VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

REKONSTRUKCE ZACHYCENÉ KOMUNIKACE A DOLOVÁNÍ DAT NA SOCIÁLNÍ SÍTI FACEBOOK

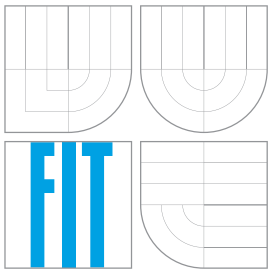
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

TOMÁŠ BRUCKNER

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

REKONSTRUKCE ZACHYCENÉ KOMUNIKACE A DOLOVÁNÍ DAT NA SOCIÁLNÍ SÍTI FACEBOOK

FACEBOOK SOCIAL NETWORK DATAMINING AND RECONSTRUCTION OF CAPTURED COMMUNICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ BRUCKNER
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN PLUSKAL
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce se zabývá sociální síti Facebook z pohledu počítačové forenzní vědy se zaměřením na získávání citlivých a potenciálně užitečných informací o sledovaných uživatelích. Cílem této bakalářské práce je implementace nástrojů k rekonstrukci síťové komunikace a dolování dat z této sociální sítě. Jádro aplikace bylo implementováno ve frameworku pro zpracování zachycené komunikace Netfox.Framework, který je vyvíjen na Fakultě informačních technologií, Vysokého učení technického v Brně. Pro zachycení dešifrovatelné komunikace byl využit útok Man-in-the-Middle. Pro dolování dat bylo využito nástroje Selenium WebDriver, který funguje jako DOM parser. Vytvořené řešení poskytuje možnost rekonstruovat konverzace mezi uživateli, přidávání stavů, komentářů a souborů. Z pohledu dolování dat poskytuje vytvořené řešení možnost získání veřejných informací o uživatelích, zejména místa, na kterých se uživatel v poslední době pohyboval, událostí, kterých se účastnil či se plánuje účastnit, fotoalba a fotky, seznam jeho kamarádů a seznam společných kamarádů s jiným uživatelem. V rámci řešení byla provedena analýza dat, implementace aplikace, testování na laboratorních datech a výkonnostní analýza.

Abstract

This thesis deals with social network Facebook from perspective of computer forensic science with focus on obtaining sensitive information about tracked users. Its main goal is implementation of the tools for reconstruction of captured communication and Facebook data mining. Core of this application has been implemented in framework for processing of captured communication Netfox.Framework, which is being developed by Faculty of Information Technology, Brno University of Technology. Man-in-the-Middle attack has been used for capturing of decipherable communication. Selenium WebDriver has been used as a tool for data mining. Developed solution is able to reconstruct Facebook conversations between users, addition of new statuses and comments, and interception of sent files. Data mining module is able to obtain public information about tracked users, especially places they recently visited, past and upcoming events, public user details, albums and photos, friendlists and mutual friends with other users. As part of the bachelor thesis, data analysis, implementation of the application, validity testing and benchmark analysis have been performed.

Klíčová slova

Facebook, Man-in-the-Middle, rekonstrukce dat, dolování dat, NetFox, SSL, HTTPS, Graph API, Selenium, Diffie-Hellman, RSA

Keywords

Facebook, Man-in-the-Middle, data reconstruction, data mining, NetFox, SSL, HTTPS, Graph API, Selenium, Diffie-Hellman, RSA

Citace

Tomáš Bruckner: Rekonstrukce zachycené komunikace
a dolování dat na sociální síti Facebook, bakalářská práce, Brno, FIT VUT v Brně, 2015

Rekonstrukce zachycené komunikace a dolování dat na sociální síti Facebook

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pluskala. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Bruckner
19. května 2015

Poděkování

Chtěl bych poděkovat svým rodičům, kteří mi umožnili studovat na vysoké škole, a kteří mne podporovali psychicky i finančně. Dále bych chtěl poděkovat Ing. Janu Pluskalovi za odbornou pomoc, cenné rady a motivaci v průběhu vypracování celé bakalářské práce.

Obsah

Kapitola 1

Úvod

Ve vědomostech je síla. Marně bychom hledali silnější zbraň, než kterou nám poskytuje síla informací. Zvláště v dnešním digitálním světě, kdy zpracování obrovského množství dat je již trivialita, o které by si naši předkové mohli jen nechat zdát. Problém ale nastává v okamžiku získávání potřebných údajů o jednotlivcích. Kde a jak se nejlépe dostat k osobním datům sledovaných osob? K tomu nám nahrávají trendy současné doby spolu s výdobytky moderních technologií. Čím dál větší množství populace začíná toužit po neustálém připojení k síti internet, ať už skrze počítač, telefon nebo tablet. Jistě lze argumentovat, že existuje určité kouzlo mít neustále na dosah studnici moudrosti, které lidstvo získalo za celou svoji existenci na planetě Zemi. Jen několik stisků tlačítka nás dělí od nejskvostnějších uměleckých děl z antického Řecka, podmanivých skladeb hudebních géniů dynamického Baroka či univerzálního slovníků všech používaných i mrtvých jazyků světa. S rozmachem internetu se nám také otevírají naprosto nové způsoby komunikace se svými blízkými či kamarády ve formě sociálních sítí. Za posledních několik let prošly sociální sítě prudkým vývojem, kdy se jako nejdominantnější projevila sociální síť Facebook. Lze se jenom dohadovat, zda její úspěch způsobil jednoduchý design, zábavný obsah, intuitivní ovládání či možnost kontaktovat staré známé. Faktem ale zůstává, že se stala neodmyslitelnou součástí života moderní počítačové generace. Sdílíme zde nejenom své fotky, pocity a nálady, ale především také své soukromí a osobní informace. Mnozí lidé si vůbec neuvědomují, kolik intimních informací o jejich osobě lze naprosto veřejně dohledat, či co všechno na sebe prozradí v integrovaném chatu pod klamným zdáním bezpečí.

Tato problematika je také základem i mé bakalářské práce, kde jsem se pokusil o nalezení způsobů získávání dat ze sociální sítě Facebook pomocí dolování dat a rekonstrukce komunikace. Jedná se o téma vypsané výzkumnou skupinou počítačových sítí a vestavěných systémů NES@FIT v oblasti bezpečnosti na internetu při projektu SEC6NET ve spolupráci s Ministerstvem vnitra ČR. V první části jsem se zabýval představením sociální sítě Facebook a identifikací způsoby komunikace z hlediska počítačových sítí. Dále byla provedena hloubková analýza formální struktury a komunikačních protokolů sociální sítě se zaměřením na způsoby zachycení dat v dešifrované podobě.

Výsledný experimentální nástroj byl implementován s využitím frameworku pro zpracování zachycené komunikace Netfox.Framework, který patří do projektu NETwork FOrensic eXtendable analysis tool (NetFox) vytvořeného z grantu projektu SEC6NET na Fakultě informačních technologií Vysokého učení technického v Brně. Framework byl realizován v jazyce C#, což přímo implikovalo i programovací jazyk této bakalářské práce.

Z hlediska dolování dat byla provedena analýza dostupných možností a využití Facebook API se zaměřením na platformu .NET. Dále bylo polemizováno nejenom nad způsobem

získávání užitečných dat, ale také nad analýzou získaných dat pomocí metody dolování dat. Lehce zde byla promítnuta morální stránka získávání dat ze sociální sítě, jelikož se ve většině případů jednalo o soukromá či citlivá data. V tomto ohledu je třeba doufat, že vytvořená aplikace bude v budoucnu použita k legálním operacím. Možnosti zneužití jsou zde lehce představitelné a zabránění daného zneužití je velmi obtížné, vzhledem k open source licenci celého projektu.

Práce se dále podrobně zabývala způsoby zabezpečení protokolu HTTPS s využitím protokolů SSL a TLS. Pochopení této problematiky bylo základem k úspěšnému implementování dešifrovacích nástrojů do námi zvoleného nástroje Netfox.Framework. Velký důraz byl kladen na způsob výměny klíčů pomocí protokolů SSL/TLS se zaměřením na algoritmy RSA a různé verze Diffie-Hellmana.

Kapitola 2

Šifrovaná komunikace

Secure Sockets Layer (SSL) a jeho moderní nástupce Transport Layer Security (TLS) jsou klient/server protokoly využívající šifrování pomocí veřejného klíče, které poskytují komunikujícím stranám následující bezpečnostní služby [7]:

- Autentizaci komunikujících stran a původu přenášených dat
- Důvěrnost vytvořeného spojení
- Integritu vytvořeného spojení

SSL funguje nad TCP jako peer-to-peer spojení. Jedná se o mezivrstvu mezi vrstvou transportní a aplikační. Varianty protokolů zabezpečené pomocí SSL většinou komunikují na jiném standardním portu, než jejích nezabezpečené varianty, např. HTTPS na portu 443 [11], IMAPS na portu 993 [6] atd.

SSL se skládá ze čtyř podprotokolů:

- SSL Handshake protokol - hlavní jádro SSL, využívá se k autentizaci komunikujících stran a vytvoření zabezpečené relace
- SSL Change Cipher Spec protokol - oznámení okamžiku, kdy se začal používat dohodnutý cipher suite viz kapitola 2.3
- SSL Alert protokol - oznámení problémů, které se vyskytly
- SSL Application Data protokol - pro přenos dat

Dále si představíme především handshake protokol, který je z hlediska SSL nejdůležitější.

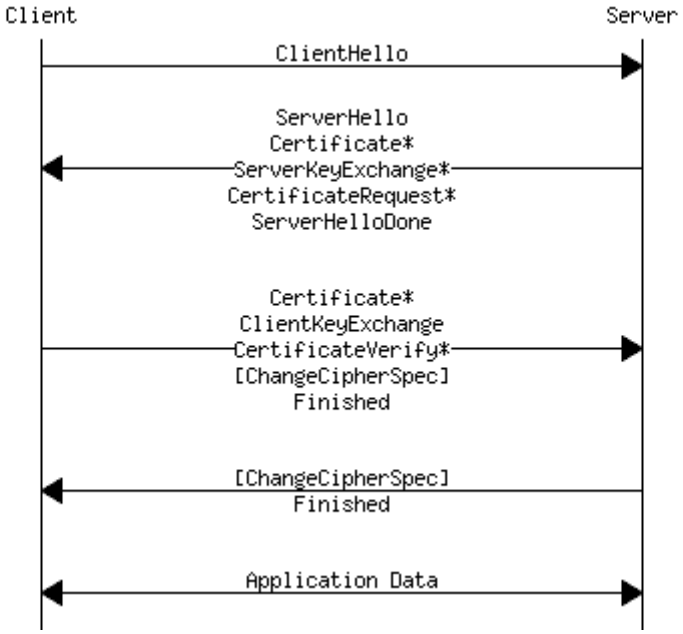
2.1 Handshake protokol

Ustanovení SSL/TLS relace a nastavení potřebných parametrů probíhá pomocí tzv. handshake protokolu. Před zahájením šifrované komunikace se klient a server dohodnou, jakou budou používat verzi SSL/TLS, jaký bude použit šifrovací algoritmus, případně dojde k autentizaci a použití techniky šifrování pomocí veřejného klíče k vygenerování sdíleného tajemství[1].

[1] V angličtině shared secret.

Nejdříve klient pošle zprávu `ClientHello`, na kterou server odpoví `ServerHello`. V těchto zprávách se domluví na verzi protokolu, ID relace, které je možno později využít k ustanovení nové relace se stejnými parametry, cipher suite a komprimační metoda. Klient pošle všechny jím podporované komprimační metody seřazené dle preference a server si z nich vybere pro něj nejvhodnější metodu. Každá strana si vygeneruje náhodné číslo o velikosti 32 bytů, které pošle protistraně. Server dále pošle svůj certifikát, pokud je nutné ho autentizovat. V případě, že server certifikát nemá nebo se jedná o certifikát pro podepisování, pošle se zpráva `ServerKeyExchange`. Dle zvoleného *cipher suite* si server může požádat o certifikát klienta a následně odešle zprávu `ServerDone`, čímž klientovi oznámí, že zatím je to od něj vše a čeká na odpověď klienta [3].

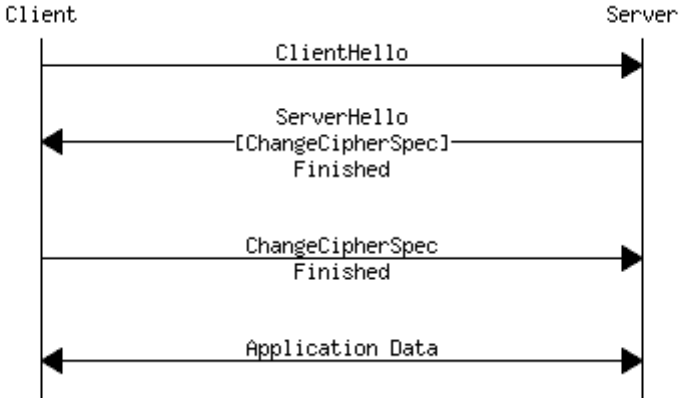
Pokud server poslal žádost o klientský certifikát, tak mu musí být klientem poslán. Klient serveru zároveň pošle digitálně podepsanou zprávu `CertificateVerify` prokazující, že je vlastníkem veřejného klíče. Následně dojde k poslání zprávy `Finished`. Pokud vše proběhlo správně, tak server posílá zprávu `Finished` a může dojít k přenosu aplikačních dat. Průběh konverzace viz diagram 2.1.



