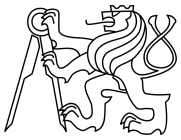
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Aplikace pro analýzu síťových toků technologie VoIP/SIP
Student:	Tomáš Jánský
Vedoucí:	Ing. Tomáš Čejka
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Seznamte se s problematikou monitorování síťových toků v počítačových sítích.
Prostudujte existující systém Nemea pro analýzu síťových toků.
Proveďte analýzu síťového provozu technologie Voice over IP (VoIP), konkrétně Session Initiation Protocol (SIP).
Navrhněte množinu potřebných informací pro následnou analýzu škodlivých síťových toků protokolu SIP, cílem je rozpoznat v síťových tocích například útoky hrubou silou na VoIP infrastrukturu.
Navrhněte softwarový modul pro analýzu sbíraných informací tak, aby byl použitelný v rámci systému Nemea.
Implementujte navržený modul.
Výsledné řešení otestujte pomocí dat, které dodá vedoucí.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 14. listopadu 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Aplikace pro analýzu síťových toků technologie VoIP/SIP

Tomáš Jánský

Vedoucí práce: Ing. Tomáš Čejka

13. května 2016

Poděkování

Rád bych poděkoval především vedoucímu práce Ing. Tomáši Čejkovi za odborné vedení, konzultace, rady a připomínky během tvorby práce. Dále děkuji sdružení CESNET z.s.p.o. za poskytnutí technických prostředků a možnosti použití programu na reálné síti, což se ukázalo jako velký přínos. Poděkování patří také mým kolegům z projektu Liberouter, kteří odvedli skvělou práci při vývoji systému NEMEA. V neposlední řadě bych rád poděkoval své rodině a přátelům za jejich psychickou podporu během mého studia a především během psaní bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2016 .....

Odkaz na tuto práci

Jánský, Tomáš. *Aplikace pro analýzu síťových toků technologie VoIP/SIP*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Stále častější využití technologie Voice over Internet Protocol (VoIP) v internetové telefonii s sebou přináší řadu bezpečnostních rizik. Častým cílem útočníků je signalizační protokol Session Initiation Protocol (SIP). Tato práce obsahuje analýzu protokolu SIP, podrobněji jsou zde popsány útoky hrubou silou, které se snaží prolomit hesla uživatelů VoIP technologie. Výstupem práce je aplikace, která detekuje tyto útoky. Součástí je i popis útoků detekovaných na reálné síti. Aplikace je implementována jako součást systému Network Measurements Analysis (NEMEA), který je vyvíjen sdružením *CESNET z.s.p.o.*

Klíčová slova Session Initiation Protocol, SIP, VoIP, útoky hrubou silou, počítačové sítě, síťová bezpečnost, detekce útoku, síťové toky, IP telefonie, NEMEA, TRAP

Abstract

The increasing use of Voice over Internet Potocol (VoIP) in internet telephony comes with a number of security risks. A frequent target of attackers is the Session Initiation Protocol (SIP). This work includes analysis of SIP protocol and more closely describes brute-force attacks which try to breach passwords of VoIP users. The outcome of this thesis is an application that detects these attacks. Description of attacks detected on a live network is also included. The application is implemented as a part of the Network Measurements Analysis (NEMEA) system, which is being developed in *CESNET z.s.p.o.*

Keywords Session Initiation Protocol, SIP, VoIP, brute-force attacks, computer networks, network security, attack detection, network flows, IP telephony, NEMEA, TRAP

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Technologie umožňující přenos digitalizovaného hlasu prostřednictvím počítačových sítí se stávají stále populárnějšími, a to nejen ve velkých organizacích, ale také v domácnostech. S příchodem chytrých telefonů a dostatečně rychlé mobilní sítě vznikají aplikace, které běžné volání a psaní zpráv nahrazují technologií Voice over Internet Protocol (VoIP). Toto řešení bývá často ekonomicky výhodnější, například při hovorech do zahraničí.

Tento relativně nový svět však také přitahuje pozornost stále více podvodníků, kteří se snaží využít bezpečnostních chyb nebo nedostatečného zabezpečení zařízení, která nabízejí VoIP služby. Podle zprávy [1] od expertů na telekomunikační podvody ze společnosti *Communication Fraud Control Association* dosáhly finanční ztráty v oblasti telekomunikací v roce 2015 částky přibližně 38,1 miliard amerických dolarů. Přestože se jedná o 18% pokles oproti částce z roku 2013, zůstávají tyto podvody i nadále velice lukrativní.

Častým cílem útoků je signalizační protokol Session Initiation Protocol (SIP) [2], který slouží k navazování a ukončování spojení mezi dvěma VoIP zařízeními. Jedná se o jednoduché útoky jako jsou například útoky hrubou silou nebo odepřením služby, až po složitější útoky typu podvržení identity, nebo úprava komunikace. Tato práce se soustředí na detekci hádání hesel uživatelů technologie VoIP. Tyto útoky spadají do kategorie útoků hrubou silou a znalost přihlašovacích údajů uživatele přináší mnoho možností, jak spáchat telekomunikační podvod. Aby se dalo předejít finančním ztrátám, je třeba tyto útoky na síti automatizovaně detekovat v reálném čase. Pokud se již někomu podaří spáchat podvod založený na předchozím hádání hesla, software vytvořený v rámci této práce může pomoci k odhalení pachatele.

V první části představím a stručně popíšu technologii VoIP, uvedu zde především její technické aspekty. Dále provedu detailnější rozbor protokolu SIP, uvedu jeho zásadní vlastnosti a funkce, názorně popíšu triviální navázání komunikace mezi dvěma koncovými zařízeními. Ukážu, jak funguje proces autentizace uživatele pomocí protokolu SIP a představím několik nejčastějších typů útoků vedených proti tomuto protokolu. Zaměřím se zejména na útoky

hrubou silou s cílem prolomit hesla uživatelů. Sekci zakončím popisem detekčního prostředí Network Measurements Analysis (NEMEA).

Cílem druhé části práce je vývoj softwaru, který umožní na reálné síti v reálném čase detekovat pokusy o prolomení hesel uživatelů registrovaných na SIP serverech. Nejprve provedu analýzu požadavků na výsledný software. S využitím této analýzy a znalostí autentizačního postupu protokolu SIP navrhnu vhodný detekční algoritmus. Výsledný software implementuji jako součást systému NEMEA. Nejdůležitějším cílem je otestovat tento program na reálné počítačové síti a úspěšně detekovat útoky hrubou silou na hesla uživatelů VoIP technologie.

Kapitola 1

Internetová telefonie

1.1 Technologie Voice over Internet Protocol

Voice over Internet Protocol (VoIP) je současná technologie umožňující přenos digitalizovaného hlasu s pomocí počítačové sítě mezi koncovými zařízeními v reálném čase. Z názvu již vyplývá, že ke své funkci využívá Internet Protocol (IP), dá se tedy použít k přenosu hlasu jak přes internet, tak přes jakoukoliv privátní datovou síť, která tento protokol podporuje.

Oproti běžné telefonní síti (Public Switched Telephone Network – PSTN) má VoIP několik zásadních výhod vyjmenovaných v článku [3]. Může přenášet data, zvuk a obraz zároveň. Dále je účtování hovoru obecně levnější než u telefonní sítě, a to mj. díky absenci poplatků za volání do zahraničí. Navíc k vylepšení poskytované služby stačí jen zmodernizovat software. Z hlediska hardwaru stačí pouze připojení k síti a počítač, případně specializovaný IP telefon, dnes ve velkých organizacích již zcela běžné zařízení.

Pobočková telefonní ústředna (Private Branch Exchange – PBX) využívající VoIP může také podporovat propojení světa internetové telefonie s tradiční telefonní sítí. Tyto ústředny umožňují připojit soukromou VoIP infrastrukturu k veřejné telefonní síti. Díky této své funkčnosti jsou však cílem většiny útočníků. Průběhy jednotlivých útoků a jejich dopady jsou popsány v dalších kapitolách.

1.1.1 Stručná historie VoIP

Joe Hallock [4] uvádí, že za vznik technologie VoIP se obecně považuje únor roku 1995, kdy malá izraelská společnost *Vocaltec, Inc.* představila veřejnosti svůj produkt *InternetPhone*. Tento software, nainstalovaný na běžný počítač, dovolil dvěma uživatelům navázat spojení a komunikovat pomocí mikrofonu a sady reproduktorů. V roce 1998 se již začala objevovat řešení využívající VoIP propojující dva telefony, nebo počítač s telefonem. Koncem roku 1998 tvořil

počet uskutečněných VoIP hovorů 1 % všech hovorů, v roce 2000 to byla 3 % a v roce 2003 vystoupal tento poměr až na 25 %.

1.1.2 Technické aspekty VoIP

Jednou z klíčových vlastností, které chceme při telefonování dosáhnout, je kvalita hovoru. Od toho se odvíjí mnoho aspektů internetové telefonie. Následující pasáž je shrnutím části článku [5].

Velikost zpoždění (delay), vysoká míra jeho kolísání (jitter) nebo ztrátovost paketů (packet loss) mohou mít za následek trhaný zvuk a celkovou nesrozumitelnost hovoru. Je tedy nutné zvuková data efektivně rozdělit do jednotlivých paketů, přenést přes celou síť, a v koncovém bodě zvuk opět složit dohromady.

Většina aplikací na internetu využívá komunikační potvrzovací protokol TCP (Transmission Control Protocol), který je založen na principu potvrzování přijatých paketů. Pro potřeby technologie VoIP je ovšem vhodnější protokol UDP (User Datagram Protocol), neboť oproti TCP nezaručuje doručení každého paketu. Tím zásadně snižuje velikost zpoždění jednotlivých paketů obzvlášť v případě, kdy je transportní síť zahlcená nebo se obecně vyznačuje vysokou ztrátovostí. Občasná ztráta paketu má na kvalitu hovoru výrazně menší vliv než velké zpoždění. K přenosu zvukových dat se nejčastěji využívá protokol RTP (Real-time Transport Protocol), který umožňuje zasílání zpráv jednomu zařízení (unicast), nebo skupině zařízení (multicast). Skupina protokolů RTP/UDP/IP je v dnešní době již osvědčeným způsobem jak přenášet zvuk na síti v reálném čase.

Před přenosem zvukových dat je potřeba nejprve navázat spojení mezi jednotlivými VoIP zařízeními. K tomuto účelu už dnes existuje několik signalizačních protokolů. Z hlediska modelu komunikace je můžeme rozdělit na master-slave, kdy jedno zařízení má kontrolu nad ostatními (Media Gateway Control Protocol, Megaco), a na peer-to-peer, kdy všechna zařízení mají stejné možnosti (H.323, Session Initiation Protocol – SIP). Nejčastěji používaným řešením signalizace v dnešní době je protokol SIP. Jeho detailním popisem se tato práce zabývá v následující kapitole.

1.2 SIP – Session Initiation Protocol

Tato sekce čerpá především z RFC 3261 [2], dále pak ze zdrojů [5] a [6].

Session, do češtiny překládané jako „relace“, je označení pro permanentní síťové spojení mezi dvěma zařízeními, během kterého dochází k výměně paketů. Protokol SIP slouží k navázání, změně nebo ukončení takovéto relace mezi dvěma koncovými zařízeními. Jedná se o signalizační protokol aplikační vrstvy podobný protokolu Hypertext Transfer Protocol (HTTP). Za jeho zrodem stojí skupina Internet Engineering Task Force (IETF) a je podrobně popsán v RFC 3261 [2]. Doporučeným a výchozím portem pro SIP je port 5060,

resp. 5061, jedná-li se o spojení šifrované pomocí protokolu Transport Layer Security (TLS). Tento port se používá jak pro TCP, tak pro UDP spojení. SIP však může komunikovat v zásadě na jakémkoliv portu.

