšie.



Obr. 6.58: Proces registrácie [15]

V testoch je skúmaná reakcia ústredňe na modifikovanú REGISTER žiadosť. Tá je rovnako ako v iných testoch posielaná zo Spirentu z rozsahu IP 192.168.20.242-45. Ústredňa s adresou 192.168.20.143 by mala, ak jej správa nevadí odpovedať 401 Unauthorized, inak inou chybovou správou, kde definuje vadnú časť žiadosti alebo dôvod. Viacero útokov je orientovaných na u tejto metódy dôležité pole Contact, používané v lokačnej službe ústredne.

Rizikom môže byť ak ústredňa modifikovanú žiadosť nezamietne špecifickou správou, ale ju prijme a reaguje na ňu autentifikačnou odpoveďou. Klient sa totiž s danou hodnotou spätne zaregistruje. A ak sa jednalo o úmyselne modifikovanú žiadosť môže dôjsť k narušeniu bezpečnosti alebo aj funkcionality. Na druhej strane moc prísne reakcie, ktoré žiadosť zamietajú, aj keď sa jednalo o neúmyselnú chybu v žiadosti. tj. nie o útok, znemožnia komunikáciu a vytvorenie relácie k potencionálnemu hovoru.

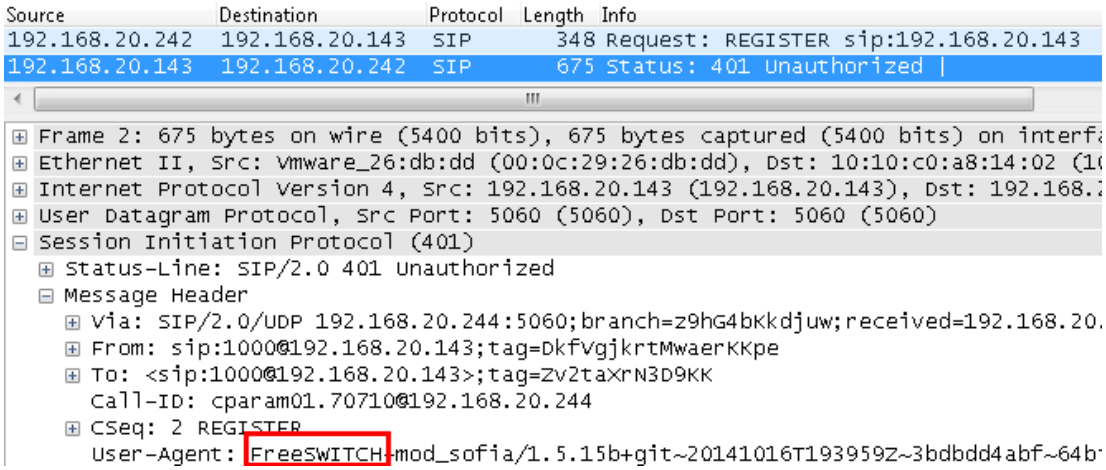
6.3.1 Neznámy parameter v poli Contact

Útok posiela SIP REGISTER správu s neznámym parametrom v hlavičke v poli *Contact*. Je to povolené, ale môže to ústredňu pomýliť. [41] Ako bolo spomenuté v úvode, toto pole sa používa k indikácii volanému, na akej adrese môže kontaktovať volajúceho.V teste má teda nasledovnú hondotu:

```
Contact: sip:1000@192.168.20.244;unknownparam
```

Za bodkočiarkou nasleduje parameter, najčastejšie býva používaný „expires“, ktorým sa definuje dĺžka platnosti registrácie. V tomto prípade sa jedná o neexistujúci typ. Odpoveď nesmie obsahovať túto hodnotu parametru. [41]

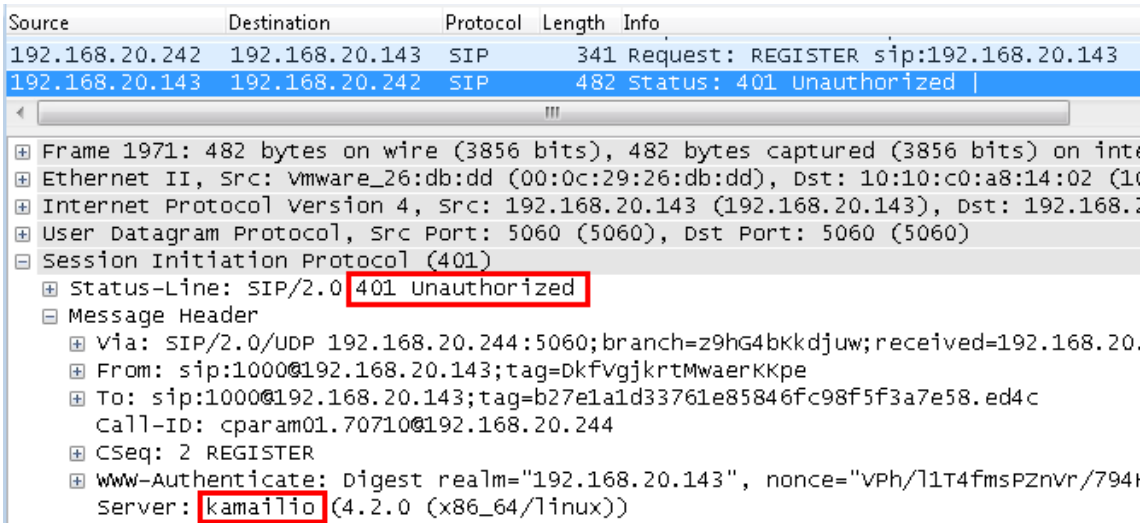
Ústredňa Asterisk odpovedá podľa štandardu správou 401. Neznámy parameter v hlavičke mu neprekáža. Takisto reaguje aj Freeswitch, pozri nasledujúci obrázok.



Obr. 6.59: Odpoveď Freeswitchu na žiadosť s neznámym parametrom

Pomýliť sa nenechá ani ústredňa Yate a reaguje rovnako ako predchádzajúce.

Kamailio a OpenSIPS sú na tom s reakciou takisto. Formát tejto odpovede u Kamamilio je na obrázku nižšie, dôležité je pole *WWW-Authenticate*, v ktorom zasiela údaje potrebné k autentizácii klienta.



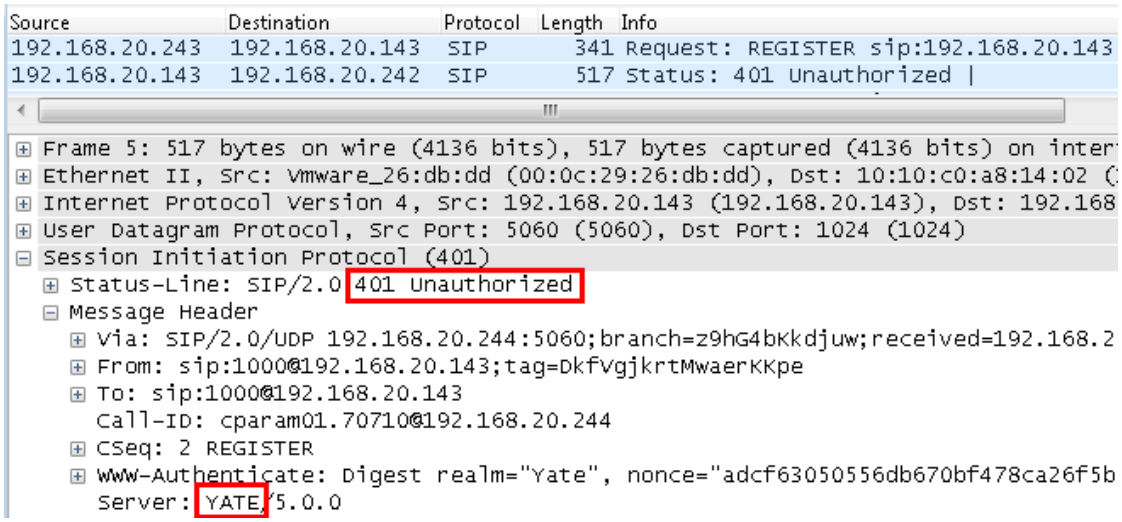
Obr. 6.60: Odpoveď Kamamilio na žiadosť s neznámym parametrom

6.3.2 Neznámy URI parameter

Útok posiela REGISTER žiadosť s neznámym parametrom v hodnote adresy URI v poli *Contact*. Tá ak je ohraničená ostrými zátvorkami, určuje, že parameter sa vzťahuje k adrese a nie k celej hlavičke. Špecifikácia [15] neznáme parametere nezakazuje, nie je to však bežné. Tvar adresy je teda:

```
Contact: <sip:1000@192.168.20.242;unknownparam>
```

Asterisk ani na neznámy parameter v adrese nereaguje inak, než štandardnou odpoveďou 401 Unauthorized. Rovnako reaguje aj ústredňa Freeswitch. Neznámy parameter neprekáža ani ústredni Yate, ktorá odpovedá rovnako, pozri nasledujúci obrázok.



Obr. 6.61: Odpoveď Yate na žiadosť s neznámym URI parametrom

Kamamilio a OpenSIPS reagujú takisto štandardnou 401 Unauthorized pokračujúc v inicializovanom procese registrácie klienta k serveru.

6.3.3 Uniklá hodnota v poli Contact

Útok posiela REGISTER žiadosť s uniklým polom *Route* (udáva, kam má byť správa smerovaná) v poli *Contact*. Je to valídne, ale neočakávané. Tvar pola *Contact* v hlavičke žiadosti je nasledovný:

```
Contact: <sip:1000@192.168.20.244?Route=%3Csip:sip.example.com%3E>
```

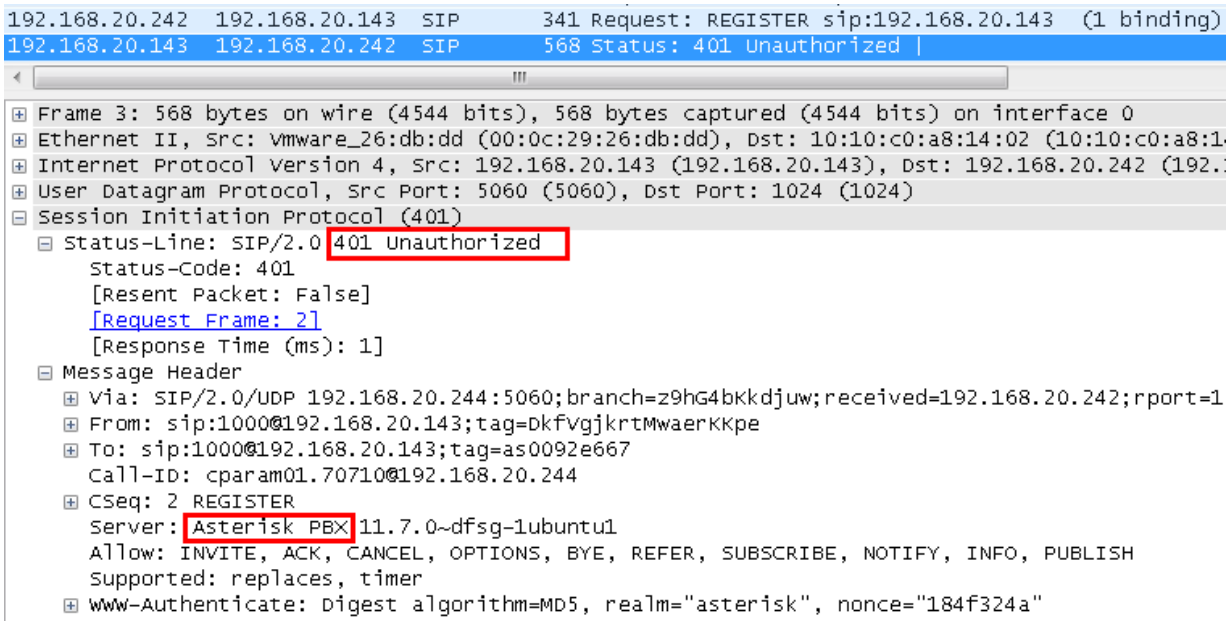
V tomto teste ragovali rovnako štandardnou odpoveďou 401 Unauthorized všetky ústredne. Potvrdzujúc tak validitu takejto formy a vlastnú schopnosť sa s takouto žiadosťou vysporiadať bez komplikácii a podľa odporučenia.

6.3.4 Neohraničená adresa

V tomto útoku je žiadosť REGSITER takisto posielaná s uniklou hodnotou v poli *Contact*, avšak neuzatvorená v ostrých zátvorkách. Je to nesprávne a neočakávané. Rozumná odpoveď na takúto žiadosť by mala byť 400 Bad Request, respektíve s bližšie špecifikovanou chybou. Druhou možnosťou je liberálny prístup, ak by si ústredňa zátvorky domyslela a odpovedala kladne. [41] Tvar pola v útoku je nasledovný:

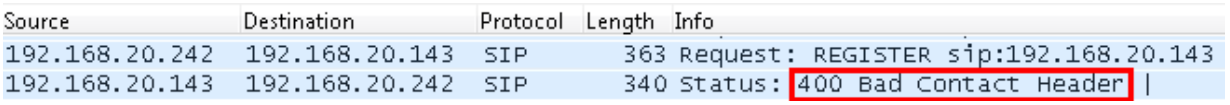
```
Contact: sip:1000@192.168.20.242?Route=%3Csip:sip.192.168.20.242%3E
```

Odpoveď Asterisku zastupuje druhú možnosť. Odpovedá pre neho typicky, pozri nasledujúci obrázok.



Obr. 6.62: Odpoveď Asterisku na žiadosť s neohraničenou URI

Freeswitch zastupuje prvú možnosť. Na takúto žiadosť odpovedá chybovou hláškou 400 Bad Contact Header, čím klientovi oznamuje, že danú žiadosť neprijíma kvôli chybnej časť *Contact* v hlavičke, pozri obr. 6.63.



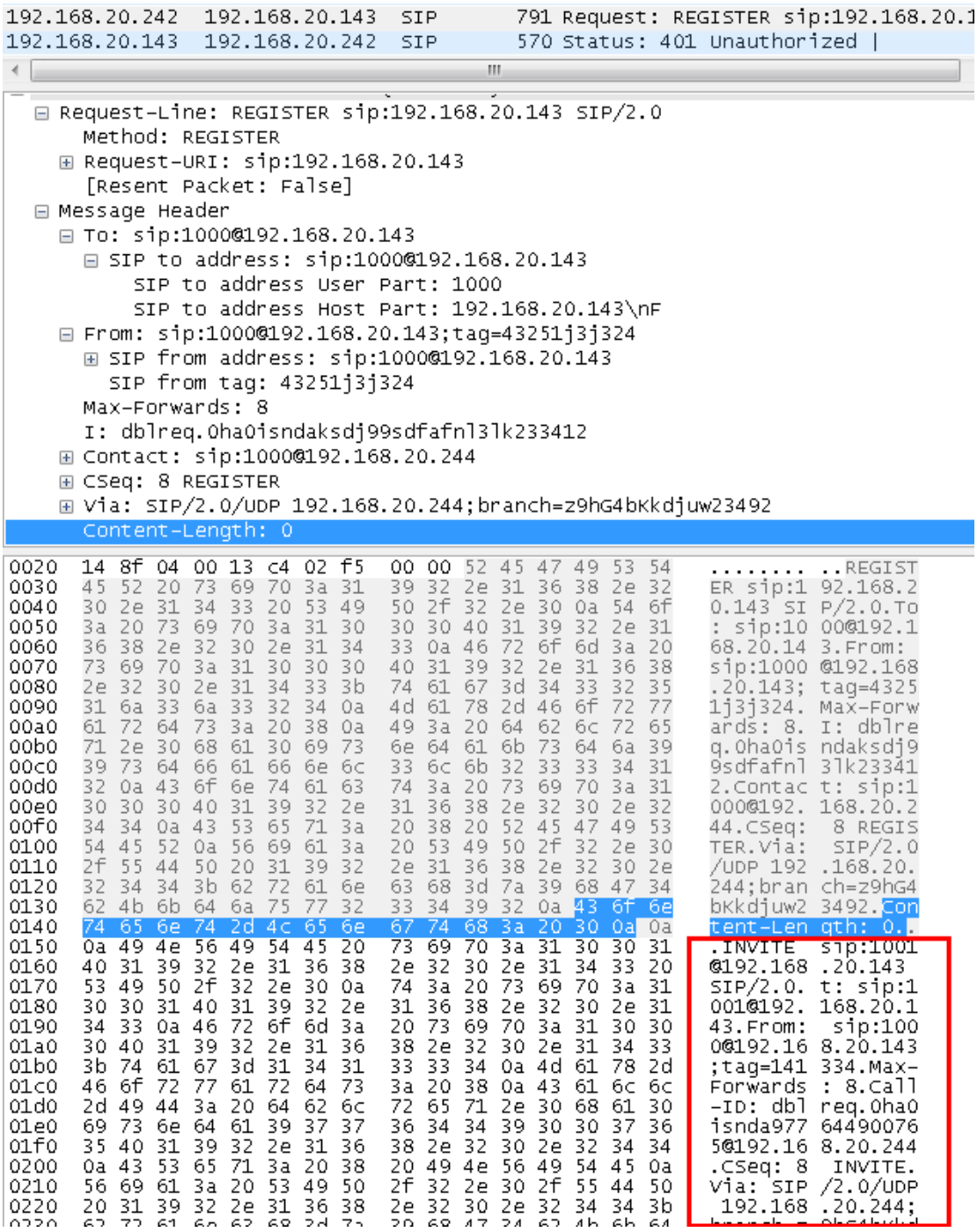
Obr. 6.63: Odpoveď Freeswitchu na žiadosť s neohraničenou URI

Yate odpovedá rovnako ako Asterisk správou 401 a pokračuje tak v procese registrácie klienta. Takisto liberálne reagujú Kamailio aj OpenSIPS a na žiadosť zasielajú výzvu k registrácii.

6.3.5 Neočakávaná druhá správa

Tento útok posiela SIP REGISTER správu s nulovou dĺžkou tela v hlavičke, avšak okamžite za ním nasleduje INVITE žiadosť. Je to síce povolené, ale žiadosť INVITE by mala byť odignorovaná. Ústredne sa môžu zachovať neočakávane, keďže tam, kde nasleduje normálna správa, by už mali dáta ignorovať.

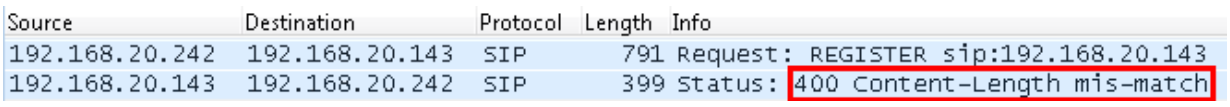
Asterisk reaguje podľa očakávaní, odpovedá len na žiadosť REGISTER (klasickou 401) a druhú žiadosť ignoruje, čo dokazuje tok z Wiresharku. Na obrázku je vidieť aj druhá žiadosť, ktorá nasleduje po poli *Content-Length*.



Obr. 6.64: Prenášaná INVITE hneď za REGISTER

Rovnako odpovedá aj Freeswitch. V debugovacom režime je vidieť, že správu INVITE ani nevidí.

Yate prekvapujúco takúto žiadosť zamieta, zašle iba všeobecnú odpoveď 400 Bad Request. Kamailio takisto reaguje odmietnutím žiadosti a vyslaním odpovede 400 Content-Length mis-match, ktorú poslal aj v iných testoch kde bol útok zameraný na toto pole (pozri 6.1.2 a 6.2.3). Tým sa potvrdzuje jeho silné zabezpečenie proti zraniteľnosti v tomto smere.



Obr. 6.65: Odpoveď Kamailio na žiadosť REGISTER s INVITE

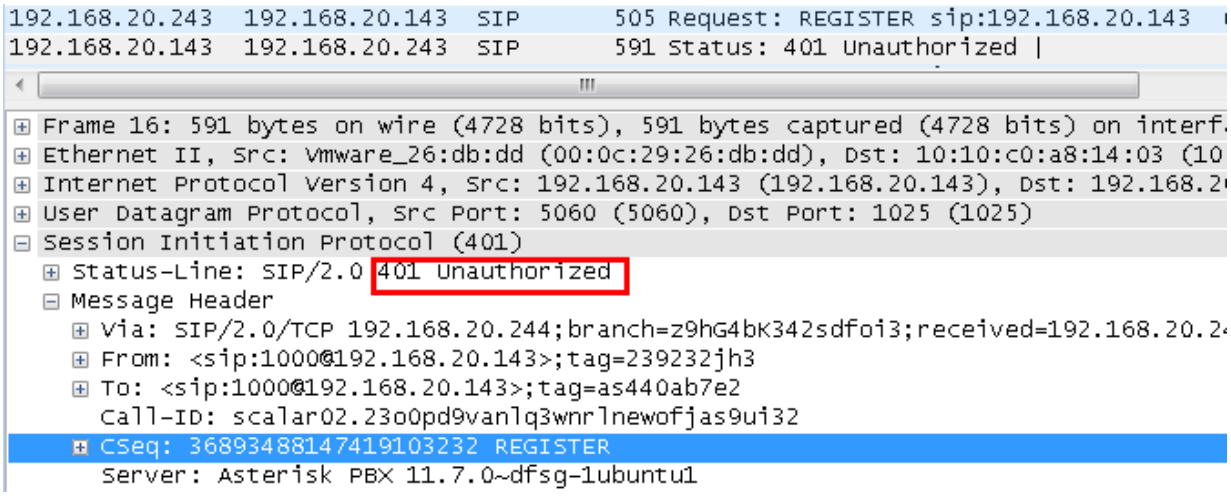
OpenSIPS zas na rozdiel od Kamailio reaguje prekvapujúco tak ako aj Asterisk a Freeswitch. Pokračuje v registračnom procese vyžiadaním autentifikácie od klienta správou 401.

6.3.6 *Hodnoty príliš veľké*

Tento útok obsahuje v REGISTER žiadosti niekoľko polí, ktorých hodnoty sú väčšie, než je povolené. Sú to *Cseq* (poradové číslo transakcie v rámci relácie s použitou metódou), *Max-Forwards* (identifikuje maximálny počet skokov a maximum je 255), *Expires* (udáva dobu životnosti správy) a parameter *expires* poľa *Contact* (udáva dobu platnosti žiadosti). [15] Takáto správa pochopiteľne nie je valídna a mala by byť zamietnutá chybovou hláškou 400 Bad Request, najmä kvôli hodnote *Cseq*. [41] Žiadosť má teda nasledovný tvar:

```
REGISTER sip:192.168.20.143 SIP/2.0
Via: SIP/2.0/TCP 192.168.20.244;branch=z9hG4bK342sdfoi3
To: <sip:1000@192.168.20.143>
From: <sip:1000@192.168.20.143>;tag=239232jh3
CSeq: 36893488147419103232 REGISTER
Call-ID: scalar02.23o0pd9vanlq3wnrlnewofjas9ui32
Max-Forwards: 300
Expires:
10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
Contact:
<sip:1000@192.168.20.244>;expires=280297596632815
Content-Length: 0
```

Ústredňa Asterisk však odpovedá neočakávane štandardnou odpoveďou 401 Unauthorized so zachovaným číslom sekvencie *Cseq*, pozri nasledujúci obrázok.