Obrázek 2.1: Plný SSL handshake protokol.

Krom plného handshake protokolu existuje i zkrácená verze. Pří plném handshaku server pošle klientovi ID relace. Pokud se později klient rozhodne vytvořit novou zabezpečenou relaci, tak může ve zprávě `ClientHello` poslat serveru ID předcházející relace. Tím prokazuje, že si stále udržuje informace o použitém *cipher suite* a klíčích z předcházejícího vytváření relace. Pokud má i server stále uložené v paměti ID této relace (tzv. i její parametry), může tuto informaci přidat do zprávy `ServerHello` a dojde ke zkrácené verzi handshake protokolu [3].

Výhody jsou především v rychlosti ustálení nové relace. Také nedochází k vytváření nového symetrického klíče pomocí asymetrické kryptografie. Jak dlouho si klient/server uchovává dané informace záleží čistě na jednotlivých nastavení klienta/serveru. Nejčastěji se jedná o hodiny až dny. Pokud došlo ke smazání informací o dané relaci, musí dojít opětovně k plnému handshaku. Průběh konverzace zkráceného verze handshake protokolu viz 2.2.



Obrázek 2.2: Zkrácený SSL handshake protokol.

2.2 Algoritmy pro výměnu klíče

Pro výměnu klíčů se používá několik algoritmů[2] a vždy záleží na dohodě klienta se serverem, který se nakonec použije. Zde si představíme jen ty nejpoužívanější algoritmy, RSA a Diffie-Hellman. Při výměně klíčů pomocí algoritmu RSA si klient vygeneruje náhodné číslo velikost 48 bytů nazývané `premaster secret`, zašifruje ho pomocí veřejného klíče serveru a výsledek pošle na server. Veřejný klíč serveru může být buď získaný z certifikátu veřejného klíče daného serveru nebo se může jednat o krátkodobý klíč vytvořený serverem konkrétně pro individuální relaci. V obou případech server rozšifruje `premaster secret` pomocí svého soukromého klíče.

Při použití algoritmu Diffie-Hellman se musí specifikovat jedna ze třech existujících variant. V pevně dané výměně pomocí Diffie-Hellmana[3] jsou pevně dané parametry pro výměnu klíče. Samotné parametry se nikdy nemění a jsou součástí certifikátu serveru, který je podepsaný příslušnou certifikační autoritou.

V emphemeral verzi Diffie-Hellmana[4] jsou parametry dynamicky generovány pro jednotlivé relace. K autentizaci těchto parametrů se nejčastěji využívá digitálního podpisu pomocí serverového privátního klíče. Uživatel si pak podpis může jednoduše ověřit pomocí veřejného klíče serveru (ten je zase autentizován pomocí certifikační autority).

Třetí verze je anonymní Diffie-Hellman[5], který je založen na ephemeral verzi Diffie-Hellmana, akorát neprobíhá autentizace dynamických parametrů. Z uvedených algoritmů je ephemeral verze Diffie-Hellmana považována za nejvíce bezpečný [7].

2.2.1 RSA

RSA je šifra s veřejným klíčem vytvořena autory Ron Rivest, Adi Shamir a Len Adleman v roce 1977. Jednalo se o první funkční implementaci myšlenky matematiků Diffieho a Hellmana z roku 1976. Schéma RSA šifry umožňuje vytvoření bezpečné komunikace mezi dvěma stranami, které se nikdy předtím nepotkali. Tím došlo k prolomení dosavadní představy, že je nezbytně nutné, aby se komunikující strany před zahájením konverzace setkaly a vyměnily si privátní klíče relace.

[2] V angličtině key exchange algorithm.
[3] V angličtině fixed Diffie-Hellman key exchange se zkratkou DH.
[4] S používanou zkratkou DHE.
[5] V angličtině anonymous Diffie-Hellman key exchange se zkratkou DH_anon.

Šifra RSA se používá jak k asymetrickému šifrování dat, tak i k digitálnímu podpisu. Základem šifry je faktorizace čísel, nebo-li rozklad čísla na násobky, a obtížnost z rozloženého čísla získat původní násobky. Přínos RSA šifry byl uznán odbornou veřejností. V roce 2002 byli její autoři oceněni Turingovou cenou[6]. Šifra RSA se i dnes nadále používá a při použití dostatečně velkých prvočísel je považována za bezpečnou.

Postup při vytváření klíče pomocí RSA algoritmu:

1. Vyberou se dvě rozdílné prvočísla p a q
2. Spočítá se modulus $n = pq$
3. Spočítá se Eulerova funkce $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1)$
   (a) e se použije pro vytvoření veřejného klíče
   (b) Multiplikativní inverze $d = e-1 \mod \phi(n)$ se použije pro vytvoření soukromého klíče

Veřejný klíč používá e jako svůj exponent a n jako modulus, privátní klíč používá d jako svůj exponent a n jako svůj modulus. Pokud máme zprávu m, pak ji zašifrujeme jako [7]:

$$c = me \mod n \tag{2.1}$$

Následně se zpráva dešifruje jako:

$$m = cd \mod n \tag{2.2}$$

2.2.2 Diffie-Hellman

Výměna klíče Diffie-Hellman je způsob vytvoření symetrického klíče bezpečným způsobem mezi dvěma uživateli bez předcházejícího kontaktu. Algoritmus je založen na modulárním umocňování.

Postup při vytváření klíče pomocí Diffie-Hellman algoritmu[7]:

1. Alice a Bob se domluví na použití prvočísla p a základu g
2. Alice si zvolí náhodné celé číslo $x_A < p$, Bob si zvolí náhodné číslo $x_B < p$
3. Alice si spočítá $y_A \equiv g^{x_A} \mod p$ a pošle ho Bobovi, Bob si spočítá $y_B \equiv g^{x_B} \mod p$ a pošle ho Alici
4. Alice si spočítá $z_A \equiv y_B^{x_A} \mod p$, Bob si spočítá $z_B \equiv y_A^{x_B} \mod p$, ale platí $z_A = z_B$

[6] Blasi, P. J. D.: Developers of Encryption Code Win ACM's Highest Honor. Association for Computing Machinery. 2002 [cit. 2014-12-20], http://www.acm.org/announcements/turing_2002.html.

[7] Baker, K. A.: Diffie-Hellman key exchange. University of Leeds. 1999 [cit. 2014-11-26], http://www.math.ucla.edu/~baker/40.1.99w/handouts/rev_DH/node1.html.

2.3 Cipher suite

Cipher suite je označení pro sadu algoritmů, které se používají v SSL/TLS pro vytvoření bezpečného spojení. Patří zde algoritmus s veřejným klíčem, např. RSA a Diffie-Hellman, hašovací algoritmus, např. MD5 a SHA, a symetrická šifra, např. RC2, RC4, IDEA-CBC, DES-CBC a 3DES-CBC[8]. Konkrétní cipher suite se volí ve zprávách `ClientHello` a `ServerHello` handshake protokolu.

Obecná forma pro výběr cipher suite je ve tvaru:

$$TypProtokolu\_WITH\_SymetrickyAlgoritmus\_HasovaciAlgoritmus \tag{2.3}$$

Příkladem cipher suite v SSL/TLS může být:

- SSL_RSA_WITH_RC4_128_SHA
- TLS_DHE_WITH_DES40_CBC_MD5
- TLS_RSA_WITH_NULL_SHA

Pokud je v políčku symetrického algoritmu uvedeno NULL, jedná se o spojení bez šifrování a standardně je zakázáno[9].

[8] Hanacek, P.; Asymetrická správa klíčů. FIT VUT v Brně, 2014. https://www.fit.vutbr.cz/study/courses/KRY/private/kry05.pdf.

[9] OpenSSL Documents. Dostupné z: https://www.openssl.org/docs/apps/ciphers.html.

Kapitola 3

Zachycení dešifrovaných dat

V této kapitole si rozebereme jednotlivé způsoby, kterými jsem postupoval při analýze Facebook protokolu a následném zachycování síťové komunikace do formátu PCAP. Dále si naznačíme princip útoku Man-in-the-Middle, který je základem pro všechny následující metody dešifrování HTTPS provozu.

3.1 Man-in-the-Middle

Man-in-the-Middle (MITM) je v kryptografii kybernetický útok, který umožňuje útočníkovi monitorovat a upravovat síťový provoz. Zneužívá především nedokonalosti v návrhu vytváření zabezpečeného spojení. V této bakalářské práci se budu zabývat aplikačním protokolem HTTPS [11], který se od protokolu HTTP liší zapouzdřením protokolu, komunikace probíhá na TCP portu 443 [11] a mezi HTTP a TCP je přidána šifrovací vrstva.

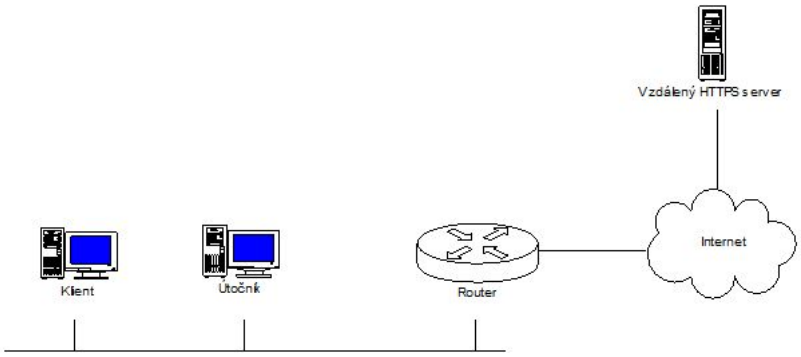
Předpokládejme zapojení z obrázku 3.1. Klientský počítač se snaží vytvořit zabezpečené spojení se vzdáleným serverem pomocí HTTPS. Útočník zde funguje pro klienta jako výchozí brána. Jakmile klient odešle požadavek na vzdálený server, útočník ho může odchytnout a zpracovat. Útočník má také vytvořený vlastní certifikát podepsaný sám sebou[1], který pošle klientovi. Pokud je daný certifikát klientem přijat, dochází k vytvoření dvou šifrovaných spojení. Jeden je mezi klientem a útočníkem a druhý je mezi útočníkem a vzdáleným serverem mimo síť LAN. Pro klienta a vzdálený server se spojení jeví jako bezpečné, ale pro útočníka je jednoduché si rozšifrovat celou komunikaci, jelikož má přístup ke všem potřebným klíčům [2].

Existuje spoustu variant MITM útoku, ale většina z nich využívá DNS a ARP poisoning[2]. Největší nevýhoda tohoto útoku je zmanipulovat klienta, aby přijal útočníkův self-signed certifikát. Ačkoliv je v dnešní době používání počítače běžná záležitost, stále je počítačová gramotnost většiny uživatelů na nízké úrovni. Tudíž je kladen důraz především na klientské rozhraní (tzv. v našem případě je myšlen webový prohlížeč), aby důrazně klienta varovalo, že se snaží použít nevěrohodnou certifikační autoritu.

Původní pokusy vývojářů webových prohlížečů byly poměrně naivní. Pokud se uživatel dostal na nezabezpečenou stránku, tak se mu zobrazilo dialogové okno, na kterém bylo standardně označeno tlačítko pro potvrzení neznámého certifikátu. Běžné chování uživatele

[1] V angličtině self-signed certificate.

[2] King, J.; Lauerman, K.; Layer 2 Attacks and Mitigation Techniques for the Cisco Catalyst 6500. Cisco Systems, 2010 [cit. 2014-12-28], http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_603839.pdf



Obrázek 3.1: Síťová topologie při útoku MITM.

je takové, že pokud něčemu nerozumí, tak odklikne implicitní možnost, což potenciálním útočníkům velmi ulehčilo případný útok[3].

V dnešní době je hrozba MITM útoku dobře zdokumentovaná a známa. Přizpůsobily se jí i vývojáři internetových prohlížečů. V dnešní době platí, že pokud se uživatel připojí na stránku, jejíž certifikát nelze ověřit, prohlížeč mu velmi důrazně oznámí případné nebezpečí, které by mělo být dostatečně odstrašující, aby většinu neznalých uživatelů odradilo od pokračování. Ukázka varování prohlížeče Firefox viz příloha D.

MITM útok nenastává při použití kvantové kryptografie [1], jelikož obě strany můžou identifikovat, že jejích kanál byl odposlechnut a jeho integrita je narušena. Při takové situaci dochází k okamžitému ukončení nedůvěryhodného spojení.

3.2 Fiddler Chrome Add-on

Pro potřeby zjištění formální struktury Facebook protokolu byl použit nástroj Fiddler Chrome add-on (o Fiddleru více v kapitole 3.3) dostupný z Chrome Web Store ve verzi 1.0.2[4]. Jedná se o multiplatformní open source projekt[5]. Tento jednoduchý nástroj má velmi omezené použití. Dokáže zaznamenávat a filtrovat síťový provoz ve webovém prohlížeči. K tomuto účelu by mohly být použity nástroje pro vývojáře přímo v prohlížeči Chrome při zapnutí logování XML HTTP požadavků (XHR), ale ten je méně přehledný a nemá dobře vyřešené možnosti filtrování požadavků. Na obrázku 3.2 lze vidět deserializovaný JSON data protokolu Facebook s pokusnou zprávou test. Protokolu Facebook je věnována kapitola 6.

