VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

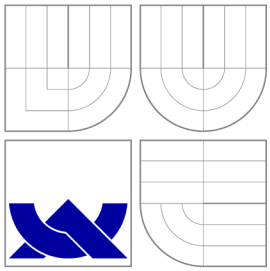
IDENTIFIKACE POMOCÍ POŽADAVKŮ HTTP

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

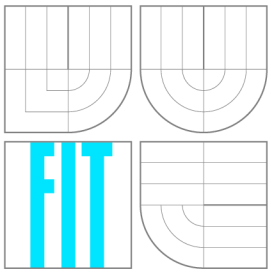
AUTOR PRÁCE
AUTHOR

JAKUB JELEŇ

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IDENTIFIKACE POMOCÍ POŽADAVKŮ HTTP

HTTP-REQUEST-BASED IDENTIFICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB JELEŇ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LIBOR POLČÁK

BRNO 2015

Abstrakt

Táto bakalářska práce sa zabýva identifikací pomocí požadavků HTTP v siťovém provoze. Je zde vysvětlen pricip komunikace HTTP a identifikace v daném protokole. Taktéž je vytvořen návrh aplikace, která identifikuje webový porhlížeč a následně je tato aplikace implementována. Aplikace je navržena jako samostatný modul, který je možné zařadit do projektů Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace. Poté jsou provedeny experimenty s daným nástrojem pro ověření funkčnosti a užitečnosti nastroje.

Abstract

This bachelor thesis deals with the identification using HTTP requests in network traffic. It explains principles of HTTP communications and identification. Additionally the application design is created, which identifies web browser and then this application is implemented. The application is designed as a separate module which can be integrated into projects Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace. Then are executed experiments with that tool to verify the functionality and utility of this tool.

Klíčová slova

HTTP, identifikace webového prohlížeče, soukromí, bezpečnosť.

Keywords

HTTP, identification of web browser, privacy, security.

Citace

Jakub Jeleň: Identifikace pomocí požadavků HTTP, bakalářská práce, Brno, FIT VUT v Brně, 2015

Identifikace pomocí požadavků HTTP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Libora Polčáka

.....
Jakub Jeleň
20. května 2015

Poděkování

Chtěl bych poděkovat svému vedoucímu bakalářskej práce Ing. Liboru Polčákovi, za věcné připomínky, obornú pomoc a trpělivost při řešení.

Obsah

Kapitola 1

Úvod

Internet ako celosvetová komunikačná sieť sa stáva čoraz väčším lákadlom útočníkov, ktorých cieľom je poškodiť užívateľa. Poškodenie užívateľa môže spočívať v napadnutí jeho počítača škodlivým softvérom, ako aj k odcudzeniu jeho súkromných informácií tzv. ukradnutie identity. Preto je potrebné dodržiavať bezpečnostné zásady a dobre zvážiť zverejnenie osobných údajov na pochybné internetové stránky. Je pravda, že aj pri najväčšej snahe predísť odcudzeniu identity sa nám to nemusí podariť. Preto ak k takémuto ukradnutiu identity dôjde, je potrebné útočníka identifikovať a zhromaždiť potrebné údaje na dokázanie jeho viny.

Pre identifikáciu trestnej činnosti slúžia nástroje na zákonné odpočúvanie a analýzu dát. Existuje veľké množstvo protokolov, niektoré sú často používané niektoré zriedka. Jeden z najrozšírenejších protokolov je HyperText Transfer Protocol (HTTP). Tento protokol poskytuje jednoduchú výmenu informácií medzi klientom a serverom.

Táto práca sa zameriava na identifikáciu užívateľa na základe sieťovej komunikácie pomocou požiadaviek HTTP. V nasledujúcej kapitole číslo 2 je rozobraný protokol HTTP, čo sa týka jeho histórie, postupného vývinu a komunikácie. Ďalej sú vysvetlené metódy podporujúce zasielanie požiadaviek klienta, hlavičky a stavové kódy servera na prijatú požiadavku. V kapitole 3 sú definované pojmy identita a identifikovateľnosť. Následne je popísaná identifikácia v HTTP. V kapitole 4 je popísaný nástroj na zákonné odpočúvanie a činnosť. Taktiež je vysvetlená dynamická identita v danom nástroji.

Následne z týchto informácií je vytvorený návrh aplikácie na identifikáciu pomocou požiadaviek HTTP. Tento návrh je popísaný v kapitole 5, kde je vysvetlená jeho základná činosť. V nasledujúcej kapitole číslo 6 je popísaná implementácia tohto nástroja. Sú tam vysvetlené implementačné riešenia rozobrané v návrhu aplikácie. V predposlednej kapitole číslo 7 sú vykonávané experimenty s aplikáciou. Experimenty, ktoré boli vykonávané sa zameriavali na funkčnosť aplikácie ako aj použiteľnosti aplikácie.

V predposlednej kapitole 8 je zhodnotená doterajšia práca na danej aplikácii a popísaná možnosť nasadenia aplikácie.

Kapitola 2

Protokol HTTP

Hypertext transfer protocol (HTTP) je protokol na prenos dokumentov medzi servermi a klientmi v hypertextovom internetovom informačnom systéme *World Wide Web* (WWW). Pracuje na aplikačnej vrstve *OSI* [1] *modelu* na princípe *požiadavka-odpoveď*. Klient posiela požiadavky (obsahujúce cestu ku stránke, ktorú požaduje) a server odpovie.

2.1 História

Prvá špecifikácia HTTP protokolu a formátu *HyperText Markup Language* (HTML) bola navrhnutá v roku 1989, za ktorou stojí *Tim Berners-Lee*. Od tejto doby vznikli tri verzie protokolu. V dnešnej dobe za využíva HTTP verzia 1.1 (HTTP/1.1), pričom už je známe, že sa pracuje na dalšej revízii protokolu na verziu HTTP/2.0.

Prehľad dostupných verzií:

- HTTP/0.9 v roku 1990 nebola špecifikovaná ako *RFC*.
- HTTP/1.0 v roku 1996 špecifikovaná v *RFC 1945* [6].
- HTTP/1.1 základná verzia vyšla v roku 1997 špecifikovaná v *RFC 2068* [7], neskôr aktualizovaná verzia vyšla v roku 1999, ktorá je špecifikovaná *RFC 2616* [8]. Posledná revízia tohto protokolu vyšla v roku 2014 špecifikovaná v *RFC 7230 až 7235* [9, 10, 11, 12, 13, 14].
- HTTP/2 najnovšia verzia, ktorá vyšla roku 2015 v štandarde *RFC 7540* [15].

2.1.1 HTTP/0.9

Prvá verzia HTTP protokolu, ktorá bola založená len na jedinej metóde s názvom **GET**. Pokiaľ *klient* chce získať dokument so *serveru*, pripojí sa na server a zašle príkaz GET. Server dostane správu a pošle klientovi požadovaný dokument. Ukážka typickej komunikácie na obr. 2.1.

[1] Model, ktorý funkčne rozdeľuje *sieťové protokoly* do siedmich vrstiev

```
Požiadavka klienta:
  GET /index.html

Odpoveď servera:
  <BOLD> Ahoj svet ! </BOLD>
```

Obrázek 2.1: Príklad HTTP komunikácie verzia 0.9.

2.1.2 HTTP/1.0

Verzia protokolu HTTP/1.0 vznikla ako nadmnožina protokolu HTTP/0.9 podporujúca všetky vlasnosti v pôvodnom protokole, pretože bolo potrebné zachovať spätnú kompatibilitu protokolu. Hlavným dvôvodom vzniku novej verzie HTTP protokolu bola neschopnosť protokolu HTTP/0.9 zahrnúť do HTTP požiadaviek a odpovedí *meta informácie* [2] o aktuálnom prenose. Tento nedostatok bol vyriešený pridaním hlavičiek pre požiadavky, dáta a aj odpovede. Pridané boli nové metódy na prenos dát ako **HEAD, POST** a aj stavové kódy v odpovedi, ktoré upresňujú, ako bola odpoveď serverom spracovaná. Ďalšie metódy, ktoré sú implementované v protokole sú **PUT, DELETE, LINK a UNLINK**, avšak ako vychádza s RFC jejich implementácia nie je dôsledná a správna vo väčšine HTTP/1.0 aplikácii. Tento protokol je spoľahlivejší a flexibilnejši ako jeho predchodca. Ukážka typickej komunikácie na obr. 2.2.

```
Požiadavka klienta:
  GET /index.html HTTP/1.0
  User-agent: Netscape 4.0
  Referer: http://www.priklad.sk

Odpoveď servera:
  HTTP/1.0 200 Ok
  Server: Apache
  Content-type: text/html
  Content-lenght: 200

  <HTML><BOLD> Obsah HTML dokumentu </BOLD></HTML>
```

Obrázek 2.2: Príklad HTTP komunikácie verzia 1.0.

2.1.3 HTTP/1.1

V dnešnej dobe najpoužívanejšia verzia protokolu prešla mnohými zmenami. Medzi zmeny patria napríklad pridanie podpory pre prenos viacej *HTML* dokumentov pri jednom spojení (trvalé pripojenie), podporu virtuálnych serverov na jednej sieťovej adrese a prenos vybranej časti dokumentu. Metódy **PUT a DELETE** boli plne implementované a proto-

[2] *Meta informácie* sú informácie o dokumente rôzneho charakteru (popis stránky, kľúčové slová, dátum vypršania platnosti, kódovanie ...)

kol bol rozšírený o nové metódy **CONNECT, OPTIONS a TRACE**. Ukážka typickej komunikácie na obr. 2.3.

```
Požiadavka klienta:
  GET /hello.txt HTTP/1.1
  User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
  Host: www.example.com
  Accept-Language: en, mi

Odpoveď servera:
  HTTP/1.1 200 OK
  Date: Mon, 27 Jul 2009 12:28:53 GMT
  Server: Apache
  Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
  ETag: "34aa387-d-1568eb00"
  Accept-Ranges: bytes
  Content-Length: 100
  Vary: Accept-Encoding
  Content-Type: text/plain

  Hello World! My payload includes a trailing CRLF.
```

Obrázek 2.3: Príklad HTTP komunikácie verzia 1.1.

2.1.4 HTTP/2.0

Najnovšia verzia protokolu, ktorá umožnuje efektívnejšie využivanie siete, zníženie odozvy zavedením kompresie hlavičky a umožnuje viac súbežných prenosov v rovnakom sieťovom pripojení pri dodržaní kompatibility s HTTP/1.1.

Komunikácia pomocu tohto protokolu, ak klient nevie či server podporuje HTTP/2 je nasledovná. Komunikácia začína zaslaním požiadavku s hlavičkamiu `Upgrade: h2c` a `HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>`, viď obrázok 2.4.

```
Požiadavka klienta:
  GET / HTTP/1.1
  Host: server.example.com
  Connection: Upgrade, HTTP2-Settings
  Upgrade: h2c
  HTTP2-Settings: <base64url encoding of HTTP/2 SETTINGS payload>
```

Obrázek 2.4: Príklad spustenia HTTP komunikácie verzia 2.

Pri obdržaní takejto požiadavky môžu nastať dva prípady. Server, ktorý túto verziu protokolu nepodporuje može odpovedať na požiadavku akoby sa jednalo o protokol HTTP/1.1.

V takomto prípade sa zaslané hlavičky v požiadavke špecifikujúce HTTP/2 spojenie ignorujú. Takúto opoveď je možné vidieť na obrázku 2.5.

Ale ak server podporuje túto verziu protokolu odpovedá so stavovým kódom *101* a server može začať odosielať dáta pomocou HTTP/2. Tieto dáta musia obsahovať odpoveď na požiadavku, ktorá inicializovala spojenie na verziu HTTP/2, viď obrázok 2.6.

```
Odpoveď servera:
  HTTP / 1.1 200 OK
  Content-Length: 243
  Content-Type: text / html
  Date: Mon, 27 Jul 2009 12:28:53 GMT
  Vary: Accept-Encoding
```

Obrázek 2.5: Príklad odpoveďe, ak server nepodporuje HTTP verziu 2.

```
Odpoveď servera:
  HTTP/1.1 101 Switching Protocols
  Connection: Upgrade
  Upgrade: h2c
```

Obrázek 2.6: Príklad odpoveďe, ak server podporuje HTTP verziu 2.

2.2 Komunikácia

Komunikácia HTTP je založená medzi *klientom* a *serverom*, podľa modelu komunikácie *klient-server* inak tiež známa ako komunikácia požiadavka-odpoveď. Táto komunikácia zvyčajne prebieha nad *Transmission Control Protocol* (TCP) spojením, kde požadovaný server načúva štandardne na porte *80*.