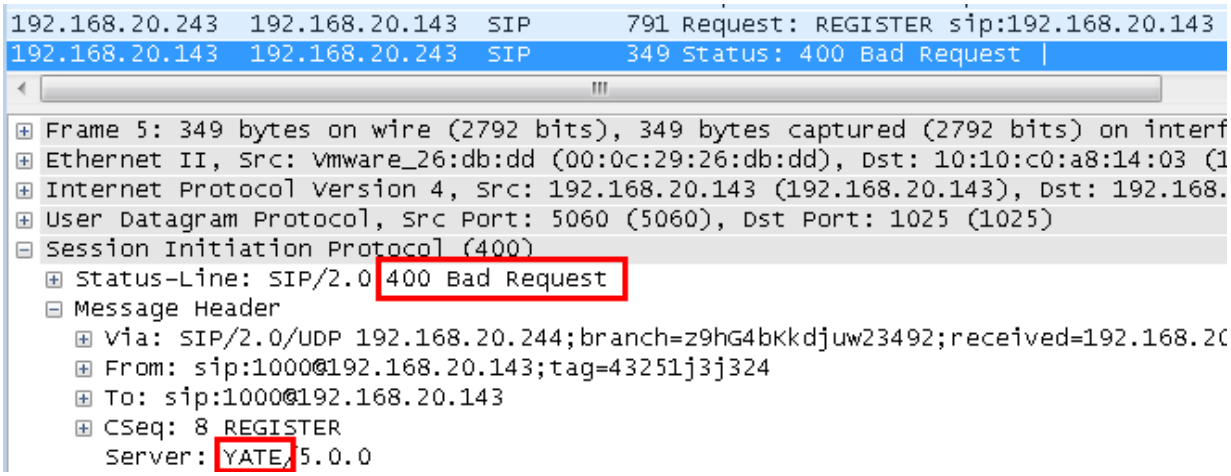
Obr. 6.66: Odpoveď Asterisku so zachovným vysokým Cseq

Freeswitch podľa správnosti žiadosť zamieta a odpovedá na ňu správou 400 Bad Cseq Header (pozri obr. 6.67), čím klientovi dáva jednoznačne najavo, kde je chyba. Pokiaľ klient neodošle žiadosť s akceptovateľnou hodnotou, Freeswitch k autentizácii nevyzve.

```
SIP/2.0 400 Bad CSeq Header
Via: SIP/2.0/TCP 192.168.20.244;branch=z9hG4bK342sdfoi3
From: <sip:1000@192.168.20.143>;tag=239232jh3
To: <sip:1000@192.168.20.143>;tag=tF4y1N43H1SpS
Call-ID: scalar02.23o0pd9vanlq3wnrlnewofjas9ui32
Content-Length: 0
```

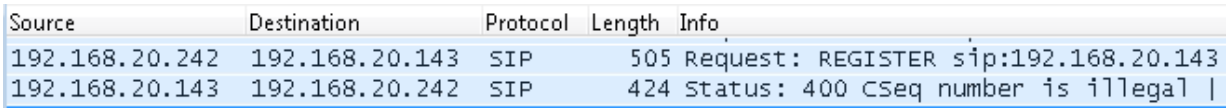
Obr. 6.67: Negatívna odpoveď Freeswitchu na žiadosť s vysokým Cseq

Yate reaguje rovnako ako v predchádzajúcom teste – správou 400 Bad Request. Jedná sa však o všeobecnú chybovú hlášku, ktorá by mala byť použitá k bližšej špecifikácii problému (napr. tak ako to uviedol Freeswitch). [15]



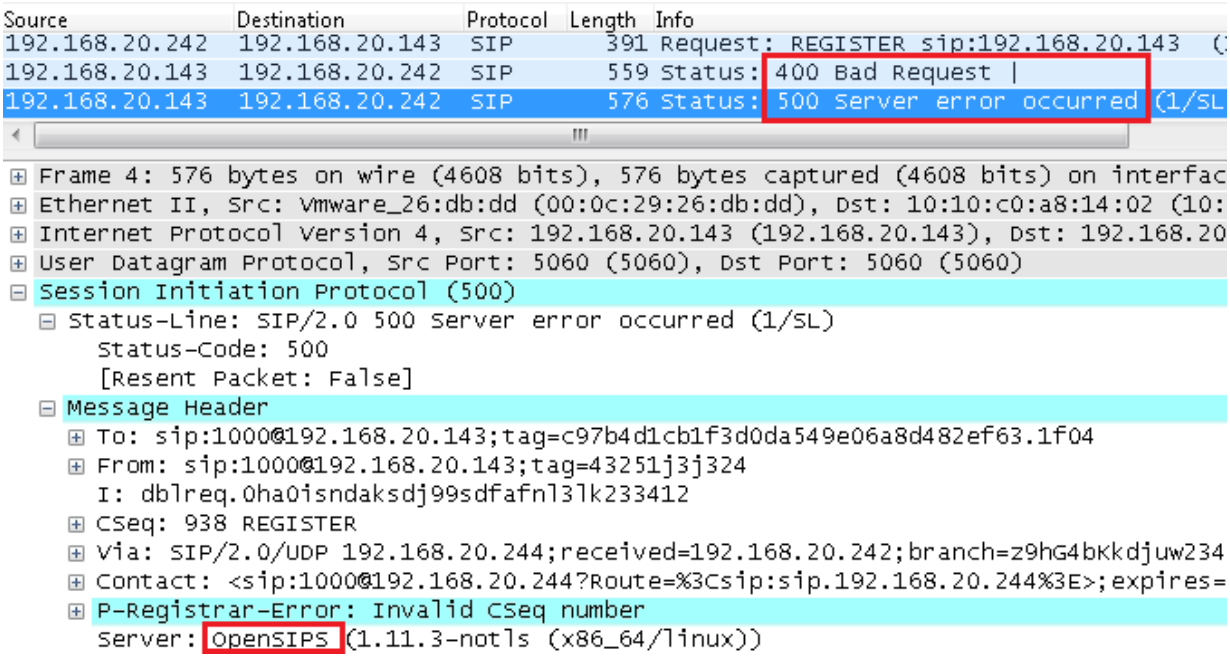
Obr. 6.68: Odpoveď Yate na žiadosť s vysokým Cseq

Žiadosť zamieta aj Kamailio a odpovedá špecifickou 400 CSeq number is illegal, čím presne ako Freeswitch naznačuje chybné pole (pozri nasledujúci obr.).



Obr. 6.69: Odpoveď Kamailio na žiadosť s vysokým Cseq

Reakcia OpenSIPS bola takisto chybová odpoveď 400 Bad Request, za ktorou však nasledovala druhá chybová správa serveru - 500 Server error occurred, kde v poli Registrar-Error je uvedený dôvod a to nesprávna hodnota Cseq (Invalid CSeq number) pozri obr. 6.70. V logu je vidieť kritická chyba, (ktorá bola pravdepodobne aj dôvodom druhej odpovede), že sa ústredne nepodarilo kontakt zapísať do registračnej SQL databázy kvôli invalidnej hodnote sekvenčného čísla.



Obr. 6.70: Odpoveď OpenSIPS na žiadosť s vysokým Cseq

6.3.7 Neznáma autorizačná schéma

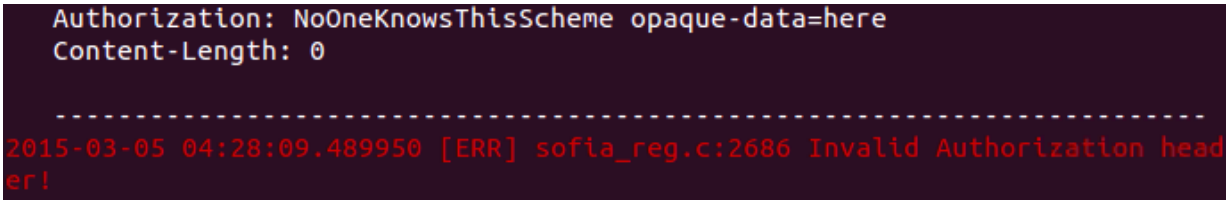
Útok posiela žiadosť REGISTER s neznámou hodnotou v poli *Authorization*. To sa používa k prenosu autentifikačných údajov klienta. Prvá býva uvedená schéma, najpoužívanejšia je Digest, v tomto teste je to ale hodnota „NoOneKnowsThisScheme“. Jej celý tvar je :

```
Authorization: NoOneKnowsThisScheme opaque-data=here
```

Ústredňa by mala takéto pole v správe odignorovať a spracovať ju akokeby v nej toto pole nebolo. Následne by mala zaslať štandardnú odpoveď s hodnotou poľa *Authorization*, ktorej rozumie (Digest). [41]

Asterisk na správu reaguje podľa očakávania. Zasiela odpoveď 401 so schémou, ktorej rozumie.

Freeswitch odpovedá takisto správou 401. Na vadnú hodnotu upozorní v logu pozri obr.



Obr. 6.71: Freeswitch upozorní na neznámu autorizačnú schému

Reakcia Yate je chybová odpoveď 400 Bad Request ako v dvoch predchádzajúcich testoch. Nie je to ideálne, pretože k registrácii nemôže dôjsť. V ideálnom prípade by ústredňa mohla pole odignorovať a poslať odpoveď so známou autentizačnou schémou ako Asterisk alebo Freeswitch.

Kamailio aj OpenSIPS reagujú tak ako aj prvé dve menované ústredne, pole odignorujú a zašlú štandardnú odpoveď 401 Unauthorized.

6.4 Modifikované IAX2 NEW rámce

Správa NEW sa v protokole IAX posiela ako žiadosť k zahájeniu hovoru. Jedná sa o riadiaci rámec (pozri kapitola 2.3.2). Je posielaný ako prvý a obsahuje viacero povinných informačných elementov (polia parametrov hlavičky), tie sú zhrnuté v nasledujúcej tabuľke. Pred vyslaním NEW správy, musí lokálny klient priradiť zdrojový identifikátor hovoru, ktorý nie je používaný v inom hovore, takisto musí priradiť časové razítko. Prvým informačným elementom musí byť číslo verzie protokolu. Nasledujúce elementy nemajú presne stanovené poradie. NEW je vlastne obdoba INVITE žiadosti v SIP protokole.

Typickou odpoveďou na žiadosť je zaslanie jedného z rámcov:

- REJECT – v prípade, že správu zamietne (mal by uviesť dôvod),
- AUTHREQ – ak vyžaduje autentifikáciu,
- ACCEPT – v prípade, že správu prijme (typicky po platnej autentifikácii),
- HANGUP – ak prepruší spojenie, uprednostňuje sa však REJECT. [26]

Útočné správy boli znova generované na Spirente z rozsahu 192.168.20.242-245 a zasielané k ústredni s IP 192.168.20.143. Ako už bolo spomenuté v práci skorej, IAX2 podporujú iba dvé ústredne – Asterisk a Yate. Narozdiel od útokov SIP zahrnutých v balíčku licencie v Spirente, boli všetky „fuzzing" útoky pre IAX2 vytvorené celé manuálne cez Attack Designer. Pri ich tvorbe bol zvažovaný možný dopad daného útoku, ale aj jeho potencionálna atraktivita k získavaniu zaujímavých reakcií od ústrední. Zdrojové XML súbory útokov sú takisto súčasťou prílohy.

Tab. 6.1: Vybrané informačné elementy pre správu NEW [26]

Informačný element	Povinnosť	Funkcia
Version	áno	Verzia protokolu
Called Number	áno	Volané číslo
Calling Number	nie	Číslo volajúceho
Codec Preferences	áno	Zoznam podporovaných kodekov
Calling Name	nie	Meno volajúceho
Username	nie	Špecifikuje užívateľa
DNID	nie	Vytočené číslo
Calling Presentation	áno	Definuje formát používaných čísel
Language	nie	Jazyk, v ktorom je signalizácia
Date	nie	Dátum odoslania

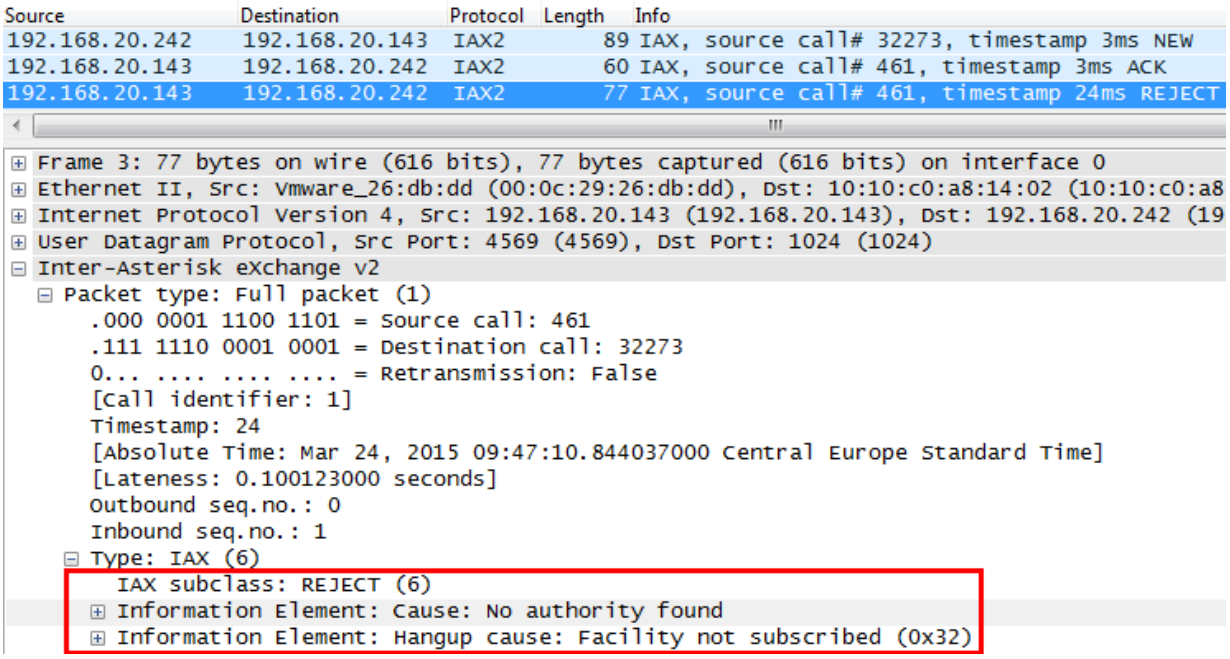
6.4.1 Neexistujúca verzia protokolu

V tomto type útoku bol zmenený informačný element riadiacej NEW správy zo súčasne definovanej verzie 2 (podľa RFC 5456) na 3. Pole má teda nasledujúci tvar:

```
Information Element: Protocol version: 0x0003
```

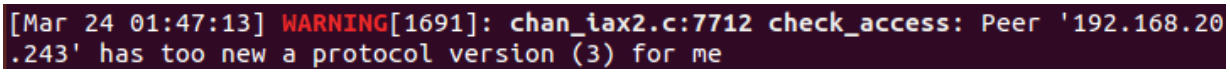
Ústredňa Asterisk na takúto žiadosť odpovedá po prvotnom prijatí správy zaslaním ACK, následným zamietnutím, tj. rámcom REJECT, v ktorom je uvedený aj dôvod zamietnutia. Ten je v nepovinnom informačnom elemente *Cause* (z ang. dôvod), konkrétne *No Authority Found* (pozri obr.) Tento dôvod však nie je v RFC uvedený a dá sa predpokladať, že

sa jedná o interný od tvorcov Asterisku. Pri kratšom hľadaní sa dá zistiť, že takýto dôvod sa vyskytuje v odpovediach na nesprávne údaje ako meno, heslo alebo typ. V tomto prípade je to aj verzia, ktorá je v štadnardnej verzii použítá v kontexte smerovacieho plánu a v tejto neštandardnej forme spôsobuje danú reakciu ústredne.



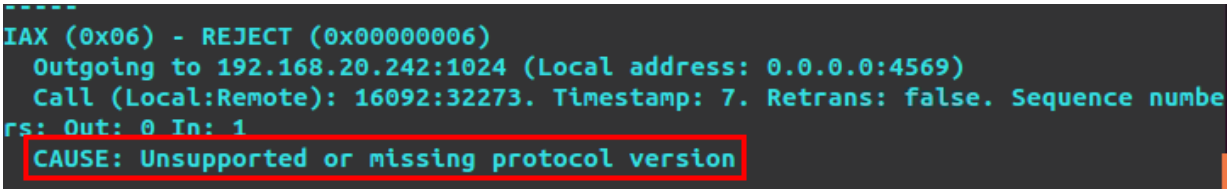
Obr. 6.72: Záporná odpoveď Asterisku na správu s neznámou verziou protokolu

Ústredňa teda správy s neštandardnou verziou zamieta a pre tento protokol aspoň dokáže odoslať odpoveď s odôvodnením (porov. SIP útok v kapitole 6.1.1). V logu je vidieť nasledujúce upozornenie, že sa jedná o neznámu verziu.



Obr. 6.73: Upozornenie Asterisku na správu s neznámou verziou protokolu

Yate odpovedá takisto ako Asterisk. V odpovedi REJECT však uvádza aj konkrétny dôvod – nepodporovanú verziu protokolu (pozri obr.). Môžeme konštatovať, že má teda vhodnejšie ošetrený daný jav.



Obr. 6.74: Dôvod Yate v odpovedi na správu s neznámou verziou protokolu

6.4.2 Zlúčenie informačných elementov

V útoku je posielaná správa NEW s opakovaným chýbajúcim riadkovým odsadením za informačným elementom *Calling number*. Jedná sa teda o deformovanú žiadosť, podobne ako v SIP útoku 6.2.7. Jej tvar je nasledovný:

```
Inter-Asterisk eXchange v2
    Packet type: Full packet (1)
        Type: IAX (6)
            IAX subclass: NEW (1)
            Information Element: Protocol version: 0x0002
            Information Element: Number/extension being called: 101
            Information Element: Calling number: 100 Information Element:
Calling presentation: 0x43            Information Element: Calling type of
number: 0x00           Information Element: Calling transit network select:
0x0000                          Information Element: Name of caller:
Information Element: Desired language: en             Information Element:
Username: 101             Information Element: Desired codec format: G.729a
Audio                          Information Element: 64-bit codec format:
Information     Element:     Actual     codec     capability:     0x00000703
Information Element: 64-bit codec capability:
```

Asterisk takúto správu zachytí v logu a upozorní na ňu, že nie je možné dekódovať daný rámec, tak ako je vidieť na obrázku. Sám však odosielateľovi neodošle žiadnu odpoveď.

```
[Mar 24 07:04:23] WARNING[1684]: chan_iax2.c:1158 iax_error_output: Information
element length exceeds message size
[Mar 24 07:04:23] WARNING[1684]: chan_iax2.c:10126 socket_process_helper: Undeco
dable frame received from '192.168.20.242'
```

Obr. 6.75: Upozornenie Asterisku na deformovanú správu

Ústredňa Yate odpovedá potvrdeným ACK, že správu obdržala a následne vyhodnotí chybnú správu a odosiela správu INVAL, ktorá sa používa ako odpoveď odosielateľovi na správu, ktorá nie je validna. Nemusí obsahovať žiaden informačný element a ten v tomto prípade ani neobsahuje. [26] V porovnaní s reakciou Asterisku je koncový užívateľ aspoň informovaný, že poslal chybnú správu.

Source	Destination	Protocol	Length	Info
192.168.20.242	192.168.20.143	IAX2	92	IAX, source call# 32273, timestamp 3ms NEW
192.168.20.143	192.168.20.242	IAX2	60	IAX, source call# 16090, timestamp 3ms ACK
192.168.20.143	192.168.20.242	IAX2	60	IAX, source call# 0, timestamp 3ms INVAL

Obr. 6.76: Reakcia Yate na deformovanú správu

6.4.3 Neznámy podporovaný kodek

V tomto útoku je zasielaný v rámci NEW povinný informačný element s neznámym kodekom požadovaným ku komunikácii. Tvar tohto poľa je nasledovný:

```
Information Element: Desired codec format: Unknown (0x005041d6)
```

Medzi bežne používané patria G.711, G.723, G.729. Ich použitie v hovore závisí od individuálneho klienta a ním podporovanej sady.

Obidve ústredne odpovedajú správou AUTHREQ a vyžadujú autentifikáciu, ktorá podlieha mechanizmu z kapitoly 2.3.4. Ide o štandartné chovanie, ústredniam neznáma hodnota neprekáža a táto hodnota ju pred autentifikovaním sa klienta nezaujíma.

6.4.4 *Chýbajúce povinné elementy*

Útok posiela IAX2 NEW správu s viacerými chýbajúcimi povinnými informačnými elemantami. Správa obsahuje len element použitej verzie protokolu, ktorý musí byť uvedený ako prvý podľa [26]. Celý prenášaný IAX2 rámec má tvar:

```
Inter-Asterisk eXchange v2
    Packet type: Full packet (1)
        .111 1110 0001 0001 = Source call: 32273
        .000 0000 0000 0000 = Destination call: 0
        0... .... .... .... = Retransmission: False
        [Call identifier: 1]
        Timestamp: 3
        [Absolute Time: Mar 25, 2015 11:53:12.835013000]
        [Lateness: -0.002000000 seconds]
        Outbound seq.no.: 0
        Inbound seq.no.: 0
        Type: IAX (6)
            IAX subclass: NEW (1)
            Information Element: Protocol version: 0x0002
```

Na takúto správu Asterisk očakávane neodpovedá štandartným vyžiadaním autentizácie, ale zamietnutím REJECT. Dôvod uvádza v dvoch elementoch:

```
      Information Element: Cause: Unable to negotiate codec
      Information Element: Hangup cause: Bearer capability not available
```

Prvým dáva zjavne odosielateľovi správy NEW najavo, že neuvedené pole s požadovaným kodekom, mu spôsobuje problém a druhým, že nie je schopný definovať schopnosti prenosu (nevie komu má smerovať správu) a preto inicializáciu hovoru ruší.

Podobne reaguje aj Yate, najprv pošle odpoveď ACK a následne v správe REJECT uvádza stručný dôvod:

```
      Information Element: Cause: nomedia
      Information Element: Hangup cause: Bearer capability not available
```

Môžeme konštatovať, že chýbajúce elementy očakávane spôsobujú zamietnutie žiadostí.

6.4.5 *Prázdne informačné elementy*

Tento útok posiela správu NEW s prázdnymi informačnými elementami *Calling Number* a *Calling Name.* Obidva sú nepovinné, ale odporúčané, nakoľko identifikujú odosielateľa.

Asterisk reaguje štandartnou správou AUTHREQ, čím vyžaduje autentifikáciu klienta. Takáto reakcia potvrdzuje, že aj keď sú polia prítomné, ale sú prázdne na funkcionalitu to u Asterisku nemá vplyv. Smerodatným mu je pole *Username*, podľa ktorého vie o akého užívateľa sa jedná. Takisto odpovedá aj ústredňa Yate.

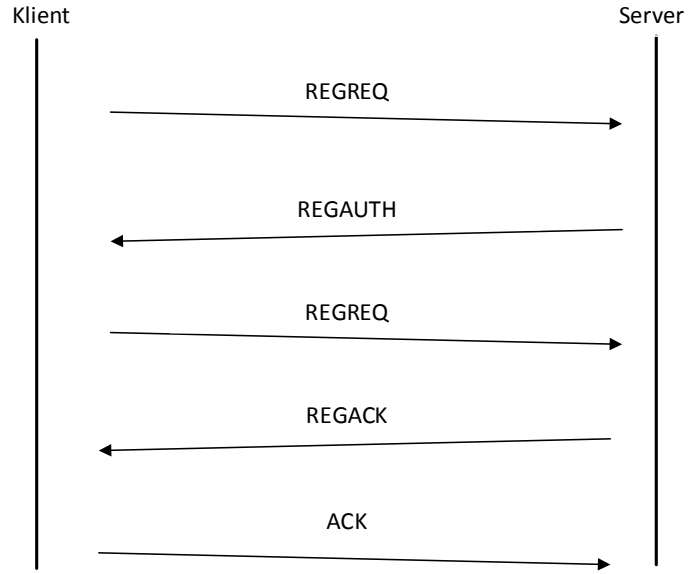
Source	Destination	Protocol	Length	Info
192.168.20.242	192.168.20.143	IAX2	89	IAX, source call# 32273, timestamp 3ms NEW
192.168.20.143	192.168.20.242	IAX2	74	IAX, source call# 3219, timestamp 14ms AUTHREQ

Obr. 6.77: Odpoveď AUTHREQ obidvoch ústrední na správu s prázdnymi hodnotami elementov

6.5 Modifikované IAX2 REGREQ rámce

Aby mohol byť IAX klient dostupný iným klientom, potrebuje volajúci klient poznať adresu volaného. Protokol IAX2 poskytuje funkcionalitu k registrácii jeho adresy a údajov s iným, tak aby sa klienti mohli kontaktovať. Jedná sa o voliteľný mechanizmus. Ak je použitý, zahrňuje dve autentifikačné metódy – pomocou MD5 hešu alebo RSA klúča (kap. 2.3.4).