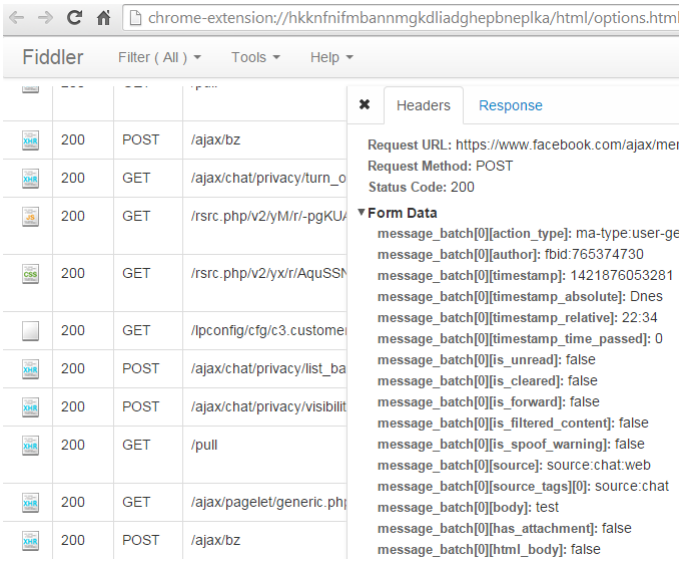
3.3 Fiddler

Pro forenzní analýzu a rekonstrukci komunikace na Facebook je podstatné, aby se daná komunikace byla odposlouchávána bez vědomí sledovaného uživatele. Z tohoto důvodu byla v laboratoři vytvořena pokusnou síťová topologie. Mezi počítač oběti a síť internet byl vložen prostředník s nainstalovaným programem Fiddler. Jedná se o volně dostupnou web proxy,

[3] MARLINSPIKE, Moxie. New Tricks for Defeating SSL In Practice. BlackHat DC Presentation on SSL Stripping, 2009. Video dostupné z: http://vimeo.com/50018478.

[4] Dostupné z: https://chrome.google.com/webstore/detail/fiddler.

[5] Na githubu dostupný z: https://github.com/welefen/Fiddler.



Obrázek 3.2: Zachycený provoz pomocí Fiddler Chrome Add-on.

která umožňuje dešifrovat provoz HTTPS na HTTP. K dešifrování využívá výše zmíněný útok MITM[6].

Fiddler odchytí požadavek na vytvoření HTTPS spojení mezi klientem na vzdáleným serverem, ale požadavek nepřepošle vzdálenému serveru. Namísto toho si s klientem vytvoří vlastní šifrovanou relaci. Dále si vytvoří šifrovanou relaci se vzdáleným serverem, na který se chce klient připojit. Klíče použité pro šifrování komunikace si můžeme následně vyexportovat a použít je k rozšifrování HTTPS provozu.

K samotnému rozšifrování byl použit nástroj pro paketovou analýzu Wireshark[7], který umožňuje při importování potřebných klíčů rozšifrovat SSL konverzaci. Následně však bylo zjištěno, že prohlížeče a servery se nejčastěji domluví na použití šifrování algoritmem Diffie-Hellman ephemereal. Tato verze není programem Wireshark podporována, tudíž nelze komunikaci rozšifrovat ani s potřebnými klíči[8]. Bylo tedy nutné najít jiný způsob, který by umožňoval specifikovat pro vytváření šifrovaného spojení algoritmus RSA.

3.4 SSLsplit

Pro účel zvolení si algoritmu, který se použije při vytváření zabezpečené HTTPS komunikace, byl použit nástroj SSLsplit ve verzi 0.4.10. Jedná se o open source nástroj pro operační systém linux, který je určený především k forenzní analýze síťového provozu[9]. Oproti jiným nástrojům umožňuje specifikovat, jaký se má použít algoritmus při vytváření klíče relace. Jádro aplikace je postaveno na variantě útoku MITM.

V laboratoři se nám následně podařilo odchytit komunikaci na sociální síti Facebook. Analýzou handshake protokolu bylo ověřeno, že k vytváření šifrovaného spojení byl skutečně použit algoritmus RSA. Dále byl z programu SSLsplit exportován privátní klíč, který byl použit na zašifrování komunikace směrem ke klientovi. Ten byl následně importován

[6] Dostupné z: http://www.telerik.com/fiddler/security-testing.
[7] Dostupné z: https://www.wireshark.org/.
[8] Dostupné z: http://support.citrix.com/article/CTX116557.
[9] Dostupné z: https://github.com/droe/sslsplit.

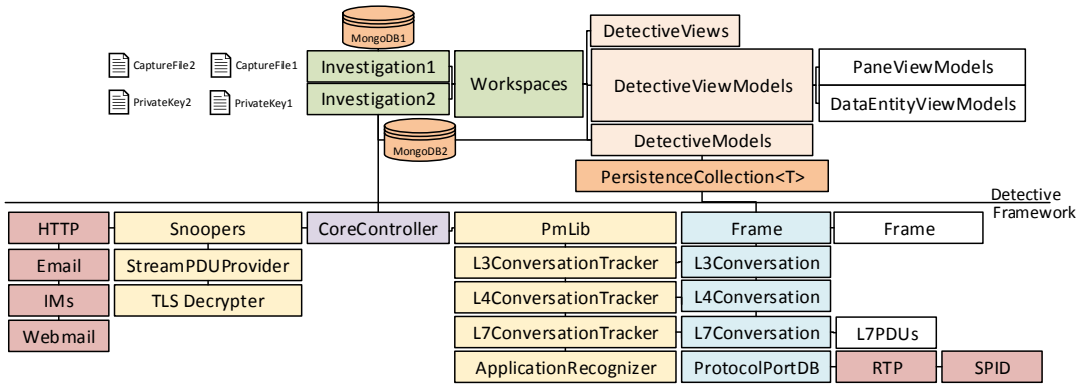
do programu Wireshark, pomocí kterého byla zachycená komunikace úspěšně dešifrována. Tímto způsobem byla splněna první část této bakalářské práce, jelikož se podařilo získat dešifrovaný HTTP provoz z původní šifrované HTTPS komunikace.

Kapitola 4

Netfox.Framework

Dle zadání bakalářské práce je nutné výslednou aplikaci implementovat do nástroj pro zpracování zachycené komunikace Netfox.Framework (NFX) [8]. Jedná se o forenzní síťový framework, který je součástí nástroje NETwork FOrensic eXtendable analysis tool (Netfox) vytvořeného z grantu projektu SEC6NET[1] na Fakultě informačních technologií Vysokého učení technického v Brně. Framework je implementován v jazyce C# zkompilovaný do dynamických knihoven a dostupný pod MIT licencí jako open source. Jeho původní verze je dostupná na CodePlex s posledním commitem z dubna 2013[2]. V aktuální verzi se jedná o placenou licenci[3], která využívá některé placené knihovny, např. Telerik UI[4].

NFX byl navržen s cílem být co nejvíce modulární, udržovatelný a s ohledem na budoucí vývoj a rozšiřování. NFX je rozdělen do vrstev, které odpovídají jednotlivým vrstvám v protokolovém zásobníku[5] nebo jsou agregovány dle funkcionality do jedné vrstvy [9].



Obrázek 4.1: Architektura a data model nástroje Netfox.Framework [10].

PmLib je knihovna poskytující rozhraní pro různé formáty zachycených dat: WiresharkTCPDump LibPcap, Microsoft Network Monitor CAP verze 2.0 a PCAP-ng verze 1.0

[1] Dostupné z: http://www.fit.vutbr.cz/research/view_project.php.cs?id=517.
[2] Dostupné z: https://netfoxframework.codeplex.com/.
[3] Dostupné z: http://www.fit.vutbr.cz/research/prod/index.php.cs?id=344&notitle=1.
[4] Dostupné z: http://www.telerik.com/aspnet-mvc.
[5] V angličtině jako network stack.

[9]. Jednotné rozhraní umožňuje manipulaci s PCAP soubory a přístup k rámcům získaným ze zachyceného provozu. Rámce jsou vyparsovány do vyšších vrstev (L2, L3 a L4) pomocí knihovny PacketDotNet podporující ethernet, PPP, IP, UDP, TCP a další. Data jsou indexována a ukládána do databáze, což umožňuje rychlou a jednoduchou práci se soubory při jejich dalším otevření bez nutnosti je znova parsovat.

Sleuths jsou komponenty nejvyšší vrstvy, které agregují informace získané extrakcí zachycené síťové komunikace. Jejich hlavní funkcionalita je obohatit vyparsovaná data o sémantický význam. Pro každý aplikační protokol existuje samostatný Sleuth modul, který rozumí sémantice konkrétního protokolu. Sleuth modul analyzuje zachycená data a následně z nich exportuje forenzně významné informace ve formě objektů, které jsou tímto připravené pro forenzní analýzu.

NFX obsahuje podporu pro velkou část nejpopulárnějších protokolů včetně HTTP, IMAP, SMTP, POP3 a XMPP [9]. Práce na tomto projektu neustále intenzivně pokračují, tudíž se i množství podporovaných protokolů neustále zvyšuje. Tato bakalářská práce se bude zabývat implementací protokolu sociální sítě Facebook do projektu NFX. Potenciální nebezpečí tohoto protokolu je, že se nejedná o schválený internetový standard a společnost Facebook ho má plně pod svou kontrolou. Lze tedy předpokládat, že v budoucnu může nastat situace, kdy dojde ke změně daného protokolu. Taková změna by zneplatnilo validitu analyzátoru tohoto protokolu a bylo by nutné implementaci modifikovat.

Netfox.Detective je další z důležitých částí nástroje Netfox. Jedná se o grafické uživatelské rozhraní pro zobrazení zpracovaných dat v NFX. Uživateli poskytuje interaktivní a snadný způsob práce s vytvořenými NFX moduly. Netfox.Detective využívá především knihovny Telerik UI[6], která nabízí spoustu komplexních tříd pro práci s uživatelským rozhraním v jazyce C#.

[6] Dostupné z: http://www.telerik.com/products/wpf/overview.aspx.

Kapitola 5

Způsoby dolování dat

V posledních deseti letech se počítače a internet stali neodmyslitelnou součástí každodenního života. S kolegy v práci komunikujeme pomocí emailu, skypu či Jabberu, úkoly a povinnosti si ukládáme do telefonu, který máme vždy při sobě, a oproti kousku papíru nám dává jistotu, že ho jen tak lehce neztratíme. Jedná se také o skvělý přísun interaktivní zábavy. Není nic jednoduššího než se ve virtuálním světě stát mocným bojovníkem počítačové hře, číst odborné či odlehčené články, sledovat naše oblíbené seriály a filmy nebo chatovat s přáteli. Tato všestrannost a jednoduchost moderních technologií měla za následek částečný přesun lidského života na sociální sítě. Sdílíme zde naše zážitky, zůstáváme v kontaktu se starými přáteli či se seznamujeme s novými lidmi.

Mezi nejpopulárnější sociální síť patří jednoznačně Facebook, který v září 2014 zaznamenal 1.35 miliardy aktivních uživatelů[1]. Statisticky pět z šesti vašich známých má Facebook účet a pravděpodobně je toto číslo reálně mnohem vyšší, pokud nemáte spoustu kamarádů ze střední Afriky či Severní Koree.

Samotná povaha sociální sítě však vybízí k možným nebezpečím. Pokud si člověk dobře nepohlídá, jaký obsah komu sdílí, může si o nás naprosto kdokoliv zjistit spoustu osobních informací. Něčím podobným se také zabývá metodologie dolování dat, kterou se budu zabývat v druhé části této bakalářské práce.

Má případová studie se bude týkat získávání dat ze sociální sítě Facebook. Vstupem aplikace pak bude Facebook účet oběti či její uživatelské ID. Výstupem budou zanalyzovaná data ve formě užitečných informací. Pokusím se vydolovat osobní informace, kde se v poslední době pohyboval, jakých událostí se zúčastnil, jakých událostí se plánuje účastnit nebo koho má v přátelích.

V dalších částech práce si představíme jednotlivé způsoby získávání dat ze sociální sítě Facebook se zaměřením na platformu .NET, ve které má být výsledná aplikace realizována.

5.1 Aplikační rozhraní pro Facebook

Graph API je oficiální aplikační rozhraní pro sociální síť Facebook. Umožnuje přístup k uživatelům, jejich příspěvkům, skupinám, událostem a mnohem více. Aktuálně ve verzi 2.2[2]. Facebook sice nabízí toto softwarové vývojové prostředí[3] zcela zdarma, bohužel je však omezeno jen na programovací jazyky JavaScript a PHP, herní engine Unity, platformu iOS

[1] Dostupné z: http://newsroom.fb.com/company-info/.
[2] Dostupné z: https://developers.facebook.com/.
[3] V angličtině Software Development Kit (SDK).

a Android. Pro účely této bakalářské práce je tedy nepoužitelné.