Príklad typickej komunikácie klient-server:

1. Klient zadá do prehliadača požadovanú stránku typicky ako URL [3].
2. Na klientskej strane dôjde k vyhodnoteniu zadanej *domény*[4] a prekladu pomocou služby DNS [5] na IP adresu servera.
3. Pomocou protokolu TCP sa nadviaže spojenie so serverom na zistenej IP adrese.
4. Klient zašle HTTP požiadavku na server napríklad pomocou metódy **GET**.
5. Server príjme zaslanú požiadavku, vykoná jej spracovanie a zašle odpoveď klientovi.

[3] *Uniform Resource Locator* - je univerzálny formát mien používaný na označenie zdroja na internete
[4] Doména je skratka s ang. *Domain Name*, ktorá určuje jedinečný názov serveru (skupinu počítačov)
[5] DNS zakladnou službou je mapovanie (prevádzanie) doménových adries na IP adresy

- Spracovanie prebehno úspešne a vráti požadovaný dokument.
- Pri spracovaní bolo zistené, že dokument sa nachádza niekde inde.
- Nastala klientska chyba (prijatá požiadavka nie je správna).
- Nastala chyba serveru.

Ako je možné vidieť na obrázku 2.3, HTTP požiadavka sa skladá z názvu použitej metódy (prvý riadok) a následne zoznamu hlavičiek, ktoré upresňujú požiadavku. Pri zasielaní HTTP požiadavky je možné použiť ľubovoľnú metódu, ktorá je dostupná v protokole. Prehľad metód a ich sémantika je vysvetlená nižšie. Taktiež aj odpoveď servera sa skladá zo stavového kódu a zoznamu hlavičiek obr. 2.3, ktoré sú vysvetlené nižšie.

Metódy

- **GET** - Primárny mechanizmus na vyhľadávanie informácií na serveri. Získava daný dokument na základe zadaného *URL*. Predávanie metainformácií pomocou tejto metódy môže užívateľ ľahko zmeniť jednoduchou modifikáciou zadaného *URL*. Takisto informácie o *URL* sú ukladané do histórie webového prehliadača alebo do logovacieho súboru webového serveru. Preto by sa na predávanie citlivých informácií (heslo) nemala používať táto metóda. Server na takúto požiadavku reaguje zaslaním odpovede skladajúcej sa z odpovedajúceho dokumentu, stavového kódu a zoznamu hlavičiek od servera.

- **HEAD** - Táto metóda je zhodná s metódou GET s tým rozdielom, že server nesmie poslať dokument v odpovedi. Metóda sa často využíva na testovanie *hypertextových odkazov*, ako je ich platnosť, dostupnosť a nedávna úprava.

- **POST** - Slúži na predávanie dát na server. Obsah správy sa nevkladá do *URL*, neukladá do histórie webového prehliadača a ani do logovacieho súboru webového serveru.

- **PUT** - Predstavuje požiadavku na uloženie dát na server. Takto uložené dáta budú dostupné následnými požiadavkami (GET). Uloženie dát do súboru na serveri prevádza priamo server a nie externá aplikácia.

- **DELETE** - Metóda, ktorá slúži na zrušenie dokumentov na serveri. Pomocou *URL* môžeme špecifikovať, ktorý dokument je potrebné zmazať. V skutočnosti je táto metóda podobná príkazu **rm** v operačnom systéme Unix.

- **CONNECT** - Metóda na nadviazanie TCP spojenia (inak označovaného ako tunel) cez HTTP proxy, tzv. vytvorenie virtuálneho pripojenia medzi koncovými uzlami (stanicami).

- **OPTIONS** - Požiadavka umožňujúca zistenie možností komunikácie medzi klientom a serverom.

- **TRACE** - Metóda umožňuje klientovi vidieť, čo prijal server a použiť tieto dáta na testovacie a diagnostické účely.

Stavové kódy

Odpovede serveru takisto rozpoznáme na základe prvého riadku. V prvom riadku odpovede je uvedený stavový kód HTTP. Stavové kódy možu byť:

- **1xx Informational**: Táto trieda kódov označuje predbežnú odpoveď stavu pripojenia alebo stav požiadavky pred jej dokončením a odoslaním finálnej odpovedi napr. 100, 101 a 102.
- **2xx Success**: Trieda týchto kódov znamená, že požiadavka klienta bola úspešne prijatá tzv. server ju pochopil a akceptoval napr. 200, 201 a 202.
- **3xx Redirect**: Skupina kódov zasielané serverom pokiaľ sa na zadanej adrese nenachádza požadovaný obsah napr. 305, 306 a 307.
- **4xx Client Error**: Tieto stavové kódy znamenajú, že v zasielanej požiadavke nastala pravdepodobne chyba, ktorá znemožnila serveru spracovať požiadavku napr. 402, 403 a 404.
- **5xx Server Error**: Stavové kódy identifikujú, že pri spracovaní požiadavky nastala vnútorná chyba serveru. Tieto chyby vznikajú problémom serveru a nie požiadavkou klienta napr. 502, 503 a 504.

2.3 Hlavičky v protokole HTTP

Hlavičky sú súčasťou správ v protokele HTTP, ktoré definujú doplňujúce informácie a parametre transakcie medzi serverom a klientom. Každá hlavička tvorí jeden riadok a skladá sa zo svojho mena, za ktorou nasleduje dvojbodka a hodnoty hlavičky. Hlavička može obsahovať aj viacej hodnôt, medzi ktorými je vždy oddeľovací znak bodkočiarka. Ukončenie aktuálnej hlavičky je zabezpečené novým riadkom (znak CR a LF). Zoznam vsetkých definovaných hlavičiek je uvedený v tabuľke 2.1 a úplná funkcia hlavičiek je popísaná v *RFC 2616* [8] alebo v *RFC 7231* [10]. Hlavičky možeme rozdeliť na:

- *Všeobecne použiteľné hlavičky* (General Header).
- *Hlavičky požiadaviek* (Request Header).
- *Hlavičky odpovedí* (Response Header).
- *Hlavičky tela správ* (Entity Header).

2.3.1 Všeobecne použiteľné hlavičky

Sú to hlavičky, ktoré maju všeobecnú použiteľnosť ako pre požiadavky, tak aj pre odpovede. Patria sem napríklad:

Cache-Control používa sa na špecifikáciu smerníc, ktoré musia byť dodržané medzi serverom a klientom pri používaní cache mechanizmov.

Connection umožnuje odosielatelovi špecifikovať možnosti, ktoré sú žiadúce pre konkrétne pripojenie.

Date predstavuje dátum a čas vytvorenia dokumentu, či už na strane servera alebo klienta.

2.3.2 Hlavičky požiadaviek

Tieto hlavičky zasiela klient na server a umožnujú poslať doplňujúce informácie o požiadavke a o klientovi samotnom. Patrí sem napríklad:

Accept umožnuje určiť typy medií, ktoré sú prijateľné v odpovedi tzv. v akom formáte si želáme požadovaný dokument.

Accept-Charset znamená, že klient uvedie, akú znakovú sadu podporuje. Cieľom tejto hlavičky je ulahčiť serveru, ktorý je schopný reprezentovať viacej rovnakých dokumentov v rôznych znakových sadách o správne zaslanie požadovaného dokumentu.

Accept-Language vymedzuje množinu jazykov, ktoré sú preferované v odpovedi servera.

Accept-Encoding podobná hlavička ako Accept ale obmedzená na kódovanie obsahu, ktorý je prijateľný v odpovedi servera.

Host umožňuje určiť server a port požiadavku. Táto hodnota hlavičky určuje autoritatívny server alebo bránu z originálnej *URL* adresy.

2.3.3 Hlavičky odpovedí

Sú to hlavičky zasielané serverom pre klienta a predávajú ďalšie informácie o odpovedi. Takisto poskytujú informácie o serveri. Patrí sem napríklad:

Location používa sa na presmerovanie klienta k inému zdroju. Hodnota hlavičky bude pozostávať z jedinej adresy *URL* idetifikujúcu server, na ktorý je klient presmerovaný.

Server obsahuje informácie o softvéru pôvodného servera, ktorý spracoval požiadavku.

Retry-After udáva, ako dlho bude služba nedostupná pre klienta, ktorý poslal požiadavku.

2.3.4 Hlavičky tela správ

Tieto hlavičky definujú metainformácie o dokumente prenášaného komunikáciou. Patrí sem napríklad:

Content-Language popisuje jazyk prenašaného dokumentu.

Content-Length určuje veľkosť dokumentu (telo správy).

Content-Type udáva typ prenášaného dokumentu.

Zoznam všetkých preddefinovaných hlavičiek v HTTP/1.1			
General Header	Request Header	Response Header	Entity Header
Cache-Control	Accept	Accept-Ranges	Allow
Connection	Accept-Charset	Age	Content-Encoding
Date	Accept-Encoding	ETag	Content-Language
Pragma	Accept-Language	Location	Content-Length
Trailer	Authorization	Proxy-Authenticate	Content-Location
Transfer-Encoding	Expect	Retry-After	Content-MD5
Upgrade	From	Server	Content-Range
Via	Host	Vary	Content-Type
Warning	If-Match	WWW-Authenticate	Expires
	If-Modified-Since		Last-Modified
	If-None-Match		
	If-Range		
	If-Unmodified-Since		
	Max-Forwards		
	Proxy-Authorization		
	Range		
	Referer		
	TE		
	User-Agent		

Tabulka 2.1: Všetky preddefinované hlavičky v HTTP/1.1.

Kapitola 3

Identifikácia

V tejto kapitole sú vysvetlené základné pojmy spojené s identifikáciou. Vysvetlenie jednotlivých pomoj je pomocov osôb, pričom takáto terminológia môže byť obecnejšia. Následne tieto definície zobecníme na našu problematiku týkajúcu sa identifikácie počítačov. V ďalšej podkapitole je rozobraná identifikácia v HTTP. Je tu rozobraný *User-Agent* reťazec a takisto *Cookies*. Posledná podkapitola obsahuje analýzu dostupného nástroja, ktorý sa touto problematikov zaoberá.

3.1 Identita

Identita [4] je akákoľvek podmnožina atribútov a hodnôt nejakej osoby, ktorá dostatočne identifikuje jednotlivú osobu v množine osôb tzv. v komunite. Atribúty osoby sú vlasnosti, ktoré sú špecifické pre danú osobu a odlišujú osobu od komunity. Identita može byť priradená ako k ľudskej bytosti, tak aj k právnickej osobe alebo počítaču. Samozrejme, že hodnoty atribútov sa možu časom meniť, preto pre jednoznačnú identifikáciu je potrebné uchovávať históriu zmien jednotlivých atribútov. Identifikácia osoby z útočníkovej perspektívy znamená, že útočník môže dostatočne určiť osobu z množiny všetkých skúmaných osôb. V tejto práci sa zameriame na identitu počítača.

3.1.1 Rola

Rolu [4] možno definovať ako očakávané správanie osoby v nejakej sociálnej skupine. Tým pádom jednotlivé atribúty a hodnoty danej osoby sú závislé na skupine, v ktorej sa daná osoba nachádza.

3.1.2 Čiastočná identita

Identita jednotlivej osoby môže obsahovať veľa čiastkových identít, z ktorej každá identita reprezentuje osobu v určitej role. *Čiastočná identita* [4] je podmožina kompletnej identity, kde kompletná identita je zjednotenie všetkých atribútov a hodnôt danej osoby.

3.1.3 Digitálna identita

Digitálna identita [4] znamená priradenie atribútov a hodnôt pre jednotlivé osoby, ktoré sú okamžite dostupné v technických zariadeniach. Napríklad digitálna identita može byť jednoduchá e-mailová adresa. Digitálna identita by mala označovať všetky dáta užívateľa, ktoré možu byť uložené a automaticky prepojené s počítačovou aplikáciou.

3.2 Identifikáciia v HTTP

Identitu v HTTP možeme určovať pomocou hlavičiek, ktoré nam udavajú informácie napr. o použitom webovom prehliadaci a jazykovu podporu. Na základe takto získaných informácii možeme vytvárať čiastočné identity.

Samotná identifikácia v HTTP je ale obiažna pretože pri komunikácii sa možu hodnoty jednotlivých hlavičiek meniť tom istom sieťovom spojení. Niektoré hlavičky sú závislé na typu média, čo može znemožniť presnú identifikáciu webového prehlidača.

Pri idetifikácii v HTTP sa zameriam na dve hlavné HTTP hlavičky *User-Agent String* a *Cookies*, ktoré sú vysvetlené nižšie a budú následne doplnené o HTTP hlavičky požiadaky popisané v kapitole 2.3.