1.2.1 Funkce

Přestože se SIP využívá zejména v souvislosti s VoIP, dokáže navázat prakticky jakoukoliv relaci nezávisle na použitém transportním protokolu či typu relace, kterou navazuje. Jeho hlavní funkce jsou [6]:

- zjištění koncového zařízení
- zjištění, zda je koncové zařízení dostupné a ochotné navázat relaci
- výměna informací o typu relace a dalších parametrech potřebných k navázání relace
- správa relace – zahájení relace, modifikace jejích parametrů a ukončení

1.2.2 Vysvětlení základní terminologie

SIP URI: K jednoznačné identifikaci koncového zařízení v síti využívá SIP jednotný identifikátor zdroje (Uniform Resource Identifier – URI). Nezáleží tedy, zdali se změní IP adresa koncového zařízení nebo fyzická podoba zařízení, pomocí všech částí SIP URI jsme schopni uživatele jednoznačně identifikovat. Syntaxe vypadá následovně:

```
sip:[uživatel[:heslo]@]host[:port][;uri-parametry][?hlavičky]
```

Popis jednotlivých částí SIP URI:

- „**sip:**“ resp. „**sips:**“ je klíčové slovo na začátku každé SIP URI; druhá varianta se používá v případě, že je spojení zabezpečeno pomocí TLS
- **uživatel** je identifikátor klienta; v souvislosti s VoIP se často využívá telefonní číslo, avšak nemusí se jednat výhradně o řetězec numerických znaků
- **heslo** je asociované s daným uživatelem; již samotní autoři RFC silně nedoporučují zasílat heslo v SIP URI, neboť je zasílané ve formě prostého textu, a tedy viditelné pro každého, kdo by komunikaci zachytil
- **host** je, kromě počátečního „sip:“ resp. „sips:“, jediná nutná součást SIP URI a označuje zařízení, se kterým vedeme relaci; jedná se o plně specifikované doménové jméno (Fully Qualified Domain Name – FQDN), nebo numerický zápis IPv4 či IPv6 adresy

- **port** – číslo portu, na kterém host očekává SIP zprávy
- **URI parametry** mohou ovlivnit zpracování požadavku; jednotlivé parametry jsou odděleny středníkem a každý parametr je zapsán ve formátu `název=hodnota`
- **hlavičky** jsou políčka, která mají být přidána do vytvořeného požadavku, jednotlivé hlavičky jsou oddělené pomocí ampersandu a každá z nich je zapsána ve formátu `název=hodnota`

SIP zpráva: Všechna SIP zařízení fungují na principu přijetí požadavku, jeho zpracování a následném odeslání odpovědi na daný požadavek. SIP požadavky a odpovědi se souhrnně označují jako SIP zprávy.

SIP transakce: Veškerá komunikace mezi dvěma SIP zařízeními započatá prvním požadavkem klienta a ukončená finální odpovědí serveru.

Uživatelský agent: Po dobu jedné SIP transakce získají klienti logický status v závislosti na tom, koho v dané transakci reprezentují. Uživatelský agent (User Agent – UA) může reprezentovat 2 role:

- *User Agent Client* (UAC) – logická část reprezentující klienta; zasílá požadavky na UAS
- *User Agent Server* (UAS) – logická část přijímající požadavky od UAC; zasílá odpovědi na UAC

Registrační server: Přijímá, ověřuje a vyřizuje REGISTER požadavky klientů.

Lokalizační služba: Ukládá do databáze informace o uživatelích a napomáhá lokalizaci volaného účastníka.

Směrovací server: Na SIP požadavek odpovídá formou požadavku na přesměrování s adresou, na které může původce požadavku hledanou entitu kontaktovat přímo.

SIP Brána (Gateway): Jedná se o zařízení s více rozhraními, které je schopné propojovat různé technologie. Díky němu lze na jednom konci přijímat komunikaci v protokolu SIP a převést ji např. na protokol H.323. Používá se hlavně k propojení internetové telefonie s PSTN.

Proxy server: Entita fungující jako zprostředkovatel mezi dvěma jinými entitami. Chová se zároveň jako server i klient. Hlavním účelem proxy serveru je směrování (routing), neboli zasílání zpráv jiné entitě, která je blíž koncovému

zařízení. SIP proxy server se liší od uživatelského agenta nebo brány ve třech směrech [6]:

1. Proxy server nevytváří požadavky. Pouze odpovídá na požadavky od uživatelského agenta.
2. Proxy server neumožňuje práci s mediálními daty.
3. Proxy server nezasahuje do těl zpráv. Spoléhá se čistě jen na informace v hlavičkách.

Proxy server, registrační server a lokalizační služba jsou logické entity. Pro účely implementace mohou být zkombinovány v jedné aplikaci.

1.2.3 Požadavky a odpovědi

Cílem požadavků zaslaných na konkrétní zařízení je spustit na něm metodu. Každý požadavek musí začínat tzv. „*Request-Line*“. Jde o řádek, na kterém je definován název požadované metody, URI cílového zařízení a verze protokolu SIP. RFC 3261 [2] specifikuje 6 základních metod. Novější RFC doplňují ještě další, jako např. SUBSCRIBE nebo NOTIFY, ty však pro účely této práce není třeba popisovat.

- OPTIONS – žádost o seznam podporovaných funkcionalit UAS
- REGISTER – žádost o registraci uživatele na registračním serveru
- INVITE – požadavek na navázání relace
- ACK – potvrzení navázání relace
- CANCEL – přerušení relace při jejím navazování
- BYE – ukončení probíhající relace

Na všechny požadavky musí UA protistrany patřičně reagovat. Pro tyto účely je vypracován seznam možných návratových kódů značících výsledek provedené metody. První řádek odpovědi, tzv. „*Status-Line*“, zasílá verzi protokolu SIP, třímístný číselný kód a frázi popisující důvod zaslaného kódu. Seznam možných kódů reprezentujících odpovědi UAS:

- **1xx** – *Prozatímní odpověď* – požadavek byl přijat, probíhá jeho zpracování
  - `100 Trying` – požadavek se zpracovává
  - `180 Ringing` – čekání na potvrzení o navázaní relace od protistrany
- **2xx** – *Úspěch* – serveru se podařilo požadavek vykonat, nebo přijmout

  - `200 OK` – požadavek byl úspěšně proveden

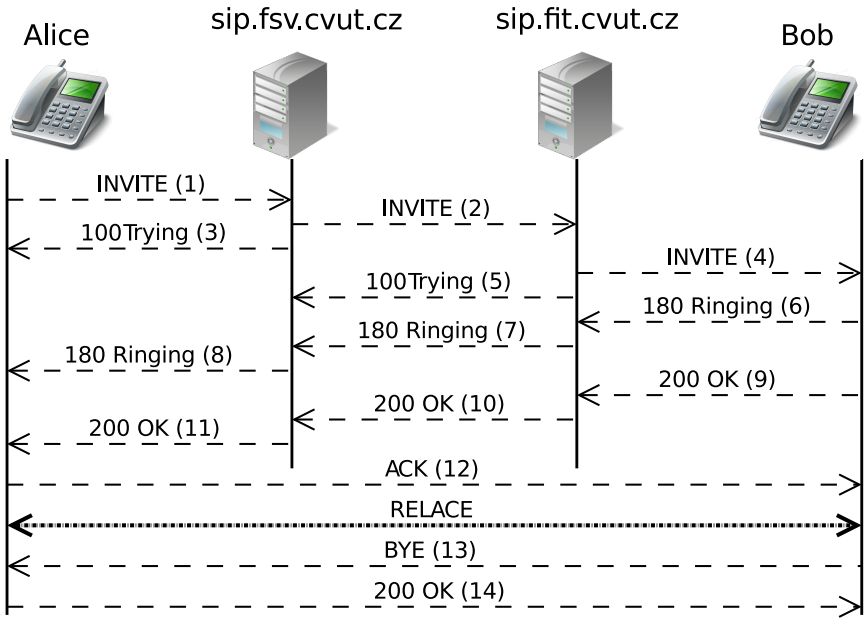
- **3xx** – *Přesměrování* – je vyžadována další akce k provedení požadavku
- **4xx** – *Chyba klienta* – požadavek obsahuje chybu v syntaxi, nebo ho nelze provést na tomto serveru
  - `400 Bad Request` – např. chybná syntaxe
  - `401 Unauthorized` – požadavek vyžaduje autentizaci
  - `403 Forbidden` – nepovolená akce z důvodu nedostatečného oprávnění; selhání autentizace; stejný požadavek by neměl být zasílán znovu
  - `404 Not Found` – požadované URI nenalezeno na registračním serveru
  - `407 Proxy Authentication Required` – SIP proxy server vyžaduje autentizaci k provedení daného požadavku
- **5xx** – *Chyba serveru* – serveru se nepodařilo provést validní požadavek
- **6xx** – *Obecná chyba* – požadavek nelze provést na žádném serveru

1.2.4 Ukázka navázání a ukončení relace

Na obrázku 1.1 je znázorněn hovor mezi uživateli Alicí (alice@sip.fsv.cvut.cz) a Bobem (bob@sip.fit.cvut.cz). U každé zprávy na obrázku je pro ilustraci napsáno číslo značící pořadí zprávy v průběhu ukázky komunikace. Telefon Alice i Boba zná pouze adresu svého proxy serveru. Rozhodne-li se Alice komunikovat s Bobem, musí znát jeho URI, tedy sip:bob@sip.fit.cvut.cz.

Telefon Alice vyšle požadavek `INVITE sip:bob@sip.fit.cvut.cz` na proxy server, neboť jinou adresu nezná. Proxy server požadavek zpracuje, najde díky službě Domain Name System (DNS) IP adresu serveru sip.fit.cvut.cz, na ni přepošle přijatý požadavek a Alici odešle odpověď `100 Trying`. Proxy server, zajišťující SIP službu Bobovi (tedy sip.fit.cvut.cz), přepošle požadavek na IP adresu uživatele `sip:bob@sip.fit.cvut.cz`, kterou zjistí pomocí lokalizační služby, a zpět vyšle opět odpověď s kódem 100. Telefon Boba obdrží požadavek `INVITE` od uživatele `sip:alice@sip.fit.cvut.cz`, začne vyzvánět a odešle odpověď `180 Ringing`, která je přeposlána přes proxy servery až do telefonního zařízení Alice.

Bob hovor přijme, což vede k odeslání zprávy `200 OK` značící úspěšné navázání spojení. Tato zpráva, kromě informací a parametrů o navazované relaci, obsahuje také hlavičku *Contact*, ve které je napsána IP adresa Bobova telefonu. Díky tomu může Alice veškerou nadcházející komunikaci v této transakci



Obrázek 1.1: Průběh relace

směrovat přímo na konkrétní zařízení. Obejde tak proxy servery a sníží zpoždění zasílaných paketů. V další zprávě potvrzuje Alice dohodnuté parametry multimediální relace zprávou `ACK` a nastává přenos dat.

V našem případě hovor ukončuje Bob zprávou `BYE` a Alice mu tuto zprávu potvrdí číselným kódem 200. V tento okamžik je relace ukončena. Následující transakce by již měla být znovu vedena přes proxy servery.

SIP je schopen také modifikovat parametry hovoru pomocí opětovného zasílání `INVITE` zpráv, odpovědí `200 OK` a zprávou `ACK` od strany, která změnu parametrů inicializovala.

Cesta, přes kterou je vedena signalizace protokolu SIP, je naprosto nezávislá na cestě, přes kterou jsou vedeny multimediální pakety. A. B. Johnston [6] označuje tento jev jako „*oddělení kontrolního kanálu od nesoucího*“.[1]

1.2.5 Popis hlaviček SIP zpráv

SIP zprávy obsahují mnoho různých hlaviček. Většina z nich je nepovinná. Níže jsou uvedeny 2 zprávy, první je požadavek na vytvoření relace a druhá zpráva je odpovědí na něj. Obě SIP zprávy mají ještě tělo (zde nezobrazeno), které obsahuje Session Description Protocol (SDP). Tento protokol slouží k popsání atributů relace, která má být navázána.

[1] V originále: separation of control channel and bearer channel

```
INVITE sip:bob@sip.fit.cvut.cz SIP/2.0
Via: SIP/2.0/UDP pc33.sip.fsv.cvut.cz:5060;branch=z9hG4bK77
To: Bob <sip:bob@sip.fit.cvut.cz>
From: Alice <sip:alice@sip.fsv.cvut.cz>
Call-ID: a84b4c76e66710@pc33.sip.fsv.cvut.cz
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.sip.fsv.cvut.cz>
Max-Forwards: 20
User-Agent: Linphone/3.7.0
Content-Type: application/sdp
Content-Length: 142

SIP/2.0 200 OK
Via: SIP/2.0/UDP sip.fit.cvut.cz:5060;branch=z9hG4mjs6
To: Alice <sip:alice@sip.fsv.cvut.cz>
From: Bob <sip:bob@sip.fit.cvut.cz>
Call-ID: a84b4c76e66710@pc33.sip.fsv.cvut.cz
CSeq: 314159 INVITE
Contact: <sip:bob@195.113.180.35>
Allow: ACK, INVITE, CANCEL, BYE, NOTIFY, SUBSCRIBE, REGISTER
Content-Type: application/sdp
Content-Length: 232
```

Via

Využívá se především k směrování odpovědí na požadavky. Uživatelský agent, který požadavek zasílá, zkopíruje svou adresu do této hlavičky. Hodnota značky *branch* vznikne vytvořením hashe hlaviček *From, To, Call-ID* a URI. Při průchodu požadavku přes proxy server se zkopírují všechny *Via* hlavičky a na začátek tohoto seznamu se přidá nová, která obsahuje adresu tohoto proxy serveru. UA nebo proxy server generující odpověď na požadavek zkopíruje všechny *Via* hlavičky a pošle odpověď na první adresu v seznamu hlaviček. Proxy server, který odpověď přijme, zkontroluje, zda se adresa ve vrchní *Via* hlavičce shoduje s jeho, smaže ji a přepošle zprávu na další adresu v seznamu. Tato hlavička je povinná v každé zprávě.