Princíp registrácie spočíva v zaslaní REGREQ správy klientom k ústredni (registračnému serveru, čo je typicky jedna z logických rolí ústredne). Ak je požadovaná autentifikácia, registračný server odpovie s REGAUTH správou, kde sú uvedené podporované metódy. Odpoveďou klienta je REGREQ s vybranou metódou. Zaujímavosťou je, že ak sa klient nemôže autentifikovať nemusí mu byť poslaná odpoveď. [26] Ak sa však môže, odošle správu REGACK s uvedenou hodnotou tzv. *refresh* intervalu, počas ktorého má byť registrácia platná. U IAX2 je to implicitne 60 sekúnd, čo je v porovnaní so SIP veľmi krátky interval a podporuje tým lepšie zabezpečenie protokolu (napr. voči ukradnutiu registrácie). K indikácii zlyhania sa používa odpoveď REGREJ. K odregistrovaniu sa používa REGREL, na ktorú je reakciou znova výzva k autentifikácii, čo tiež podporuje zabezpečenie (voči útoku odstránenia registrácie). Celý proces sa riadi nasledujúcim diagramom.



Obr. 6.78: Registračný proces v IAX2

Jediným povinným elementom v REGREQ správe je *username*, tj. špecifikácia úžívateľa, ktorý sa chce zaregistrovať. Ak sa jedná o odpoveď na autentifikačnú výzvu, musí obsahovať aj výsledok výzvy z použitej metódy. Nepovinným elementom je vyššie spomenutý *refresh* interval.

V testoch je skúmaná reakcia ústredňe na modifikovanú prvotnú REGREQ správu od klient k ústredni. Tá bola posielaná zo Spirentu z rovnakého IP rozsahu ako doteraz. Ústredňa na adrese 192.168.20.143 by mala, ak jej správa nevadí odpovedať REGAUTH, inak inou chybovou správou, kde definuje vadnú časť alebo dôvod.

6.5.1 Hodnota refresh nulová

V tomto útoku je posielaná správa REGREQ s nulovou hodnotou poľa *refresh.* Tým je dané najavo, že registrácia by mala platiť nula sekúnd, čo je pre registrujúceho klienta nezmyselné. V ideálnom prípade je reakcia ústredne na takúto správu zamietnutie registrácie s uvedeným dôvodom. [26]

Obidve ústredne v tomto teste reagujú zaslaním ACK (potvrdzuje príjem správy) a následným REGAUTH, čím žiada klienta k autentifikácii aj keď tá je nezmyselná, nakoľko

registrácia by hneď mala zaniknúť. Takouto reakciou je dokázané, že analyzátory IAX2 správ v ústredniach takýto nelogický scenár nemajú ošetrený. Na obrázku je vidieť tento sled správ v Yate.

```
IAX (0x06) - REGREQ (0x0000000d)
  Incoming from 192.168.20.243:1025 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 0:32248. Timestamp: 3. Retrans: false. Sequence numbers:
Out: 0 In: 0
  USERNAME: 100
  REFRESH: 0 second(s)
-----
20150327081701.519833 <iaxengine:INFO> Sending frame [0x7ffcc27ff9b0]
-----
IAX (0x06) - ACK (0x00000004)
  Outgoing to 192.168.20.243:1025 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 22734:32248. Timestamp: 3. Retrans: false. Sequence numb
rs: Out: 0 In: 1
  No Information Element(s)
-----
20150327081701.519999 <iaxengine:INFO> Sending frame [0x7ffcc27ff9b0]
-----
IAX (0x06) - REGAUTH (0x0000000e)
  Outgoing to 192.168.20.243:1025 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 22734:32248. Timestamp: 4. Retrans: false. Sequence numb
rs: Out: 0 In: 1
  USERNAME: 100
  AUTHMETHODS: 0x0002 (MD5)
  CHALLENGE: 656276075
```

Obr. 6.79: Reakcia Yate na REGREQ správu s nulovým refresh

6.5.2 *Chýbajúci povinný element*

V útoku bola posielaná k ústredniam správa REGREQ bez povinného informačného poľa *username,* ktorý jednoznačne identifikuje klienta žiadajúceho o registráciu k ústredni. [26] Správa obsahuje iba nepovinný element *refresh*. Jej tvar je teda nasledovný:

```
Inter-Asterisk eXchange v2
    Packet type: Full packet (1)
        .111 1101 1111 1000 = Source call: 32248
        .000 0000 0000 0000 = Destination call: 0
        0... .... .... .... = Retransmission: False
        Timestamp: 3
        Outbound seq.no.: 0
        Inbound seq.no.: 0
        Type: IAX (6)
            IAX subclass: REGREQ (13)
            Information Element: When to refresh registration: 300
```

Asterisk reaguje odoslaním ACK, ktorým naznačuje že správu prijal a následným REGREJ, v ktorej uvádza iba všeobecný nasledovný dôvod:

```
            Information Element: Cause: Registration Refused
```

Presnejší dôvod je uvedený iba v oznámení v logu v tvare:

```
NOTICE[1692]: chan_iax2.c:8074 register_verify: Empty registration from
192.168.20.243
```

Takýto dôvod je ale nedostatočný, nakoľko aj samotný administrátor musí dôvod určiť až zo samotnej štruktúry správy.

Ústredňa Yate reaguje podobne, po prijatí správy odoslaním ACK rámca a po okamžitom rozanalyzovaní správy odiela REGREJ správu, v ktorej uvádza presný dôvod zamietnutia žiadosti k registrácii (pozri nasledujúci obr.), ktorým je chýbajúce povinné pole. Ústredňa teda v tomto útoku narozdiel od predchádzajúceho, má takýto neštandartný tvar podchytení lepšie.

```
IAX (0x06) - REGREJ (0x00000010)
  Outgoing to 192.168.20.242:1024 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 22735:32248. Timestamp: 3. Retrans: false. Sequence numbe
rs: Out: 0 In: 1
  CAUSE: Username is missing
  CAUSECODE: 0x60 (missing-mandatory-ie)
```

Obr. 6.80: Zamietnutie REGREQ od Yate kvôli chýbajúcemu username

6.5.3 *Username kódovaný v inej sade*

Tento útok posiela povinný element *username* správy REGREQ v tvare, ktorý nie je definovaný v sade UTF-8. Tá používa jeden bajt na znak a je kompatibilná s ASCII sadou. Používa sa pre zakódovanie užívateľského mena v IAX2 protokole. [26] Použité tu ale boli znaky zo sady UTF-16, ktorá používa dva bajty na znak. Užívateľské meno má konkrétne znak sýrskej a arabskej abecedy (ڤب). Skúmaná je možnosť ústredne interpretovať takéto meno, aj keď by malo byť interpretované v UTF-8 resp. ASCII sadou a znaky by mali byť prekódované za iné.

Asterisk potvrdil naznačené očakávanie a zachytený informačný element vo Wiresharku má tvar:

```
Username (peer or user) for authentication: u\aO
```

Reaguje štandartným zaslaním REGAUTH. Ústredňa Yate takisto zasiela štandartnú výzvu k autentifikácii REGAUTH (na obrázku). Neznáme znaky sú teda interpretované v rámci sady.

```
20150327085734.272720 <iaxengine:INFO> Sending frame [0x7ffcc27ff9b0]
-----
IAX (0x06) - REGAUTH (0x0000000e)
  Outgoing to 192.168.20.243:1025 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 22738:32248. Timestamp: 5. Retrans: true. Sequence number
s: Out: 0 In: 1
  USERNAME: uO
  AUTHMETHODS: 0x0002 (MD5)
  CHALLENGE: 1169348754
```

Obr. 6.81: Reakcia Yate na REGREQ správu s username v UTF-16

6.5.4 *Neznámy typ správy*

V tomto útoku bola posielaná správa s neznámym typom rámca (hexa hodnota 23), ktoré je vyhradené pre budúce použitie. REGREQ má hodnotu 12 tj. (0d). Telo správy bolo použité rovnaké ako zo štadnartnej REGREQ správy a teda obsahuje dva elementy, povinný *username* a nepovinný *refresh*. Pri zachytení vo Wiresharku však namiesto typu neznámeho rezervovaného typu bola vyhodnotená ako PROVISION (z ang. rezerva), tá však v RFC5456 nie je definovaná a jedná sa pravdepodobne o definíciu rezervovaného typu. Správa mala teda nasledujúcu štruktúru:

```
Inter-Asterisk eXchange v2
    Packet type: Full packet (1)
        .111 1101 1111 1000 = Source call: 32248
        .000 0000 0000 0000 = Destination call: 0
        0... .... .... .... = Retransmission: False
        Timestamp: 3
        Outbound seq.no.: 0
        Inbound seq.no.: 0
        Type: IAX (6)
            IAX subclass: PROVISION (23)
            Information Element: Username for authentication: 100
            Information Element: When to refresh registration: 300
```

Asterisk na takúto správu vôbec nereaguje. Odosielateľ teda nemusí vedieť prečo neprišla odpoveď (napr. kvôli chybe v sieti) a či vôbec zaslanú správu ústredňa obdržala. V logu je iba vidieť, že správu obdrží, ale nereaguje na ňu.

Ústredňa Yate reaguje zaslaním správy INVAL, ktorou reagovala aj na modifikovanú správu v kapitole 6.4.2. Neznámy typ správy teda vyhodnocuje ako nesprávny.

```
20150327093958.663565 <iaxengine:INFO> Received frame [0x7ffcc27ff9b0]
-----
IAX (0x06) - PROVISION (0x00000023)
  Incoming from 192.168.20.243:1025 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 0:32248. Timestamp: 3. Retrans: false. Sequence numbers:
Out: 0 In: 0
  USERNAME: 100
  REFRESH: 300 second(s)
-----
20150327093958.664120 <iaxengine:INFO> Sending frame [0x7ffcc27ff9b0]
-----
IAX (0x06) - INVAL (0x0000000a)
  Outgoing to 192.168.20.243:1025 (Local address: 0.0.0.0:4569)
  Call (Local:Remote): 0:32248. Timestamp: 3. Retrans: false. Sequence numbers:
Out: 0 In: 0
  No Information Element(s)
```

Obr. 6.82: Reakcia INVAL zaslaná ako odpoveď na neznámy typ IAX2 rámca

6.6 Zhrnutie výsledkov testov a návrh opatrenia

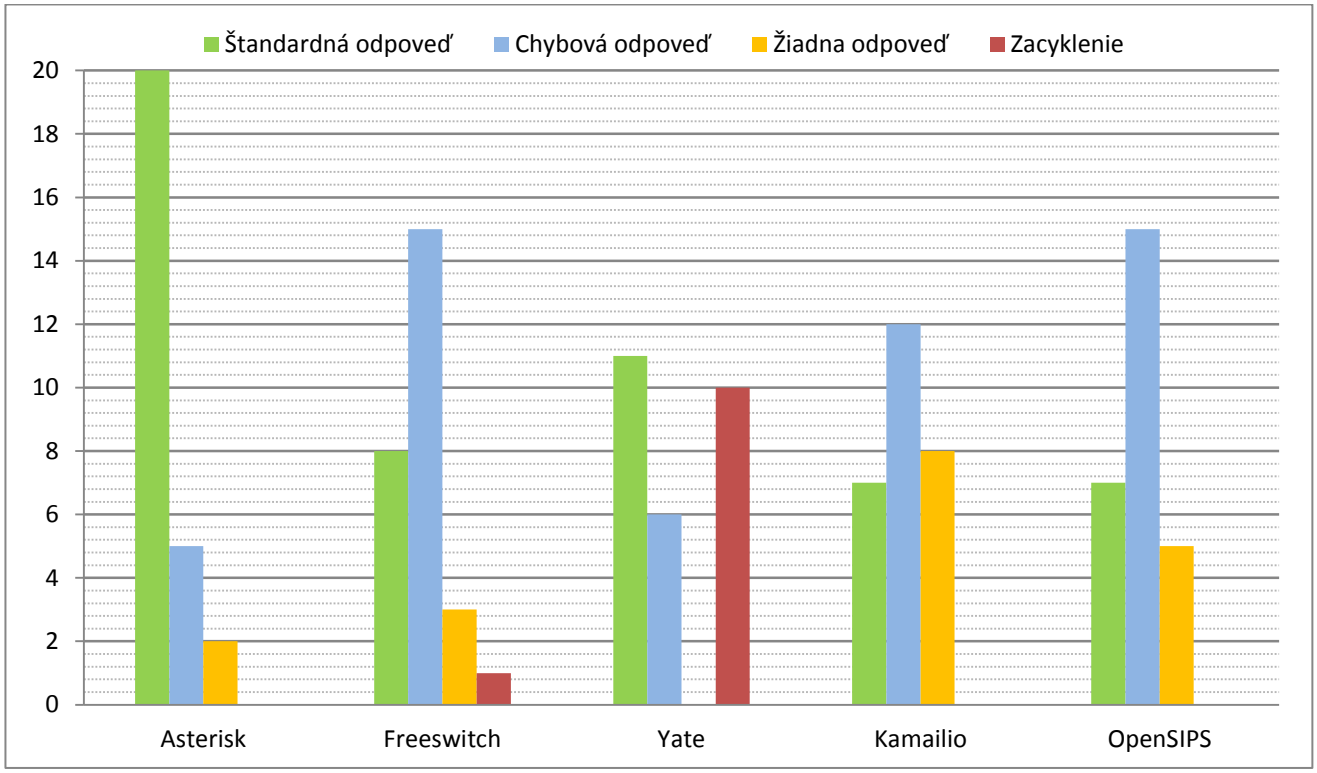
Pri analyzovaní výsledkov sú možné dva prístupy k ich vyhodnoteniu. Prvým sa dá zaoberať, ak chyby v hlavičke vzniknú neúmyselným spôsobením procesných chýb v samotných koncových zariadeniach (pri ich zostavovaní) resp. analyzátoroch syntaxu ústrední (spracovávaní alebo preposielaní po ceste k cieľu). Vtedy by bolo žiadúce od ústrední pristupovať k takýmto správam čo najviac tolerantne. V ideálnom prípade dokázať chybu modifikovať (napr. pozri Asterisk v 6.3.4) alebo odignorovať (Asterisk v 6.3.5) a tak zaistiť jej doručenie k cieľu a umožneniu komunikácie. V horšom prípade tu môže nastať situácia, kedy ústredňa v maximálnej snahe dosiahnuť tento stav, takúto správu modifikuje vadným spôsobom a povedie to k dosiahnutiu opačného efektu – nedosiahnutiu úspešnej výmeny správ v komunikácii (napr. Yate v 6.2.9 a 6.2.10). Riešenie pri takomto spôsobe uvažovania je jedine v kvalitne napísanom jadre a obslužných modulov, čo zo všetkých piatich ústrední má na

základe dosiahnutých výsledkov (pozri zhrnutie v tab. 6.2) najznámejšia, Asterisk. Grafické porovnanie je pre SIP na grafe obr. 6.83 a pre IAX2 na obr. 6.85, kde je prehľadne farebne odlíšený počet jednotlivých typov reakcií, ktoré boli zaznamenané pre jednotlivé ústredne. Zelenou farbou sú odlíšené štandardné odpovede, ktoré ústredňa zaslala ak obdržanú správu vnímala ako bežnú. Modrou farbou sú odlíšené štandardné chybové odpovede, ktorými ústredňa dáva najavo, že požiadavku nie je schopná spracovať. Oranžovou farbou sú odlíšené reakcie, kedy ústredňa žiadnu odpoveď neposlala a z hľadiska tohto prvotného prístupu k vyhodnoteniu sa jedná o nežiadúci stav, pretože klient nevie prečo komunikácia zlyháva. A napokon bordovou farbou sú zvýraznené stavy, kedy došlo až k obmedzeniu funkcionality ústredne. Percentuálne zastúpenie štandardných odpovedí, ktoré sú pri bežnej komunikácii vidieť je pre SIP na obr. 6.84 a pre IAX2 na obr. 6.86.

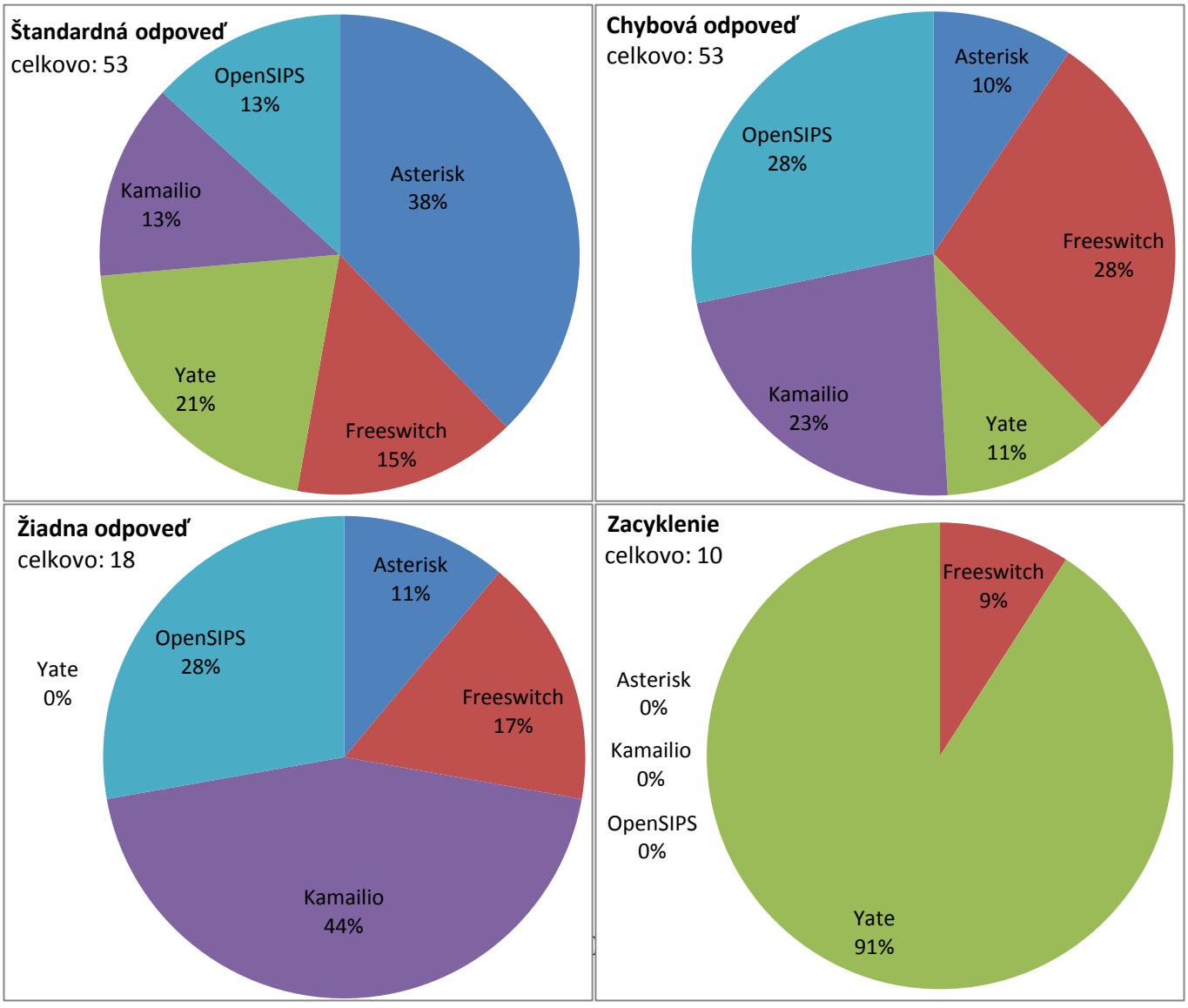
Druhým prístupom k vyhodnoteniu dosiahnutých výsledkov je ten pre prácu relevantnejší, a teda, že takéto modifikované správy by boli vytvorené úmyselne útočníkom resp. jeho skriptami alebo aplikáciami. Vtedy by také správy mohli viesť, jednak k útoku na iného klienta alebo až k znefunkčneniu ústrední či ich obmedzeniu. K čomu došlo pri cyklickom zasielaní odpovedí viac ráz u ústredne Yate a raz u Freeswitchu (kap. 6.1.5). Percentálne zhrnutie je znova vidieť na obr. 6.84.

Okrem toho je potencionálne riziko v zaslaní škodlivého kódu, či už vo forme uniklej časti hlavičky v niektorom z polí alebo neštandardizovanej hodnoty určitého poľa, na čo boli zamerané aj viaceré z testov. V tomto prípade prístupu by bolo pre ústredne vhodné, aby takéto správy dokázali rozpoznať a ďalej s nimi nepracovať, tj. zabrániť v komunikácii. V ideálnom prípade by aj inicátorovi, od ktorého správy pochádzajú zaslali odpoveď s definovanou chybou, kvôli ktorej žiadosť nespracovávajú. Čo je informatívne, jednak aj keby správa náhodou nepochádza od útočníka, ale iba z chybného vygenerovania zo zdroja. Takéto chybové hlášky majú najlepšie naimplementované Freeswitch a OpenSIPS v tesnom závese s Kamailio (pozri obr. 6.84).

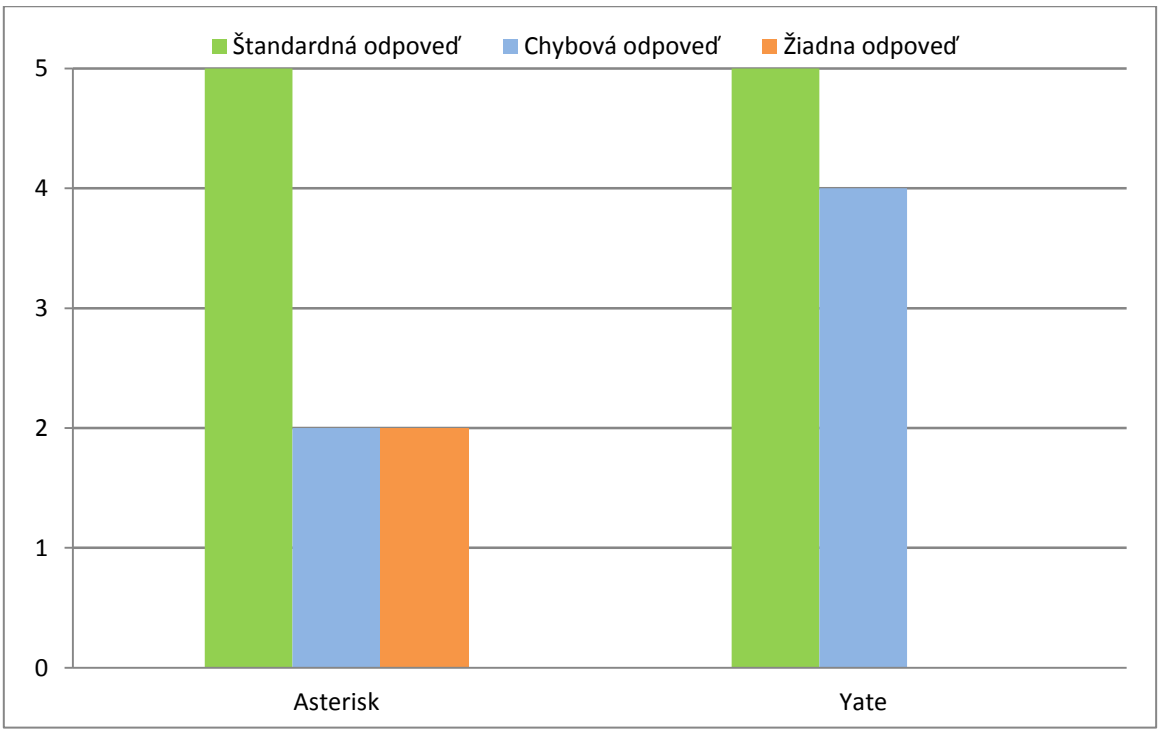
V horšom prípade ústredne modifikovanú žiadosť síce nesprácovávajú, ale ani o tom neinformujú odosielateľa (napr.pozri testy v 6.1.1 a 6.2.7). Na druhej strane je možné pre administrátora aspoň z logov a debugovacieho režimu dostať čiastočné informácie, čo konkrétne ústredni prekáža. Takéto správanie bolo najviac vidieť u ústredne Kamailio, ktorá má však aspoň obsiahlejšie logovacie záznamy (pozri obr. 6.84).