3.2.1 User-Agent String

Každý klientsky softvér odosiela vo svojich požiadavkách tzv. *User-Agent String*, inak povedané svoju identifikáciu. Identifikáciou je myslený krátky text zodpovedajúci webovému prehliadaču, verzii a dodatočným informáciam. Za klienstky softvér považujeme vo väčšine prípadov *webový prehliadač*. V HTTP sa takýto identifikátor posiela hlavičkou *User-Agent*. V tejto sekcii som čerpal s knihy [16].

Z historického hľadiska ako prvý vznikol identifikátor webového prehlidača *Mosaic*, ktorý mal jednoduchý identifikátor `Mosaic/0.9`. Text pred lomítkom je názov a text za lomítkom udáva verziu webového prehliadača. Ďalším krokom vývoja bolo predstavenie prehliadača *Netscape Navigator 2* s kódovým označením *Mozilla* (skratka slov Mosaic Killer). Prvotná verzia identifikátora tohto prehliadača mala tvar `Mozilla/Verzia [Jazyk] (Platforma; Šifrovanie)`. Jazyk bol určený kódom, ktorý určuje jazykové nastavenie klienta. Platforma značila operačný systém, na ktorej bežala aplikácia a šifrovanie udávalo typ zabezbečenia. Pri vydaní ďalšej verzie prehliadača *Netscape Navigator 3* sa tomuto prehliadaču dostalo veľkému obdivu a okamžite sa stal najobľúbenejší webový prehliadač. Identifikátor tohto prehliadača sa zmenil a to tak, že bol odobraný jazyk a pridaný voliteľný popis operačného systému (OS) alebo precesoru (CPU). Príklad takéhoto identifikátora mal tvar `Mozilla/Version (Platforma; Šifrovanie [; OS-alebo-CPU popis])`.

Pri tvorbe nasledujúcich prehliadačov sa ich tvorcovia stretli s problémom, že pokiaľ chceli mať webové stánky takiež zobrazené s ich úplnou funkčnosťou museli identifikátor *Netscape* napodobniť a tým vo väčšine prípadov zachovať aj kódové označenie *Mozilla*.

Chrome

WebKit je kódové označenie prehliadača *Safari*, ktorý bol vyvinutý firmou Apple. Webový prehliadač využíva na rendrovanie *WebKit*, ale s rozdielnym *JavaScript* jadrom. Formát identifikátora začiatku prehliadača bol napr. u verzie Chrome 0.2 `Mozilla/5.0 (Platform; Encryption; OS-or-CPU; Language) AppleWebKit/AppleWebKitVersion (KHTML, like Gecko) Chrome/ChromeVersion Safari/SafariVersion`.

Príklady chrome identifikátorov :

- *Chrome 41.0.2228.0*: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36
- *Chrome 41.0.2227.1*: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2227.1 Safari/537.36

- *Chrome 41.0.2227.0*: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2227.0 Safari/537.36
- *Chrome 0.2.149.27*: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/525.13 (KHTML, like Gecko) Chrome/0.2.149.27 Safari/525.13

Opera

Opera je jeden z mála prehliadačov, ktorý svoj identifikátor postavili na vlastnom názve a nepokúšal sa napodobniť ostatné identifikátory. Toto riešenie je najlogickejšie a vychádza aj s *RFC 7231*[10]. Terajší identifikátor vychádza z pôvodného identifikátora, ktorý mal tvar `Opera/Version (OS-or-CPU; Encryption; Language)`.

Príklady opera identifikátorov (Hlavná verzia je 12 a čísla udávaju aktualizáciu) :

- *Opera 16*:Opera/9.80 (X11; Linux i686; Ubuntu/14.10) Presto/2.12.388 Version/12.16
- *Opera 14*:Opera/9.80 (Windows NT 6.0) Presto/2.12.388 Version/12.14 Safari/537.36
- *Opera 02*:Opera/12.80 (Windows NT 5.1; U; en) Presto/2.10.289 Version/12.02

Internet Explorer

Prehliadač, ktorý ako jeden z veľa prehliadačov bol nútený zachovať kódové označenie *Mozilla* a vytvoriť podobný identifikátor. Jeho identifikátor mal tvar `Mozilla/4.0(compatible; MSIE verzia, Operačný systém)`. Prvotné zmeny prišli až s verziou *Internet Explorer 8*, kde bol mierne upravený *User-Agent String*.

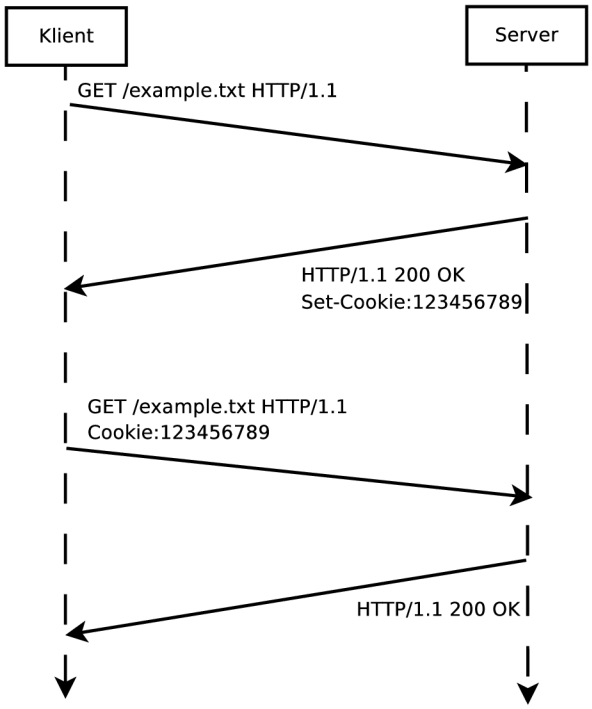
Príklady Internet Explorer identifikátorov :

- *Internet Explorer 11.0*: Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko
- *Internet Explorer 10.0*: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
- *Internet Explorer 9.0*: Mozilla/5.0 (Windows; U; MSIE 9.0; WIndows NT 9.0; en-US))

3.2.2 Cookies

Cookies [3] sú textové reťazce, ktoré môžu vznikať v priebehu komunikácie medzi klientom a serverom. Ak užívateľ prvýkrát pristúpi na stránku a má povolené cookies, tak si server vygeneruje textový reťazec, ktorý následne uloží na strane klienta. Pri ďalšej návšteve danej stránky, bude klient automaticky zasielať dané cookies v požiadavke na server. Tento mechanizmus je možné vidieť na obrázku 3.1. Cookies môžeme rozdeliť do kategórií trvalé/dočasné alebo vlastné/cudzie cookies.

- *Trvalé cookies*(Persistent Cookies) zostavajú uložené v počítači klienta, kým nevyprší ich platnosť alebo ich užívateľ manuálne nevymaže.



Obrázek 3.1: Mechanizmus cookie

- *Dočasné cookies*(Session Cookies) sú vytvorené pri začatí spojenia s webovou lokalitov ale po ukončení sú vymazané. Po opetovnom navštívení danej stránky sa tieto cookies znova vytvoria.
- *Vlastné cookies*(First Party Cookies) takéto súbory zanecháva stránka, ktorá bola navštívená.
- *Cudzie cookies*(Third Party Cookies) sú vytvárané inými stránkami, nie tou kde sa užívateľ nachádza.

V tejto práci sa zameriame na trvalé a vlastné cookies, pretože nám najlepšie umožnujú idetifikáciu klienta. Databáza cookies aj s čiastočným vysvetlením je dostupná na adrese `http://cookiepedia.co.uk/`.

3.2.3 Existujúce prístupy k identifikácií

Dostupná služba, ktorá sa zaoberá identifikáciou je služba *panopticlick*. Je dostupná s `https://panopticlick.eff.org/`. Jej prístupy a štúdie sú spomenuté v článku [2]. Pre idenfifikáciu využivá nasledujuúce identifikátory.

- **User-Agent**, získavaný s prenosu HTTP.
- **HTTP Accept hlavičky** taktiež získavané s prenosu HTTP.
- **Povolenie cookies**, získavaný s prenosu HTTP.
- **Rozlíšenie obrazovky** získané pomocou technológií JavaScript [1] a AJAX [2].

[1] JavaScript, je programovací skriptovací jazyk, používaný najmä pri tvorbe webových stránok.
[2] AJAX je označenie pre technológie vývoja interaktivých webových aplikácií.

- **Časová zóna** získané pomocou technológií JavaScript a AJAX.
- **Doplnky prehliadača, verzia doplnku a MIME** [3] **typy** získané pomocou technológií JavaScript a AJAX.
- **Systémové písma**, získavané pomocou Java alebo Flash appletov [4].
- **Test supercookie** vykonaný pomocou technológií JavaScript a AJAX.

S týchto ôsmych identifikátorov vytvára databázu rôznych identít. V článku z roku 2010 [1] vychádza, že *User-Agent* zvyčajne pozostáva s 5 až 15 bitov informácií, v priemere je to 10,5 bitov. Na základe tejto informácie možeme vypočítať aká je rozlišiteľnosť uživateľa pomocou *User-Agent* hlavičky, viď rovnicu 3.1.

$$P(x) = 2^{bits} \tag{3.1}$$

Hodnota *bits* udáva, koľko bitov informácie obsahuje webový prehliadač pre identifikáciu. Vyýsledkom je hodnota udavajúca veľkosť skupiny, v ktorej bude identifikovateľný. Napríklad ak chceme rozlišiť uživateľa na základe *User-Agent* hlaičky a po dosadení hodnoty 10,5 bitov nám vychádza približne hodnota 1500. To znamená že v priemere iba jedna osoba zo skupiny 1500 osôb bude mať takúto *User-Agent* hlavičku. To síce nestačí pre jednoznačnú identifikáciu ale po pridaní ostatných identifikátorov sa nám hodnota *bits* zväčší a to na hodotu 22,36 bitov. V takomto prípade to vychádza že iba jedna osoba zo skupiny 5 383 078 osôb bude mat takúto konfiguráciu sôjho webového prehliadača.

[3] MIME (Multi-Purpose Internet Mail Extensions), je internetový štandard, ktorý rozširuje základný formát emailu.

[4] Applet je jednoduchá aplikácia, ktorá sa spúšta z iného programu napr. weboého prehliadača.

Kapitola 4

Dynamická identita v projekte Sec6Net

Systém pre zákonné odpočúvanie *(Sec6Net Lawful Interception System - SLIS) [5]* bol vytvorený na Vysokém učení technickém v Brně pre potreby výzkumu v oblasti zákonných odpočúvaní, analýzy dát, ich rekonštrukcie a hardwarovej akcelerácií sond určených pre zber a identifikáciu dát. SLIS dokáže spolupracovať so sondami vytvorenými v rámci projektu *Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace* (Sec6Net).

Okrem nástroja SLIS vznikla v rámci činnosti aj rada ďalších nástrojov, ktoré boli použité v samotnom systéme. Jedným z výsledkov je systém pre dynamicku správu identity *(Sec6Net Identity Management System - SIMS) [5]*. Tento nástroj vychádza z implementácie bloku IRI-IIF v rámci SLIS a zpístupňuje graf identifikátorou.

V tejto práci sa zameriam na nástroj SIMS, pretože je určený pre obecné využitie a nieje obmedzený pre oblasť zákonných odpočúvaní.

4.1 Činnosť bloku IRI-IIF

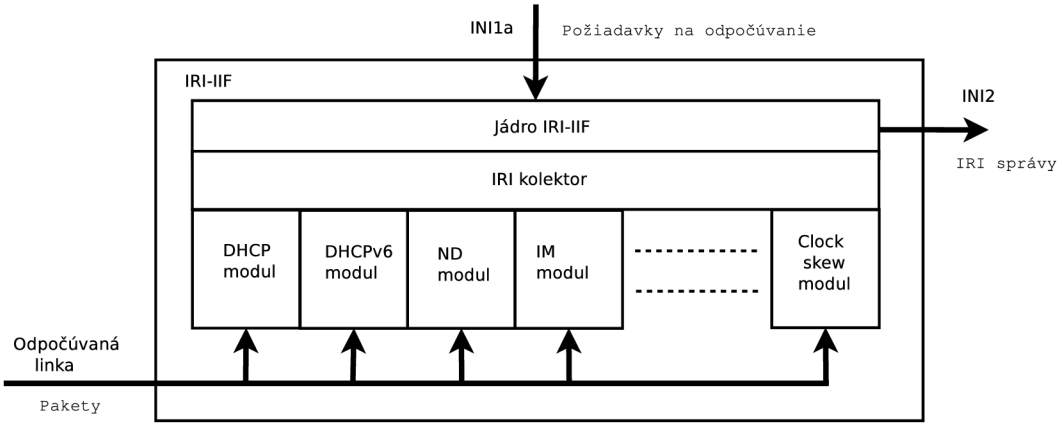
Identita sledovaného užívateľa sa na strane poskytovateľa može dynamicky meniť napríklad pomocou protokolov DHCP, RADIUS, ND apod. Túto zmenu a identitu sleduje blok IRI-IIF, ktorý informuje o zmenách v identifikátoroch tak, aby mohli byť včas prekonfigurované napojené sondy. Architektúra bloku IRI-IIF navrhnutého v rámci projektu Sec6Net je znázornená na obrázku 4.1.