To

V praxi se často shoduje se SIP URI, avšak nikdy nesmí být používána pro účely směrování. Slouží k identifikaci volaného. Může obsahovat tzv. „*display name*“, v tom případě se SIP URI obalí pomocí <>. V okamžiku přijetí zprávy koncovým zařízením může UA přidat parametr *tag* za tuto hlavičku. Ten pak napomáhá k identifikaci konkrétního hovoru. Tato hlavička je povinná v každé zprávě.

From

Slouží k identifikaci volajícího. Obsah je stejný jako u hlavičky *To*. Parametr *tag* vkládá UA odesílatele. Tato hlavička je povinná v každé zprávě.

Call-ID

Jedná se o řetězec, sloužící k identifikaci konkrétního hovoru. Musí být globálně unikátní, pokud se nenalézá v požadavku za účelem registrace. Je vždy vytvořen uživatelským agentem a nesmí být modifikován serverem. Dříve bylo doporučováno jej sestavovat z náhodných znaků zakončených názvem hosta, avšak v novějších implementacích uživatelských agentů je řetězec *Call-ID* vytvářen jako kryptograficky náhodný řetězec. Společně s parametry *tag* u hlaviček *From* a *To* tvoří unikátní kombinaci, která jasně definuje peer-to-peer spojení mezi Alicí a Bobem a označuje se jako tzv. „*dialog*“. Tato hlavička je povinná v každé zprávě.

CSeq

Tato hlavička se skládá z dekadického čísla a metody, kterou požadavek vyvolává. V případě odpovědi se jedná o metodu, která odpověď vyvolala. Každý nový požadavek toto číslo sekvenčně zvyšuje (většinou o 1), kromě požadavků `ACK` a `CANCEL`, které používají číslo uvedené u `INVITE` požadavku, které potvrzují resp. ruší. Napomáhá k sestavování časového sledu zpráv, odhalování nedoručených zpráv a přiřazování odpovědí k požadavkům. Tato hlavička je povinná v každé zprávě.

Contact

Obsahuje URI, referující přímo na zařízení, na kterém se dá v daný moment uživatel rovnou kontaktovat. Toho je možné využít v nadcházejících požadavcích, avšak odpovědi musejí být stále směrovány cestou, kterou byly doručeny. Hlavička *Contact* musí být přítomna v požadavcích `INVITE` a odpovědích `200 OK`, které relaci navazují. Po jejím navázání může následná komunikace obcházet proxy servery, avšak přítomnost hlavičky *Record-Route* v předešlém požadavku, či v základní konfiguraci proxy serverů, tuto možnost potlačuje.

Max-Forwards

Značí maximální počet proxy serverů, přes které může požadavek být směrován. Jednotlivé proxy servery snižují tuto hodnotu o 1. Pokud zařízení obdrží požadavek s nulovou hodnotou v hlavičce *Max-Forwards*, pak vrátí odpověď `483 Too Many Hops`. Tato hlavička je povinná v každém požadavku od RFC 3261.

User-Agent

Nese informace o uživatelském agentovi, kterého klient používá. Většinou se jedná o název a verzi.

Content-Type

Specifikuje typ internetového média (Internet Media Type) v těle SIP zprávy. Pokud tato hlavička chybí, předpokládá se výchozí hodnota `application/sdp`.

Content-Length

Indikuje počet oktetů v těle SIP zprávy. Hodnota 0 značí, že zpráva nemá tělo. Dříve byla tato hlavička vyžadována vždy, ale kvůli automaticky generovaným tělům zpráv, kdy velikost těla není dopředu známa, je její přítomnost nyní pouze důrazně doporučena. V případě absence této hlavičky se počítá s tím, že tělo zprávy končí koncem datagramu při použití protokolu UDP, resp. koncem spojení, pokud je použit protokol TCP.

Allow

Poskytuje informace o tom, jaké metody UA podporuje. Tato hlavička se musí objevit v odpovědi `405 Method Not Allowed` a také v odpovědi na požadavek `OPTIONS`.

1.2.6 Autentizace v protokolu SIP

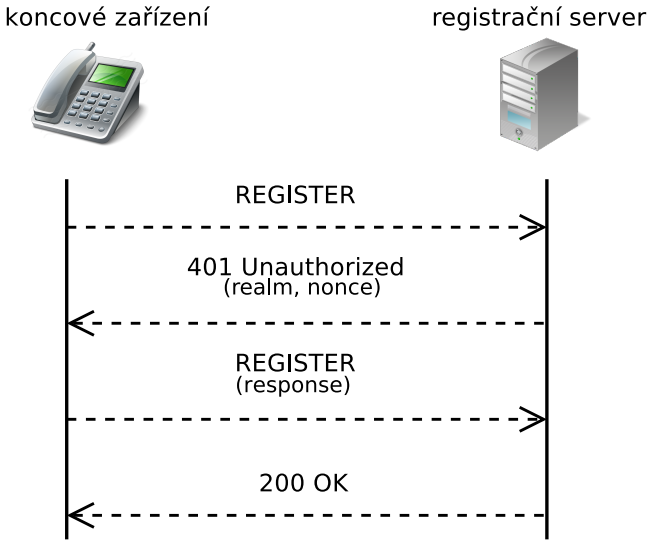
SIP server nebo proxy server může vyzvat původce požadavku k autentizaci. Autentizovat se dá prakticky kterýkoliv z požadavků, kromě `ACK` a `CANCEL`. Stejně tak není možné autentizovat odpovědi. Proces autentizace je názorně popsán v [7] a [6].

Obrázek 1.2 popisuje úspěšný průběh autentizace požadavku `REGISTER`. Uživatel se hodlá zaregistrovat na registračním serveru, aby mohl přijímat hovory na tomto zařízení. Jeho uživatelský agent tedy pošle požadavek na registraci, kde je v hlavičkách *To* a *From* zapsáno uživatelské jméno. Server vygeneruje číslo *nonce* a zašle klientovi odpověď `401 Unauthorized`, resp. `407 Proxy Authentication Required` v případě proxy serveru, s hlavičkou *WWW–Authenticate*, resp. *Proxy–Authenticate*:

```
WWW-Authenticate: Digest algorithm=MD5, realm="asterisk",
nonce="5b617850"
```

WWW–Authenticate nebo *Proxy–Authenticate* obsahují informace, které umožní UAC vytvořit hash, díky které klient prokáže znalost hesla, a následně ji odešle v dalším `REGISTER` požadavku serveru. Tato hash je obsažena v parametru *response*:

```
WWW-Authenticate: Digest username="500", realm="asterisk",
nonce="5b617850", uri="sip:localhost", algorithm=MD5,
response="44b795fe2754d708dba2d691eb9070a2"
```



Obrázek 1.2: Úspěšná registrace

Server nalezne v databázi hash a porovná ji s hodnotou obdrženou v parametru *response*. Tím klient prokázal svou identitu a server mu odpoví `200 OK`.

Parametr *algorithm* udává, jaký hashovací algoritmus se má použít. Parametry *nonce*, *realm*, *username*, *SIP URI* a samotné heslo jsou použity k vytvoření výsledného hash řetězce.

1.3 Podvody související s VoIP

Telekomunikační podvody jsou stále velice častým jevem. Podle expertů z *Communication Fraud Control Association*, kteří se telekomunikačními podvody podrobně zabývají, dosáhly ztráty způsobené podvodníky v roce 2015 přibližně 38,1 miliard amerických dolarů [1]. Přesto se však zdá, že počet útoků na telekomunikační infrastrukturu klesá, neboť tato suma je o 18 % nižší, než částka uvedená roku 2013. Důvodem podvodů je hlavně možný nemalý finanční zisk útočníků, avšak může se také jednat o útok s cílem finanční ztráty určité společnosti. Zde jsou nejčastější typy podvodů:

- *Mezinárodní podvod sdílením zisku* (International Revenue Share Fraud – IRSF): Jedná se o scénář, kdy si podvodník zakoupí telefonní čísla v určitém rozsahu od zahraničního poskytovatele těchto čísel a sdílí s ním určité procento finančního zisku z poplatků za volání na tato čísla. Podaří-li se podvodníkovi odcizit identitu nějakému uživateli, může pak

voláním na tato čísla z jeho účtu vytvořit nemalou škodu během několika hodin. Nejčastějšími státy, do kterých tyto hovory směřují jsou následující: Kubánská republika, Somálská federativní republika, Bosna a Hercegovina, Estonská republika a Lotyšská republika. Odhadovaná finanční ztráta činí 10,76 miliard amerických dolarů, což je přibližně 500% nárůst oproti roku 2013. [8]

- *Podvod obcházením spojovací sítě* (Interconnect Bypass Fraud): Během volání do zahraničí může být hovor veden přes více zahraničních operátorů přičemž si každý z nich účtuje určitý poplatek. Tyto poplatky se dají obejít pomocí internetu a tzv. *SIM box*. Jedná se o zařízení, ve kterém může být uloženo tisíce SIM karet. Pokud podvodník vlastní aktivní SIM karty, je schopný spojovat až tolik hovorů, kolik má karet. Mobilním operátorům tedy vzniká škoda způsobená menším množstvím hovorů vedených přes jejich síť, zatímco podvodník dostává zisk vzniklý rozdílem ceny volání do zahraničí a ceny za vnitrostátní hovor. Škoda pro rok 2015 činí 5,97 miliard amerických dolarů. [9]

- *Podvod využitím služeb se zvýšenou sazbou* (Premium Rate Services Fraud): Ve své podstatě se jedná o téměř stejný princip, jako u IRSF. Jen místo zahraničních čísel vlastní podvodníci čísla se zvýšenou sazbou, na která pak volají pod identitou někoho jiného. Jsou známy také případy, kdy podvodníci zašlou zprávu o zameškaném hovoru na telefony uživatelů. Ti, pokud se pokusí volat zpět, jsou přesměrováni na tato čísla. Zvýšená sazba může znamenat nejen zvýšenou cenu každé provolané minuty, ale také poplatek za započaté spojení, takže finanční ztráta vzniká okamžitě. Odhadovaná ztráta pro rok 2015 se pohybuje okolo 3,77 miliardami amerických dolarů. [8]

1.4 Bezpečnost protokolu SIP

Jak poukazuje A. B. Johnston [6], bezpečnost aplikačního protokolu SIP závisí mimo jiné na bezpečnosti protokolů používaných na nižších vrstvách. Podaří-li se narušit bezpečnost některých klíčových protokolů, na kterých je protokol SIP závislý, nebude možné vytvořit bezpečné prostředí pro signalizaci. Tím budou kompromitovány i případné protokoly vyšších vrstev. Další možnou hrozbou je prolomení zabezpečení operačních systémů na zařízeních poskytujících VoIP služby.

V případě, že se podaří zabezpečit tyto aspekty, existuje stále celá řada možných bezpečnostních hrozeb, jimž protokol SIP může čelit. Možné útoky, jejich důsledky a možnosti zabezpečení jsou podrobně popsány v [7] a [10]. Následuje popis těch nejčastějších a nejnebezpečnějších z nich.

Odepření služby (Denial of Service – DoS)

Útočník se snaží zahltit cílové zařízení zbytečnými požadavky za účelem vzniku částečného, nebo úplného odepření služby. Typicky se jedná o požadavky typu `INVITE`, `REGISTER` nebo `OPTIONS`. Je-li do útoku zapojeno více zařízení, pak hovoříme o tzv. DDoS (Distributed DoS) útoku.

Úprava komunikace (Man in the Middle)

Pokud má útočník možnost odposlouchávat komunikaci mezi serverem a UA, je schopný také libovolně modifikovat zprávy nebo je přeposílat. Případně může zachytit požadavek na autentizaci a následně jednorázově ukrást cizí identitu. Tento útok je obtížné uskutečnit, nicméně není nemožný, pokud má útočník kontrolu nad směrovačem, přes který prochází SIP zprávy.

Podvržení identity

Vzhledem k faktu, že SIP zprávy jsou v prostém textu čitelném pro člověka, je jednoduché podvrhnout některé SIP hlavičky a vydávat se tedy za někoho jiného.

Útok odregistrováním

Tento útok spočívá v ukončení platnosti registrace uživatele na registračním serveru. Útočník vytvoří požadavek `REGISTER` s podvrženou identitou a s hlavičkou *Expires*, kterou nastaví nulovou hodnotu. Toto simuluje chování koncového zařízení, které se vypíná a už nechce přijímat další hovory.

Útok ukončením hovoru

Pokud útočník zachytí komunikaci, která navazuje relaci mezi dvěma zařízeními, je schopný vytvořit požadavek `BYE` se správnými parametry *tag* a hodnotou hlavičky *Call-ID*. Zasláním tohoto fabrikovaného požadavku jednomu z klientů dokáže relaci přerušit.