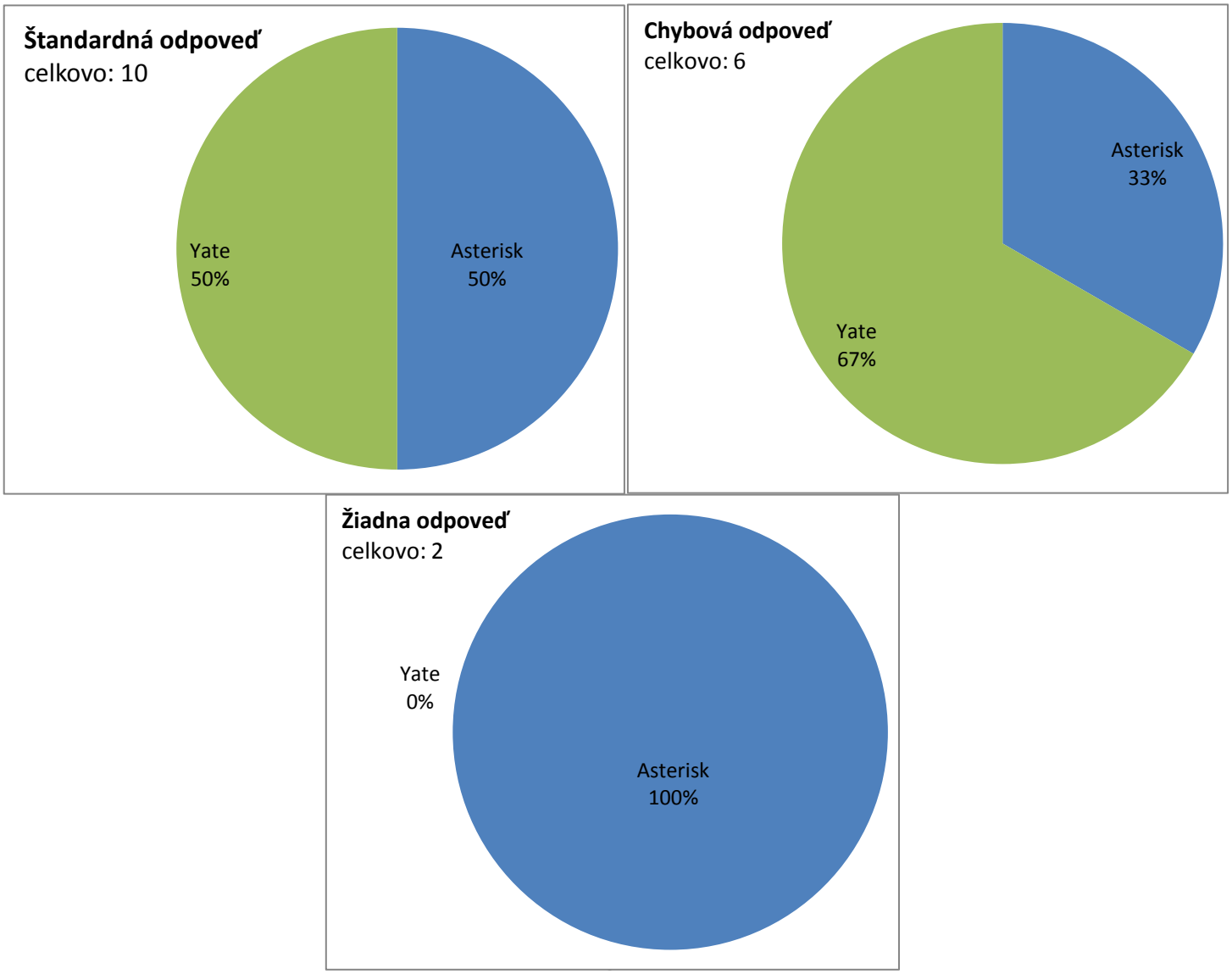
Obr. 6.83: Počet jednotlivých reakcí ústrední v testoch s modifikovanými SIP žiadosťami



Obr. 6.84: Percentuálne zastúpenie reakcií na SIP správy u jednotlivých ústrední



Obr. 6.85: Počet jednotlivých reakcí ústrední v testoch s modifikovanými IAX2 žiadosťami



Obr. 6.86: Percentuálne zastúpenie reakcií na IAX2 správy u jednotlivých ústrední

Tab. 6.2: Zhrnutie reakcii ústrední na útoky s neštandardnými SIP správami

Typ správy	útok \ ústredňa	Asterisk 11.6	Freeswitch 1.5	Yate 5.0	Kamailio 4.2	Opensips 1.11
OPTIONS	**Nesprávna verzia SIP záhlavia**	• žiadna odpoveď	• žiadna odpoveď	• 200 OK • zachová verziu	• žiadna odpoveď • rozpozná chybnú hlavičku (Via)	• žiadna odpoveď • rozpozná chybnú hlavičku (Via)
	Viacnásobné pole Content-Length	• 200 OK	• 400 Bad Request • dôvod: Bad Content-Length Header	• 200 OK • cyklicky posielaná	• 400 Bad Request • dôvod: Content-Length mis-match	• odpoveď kópia správy • alebo 408 Request Timeout
	Chýbajúca časť branch v položke Via	• 200 OK	• 200 OK	• 200 OK	• 200 OK	• 200 OK
	Medzery v adrese poľa To:	• 200 OK	• 200 OK	• 200 OK • cyklicky posielaná	• 403 Not relaying	• 403 Rely forbiden
	Chýbajúca medzera v poli From:	• 200 OK	• 200 OK • cyklicky posielaná	• 200 OK • cyklicky posielaná	• žiadna odpoveď • rozpozná chybnú hlavičku	• žiadna odpoveď • rozpozná chybnú hlavičku
	Aliasy bez úvodzoviek	• 200 OK	• 400 Bad From Header	• 200 OK	• žiadna odpoveď • rozpozná chybnú hlavičku	• 403 Rely forbiden
	Medzery na konci tzv. Request-Line	• 200 OK	• žiadna odpoveď	• 200 OK	• žiadna odpoveď • rozpozná chybnú hlavičku	• 403 Rely forbiden
	Neznáme Require a Proxy-Require hodnoty	• 200 OK	• 420 Bad Extension	• 200 OK • odignorované hodnoty	• žiadna odpoveď • rozpozná chybnú hlavičku	• 403 Rely forbiden
	Nesprávna adresná URI schéma	• 416 Unsupported URI Scheme	• žiadna odpoveď	• 200 OK • cyklicky posielaná	• 200 Keepalive	• 484 Address Incomplete

	Nulová hodnota v poli Max-Forwards	• 200 OK	• 200 OK	• 200 OK • cyklicky posielaná	• 483 Too Many Hops	• 483 Too Many Hops
INVITE	**Neštandardné pole Accept**	• 100 Trying • uskoční spojenie	• 406 Not Acceptable	• 100 Trying • neuskoční spojenie	• 400 Content-Length mis-match	• 403 Rely forbiden
	Nesprávna hodnota časovej zóny v Date poli	• 100 Trying • uskoční spojenie	• 400 Bad Session Description	• 100 Trying • neuskoční spojenie	• 403 Not relaying	• 403 Rely forbiden
	Hodnota Content Length väčšia ako telo správy	• 100 Trying • uskoční spojenie	• 400 Bad Request	• 503 Service Unavailable • cyklicky posielaná	• 400 Content-Length mis-match	• 403 Rely forbiden
	Zdeformovaná URI adresa (uniklé znaky)	• 480 Temporarily unavailable	• 400 Bad Session Description	• 503 Service Unavailable • cyklicky posielaná	• 400 Bad Request URI	• 484 Address Incomplete
	Zdeformovaná URI adresa (ohraničená < >)	• 480 Temporarily unavailable	• 400 Bad Request	• 503 Service Unavailable • cyklicky posielaná	• 400 Bad Request URI	• 484 Address Incomplete
	Chýbajúce povinné polia v hlavičke žiadosti	• 400 Bad Request	• 400 Bad Request	• 400 Bad Request	• 400 Missing Required Header in Request	• 403 Rely forbiden
	Plne zdeformovaná žiadosť	• žiadna odpoveď • rozpozná chybnú hlavičku	• 400 Bad Request	• 100 Trying + • 481 Call/Transaction Does Not Exist	• žiadna odpoveď • rozpozná chybnú hlavičku	• žiadna odpoveď • rozpozná chybnú hlavičku
	Medzery v Request-Line	• 100 Trying • uskoční spojenie	• 400 Bad Request	• 100 Trying • uskoční spojenie	• žiadna odpoveď • rozpozná chybnú hlavičku	• žiadna odpoveď • rozpozná chybnú hlavičku
	Neznámy Content-Type	• 100 Trying • uskoční spojenie	• 400 Bad Request	• 503 Service Unavailable • cyklicky posielaná	• 403 Not relaying	• 403 Rely forbiden

	Chýbajúce úvodzovky v mene volajúceho	• 100 Trying • uskoční spojenie	• 400 Bad To Header	• 503 Service Unavailable • cyklicky posielaná	• žiadna odpoveď • rozpozná chybnú hlavičku	• žiadna odpoveď • rozpozná chybnú hlavičku
REGISTER	**Neznámy parameter v poli Contact**	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized
	Neznámy URI parameter	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized
	Uniklá hodnota v poli Contact	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized
	Neohraničená adresa	• 401 Unauthorized	• 400 Bad Contact Header	• 401 Unauthorized	• 401 Unauthorized	• 401 Unauthorized
	Neočakávaná druhá správa	• 401 Unauthorized	• 401 Unauthorized	• 400 Bad Request	• 400 Content-Length mis-match	• 401 Unauthorized
	Hodnoty príliš veľké	• 401 Unauthorized • neočakávaná	• 400 Bad Cseq Header	• 400 Bad Request	• 400 Cseq number is illegal	• 400 Bad Request • 500 Server error occurred
	Neznáma autorizačná schéma	• 401 Unauthorized	• 401 Unauthorized	• 400 Bad Request	• 401 Unauthorized	• 401 Unauthorized

Tab. 6.3: Zhrnutie reakcii ústrední na útoky s neštandardnými IAX2 správami

Typ správy	útok \ ústredňa	Asterisk 11.6	Yate 5.0
NEW	Neexistujúca verzia protokolu	• REJECT • No Authority Found	• REJECT • Unsupported protocol version
	Zlúčenie informačných elementov	• žiadna odpoveď • rozpozná chybnú hlavičku	• INVAL
	Neznámy podporovaný kodek	• REGAUTH	• REGAUTH
	Chýbajúce povinné elementy	• REJECT • Unable to negotiate codec	• REJECT • nomedia
	Prázdne informačné elementy	• REGAUTH	• REGAUTH
REGREQ	Hodnota refresh nulová	• REGAUTH	• REGAUTH
	Chýbajúci povinný element	• REGREJ • všeobecný dôvod	• REGREJ • presne určí dôvod
	Username kódovaný v inej sade	• REGAUTH • zmení username	• REGAUTH • zmení username
	Neznámy typ správy	• žiadna odpoveď	• INVAL

6.6.1 Zabezpečenie pomocou ipfilter u32 modulu

Najhoršie sú teda útoky, ktoré spôsobia komplikácie s chodom ústredne. Preto by bolo dôležité pri častejšom posielaní takýchto správ, čo by potencionálny útočník mal skúsiť, eliminovať tieto správy pred tým, než ich ústredňa začne spracovávať. Jedným z možných riešení ako to dosiahnuť, je využitie firewallu (kap. 3.5.3). Ubuntu má implicitne vstavaný firewall *Netfilter*/*iptables,* ten v spojení s modulom *u32* dokáže priradiť pravidlo k jednotlivým bitom paketu. [4] To je však mierne komplikované, keďže tento modul vyžaduje zápis pravidiel s filtrovanými hodnotami v hexa tvare. Pomocou python skriptu na CD v prílohe, je však možné efektívne a relatívne jednoducho napísať pravidlá, ktoré budú hľadať zhodu v obsahu modifikovaných správ.

Príkladom takého pravidla napr. pre útok 6.2.5 je:

```
iptables -a INPUT –i eth0 -p udp \! -f -m u32 --u32 $(./iptables-SIPu32.py
"INVITE <sip:1001@192.168.20.244> SIP/2.0") -j DROP –j LOG--log-prefix "--
zahodena-ziadost",
```

kde je definované vstupné pravidlo na rozhraní eth0, využívajúce spomínaný modul a v zátvorke výstup skriptu, ktorý pravidlo zjednodušuje (inak by bol nutný jeho zápis v hexa hodnotách po bajtoch) a podľa prvého riadku hľadá zhodu v príchodzích UDP datagramoch. Následne je žiadosť s odpovedajúcou hlavičkou na vstupe iptables firewallu zahodená, nedostane sa do ústredne a aktivita zapísaná vo výpise s označením „*--zahodena-ziadost*“.

Takéto riešenie však predpokladá, že administrátor ústrední by štruktúru modifikovanej žiadosti mal odsledovanú a teda ju poznal, aby mohol vytvoriťpríslušné pravidlo. To by však za bežných okolností zistil, až keď by sa vyskytli komplikácie a spätne by dohladal príčinu a nastavil preventívne pravidlo, ak by sa situácia opakovala.

6.6.2 Zabezpečenie pomocou fail2ban

Ďalším možným riešením, viac sofistikovaným, je použitie nástroja *fail2ban.* Je napísaný v python jazyku a umožňuje automatické vytváranie pravidiel na základe sledovania log súborov ústrední. Je spustení automaticky pri štarte operačného systému a pomocou dvoch nakonfigurovaných súborov hľadá zhody v prijatých (alebo odoslaných) paketoch. Vychádza z netfiltru/iptables a dalo by sa povedať, že je jeho nádstavbou. Zdrojové súbory je možné stiahnuť z oficiálnych stránok vývojarov. [3]

Samotná konfigurácia spočíva v nastavení súborov v zložke fail2ban:

```
/etc/fail2ban/jail.conf
/etc/fail2ban/filter.d .
```

Prvý súbor je najdôležitejší a obsahuje nastavenie zákazov. Druhý obsahuje podmienky detekcie pre danú ústredňu, tie vychádzajú z konfiguračných súborov ústrední (používa z nich výrazy a premenné). Dôležité je mať aj nastavenú výnimku na svoju IP, tá je definovaná v súbore

```
/etc/fail2ban/jail.local .
```

Štruktúra vlastných zákazov v súbore *jail.conf* má pre každú ústredňu nasledujúci tvar:

```
[nazov_ustredne]
enabled  = true
port     = 5060,5061
filter   = [nazov_podla_ustredne]
logpath  = /var/log/[cesta k log súboru ústredne]
maxretry = 3
action   = iptables-allports[name=nazov, protocol=all]
```

Názov filtru odpovedá názvu, ktorý má daná ústredňa. Podmienky detekcie v tomto súbore sú vytvorené každou skupinou vývojarov pre danú ústredňu. Základný filtračný súbor majú výrobcovia voľne uverejnený na svojich oficiálnych stránkach a je ho možné modifikovať a rozširovať o ďalšie podmienky. Filter obsahuje zápisy podmienok detekcie v tvare:

```
failregex = hladaná zhoda v logu ,
```

príkladom takého zápisu podľa zápisu v logu z obr. 6.70 je:

```
failregex = \[ERR\] sofia_reg.c:\d+ Invalid Authorization header! from ip <HOST> ,
```

kde časť <HOST> určuje IP odkiaľ správa prišla. Podľa tohto zápisu sa prehľadáva v logu Freeswitchu zadaný reťazec a pri nájdený zhody sa inkrementuje počítadlo až do hodnoty definovanej v poli *maxretry*. Toto pole teda definuje počet zhôd, ktorý po dosiahnutí danej hodnoty spôsobí zablokovanie danej IP adresy. Môže byť rozšírený parametrami *findtime* (čas ako dlho sa budú hľadať zhody) a *bantime* (čas ako dlho bude platiť blokácia danej IP adresy).

Názov log súboru za premenou *logpath,* odpovedá súboru danej ústredne. Do neho je potrebné pridať pre každú ústredňu parametre, podľa ktorých sa bude prehľadávať daný log, tie sú takisto uverejnené výrobcami ústrední. Dôležitou hodnotou je *action,* ktorá definuje použité pravidlo a protokol. Tie sú uložené v zložke:

```
/etc/fail2ban/action.d .
```

Súbory pre všetky testované ústredne sú takisto súčasťou prílohy. Nevýhoda tejto možnosti je, že administrátor si musí pre dané chybové hlášky vytvoriť vlastné pravidlá, čiže ich musí znova najprv odsledovať ako v predchádzajúcom zabezpečení, tento raz z logu. Základné konfigurácie od vývojarov ústrední sú zamerané na ochranu proti viacnásobným zlyhaniam v autentizácii (registration hijack útok) a dos útoku.

6.6.3 Zabezpečenie pomocou IDPS systému SNORT

Jedná sa o sieťový detekčný a prevenčný systém (pozri kapitola 3.5.4), ktorý je podporovaný na všetkých známych platformách. Hlavnou zložkou jeho funkcionality je detekčné jadro. To používa definované pravidlá k zamedzeniu ohrozenia. Takisto dokáže logovať svoju aktivitu a informovať cez upozornenia. Výkonosť jadra je ovplyvnená HW parametrami zariadenia, na ktorom je spustený a takisto aj zložitosťou pravidiel. Pravidlá sa môžu vzťahovať na všetky časti paketu, od IP hlavičky, cez transportnú po aplikačnú a nesenú záťaž. Snort systém má dva režimy: odchytávací (sniffer) a sieťový prevenčný (NIDS). Prvý analyzuje príchodzie dáta. Akonáhle je nájdená zhoda s nejakým pravidlom, ďalšiu zhodu už nehladá a vykoná definovanú akciu v rámci druhého režimu. [42]

Systém dokáže pracovať v rôznych konfiguráciach v závislosti od požiadaviek konkrétnej siete. Jednou z hlavných výhod je jeho kompatibilita s rôznymi firewallmi. K tomu

používa agenta SnortSam, ktorý beží na zariadení s firewallom. Agent je prepojený s manažovacím serverom, na ktorom je spustený Snort a podľa požiadaviek spolupracuje s firewallom k zamedzeniu podozrivých spojení.

Pravidlá sú tvorené v jednoduchom syntaxe, väčšinou na dĺžku riadku. Uložené sú vo vlastných súboroch definovaných v konfiguračnom súbore *snort.conf*. Skladajú sa z dvoch logických častí, hlavičky a možností. Hlavička obsahuje typ akcie, ktorá sa má pri zhode vykonať, protokol, zdrojovú a cieľovú adresu a port. Medzi existujúce akcie patria: *alert* (upozornenie a zalogovanie paketu), *log*, *pass* (ignoruje paket), *drop* (zablokuje a zaloguje paket), *reject* (to isté ako *drop*, ale ešte pošle TCP reset, ak je použitý TCP alebo ICMP port unreachable, ak je použitý UDP) a *sdrop* (zablokuje paket, ale nezaloguje). Ďalším dôležitým parametrom v pravidlách je znak orientácie premávky, tj. znaky <-, -> a <>. V pravidle definujú, ktorá adresa resp. obidve majú byť sledované.

Okrem toho sa dajú v konfiguračnom súbore definovať vlastné akcie a premenné, ktoré bude možné následne pri písaní pravidiel použiť. Takisto sa tam nachádzajú aj definície aktívnych tzv. preprocessorov, ktoré tvoria vlastne detekčné moduly pre jednotlivé aplikačné protokoly a časti ich záhlaví. Nakoniec je v konfiguračnom súbore definovaný aj logovací súbor a formát logovania (typicky xml).

Časť možností obsahuje zadefinované upozorňovacie hlášky a parametre paketu, ku ktorým bude hľadaná zhoda. Využívajú sa tu tzv. keywords definované v manuáli pre jednotlivé časti aplikačných protokolov (napr. sip_body). Parametre sa oddeľujú bodkočiarkou.

Príkladom pravidla pre útok 6.1.1 je:

```
drop udp $EXTERNAL_NET any -> $HOME_NET 5060 (msg:"FUZZING SIP UDP";
sip_header; content:"SIP/7.0"; classtype:misc-attack; sid:333; rev:1;),
```

kde dôjde k zahodeniu daného datagramu z definovaného externého rozsahu (premenná `$EXTERNAL_NET`) smerujúceho na port 5060 v privátnom rozsahu `$HOME_NET`, ak obsahuje v SIP hlavičke reťazec „SIP/7.0“. Do definovaného log súboru bude o tom zapísaný záznam s úvodným reťazcom „FUZZING SIP UDP“. *Classtype* popisuje všeobecnou značkou, podľa vývojármi definovaných kategórii, typ útoku. *Sid* je identifikačné číslo pravidla a *rev.* označuje číslo revízie daného pravidla. V konfiguračnom súbore snort.conf, sú teda definované rozsahy pre premenné rozsahov nasledovne:

```
var EXTERNAL_NET 192.168.20.240/28
var HOME_NET $eth0_ADDRESS
```

Použitie Snort systému ako prevencia k útokom sa viaže na poznanie štruktúry príchodzích správ. Po vytvorení pravidla s ohľadom na obsah jednotlivých typov útokov - fuzzing správy je následne takáto správa automaticky po prijatí na PC s ústredňami zahodená.

Systém je použitý aj ako prevencia voči DoS typu útoku, kde je jeho použitie popísané v príslušnej kapitole. Súbor snort.conf a súbory pravidiel s použitými premennými a pravidlami sú súčasťou prílohy v zložke „*snort files*“.

7 DoS záplavové útoky

Ako už bolo spomenuté v kapitole 3.4, útoky typu DoS (Denial of Service) sú zamerané na obmedzenie resp. znefunkčnenie služby. To je možné dosiahnuť viacerými spôsobmi, v tejto kapitole je to záplavou správ (z ang. flooding), ktoré sú falošné, poškodené alebo modifikované. Cieľom je zahltiť server požiadavkami natoľko, že prestane reagovať alebo aspoň začne nestíhať alebo robiť chyby pri spracovávaní regulérnych správ. Útoky môžu byť

zamerané na konkrétne časti ako systémové prostriedky (pamäť a procesor) alebo prvky siete mimo server ako zahltenie celej šírky prenosového pásma.

V nasledujúcich testoch boli realizované útoky s cieľom zamestnať procesor do takej miery až sa začnú vyskytovať chyby resp. zlyhanie ústredne. Najnáročnejšie typy správ na procesorový čas sú tie, ktoré potrebujú dodatočné komplexnejšie výpočty. Keďže ústredňa musí každú správu po prijatí analyzovať, je pri prijatí veľkého množstva za krátky čas, možné dosiahnuť jej výkonnostné limity.

Útoky boli uskutočnené pre protokol SIP s tromi najnáročnejšími typmi žiadostí INVITE, REGISTER a OPTIONS a pre protokol IAX2 správami NEW a REGREQ. Žiadosti boli generované z útočného rozsahu IP 192.168.20.242/28 a z rôznych zdrojových portov. Na generátori Spirent bol nakonfigurovaný test trvajúci 180 sekúnd, kde počas prvých 60 sekúnd dochádza k nárastu počtu správ za sekundu na požadovanú hodnotu. Tá bola zvolená postupne v realizovaných testoch na 10 000, 20 000 a napokon 50 000 správ za sekundu pre SIP. Prvotne bola overená početnosť 2000 správ za sekundu, ktorá však úroveň zaťaženia procesoru pre všetky testované ústredne držala v nízkych hodnotách (10% a menej). Preto boli následne realizované testy s vyššími početnosťami, ich porovnanie je vidieť v príslušných grafoch nižšie. Pre IAX2 boli početnosti zvolené nižšie nakoľko pri početnosti 10 000 správ za sekundu dochádzalo k spadnutiu ústredne alebo zamrznutiu systému a systémové prostriedky boli vyťažené na maximum.

Nutno podotknúť, že Spirent dané hodnoty generuje v kolísavých intervaloch a počet v danej sekunde nie je stabilný, ale pohybuje sa okolo danej hranice. Miestami však generátor z neznámych príčin počas testu mal výraznejší pokles v generovaní požiadavkov, čo sa prejavilo aj následne vo vykreslovaných grafoch.

Na systéme s ústredňou nebol spustený žiaden iný program okrem utility nmon, ktorá dokáže sledovať a zaznamenávať rôzne systémove štatistiky v čase. Sledovaný bol hlavne parameter vyťaženia procesoru v percentách. Na to bol použitý príkaz:

```
nmon -f -s2 -c 240
```

kde parameter –f určuje, že údaje sa budú zapisovať do implicitne vytvoreného log súboru v koreňovej zložke užívateľa, parameter –s určuje interval odoberania dát, tj. 2 sekundy a paratemer –c dobu trvania. Výsledné hodnoty boli importované do tabuľkového editoru a vygenerované do prehľadných grafických kriviek slúžiacich k názornému porovnaniu zaťaženia ústrední. Pred útokom sa zaťaženie procesoru pohybovalo na minime tj. okolo 3%.