Facebook SDK pro .NET umožnuje vývoj aplikací pracující se sítí Facebook na desktop, web, Silverlight, Windows Phone a Windows Store[4]. Jedná se o open source projekt, který je distribuován pomocí nástroje NuGet a umožnuje vývoj pro .NET 3.5, 4.0 a 4.5[5].

Jedná se o adaptér k oficiálnímu Graph API, který momentálně není nijak omezený a implementuje plnou funkcionalitu oficiálního rozhraní. K využívání tohoto rozhraní je potřeba se zaregistrovat na stránkách Facebooku jako vývojář a vytvořit si přístupový token pro danou aplikaci.

Projekt má kvalitně zpracovanou dokumentaci s několika ukázkami kódu. Spoustu věcí však nevysvětluje a odkazuje se na dokumentaci k Graph API bez jejíchž znalosti se vývojář neobejde. V ukázkovém příkladu si ukážeme, jak pomocí tohoto rozhraní získat veřejné informace uživatele *tomas.bruckner*. Samotný dotaz viz ukázka kódu 5.1, návratová hodnota dotazu viz ukázka kódu 5.2.

```
1 var client = new FacebookClient();
2 dynamic user = client.Get("tomas.bruckner");
```

Výpis programu 5.1: Získání veřejných informací o uživateli tomas.bruckner.

Odpověď na tento dotaz je dynamický objekt, který obsahuje atributy dle vrácených JSON dat:

```
1 {
2       "Id": "123456",
3       "First_name": "Tomáš",
4       "Gender": "male",
5       "Last_name": "Bruckner",
6       "Link": "https://www.facebook.com/tomas.bruckner",
7       "Name": "Tomáš Bruckner",
8       "Username": "tomas.bruckner"
9 }
```

Výpis programu 5.2: JSON data s veřejnými informacemi o uživateli.

5.2 Selenium

Selenium je nástroj pro práci s webovými aplikacemi, především pro automatické testování webových stránek. Skládá se ze čtyř komponent[6]:

- Selenium IDE
- Selenium Grid
- Selenium Remote Control (předchůdce Selenium Webdriveru)
- Selenium WebDriver

[4] Dostupné z: http://facebooksdk.net/docs/web/getting-started/.
[5] Dostupné z: https://github.com/facebook-csharp-sdk/facebook-csharp-sdk.
[6] Dostupné z: http://www.seleniumhq.org.

Selenium IDE je kompletní vývojové prostředí pro Selenium testy, které je poskytováno jako add-on do webového prohlížeče Firefox. Umožňuje funkcionalitu k vytváření automatizovaných skriptů pro ověření platnosti libovolné webové stránky. Samotné skripty lze vytvářet manuálně či je nechat vygenerovat automaticky pomocí záznamu z relace uživatele. Vzhledem k formě distribuce *Selenium IDE* (add-on do prohlížeče) je tento nástroj nevyužitelný pro účely této bakalářské práce, kterou je potřeba implementovat v jazyce C#.

Selenium Grid je server, který umožňuje uskutečnit testy na instancích webového prohlížeče, které běží na vzdálených strojích. Jeden Selenium server slouží jako centrála, která si získá přístup do daných instancí prohlížeče od ostatních Selenium serverů. Tato Selenium komponenta je výhodná především pro snížení zátěže při testování velkých webových aplikací. *Selenium Grid* dále umožňuje spouštět testy paralelně na všech vzdálených stanicích, což zjednodušuje testování webových aplikací v různých webových prohlížečích.

Selenium WebDriver je nástupce *Selenium Remote Controlu*. Pracuje nad jádrem libovolného webového prohlížeče a je implementován do několika programovacích jazyků včetně PHP, Pythonu, Javy, Perlu, Ruby a C#. *Selenium WebDriver* posílá příkazy pro práci nad DOMem stránky, které se následně uskuteční a prohlížeč pošle výsledek zpátky isntanci Selenia. Oproti *Selenium Remote Controlu* už není potřeba vytvářet speciální server, který by prováděl dané testy[7]. *Selenium WebDriver* totiž vytváří novou instanci zvoleného prohlížeče, nad kterým následně pracuje. V roce 2012 vznikl ve spolupráci se Simonem Stewartem (hlavní vývojářem Selenium WebDriveru) a Davidem Burnsem (zástupcem společnosti Mozzila) W3C návrh standardu pro WebDriver[8].

Selenium WebDriver se jeví jako dobrá alternativa pro získávání dat ze sociální sítě Facebook především díky možnosti práce nad jednoduché práci nad DOMem HTML stránky, což by mohlo nahradit potenciální nedostatky práce s Facebook API.

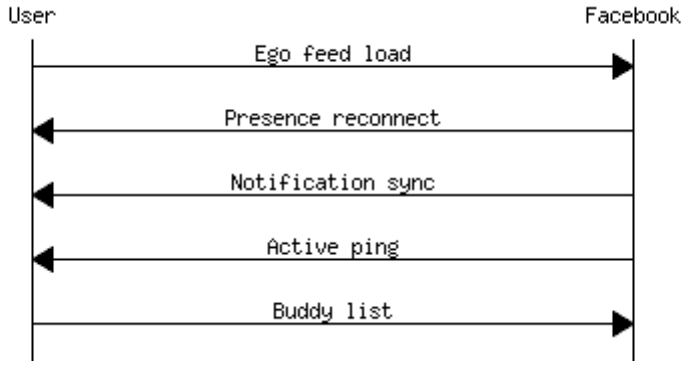
[7] Dostupné z: https://www.nuget.org/packages/Selenium.WebDriver.
[8] Dostupné z: http://www.w3.org/TR/2012/WD-webdriver-20120710.

Kapitola 6

Facebook protokol

V této kapitole byl představen analyzovaný Facebook protokol. Na příkladech byly popsány nejdůležitější zprávy. Pro názornost byly vytvořeny sekvenční diagramy průběhu zasílání jednotlivých zpráv. Pro přehlednost jsou v této kapitole uvedeny pouze diagramy pro přihlášení uživatele a pro zaslání nové zprávy. Zbylé diagramy jsou uvedeny v příloze E.

Při přihlašování registrovaného uživatele na sociální síť Facebook dochází k odeslání zprávy `EgoFeedLoad`, která stáhne příspěvky ostatních uživatelů na hlavní stránku. Server následně posílá žádost o zjištění stavu uživatele, zda je pro ostatní uživatele online či offline, pomocí zprávy `PresenceReconnect`. Dále uživateli přijde zpráva `NotificationSync`, která obsahuje všechny zatím nezobrazené notifikace. Server klientovi průběžně posílá zprávu `ActivePing`, která zjišťuje, zda instance spojení nezanikla. Analogicky si uživatel žádá zprávou `BuddyList` o aktuální informace stavu jednotlivých přátel. Průběh přihlášení lze vidět na diagramu 6.1.



Obrázek 6.1: Přihlášení

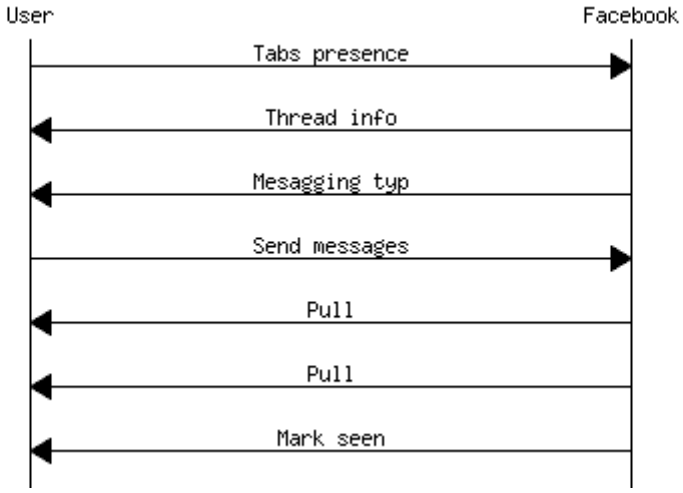
Při zakliknutí textboxu pro vytváření nového stavu se uživatel registruje na serveru zprávou `StructuredSuggestion`, že se chystá psát nový stav. Server na to reaguje průběžným zasíláním zpráv `Search`, které zjišťují aktuální obsah textboxu a případně navrhují uživateli označení místa/osob v příspěvku. Velmi zajímavá je zpráva `RecordBasicMetrics`, kterou průběžně posílá uživatel. Zde se nachází kolik přátel má daný uživatel, v kolika je přidán skupinách, kolik má pozvánek na události, kolik použil aplikací atd. Po odkliknutí se zasílá zpráva `UpdateStatus`, která obsahuje daný příspěvek.

Při přidání se do nové skupiny posílá uživatel zprávu `FetchAppColumns`, která mu stáhne příspěvky dané skupiny. Dále posílá zprávu `UpdateLikeCountDOMString`, která

aktualizuje počítadlo přidaných fanoušků dané skupiny. Server následně posílá poslední zprávu `PagesTimelineChainingPagelet`, která uživateli zobrazí nabídku, kde si může vybrat zobrazení příspěvků na stránce skupiny dle data přidání.

Pokud se uživatel rozhodne využít Facebook Messenger a zaklikne uživatele, kterému chce psát, dochází k poslání zprávy `TabsPresence`, která zaktualizuje stav daného uživatele na serveru. Pokud uživatel A píše uživateli B, tak server posílá uživatele B zprávu `MessagingTyp`, která ho informuje, že mu momentálně uživatel A píše. Samotná soukromá zpráva pak putuje v protokolové zprávě `SendMessages`. Její obsah lze nalézt v příloze E. Při úspěšném doručení uživateli dojde k poslání zprávy `Pull` směrem od serveru. V případě, že si uživatel B zprávu zobrazí, přichází klientovi A zpráva `MarkSeen`. Průběh poslání zprávy lze vidět na sekvenčním diagramu 6.2

Pokud se uživatel rozhodne libovolný svůj příspěvek či stav smazat, posílá serveru zprávu `DeleteConfirm`, což má za následek zobrazení modálního okna s potvrzením smazání. Při potvrzení dojde k poslán zprávy `Delete`.



Obrázek 6.2: Nová messenger zpráva

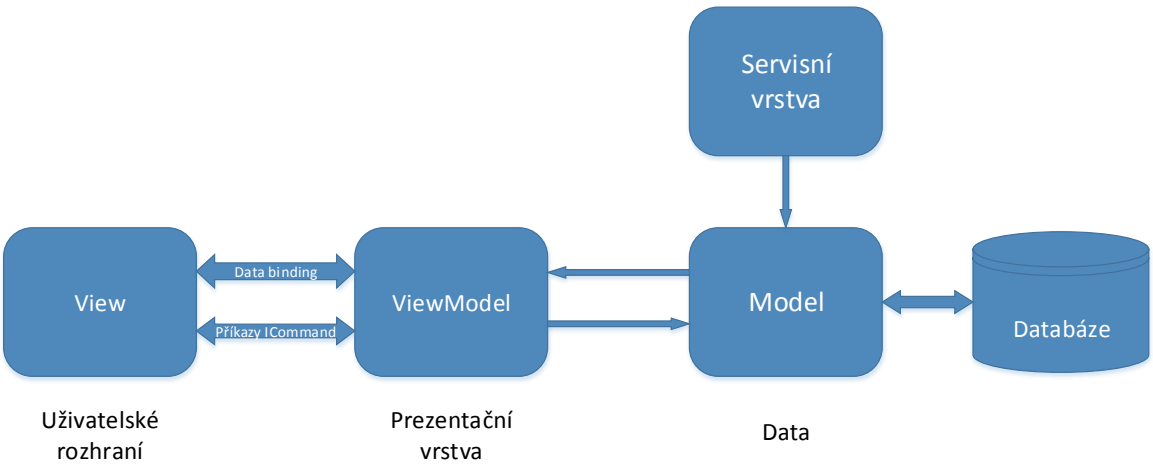
Kapitola 7

Implementace rekonstrukce

Pro implementaci aplikace byl použit návrhový vzor Model-View-ViewModel (MVVM), který rozděluje aplikaci do tří samostatných částí dle funkcionality[1]:

- Model popisující data, se kterýma aplikace pracuje
- View reprezentující uživatelské rozhraní v jazyce XAML
- ViewModel spojující Model a View a drží si stav aplikace

Tímto způsobem dochází k oddělení logiky aplikace od uživatelského rozhraní, což má za následek jednodušší vývoj, udržovatelnost a rozšiřitelnost projektu. Vzhledem ke komplexnosti a zaměření projektu Netfox.Framework bylo nutné návrhový vzor MVVM rozšířit o servisní vrstvu, ve které probíhá vyšší úroveň logiky aplikace. Schéma viz obrázek 7.1.