Neoprávněné volání do sítě PSTN

Některé VoIP ústředny propojují IP telefonii s běžnou telefonní sítí. Je tedy možné volat z UA na klasický telefon a naopak. Pokud je zdrojová IP adresa považována za důvěryhodnou (například když se nachází v podsíti), nemusí ústředna vyžadovat autentizaci zasílaných požadavků z této adresy. Útočník zasíláním `INVITE` zpráv s různou předvolbou u telefonního čísla zjišťuje nastavení ústředny a hledá předvolbu, která přesměruje

hovor do PSTN za cílem úspěšného navázání spojení, a to nejlépe na prémiová nebo zahraniční čísla.

Útoky hrubou silou

Útočník se může pokusit odhalit slabá hesla uživatelů pomocí slovníkových a enumeračních útoků. Následně získá identitu tohoto uživatele a s tím i otevřené dveře k vykonání *International Revenue Share Fraud* nebo *Premium Rate Services Fraud*. Detekcí tohoto typu útoků se zabývá druhá část této práce.

1.5 Analýza útoků hrubou silou

Znalost uživatelského jména a hesla je pro podvodníky klíčová, neboť mnoho serverů vyžaduje autentizaci požadavků, které útočník vytvoří. K odhalení registrovaného uživatelského jména na určitém SIP serveru stačí odchytit část komunikace, kde jsou v hlavičkách *From* a *To* uživatelská jména v prostém textu zobrazena. V případě, že útočník nemá možnost komunikaci monitorovat, může se pokusit uhádnout uživatelské jméno zasíláním zpráv na server a sledováním jeho odpovědí. Tento způsob popsal v [11] Sandro Gauci, autor sady penetračních testů *SipVicious*, která je paradoxně jedním z nejčastěji používaných softwarových nástrojů, které jsou zneužívány k útokům.

Jakmile útočník zná uživatelské jméno a server, na němž je uživatel registrován, může se pokusit prolomit jeho heslo. Heslo se v síťovém toku nevyskytuje jako prostý text, nýbrž jako hash vytvořená pomocí algoritmu MD5, a to způsobem uvedeným v [7]. Finální hash zpráva vznikne ve třech krocích:

1. Vytvoří se MD5 hash z (`Username:Realm:Password`). *Username* je uživatelské jméno účtu, *password* je heslo k tomuto uživatelskému jménu a *realm* značí doménu serveru, kterou dostane UAC od serveru v hlavičce *WWW-Authenticate*.

2. Vytvoří se MD5 hash z (`Method:URI`). *Method* označuje název metody v požadavku, který se UAC snaží autentizovat. *URI* je popsáno v sekci 1.2.2.

3. Vytvoří se MD5 hash z (`Hash z bodu 1:Nonce:Hash z bodu 2`). *Nonce* je časová značka, kterou UAC obdrží od serveru jako součást hlavičky *WWW-Authenticate*. Po uplynutí určitého času, budou požadavky autentizované pomocí této značky vyhodnoceny serverem jako neplatné. Server by měl parametr *nonce* měnit při každé výzvě k autentizaci. Tím zabraňuje možnosti prolomení hesla pomocí kolize 2. řádu v případě, kdy se útočníkovi podaří odchytit platnou hash na síti.

Pokud útočník dokáže zachytit část komunikace mezi serverem a klientem, ve které se nachází úspěšný pokus o autentizaci požadavku, může zahájit offline slovníkový nebo enumerační útok. V případě, že se mu podaří nalézt opravdu heslo a nikoliv jen kolizi 2. řádu, může vytvořit požadavek a následně jej autentizovat. Tento útok je z hlediska monitorování komunikace na síti prakticky nemožné detekovat.

V případě, že útočník nemá možnost komunikaci na síti monitorovat, může se stále pokusit o online útok hrubou silou, také popsaný v [11]. Útočník zasílá `REGISTER` požadavky na server, pod identitou uživatele, jenž je na daném serveru registrován. Server vyzve útočníka k autentizaci každého takového požadavku odpovědí `401 Unauthorized`, ve které zašle v hlavičce *WWW-Authenticate* parametry *realm* a *nonce*. Útočník vypočte hash ze zadaných parametrů a předpokládaného hesla a odešle stejný požadavek zpět serveru. Ten dohledá v databázi hash z bodu 1 a vytvoří hash z bodu 3. Dojde k porovnání obdržené a vypočtené hodnoty hash. V případě, že útočník heslo uhádl, odpoví server zprávou `200 OK`. Pokud útočník heslo neuhádl, odpověď serveru není jasně standardizována, ale bývá buď opět `401 Unauthorized`, nebo `403 Forbidden`.

Kapitola 2

Detekční prostředí

Tato kapitola popisuje způsob, jakým je prováděno monitorování síťových toků v praxi sdružením *CESNET z.s.p.o* na síti CESNET2 [12]. Tato síť se vyznačuje vysokou rychlostí a propustností. Jsou na ni napojeny mnohé akademické a výzkumné sítě.

2.1 Síťové toky

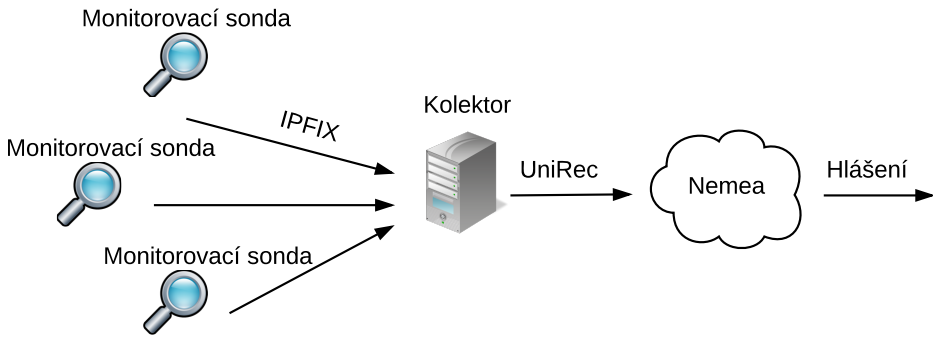
RFC 2722 [13] definuje síťový tok jako „*umělý logický ekvivalent hovoru nebo spojení*“[2]. Je charakteristický časem zahájení a ukončení. Obsahuje klíčové atributy pro daný tok (IP adresa zdroje a cíle, používané porty atd.) a také atributy, které se v průběhu toku agregují (počet přenesených paketů, počet přenesených bytů atd.).

2.2 Network Measurements Analysis – NEMEA

Sekce čerpá z technické zprávy o projektu NEMEA [14] z roku 2013.

NEMEA je projekt realizovaný výzkumnou skupinou Liberouter. V současnosti je v provozu na akademické síti CESNET2. Jedná se o open-source systém pro automatizovanou analýzu síťového provozu v reálném čase. Využívá tzv. „modulů“, což jsou spustitelné programy. Tyto moduly mohou například detekovat anomálie v síťových tocích, či počítat různé statistiky z těchto toků. Moduly mezi sebou mohou navzájem komunikovat pomocí komunikačního rozhraní Traffic Analysis Platform (TRAP). Moduly používají pro výměnu dat společný binární datový formát nazývaný Unified Record (UniRec). Kromě UniRec formátu je podporován také formát JavaScript Object Notation (JSON).

[2] V originále: an artificial logical equivalent to a call or connection



Obrázek 2.1: Detekční systém

2.2.1 Traffic Analysis Platform – knihovna TRAP

Všechny NEMEA moduly jsou propojeny komunikačním rozhraním TRAP. Každý z nich pouze volá funkce na přijímání a odesílání jednotlivých zpráv. Moduly nepotřebují znát typ rozhraní, ze kterého data přijímají nebo na které data posílají, o to se stará knihovna TRAP. Libtrap 0.7.5, což je aktuální verze knihovny v době vzniku této práce, podporuje komunikaci přes TCP nebo Unix soket. Dále také dokáže číst data ze souboru stejně jako je do souboru ukládat, což je velice užitečné k testování různých algoritmů. Typ komunikačního rozhraní se volí parametrem při spuštění modulu.

2.2.2 Unified Record – UniRec

Jedná se o binární datový formát zpráv, které si mezi sebou jednotlivé moduly vyměňují. Každá zpráva se skládá z jednotlivých položek, u kterých je definovaný název a jeho hodnota. Položky mohou mít statickou nebo dynamickou velikost v závislosti na tom, o jaký datový typ položky se jedná. Jednotlivé moduly si definují množinu položek, kterou očekávají či zasílají na svých rozhraních. V knihovně libtrap při zahájení komunikace funguje tzv. „negociace“ UniRec položek. Ta zajistí, že dva moduly spolu mohou vzájemně komunikovat v případě, že množina položek vyžadovaná přijímacím modulem, je podmnožinou množiny položek odesílacího modulu.

2.3 Způsob monitorování komunikace

K monitorování komunikace na síti je zapotřebí specializované zařízení, tzv. „monitorovací sonda“, které dokáže v reálném čase zpracovávat data ze sítě a odesílat údaje o síťových tocích například ve formátu Internet Protocol Flow Information eXport (IPFIX).

Obrázek 2.1 popisuje nasazení detekčního systému na síti CESNET2. Monitorovací sondy posílají data ve formátu IPFIX na kolektor, který síťové toky zpracovává a na své výstupní rozhraní posílá data ve formátu UniRec. Systém NEMEA svými jednotlivými moduly tyto toky proudově zpracovává a detekované anomálie ukládá či odesílá k dalšímu zpracování nebo vizualizaci.

Kapitola 3

Návrh detekčního modulu

Výsledný program je implementován jako součást systému NEMEA. Využívá již vytvořené API sloužící k přijímání a odesílání dat, což značně usnadňuje vývoj softwaru.

3.1 Požadavky na software

Funkční požadavky

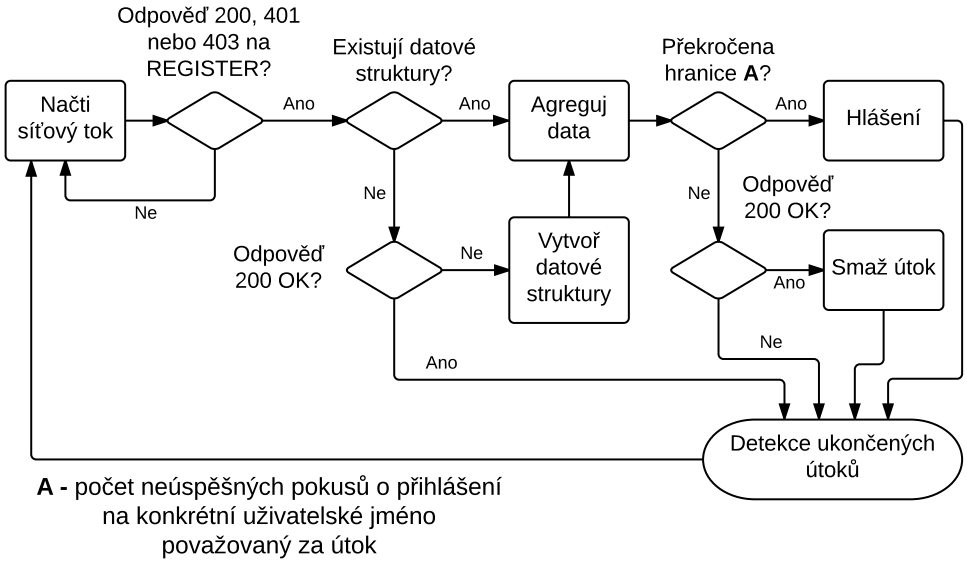
- zpracovávání zachycené SIP komunikace (průběžně za běhu na síti, nebo offline ze souboru)
- vyhodnocování toho, zdali nedošlo k pokusu o útok hrubou silou
- vytváření zpráv o útocích a předávání těchto zpráv dál

Nefunkční požadavky

- integrace jako součást systému NEMEA
- schopnost operovat i na sítích s vysokým objemem dat
- implementace v jazyce C/C++ nebo Python