Počas útoku bola overená schopnosť realizovať hovor cez ústredňu medzi dvoma klientami, čím bola kontrolovaná funkcionalita ústredne počas útoku. Súbory použitých útokov sú súčasťou prílohy.

7.1 Útok DoS s využitím SIP INVITE správ

K testu bola použitá štandardná INVITE žiadosť, ktorá mala nasledovnú štruktúru:

```
INVITE sip:1002@192.168.20.244 SIP/2.0
To: sip:1002@192.168.20.244
Contact: <sip:1003@192.168.20.244>
From: sip:1003@192.168.20.244;tag=234
Max-Forwards: 5
Call-ID: sdp01.ndaksdj9342dasdd
CSeq: 8 INVITE
Via: SIP/2.0/UDP 192.168.20.143;branch=z9hG4bKkdjuw
Content-Length: 155
Content-Type: application/sdp

v=0
o=mhandley 29739 7272939 IN IP4 192.168.20.244
s=-
c=IN IP4 192.168.20.244
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC
```

Z hľadiska výpočtu, musí ústredňa správu rozanalyzovať a overiť, či je daný úžívateľ dostupný, správu preposlať a iniciátorovi poslať odpoveď. Pri veľkom množstve správ sa preto pochopiteľne dajú očakávať komplikácie, napríklad že ústredňa nebude stíhať vykonať menované operácie s každou s príchodzích správ.

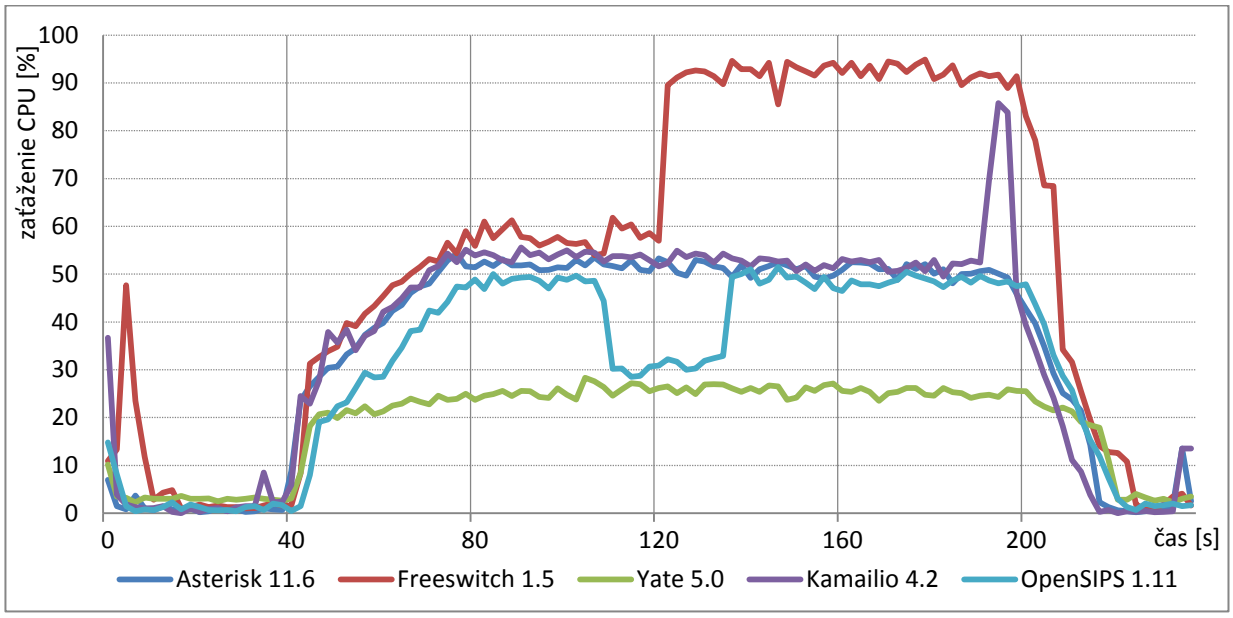
Po uskutočnení testov na Asterisk, je vidieť jeho viditeľné zlepšenie, čo sa týka zvládania veľkého množstva transakcií, oproti starším verziám v [37]. Hovor sa dal uskutočiť pri všetkých početnostiach útoku, no pri množstve 50 000 dochádzalo k asi sekundovému meškaniu spojenia.

Záťaž procesoru u Freeswitchu sa prejavovala dvojfázovo, zprvu narástla na určitú úroveň (prvých 60 sekúnd, kedy sa generátor dostal na požadovanú hodnotu transakcii za sekundu) a následne znova narástla na vyššiu úroveň, čo pri 50 000 správ za sekúnd úplne zaťažilo procesor. Hovor však bolo možné realizovať, aj keď s niekoľko sekundovým oneskorením.

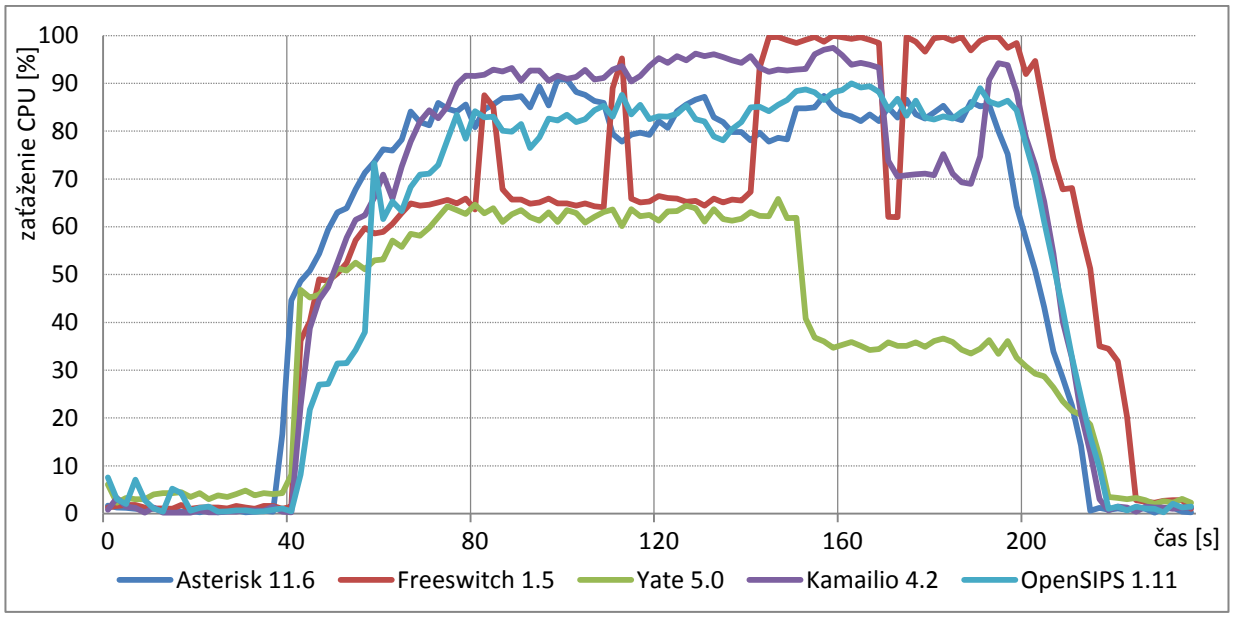
U Yate sa pri 10 000 správach za sekundu prejavilo oneskorenie na pokusnom hovore a pri vyšších početnostiach správ sa už nedal hovor realizovať, zaujímavosťou bolo, že procesor však nebol vyťažený na maximum. Po skončení útoku bolo možné hovor nadviazať okamžite. Dalo by sa teda znova konštatovať, že táto ústredňa má výrazné vývojové nedostatky, čo sa prejavilo aj v niektorých testoch s modifikovanými správami.

Kamailo mal síce procesor výrazne zaťažený a pri útoku s 50 000 správami dokonca na maxime takmer počas celého testu, ale napriek tomu sa hovor dal nadviazať bez oneskorenia.

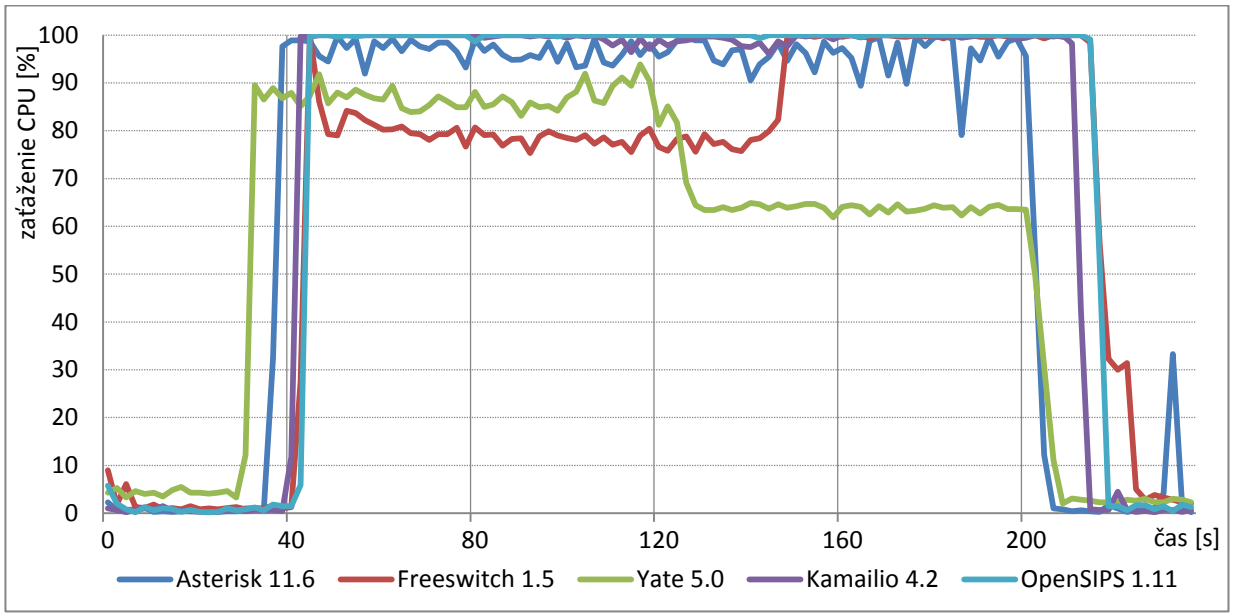
Ústredňa OpenSIPS nejavila žiadne problémy pre realizovaný hovor takisto, ani pri početnosti 50 000 správ. Záťaž procesora síce bola maximálna počas tohto útoku, ale na spoľahlivosť a funkcionalitu to nemalo výrazný vplyv. Takéto správanie posledných dvoch ústrední potvrdzuje aj ich výkonostný charakter, zameraný na spoľahlivé smerovanie veľkého množstva SIP správ vo vyšších stupňoch architektúry (chrbtová časť sietí).



Obr. 7.1: Zaťaženie CPU pri 10 000 INVITE/s



Obr. 7.2: Zaťaženie CPU pri 20 000 INVITE/s



Obr. 7.3: Zaťaženie CPU pri 50 000 INVITE/s

7.2 Útok DoS s využitím SIP OPTIONS správ

K realizácii útoku bola použitá štandardná OPTIONS správa ako v kapitole 6.1. Na ňu musí ústredňa reagovať zaslaním odpovede 200 OK.

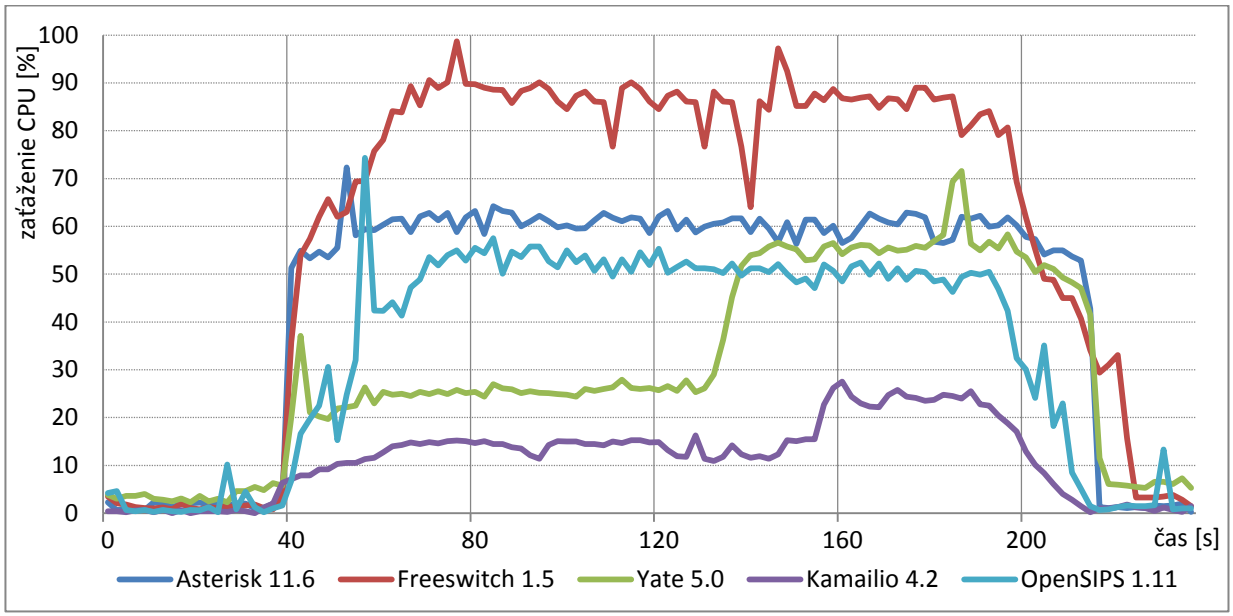
U Asterisku pri početnosti 20 000 správ dochádza k nárastu správ, ktoré ústredňa nie je schopná obslúžiť, čo sa prejavuje aj pri pokuse realizovať hovor, ktorý sa síce nadviaže, avšak s výrazným oneskorením. To je oproti predchádzajúcemu testu zmena smerom k horšiemu. Takisto aj pri početnosti 50 000 správ dochádza k zhoršeniu stavu a pri takmer maximálnom vyťažení počas útoku, už nie je možné uskutočniť hovor.

Ústredňa Freeswitch pri počte 10 000 správ za sekundu vyťažuje procesor na 80% a hovor sa dá realizovať bez problémov. Pri dvojnásobnej početnosti však dochádza k zaťaženiu procesora na úrovni 90% až maximum a to sa prejaví aj na realizovanom hovore, ktorý má niekoľkosekundové oneskorenie. Pri početnosti 50 000 už ústredňa nie je schopná hovor presmerovať, čo je rozdiel oproti útoku so správami INVITE, kedy sa hovor ešte realizovať dal, aj keď mal viacsekundové oneskorenie. Útoky s OPTIONS správami na túto ústredňu vyťažujú procesor najviac zo všetkých porovnávaných ústrední.

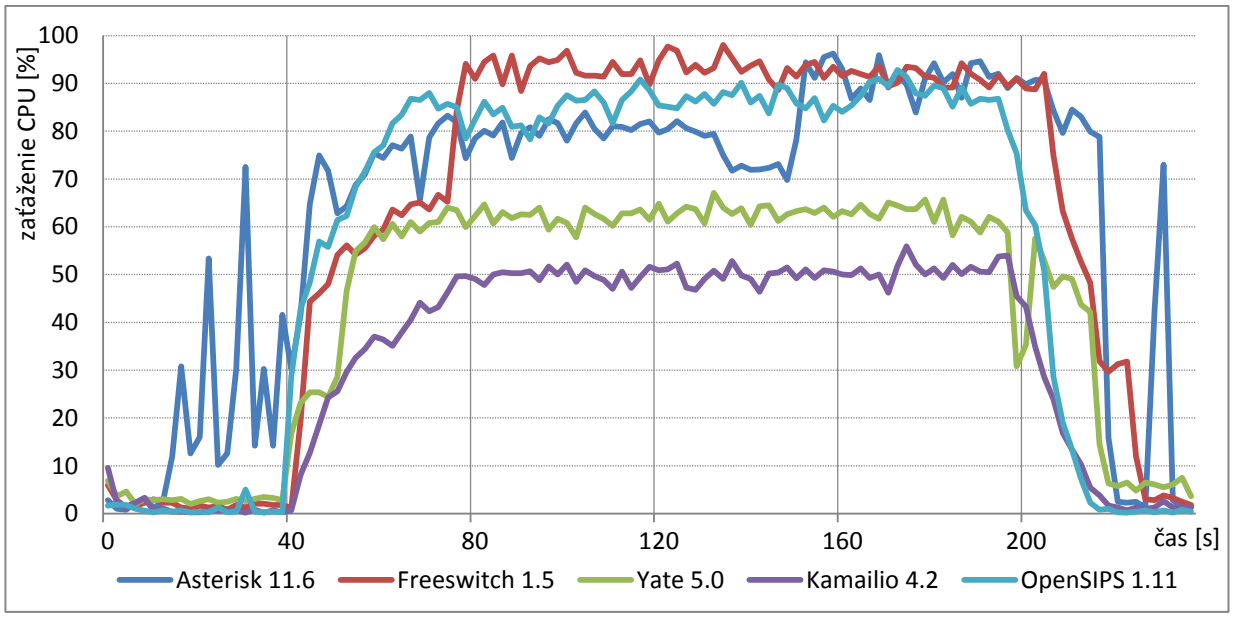
U ústredni Yate sa pri tomto type útoku nedal hovor realizovať už ani pri početnosti 10 000 správ za sekundu, na rozdiel od predchádzajúceho prípadu, kedy sa to dalo. To znova odporuje stavu zaťaženia procesoru, ktorý sa pohybuje na maximálnej úrovni okolo 60%, čo naznačuje, že táto ústredňa nemá efektívne implementované využívanie systémových prostriedkov a najmä procesoru.

Ústredňa Kamailio sa v tomto útoku javila ako najodolnejšia, čo sa týka vyčerpania systémových prostriedkov, ktorých úroveň dosahovala najnižšie hodnoty pri každej početnosti a zároveň spoľahlivo dokázala uskutočniť hovor.

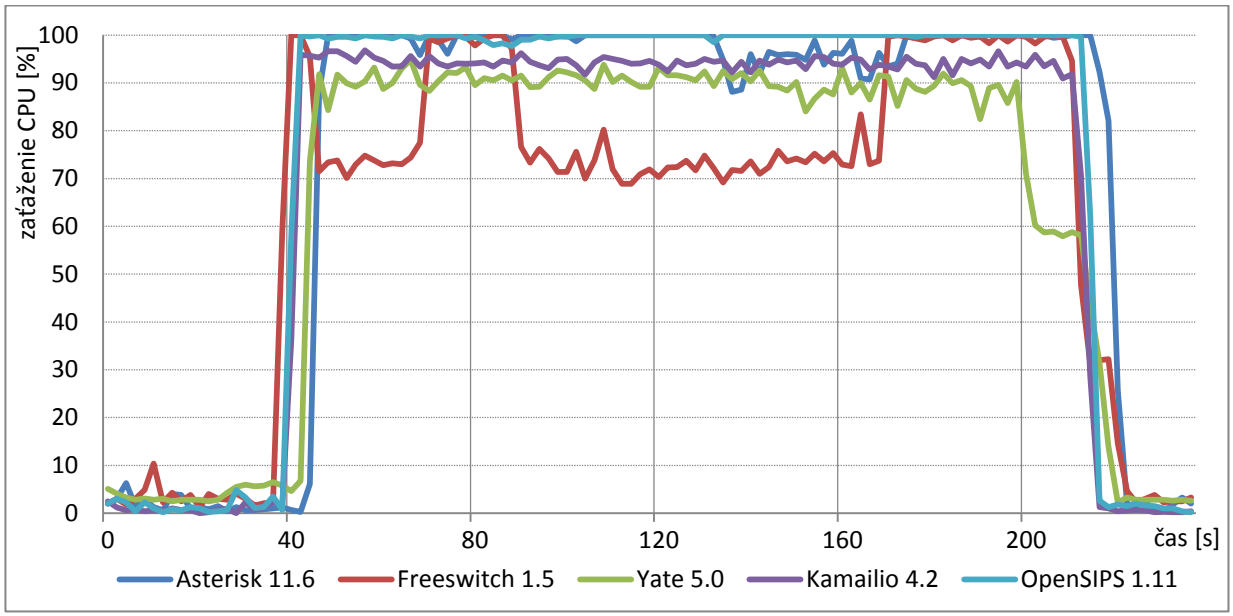
OpenSIPS rovnako ako Kamailio spoľahlivo zvláda hovor aj pri najvyššej početnosti prijmaných OPTIONS správ, a to aj napriek tomu, že záťaž procesoru je takmer konštatne na hranici 100% pri najvyššom teste.



Obr. 7.4: Zaťaženie CPU pri 10 000 OPTIONS/s



Obr. 7.5: Zaťaženie CPU pri 20 000 OPTIONS/s



Obr. 7.6: Zaťaženie CPU pri 50 000 OPTIONS/s

7.3 Útok DoS s využitím SIP REGISTER správ

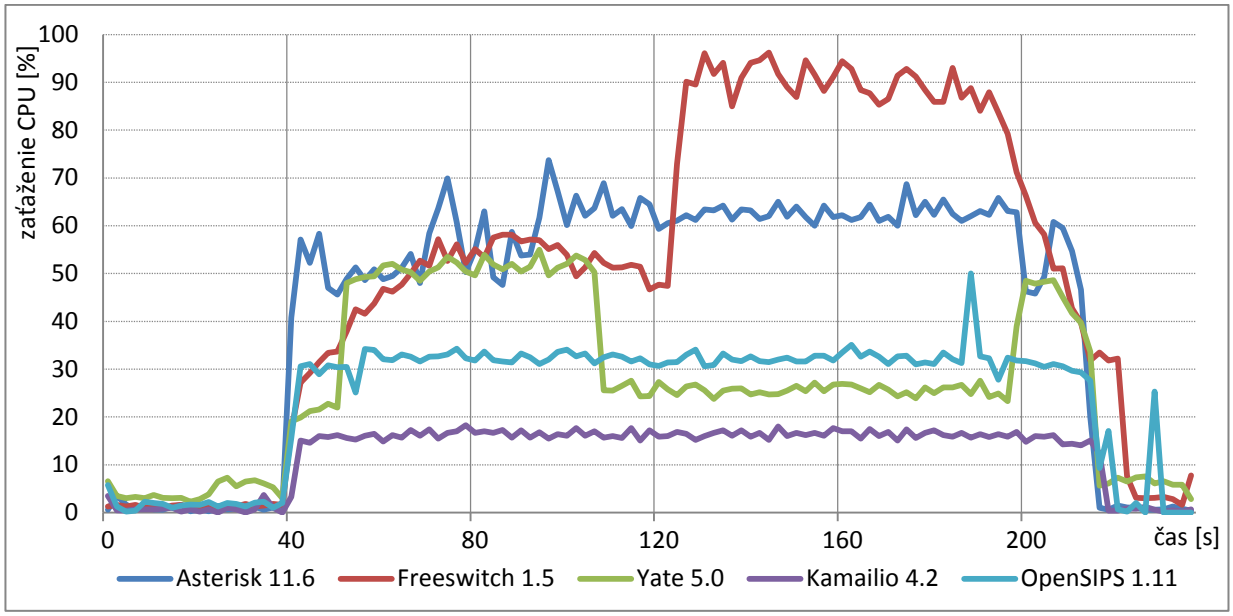
V tomto testovaní boli realizované útoky zasielaním štandardnej REGISTER správy ako je v kapitole 6.3. Ústredňa musí na každú žiadosť odoslať odpoveď, do ktorej musí najprv vygenerovať jedinečnú hodnotu *nonce* údaju. To predstavuje určitý nárok na prostriedky.

Asterisk najprv nemal problémy, až pri početnosti 50 000 správ za sekundu dochádza k výraznemu oneskoreniu hovoru a výraznej strate dát. Ani jeden z viacerých pokusov o korektne ukončený hovor sa počas tohto útoku nepodaril a dá sa teda konštatovať, že ústredňa už pri takomto množstve dát nezvláda fungovať funkčne.