Popis bloku IRI-IIF:

1. Vstup INI1a združuje požiadavky na odposluch. Požiadavka je definovaná jednoznačným identifikátorom odpočúvania (Lawful Interception Identifier - LIID) a jednoznačným idetifikátorom odpočúvaného užívateľa (Network Identifier - NID).

2. Vstup tvoria informácie o zmene identity užívateľa, ktoré možu byť vo forme sieťovej komunikácie (DHCP, RADIUS, DHCPv6).

3. Výstup bloku tvoria takzvané *IRI správy* informujúce o identite odpočúvaného užívateľa.



Obrázek 4.1: Architektúra bloku IRI-IIF.

Blok IRI-IIF je navrhnutý modulárne. Skladá sa s *jadra IRI-IIF* a jednotlivých *modulov*. Moduly slúžia na analýzu protokolov, pre ktoré sú navrhnuté. Medzi samotným jadrom a modulami existuje medzivrstva, tzv. *IRI kolektor*, ktorého úlohou je prijímať správy od jednotlivých modulov a preposielať ich jadru IRI-IIF.

Modul analyzuje a spracováva informácie o zmene identity užívateľov v rámci sledovanej sieti. Získané informácie predáva jadru IRI-IIF, ktoré je schopné na ich základe generovať výstupné IRI správy a spravovať graf NIDov.

Jadro IRI-IIF prijíma vstupné požiadavky na odpočúvanie a udržuje si tabuľku aktuálne prebiehajúcich odpočúvaní. Ďalej príjma správy od modulov, filtruje ich na základe tabuľky aktuálnych odpočúvaní a generuje výstupné IRI správy. Jadro ďalej je zodpovedné aj za prepojenie informácií z modulov, ktoré sa týkajú odpočuvaného užívateľa a udržuje si zoznam identít všetkých užívateľov v sieti.

Tvorba grafu NIDov sa riadi nasledujúcimi pravidlami:

1. Pri prijatí správy BEGIN alebo CONTINUE je do grafu vložená nová hrana medzi vrcholmi popisanými v správe. Pokiaľ tieto vrcholy v grafe zatiaľ neexistujú, pridajú sa do grafu.

2. Pri prijatí správy END je z grafu odstránená hrana medzi vrcholmi popísanými v správe.

3. Z grafu sú automaticky odstraňované vrcholy, medzi ktorými neexistuje žiadna hrana.

4.2 Rozhranie medzi modulmi a jadrom IRI-IIF

Toto rozhranie je navrhnuté tak, aby malo jádro s prípadnou úpravov predavaných informácií čo najmenšiu prácu. Jednotlivé moduly zasielajú informácie o pokusu alebo pridelení identity vo forme podobnej výstupným IRI správam, jádro potom vykonáva len ich filtráciu a prípadnú úpravu. Zasielanie správ jednotlivými modulmi je možné vidieť na obrázku 4.2. Typy správ a ich formát je uvedený v tabuľke 4.1.

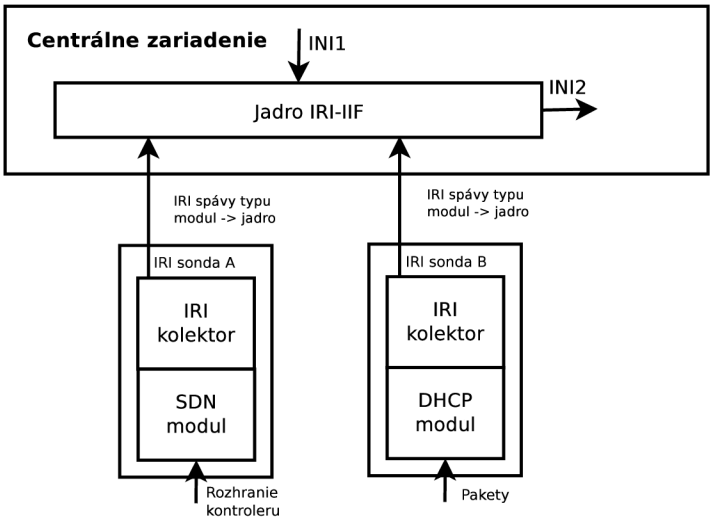
U spáv typu BEGIN, CONTINUE a END sa zvyčajne jedná o označenie začiatku, predĺženie alebo ukončenie obdobia, kedy bola priradená IP adresa alebo prebiehalo dané TCP spojenie. V správach od jednotlivých modulov je potrebné zasielať informácie nie len

o pridelení IP adries a prebiehajúcich TCP spojeniach ale aj informácie o autentifikácii. Na to slúži sieťový identifikátor *Network Identifier* (NID). Tento identifikátor slúži k označeniu konkrétneho spojenia. Na úrovni protokolov pre autentifikáciu podporuje napríklad prihlasovacie mená protokolov PPP a RADIUS.

Formát NIDov

(Názov NIDu, Hodnota NIDu)

Napríklad:
`('MAC', '00:0c:29:11:7c:14')` alebo `('PPPLogin', 'user')`



Obrázek 4.2: Modulárna architektúra IRI-IIF.

Typ spávy

- *BEGIN* - najdená nová väzba medzi identifikátormi.
- *COUNTINUE* - väzba medzi identifikátormi stále trvá.
- *END* - väzba medzi identifikátormi nieje ďalej platná.
- *REPORT* - pomocná informácia nemeniaca predtým zaslané väzby.

Formát správy

(Meno modulu, Časová značka, Typ správy: BEGIN/CONTINUE/END/REPORT, Popis spávy, Zoznam NIDu, ktorých väzieb sa správa týka)

Tabulka 4.1: Formát a typy správ medzi modulmi a jadrom IRI-IIF

Napríklad:
`('pppoe', 1302818400.0, 'BEGIN', 'Client made connection with BRAS', [('PPP SESSION', '1234'), ('MAC', '00:0c:29:11:7c:14'), ('PPPLogin', 'user'), ('IP', '192.168.0.1')])`

V zozname NIDov môžu voliteľne následovať ďalšie 2 zoznami. V takomto prípade je rozdelenie nasledujúce:

1. *zoznam*: NIDy určené pre umiestnenie do grafu spravovaného jádrom IRI-IIF. Tieto NIDy sú pri odoslaní správy END z grafu jádra IRI-IIF odstranené.

2. *zoznam*: NIDy, ktoré niesú určené pre umiestnenie do grafu jádra IRI-IIF, ale iba do prípadných správ IRI.

3. *zoznam*: NIDy určené pre umiestnenie do grafu spracovavaného jádrom IRI-IIF, ktorých väzba by mala byť uchovávaná aj po odoslaní správy END a nemala by byť z grafu jádra IRI-IIF vymazaná.

Kapitola 5

Návrh identifikácie pomocou HTTP

V tejto kapitole je popísaný návrh aplikácie pre identifikáciu pomocou HTTP požiadavkov zo sieťovej prevádzky.

Aplikácia je navrhnutá ako samostatný modul, ktorý možeme následne zaradiť do projektu Sec6Net a to konkrétne do nástroja SIMS.

Fungovanie celej aplikácie spočíva zo štyroch hlavných častí, viď obrázok 5.1:

1. *Zachytávanie paketov*
2. *Spracovávanie paketov*
3. *Tvorba tabuľky identít*
4. *Generovanie výstupných správ*

5.1 Výber identifikátorov sieťového spojenia

Pri reálnom behu aplikácie sa bude spracovávať veľké množstvo paketov, preto je potrebné určiť, ktoré pakety požadujeme a ktoré sú pre naše potreby zanedbateľné. Výber paketov sa deje na základe vhodne zvolených identifikátorov, čim docielime zmenšenie dát na spracovanie. Seťové spojenie rozlišujeme na základe týchto identifikátorov:

- *Zdrojová IP adresa*, ktorá identifikuje zdroj HTTP požiadavky. Zdroj požiadavky nazývame *klient* a určuje nám počítač, z ktorého bola požiadavka zaslaná.
- *Cieľová IP adresa*, ktorá na rozdiel od zdrojovej IP adrese určuje cieľ HTTP požiadavky tzv. *server*, na ktorý je požiadavka zaslaná.
- *Zdrojový port* slúži pre rozlišenie sieťových spojení pri súčasnej viacnásobnej komunikácii so serverom.
- *Cieľový port* je vybratý na určenie služby klienta komunikujúceho so serverom. V našom prípade služba HTTP využíva port 80, ale pre väčšiu flexibilitu môžeme využívať rôzne porty.
- *Protokol* bude stále TCP, ale kvôli rozlíšeniu UDP a TCP komunikácie je potrebné ho zohľadniť vo filtrácii paketov.

5.2 Popis činnosti

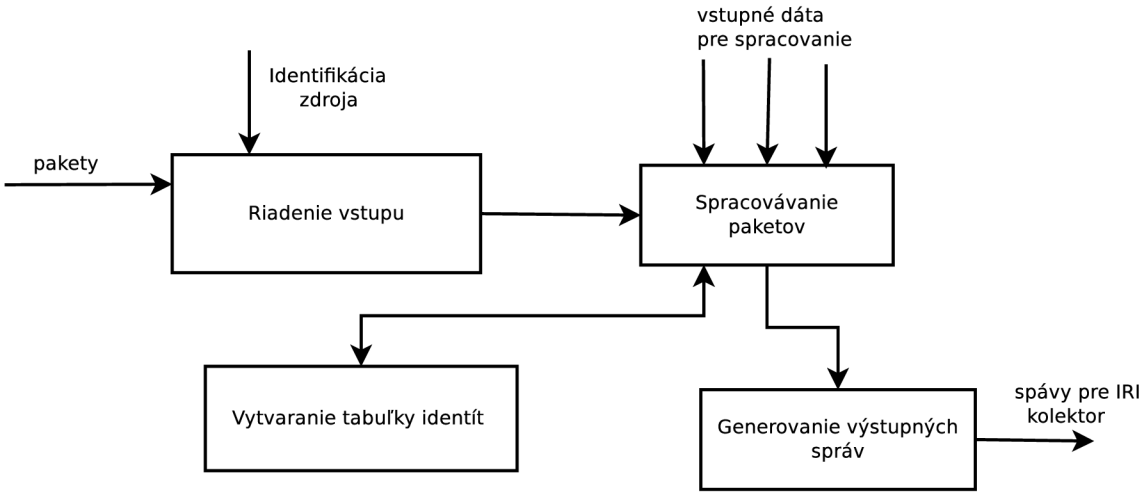
Modul na identifikáciu sa skladá zo štyroch blokov. Spracovanie dát je sekvenčné, pretože modul musí byť schopný generovať výstupné správy v reálnom čase.

Prvý blok riadi vstupnú sieťovú komunikáciu. Jeho úlohou je identifikácia zdroja tzv. či sa jedná o reálnu komunikáciu alebo o zachytenú komunikáciu napr. programom *Wireshark*. Taktiež modul vykoná otvorenie zachytenej komunikácie alebo zapezpečenie otvorenia rozhrania na odpočúvanie sieťovej komunikácie. Taktiež vykonáva kontrolu, či daný paket je paket HTTP komunikácie, a pokiaľ áno tak tento paket preposiela nasledujúcemu bloku.

Druhý blok vykonáva spracovanie paketu na základe vybratých identifikátorov, ktoré presne špecifikujú sieťové spojenie. Takéto sieťové spojenie nazývame čiastočná identita, pretože môžeme na základe identifikátorov presne odlíšiť požadované spojenie od celej množiny sieťových spojení. Následne blok získavá požadované hlavičky s daného paketu.

Tretí blok vytvára tabuľku všetkých zachytených identít, zachytené identity blok obdrží od bloku *Spracovávanie paketov*. Následne pre každú identitu uchováva záznamy o HTTP hlavičkách. Vytváranie tabuľky identít je popísané v kapitole 5.3.

Štvrtý blok generuje správy pre nástroj SIMS. Správy sú posielané na *IRI kolektor jadra IRI-IIF*, ktoré je popísané v kapitole 4.



Obrázek 5.1: Architektúra modulu HTTP.