3.2 Návrh detekčního algoritmu

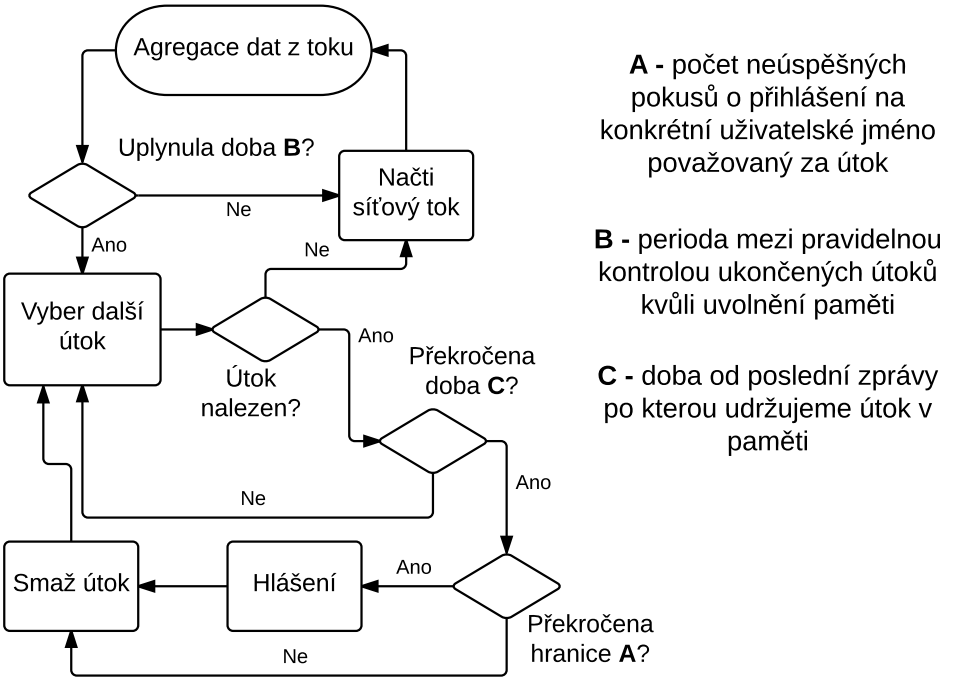
Detekce offline útoků hrubou silou na protokol SIP je z pohledu monitorování komunikace prakticky nemožná. Monitorováním komunikace však dokážeme jednoduše odhalit pokusy o online útoky hrubou silou za účelem prolomení hesla. Nutně se projeví nárůstem `REGISTER` požadavků směřujících na konkrétní server. Uživatelské jméno, jehož heslo se snaží útočník prolomit, se nachází v hlavičkách *From* a *To*. Na každý pokus o autentizaci musí server



Obrázek 3.1: Navržený algoritmus agregace a detekce útoků

odpovídat. Sledováním odpovědí serveru na příslušné `REGISTER` požadavky je možné určit, zda se pokus o autentizaci zdařil, či nikoliv. Přiřazováním odpovědí `200 OK`, `401 Unauthorized` a `403 Forbidden` k příslušným uživatelům na každém serveru a jejich agregováním lze kontrolovat počet úspěšných a neúspěšných pokusů o autentizaci. Sledováním IP adres klientů, kteří se snaží autentizovat jako daný uživatel, můžeme také odhalit distribuovaný pokus o prolomení hesla, neboli situaci, kdy se o autentizaci snaží více klientů zároveň. Na obrázku 3.1 je popsán diagram první části navrženého detekčního algoritmu. V té je ukázáno agregování odpovědí `401 Unauthorized` a `403 Forbidden`. Pokud jejich počet u konkrétního uživatele překročí zvolenou hranici, je vygenerováno jednorázové hlášení o probíhajícím útoku. V případě, kdy se objeví odpověď `200 OK`, mohou nastat 3 možnosti:

- Pokud se vztahuje k uživatelskému jménu, které nebylo doposud programem detekováno, nebo bylo již smazáno, program tuto zprávu zcela ignoruje.
- Pokud program detekoval předešlé nezdařilé pokusy o autentizaci tohoto uživatele, ale počet těchto pokusů je nižší než zvolená hranice pro generování hlášení, program vyhodnotí toto chování jako normální a smaže dosavadní datové struktury související s tímto uživatelským jménem.
- Pokud program detekoval předešlé nezdařilé pokusy o autentizaci tohoto uživatele a počet těchto pokusů překročil zvolenou hranici pro generování hlášení, program předpokládá, že došlo k prolomení hesla a je vygenerováno nové hlášení rozšířené o informace související s prolomením hesla



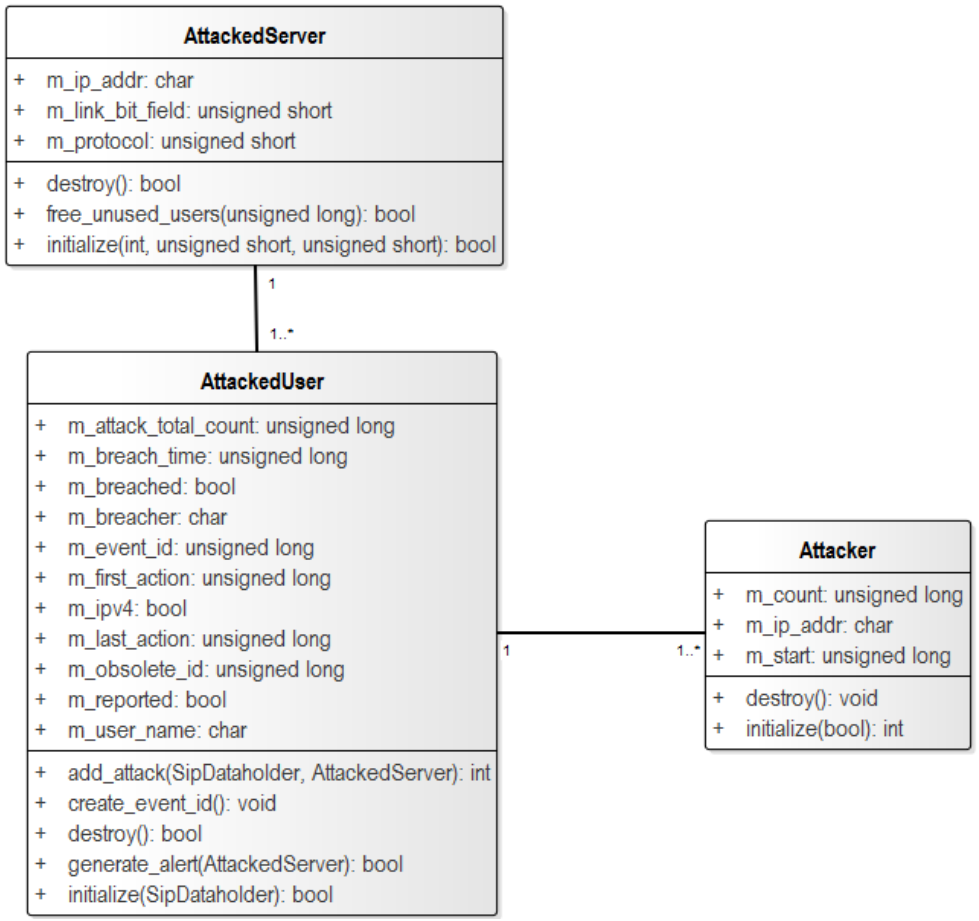
Obrázek 3.2: Navržený algoritmus detekce ukončených útoků

(čas, IP adresa úspěšného útočníka). Pokud program obdrží další zprávu `200 OK`, nové hlášení již není generováno.

Aby program efektivně pracoval s pamětí a podával závěrečné statistiky o proběhlých útocích, je nutné zajistit pravidelnou kontrolu agregovaných informací. K tomu slouží druhá část detekčního algoritmu, která je znázorněna na obrázku 3.2. Pokud se časová značka posledního obdrženého síťového toku liší od poslední provedené kontroly o více než je stanovený limit, dojde k iteraci přes všechna uživatelská jména uložená v programu. U každého uživatelského jména je vedena informace o čase posledního pokusu o autentizaci. Pokud se tento čas od aktuální časové značky liší o více, než je stanovená doba pro udržení záznamu v paměti, dojde k smazání tohoto uživatelského jména z programu. Pokud však počet pokusů o autentizaci u tohoto uživatelského jména překročil zvolenou hranici, je nejprve vygenerováno závěrečné hlášení se statistikami.

3.3 Návrh datových struktur

Data z přijímaných síťových toků je zapotřebí agregovat do logických struktur. U každého načteného síťového toku je nejprve zjišťováno, zda se jedná o SIP odpověď typu 200, 401, nebo 403. Pokud ano, vybrané důležité části síťového



Obrázek 3.3: Diagram datových struktur

toku jsou uloženy do pomocné datové struktury `SipDataholder`. Data z této struktury jsou následně využita k vytvoření či aktualizaci struktur reprezentujících SIP server, uživatele a klienta, a to v tomto pořadí. Na obrázku 3.3 je znázorněn diagram těchto struktur. Níže jsou sepsány jejich klíčové vlastnosti:

- `AttackedServer`:
  - reprezentuje SIP server
  - jako jednoznačný identifikátor slouží IPv4, nebo IPv6 adresa
  - obsahuje ukazatel na strukturu se všemi uživatelskými jmény, na která byl veden neúspěšný pokus o autentizaci
  - dokáže vyvolat kontrolu ukončených útoků přes všechna uživatelská jména asociovaná s tímto serverem

- `AttackedUser`:
  - reprezentuje uživatele na konkrétním SIP serveru
  - identifikace podle uživatelského jména
  - obsahuje mj. informace o celkovém počtu nesprávných autentizací, čas první a poslední zprávy, příznak, zda bylo heslo prolomeno a další
  - drží v sobě ukazatel na seznam klientů, kteří se pokoušejí autentizovat
  - dokáže mimo jiné vygenerovat hlášení o útoku na daného uživatele a vytvořit unikátní identifikační číslo pro toto hlášení založené na čase první útočné zprávy
- `Attacker`:
  - reprezentuje klienta, který se snaží autentizovat jako konkrétní uživatel
  - definován IPv4, nebo IPv6 adresou
  - obsahuje informace o počtu pokusů o přihlášení a čas prvního zaslaného požadavku

Kapitola 4

Implementace detekčního modulu

Mnoho implementačních vlastností navrženého programu se odvíjí od požadavku kompatibility se systémem NEMEA. Implementace modulu je provedena v jazyce C++ z několika důvodů:

1. Je kladen velký důraz na úsporu paměti a procesorového času.

2. Implementace většiny částí systému NEMEA, především pak knihovna TRAP, je provedena v jazyce C.

3. C++, oproti jazyku C, poskytuje možnost objektového přístupu. Ten je využit u datových struktur reprezentujících SIP server, uživatele a klienta.

Navržený modul používá jedno vstupní a jedno výstupní komunikační rozhraní, která jsou poskytována knihovnou TRAP. Modul se spouští s parametrem, který definuje nastavení těchto rozhraní. Bezprostředně po spuštění modulu dojde k inicializaci knihovny TRAP a požadovaných rozhraní. Následuje zpracování parametrů modulu popsaných v sekci 4.4. Po úspěšné inicializaci jsou v cyklu vždy data načítána, zpracovávána, ukládána do příslušných datových struktur a vyhodnocována. Po této fázi agregování dat nastává fáze kontroly ukončených útoků a cyklus se opakuje do doby, než je obdržen signál `SIGINT`. Po přerušení hlavní smyčky modulu následuje hlášení útoků, které se stále nalézají v paměti programu, uvolnění datových struktur a samotné ukončení programu.

V následujících částech této kapitoly jsou podrobněji popsány jednotlivé fáze běhu programu a vybrané zajímavé implementační detaily.

Tabulka 4.1: Přijímaná UniRec políčka

Název	Popis
SRC_IP	IPv4 nebo IPv6 adresa serveru
DST_IP	IPv4 nebo IPv6 adresa útočníka
LINK_BIT_FIELD	číslo linky, kde byla data zachycena
PROTOCOL	TCP nebo UDP
TIME_FIRST	čas zachycení komunikace
SIP_MSG_TYPE	zda se jedná o SIP metodu či odpověď
SIP_STATUS_CODE	číselný kód SIP odpovědi
SIP_CSEQ	určuje metodu, ke které se odpověď vztahuje
SIP_CALLING_PARTY	uživatelské jméno, na které je útok veden

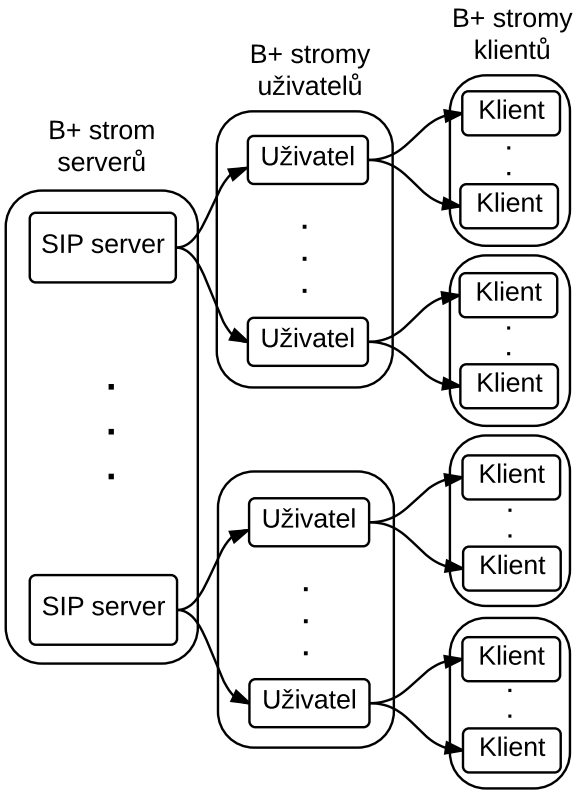
4.1 Vstupní rozhraní a vyžadovaná komunikace

Navržený modul očekává na vstupním rozhraní SIP komunikaci v datovém formátu UniRec. Tabulka 4.1 popisuje všechna UniRec políčka potřebná k detekci útoků hrubou silou. Navržený detekční algoritmus vyžaduje pouze 9 z 23 možných exportovaných UniRec políček, přičemž políčka `LINK_BIT_FIELD` a `PROTOCOL` mají informativní charakter a jsou uvedeny pouze v hlášení detekovaného útoku. Načtení dat ze vstupního rozhraní zajišťuje volání knihovní funkce `trap_recv()`.