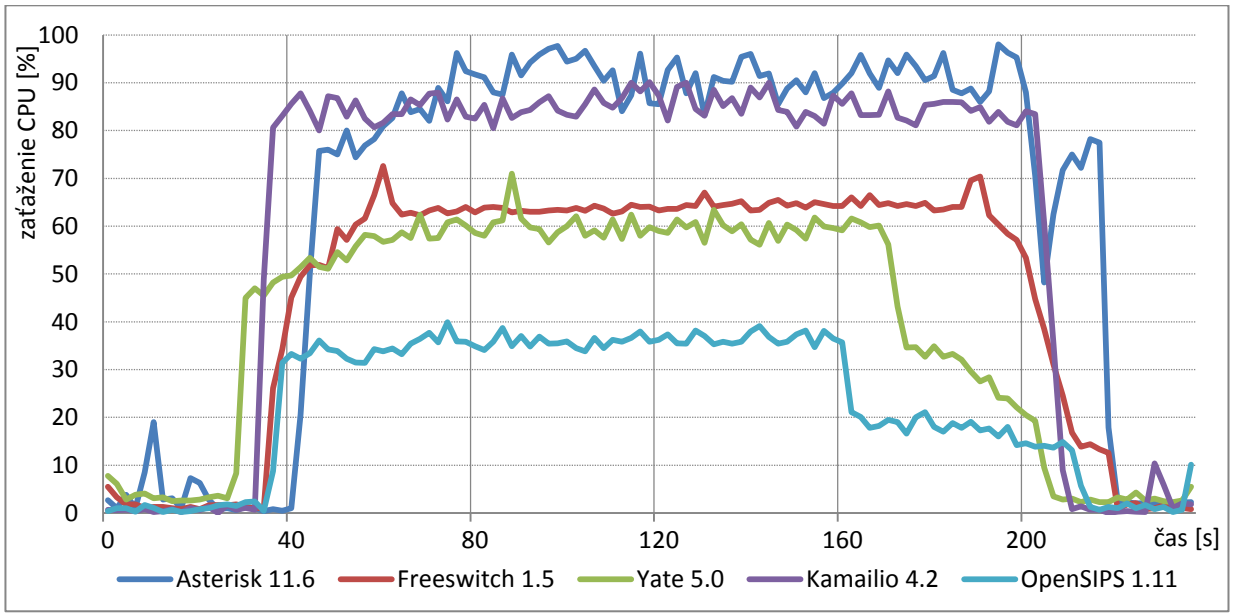
Freeswitch narozdiel od predchádzajúcich útokov dokázal nadviazať spojenie aj pri najvyššej početnosti správ, vtedy bol síce hovor čiastočne oneskorený, ale aj záťaženie procesora nedosahovalo vrcholné limity. Z toho vyplýva, že REGISTER žiadosti vie efektívne obsluhovať a to aj v porovnaní s ostatnými ústredňami.

U Yate sa potvrdilo, že záplavové útoky zvláda najhoršie spomedzi všetkých testovaných ústrední. Opakoval sa rovnaký stav ako doteraz a hovor sa nepodarilo nadviazať ani pri početnosti 10 000 správ za sekundu. Okamžite po skončení útoku však hovor spojí. Dodatočne bolo overené, že početnosť 2000 správ už zvláda, no stále je to výrazný rozdiel oproti ostatným ústredniam.

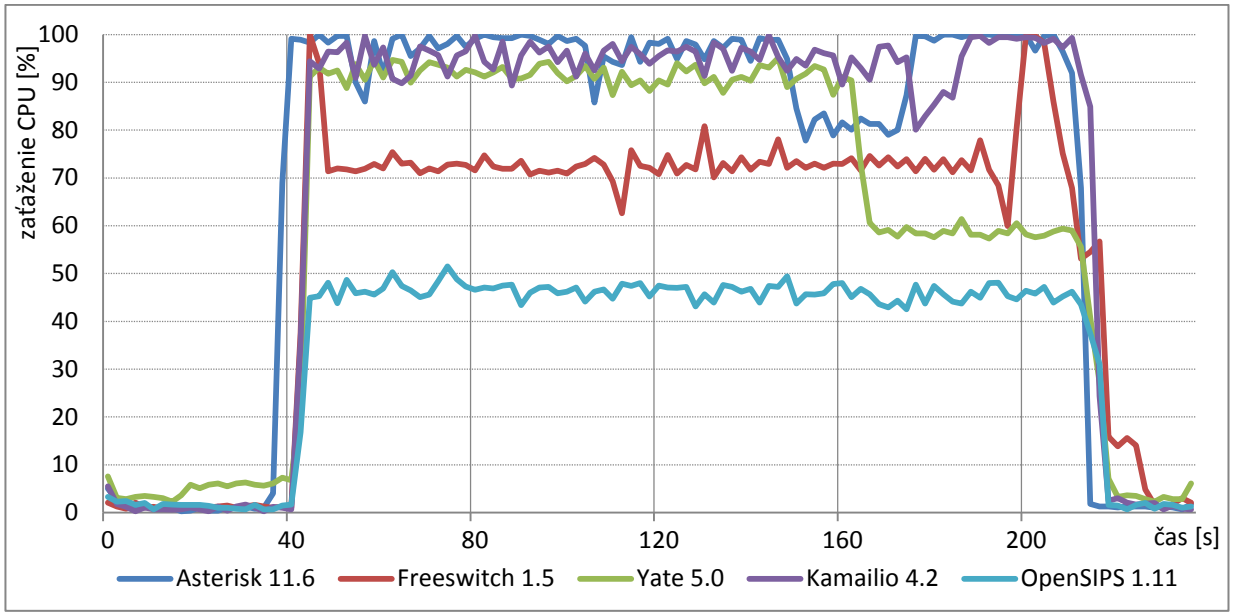
Prekvapujúce boli výsledky u ústrední Kamailio a OpenSIPS, ktoré počas testov nedokázali prenášať funkčný hovor ani pri najnižšej testovanej početnosti. Prvotné správy sa podarilo preniesť, pri incializácii sa však strácala nejaká potrebná správa, typicky ACK na potvrdenie odpovede podľa schémy inicializácie hovoru na obr. 6.26, čo značil volaný klient. Takéto reakcie však neodpovedajú záťaži procesora, najmä u OpenSIPS, kde sa záťaž pohybuje pri najvyšej početnosti okolo 60%. Z toho vyplýva, že spracovávanie množstva REGISTER správ nemajú tieto ústredne efektívne implementované, narozdiel od predchádzajúcich.



Obr. 7.7: Zaťaženie CPU pri 10 000 REGISTER/s



Obr. 7.8: Zaťaženie CPU pri 20 000 REGISTER/s



Obr. 7.9: Zaťaženie CPU pri 50 000 REGISTER/s

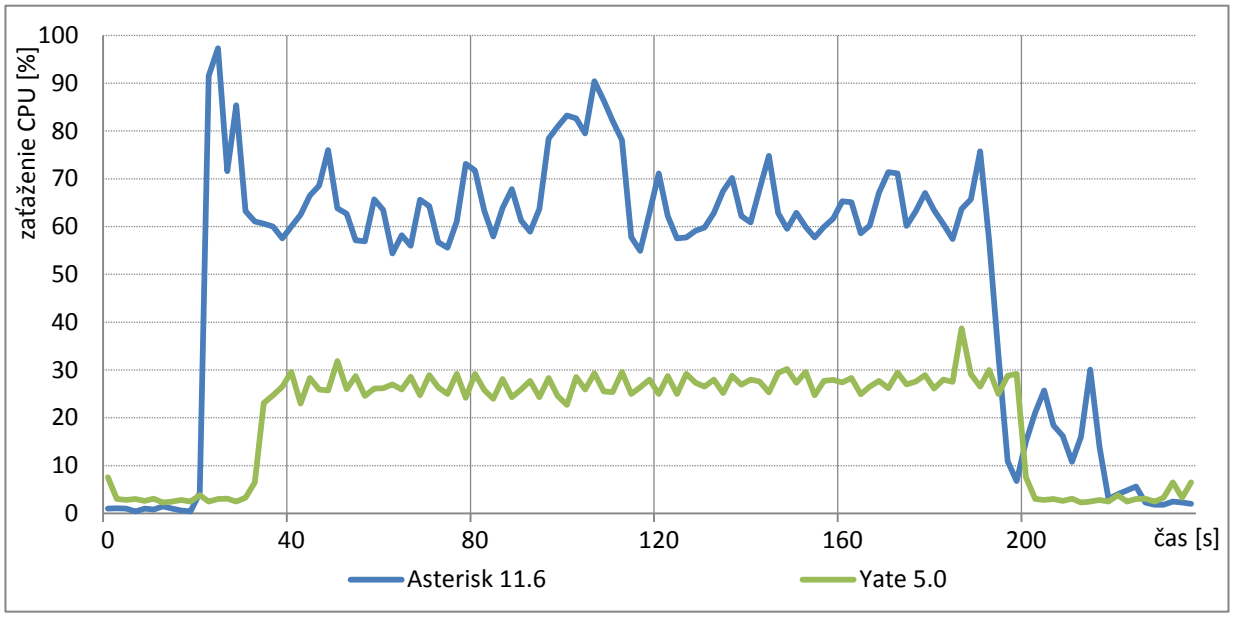
7.4 Útok DoS s využitím IAX2 NEW správ

Ako už bolo spomenuté, IAX2 podporujú iba dve testované ústredne. V útoku je prenášané veľké množstvo správ NEW. Na tie musí ústredňa poslať autentifikačnú výzvu v správe AUTHREQ a následne čakať na odpoveď, keď tá pochopiteľne od útočníka neprichádza posiela PING alebo LAGRQ správu na overenie konektivity. Nakoniec posiela REJECT správu, ktorou zamieta pôvodnú žiadosť. Celkovo teda ústredňa na každú NEW reaguje v priebehu času dvoma až troma ďalšími správami, čo jednoznačne vedie k rýchlejšiemu vyčerpaniu sytémových prostriedkov.

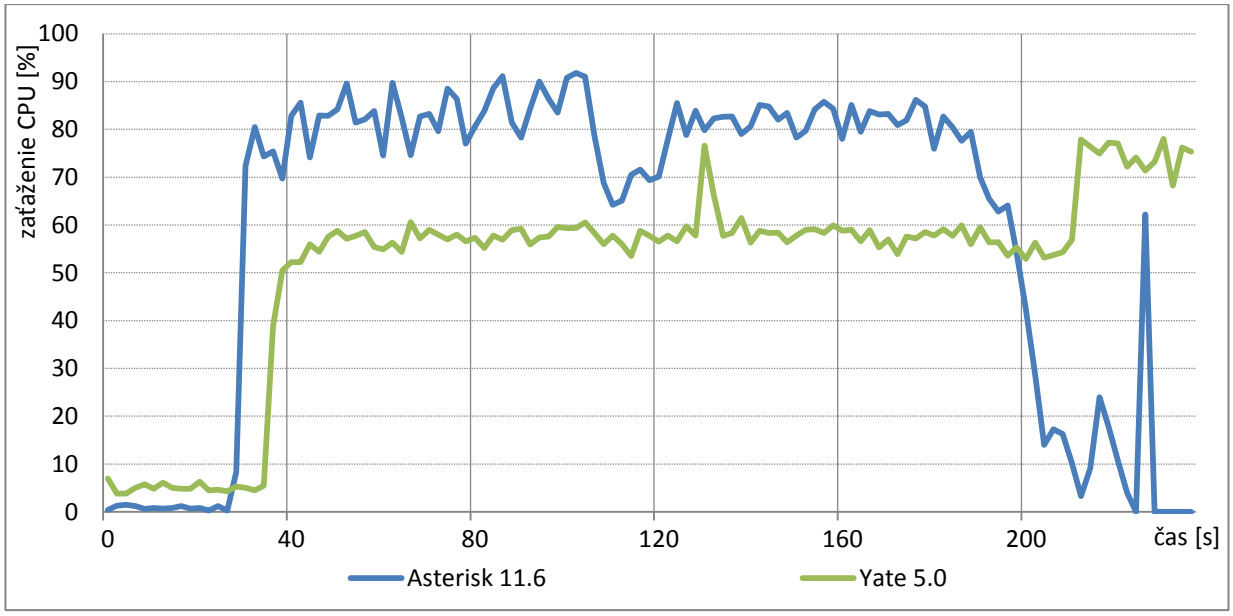
Útok bol postupne realizovaný v troch početnostiach od úrovne 1000 správ za sekundu po 10 000. Ústredňa Asterisk pri početnosti 1000 správ dokázala hovor realizovať s minimálnym oneskorením, pri 5000 správach za sekundu mala problémy s realizáciou hovoru a miestami dochádzalo k výraznej strate paketov, čo sa prejavilo napríklad pri zrušení hovoru jedným z klientov, ale druhý klient neobdržal HANGUP správu a hovor sa automaticky u neho ukončil po vypršaní implicitného hluchého intervalu. Záťaž procesoru sa pohybovala tesne pod úrovňou 90 %. Pri početnosti 10 000 správ už dochádza k zamrznutiu ústredne a dokonca k pádu *nautilusu* (obdoba *exploreru*, grafický správca súborov v Ubuntu). Procesor je vyťažený na maximálnej úrovni a celá stanica nie je počas útoku použiteľná.

Ústredňa Yate podobne pri 1000 správach neprejavovala problémy, hovor sa podarilo naviazať takmer okamžite, avšak pri početnosti 5000 správ za sekundu už dochádzalo k výraznej strate riadiacich rámcov a hovor už neprebiehal korektne. Zaujímvosťou je, že po skončení útoku nedôjde k zníženiu záťaže procesora, čo sa dá vysvetliť nekorektným ukončením a ovplyvnením ústredne po útoku natoľko, že nie schopná prostriedky korektne uvoľniť. Pri útoku s 10 000 správami sa softwarovým klientom nepodarilo ani k ústredni opakovane registrovať a hovor tak nebolo možné realizovať. Na druhej strane však počítač s ústredňou nespôsoboval funkčné problémy a ani zaťaženie procesoru nedosahovalo takej úrovne ako u Asterisku. Akonáhle útok skončil, klientom sa podarilo IAX2 hovor nadviazať takmer okamžite.

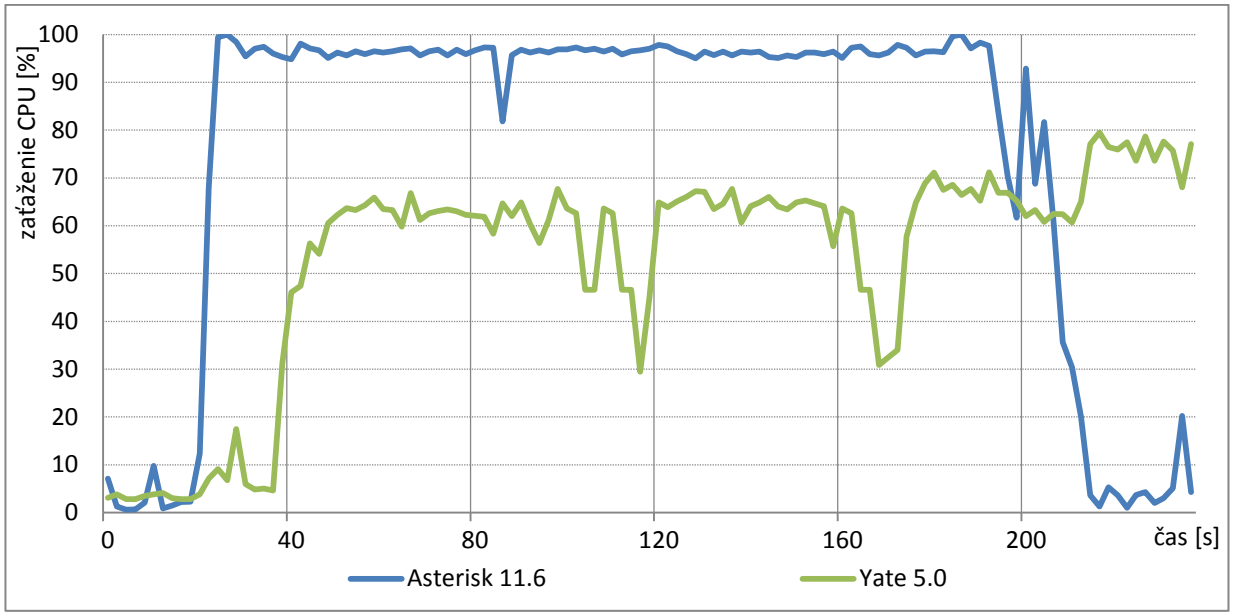
Z týchto výsledkov by sa dalo zkonštatovať, že obidve ústredne sú si veľmi podobné pri záplave IAX2 správami a Yate zvláda odbavovať požiadavky podobne efektívne ako u SIP protokolu, kde dokázal realizáciu postraného hovoru pri hodnote 2000 správ za sekundu.



Obr. 7.10: Zaťaženie CPU pri 1000 NEW/s



Obr. 7.11: Zaťaženie CPU pri 5000 NEW/s

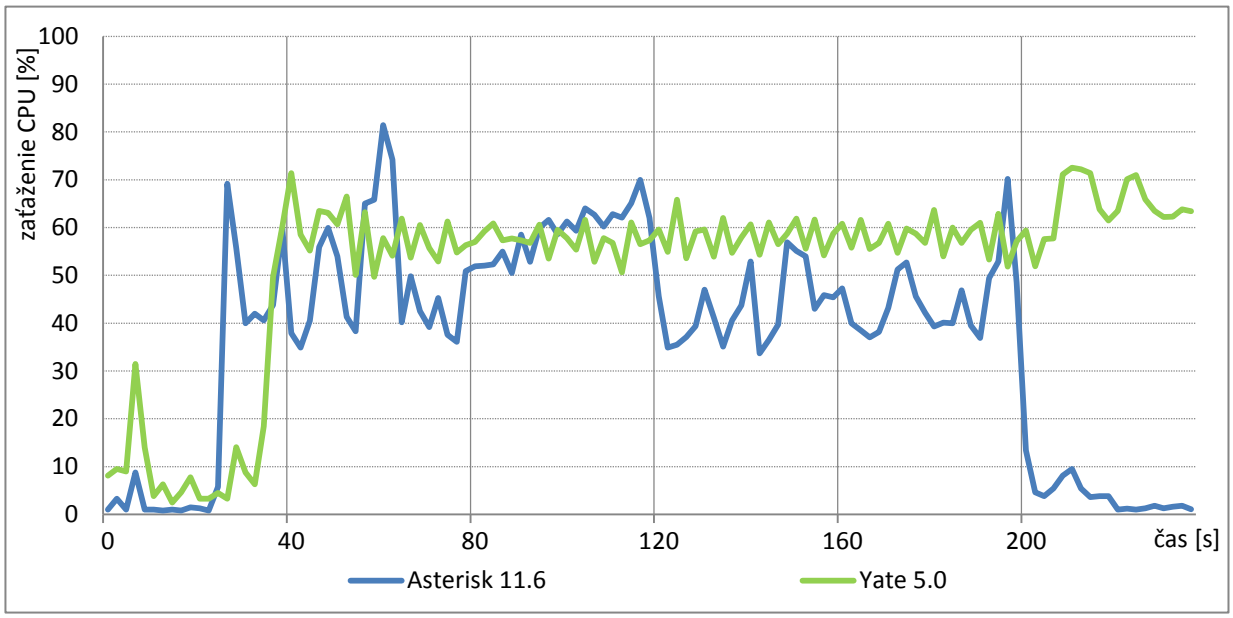


Obr. 7.12: Zaťaženie CPU pri 10 000 NEW/s

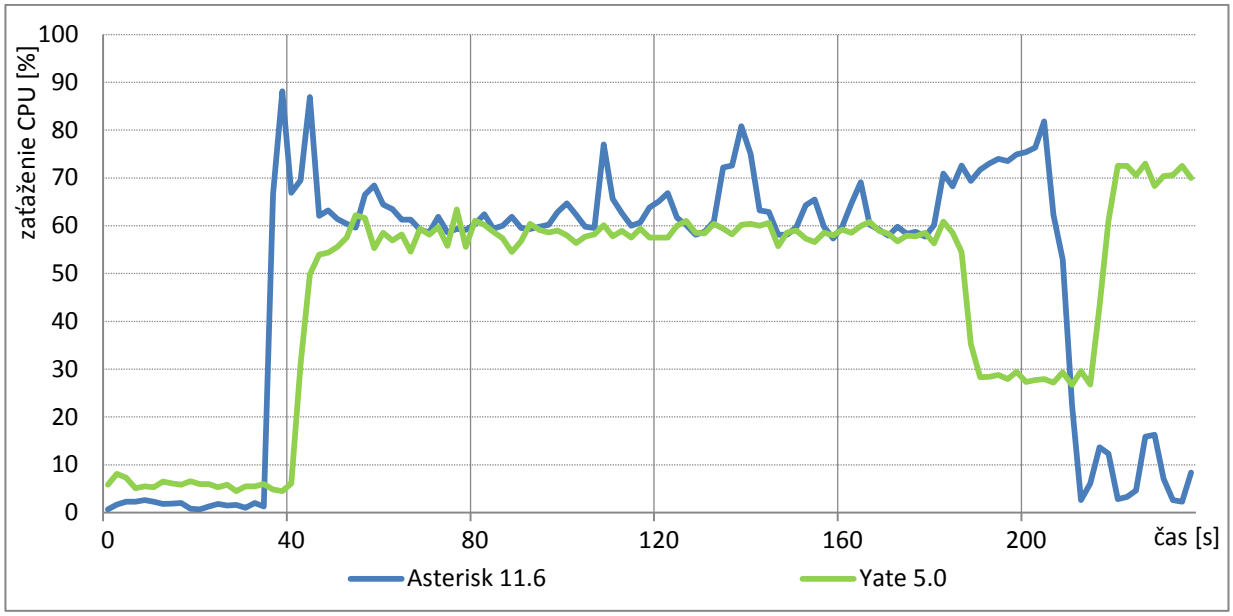
7.5 Útok DoS s využitím IAX2 REGREQ správ

Pri útoku bola posielaná štandartná IAX2 žiadosť o registráciu REGREQ v rovnakých početnostiach ako v predchádzajúcej kapitole. Na každú takúto správu musí ústredňa odpovedať správou REGAUTH, nakoľko vyžaduje autentifikáciu (implicitné nastavenie). V odpovedi posiela podporované metódy zabezpečenia. Nemusí do nej generovať žiadne hodnoty, čo znižuje nároky na systém.

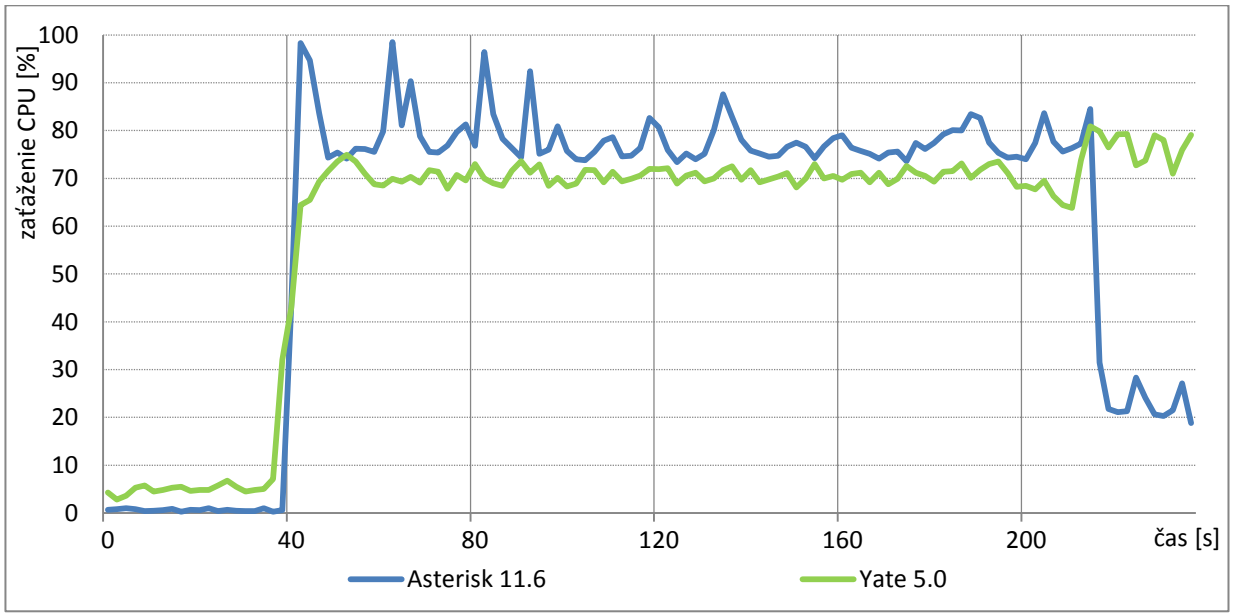
Ako je vidieť u Asterisku z výsledných záťaží a schopnosti uskutočniť hovor (tab. 7.2) aj pri najvyššej testovanej početnosti, ale až s niekoľkosekundovým oneskorením, potvrdzuje sa tým nižšia záťaž na systémové prostriedky než u správy NEW. U Yate však boli výsledky obdobné ako v testoch predtým a hovor bol funkčný len pri 1000 správach.