5.3 Tvorba tabuľky

Pri obdržaní správy od bloku Spracovávanie paketov sa hľadá zodpovedajúca identita v tabuľke identít 5.1. Ak v tabuľke existuje záznam s takouto identitou, zistí sa ukazateľ na ďalšiu tabuľku obsahujúcu záznamy hlavičiek danej identity. Ak záznam neexistuje, vytvorí sa nový záznam v tabuľke identít s odpovedajúcou identitou a takiež sa vytvorí tabuľka pre záznamy hlavičiek napr. 5.2 a 5.3. Do tejto tabuľky sa ukladajú všetky potrebné HTTP cookies, HTTP hlavičky a URI požadovanej stránky. Všetky riadky tabuľky sú unikátne.

IP adresa		Port		
Zdrojová	Cieľová	Zdrojový	Cieľový	Ukazovateľ na záznamy
192.168.0.1	147.220.145.200	1450	80	záznam 1
192.168.0.1	108.22.11.45	2300	80	záznam 2
10.10.10.125	147.220.145.200	5684	443	záznam 3
10.10.10.125	147.220.145.200	8784	80	záznam 4
125.45.145.25	45.45.125.154	2354	149	záznam 5

Tabulka 5.1: Tabuľka identít.

Záznam 1		
číslo	hlavička (header=)	hodnota hlavičky
1	User-Agent	Googlebot/2.1 (+http://www.googlebot.com/bot.html)
2	Accept-Language	en-us
3	Accept-Encoding	gzip

Tabulka 5.2: Tabuľka záznamov hlavičiek (pre záznam 1).

Záznam 4		
číslo	hlavička (header=)	hodnota hlavičky
1	User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36
2	Accept-Language	cs-CZ,cs;q=0.8
3	Accept-Encoding	gzip,deflate,sdch

Tabulka 5.3: Tabuľka záznamov hlavičiek (pre záznam 4).

Kapitola 6

Implementácia nástroja

V tejto kapitole je popísaná implementácia nástroja. Princíp aplikácie je vysvetlený pomocou vyvojových diagramov. Prvá podkapitola je venovaná inicializácií aplikácie. V nasledujúcej kapitole je rozobraná implementácia jednotlivých modulov. Sú tu spomenuté jednotlivé implementačné problémy a následne ich riešenie. Takiež je popísaná implementácia rozhrania pre komunikáciu aplikácie s nástrojom *SIMS*. Celá aplikácia je implementovaná v jazyku *Python*.

6.1 Inicializácia aplikácie a spracovanie vstupných dát

Pre inicializáciu aplikácie sa využívajú dáta dostupné v inicializačnom súbore `start.ini`. Tento súbor je parsovaný modulom *ConfigParser*[1]. Z daného súboru sa načítavá zoznam sledovaných HTTP hlavičiek, zoznam odpočúvaných portov, názvy sledovaných cookies a ich platnosť a taktiež je možné určiť podreťazec sledovaného reťazca hlavičky cookies. Výber vstupných dát je nutné špecifikovať ako parameter spúštanej aplikácie. Za vstupné dáta považujeme pakety. Dáta možu byť uložené v súbore alebo môžeme zadať názov rozhrania, z ktorého sa budú dáta získavať. Otvorenie súboru alebo otvorenie zadaného rozhrania je implementované pomocou python modulu *Scapy* [2]. O samotné otvorienie zdroja sa stará modul *Riadenie vstupu*. Pre zabezpečenie správneho ukončenia aplikácie sú v inicializačnej fáze nastavené funkcie pre obsluhu signálov *SIGINT*, *SIGTERM* a *SIGQUIT*.

6.2 Implementácia rozhrania na komunikáciu

Pri implementácii rozhrania je potrebné špecifikovať formát spávy, ktorá sa bude posielať aplikácií SIMS. Očakávaný formát pre aplikáciu SIMS je špecifikovaný v kapitole 4.2. Pre potreby napojenia HTTP modulu s aplikáciou bolo potrebné implementovať NIDy, ktoré budú sprostredkovávať informácie z tohto modulu.

Pridané NIDy a ich formát:

- **HTTP hlavičky** - *('HTTP Headers',{'Názov_Hlavičky1': 'Hodnota_Hlavičky1', 'Názov_Hlavičky2': 'Hodnota_Hlavičky2', ...})*

[1] Modul dostupný v Pythone 2.7 na parsovanie inicializačných súborov
[2] Scapy je interaktívny modul na prácu so širokou škálou sieťových protokolov

- **HTTP cookies** - *('HTTP Cookies',{'Názov_Cookies1': 'Hodnota_Cookies1', 'Názov_Cookies2': 'Hodnota_Cookies2', ...})*
- **URI** - *('URI','Hodnota_URI')*

Pre zasielanie správ je použité zasielanie NIDov pomocou 3 zoznamov, pretože URI nieje identifikátor potrebný pre vytváranie grafu jadra IRI-IIF. Adresa URI je zasielaná ako druhý zoznam čim je dosiahnuté aby sa tento NID nezohľadnoval v grafe jadra ale iba do prípadných IRI správ. Príklad správy od HTTP modulu bude mať nasledovný tvar:

('http', 1302818400.0, 'BEGIN', 'Začiatok spojenia', [('HTTP Headers, {'Accept': '*', 'Accept-Encoding': 'gzip'}), ('HTTP Cookies', {'__utma': '4598754.45454', 'xs': 'AsKJ.7895'})], [('URI', 'http://example.com/picture.jpg')],[])

6.3 Implementácia jednotlivých modulov

Pri implementácii jednotlivých moulov sa vychádzalo s návrhu v kapitole 5. Celková funkcionalita sa zachoval ako bolo popísane v návrhu ale jednotlivé konštrukcie v blokoch boli pozmenené. Največšou zmenou je tvorba tabuľky identít a jej formát.

6.3.1 Riadenie vstupu

Tento modul zabezpečuje správne otvorenie požadovaného zdroja. Pri úspešom otvorení zdroja modul začne získavať pakety. Keďže pri reálnej prevádzke modul odchytí veľke množstvo paketov je potrebné určiť či nás tento paket zaujíma. Takáto jednoduchá filtrácia využíva dáta získané s inicializačnej fázy a to konkrétne zoznam sledovaných portov.

Prednastavená hodnota porovnavaného portu je *80*, pretože na tomto porte prebieha komunikácia HTTP. Každý paket ktorý prejde takouto jednoduchou filtráciou je preposlaný modulu *Spracovanie paketov*.

Vývojový diagram pre riadenie vstupu je na obrázku 6.1.

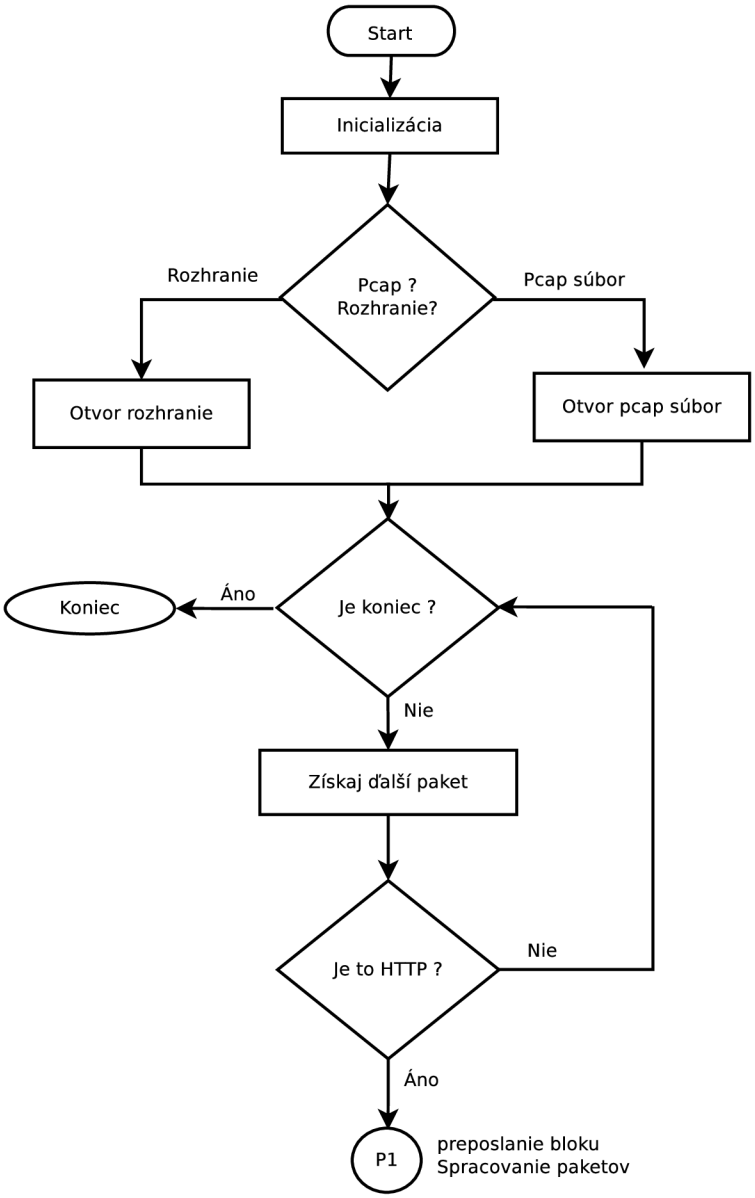
6.3.2 Spracovanie paketov

Pri obdržaní paketu od modulu *Riadenie vstupu* sa ako prvé vyhodnocuje či poslaný paket obsahuje príznak *FIN* [3]. Pokiaľ sa takýto príznak nachádza v pakete získavajú sa položky pre srávu END a následne sa posiela vnútorná správa END a dané spojenie sa vymaže s tabuľky identít. Všetky vnútorné správy typu BEGIN/COUNTINUE/END sa zasielajú modulu *Generovanie výstupných správ*.

Ak sa príznak *FIN* nenachádza v pakete kontorluje sa či sa jedná o paket HTTP požiadavky. Táto kontrola prebieha kontrolovaním prvého riadku dát paketu, kde musí byť použité klúčové slovo použitej metódy. Zoznam podporovaných metód je v kapitole 2.2. Pri úspešnej kontrole sa takéto sieťové spojenia prehľadáva v tabuľke identít, kontrolujú sa identifikátory uvedené v návrhu, viď kapitola 5.1. Pri kontrole môžu nastať dva prípady.

- *Spojenie sa nachádza v tabuľke identít*: V takomto prípade sa aktualizujú položky v tabuľke identít a generuje sa požadovaná správa. Problém by nastal, ak by paket

[3] Príznak posielaný v HTTP komunikácii pri ukončení spojenia



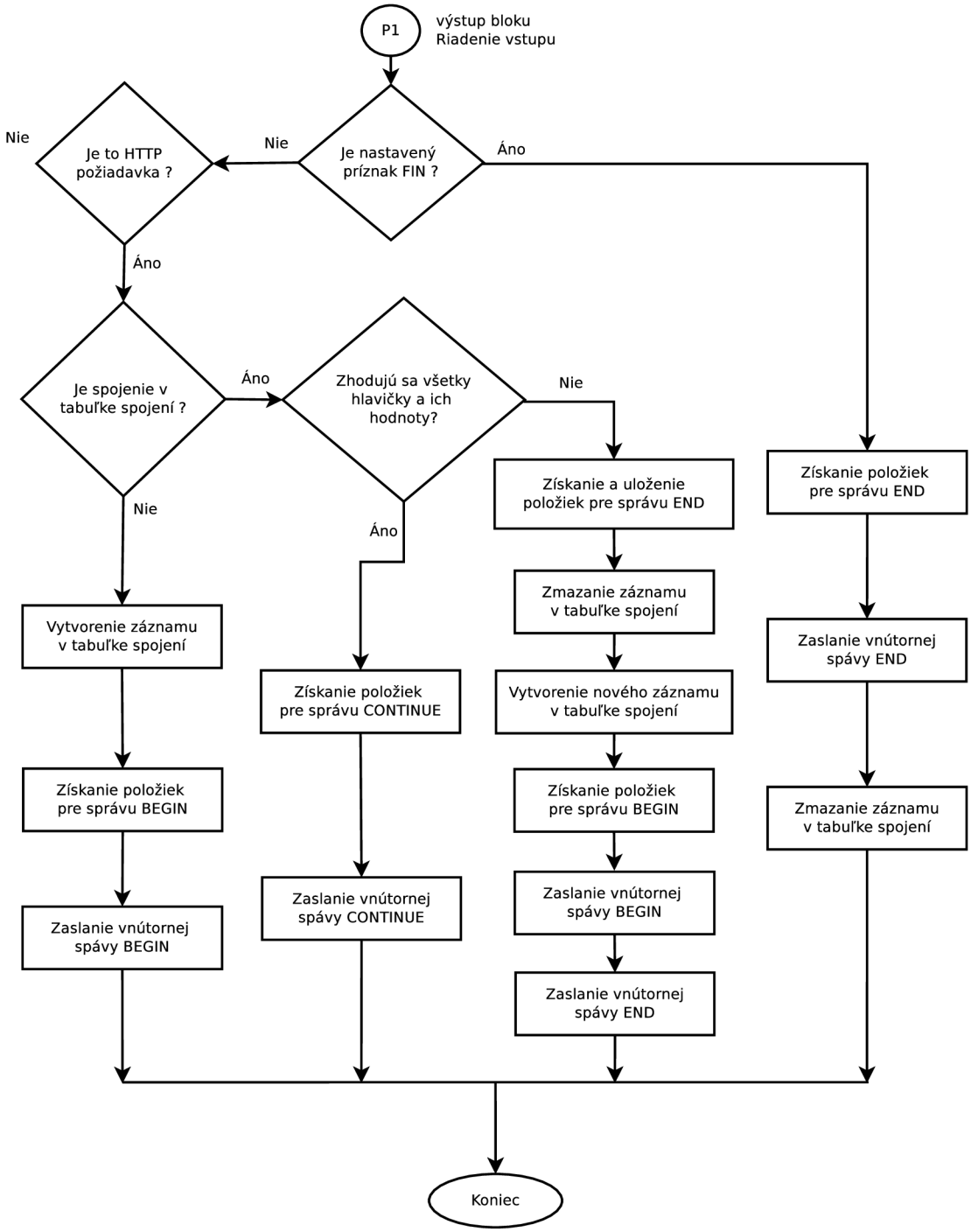
Obrázek 6.1: Bloková schéma riadenia vstupu.