4.2 Ukládání dat

Z hlediska návrhu a nároku na šetření procesorového času je zapotřebí datové struktury navržené v sekci 3.3 ukládat do abstraktivního datového typu, nad kterým bude časově nenáročné provádět operace, které jsou klíčové a časté v navrženém algoritmu. Jedná se především o vložení, nalezení a odstranění prvku. Dále je potřeba přes všechny prvky jednoduše iterovat. Tento abstraktní datový typ je implementován jako B+ strom, který vkládání, nalezení a odstranění prvku řeší v logaritmickém čase a iterace přes všechny prvky ve stromu je realizována v čase lineárním.

V programu se nacházejí 2 základní B+ stromy. V prvním z nich jsou uloženy SIP servery komunikující pomocí IPv4 a v druhém SIP servery komunikující pomocí IPv6. V každém uloženém serveru se nachází ukazatel na B+ strom uživatelů. Tento strom uživatelů je unikátní pro každý server a jsou v něm uložena uživatelská jména, která jsou na daném serveru registrována. Pod každým uživatelským jménem se navíc nachází další B+ strom s potenciálními útočníky, posílající `REGISTER` požadavky na toto konkrétní jméno na určitém serveru. Tento koncept je znázorněn na obrázku 4.1. Při běžném útoku hrubou silou (nikoliv distribuovaném) bude vytvořen jen jeden prvek ve stromu serverů, jeden prvek ve stromu uživatelů a jeden prvek ve



Obrázek 4.1: Ukládání dat

stromu klientů. Paměťová náročnost této implementace při očekávaném chování útočníků bude tedy malá. Implementace B+ stromu je součástí systému NEMEA.

4.2.1 B+ strom

B+ strom [15] je stromová datová struktura. Skládá se z kořene, vnitřních uzlů a listů. Každý uzel stromu obsahuje pole o velikosti N prvků. Všechny uzly takového stromu mají maximálně N potomků a každý uzel je vždy minimálně z poloviny zaplněn. Charakterizuje se tím, že má v každém uzlu seřazený seznam klíčů a ukazatelů do nižších pater stromu. Na rozdíl od klasického B stromu má B+ strom ve vnitřních uzlech pouze klíče a data jsou uložena až v listech. Všechny listy jsou uloženy ve stejné hloubce a každý z nich obsahuje ukazatel na sousední list.

Tabulka 4.2: Data v hlášení o útoku

Název	Popis
EventID	unikátní číslo vygenerované pro toto hlášení
ObsoleteID	ID hlášení, které má být nahrazeno
TargetIP	adresa SIP serveru
SIPTo	uživatelské jméno, které je cílem útoku
AttemptCount	počet všech pokusů o autentizaci
EventTime	čas odpovědi na první `REGISTER` zprávu
CeaseTime	čas odpovědi na poslední `REGISTER` zprávu
LinkBitField	číslo linky, kde byla data zachycena
Protocol	TCP nebo UDP
Breach	příznak určující, zda došlo k prolomení hesla
BreacherIP	IP adresa útočníka, který prolomil heslo
BreachTime	čas prolomení hesla
Sources	útočníci (IP adresy, počty zpráv, časy prvních zpráv)

4.3 Výstupní rozhraní a hlášení útoků

Z návrhu detekčního algoritmu plyne generování hlášení ve třech různých situacích:

1. Na určitém serveru u konkrétního uživatelského jména došlo k překročení hranice pro počet neplatných pokusů k autentizaci.

2. Došlo k prolomení uživatelského hesla.

3. Při mazání záznamu o útoku z paměti, a to buď při pravidelné kontrole, nebo při ukončení programu.

Vždy jsou ke každému útoku vygenerována minimálně 2 a maximálně 3 hlášení v závislosti na tom, zda došlo k prolomení hesla. Hlášení o útoku je následně pomocí knihovní funkce `trap_send()` odesláno na výstupní rozhraní modulu. Pro exportování útoku přes komunikační rozhraní knihovny TRAP se využívá formát dat JavaScript Object Notation (JSON). Tento datový formát oproti formátu UniRec, dovoluje odesílat pole hodnot. V tomto případě se jedná o seznam útočníků. Tabulka 4.2 popisuje data, která jsou hlášena při detekci útoku hrubou silou.

4.3.1 JavaScript Object Notation – JSON

JSON je podrobně popsán v RFC 7159 [16]. Jedná se o textový datový formát, kódovaný v UNICODE, sloužící k výměně dat nezávisle na programovacím jazyku. Dokáže reprezentovat primitivní datové typy (čísla, booleovské hodnoty,

řetězce, null). Dokáže však také reprezentovat pole a struktury. Struktura tohoto formátu je dána šesticí znaků. Začátek a konec pole `[ ]`; začátek a konec objektu `{ }`; oddělovač názvů `,` (čárka); oddělovač hodnot `:` (dvojtečka). Celý formát je založený na kolekci párů název/hodnota. Ukázka výstupu modulu v tomto formátu s anonymizovanými daty se nachází v příloze A.

4.4 Parametry modulu

Aby se dal program lehce testovat a následně také využívat na různých sítích, musí být jeho klíčové konstanty konfigurovatelné. K přepsání výchozích hodnot těchto konstant slouží volitelné parametry zadávané při spuštění programu:

- **a** (alert threshold): číslo určující hranici mezi běžnou SIP komunikací a pokusy o prolomení uživatelského hesla; udává počet neúspěšných pokusů o autentizaci `REGISTER` požadavků u konkrétního uživatelského jména; po dosažení hranice je vygenerováno hlášení o probíhajícím útoku
- **c** (check memory interval): počet sekund, které musí uplynout mezi pravidelnými kontrolami ukončených útoků
- **f** (free memory delay): počet sekund, po které program drží v paměti každý ukončený útok

Kapitola 5

Testování a vyhodnocení

Pro účely testování jsem implementoval SIP plugin do již existujícího exportního modulu. Plugin dokáže exportovat rozšířené záznamy o SIP tocích, bez kterých by nebyla možná jakákoliv detekce popsaných útoků na aplikační vrstvě. Program byl odladěn nejprve pomocí zachycených dat a poté i na online datech sérií krátkodobých testů. Následně byl proveden dlouhodobý test, kdy byl modul spuštěn po dobu 8 dní. Exportní modul se SIP pluginem byl spuštěn na jedné ze sond sdružení CESNET a zachycené toky byly ukládány jak do souboru, tak posílány přímo do detekčního modulu. Pro účely detekce bylo za tuto dobu zpracováno 8 220 925 síťových toků, což odpovídá 1,7 GB dat. Poznatky a závěry, které z těchto dat plynou jsou diskutovány v následujících sekcích této kapitoly. Testovaný modul pojmenovaný *SIP Brute-Force Detector* (`sip_bf_detector`)[3] je veřejně dostupný.

5.1 Problémy s reálnými SIP servery

Jelikož existuje spousta různých implementací SIP serverů, a odpovídání serverů na neúspěšný pokus o autentizaci není ve specifikaci standardizováno, vyskytl se problém s určováním počtu vyzkoušených hesel v jednotlivých útocích.

V původním návrhu modulu docházelo ke sčítaní počtů zachycených odpovědí typu 401 a 403, výsledný počet pokusů vznikl vydělením tohoto čísla dvěma. Podle RFC 3261 [2] však servery, které odpovídají `401 Unauthorized` na nesprávné heslo, musejí mít v této odpovědi nové údaje, díky kterým může klient provést nový pokus o autentizaci, aniž by poslal `REGISTER` požadavek o zaslání údajů klíčových k vygenerování nové hash z hesla. Tím se počet vyzkoušených hesel zdvojnásobuje. Podobný dopad vzniká v situaci, kdy server nemění hodnotu *nonce*. Útočník, který si je tohoto faktu vědom, nemusí žádat o údaje k vytvoření nové hash, ale může se přímo pokusit registrovat.

[3] `https://github.com/CESNET/Nemea-Detectors/tree/master/sip_bf_detector`

Opět je počet vyzkoušených hesel dvojnásobný. Navíc jsou uživatelé, kteří jsou na takovém serveru registrováni, vystaveni velké bezpečnostní hrozbě. Pokud útočník odchytí komunikaci v níž bude úspěšný pokus o autentizaci, může použít offline slovníkový útok, kterým nalezne kolizi 2. řádu a následně se úspěšně autentizovat. V penetrační sadě nástrojů *SipVicious* [17] je dokonce implementována optimalizace založená na faktu, kdy server hodnotu *nonce* nemění, nejedná se tedy pravděpodobně o zcela výjimečný jev.

Pozorován byl také server, který odpovídal na neúspěšný pokus o autentizaci hodnotou `401 Unauthorized`, ale pouze než byl vyčerpán určitý počet pokusů a následně na tyto požadavky začal odpovídat `403 Try Later`. Po určité době bylo autentizaci možno opakovat.

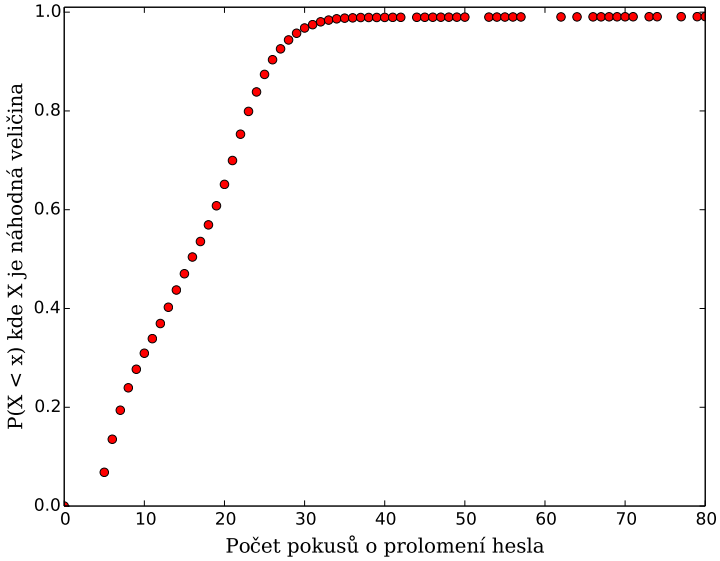
Jak lze vidět, reagování SIP serverů na neúspěšné pokusy o přihlášení je značně diverzní a komplikuje určování skutečného počtu vyzkoušených hesel v útocích. Vhodným východiskem bylo určovat počet pokusů pouze podle odpovědí `401 Unauthorized` a zprávy typu 403 zcela ignorovat. Zpráva 401 je v autentizačním procesu vyžadována vždy, neboť se u ní očekává přítomnost hlavičky *WWW-Authenticate*.

Tento postup selže pouze pokud sever odpovídá na nesprávné heslo zprávou typu 403 a nemění hodnotu *nonce*. V takovém případě by se v datech mohla nacházet jen jedna zpráva typu 401 potřebná k zjištění *nonce* a ostatních hodnot potřebných k vytvoření hash a zbytek komunikace by se skládal výlučně z odpovědí typu 403. Tento případ nebyl v datech detekován, je však vhodné do budoucna modul rozšířit o detekci útoků tohoto typu.

5.2 Hledání hranice pro generování hlášení

Důležitým faktorem při generování hlášení o útocích je vhodně určená hodnota hranice počtu neúspěšných pokusů o autentizaci. Pokud by tato hranice byla nastavena příliš nízko, modul by vytvářel velké množství falešných poplachů. Naopak nastavením hranice příliš vysoko by mohlo dojít k ignorování některých útoků. Na obrázku 5.1 je znázorněna empirická distribuční funkce počtu neúspěšných pokusů o autentizaci v jednotlivých útocích. Z grafu je zřejmé, že více než 90 % útoků zkouší k prolomení zabezpečení uživatelského účtu maximálně 30 různých hesel. Z toho vyplývá, že většina útočníků používá určitou sadu častých hesel, kterou zkoušejí na všechny účty, spíše než aby se snažili o prolomení jednoho konkrétního účtu. Nastavením hranice na hodnotu vyšší než 30 je možné většinu útoků tohoto typu zcela vyfiltrovat a zůstanou jen útočníci, kteří cílí své útoky na konkrétní účty.