Obr. 7.13: Zaťaženie CPU pri 1000 REGREQ/s



Obr. 7.14: Zaťaženie CPU pri 5000 REGREQ/s



Obr. 7.15: Zaťaženie CPU pri 10 000 REGREQ/s

7.6 Zhrnutie výsledkov a návrh zabezpečenia

Z hľadiska odolnosti voči záplavovým útokom, ktorým bola venonavaná celá kapitola, boli výsledky vyhodnotené na základe sledovaných štatistík zaťaženia procesoru (grafy v príslušných podkapitolách) a ďalších systémových prostriedkov, ktoré sú súčasťou súborov v zložke *„DoS statistiky“* v prílohe na CD. Ďalším kritériom slúžiacim k vyhodnoteniu bola schopnosť realizovať hovor cez ústredňu počas útoku.

Pri podrobnejšom rozanalyzovaní výsledkov pre SIP sa ukázali najlepšie ústredne Kamailio a OpenSIPS, avšak s výnimkou útoku s pomocou REGISTER správ. Ten najlepšie zvládla ústredňa Freeswitch. V testoch sa ukázala variabilita efektivity využitia procesoru jednotlivých ústrední. Najnižšie zaťaženie procesoru dosahovala najčastejšie ústredňa Yate, avšak to vôbec neodpovedalo jej efektivite, ktorá bola veľmi zlá a hovor počas útoku nebol takmer nikdy (s výnimkou INVITE útoku) vytvorený. Naopak zaťaženie procesoru u Asterisku a Freeswitchu bolo výraznejšie, avšak aj schopnosť realizovať hovor bola najmä u Freeswitchu

veľmi dobrá aj pri najnáročnejších útokoch s najvyššou početnosťou zasielaných správ (vtedy už Asterisk zaostával a hovor nespojil). Zhrnutie schopností spojiť hovor počas útoku je farebne znázornené v tabuľke 7.1.

U IAX výsledky testov ukázali, že ústredňa Asterisk aj napriek výraznejšiemu zaťaženiu procesoru počas útoku dokáže hovor uskutočniť. Znova sa potvrzuje, že Yate sa v testovaní javí ako najhoršie odladená a napísaná ústredňa. Jej schopnosť uskutočniť hovor je veľmi slabá, aj keď zaťaženie procesoru nie je nijak vyrázné. Prehľadné zhrnutie pre obidve ústredne a všetky realizované útoky je v tabuľke 7.2.

Tab. 7.1: Zhrnutie schopnosti ústrední uskutočniť hovor počas útokov záplavou SIP

Typ protokolu a správy	ústredňa / početnosť za sekundu	Asterisk 11.6	Freeswitch 1.5	Yate 5.0	Kamailio 4.2	Opensips 1.11
SIP INVITE	10 000	• hovor OK	• hovor OK	• hovor oneskorený	• hovor OK	• hovor OK
	20 000	• hovor OK	• hovor oneskorený	• hovor neuskutočnený	• hovor OK	• hovor OK
	50 000	• hovor oneskorený	• hovor oneskorený	• hovor neuskutočnený	• hovor OK	• hovor OK
SIP OPTIONS	10 000	• hovor OK	• hovor OK	• hovor neuskutočnený	• hovor OK	• hovor OK
	20 000	• hovor oneskorený	• hovor oneskorený	• hovor neuskutočnený	• hovor OK	• hovor OK
	50 000	• hovor neuskutočnený	• hovor neuskutočnený	• hovor neuskutočnený	• hovor OK	• hovor OK
SIP REGISTER	10 000	• hovor OK	• hovor OK	• hovor neuskutočnený	• hovor veľmi nekvalitný	• hovor veľmi nekvalitný
	20 000	• hovor OK	• hovor OK	• hovor neuskutočnený	• hovor neuskutočnený	• hovor veľmi nekvalitný
	50 000	• hovor neuskutočnený	• hovor oneskorený	• hovor neuskutočnený	• hovor neuskutočnený	• hovor veľmi nekvalitný

Tab. 7.2: Zhrnutie schopnosti ústrední uskutočniť hovor počas útokov záplavou IAX2

Typ protokolu a správy	ústredňa / počtnosť za sekundu	Asterisk 11.6	Yate 5.0
IAX2 NEW	1000	• hovor OK	• hovor OK
	5000	• hovor veľmi nekvalitný	• hovor veľmi nekvalitný
	10000	• hovor neuskutočnený • zamrznutie ústredne	• hovor neuskutočnený
IAX2 REGREQ	1000	• hovor OK	• hovor OK
	5000	• hovor oneskorený	• hovor veľmi nekvalitný
	10000	• hovor oneskorený	• hovor neuskutočnený

7.6.1 Zabezpečenie pomocou IDPS systému SNORT

Na zabezpečenie proti záplavovým útokom bol znova použitý systém SNORT (pozri aj kap. 6.6.3). Jeho použitie je veľmi jednoduché a efektívne. Primárne sa využíva aj v praxi práve ako ochrana pred záplavovými útokmi.

Príkladom pravidla pre útok s OPTIONS správou je:

```
alert ip any any -> $HOME_NET 5060 (msg:"OPTIONS SIP flood";
content:"OPTIONS"; depth:7; threshold: type both , track by_src, count 100,
seconds 3; sid:5000004; rev:1; fwsam: src, 60 minutes;)
```

kde premenná *$HOME_NET* určuje sieť s ústredňou definovanú v snort.conf súbore. Ďalej je určený SIP port 5060 na ktorom sa komunikácia odpočúva. V zátvorke je určená správa, ktorou systém upozorní a zapíše informáciu do log súboru, parameter *depth* určuje počet bitov, po ktorý bude hľadaný v príchodzej správe reťazec OPTIONS (nahrádza *keyword* parameter). Parameter *treshold* určuje spôsob sledovania správ, tj. podľa zdroja a počíta 100 správ v priebehu 3 sekúnd. To je pri testovaných útokoch vyčerpané takmer okamžite. Nasleduje id parametru 5000004 a parameter určujúci SnortSam agentovi blokovať zdroj po dobu jednej hodiny.

Agent systému v spolupráci s firewallom zabránil na základe daného identifikátora pravidla komunikácii, informuje o tom sériou hlásení počnúc nasledujúcou správou.

```
2015/04/26, 11:21:11, 192.168.20.143, 2, snortsam, Blocking host
192.168.20.244 completely for 3600 seconds (Sig_ID: 5000004).
```

Ostatné pravidlá sú súčasťou prílohy v súbore local.rules. Po ich aplikovaní počas útoku dochádza k očakávanému a takmer okamžitému (vďaka pravidlu na sledovanie početnosti počas prvých 3 sekúnd) uvoľneniu systémových prostriedkov. Je možné realizovať hovor aj pri útoku s najvyššou početnosťou, nakoľko správy už k ústredni neprídu a sú zahodené firewallom.

7.6.2 Zabezpečenie IAX2 pomocou Call Token validácie

Prvou možnosťou, ktorá je implicitne aktívna je využitie validácie (kap. 2.3.5) pre IAX klientov, ktorí ju podporujú. V testoch boli útoky generované bez jej vyžiadania, avšak po jej vynútení v nastaveniach pomocou zmeny parametru (*calltokenoptional = no*), je jej použitie priamou a efektívnou metódou ako znížiť účinok DoS útoku na ústredňu. Implicitne nastavená hranica tzv. Call number limit je 8192 spojení, to je pri početnosti 1000 správ za sekundu vyčerpané v priebehu sekúnd a následne už len ústredňa automaticky ignoruje každú príchodziu správu v útoku a informuje o tom nasledujúcim log záznamom.

```
WARNING[42147]: chan_iax2.c:2947 __find_callno: No more space

WARNING[42141]: chan_iax2.c:2722 get_unused_callno: NON-CallToken callnumber
limit is reached. Current:8192 Max:8192
```

Nastavenie sa dá upraviť a ubidvoch ústrední sú s ním spojené ešte ďalšie možnosti ako limit pre spojenia, ktoré nepodporujú validáciu a nastavenie rozsahu adries, odkiaľ bude validácia voliteľná alebo povinná. Konfiguračné súbory, kde sa dá validácia nastaviť sú *iax.conf* pre Asterisk a *yiaxchan.conf* pre Yate a obdiva sú súčasťou prílohy.

8 Iné útoky

V tejto časti práce boli realizované ďalšie útoky s cieľom znefunkčniť reláciu (pozri. kap. 3.3). Postupne boli s testerom Spirent realizované útoky typu násilného ukončenia relácie, zrušenia registrácie klienta na ústredni a ukradnutiu hovoru. V testoch je podrobne popísané vytvorenie každého útoku cez aplikáciu Attack Designer, v čom útok spočíva a analýza správania ústredne. Nutno znova podotknúť, že aplikácia na tvorbu takýchto útokov poskytuje detailné možnosti nastavení jednotlivých hlavičiek zapúzdrených protokolov, je to však menej úžívateľsky prívetivé než iné voľne dostupné nástroje. Spirent tester dokáže síce generovať hovory avšak, aby bolo možné takéto hovory napádať je nutné poznať niektoré parametre a podľa nich vytvoriť útočný scénár a ten implementovať, to by však nebolo už možné ak beží scénár generácie hovoru. Preto bolo nutné na tvorbu hovoru použiť softwarové telefóny (Blink a PhonerLite) spustené na PC s IP 192.168.20.123 a s nakonfigurovanými účtami. Následne bol realizovaný hovor z jedného na druhý s prepojením cez ústredňu bežiacu na virtuálnom stroji s IP 192.168.20.143. Pre zachytenie potrebných parametrov hovoru je možné použiť sieťový analyzátor ako Ettercap alebo Wireshark. Následne parametre použiť k vytvoreniu útočnej správy v Attack Designeri a po uložení implementovať do testovacieho scenára v Spirente, ktorý následne dané správy generuje a posiela k cieľu. Nakoľko spustenie generácie testu vždy Spirentu trvá niekoľko málo desiatok sekúnd, a to aj pri zasielaní jednej správy, bolo by vhodnejšie použiť ľubovoľný voľne dostupný program alebo skript k dosiahnutiu toho istého.

8.1 Útok ukončenia relácie

Útok spočiva v útočníkovom poslaní žiadosti o zrušenie hovoru k ústredni. Pre vytvorenie takejto SIP BYE žiadosti je najprv potrebné z prebiehajúceho hovoru zachytiť parametre: *Call-ID, To* a *From*. Prvý unikátne identifikuje konkrétnu žiadosť k začatiu hovoru

(INVITE) a následne všetky požiadavky daného klienta tj. jednoznačne identifikuje reláciu UAC – UAS. Druhý logicky identifikuje cieľ a tretí menovaný informuje druhú stranu o zdroji, odkiaľ žiadosť prichádza. [15]

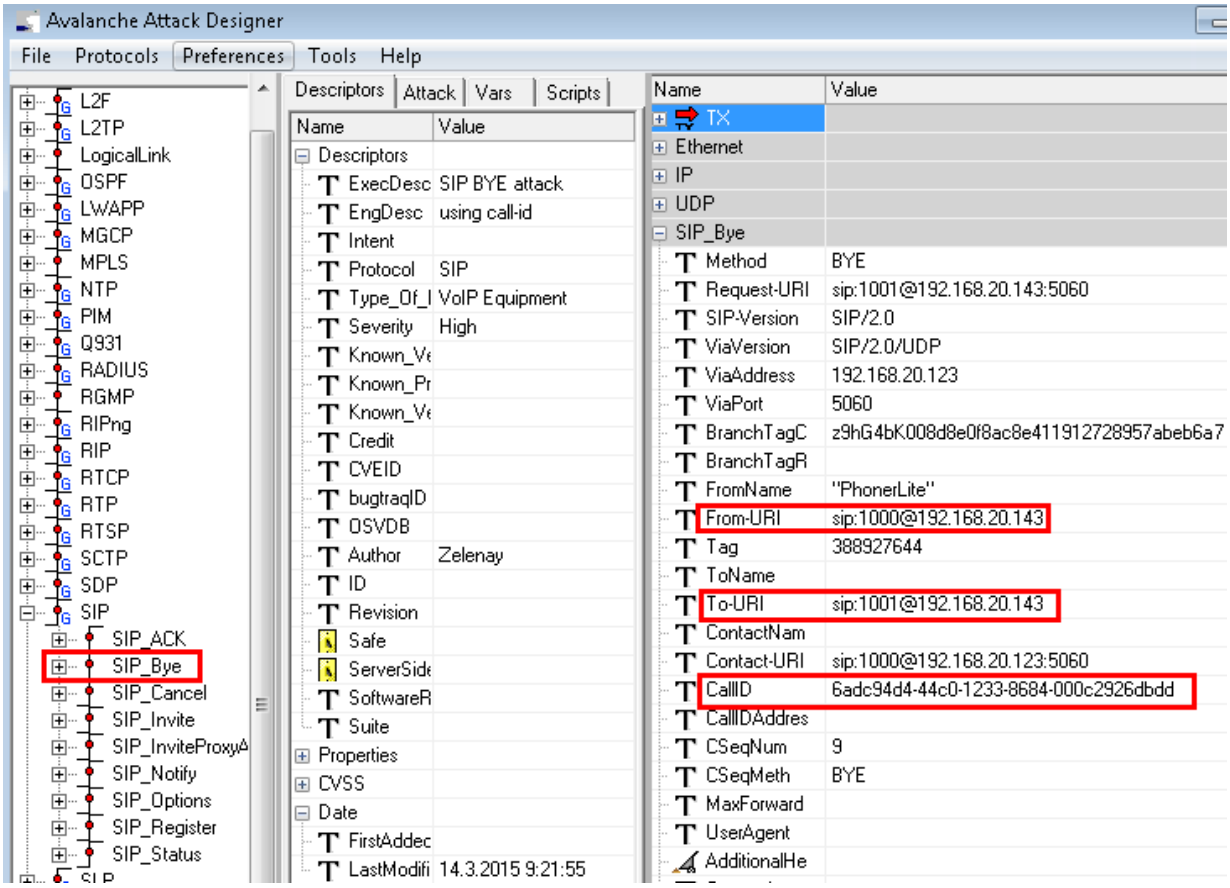
V tomto útoku bol na zachytenie potrebných údajov použitý Wireshark. Po vytočení čísla volaného klienta z telefónu Blink nasleduje štandardný inicalizačný proces hovoru (pozri obr. 6.26). Hneď prvotná správa INVITE obsahuje vyšie spomínané potrebné parametre. Tie sú použité v nasledujúcej tvorbe útočnej BYE žiadosti.

V Attack Designeri bola zo sady protokolov vybraná SIP_Bye šablóna (obr. 8.1). Následne bolo nutné vyplniť potrebné parametre hlavičky podľa zachytenej INVITE žiadosti. Po vypísaní základných popisných polí v sekcii *Descriptors* bol útok uložený a importovaný do zložky s útokmi. Nie všetky polia je povinné vypĺňať.

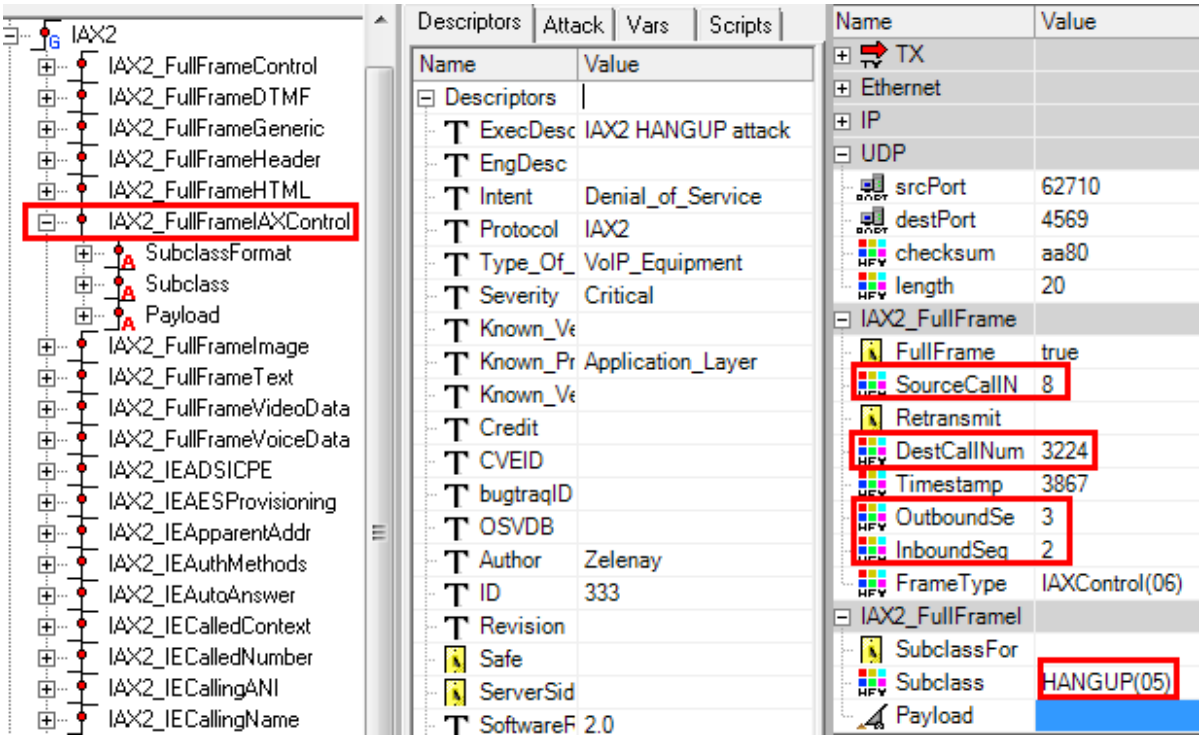
Pri zadaní nesprávnej hodnoty Call-ID (hodnota je citlivá na veľkosť písmena a je kontrolovaná po bitoch), ústredňa odošle informatívnu chybovú odpoveď *481 Call/Transaction Does Not Exist*. Útok je možné uskutočniť aj poslaním BYE žiadosti priamo druhému klientovi a nie ústredni. Je však nutné zaslať ho na správny port, na ktorom klient počúva.

Variácia tohto typu útoku pre protokol IAX2 spočíva takisto v sledovaní hovorov sieťovým analyzátorom. Konkrétne je potrebné zachytiť správu ANSWER, PING alebo PONG. Každá z nich v sebe nesie parametre spojenia a to zdrojové a cieľové označenie hovoru (S-DCID) a sekvenčné čísla (iseq a oseq). Prvé dva jednoznačne identifikujú hovor a sú teda stanovené obidvoma stranami pri jeho inicializácii, druhé dva sa inkrementujú pri vymienaní rámcov počas hovoru a slúžia k určovaniu správneho poradia prijatých paketov. Následne po zachytení potrebného ANSWER rámca z hovoru bol v Attack Designeri vytvorený IAX2 HANGUP rámec s danými parametrami hovoru (pozri obr. 8.2). Ako už bolo spomenuté v úvode kapitoly, inicializácia testu generátoru však trvá niekoľko málo desiatok sekúnd, preto je nutné aby hovor trval dostatočne dlhú dobu. Dá sa teda znova skonštatovať, že na takýto typ útoku je vhodnejšie použiť nejaký jednoduchý dynamický nástroj (napr. IAXhangup), s ktorým by takýto útok trvalo zrealizovať zopár sekúnd a hlavne automaticky.

Ochranou proti tomuto typu útoku je použitie protokolu TCP miesto UDP, nakoľko ten vytvára spoľahlivé spojenie a využíva aj mechanizmus sekvenčných čísel. Ešte lepšou ochranou je použitie šifrovaného TLS (pozri kap. 2.5.2), tým by bolo útočníkovi úplne znemožnené zachytiť potrebné parametre (ako je vidieť v teste odposluchu, kap. 4.6).



Obr. 8.1: Tvorba útočnej SIP BYE žiadosti cez Attack Designer



Obr. 8.2: Tvorba útočnej IAX2 HANGUP cez Attack Designer

8.2 Útoky odstránenia a ukradnutia registrácie

Jedná sa o útok, ktorým sa dosiahne zrušenie záznamu v lokačnej službe v rámci logickej role registrar server v ústredni. Klient, ktorého registrácia bude odstránená nebude môcť prijímať hovory, nakoľko ústredňa nebude mať o ňom registrovaný lokačný záznam.

V SIP protokole existujú dva možné stavy, jeden s deaktivovanou autentizáciou a druhý s aktívnou. V prvom prípade sa odregistrovanie dá dosiahnuť pomocou žiadosti, ktorá slúži k registrácii tj. REGISTER. Tá vždy obsahuje dôležité povinné pole *Expires*, ktoré udáva časovú hodnotu platnosti registrácie (po jej uplynutí aktívny klient znova posiela žiadosť o re-registráciu). Pri zaslaní žiadosti s nulovou hodnotou tohto poľa je klient odregistrovaný. [23]

Ak je používaná autorizačná schéma (je možné nastaviť na každej z ústrední), ktorá implicitne býva aktívne prednastavená, je potrebné prekonať Digest autentizáciu (kap. 3.5.1). Tá v sebe prenáša zahešované heslo, ktoré je možné rozlúštiť napr. slovníkovou metódou nejakého programu na to určeného (napr. SIPcrack alebo Cain&Abel).