obsahoval pozmenené dáta. Vzinkalo by to že pri spájani s nástrojom SIMS by tento nástroj nebol uvedomený o takejto zmene. Aby sa predišlo tomuto problému je potrbné skontrolavať položky *Slovník HTTP hlavičiek a Slovník HTTP cookies* v tabuľke identít. Pokiaľ sa tieto položky budú zhodovať s dátami aktuálneho paketu zasiela sa vnútorná správa CONTINUE. Ale ak sa tieto položky v tabuľke identít nerovnajú je potrebné najprv zaslať správu BEGIN s novými položkami a následne zaslať správu END s položkami uloženými v tabuľke a až následne aktualizovať tabuľku.

- *Spojenie sa nenachádza v tabuľke identít*: Záznam o takomto sieťovom spojení bude vložený do tabuľky identít a generuje sa správa. *BEGIN*.

Celá funkcionalita tohto bloku je znázornena na vývojovom diagrame 6.2. Ako je možné vidieť daný modul komunikuje s *tabuľkov identít* pre vytváranie alebo vymazávanie zá-

znamu, a taktiež pre získavanie položiek pre konkrétne vnútorné správy. Konštrukcia tabuľky identít a jej popis je vysvetlený v nasledujúcej sekcii.



Obrázek 6.2: Bloková schéma spracovania paketov.

6.3.3 Vytvaranie tabuľky identít

Pri implementácii tabuľky sa vychádzalo z návrhu aplikácie ale bolo potrebné niektoré položky pridať a niektorým zmeniť formát uloženia.

Tabuľka obsahuje jednotlivé položky tak, ako je to implementované v aplikácii:

1. *Zdrojová IP adresa* uchováva sa textový reťazec zadanej IP adresy.
2. *Cieľová IP adresa* uchováva sa textový reťazec zadanej IP adresy.
3. *Zdrojový port* uchováva sa číslo zadaného portu.
4. *Cieľový port* uchováva sa číslo zadaného portu.
5. *Stav* určuje, či dané spojenie sa práve vytvorilo alebo už existuje v tabuľke identít.
6. *Čas* zadaný v sekundách od 1.1.1970.
7. *Slovník HTTP hlavičiek* obsahujúci názov hlavičky a jeho hodnotu. Príklad takéhoto záznamu je {`'Accept-Encoding': 'gzip', 'Accept-Language': 'cs,sk;q=0.8'`}
8. *Slovník HTTP cookies* obsahujúci názov cookies a jeho hodnotu. Príklad takéhoto záznamu je {`'__utma': '787878787.99999.45454545', 'datr': 'Ak89lakskskS'`}
9. *URI stránky*

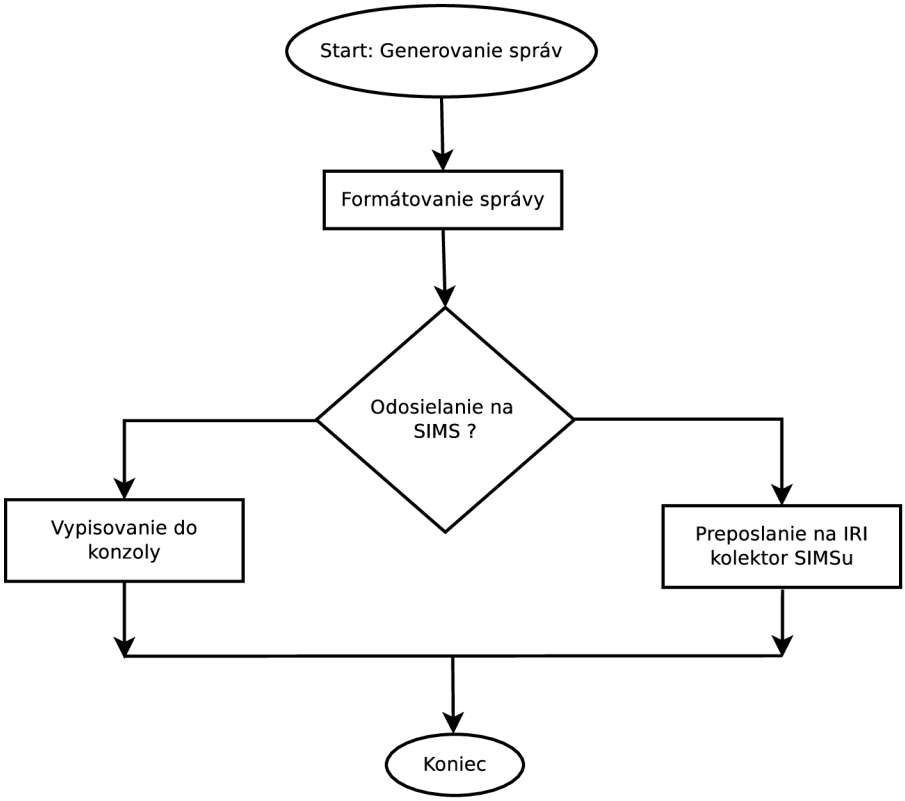
Pri hľadaní identity v tabuľke sa kontrolujú prvé štyri položky. Pri neúspešnom hľadaní sa v hľadanom spojení prehodia zdrojové a cieľové položky a znova sa začne s hľadaním. Takáto výmena je potrebná, ak by náhodou ukočenie spojenia prišlo od servera, kde sú tieto položky vymenené. Pokiaľ sa nenašla zhoda, vytvorí sa nový záznam a jednotlive políčka sa naplnia požadovanými položkami.

6.3.4 Generovanie výstupných správ

Funkcia bloku spočíva v naformatovaní správy a následnom odoslaní aplikácií SIMS alebo vypísanie do konzoly na štandartný výstup. Formátovanie jednotlivých spáv je vďaka formátu uloženenému v tabuľke veľmi jednoduché. Formát očakavaných správ je definovaný v kapitole 6.2. Voľba výstupu je získaná s príkazového riadku.

Odosielanie správ nástroju SIMS je implementované cez *unixové sockety*[4].

[4] *Unixové sockety* služia na predávanie dát medzi rôznymi procesmi v rámci jedného počítača



Obrázek 6.3: Bloková schéma generovania výstupných správ.

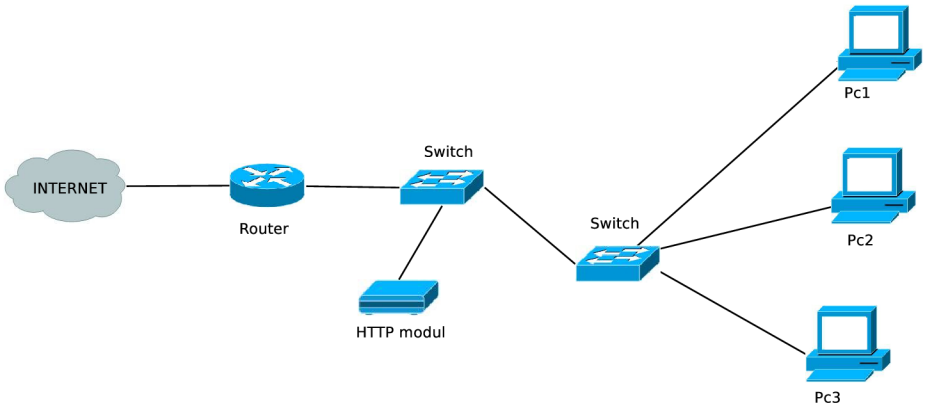
Kapitola 7

Experimenty

V tejto kapitole sa zameriame na funkčnost a použiteľnosť vyvynutého nástroja. Experimenty sú rozdelené na dve časti. V prvej časti je zhodnotená funkčnosť násroja a napojenie na SIMS. Druhá časť obsahuje experimenty identifikácie a experiment zaoberajúci sa identifikátorom cookies. Všetky testovacie súbory a prípadne ich výstupy testovania sú na priloženom CD. Testovanie prebehlo na počítači s operačným systémov `Ubuntu 14.01 LTS 64bit`, pamäťou `7,7 GiB` a procesorom `Intel Core i7 1,73 GHz x 8`.

7.1 Testovanie funkčnosti aplikácie

Pri testovaní navrhnutej apliácie sa zameriame na spávnosť výstupu aplikácie, počet spracovaných paketov a následne správnost zasielania správ aplikácii SIMS. Zapojenie aplikácie do reálnej prevádzky je možné vidieť na obrázku 7.1.



Obrázek 7.1: Zapojenie modulu HTTP do reálnej prevádzky.

Výstup apliácie

Očakavaný výstup aplikácie pri menších dátach som určil na základe analýzy sieťového toku v programe *WireShark*. Pri analýze sieťového toku v tomto programe som pre prehliadnosť použil filter `tcp.port == 80`, ktorý vymedzuje zobrazenie toku HTTP. Následne s daného vypisu bolo zistené že v danom toku je len jedna HTTP požiadavka, viď obrázok 7.2 a taktiež že dané sieťové spojenie sa ukončí. Ukončenie spojenia bolo určené na základe

paketu, ktorý obsahoval príznak FIN. Z tejto analýzy vyplíva že navrhnutá aplikácia by mala generovať dve správy a to typu *BEGIN* a *END*. V tomto teste bol použitý inicializačný súbor, ktorý obsahoval sledovanie hlavičiek *User-Agent,Accept-Encoding,Accept-Language* a portu 80. Po kontrole bolo preukazané že výstup aplikácie sa zhoduje s očakavaným výstupom, získaného pri analýze tohto sieťového toku. Výstup správ aplikácie je možné vidieť nižšie. Pri tomto teste je využitý súbor `test01.pcap`.

```
Požiadavka klienta:

  GET / xjelen07/ HTTP/1.1
  Host: www.stud.fit.vutbr.cz
  Connection: keep-alive
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
  image/webp,*/*;q=0.8
  User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36
  Accept-Encoding: gzip, deflate, sdch
  Accept-Language: sk-SK,sk;q=0.8,cs;q=0.6,en-US;q=0.4,en;q=0.2
  Cookie: _ga=GA1.2.1226546804.1412882412;
  __utma=257111820.1226546804.1412882412.1427400768.1428586233.65;
  __utmz=257111820.1423736194.56.9.utmcsr=feec.vutbr.cz|utmccn=
  (referral)|utmcmd=referral|utmcct=/vyjezd/detail.php.cz
  If-Modified-Since: Sun, 02 Nov 2014 23:36:49 GMT
```

Obrázek 7.2: Hodnoty zobrazené v programe WireShark pri súbore test01.pcap

Výstupná správa aplikácie typu BEGIN:
`('http', 1428771501.416402, 'BEGIN', 'Otvorenie noveho spojenia', [('TCP', ('147.229.176.14', 80, '147.229.220.216', 38943)), ('HTTP Headers', 'Accept-Language': 'sk-SK,sk;q=0.8,cs;q=0.6,en-US;q=0.4,en;q=0.2', 'Accept-Encoding': 'gzip, deflate, sdch', 'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36')], [('URI', 'http://www.stud.fit.vutbr.cz/ xjelen07/')], [])`

Výstupná správa aplikácie typu END:
`('http', 1428771501.416402, 'END', 'Koniec spojenia', [('TCP', ('147.229.176.14', 80, '147.229.220.216', 38943)), ('HTTP Headers', 'Accept-Language': 'sk-SK,sk;q=0.8,cs;q=0.6,en-US;q=0.4,en;q=0.2', 'Accept-Encoding': 'gzip, deflate, sdch', 'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36')], [('URI', 'http://www.stud.fit.vutbr.cz/ xjelen07/')], [])`

Pri testovaní väčších sieťových tokov nieje takato métoda vhodná. Overenie takýchto súborov je možné určiť na základe počtu očakavaných správ. Na zistenie počtu HTTP požiadaviek bol takisto použitý program *WireShark*. Pri analýze súboru `test02.pcap`

boli zistené nasledujúce údaje. Celkový počet HTTP požiadaviek je `1403`, pričom `1399` HTTP požiadaviek komunikuje na porte `80` a `4` HTTP požiadavky komunikujú na porte 2869. V tomto teste bol použitý inicializačný súbor, ktorý obsahoval sledovanie hlavičiek `User-Agent,Accept-Encoding,Accept-Language` a sledované porty uvedené v tabuľke. Výsledky testovania je možné vidieť v tabuľke 7.1. Pri tomto testovaní neboli vyhľadavané HTTP cookies. Aplikácia bola spustená s parametrami `-e` a *-t*.