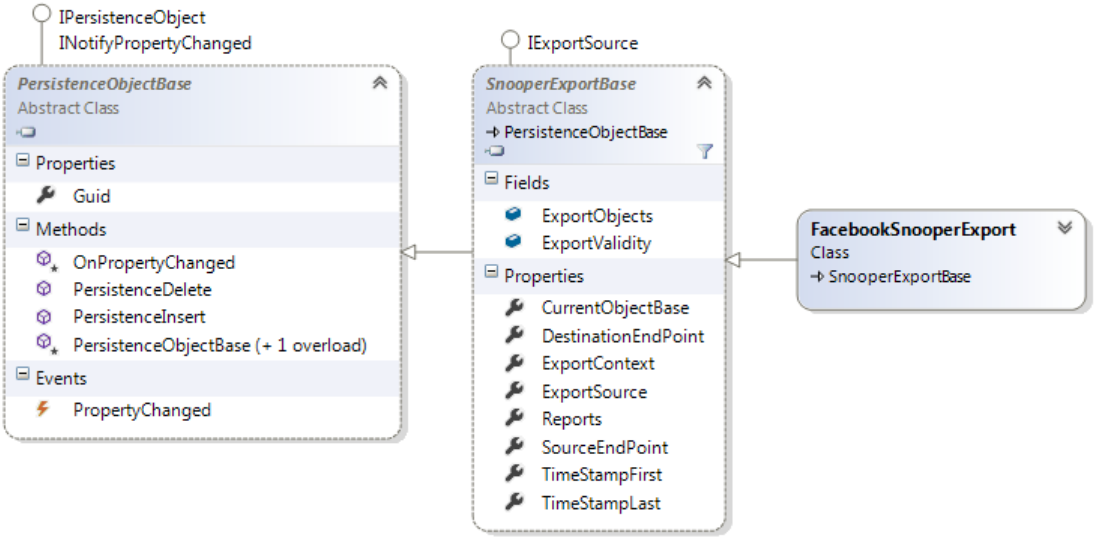


Obrázek 7.1: Schéma architektury návrhového vzoru MVVM.

[1]The MVVM Pattern dostupný z: https://msdn.microsoft.com/cs-cz/library/hh848246.aspx.

7.1 Struktura modelů Facebook objektů

FacebookSnooperExport je exportní objekt Facebook Snooperu. Dědí z *SnooperExportBase* a *PersistenceObjectBase* viz obrázek 7.2. V seznamu *ExportObjects* jsou uloženy všechny exportní Facebook objekty, které musí být potomky abstraktní třídy *SnooperExportedObjectBase*. Dále obsahuje seznam *ExportReportů*, které reprezentují informace o vzniklých nečekaných situacích, např. byla zjištěna Facebook zpráva neznámého typu nebo se nepodařilo parsování Facebook objektu.



Obrázek 7.2: Dědičnost modelu FacebookSnooperExport.

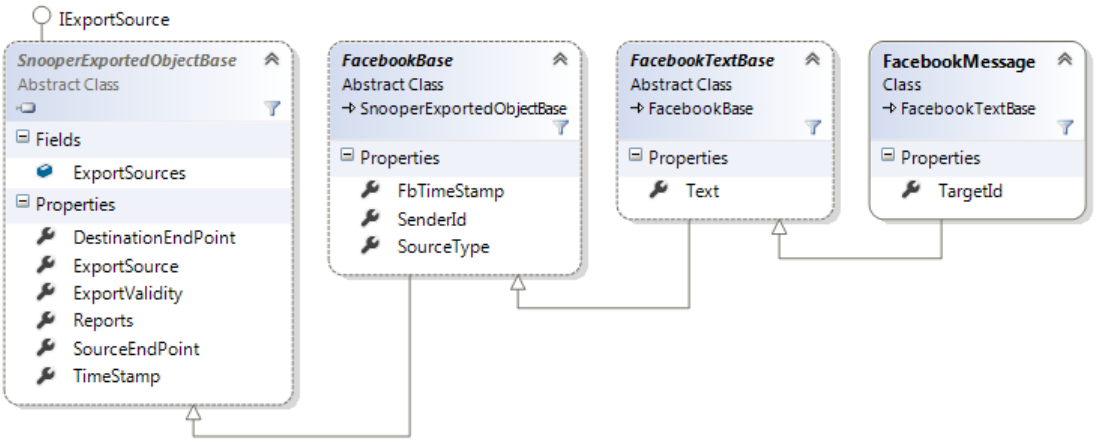
FacebookBase je abstraktní třída, která slouží jako bázová třída pro ostatní Facebook objekty. Dědí z abstraktní třídy *SnooperExportedObjectBase*, která obsahuje seznam *ExportReportů*, které slouží pro záznam o nestandardních situací při parsování konkrétního objektu, a property *ExportValidity*, která udržuje informaci, zda je validní celý objekt nebo pouze část objektu. Samotný *FacebookBase* obsahuje property *FbTimeStamp*, která je naplněna časem události, která je obsahem Facebook zprávy. Property *SenderId* je typu unsigned long, jelikož identifikační čísla nových Facebook uživatelů již přesáhly maximální rozsah datového typu unsigned integer. Poslední společná property pro Facebook objekty je string *SourceType*, která obsahuje informaci, z jaké platformy byla Facebook zpráva zaslána (web, telefon, tablet atd.).

FacebookStatus je objekt, který nedědí z *FacebookBase*, jelikož zpráva neobsahuje zdroj události (web, telefon). Implementuje property pro text statusu, čas vzniku, autora a identifikační číslo uživatele, na jehož zeď byl status poslán.

7.1.1 Textové zprávy

FacebookTextBase je abstraktní třída pro textové zprávy vyjma statusů. Dědí z třídy *FacebookBase* a obohacuje ji o string property *Text* pro textovou reprezentaci zasílané zprávy.

FacebookMessage je model pro konkrétní zprávu mezi dvěma uživateli. Obohacuje bázovou třídu o property *TargetId* reprezentující identifikační číslo uživatele, kterému byla zpráva zaslána. Podrobnější přehled lze vidět na obrázku 7.3.



Obrázek 7.3: Dědičnost modelu FacebookMessage.

FacebookGroupMessage je model pro skupinové zprávy mezi více uživateli. Každá skupinová konverzace obsahuje název skupiny, která je uložena v string property *GroupName*. Facebook identifikační čísla všech uživatelů, kteří se účastní konverzace, jsou uložena v seznamu typu unsigned long.

FacebookComment reprezentuje uživatelský komentář, kde *TargetId* je identifikační číslo uživatele, na jehož zeď byl komentář zaslán.

7.1.2 Soubory

Pomocí sociální sítě Facebook lze přeposílat různé typy souborů. Samotný Facebook aktuálně rozlišuje dva typy souborů, fotky a ostatní soubory. Inspiroval jsem se tímto dělením i při vytváření modelů pro export odchycených souborů.

FacebookFileBase je abstraktní třída, která dědí z třídy *FacebookBase*. Rozšiřuje ji o prvky společné pro všechny soubory, které se neposílají ve skupinových konverzacích. Property *TargetId* uchovává informaci o cílovém uživateli, kterému byl soubor poslán. Property *Url* udržuje odkaz na stáhnutí přeposílaného souboru. Tento odkaz je veřejný a není k němu potřeba přístupový token ani instance s přihlášeným uživatelem Facebooku, ale pouze po omezenou dobu (odkazy na soubory mají delší životnost). String property *Name* pak reprezentuje název daného souboru.

FacebookFile a FacebookPhoto jsou třídy implementující bázi *FacebookFileBase* a nijak ji již neobohacují o další funkcionalitu. Do property *Name* se ukládá skutečný název souboru, jak ho pojmenoval uživatel. V případě fotky se jedná o identifikační číslo fotky, jež ji bylo přiděleno Facebookem.

FacebookGroupFileBase je analogicky totožná třída jako *FaacebookFileBase* pro skupinové konverzace. Krom *Url* a *Name* však nabízí seznam uživatelů konverzace *Participants* namísto *TargetId*. Taky poskytuje string property *GroupName*, jelikož každá skupinová konverzace je pojmenovaná.

FacebookGroupFile a FacebookGroupPhoto jsou jednoduché modely implementující *FacebookGroupFileBase* obdobně jako *FacebookFile* a *FacebookPhoto*.

7.1.3 Události

Události[2] jsou metadata, které Facebook zasílá uživatelům, aby je informoval o různých změnách stavů nebo vynutil některou akci. Z pohledu forenzní analýzy jsem se zaměřil pouze na události, které informují o aktivní přítomnosti uživatele. Nebylo by výhodné exportovat i události, které pouze kontrolují, zda je instance prohlížeče stále otevřena (např. ping nebo fullreload), jelikož uživatel u klientské stanice již nemusí být přítomen.

FacebookEventBase je bázová třída, která dědí z *SnooperExportedObjectBase* a k modelu přidává property string property typu eventu *EventType* a uživatelské jméno *UserId*.

FacebookEventRead je model události, kterou přihlášený uživatel informuje ostatní, že si přečetl jejich příspěvky. Zpráva obsahuje pole identifikačních čísel uživatelů *TargetIds*, kterým se má tato zpráva doručit, a identifikační číslo konverzace *ThreadId*, ve které se daná událost o přečtení uskutečnila.

FacebookEventReadReceipt je událost o přečtení příspěvků jinými uživateli, než je námi sledovaný a přihlášený uživatel. Jedná se o protipól události *FacebookEventRead*. Daný model obsahuje property *ThreadId*, která identifikuje konkrétní konverzaci, *ReaderId*, nebo-li identifikační číslo uživatele, který si příspěvek přečetl, a čas události *FbTimeStamp*.

FacebookEventSentPush je událost, kterou přihlášený uživatel posílá serveru, aby ho informoval, že aktuálně píše jinému uživateli soukromou zprávu. Událost obsahuje čas *FbTimeStamp* a cílového uživatele *TargetId*.

FacebookEventTyping je událost, která informuje uživatele o tom, že mu jiný uživatel píše soukromou zprávu. Tuto událost posílá serveru přihlášenému uživateli, pokud mu píše jiný uživatel a zároveň je online. Bázový model obohacuje o *SenderId* a identifikační číslo konverzace *ThreadId*.

7.2 Servisní vrstva rekonstrukce

Hlavní jádro aplikace tvoří servisní vrstva, která se stará o parsování seznamu HTTP zpráv, ve kterých se snaží identifikovat jednotlivé Facebook objekty dle podkapitoly 7.1. Samotná vrstva je implementována v třídě *FacebookSnooper*, která je potomkem abstraktní třídy *SnooperBase*. Třída implementuje především metodu *RunBody()*, která je volána generický pro všechny potomky třídy SnooperBase pro spuštění konkrétního Snooperu. Tato metoda připravuje aplikaci pro zahájení zpracování konverzace. Následně je volána metoda *ProcessConversation()*, která zpracovává všechny exportní Snooper objekty, které mu byly předány. Každý exportní Snooper objekt obsahuje seznam *SnooperExportedObjectBase* objektů viz 7.1. Tyto objekty se zkontrolují, zda se jedná o HTTP zprávy a následně proběhne identifikace dle diagramu B.1.

Facebook používá k zasílání zpráv data typu JSON, které jsou obsahem těla HTTP objektu. Facebook dává před začátek každého JSON objektu nekonečnou smyčku ve tvaru *for(;;);*, která je navržena k přecházení použití funkce *eval* v kódu při vyhodnocování JSON dat. Také je to forma obrany proti útoku zvaném JSON hijacking[3]. Bylo tedy nutné tento kus kódu odstranit před použití knihovny na parsování JSON dat. K tomuto účelu byla použita knihovna Json.NET[4], která je nejpopulárnější dle počtu stažení v galerii balíčků NuGet.

[2] V angličtině events.

[3] Dostupné z portálu StackOveflow: http://stackoverflow.com/questions/6339790/what-does-a-ajax-call-response-like-for-json-data-mean.

[4] Dostupné z: https://www.nuget.org/packages/newtonsoft.json/.

V případě pozitivní identifikace objektu se vytvoří odpovídající třída reprezentující daný model, která se následně exportuje. Pokud se nejedná o relevantní Facebook objekt, pokračuje se ve zpracování dalšího objektu. Při neočekávané situaci dojde k vytvoření záznamů o nečekané situaci vytvořením objektu *ExportReport*, který slouží k následné analýze výsledného exportu.

7.3 ViewModel rekonstrukce

Stavu aplikace se udržuje ve ViewModelu. V případě Facebook rekonstrukce je pro tyto účely vytvořena třída *FacebookViewModel*. Třída obsahuje kolekce pro všechny modely dle kapitoly 7.1. Konstruktor třídy má jako parametry třídu *WindsorContainer*, který zajišťuje Inversion of Control, třídu *ExportVm*, která reprezentuje modely exportované pomoci servisní vrstvy, a rozhraní *IFacebookView*, které reprezentuje View pro zobrazení exportovaných objektů. Diagram *FacebookViewModel* viz obrázek C.2.

7.4 Uživatelské rozhraní rekonstrukce

Pro implementaci view byla použita knihovna Telerik, která nabízí jednoduchou práci s uživatelským rozhraním pro jazyk C#. Jednotlivé komponenty se využívají pro zobrazování v celém projektu Netfox.Detective, do kterého jsem i já implementoval svoje zobrazovací okna. Jedná se však o placenou licenci, která je k projektu draze zakoupena, tudíž tuto knihovnu nemůžu poskytnout jako volně dostupnou do závěrečného odevzdání na DVD k bakalářské práci.

Pro implementaci byla využita Telerik UI pro WPF, která nabízí intuitivní aplikační rozhraní, podporuje návrhový model MVVM, nativní podporu pro technologii *Táhni a pusť*[5] a jednoduchost testování[6]. Pro zobrazování exportovaných dat byla využita komponentu pro zobrazování dat v tabulce *RadGridView*. Nabízí mimo jiné i vestavěnou podporu pro manipulaci se sloupci jako je filtrování, agregování, přesunování, početní operace nad řádky a sloupci či data binding i s validací[7].