Druhým zásadním prvkem při určování hodnoty hranice jsou falešně pozitivní hlášení. V tabulce 5.1 je sepsáno, na kolikátý pokus se v běžné komunikaci podařilo klientům přihlásit (nikoliv prolomením hesla útokem). Kromě zřejmých důvodů, jako jsou zapomenutá hesla, nebo překlepy, má velký vliv na počet pokusů před úspěšnou autentizací také ztrátovost paketů na síti.



Obrázek 5.1: Empirická distribuční funkce počtu neúspěšných pokusů o prolomení hesla v jednotlivých útocích

Tabulka 5.1: Počty pokusů potřebných k úspěšné autentizaci

Počet pokusů potřebných k autentizaci	Počet výskytů
1	167561
2	4802
3	248
4	119
5	89
6	33
7	12
8	15
9	5
$\langle 10; 15)$	19
$\langle 15; 40)$	11
> 40	5

Zkoumáním vstupních dat se zjistilo, že někteří UAC odesílají `REGISTER` požadavek na server například každou hodinu. Za běžné situace lze v datech pozorovat střídání zpráv `401 Unauthorized` a `200 OK`, které odpovídá autentizačnímu schématu popsanému v sekci 1.2.6. Pokud je však síť dočasně zahlcená, nebo se odpovědi serveru z nějakého jiného důvodu nemohou dostat ke klientovi, toto střídání zmizí a v toku dat jsou pouze nedoručené odpovědi

`401 Unauthorized`. Po určitém čase se toto chování ustálí a lze opět pozorovat periodickou úspěšnou autentizaci. Trvá-li však toto chování příliš dlouho, může počet nedoručených zpráv překročit hranici pro generování útoků a bude vytvořeno falešně pozitivní hlášení o útoku.

Z obrázku 5.1 a tabulky 5.1 vyplývá, že vhodná hranice pro generování hlášení je přibližně 20 neúspěšných pokusů.

5.3 Časy mezi neúspěšnými pokusy

Zkoumáním dat bylo zjištěno, že více než 97 % útoků skončí do 10 vteřin a čas mezi jednotlivými zkoušenými hesly byl v 99,9 % útoků kratší než 2 vteřiny. Nastavením parametru pro pozdržení dat v paměti programu na nízkou hodnotu (např. 5 minut) bude program v průběhu chodu vyžadovat menší množství paměti. Může se však stát, že jeden útok bude rozdělen do více hlášení. Pokud budou data držena v paměti příliš dlouho, je pravděpodobné, že bude docházet ke slučování útoků do jednoho hlášení.

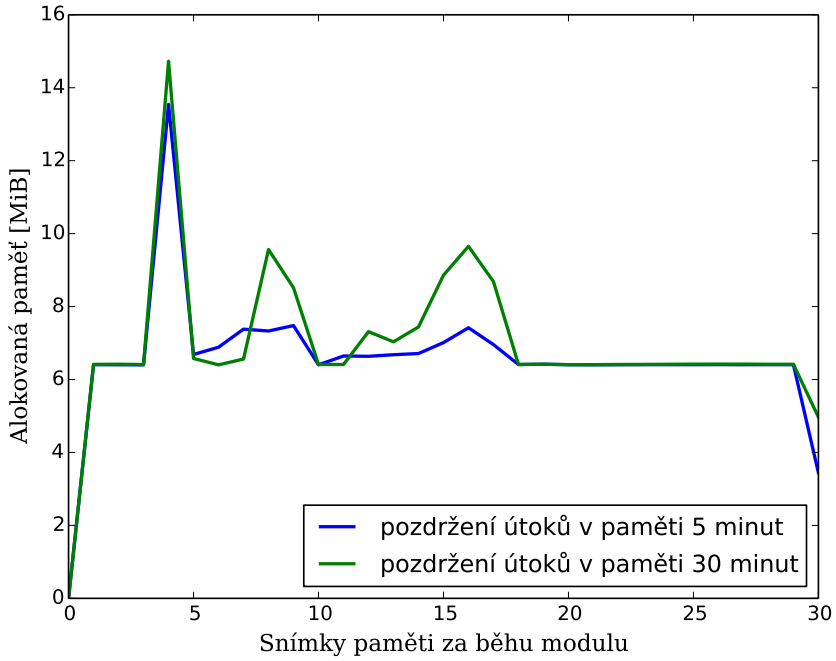
Jelikož útoky trvají relativně krátkou dobu, není je potřeba držet dlouho v paměti programu a implementace 3 typů hlášení je tedy zbytečná. Proto byl do programu přidán další volitelný parametr **`-s`** (skip alerts), který slouží k potlačení generování hlášení prvního, nebo prvního a druhého typu.

5.4 Využití paměti

Značnou část alokované paměti zabírají struktury knihovny TRAP. Samotný detekční algoritmus vyžaduje paměť k dočasnému ukládání informací z načteného síťového toku a reprezentaci SIP serverů, uživatelů a potenciálních útočníků. Z implementace modulu plyne, že při detekci SIP odpovědi na `REGISTER` požadavek dojde k vytvoření tří struktur (pokud ještě vytvořené nejsou), a to serveru, uživatele a klienta.

Na obrázku 5.2 je znázorněn graf množství alokované paměti za běhu programu, a to ve dvou různých konfiguracích. Konfigurace se od sebe liší časem, po který byla data držena v paměti programu, než byla z paměti uvolněna a bylo vygenerováno hlášení 3. typu. Z grafu je na první pohled zřejmé, že na snímku č. 4 došlo ke značnému nárůstu alokovaných prostředků. Po přezkoumání zachycené komunikace bylo zjištěno, že za tuto událost mohou dva faktory:

1. Útok hádáním uživatelských jmen. Jedná se o situaci, kdy útočník posílá na server požadavky typu `REGISTER`, `INVITE` nebo `OPTIONS` a z jeho odpovědí zjišťuje, zda je dané uživatelské jméno na konkrétním serveru registrováno. Po tomto útoku většinou následuje útok za účelem prolomení hesel nalezených uživatelských jmen.



Obrázek 5.2: Porovnání alokované paměti v závislosti na pozdržení dat v paměti programu

2. Některé SIP servery jsou nakonfigurovány tak, že odpovídají zprávou `401 Unauthorized` na požadavek o autentizaci uživatelského jména nehledě na to, zda se toto uživatelské jméno na daném serveru vůbec nachází. Očekávaná odpověď na takovýto požadavek je však `404 Not Found`. Toto chování je implementováno na některých SIP serverech právě jako ochrana před hádáním uživatelských jmen.

V kritický okamžik došlo k rozsáhlému útoku hádáním uživatelských jmen pomocí `REGISTER` zpráv. V tomto útoku bylo vyzkoušeno přibližně 10 000 jmen. Útok tohoto typu sám o sobě není problematický, neboť od serveru se očekává odpověď `404 Not Found`, pokud hádané jméno není na daném serveru registrováno. Detekční modul tyto zprávy ignoruje a ukládá pouze odpovědi `401 Unauthorized`, které se objeví, pokud se dané jméno na serveru vyskytuje. Avšak SIP server, na který byl útok veden, byl nakonfigurován tak, aby odpovídal zprávou `401 Unauthorized` v obou případech. Tím došlo k obrovskému růstu B+ stromu s uživateli pod tímto konkrétním serverem a alokaci většího množství paměti. Těchto skenů uživatelských jmen bylo zaznamenáno několik, ale již nepřesáhly 1 000 zkoušených uživatelských jmen.

Jak je patrné z grafu, nastavením parametru pro pozdržení dat v paměti

Tabulka 5.2: Statistika detekovaných událostí

Počet detekovaných událostí	10 099
Počet falešně pozitivních událostí	3
Počet událostí, kdy došlo k prolomení hesla	2
Počet událostí, které neměly skenovou formu	38
Celkový počet vyzkoušených hesel	3 506 346
Medián počtu vyzkoušených hesel v události	24
Nejčastější počet vyzkoušených hesel v události	22 (1 372×)
Průměrný počet vyzkoušených hesel v události	347,3
Maximální počet vyzkoušených hesel v události	610 794
Četnost událostí s počtem vyzkoušených hesel > 1 000	17
Průměrná délka události	44,5 sekund
Nejdéle trvající událost (734 pokusů o prolomení)	26 h 33 min
Počet událostí s dobou trvání delší než 1 minuta	32
Zastoupení IPv4 vs. IPv6 v událostech	100 % vs. 0 %
Zastoupení UDP vs. TCP v událostech	100 % vs. 0 %

na vyšší hodnotu (např. 30 minut) nedojde k markantní změně ve velikosti alokované paměti. Není tedy nutné nastavit tento parametr na co nejmenší možnou hodnotu.

Pokud by probíhalo více útoků tohoto typu zároveň, program by mohl alokovat příliš velké množství paměti. V budoucnu by bylo vhodné upravit modul tak, aby tyto skeny agregoval optimálním způsobem, nebo nastavit hranici pro počet alokovaných prvků v jednotlivých B+ stromech, čímž by se dalo kontrolovat maximální množství využívané paměti.

5.5 Závěrečná statistika

Program byl spuštěn nad daty z testovacího týdne v konfiguraci:

- hranice pro generování hlášení podle počtu selhaných autentizací: **20**
- frekvence kontroly útoků, které již pominuly: **5 minut**
- čas, po který jsou útoky drženy v paměti, než je program odstraní: **30 minut**

V tabulce 5.2 jsou sepsána nejdůležitější pozorování z hlášených událostí. Za událost je považován pokus o prolomení hesla konkrétního uživatele na určitém serveru jedním nebo více klienty v daný čas. Jak je možno vidět, drtivá většina útoků je vedená formou skenu přes uživatele. Útočník, který vede tento útok, využívá většinou od 20 do 100 hesel, která zkouší na určitý rozsah uživatelských jmen na serveru. Typicky se jedná o uživatelská jména 0

až 1 000, avšak v nahlášených událostech lze pozorovat i sken, který využíval křestní jména (leo, owen, harriet, atd.). Útoky formou skenu pokrývají drtivou většinu hlášených událostí a méně než 1 % útoků je cíleno na konkrétního uživatele.

Přezkoumáním nahlášených událostí a zdrojových dat byly odhaleny 3 falešně pozitivní hlášení, která vznikla způsobem popsaným v sekci 5.2.

Velkým úspěchem je detekce dvou úspěšných útoků. Oba byly vedeny ze stejné IP adresy v České republice. Cílem prvního útoku byl SIP server v Jižní Dakotě a k prolomení hesla uživatele *701* bylo vyzkoušeno 5 838 různých hesel v rozmezí 6 minut. Druhým cílem se stal uživatel *6001* registrovaný na serveru v Brazílii. Na prolomení jeho hesla potřeboval útočník 4 052 pokusů odeslaných v rámci 2 minut. Hlášení 3. typu o druhém útoku s anonymizovanými daty je dostupné v příloze A.

5.6 Použité softwarové nástroje

- Celý vývoj programu byl verzován pomocí verzovacího systém *Git* [18].
- K odladění chyb v práci s pamětí byl využit nástroj *Valgrind* [19].
- K vytvoření spustitelného souboru v rámci systému NEMEA byl použit balík *Autotools – Autoconf* [20], *Automake* [21] , *Libtool* [22].

Závěr

V první části této práce jsem popsal základní funkce protokolu SIP a jeho technické aspekty. Názorně jsem demonstroval proces autentizace, od kterého se odvíjí celá praktická část této práce. Dále jsem představil několik nejčastějších podvodů, které souvisejí s technologií VoIP, a útoků, které se zaměřují na slabá místa bezpečnosti protokolu SIP. Podrobněji jsem rozebral útok hrubou silou za účelem prolomení hesel uživatelů registrovaných na SIP serverech. V závěru teoretické části je popsáno detekční prostředí a také celý proces monitorování síťových toků.

V praktické části jsem nejprve provedl analýzu požadavků na výsledný software sloužící k detekci útoků hrubou silou na hesla uživatelů registrovaných na SIP serverech. Na základě této analýzy a poznatků o autentizačním procesu pomocí protokolu SIP jsem navrhl algoritmus schopný tyto útoky detekovat. Díky analýze požadavků na software a navrženému detekčnímu algoritmu se mi úspěšně podařilo implementovat program a integrovat ho do detekčního systému NEMEA. Program je díky svému návrhu schopný pracovat i ve velkých počítačových sítích a splnil funkční a nefunkční požadavky, které na něj byly kladeny.