Sledované porty	Počet všetkých paketov	Počet vytvorených správ			Počet správ zistený alalýzou
		BEGIN	CONTINUE	Spolu	
80	34 201	588	811	1399	1399
2869	34 201	4	0	4	4
80 a 2869	34 201	592	811	1403	1403

Tabulka 7.1: Počet zaslaných a očakavaných správ súboru test02.pcap.

Aktualizácia tabuľky

Ďalší test, bude demonštrovať aktualizačný mechanizmus, viď 6.3, ktorý prebieha ak sa hodnoty HTTP hlavičiek alebo HTTP cookies menia. V tomto teste bol použitý inicializačný súbor, ktorý obsahoval sledovanie hlavičiek `User-Agent,Accept-Encoding,Accept-Language`, takiež sledovanie cookies `'ulv'` s globálnou platnosťou a portu `80`. Analýzou súboru `test03.pcap` v programe *WireShark*, bolo zistené že súbor obsahuje dva pakety toho istého sieťového spojenia. Hlavičky týchto paketou sa zhodujú ale hodnota cookies `ulv` je v každom pakete iná. Pri takejto aktualizácii sa aplikácia riadi aktualizačným mechanizmom, ktorý je možné vidieť na vývojovom diagrame 6.2.

Výstup testovania (pre jednoduchosť boli vynechané všetky NIDy okrem HTTP Cookies) :

```
('http', 1428694299.670724, 'BEGIN', 'Otvorenie noveho spojenia',
[('HTTP Cookies', 'ulv': '1416196977-13')])

('http', 1428694306.971214, 'BEGIN', 'Otvorenie noveho spojenia',
[('HTTP Cookies', 'ulv': '1417890787-100')])

('http', 1428694299.670724, 'END', 'Zatvorenie spojenia', [('HTTP Coo-
kies', 'ulv': '1416196977-13')])
```

Pre porovnanie výstupu, znova spustíme súbor `test02.pcap`, pričom inicializačný súbor obsahuje sledovanie hlavičiek `User-Agent,Accept-Encoding,Accept-Language`, sledovanie cookies `__utmz, __utma, Gdyn, ulv, ulvt` s globálnou platnosťou a sledovanými portmi uvedenými v tabuľke 7.2. Ako je vidno, počet správ BEGIN a CONTINUE sa zmenil pričom počet vytvorených správ ostáva nezmenený. Tento rozdiel oproti tabuľke 7.1 je preto, že v aplikácií došlo 33-krát k aktualizácii tabuľky identít.

Sledované porty	Počet všetkých paketov	Počet vytvorených správ			Počet správ zistený alalýzou
		BEGIN	CONTINUE	Spolu	
80	34 201	621	778	1399	1399
2869	34 201	4	0	4	4
80 a 2869	34 201	625	778	1403	1403

Tabulka 7.2: Počet zaslaných a očakavaných správ súboru test02.pcap s vyhľadávaním cookies.

Spojenie aplikácie s nástrojom SIMS

Tento test je zameraný na spávnosť prijatých správ nastrojom SIMS poslaných s modulu HTTP a tvorbu grafu s týchto správ. Pomocou zapnutí nástroja SIMS v administratívnej funkcii je možné sledovať prijaté správy od jednotivých modulov. Na tento test bol využitý súbor `test05.pcap`. Jeho analýzov bolo zistené že obsahuje dve HTTP požiadavky v tom istom sieťovom toku s rovnakými hlavičkami a ich hodnotou. S toho je možne určiť že modul vygeneruje dve správy typu BEGIN a CONTINUE. V danom sieťovom toku sa nanachádza žiaden paket obsahujúci príznak FIN, preto aby modul nevytváral správu END, vďaka ktorej by nástroj SIMS v grafe zmazal všetky prepojenia tohto sieťového spojenia. Aplikácia modul HTTP bola spustená s parametrami `-s` a *-t*.

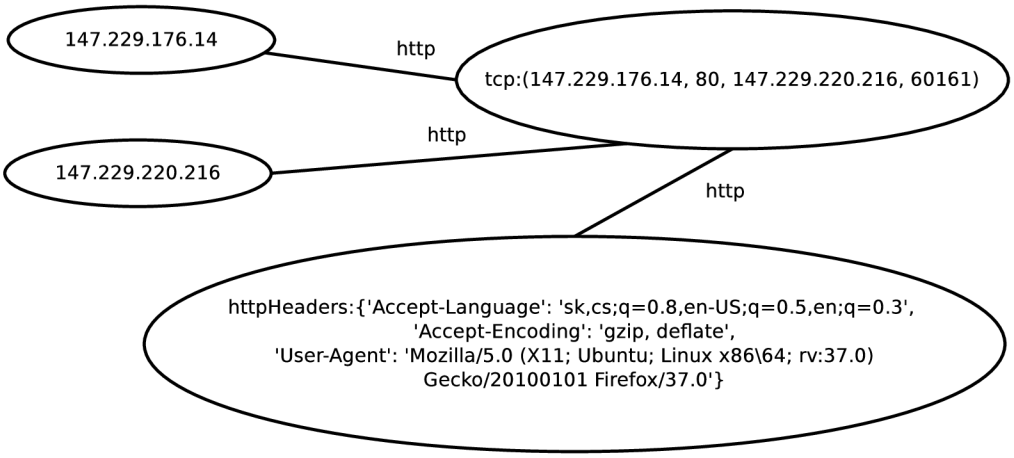
Kontrolov bolo zistené že, ze SIMS prijal vytvorené správy BEGIN a CONTINUE a vytvoril súbor `irip.content`, ktorý osahoval graf NIDov. Pre pravidlá tvorby grafu NIDov, viď sekcia 4.1.

Obsah súboru `irip.content`:

1. `http  147.229.176.14  tcp:(147.229.176.14, 80, 147.229.220.216, 60161)`
2. `http  tcp:(147.229.176.14, 80, 147.229.220.216, 60161)`
   `147.229.220.216`
3. `http  httpHeaders:{'Accept-Language': 'sk,cs;q=0.8,en-US;q=0.5,en;q=0.3', 'Accept-Encoding': 'gzip, deflate', 'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:37.0) Gecko/20100101 Firefox/37.0'}  tcp:(147.229.176.14, 80, 147.229.220.216, 60161)`

V tomto vytvorenom súbore každý riadok určuje hranu medzi vrcholmi uvedenými v danom riadku. Obsahuje názov modulu, ktorý vytvoril správu a dva vrcholy medzi ktorými vznikne hrana. Všetky položky v riadku sú oddelené tabulátorom. Pre tvorbu grafickej podoby grafu som využil skript `dotify-iri.py` v dostupný v nástoji SIMS. Tento skript prevádza obsah súboru `irip.content` do formy určenej pre aplikáciu `dot`, ktorá vytvára graf na základe formatovaných vstupných dát. Celý príkaz ma tvar `./dotify-iri.py | dot -Tsvg -o graf.svg`. Vytvorený graf je možné vidieť na obrázku 7.3.

Následne brebehla rada testou výkonnosti aplikácie pri zasielaní správ nástroju SIMS. Testy prebehli na čas vykonávania, počet všetkych paketov, ktoré modul HTTP prijal, počet sledovaných paketov určených na základe sledovaného portu, počet spracovaných paketov vyhovujúcim metodám a následne čas takého to spracovania súboru. Výsledky je možné vidieť v tabuľke 7.3.



Obrázek 7.3: Graf NIDov súboru test05.pcap.

Názov súboru	Počet všetkých paketov	Počet sledovaných paketov	Počet spracovaných požiadaviek	Čas vykonávania
test01.pcap	2497	21	1	1.646 s
test02.pcap	34201	30519	1399	27.196 s
test03.pcap	4	21	1	0.267 s
test05.pcap	3008	23	1	1.912 s
test00.pcap	0	0	0	0.255 s
test08.pcap	69084	61038	2798	59.648 s
test09.pcap	272050	238027	11455	231.460 s

Tabulka 7.3: Prehľad jednotlivých testov a čas vykonávania.

7.2 Testovanie použiteľnosti

Rozlišiteľnost webových prehliadačov

V tejto časti sa zameriam na identifikátory HTTP požiadaviek `Accept-Encoding`,`User-Agent`,`Accept-Language` a HTTP Cookies `__utma`, `__utmz` v inicializačnom súbore. Tento experiment sa bude skladať s dvoch krokov. Všetky získané dáta a grafy sú uložené na priloženom CD a to pre krok *A* v adresári `test07/A/` a pre krok *B* v adresári `test07/B/`.

Krok A

Aplikácia bola spúšťaná s parametrami `-s -e -t`, súborom `test07.pcap` a inicializačný súbor obsahoval sledovalie hlavičky *Accept-Encoding* na porte `80`. Po spustení nástroj SIMS vytvoril graf NIDov v súbore `irip.content`. Po vytvorení grafickej formy tohoto grafu `graf1.svg`, som zistil že graf nieje prehľadný preto som vykonal upravenie grafu na sledovanú položku HTTP Header a to príkazom `cat irip.content | grep 'httpHeaders' > hlavicky`. Následne som z tohoto súboru vytvoril graficku podobu grafu `graf2.svg` a to príkazom `./dotify-iri.py hlavicky | dot -Tsvg -o graf2.svg`. Pri analýze vytvoreného grafu som zistil že každé sieťové spojenie (čiastočná identita) má rovnaký slovník HTTP hlavičiek a že klientske IP adresy boli `10.10.10.111, 10.10.10.118`

a `10.10.10.131`. Následne základe ziskaných informácií musíme predpokladať že všetký sieťové spojenia sú od jedného webového prehliadača.

Krok B

Aplikácia bola spúťaná s parametrami `-s -e -t`, súborom `test07.pcap` a inicializačný súbor obsahoval sledovanie hlavičiek *Accept-Encoding,User-Agent,Accept-Language* na porte `80` a HTTP Cookies `__utma` a `__utmz` s globálnou platnosťou. Tak ako v kroku *A* nástroj SIMS vytvoril graf NIDou. Pri analýze som zistil, že vytvorená grafická forma grafu `graf3.svg` je taktiež neprehľadná a upravil som ho na sledovanú položku HTTP Header a to príkazom `cat irip.content | grep 'httpHeaders' > hlavicky2`. Pri anlýze upraveného grafu bolo zitené že sieťové spojenia s IP adresy `10.10.10.118` a `10.10.10.131` majú spoločné HTTP požiadavk a HTTP Cookies narozdiel od sieťových spojení s IP adresy `10.10.10.111`. Na základe týchto informácii možeme určiť že sa jedná o ten istý prehliadač, ktorý komunikoval so serverom s IP adresy `10.10.10.111` a `10.10.10.131`. Oproti kroku *A* je vidieť že pridaním dalších identifikátorou rozlišiteľnost vzrastá a aj pravdepodobnosť úspešnej identifikacií. Pravdebodobnosť úspešnej identifikácie možeme určiť na základe zhodných identifikátorov. V grafe je možné vidieť že sieťové spojenia s IP adresy `10.10.10.111` a `10.10.10.131` sa zhodujú nielen v HTTP hlavičkách ale aj v HTTP cookies.