Jednotlivé kolekce objektů jsou zobrazeny v záložkách dle typu. Každá záložka obsahuje tabulku *RadGridView* s odpovídajícími daty. Při kliknutí na konkrétní řádek se zobrazí pod tabulkou detail objektu s podrobnějšími informacemi. Všechna data jsou z pohledu uživatele pouze pro čtení, tudíž není podporováno mazání řádku či jejich editace. Pro čistější návrh aplikace a dodržování zásad čistého kódu nebyl použit Code Behind a všechna implementace probíhala v odpovídajícím XAML souboru.

[5] V angličtině drag and drop.
[6] Dostupné z: http://www.telerik.com/products/wpf/overview.aspx.
[7] Dostupné z: http://docs.telerik.com/devtools/wpf/controls/radgridview/overview2.

Kapitola 8

Implementace dolování dat

Pro forenzní analýzu virtuálního profilu zkoumané osoby by byla v ideálním případě nejvýhodnější přímá spolupráce poskytovatelů sociální sítě Facebook. Z pohledu reálného světa se však jedná o utopii v případě běžného sledování uživatelů. Existují zde sice možnosti, jak si vyžádat kooperaci a sdílení uživatelských dat[1], ale ne vždy je možné získat soudní příkaz či je na daného uživatele vydán mezinárodní zatykač. Velmi uklidňující je informace, že Facebook dobrovolně poskytne informace, pokud se jedná o případy zneužívání dětí. Také umožňuje uživatelům nahlásit odsouzeného sexuálního delikventa, kteří mají dle podmínek použití zakázáno používat sociální síť Facebook[2].

Pro všechny ostatní případy bylo nutné implementovat jiný způsob získávání dat. Nejdříve jsem se pokusil využít Graph API viz podkapitoly 5.1. Při zaslání požadavku na seznam přátel libovolného uživatele však přicházela odpověď vždy JSON s prázdným polem přátel:

```
{
  "data": [
  ],
  "summary": {
    "total_count":0
  }
}
```

Výpis programu 8.1: Návratová hodnota JSON dat při použití Graph API

Tento zásadní problém, který bránil dalšímu pokračování při implementace, se podařil objasnit hlubším prozkoumání informací o změnách a aktualizacích Graph API na stránkách Facebook Developers. V dubnu roku 2014 byla vydána Graph API verze 2.0, která přinesla spoustu nových změn a omezení pro vývojáře[3]. Od nové verze není povoleno získávat žádné informace o uživatelích bez jejich explicitního svolení. Jakákoliv aplikace, která využívá Graph API, musí mít přihlašovací formulář pro své uživatele, ve kterém musí uživatelé potvrdit souhlas s poskytnutím požadovaných informací. Dále už také není možné přistupovat ani k seznamu přátel přihlášeným uživatelů. Každý přítel má svůj vlastní *bezpečnostní token* a tudíž lze získat jenom přátele, kteří také dali svolení dané aplikaci získávat jejich data.

Pokusil jsem se tedy prověřit způsoby použití starší a benevolentnější verze 1.0. Toto řešení však již není možné, jelikož Graph API verze 1.x můžou využívat pouze aplikace,

[1] Dostupné z: https://www.facebook.com/safety/groups/law/guidelines/.
[2] Dostupné z: https://www.facebook.com/help/473784375984502.
[3] Dostupné z: https://developers.facebook.com/docs/games/migrate.

které vznikly nejpozději 30. dubna 2014 (resp. jejich vygenerovaný přístupový token)[4]. Navíc by toto řešení nebylo ani z dlouhodobého hlediska reálné, jelikož v květnu 2015 bude kompletně ukončena podpora verze 1.x a Facebook servery již na tyto požadavky nebudou odpovídat.

8.1 Parsování HTML

Vzhledem k nemožnosti použití oficiálního API bylo nutné použít alternativního řešení. Pokud se daný obsah vyskytuje na přístupné webové stránce, pak vždy bude existovat možnost získat požadovaná data pomocí parsování specifikované HTML stránky. K tomuto účelu se jako nejvýhodnější nástroj jeví Selenium viz podkapitola 5.2.

Při implementaci však došlo k další závažné komplikaci. Facebook využívá pro renderování HTML svoji vlastní knihovnu React[5], která obfuskuje (znepřehledňuje, zatemňuje) HTML obsah metadat. Konkrétní projevy obfuskace jsou např. z hodnoty atributu ID "user_friend"se stane nečitelná hodnota "_ic48". Krom naprosté náhodnosti také dochází k časté obměně těchto hodnot, což naprosto zamezuje parsování HTML dle hodnot většiny atributů.

Lze však využít znalosti URL, která specifikuje, jaké informace můžeme očekávat na dané stránce. Pokud chceme získávat jména uživatelů, které má zkoumaná osoba v přátelích, pak je lze nalézt na URL ve tvaru https://www.facebook.com/{UZIVATEL}/friends. Dále lze využít informace, že na každého přítele existuje klikatelný odkaz, který obsahuje v části své URI uživatelské jméno a informaci *&hc_location=friends_tab*, která specifikuje, že se jedná o odkaz na přítele. Lze tedy vyparsovat všechny odkazy na dané stránce a regulárním výrazem z nich dostat námi potřebnou informaci o jménech všech přátel uživatele.

Dalším znepříjemněním dolování je skutečnost, že Facebook využívá moderní způsob posunování pomocí tzv. *infinite scrollu*. Jedná se o jednu z implementací *lazy loading*, kdy se data dotahují dynamicky až v případě, že jsou opravdu potřeba. Pro seznam přátel to tedy znamená, že se zobrazí omezený počet přátel a až při najetí na konec seznamu se pošle pomocí technologie AJAX požadavek na server o zaslání dalšího množství přátel. Je tedy potřeba vyřešit problém, jak dlouho má Selenium posunovat zobrazovací plochu prohlížeče, než budou načteny všechny informace. Tento problém byl vyřešen kontrolou změny počtu elementů seznamu, jelikož si každý AJAX dotaz stáhne nový seznam přátel (nikoliv prvky seznamu, jak by se dalo očekávat), tak pokud se při najetí na konec stránky již nezmění počet seznamů na stránce, můžeme předpokládat, že již jsou zobrazeni všichni přátelé.

8.2 Modely pro dolování dat

FacebookMiningBase je abstraktní třída, která zastřešuje další modely. Při dolování dat se vždy budeme dotazovat na konkrétního uživatele, tudíž je zde vytvořena property *UserId* typu string. Jelikož budeme data dolovat pomocí nástroje Selenium na základě odkazů na jiný Facebook obsah, bylo nutné přidat string property *Url*. V poslední řadě bude každá informace obsahovat jméno (např. události, uživatele, alba, místa atd.), kterou budeme ukládat do string property *Name*.

FacebookPastEvent a FacebookUpcomingEvent jsou modely událostí, kterých se uživatel účastnil či se bude účastnit. Jedná se o potomky třídy *FacebookMiningBase*,

[4]Dostupné z: https://developers.facebook.com/docs/games/migrate.
[5]Dostupné z: https://facebook.github.io/react/.

kde *Name* je reprezentace jména konkrétní události, *Url* je odkaz na Facebook událost.

FacebookRecentPlace je model místa, kde se sledovaný uživatel v poslední době pohyboval. Může se jednat jak o města (Brno, Praha, ...), lokality (národní parky, památky, ...), podniky (restaurace, hotely, kluby, ...) a další. Vždy se jedná o místo, které má Facebook stránku, na kterou je odkaz v property *Url*. *FacebookRecentPlace* dědí z abstraktní třídy *FacebookMiningBase* a nijak ho nerozšiřuje o žádné nové property.

FacebookAlbum je poslední model pro dolování dat, který implementuje abstraktní třídu *FacebookMiningBase*, který třída *FacebookAlbum* neobohacuje a pouze využívá property z předka. *Name* zde reprezentuje název alba, *Url* pak odkaz na album.

FacebookPublicInfo model pro ukládání veřejných informací o uživateli. Model obsahuje property *Id*, což je identifikační číslo Facebook uživatele. Jedná se o typ *unsigned long*, jelikož nový uživatelé už přesáhli rozsah datového typu *unsigned integer*. Dále je zde textová reprezentace pohlaví v property *Gender*, uživatelského jména *Username*, skutečného jména *Name*, odkaz na uživatelův účet *ProfileLink* a uživatelem zvolený jazyk prostředí *Locale*.

FacebookFriendList reprezentuje seznam všech přátel zvoleného uživatele. String property *Username* vyjadřuje uživatelské jméno osoby, o které zjišťujeme informace, a list typu string *Friends* je seznam všech přátel daného uživatele.

FacebookMutualFriends je model reprezentující společné přátele zvolených uživatelů. Obsahuje dva seznamy typu string. V *Participants* jsou všichni uživatelé, jichž se společní přátelé týkají. V seznamu *MutualFriends* jsou společní přátelé daných uživatelů.

FacebookProfilePicture je poslední model pro dolování dat. Obsahuje unsigned long property *UserId* a string *PictureUrl* s odkazem na profilovou fotku uživatele.

8.3 Servisní vrstva

Hlavní jádro modulu pro dolování dat tvoří servisní vrstva, která je reprezentována třídou *SnooperFacebookMining*. Vytváří si instanci Selenium WebDriveru5.2 pro jádro prohlížeče PhantomJS[6], který je navržen jako singleton[7]. Samotný proces dolování dat pak probíhá tak, že se Selenium připojí na Facebook pod zvoleným účtem. Tento účet je modifikovatelný pomocí property *Email*, který reprezentuje přihlašovací jméno uživatele, a hesla v property *Password*. Výchozí hodnota je nastavena na účet vytvořený speciálně pro Netfox.

Následně si uživatel volá jednotlivé metody, které získávají konkrétní relevantní data. Získávat lze všechny informace popsané v podkapitole 8.2. Jednotlivé metody pro parsování přejdou pomocí Selenia na stránku s požadovanými informacemi, kde prohledávají DOM. Facebook využívá k zobrazování dat infinity scroll, tudíž je potřeba načíst všechna data. Tento problém je řešen pomocí cyklu, který pošle Seleniu javascriptový příkaz na posunutí se na konec stránky. Následně se vlákno na definovaný okamžik uspí a po probuzení zkontroluje, zda se změnil počet parsovaných prvků přes css selector[8]. Pokud ano, provede další iteraci, v opačném případě vyparsuje všechny získané elementy a získá z nich relevantní data.

Toto řešení je silně závislé na kvalitě připojení k internetu a aktuálním zatížení Facebook serverů. Také může dojít k situaci, kdy se Facebook rozhodne změnit URI k informacím nebo samotné rozložení elementů, což by znamenalo nutnou modifikaci těchto metod. Samotný diagram třídy *FacebookDataMining* viz obrázek C.1.

Pro účely zjištění společných přátel vznikla metoda, která porovnává dvě třídy *FacebookFriendlist*. V obou třídách hledá společné uživatele, kteří jsou následně uloženy do třídy

[6] Dostupný z: http://phantomjs.org/.
[7] Dostupný z: https://msdn.microsoft.com/en-us/library/ff650316.aspx.
[8] Dostupný z: http://www.w3schools.com/cssref/css_selectors.asp.

FacebookMutualFriends. Jedná se o přetíženou metodu, která zvládá i porovnání dvou již vytvořených tříd *FacebookMutualFriends*.

8.4 ViewModel dolování dat

Pro dolování dat vznikl ViewModel, který udržuje aktuální stav aplikace. Jedná se o třídu *FacebookMiningViewModel*, která obsahuje kolekce jednotlivých objektů dle podkapitoly 8.2. Pro kolekce byla použita Telerik třída *RadObservableCollection*, která obohacuje systémovou *ObservableCollection* pro kvalitnější implementaci s knihovnou uživatelského rozhraní Telerik.

Při implementaci bylo nutné zvolit mezi dvěma možnými způsoby návrhu ViewModelu. Buď vytvořit jeden ViewModel, který bude zaštiťovat celý exportní Facebook objekt spolu se všemi typy modelů, které obsahuje. Druhou možností bylo rozdělit tento ViewModel do samostatných tříd podle typu objektu dle rozdělení z podkapitoly 8.2. Pro účely tohoto projektu bylo vhodnější využít jeden hlavní ViewModel, především kvůli vztahu mezi objekty. Jeden běh dolování dat bude získávat data o jednom uživateli, které se následně i hromadně zobrazí v uživatelském rozhraní. Nebylo by tedy vhodné tuto celistvost porušovat rozdělování do více tříd, jelikož se vždy bude pracovat se všemi typy objektů.

8.5 Uživatelské rozhraní dolování dat

Uživatelské rozhraní pro dolování dat bylo silně inspirováno uživatelským rozhraním rekonstrukce komunikace viz 7.4. Jedná se o jednoduché okno, které je rozděleno do záložek podle typu vydolovaných objektů viz 8.2. Pro samotná tabulku byla použita placená knihovna Telerik, kde kvůli své licenci nemůže být součástí DVD této bakalářské práce. Pro funkčnost výsledného řešení je tedy nutné si knihovnu Telerik obstarat jiným způsobem.