Navržený modul je určený pro nepřetržitý běh a je schopný detekovat i distribuované útoky na jednotlivé uživatele. Program dokáže detekovat útoky nezávisle na použitém transportním protokolu (TCP/UDP) a také nezávisle na verzi internetového protokolu (IPv4/IPv6). Program je konfigurovatelný pomocí přepínačů. Nejdůležitějším parametrem je hodnota hranice počtu neúspěšných autentizací, jejíž překročení je považováno za útok. Dalším důležitým parametrem je doba, po kterou program uchovává data o potenciálním útoku v paměti, než je útok vyhodnocen jako ukončený a data jsou z paměti programu uvolněna. Výchozí hodnoty těchto parametrů byly experimentálně nalezeny pomocí analýzy reálného provozu.

Pro účely testování jsem implementoval SIP plugin do již existujícího exportního modulu. Plugin dokáže exportovat rozšířené záznamy o SIP tocích, bez kterých by nebyla možná jakákoliv detekce popsaných útoků na aplikační

vrstvě.

Po odladění detekčního modulu na zachycených datech a sérii krátkodobých online testů jsem vytvořený modul podrobil dlouhodobému testu. Díky datům získaným z dlouhodobého testu bylo zjištěno více různých reakcí SIP serverů na neúspěšné pokusy o autentizaci a detekční algoritmus byl tomuto faktu lehce přizpůsoben za účelem zajištění co nejpřesnějších hodnot v generovaných hlášeních.

Důležitým výsledkem této práce je zjištění chování útočníků. Oproti předpokládanému průběhu útoku, kdy si jeden útočník vytipuje několik uživatelských jmen a na tato jména posílá velké množství různých hesel, bylo zjištěno, že většina útočníků zkouší menší množství hesel (mezi 20 až 40) na větší množství uživatelů na serveru (kolem 1 000). Útočníci tak spojují dva útoky hrubou silou do jednoho. Hádají uživatelská jména a zároveň pravděpodobně používají připravenou sadu častých hesel, která u každého uhodnutého uživatelského jména rovnou zkoušejí. Velkým úspěchem je také detekování dvou útoků v reálné síti, kdy došlo k prolomení hesla.

Tvorbou této práce a vývojem detekčního programu jsem získal značné množství jak teoretických, tak praktických zkušeností s monitorováním síťových toků a technologií VoIP. Velkými výhodami při implementaci detekčního programu bylo detekční prostředí NEMEA a možnost nasazení modulu na reálné síti CESNET2. Pokud by byl modul testován pouze na simulovaných útocích, nedošlo by ke zjištění značně diverzního chování různých implementací SIP serverů ani k pozorování chování reálných útočníků.

Na základě poznatků z testování modulu by bylo v budoucnu vhodné program rozšířit především o efektivnější správu paměti programu. Vyvinutý modul je již nyní součástí veřejné verze systému NEMEA pod názvem *SIP Brute-Force Detector* a může být kýmkoliv využíván k detekci útoků. Výstup může sloužit k ochraně před následnými telekomunikačními podvody a potenciální finanční ztrátou.

Literatura

[1] Communication Fraud Control Association: Global Fraud Loss Survey. *Press Release, Roseland, NJ (CFCA) October*, 2015, [cit. 2016-04-05]. Dostupné z: `http://www.cfca.org/pdf/survey/2015_CFCA_Global_Fraud_Loss_Survey_Press_Release.pdf`

[2] Rosenberg, J.; Schulzrinne, H.; Camarillo, G.; aj.: SIP: session initiation protocol. RFC 3261, RFC Editor, Červen 2002, [cit. 2016-04-05]. Dostupné z: `http://www.rfc-editor.org/rfc/rfc3261.txt`

[3] Simmons, K.: VOIP Vs PSTN: The Pros And Cons. Březen 2013. Dostupné z: `http://www.mytechlogy.com/IT-blogs/722/voip-vs-pstn-the-pros-and-cons`

[4] Hallock, J.: A brief history of VoIP. *Evolution and Trends in Digital Media Technologies*, 2004.

[5] Goode, B.: Voice over internet protocol (VoIP). *Proceedings of the IEEE*, ročník 90, č. 9, 2002: s. 1495–1517, ISSN 0018-9219, doi:10.1109/JPROC.2002.802005.

[6] Johnston, A. B.: *SIP: understanding the session initiation protocol*. Artech House, 2009, ISBN 15-805-3655-7.

[7] Dwivedi, H.: *Hacking VoIP: protocols, attacks, and countermeasures*. No Starch Press, 2009, ISBN 978-1-59327-163-3.

[8] Shankar: Premium Rate Service Fraud and International Revenue Share Fraud. Listopad 2014, [cit. 2016-04-05]. Dostupné z: `http://frslabs.com/frsblog/2014/11/26/premium-rate-service-fraud-international-revenue-share-fraud-note-difference-accurately-detect-prevent`

[9] Bignell, C.: The different frauds: Interconnect Bypass. Srpen 2012, [cit. 2016-04-05]. Dostupné z: http://fraudforthought.blogspot.cz/2012/08/the-different-frauds-interconnect-bypass.html

[10] El-Moussa, F.; Mudhar, P.; Jones, A.: Overview of SIP attacks and countermeasures. In *Information Security and Digital Forensics*, Springer, 2009, s. 82–91.

[11] Gauci, S.: Storming SIP Security. Únor 2008, [cit. 2016-04-05]. Dostupné z: https://resources.enablesecurity.com/resources/22_29_storming_sip.pdf

[12] CESNET z.s.p.o.: CESNET / Síť CESNET2. [cit. 2016-04-05]. Dostupné z: https://www.cesnet.cz/sluzby/pripojeni/sit-cesnet2

[13] Brownlee, N.; Mills, C.; Ruth, G.: Traffic Flow Measurement: Architecture. RFC 2722, RFC Editor, Říjen 1999, [cit. 2016-04-05]. Dostupné z: http://www.rfc-editor.org/rfc/rfc2722.txt

[14] Bartos, V.; Zadnik, M.; Cejka, T.: Nemea: Framework for stream-wise analysis of network traffic. *CESNET Technical Report 9/2013*, 2013.

[15] TutorialCup: Concepts of B+ Tree and Extensions - B+ and B Tree index files in DBMS. 2015, [cit. 2016-04-17]. Dostupné z: https://www.tutorialcup.com/dbms/b-tree.htm

[16] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, RFC Editor, March 2014, [cit. 2016-04-18]. Dostupné z: http://www.rfc-editor.org/rfc/rfc7159.txt

[17] SIPVicious (Tools for auditing SIP based VoIP systems). [cit. 2016-05-04]. Dostupné z: https://github.com/sandrogauci/sipvicious

[18] Git – Distributed version control system. [cit. 2016-05-04]. Dostupné z: https://git-scm.com/

[19] Valgrind – System for debugging and profiling Linux programs. [cit. 2016-05-04]. Dostupné z: http://valgrind.org/

[20] Autoconf. [cit. 2016-05-04]. Dostupné z: http://www.gnu.org/software/autoconf/

[21] Automake. [cit. 2016-05-04]. Dostupné z: https://www.gnu.org/software/automake/

[22] Libtool – A generic library support script. [cit. 2016-05-04]. Dostupné z: http://www.gnu.org/software/libtool/

Příloha **A**

Výstup modulu

Ukázka 3. typu hlášení s anonymizovanými daty. Útok trval necelé 2 minuty a byl veden z jedné IP adresy. Cílem bylo uživatelské jméno *6001*. V tomto útoku došlo k prolomení hesla. Čas prolomení se nachází v položce `BreachTime`.


```
{
    "EventID": "2016042004042100003",
    "ObsoleteID": "2016042004042100002",
    "TargetIP": "189.61.XXX.XXX"
    "SIPTo": "6001@189.61.XXX.XXX",
    "EventTime": "2016-04-20 04:04:21",
    "CeaseTime": "2016-04-20 04:06:14",
    "AttemptCount": 4052,
    "Breach": true,
    "BreacherIP": "147.32.XXX.XXX",
    "BreachTime": "2016-04-20 04:06:14",
    "LinkBitField": 0,
    "Protocol": "UDP",
    "Sources": [
        {
            "AttemptCount": 4052,
            "EventTime": "2016-04-20 04:04:21",
            "SourceIP": "147.32.XXX.XXX"
        }
    ],
}
```

Příloha B

Seznam použitých zkratek

API Application Programming Interface

DDoS Distributed Denial of Service

DNS Domain Name System

DoS Denial of Service

FQDN Fully Qualified Domain Name

HTTP Hypertext Transfer Protocol

IETF Internet Engineering Task Force

IPFIX Internet Protocol Flow Information eXport

IP Internet Protocol

IRSF International Revenue Share Fraud

JSON JavaScript Object Notation

NEMEA Network Measurements Analysis

PBX Private Branch Exchange

PSTN Public Switched Telephone Network

RFC Request for Comments

RTP Real-time Transport Protocol

SDP Session Description Protocol

SIP Session Initiation Protocol

TCP Transmission Control Protocol

TLS Transport Layer Security

TRAP Traffic Analysis Platform

UA User Agent

UAC User Agent Client

UAS User Agent Server

UDP User Datagram Protocol

UniRec Unified Record

URI Uniform Resource Identifier

VoIP Voice over Internet Protocol

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `doc`
  - `html` ..... dokumentace zdrojového kódu ve formátu HTML
  - `pdf` ..... dokumentace zdrojového kódu ve formátu PDF
- `src`
  - `impl` ..... balík se zdrojovými kódy implementace detekčního modulu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text`
  - `BP_Jansky_Tomas_2016.pdf` ..... text práce ve formátu PDF
  - `zadani_prace.pdf` ..... zadání bakalářské práce ve formátu PDF

Příloha D

Instalace detekčního modulu

D.1 Závislosti

Doporučený systém pro instalaci systému Nemea-framework a detekčního modulu je `CentOS/7`. Před instalováním `Nemea-framework` a detekčního modulu `sip_bf_detector` je zapotřebí mít nainstalované následující závislosti:

- gcc
- gcc-c++
- libtool
- libxml2-devel
- make
- pkg-config

Všechny závislosti lze nainstalovat pomocí:

```
sudo yum install -y gcc gcc-c++ libtool libxml2-devel make pkg-config
```

Na novějších systémech použijte `dnf(8)` místo `yum(8)`.

D.2 Instalace Nemea-framework

Na přiloženém CD jsou archivy se zdrojovými kódy. Zkopírujte adresář `src` z CD do domovského adresáře.

1. Rozbalení:

   ```
   tar -xvf src/impl/nemea-framework-2.0.0.tar.gz
   ```

2. Přechod do nově vytvořeného adresáře:

```
cd nemea-framework-2.0.0/
```

3. Konfigurace a kompilace:

```
./configure --prefix=/usr/ --libdir=/usr/lib64 \
--bindir=/usr/bin/nemea && make
```

4. Instalace:

```
sudo make install
```

Nyní došlo k nainstalování Nemea-framework do `/usr/lib64/`.

D.3 Instalace SIP Brute-Force Detector

Přesuňte se do domovského adresáře.

1. Rozbalení:

```
tar -xvf src/impl/sip_bf_detector-1.0.0.tar.gz
```

2. Přechod do nově vytvořeného adresáře:

```
cd sip_bf_detector-1.0.0/
```

3. Konfigurace a kompilace:

```
./configure --prefix=/usr/ --libdir=/usr/lib64 \
--bindir=/usr/bin/nemea && make
```

4. Instalace:

```
sudo make install
```

Nyní došlo k nainstalování modulu `sip_bf_detector` do `/usr/bin/nemea/`. Exportní modul `flow_meter`, který dokáže exportovat toky rozšířené o políčka SIP protokolu, je k nalezení na: `https://github.com/CESNET/Nemea-Modules/tree/master/flow_meter`. Pro spuštění modulu `flow_meter` je zapotřebí knihovna `libpcap`.

Příloha E

Spuštění detekčního modulu

Detailní popis spuštění detekčního modulu je popsán v souboru *README.md*, který je přiložen v balíku se zdrojovými kódy modulu.

Pro spuštění modulu s anonymizovanými daty umístěnými na CD je zapotřebí:

1. Přesunutí se do domovského adresáře.
2. Rozbalení anonymizovaných dat:

```
tar -xvf src/impl/sipdata.tar.gz
```

3. Přechod do adresáře s nainstalovaným modulem:

```
cd /usr/bin/nemea
```

4. Spuštění modulu s daty:

```
./sip_bf_detector -i "f:~/sipdata_anon.trapcap,"\
"f:~/alerts:w" -a 20 -c 300 -f 1800
```

5. Došlo k vygenerování hlášení do souboru `~/src/impl/alerts`. Hlášení jsou v binárním souboru. Pro převedení do čitelné formy je možné využít například:

```
strings ~/alerts
```