Experiment identifikátora Cookie

Pri experimentovaní s danou aplikáciou som zistil, že identifikátory HTTP cookies sa možu členiť. Preto bola implementovaná možnosť určiť si, ktorú čas danej cookie má aplikácia brať do úvahy pre identifikáciu. Ale keďže server nemusí špecifikovať načo sa hodnota cookies využiva, tak jednotlivé jej časti je možné určiť iba experimenálne.

Pri tomto teste som sa zameral na cookie `Gdyn`, ktoré nastavuje a spravuje server pre sledovanie návštěvnosti stránok a chovanie užívateľov `gemius.pl`. V tabuľke 7.4 je možné vidieť hodnoty skráteného identifikátoru `Gdyn`, pre rôznych užívateľov a ich webových prehliadačoch. Všetky sledované dáta tejto cookie je možné vidieť v súbore `test06.txt`.

Užívateľ	Prehliadač	Cookie `Gdyn`
1	Mozzila	KlQH8RG**GQMQGtXO2yjFbkM9IssGM**81DiLvnxmG89V
1	Mozzila	Klxz7Ma**GQMQGtXO2yjFbkM9IssGM**F1giLvnxmG89V
1	Chrome	KlSLQMX**GQMQGS4nl2ejSml9IssGM**QS1Zg8gxLbGHm
1	Chrome	KlQgLRM**GQMQGS4nl2ejSml9IssGM**Xj13g8gxLbGHm
2	Mozzila	KlSKKMM**GQMQGaZeXGBMykG2IssGM**Kn6hg8gxLbGH4
2	Mozzila	KlGb_MX**GQMQGaZeXGBMykG2IssGM**wMJhg8gxLbGH4
2	Chrome	KlS72MM**GQMQGtunQDIKbNsRIssGM**XQehg8gxLbGr3
2	Chrome	KlQH1RX**GQMQGtunQDIKbNsRIssGM**iQehg8gxLbGr3

Tabulka 7.4: Cookie Gdyn, pre rôznych užívateľov a webové prehliadače.

Na základe výsledkov experimentu je možné v inicializačnom súbore `start.ini`, zmeniť časť, ktorú modul HTTP bude brať do úvahy pri vytvárani slovníka HTTP Cookies. Takúto zmenu vykonáme pridaním riadku `Gdyn=7:28` do sekcie `[cookies_len]` v inicializačnom súbore.

Kapitola 8

Záver

Tento dokument vznikol ako bakalárska práca a je venovaný identifikácii pomocou požiadavkov HTTP. V úvode tohto dokumentu je vysvetlený protokol HTTP a princíp komunikácie. Detailnejšie sú rozobrané hlavičky protokolu. V ďalšej kapitole sú vysvetlené pojmy identita a identifikácia. Doležitý rozbor je taktiež venovaný HTTP hlavičke *User-Agent* a HTTP *Cookies*. Následne je spomenutý existujúci nástroj zaoberajúci sa identifikácii v HTTP. Nasledujúca kapitola je venovaná dynamickej identite v projekte *Sec6Net* a to konkrétne nástroja SIMS. Ďalej je rozobraná komunikácia modulov a nástroja SIMS. Zo všetkých týchto informácií z predchádzajúcich kapitol vznikol návrh aplikácie pre identifikáciu pomocou HTTP. Návrh aplikácie sa zaoberá vhodným výberom identifikátorov sieťového spojenia, základnou činnosťou aplikácie a taktiež integrovania navrhnutej aplikácie do nástroja SIMS.

Ďalšia kapitola rozoberá implementáciu daného nástroja. Sú v nej vysvetlené implementačné podrobnosti, ktoré sú vysvetlené pomocou vývojových diagramov. V predposlednej kapitole sú s danou aplikáciou vykonávané experimenty, ktoré overujú funkčnost aplikácie, správnosť vytvárania správ a následne je overená funkčnosť aplikácie s nástrojom SIMS. Taktiež je testovaná použiteľnosť nástroja a experiment vďaka ktorému, bolo možné prispôsobiť hodnotu HTTP cookies tak, aby sa dala považovať za identifikátor webového prehliadača.

Ďalší vývoj tejto bakalárske práce by mohol smerovať na komplexnejšie testovanie cookies, pretože počas testovania bolo zistené že cookies sa javí ako najlepší identifikátor. Ale keďže nieje špecifikované načo jednotlivé časti cookie slúžia je nutné každé cookies podrobné testovať. Taktiež by bolo možné, na základe ďalšieho výskumu priraďovať jednotlivým identifikátorom váhy, vďaka ktorý by sme mohli určovať dôležitosť zachytených identifikátov. Prípadne by bolo možné sledovať aj HTTP odpovede, nakoľko pri testovaní bol vidieť mechanizmus kde pri návšteve internetovej stránky, server zakaždým zmení užívateľovu cookie. Tieto informácie by bolo možné použiť pri tvorbe tabuľky kde by sme si zakaždým túto zmenu pamätali a na základe týchto informácií užívateľa identifikovali.

Literatura

[1] Eckersley, P.: Browser Versions Carry 10.5 Bits of Identifying Information on Average. Electronic Frontier Foundation.
URL https://www.eff.org/deeplinks/2010/01/tracking-by-user-agent

[2] Eckersley, P.: How Unique Is Your Web Browser? Electronic Frontier Foundation, str. 19.
URL https://panopticlick.eff.org/browser-uniqueness.pdf

[3] Kristol, D. M.: HTTP Cookies: Standards, Privacy and Politics. ACM Transactions on Internet Technology (TOIT), November 2001.
URL http://dl.acm.org/citation.cfm?id=502152.502153&coll=DL&dl=ACM&CFID=511713221&CFTOKEN=75504907

[4] Pfitzmann, A.; Hansen, M.: *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. 2010, 98 s.
URL https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf

[5] Polčák, L.; Martínek, T.; Hranický, R.; aj.: Zákonné odposlechy v moderních sítích - Shrnutí výsledků skupiny pro zákonné odposlechy projektu Moderní prostředky pro boj s kybernetickou kriminalitou na Internetu nové generace. Technická zpráva, 2014.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10788

[6] RFC 1945: Hypertext Transfer Protocol – HTTP/1.0. RFC 194975, Network Working Group, May 1996.
URL http://tools.ietf.org/html/rfc1945

[7] RFC 2068: Hypertext Transfer Protocol – HTTP/1.1. RFC 2068, Network Working Group, January 1997.
URL https://www.ietf.org/rfc/rfc2068.txt

[8] RFC 2616: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, Network Working Group, June 1999.
URL http://tools.ietf.org/html/rfc2616

[9] RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC 7230, Network Working Group, June 2014.
URL http://tools.ietf.org/html/rfc7230

[10] RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231, Network Working Group, June 2014.
URL http://tools.ietf.org/html/rfc7231

[11] RFC 7232: Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. RFC 7232, Network Working Group, June 2014.
URL https://tools.ietf.org/html/rfc7232

[12] RFC 7233: Hypertext Transfer Protocol (HTTP/1.1): Range Requests. RFC 7233, Network Working Group, June 2014.
URL https://tools.ietf.org/html/rfc7233

[13] RFC 7234: Hypertext Transfer Protocol (HTTP/1.1): Caching. RFC 7234, Network Working Group, June 2014.
URL https://tools.ietf.org/html/rfc7234

[14] RFC 7235: Hypertext Transfer Protocol (HTTP/1.1): Authentication. RFC 7235, Network Working Group, June 2014.
URL https://tools.ietf.org/html/rfc7235

[15] RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, Internet Engineering Task Force, May 2015.
URL https://tools.ietf.org/html/rfc7540

[16] Zakas, N. C.: Professional JavaScript for Web Developers. January 2009: str. 840.

Příloha A

Obsah CD

Priečinok `/Modul HTTP` - obsahuje zdrohový kód modulu HTTP a inicializačný súbor.

Priečinok `/SIMS` - obsahuje aplikáciu SIMS bez modulu HTTP.

Priečinok `/SIMS-HTTP` - obsahuje aplikáciu SIMS, v ktorej je už integrovaný modul HTTP.

Pričinok `/Testy` - testované dáta a pre jednotlivé testy im odpovedajúce inicilizačné súbory.

Pričinok `/BP_Latex` - obsahuje zdrojové kódy technickej časti bakalárskej práce.

`README` - Návod na inštaláciu a použitie.

Súbor `ISA2014.ova` - Obraz virtuálneho počítača prevzatý s predmetu Sieťové aplikácie a správa sieti 2014/2015.

Příloha B

Manual

V tejto kapitole je popísaná inštalácia a práca s aplikáciou. Aplikácia pre svoju funkčnosť vyžaduje knižnicu Scapy, ktorá je dostupná s `http://www.secdev.org/projects/scapy/`.

B.1 Inštalácia

1. Krok - Prekopírujeme obsah adresára `/SIMS` do adresára na disku počítača (napr. do priečinku /aplikacia).
2. Krok - Do adresára s nakopírovanou aplikáciou SIMS, nakopírujeme obsah adresára `/Modul HTTP` a to konkrétne do adresára `/src/modules/` (napr. /aplikacia/src/modules/). Pri správnej inštalácii by sa mal zdrojový kód modulu HTTP nachádzať v adresáry `/src/modules/iriiif/http/` (napr. aplikacia/src/modules/iriiif/http/http.py).

B.2 Použitie

Pri používaní daných aplikácií je potrbné aby užívateľ mal práva administrátora tzv. ROOT práva. Aplikácie sa spúštajú z konzoly.

B.2.1 Aplikácia SIMS

- Aplikácia SIMS sa ovláda skriptom `sims.sh`, ktorý je umiestnený v zložke `/src/` (napr. aplikacia/src/)
  - `./sims.sh start`, znamená zapnutie aplikácie SIMS.
  - `./sims.sh start info`, znamená zapnutie aplikácie SIMS s pomocným výpisom.
  - `./sims.sh stop`, znamená ukončenie aplikácie SIMS.
- Príklad spustenia a následného ukončenia:

  kubo@kubo: /aplikacia/$ sudo su
  root@kubo: /aplikacia/src# cd src
  root@kubo: /aplikacia/src# ./sims.sh start info
  root@kubo: /aplikacia/src# ./sims.sh stop

- Vytváranie grafov ich úprava príklady (pokiaľ skript dotify-iri.py nemá argument tak sa vytvára graf z obsahu irip.content):

  kubo@kubo: /aplikacia/$ ./dotify-iri.py — dot -Tsvg -o graf.svg

  root@kubo: /aplikacia/src# cat irip.content — grep httpHeaders ¿ hlavicky

  root@kubo: /aplikacia/src# ./dotify-iri.py hlavicky — dot -Tsvg -o graf2.svg

B.2.2 Modul HTTP

- Aplikácia Modul HTTP sa spúšťa skriptom `http.py`, ktorého cesta je zadávaná zo zložky `/src/` a nasledujúcimi argumentmi:
  - `-i=rozhranie`, znamená zapnutie sledovania na zadanom rozhraní napr. eth0. Pri takomto zapnutí aplikácia beží v nekonečom cykle, a ukončenie je možné na základe prijatých signálov SIGINT, SIGTERM a SIGQUIT. Tento parameter nemožno kombinovať s parametrom `-p=nazov`.
  - `-p=nazov`, znamená zapnutie načitávania pcap súboru zadaného ako hodnota `nazov`. Tento parameter nemožno kombinovať s parametrom `-i=rozhranie`.
  - `-s`, znamená zasielanie správ aplikácií SIMS.
  - `-r`, znamená že pri tvorbe správ bude aplikácia odosielať to čo v prvom zozname aj v treťom.
  - `-e`, znamená že aplikácia nebude vytvárať END správy.
  - `-t`, znamená že aplikácia po ukončení vypíše tstovacie štatistiky.
- Pri spúštaní aplikácie so súbormi pcap musia byť súbory nakopírované v adresáry, kde sa nachádza zdrojový súbor aplikácie, čiže v adresáry `/src/modules/iriiif/http/` (napr. aplikacia/src/modules/iriiif/http/).
- Príklad spustenií aplikácie:

  kubo@kubo: /aplikacia/$ sudo su

  root@kubo: /aplikacia/src# cd src

  root@kubo: /aplikacia/src# python modules/iriiif/http/http.py -p=modules/iriiif/http/test02.pcap

  root@kubo: /aplikacia/src# python modules/iriiif/http/http.py -i=eth0

  root@kubo: /aplikacia/src# python modules/iriiif/http/http.py -p=modules/iriiif/http/test02.pcap -s -t -r -e