Pro implementaci byla použita třída *RadGridView*, která nabízí spoustu vestavěných funkcí pro práci se sloupci v tabulce. Mezi nejpoužívanější patří filtrování, řazení a přesunování sloupců a řádků tabulky.

Kapitola 9

Porovnání výsledků výsledné aplikace

Validace je jeden z nejdůležitějších úkonů při vývoji aplikace. Bez ověření funkčnosti implementovaného řešení je naprosto nemyslitelné, že by se výsledný projekt použil v reálném světě. Pro potřeby analýzy a testování bylo nutné vytvořit sady jednotkových testů, které důsledně ověří platnost jednotlivých částí implementovaných modulů.

9.1 Porovnání rekonstrukce

Pro potřeby ověření správnosti rekonstrukce komunikace exportu Facebook objektů byla vytvořena adekvátní laboratorní data s komunikací na sociální síti Facebook. Zaměřil jsem se na vytvoření několika menších souborů, pomocí kterých se dají jednotlivé typy Facebook objektů izolovat a otestovat zvlášť. Soubory s jejich popisem viz tabulka 9.1.

Název souboru	Popis souboru
fb_comment.pcapng	Přidání komentáře
fb_file.pcapng	Poslání textového souboru test.txt mezi dvěma uživateli
fb_group.pcapng	Skupinová konverzace mezi třemi uživateli
fb_groupfile.pcapng	Skupinové poslání textového souboru test.txt
fb_groupphoto.pcapng	Skupinové poslání fotky
fb_chat.pcapng	Konverzace mezi dvěma uživateli
fb_photo.pcapng	Poslání fotky mezi dvěma uživateli
fb_status.pcapng	Přidání nového statusu
fb_pk.pem	Privátní klíč použit pro šifrování síťového provozu

Tabulka 9.1: Přehled jednotlivých souborů pro testování rekonstrukce.

K odchytávání byly použity dva virtuální stroje (oběť a útočník) s operačním systémem Ubuntu ve verzi 14.04 LTS. Na virtuálním stroji útočníka bylo potřeba:

1. Nainstalovat program SSLsplit z kapitoly 3.4
2. Vytvořit privátní klíč, na základě kterého se budou generovat podvrhnuté certifikáty programem SSLsplit
3. Vytvořit privátní klíč na generaci self-signed certifikátu, který se použije k podepisování dalších podvrhnutých certikátů

4. Nastavit přeposílání paketů přes virtuální stroj útočníka s přesměrováním HTTP a HTTPS provozu na porty, kde chceme, aby naslouchal program SSLsplit

5. Spustit SSLsplit s vynucením algoritmu RSA z kapitoly 2.2.1

Na virtuálním stroji oběti bylo potřeba:

1. Nastavit výchozí bránu na IP adresu útočníka

2. Přidat mezi důvěryhodné certifikáty self-signed certifikát vytvořený útočníkem

Jméno uživatele	Facebook ID uživatele
Uživatel A	765374730
Uživatel B	100007717846239
Uživatel C	100009331491476

Tabulka 9.2: Přehled uživatelů použitých při testování.

Pro potřeby vytvoření testových dat byli použiti tři uživatelé viz tabulka 9.2. Jako vzorová data pro konverzaci mezi dvěma uživateli byla použita následující konverzace:

UŽIVATEL A: Lorem ipsum dolor sit amet
UŽIVATEL B: mea quis doming nemore ad
UŽIVATEL A: an graece meliore has
UŽIVATEL B: vis cu dolorem minimum omittam
UŽIVATEL A: brute graece platonem usu ea
UŽIVATEL B: ius vero consulatu complectitur ad
UŽIVATEL A: id usu detracto iracundia euripidis

Pro jednoduché odlišení obsahu konverzace byl vytvořen odlišný dialog pro skupinovou konverzaci skládající se z citátů Malého prince.

UŽIVATEL B: And now here is my secret, a very simple secret: It is only with the heart that one can see rightly; what is essential is invisible to the eye.
UŽIVATEL C: You're beautiful, but you're empty.... No one could die for you.
UŽIVATEL A: Only the children know what they are looking for.
UŽIVATEL C: For the travelers the stars are guides. For others they are nothing but tiny lights.
UŽIVATEL A: What makes the desert beautiful, said the little prince, is that somewhere it hides a well.
UŽIVATEL B: It is such a mysterious place, the land of tears.

Soubory s odchycenými daty dle tabulky 9.1 byly následně analyzovány modulem pro rekonstrukci dat ze sociální sítě Facebook. Porovnáním výstupů exportního modulu s laboratorními daty bylo zjištěno, že byly správně identifikovány všechny Facebook objekty nacházející se v odchytaných datech.

9.2 Porovnání dolování dat

Porovnání výsledků dolování dat je z pohledu jednotkových testů náročnější záležitost, než porovnání výsledků rekonstrukce. Pro dolování dat se nedají odchytit data, které lze následně jednoduše testovat i bez připojení k internetu. Taky není problém spustit jakýkoliv test o rok později a stále budou vycházet stejné výsledky. V případě dolování dat je nutné si uvědomit, že jednotlivé metody získávají vždy aktuální informace. Nelze tedy zabránit změnám laboratorních dat. Například informace o událostech, kterých se uživatel hodlá zúčastnit, se automaticky přesunou do uplynulých událostí, jakmile tato událost skončí. Také profilová fotka nebo nedávno navštívené místa jsou problémové místa na vytvoření jednotkových testů.

Z toho důvodu nejsou vytvořené jednotkové testy plně automatizované. Je nutná kooperace uživatele, který musí výsledky dolování dat manuálně ověřit, zda se jedná o očekávaný výsledek a je silně nedoporučeno tyto testy spouštět jako automatizované. Dále bylo potřeba vytvořit laboratorního uživatele, přes kterého bude dolování probíhat. Facebook je v tomto ohledu velmi striktní a nepovoluje vytvářet více než jeden osobní účet. Také nedovoluje samotné dolování dat[1]. Při vytváření účtu se ve dvou ze třech případů stalo, že Facebook sám odhalil, že se jedná o duplicitní účet a nepovolil ho vytvořit. Pomocí virtuálního stroje připojeného přes VPN se nakonec povedlo vytvořit jeden účet s identifikačním číslem 100007717846239, který tuto kontrolu oklamal.

Výhoda použití tohoto účtu spočívá v množství informací, které se dají dolovat. Bezpečnostní politika sociální sítě Facebook neposkytuje žádné forenzně relevantní informace nepřihlášeným uživatelům. Naproti tomu přihlášený uživatel již má přístup k veřejným informacím ostatních uživatelů, i když nejsou vzájemní Facebook přátelé.

Jednotkovými testy byla prokázána funkčnost implementovaných nástrojů pro dolování dat. Výstupy modulu pro dolování dat odpovídaly vytvořeným laboratorním datům. Při analýze však bylo zjištěno, že výsledky jsou velmi závislé na kvalitě připojení a aktuálním zatížení Facebook serverů. Nejslabším místem implementace je vypořádání se s interaktivním nekonečným posuvníkem[2]. Implementace dolování dat pošle požadavek o další data a následně čeká určitý časový okamžik, jestli server odpoví zasláním dalších dat. V laboratorním prostředí se ukázalo, že během večerních hodin může odpověď trvat i několik desítek vteřin. V takovém případě aplikace předpokládá, že již žádná další data neexistují a ukončí dolování. Tento problém lze vyřešit nastavením delšího intervalu čekání na odpověď serveru.

[1] Dostupné z: https://www.facebook.com/legal/terms.
[2] V angličtině infinite scroll.

Kapitola 10

Výkonnostní analýza implementace

Výkonnostní analýza je v počítačové vědě spuštění počítačového programu, skupiny programů nebo jejich částí za účelem zhodnocení relativního výkonu modulu, objektu či funkce. K uskutečnění validní analýzy je nutné mít jednotný vzorek dat, vůči kterému se budou jednotlivé části aplikace zkoumat.

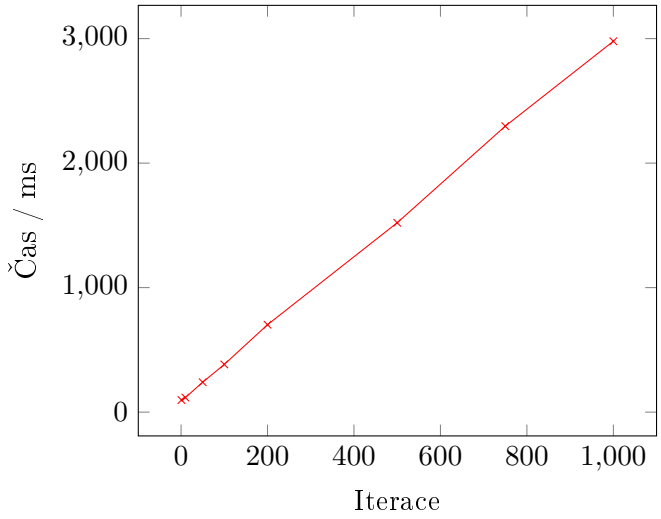
10.1 Analýza rekonstrukce

Pro účely analýzy rekonstrukce komunikace byla využita data vytvořená v kapitole 9. Konkrétně byl využit soubor *fb_chat.pcapng* viz tabulka 9.1, který obsahuje 51 HTTP zpráv a 17 Facebook zpráv. Tento soubor byl v předem určeném počtu iterací analyzován vytvořeným modulem pro rekonstrukci. Tímto způsobem bylo možné ověřit, že analýza souboru a následný export Facebook zpráv probíhá v přiměřeném množství času. V případě, že by se aktuální implementace ukázala jako pomalá, bylo by nutné zjistit časově náročná místa aplikace a navrhnout vhodnou optimalizaci. Aplikace musí pracovat především správně, ale také i dostatečně rychle, aby ji bylo možné využít ve výsledné aplikaci pro účely použití ministerstvem vnitra.

Při výkonnostní analýze bylo vytvořeno osm variant testů dle počtu iterací. Každý test se následně spouštěl pětkrát, aby se omezil vliv výkyvů při měření a zpřesnila se průměrná doba trvání testů. V tabulce 10.1 lze vidět konkrétní naměřené hodnoty pro jednotlivá spuštění. Všechny hodnoty jsou uvedeny v milisekundách. Následně byly spočítány průměrné hodnoty, které byly zaznamenány do grafu 10.1. Vytvořená implementace má lineární složitost a rychlost zpracování dosahuje požadovaných výsledků.

Iterace	Test 1 [ms]	Test 2 [ms]	Test 3 [ms]	Test 4 [ms]	Test 5 [ms]	Průměr [ms]
1	113	87	79	107	99	97
10	135	107	108	108	126	117
50	219	218	254	259	248	240
100	358	419	420	356	364	383
200	780	659	640	777	652	702
500	1 484	1 579	1 545	1 484	1 513	1 521
750	2 331	2 249	2 412	2 272	2 220	2 297
1 000	3 028	2 836	3 198	2 937	2 898	2 979

Tabulka 10.1: Výsledky měření výkonnostní analýzy.

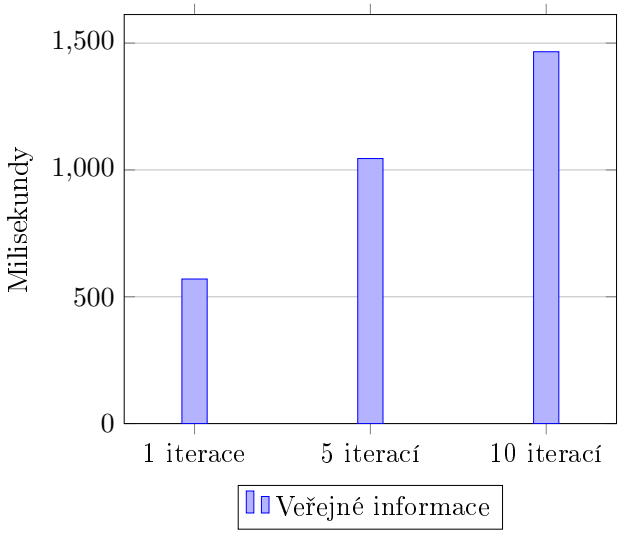


Obrázek 10.1: Graf závislosti času na počtu zpracovaných dat rekonstrukce.