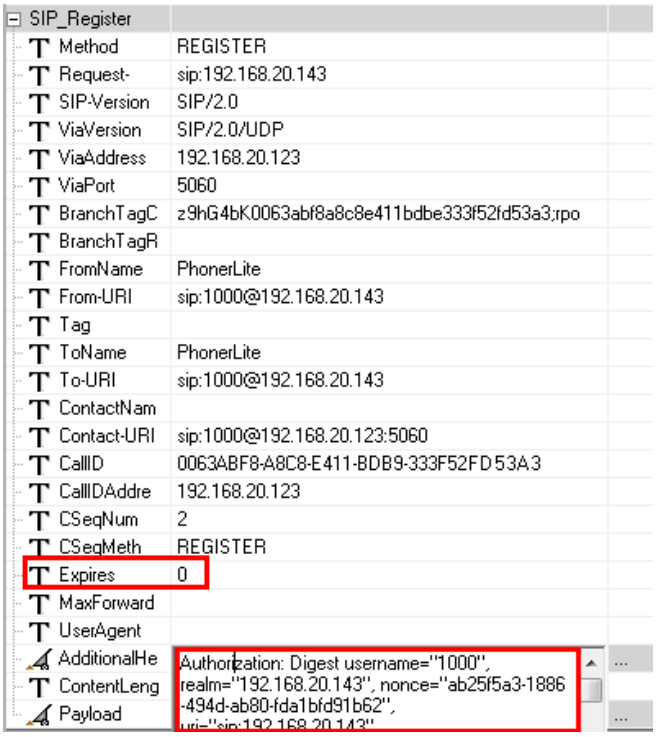
Pri odposluchu existuje iná jednoduchšia možnosť k dosiahnutiu odregistrovania klienta. Jediné čo je potrebné je zachytenie druhej REGISTER žiadosti v registračnom procese (pozri obr. 6.57), ktorá v sebe nesie autentizačný reťazec v tvare:

```
Authorization: Digest username="1000", realm="192.168.20.143",
nonce="ab25f5a3-1886-494d-ab80-fda1bfd91b62",
uri="sip:192.168.20.143",
response="e54899f6296ceab0877cc56fb29e7cdf", algorithm=MD5,
cnonce="0063abf8a8c8e411bdbd333f52fd53a3", qop=auth, nc=00000001
```

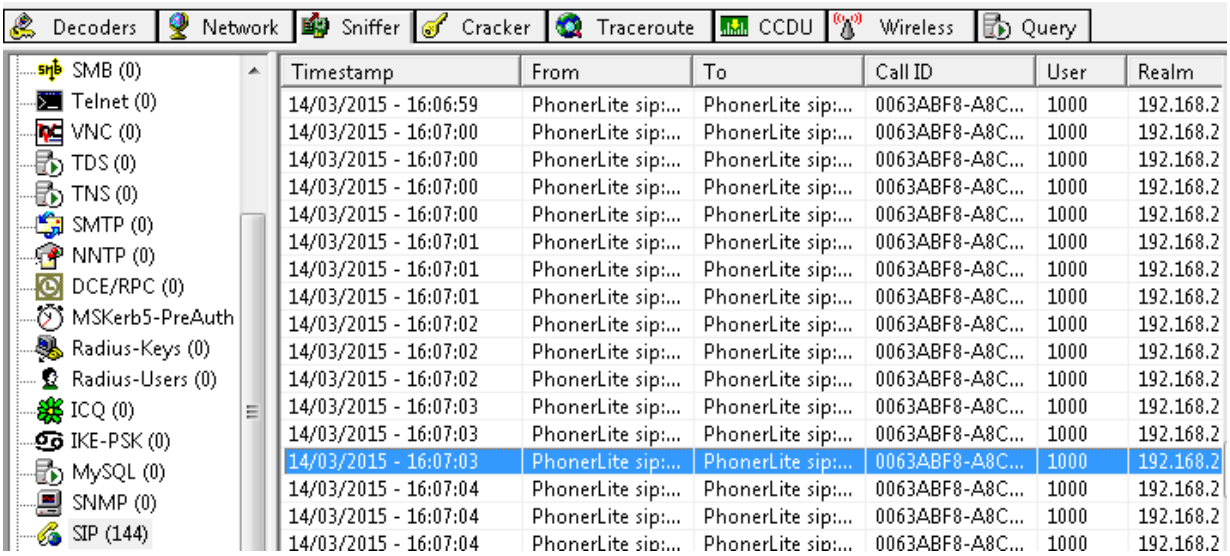
A následne tento reťazec použiť vo vytvorenej falošnej REGISTER žiadosti. Takisto je potrebné zmeniť hodnotu poľa *Expires* a inkrementovať hodnotu *C-Seq*. Útok v podstate vychádza z tzv. replay útoku, nakoľko bola použitá identická časť správy, aj keď samotná zahešovaná hodnota nie je známa. Samotná realizácia útoku spočíva v zachytení sieťovým analyzátorom (bol znovu použitý Wireshark) žiadosti REGISTER od klienta, ktorého chceme odregistrovať a následným vytvorením útočnej žiadosti. Tu bola využitá schopnosť Attack Designeru importovať zachytený pcap súbor (pozri kap. 5.2.1), podľa ktorého vytvorí útočnú správu. Stačí len modifikovať pole *Expires* na nulovú hodnotu (obr. 8.3) a útok nastaviť do realizovaného scenára v Spirent testeri. Následnymi testami sa zistilo, že túto metódu však majú už dnes všetky testované ústredne ošetrenú a pokiaľ je na nich nastavená aktívna autentifikácia, na každú novú žiadosť REGISTER posielajú odpoveď s nutnou autentifikáciou, kde posielajú novú hodnotu *nonce*. To predpokladá, že ústredne si sledujú štruktúry prijatej žiadosti a nielen jej číselné parametre (došlo k zmene poľa *Expires*). Útočníkovi teda nestačí len skopírovať autentizačný reťazec, ako je uvádzané v starších zdrojoch [2], [37] a [39]. Správna implementácia Digest autentifikačného mechanizmu vývojármi ústrední, je teda prvým krokom k zlepšeniu ochrany pred týmto typom útoku.

Ak útočník nepozná heslo registrovaného účtu, zostáva mu zložitejšia možnosť a to využitie nástrojov na rozlúštenie hesla. Po zaslaní prvej žiadosti o odregistrovanie, jen nutné získať z autentizačnej odpovedi 401 Unauthorized hodnoty k odpovedi. Na to bol použitý spomínaný program Cain&Abel, ktorý slúži k lámaniu hesiel, či už slovníkovou metódou alebo brute-force útokom. Jedná sa o poslednú aktuálnu verziu 4.9.56 z konca roku 2014. Na obrázku 8.4 je vidieť jeho rozhranie a zachytená SIP komunikácia. Z nej je zvolený príslušný paket, ktorý nesie hodnotu hesla, ten je následne dešifrovaný a výsledkom je heslo a aj ďalšie potrebné údaje (pozri obr. 8.5), vrátane hodnoty *response*, ktorú je potrebné poslať v druhej žiadosti ako reakciu na autentifikačnú odpoveď ústredňe 401. Tento proces teda vyžadoval s testerom Spirent, spustenie dvoch testovacích scenárov. Prvý, ktorý poslal inicializačnú žiadosť, ktorá slúžila k zachyteniu odpovede a následnému rozlúšteniu heša a vytvoreniu response parametru. Ten bol implementovaný do druhej REGISTER žiadosti, ktorú obsahoval

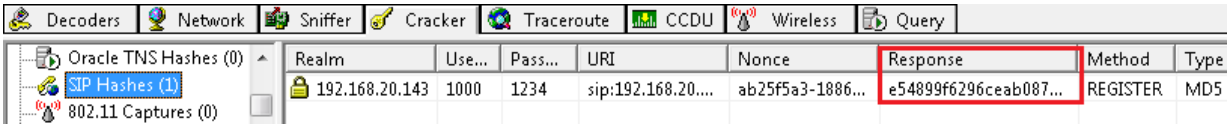
druhý spustený scénár. Takýto postup je pochopiteľne veľmi nepraktický a Spirent k realizácii takéhoto útoku nie je efektívny nástroj (oproti iným voľne dostupným riešeniam ako SiVuS).



Obr. 8.3: Žiadosť REGISTER na odregistrovanie klienta 1000



Obr. 8.4: Zachytenie SIP správ s potrebnými zahešovanými parametrami v programe Cain&Abel



Obr. 8.5: Rozšifrované potrebné údaje

Pre IAX2 je situácia podobná, nakoľko jeho autentifikačný mechanizmus (kap. 2.3.4) pri odregistrovávaní býva implicitne aktívny, je nutné použiť nástroje k vytvoreniu potrebného hešu. Pochopiteľne na internete nie sú dostupné tak vyspelé nástroje ako pre SIP ale medzi použiteľné patria IAX2brute alebo enumIAX. Samotná realizácia spočíva teda v zachytení

správy REGAUTH správy od ústredne, kde je uvedený heš a *challange* údaj. Z nich je pomocou IAX2brute dešifrované heslo a vygenerovaný odpovedajúci heš, ktorý je skopírovaný do útočnej správy poslanej k ústredni.

V takomto útoku môže byť ochranou dostatočne silné heslo, ktoré by útočníkovi trvalo rozlúštiť neúmerne dlhú dobu alebo znova použitie TLS šifrovacieho protokolu na signalizačné dáta u SIP.

V útoku ukradnutia registrácie je princíp podobný, tu je potrebné zmeniť hodnotu poľa *Contact,* tá ústredni udáva, kam majú byť smerované požiadavky na inicializáciu hovoru. Útočníkovi stačí použiť rovnakú falošnú žiadosť ako v predchádzajúcom prípade, je však nutné zmeniť hodnotu *expires* na časový úsek, počas ktorého bude platná registrácia (napr. bežných je 3600 sekúnd) a hlavne zmeniť spomínanú kontaktnú adresu *Contact-URI* za útočníkovu. Všetky hovory smerované na pôvodného klienta sú následne presmerovávané na útočníkovu adresu (čo môže predstavovať ďalšie bezpešnostné riziká).

Obmenou týchto útokov sú útoky realizované na klienta a nie priamo na ústredňu (napr. útok odmietnutia registrácie zaslaním REGREJ v IAX2 protokole). Podmienkou však je doručiť útočnú správu koncovému užívateľovi skôr než mu príde štandardná správa od ústredne. Takéto útoky však nie sú zamerané na testovanie odolnosti ústrední vočí útokom a preto nie sú predmetom práce.

Záver

Cieľom práce bolo analyzovať chovanie ústrední pod vplyvom útokov a navrhnúť zabezpečenie. Z hľadiska možností boli útoky koncipované do primárnej časti – testovania útokov s modifikovanými správami, a sekundárnej – testovaniu záplavových a ďalších typov ako útoky zrušenia hovoru a odregistrácie klienta.

Útoky zahrnuté v licencii zariadenia Spirent Avalanche, mali charakter modifikovaných a neštandardných SIP signalizačných správ. Skúmaniu reakcií na jednotlivé typy útokov, ktorých je niekoľko desiatok, boli podrobené všetky ústredne v LTS verziách, ktoré reagovali velmi často odlišne. Tu sa ukázala variabilita ich vývoja, zamerania a koncepcie. Jednotlivé útoky sú rozanalyzované vždy v samostatnej kapitole. Reakcie ústrední sú doplnené výpismi logovacích a diagnostických správ, výstupmi zo sieťového analyzátora a vysvetlujúcimi komentármi. Na konci nasleduje prehľadné a stručné zhrnutie dôležitých bodov v tabuľkách a grafoch.

V rámci primárnej časti práce vo výsledkoch pre SIP protokol najlepšie z hľadiska diagnostiky raegujú ústredne Kamailio a Opensips, čo potvrdzuje aj ich výslovné zameranie na SIP protokol a orientácia na SIP proxy server – smerovač. Ich logy a diagnostické výstupy sú najbohatšie a podrobne poukazujú na chybu, nesprávnu (z hľadiska RFC štandardu) položku v hlavičke správy. Takisto štruktúra ich logovacích správ a spôsobu tvorenia odpovedí je veľmi podobná, miestami priam identická. Z toho je vidieť, že obidve ústredne boli kedysi vyvíjané ako jeden projekt, ktorý sa neskôr rozpadol. Z pohľadu reakcií sa obidve ústredne držia štandardmi stanovenými pravidlami, pričom Kamailio na rozdiel od Opensips ani raz nereagoval zmätenou alebo neočakávanou odpoveďou. Ústedňa Freeswitch sa v realizovaných testoch javí ako najlepšia alternatíva k multifunkčným ústredniam Asterisk a Yate. Jej chovanie je dokonca veľmi podobné ako u na SIP špecializovanej ústredne Kamailio. Reakcie Asterisku sú najviac liberálne a častokrát tak jej nezamietnutie modifikovanej správy predstavuje bezpečnostné riziko. Najhoršia sa však ukázala ústredňa Yate, ktorá sa často prejavovala neočakávanými odpoveďami a niekoľkokrát sa dostala do zacykleného stavu pri generovaní odpovedí.

Pre IAX2 protokol, ktorý podporujú iba dve ústredne – Asterisk a Yate reagovala druhá menovaná výrazne lepšie než v testoch so SIP protokolom a z hľadiska bezpečnosti v dvoch prípadoch dokonca jej reakcie boli vhodnejšie než u ústredne Asterisk. Nasledujúce navrhnuté spôsoby zabezpečenia berú do úvahy charakter správ a pri využití Snort prevenčného systému poskytujú efektívnu ochranu.

V sekundárnej časti sa pri záplavových útokoch s protokolom SIP potvrdilo veľmi dobré využitie systémových prostriedkov u ústrední Kamailio a OpenSIPS, ktoré dokázali realizovať hovor aj pri najvyšších testovaných početnostiach správ za sekundu. Výnimkou boli však útoky s REGISTER správami, kedy nedokázali skušobný hovor počas útoku realizovať vôbec. Veľmi dobrú odolnosť mala aj ústredňa Freeswitch tesne nasledovaná Asteriskom, s ktorým sa vo výsledku realizovať hovor počas testu líšila iba pri útoku s REGISTER žiadosťami a najvyššej početnosti. Znova sa opakovali výsledky, kedy ústredňa Yate mala najhoršie reakcie a nedokázala realizovať funkčný hovor počas útokov ani pri nižších početnostiach, kedy to ostatným ústredniam problém nerobilo. Implementované zabezpečenie s využitím Snort sytému poskytuje efektívnu ochranu voči záplavovým útokom. U IAX2 protokolu je vhodnou ochranou aj použitie call token validácie, ktorá je súčasťou štandartu.

Uvedená analýza je záverečným zhrnutím vysporiadania sa ústrední s testovanými útokmi a na základe výsledkov viac ako 150 testov práca poskytuje komplexné vyhodnotenie bezpečnostných schopností jednotlivých ústrední, ktoré môže slúžiť aj ako prínosný materiál pre záujemcov o implementáciu ľubovoľnej z ústrední alebo členov vývojových komunít.

Použitá literatúra

[1] Guide to Intrusion Detection and Prevention Systems (IDPS): Recommendations of the National Institute of Standards and Technology [online]. Gaithersburg, 2007, 127 s. [cit. 13.3.2015]. Dostupné z: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf

[2] Unregister Attacks in SIP. In: Secure Network Protocols, 2006. 2nd IEEE Workshop [online]. 2006 [cit. 2015-03-14]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4110434

[3] Fail2Ban Manual. Fail2ban.org [online]. 2013, 24.5.2013 [cit. 2015-03-10]. Dostupné z: http://www.fail2ban.org/wiki/index.php/MANUAL_0_8

[4] Netfilter extensions. Netfilter.org [online]. 2001 [cit. 2015-03-07]. Dostupné z: http://www.netfilter.org/documentation/HOWTO/netfilter-extensions-HOWTO-3.html#ss3.21

[5] Iptables History. Netfilter.org [online]. 2014 [cit. 2015-03-03]. Dostupné z: http://www.netfilter.org/about.html

[6] VOZŇÁK, Miroslav. TELEFONNÍ ÚSTŘEDNY ASTERISK. In: *Teorie a praxe IP*

telefonie [online]. 2008 [cit. 2014-10-20]. Dostupné z: http://www.iptelefon.cz/archiv/dok_osta/ipt-2008_Telefonni_ustredny_Asterisk.pdf

[7] RUSSELL, Bryant. Asterisk Versions. Asterisk [online]. 2013 [cit. 2014-10-20]. Dostupné z: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[8] VAN MEGGELEN, J., MADSEN, L., SMITH, J. *AsteriskTM: The Future of Telephony*, Fourth Edition. USA: O'Reilly Media, Inc., 2013. ISBN-13: 978-0-596-51048-0

[9] VoIP Signaling Protocols: Introducing VoIP Signaling Protocols. *Trainsignal.com* [online]. 2.6.2010 [cit. 2014-11-01]. Dostupné z: http://www.trainsignal.com/blog/voip-signaling-protocols

[10] Yate.ro. *Yate architecture* [online]. 2013 [cit. 2015-03-01]. Dostupné z: http://docs.yate.ro/wiki/Yate_architecture

[11] Network Security Protocols. K2E Security [online]. 2013 [cit. 2015-03-01]. Dostupné z: http://www.k2esec.com/secure-communications/network-security-protocols-ipsec-vs-tlsssl-vs-ssh-part-ii

[12] RFC 3851. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1: Message Specification. 2004. [cit. 2015-1-14] Dostupné z: http://www.ietf.org/rfc/rfc3851.txt

[13] HANDLEY, M. ai. SIP: Session Initiation Protocol. In: *RFC 2543* [online]. Internet Engineering Task Force, 1999. Dostupné z: http://www.ietf.org/rfc/rfc2543.txt

[14] CAMARILLO, Gonzalo. *SIP demystified*. New York: McGraw-Hill, 2002. ISBN 00-713-7340-3.

[15] Kolektív autorov. SIP: Session Initiation Protocol. In: *RFC 3261* [online]. Internet Engineering Task Force, 2002. Dostupné z: http://www.rfceditor.org/rfc/rfc3261.txt

[16] JOHNSTON, Alan B. *SIP: understanding the Session Initiation Protocol*. Boston: Artech House, 2001, xxi, 201 s. ISBN 15-805-3168-7.

[17] JENKINS, Tom. A Guide To Session Initiation Protocol (SIP) [online]. 2010 [cit. 2014-11-03]. Dostupné z: http://www.scribd.com/doc/54600750/SIP-Tutorial-A-Guide-to-Session-Initiation-Protocol-SIP

[18] Kolektív autorov. HTTP Authentication. In: *RFC 2617*[online]. Internet Engineering Task Force, 1999. Dostupné z: https://www.ietf.org/rfc/rfc2617.txt

[19] Schulzrinne, H.; Casner, S.; Frederick, R.; ai. RTP: A Transport Protocol for Real-Time Applications. In: *RFC 3550* [online]. Internet Engineering Task Force, 2003. Dostupné z: http://www.ietf.org/rfc/rfc3550.txt

[20] Real-Time Transport Protocol (RTP) Parameters. *Iana.org* [online]. 2012 [cit. 2014-11-05]. Dostupné z: http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xml

[21] PERKINS, Colin. *RTP: audio and video for the Internet*. Boston: Addison-Wesley, 2003, 414 s. ISBN 06-723-2249-8.

[22] ČÍKA, Petr. *Multimediální služby.* Brno: Vysoké učení technické v Brně, 2012, 1.vyd., ISBN 978-80-214-4443-0.

[23] Dwivedi, Himanshu*. Hacking VoIP :protocols, attacks, and countermeasures* / San Francisco : No Starch Press, 2009. ISBN 978-1-59327-163-3.

[24] Kamailio Features. *Kamailio* [online]. 2014 [cit. 2014-10-15]. Dostupné z: http://www.kamailio.org/w/features/

[25] Kamailio SIP Router Releases. *Kamailio* [online]. 2014 [cit. 2014-10-15]. Dostupné z: http://www.kamailio.org/w/kamailio-sip-router-releases/

[26] SPENCER, et al. IAX (Inter-Asterisk Exchange) (RFC 5456) [online]. [cit. 2014-10-04]. Dostupné z: http://tools.ietf.org/html/rfc5456.txt

[27] IAX Parameters. *Iana.org* [online]. 2014 [cit. 2014-10-16]. Dostupné z: http://www.iana.org/assignments/iax-parameters/iax-parameters.xhtml#iax-parameters-3

[28] *Www.voip-info.org* [online]. 2010 [cit. 2014-10-16]. FreeSwitch. Dostupné z WWW: http://www.voip-info.org/wiki/view/FreeSwitch

[29] *FreeSWITCH wiki* [online]. 2011 [cit. 2014-10-21]. Core Outline of FreeSWITCH. Dostupné z WWW: http://wiki.freeswitch.org/wiki/Core_Outline_of_FreeSWITCH

[30] Get OpenSIPS. OpenSIPS [online]. 2014. [cit. 11-9-2014]. Dostupné z http://www.opensips.org/Downloads/Downloads

[31] OpenSIPS Webinar – Introduction to OpenSIPS. OpenSIPS [online]. 2014. [cit. 28-10-2014]. Dostupné z: http://www.opensips.org/html/docs/video/webinar001/

[32] Šilhavý, P. *Telekomunikační a informační systémy*. Brno: Vysoké učení technické v Brně, 2014. s. 1-140. ISBN: 978-80-214-5027-1.

[33] RFC 3711. The Secure Real-time Transport Protocol (SRTP) [online]. 2004. [cit. 2014-10-14]. Dostupné z WWW: http://www.ietf.org/rfc/rfc3711.txt

[34] IAX2 Security. *DIGIUM, Inc.* [online] 2009 [cit. 2014-10-13]. Dostupné z: http://downloads.asterisk.org/pub/security/IAX2-security.pdf

[35] RFC 4347. Datagram Transport Layer Security [online]. 2006. [cit. 2014-10-16]. Dostupné z WWW: http://www.rfc-editor.org/rfc/rfc4347.txt

[36] SISALEM, Dorgham. *SIP security*. Chichester, U.K.: Wiley, 2009, xiv, 336 s. ISBN 04-705-1636-4.

[37] Jackson, Benjamin. *Asterisk hacking*. Burlington: Syngress, 2007. ISBN 978-1-59749-151-8.

[38] BEDNÁR, Jakub. *Výkonnostní limity, spolehlivost a bezpečnost Open source PBX: diplomová práca*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2012. 145 s.

[39] Endler, David. *Hacking exposed VoIP :voice over IP security secrets & solutions* . New York : McGraw-Hill, 2007. ISBN 978-0-07-226364-0.

[40] Csipsimple: Issue 1176: TLS and SRTP and Wireshark. Code.google.com [online]. 2011 [cit. 2014-11-02]. Dostupné z: https://code.google.com/p/csipsimple/issues/detail?id=1176

[41] Kolektív autorov. Session Initiation Protocol (SIP) Torture Test Messages. In: *RFC 4475* [online]. Internet Engineering Task Force, 2006. [cit. 2015-1-14]. Dostupné z: https://tools.ietf.org/html/rfc4475

[42] REHMAN, Rafeeq Ur. Intrusion detection systems with Snort: advanced IDS techniques using Snort, Apache, MySQL, PHP, and ACID. Upper Saddle River, N.J.: Prentice Hall PTR, 2003. ISBN 01-314-0733-3

Zoznam použitých skratiek

AES	-	Advanced Encryption Standard
BSC	-	Border Session Controller
CC	-	Country Code
DES	-	Data Encryption Standard
DoS	-	Denial of Service
GPL	-	General Public License
HMAC	-	keyed-Hash Message Authentication Code
HTML	-	HyperText Markup Language
IETF	-	Internet Engineering Task Force
IP	-	Internet Protocol
ISDN	-	Integrated Services Digital Network
ITU	-	International Telecommunication Union
IAX	-	Inter Astersik eXchange
IVR	-	Interactive Voice Response
LAN	-	Local Area Network
LTS	-	Long Term Support
MGCP	-	Media Gateway Control Protocol
NAT	-	Network Address Translation
NDC	-	National Destination Code
NTP	-	Network Time Protocol
PBX	-	Private Branch eXchange / Public Based eXchange
PSTN	-	Public Switched Telephone Network
QoE	-	Quality of Experience
QoS	-	Quality of Service
RAS	-	Registration Admission Status
RTCP	-	Real–Time Transport Control Protocol
RTP	-	Real–Time Transport Protocol
RTSP	-	Real–Time Streaming Protocol
SAP	-	Session Announcement Protocol
SCCP	-	Skinny Call Control Protocol
SDP	-	Session Description Protocol
SIP	-	Session Initiation Protocol
SHA	-	Secure Hash Algorithm
SRTP	-	Secure Real–Time Transport Protocol
SS7	-	Signaling System č.7
SSL	-	Secure Socket Layer
TCP	-	Transmission Control Protocol
TSL	-	Transport Layer Security
UDP	-	User Datagram Protocol
UMTS	-	Universal Mobile Telecommunications System
URI	-	Uniform (Unique) Resource Identifier
URL	-	Uniform (Unique) Resource Locator
VoIP	-	Voice over Internet Protocol
WAN	-	Wide Area Network
XMMP	-	eXtensible Messaging and Presence Protocol

Príloha A

Sada užitočných a často používaných príkazov pre každú z ústrední.

- Asterisk

  spustit: `asterisk -rvvvvv`

  zastavit: `core stop now`

  debug: `sip/iax2 set debug on/off`

- Freeswitch

  Spustit: `/usr/local/freeswitch/bin/freeswitch`

  do konzole sa prihlasi cez: `/usr/local/freeswitch/bin/fs_cli`

  opustit konzolu cez: `/exit`

  zastavit cez: `shutdown`

  debug: `sofia global siptrace on`

- Yate

  Spustit s logom na obrazovke: `yate -vvvvDof`

  do konzole: `telnet 127.0.0.1 5038`

  opustit: `quit`

- Kamailio

  Spustit s logovanim na obrazovke: `kamailio -E -d`

  konzola: `kamcmd`

  pridanie účtu: `kamctl add 1000@domain heslo`

  ukázanie registrácii: `kamctl db show subscriber`

- Opensips

  Spustit: `/etc/init.d/opensips start`

  konzola start: `/usr/local/opensips/sbin/opensipsctl start`

  otvorenie: `/usr/local/opensips/sbin/osipsconsole`

  pridanie a ukázanie rovnako ako v kamilio ale: `opensipsctl`