10.2 Analýza dolování dat

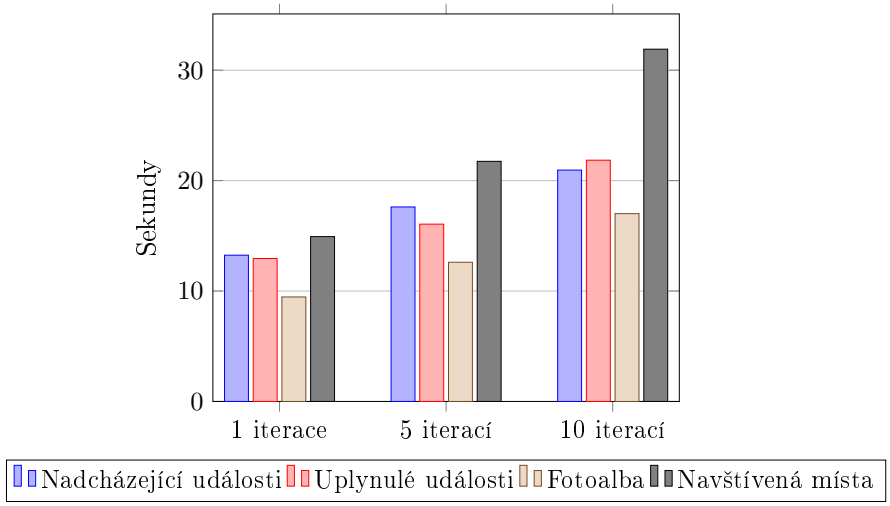
Pro analýzu modulu na dolování dat ze sociální sítě Facebook byl využit uživatel vytvořený v laboratorních podmínkách viz podkapitola 9.2. Následně byly spouštěny jednotlivé metody pro dolování dat v předem určeném počtu iterací. Všechny testy byly spuštěny s jedním, pěti a deseti opakováními. Všechny testy byly spouštěny pětkrát, aby se zjistila průměrná hodnota a omezil se vliv odchylek při měření.

Metody pro získávání veřejných informací se projevily jako výrazně nejrychlejší, jelikož se tyto informace nezjišťují pomocí parsování HTML, ale přímo z Facebook API. Na obrázku 10.2 jsou zobrazeny průměrné naměřené hodnoty vůči počtu opakování.



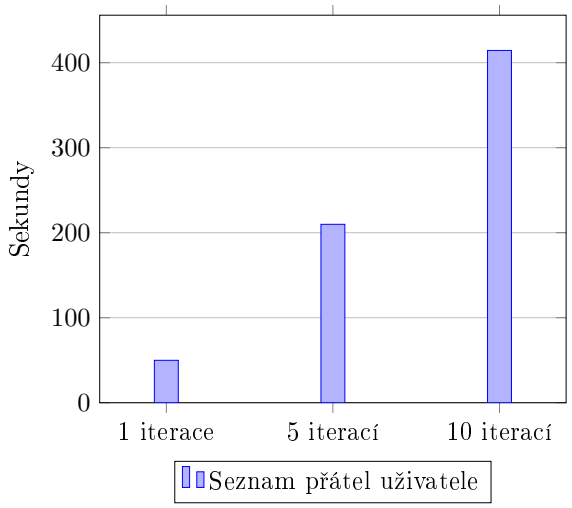
Obrázek 10.2: Graf zavislosti času na počtu opakování při dolování veřejných informací uživatele.

Dále byly analyzovány metody pro dolování informací o nadcházejících událostech, uplynulých událostech, navštívená místa a fotoalba uživatele. V laboratorním prostředí bylo vytvořeno 8 nadcházejících událostí, 3 uplynulé události, 1 navštívené místo a 8 fotoalb předem připraveného uživatele. Výsledky měření jsou uvedeny v obrázků 10.3. Nejpomaleji se jeví dolování informací o nedávno navštívených místech, což je zapříčiněno komplexností stránky, ze které se data získávají. Na dané webové stránce se nachází interaktivní mapa s označením daných míst.



Obrázek 10.3: Graf zavislosti času na počtu opakování při dolování pomocí parsování HTML.

Poslední byla analyzována část modulu pro dolování přátel uživatele. Se souhlasem byl dolován účet anonymního dobrovolníka, který měl 208 přátel. Výsledky měření jsou zaneseny do grafu na obrázku 10.4. Jedná se o nejpomalejší část implementace, což je zapříčiněno především větším množstvím dat. Počet přátel má většina uživatelů vyšší, než je počet jejich fotoalb či počet událostí, kterých se hodlají účastnit.



Obrázek 10.4: Graf závislosti času na počtu opakování při dolování přátel uživatele.

Během analýzy se ukázalo, že rychlost dolování dat je silně ovlivněna kvalitou internetového připojení a zatížením Facebook serverů. Nejvyšší zátěž bývá nejčastěji ve večerních hodinách, kdy se dolování dat může až několikanásobně zpomalit. Výsledky měření této bakalářské práce jsou získány v odpoledních hodinách, kdy byla zátěž serverů průměrná. Nejnižší zátěž pak byla pozorována v brzkých ranních hodinách.

Celkově lze implementované řešení považovat za úspěšné. Vydolování všech požadovaných informací o specifikovaném uživateli zabere řádově několik minut. Při dolování dat ze sociální sítě Facebook je nutné vzít v úvahu zatížení Facebook serverů v průběhu dne. Pro budoucí nasazení aplikace je silně doporučeno dolovat data v brzkých ranních hodinách, kdy je zatížení nejmenší.

Kapitola 11

Závěr

V této práci jsem se zaměřil na prozkoumání sociální sítě Facebook z pohledu forenzní analýzy. Cílem bylo vytvořit nástroj, který by zvládal rekonstrukci komunikace a dolování dat z této sociální sítě. Výsledek bakalářské práce byl implementován jako modul do nástroje pro zpracování zachycené komunikace Netfox.Framework, který je vyvíjen na Fakultě informačních technologií Vysokého učení technického v Brně ve spolupráci s ministerstvem vnitra České republiky.

Z pohledu rekonstrukce bylo nutné se zaměřit na forenzně relevantní data. Při analýze rekonstruované komunikace se exportují především uživatelské zprávy, soubory, komentáře a statusy. Z obrovského množství metadat posílajících se na pozadí byly vybrány především ty, které nějakým způsobem signalizují aktivitu uživatele při počítači. Zejména tedy informace o přečtení uživatelské zprávy nebo informaci o psaní zprávy jinému uživateli.

Toto řešení bylo podrobeno důslednému testování k ověření validity řešení na laboratorních datech. Také byla provedena výkonnostní analýza výsledného řešení, která prokázala schopnost využití daného řešení k reálným forenzním účelům. Zpracování a export i stovek souborů probíhá ve vteřinách. Hlavní nevýhoda výsledného řešení je vazba na Facebook protokol. Jedná se o protokol, který má plně v rukách komerční společnost, která se může rozhodnout ho kdykoliv změnit. Pokud by se jednalo o standard, nemohlo by k takové situaci dojít. Další nevýhoda spočívá v tom, že všechna komunikace na sociální síti Facebook probíhá na zašifrovaném kanálu. Aby bylo možné získat dešifrovanou komunikaci, je potřeba získat odpovídající privátní klíč, kterým byla komunikace zašifrována.

Z pohledu dolování dat bylo zjištěno, že oficiální Facebook API je od dubna 2014 naprosto nepoužitelné. Pro získání jakýchkoliv informací prostřednictvím Facebook API je nutný explicitní souhlas sledovaného uživatele. V opačném případě vrací Facebook na zasílané dotazy jenom prázdné odpovědi. Pro forenzní analýzu je toto řešení nevyhovující, jelikož nelze předpokládat souhlas osob s jejich sledováním. Také lze předpokládat, že sledovaný uživatel by se pokusil ze svého profilu odstranit všechny forenzně užitečné data, aby tyto citlivá data ochránil před uniknutím do nepravých rukou.

Tento problém byl řešen pomocí nástroje Selenium, který slouží primárně k automatickému testování webových aplikací. Tento nástroj nám umožňuje přihlášení se pod speciálně vytvořeným uživatelem do sociální sítě Facebook. Následně dojde k přechodu na stránku s profilem sledovaného uživatele a vyparsování užitečných informací přímo z HTML stránky. Také bylo nutné udělat rozbor získatelných informací a rozhodnout, které jsou užitečné pro účely forenzní analýzy. Zaměřil jsem se především na události, kterých se sledovaný uživatel účastnil nebo se hodlá účastnit, na seznam jeho kamarádů a na místa, kde se v poslední době pohyboval. Výsledné řešení také umožňuje možnost zjištění společných přátel

více uživatelů a veřejných informací o uživateli. Jednotkovými testy byla potvrzena validita jednotlivých částí modulu pro dolování dat.

Největší nevýhoda výsledného modulu spočívá ve vazbě na HTML stránku. Existuje možnost, že v budoucnu dojde ke změně v elementech webové aplikace, což by znamenalo nutnost ručně modifikovat odpovídající metody daného modulu. Další nevýhodou je, že si uživatel může většinu svého obsahu nastavit jako soukromý, což by zamezilo přístupu k těmto datům.

Další práce na modulu pro rekonstrukci a dolování dat bude zaměřena především na udržování funkčnosti jednotlivých funkcí v průběhu času. Ve specifikovaných časových interval bude nutné zkontrolovat změny ve struktuře sociální sítě Facebook a modifikovat výsledné řešení, aby odpovídalo aktuálnímu stavu.

Literatura

[1] Bennet, C. H.; Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. *Proceedings of IEEE International Conference on Computers, Systems and Signal Processing*, ročník 175, 1984: str. 8, http://www.cs.ucsb.edu/~chong/290N-W06/BB84.pdf.

[2] Callegati, F.; Cerroni, W.; Ramili, M.: Man-in-the-middle attack to the HTTPS protocol. *IEEE Security and Privacy*, ročník 7, č. 1, 2009: s. 78–81, http://www.researchgate.net/publication/220496617_Man-in-the-Middle_Attack_to_the_HTTPS_Protocol/file/e0b495231a356171f5.pdf.

[3] Freier, A.; Karlton, P.; Kocher, P.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101, RFC Editor, August 2011, http://www.rfc-editor.org/rfc/rfc6101.txt.
URL http://www.rfc-editor.org/rfc/rfc6101.txt

[4] Katz, J.; Lindell, Y.: *Introduction to Modern Cryptography*. Taylor and Francis Group, 2008, 534 s., iSBN 978-1-58488-551-1.

[5] Mulazzani, M.; Huber, M.; Weippl, E.: Social Network Forensics : Tapping the Data Pool of Social Networks. *IFIP WG 11.9 International Conference on Digital Forensics*, ročník 8, 2012.

[6] Newman, C.: Using TLS with IMAP, POP3 and ACAP. RFC 2595, RFC Editor, June 1999, http://www.rfc-editor.org/rfc/rfc2595.txt.
URL http://www.rfc-editor.org/rfc/rfc2595.txt

[7] Opplinger, R.: *SSL and TLS: Theory and Practice*. Boston: Artech House, 2009, 257 s., iSBN 978-1-59693-447-4.

[8] Pluskal, J.: *Framework for Captured Network Communication Processing*. Diploma thesis, FIT VUT v Brně, 2014, https://www.fit.vutbr.cz/study/DP/DP.php.cs?id=16748.

[9] Pluskal, J.: NetFox.Framework - The network forensic extandable analysis tool. In *Proceedings of the 20th Conference STUDENT EEICT 2014 Volume 2*, Brno University of Technology, 2014, ISBN 978-80-214-4923-7, s. 280–282.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10692

[10] Pluskal, J.; Matoušek, P.; Ryšavý, O.; aj.: Netfox Detective: A tool for advanced network forensics analysis. In *Proceedings of the Conference Security and Protection of Information*, Brno University of Defence, 2015, ISSN 2336-5587.

[11] Rescorla, E.: HTTP Over TLS. RFC 2818, RFC Editor, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt.
URL http://www.rfc-editor.org/rfc/rfc2818.txt

Příloha A

Obsah DVD

Přiložené DVD nosič obsahuje:

- Text bakalářské práce ve formátu PDF
- Zdrojové soubory bakalářské práce pro systému LaTeX
- Zdrojové soubory aplikace
- Soubor Readme.txt
- Sada testovacích dat

Příloha B

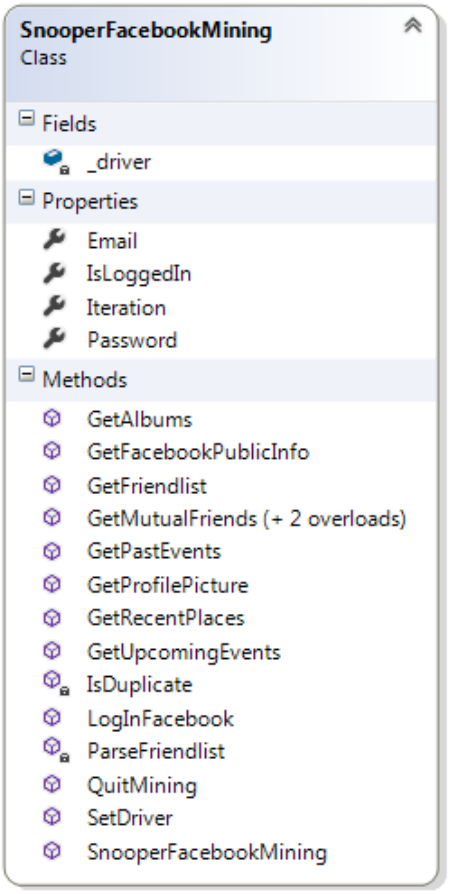
Flowchart identifikace Facebook objektu

Z důvodu snížení komplexnosti diagramu byly některé stavy vypuštěny. Vyšší úroveň abstrakce nám poskytuje jasnější pohled na danou problematiku a jednodušší pochopení analyzátoru Facebook objektů. Vynechány byly také stavy vytvoření reportů při výskytu nečekané události (např. neznámý typ eventu, chybějící data apod.) z důvodu omezení se na množinu hlavních prvků funkcionality programu.

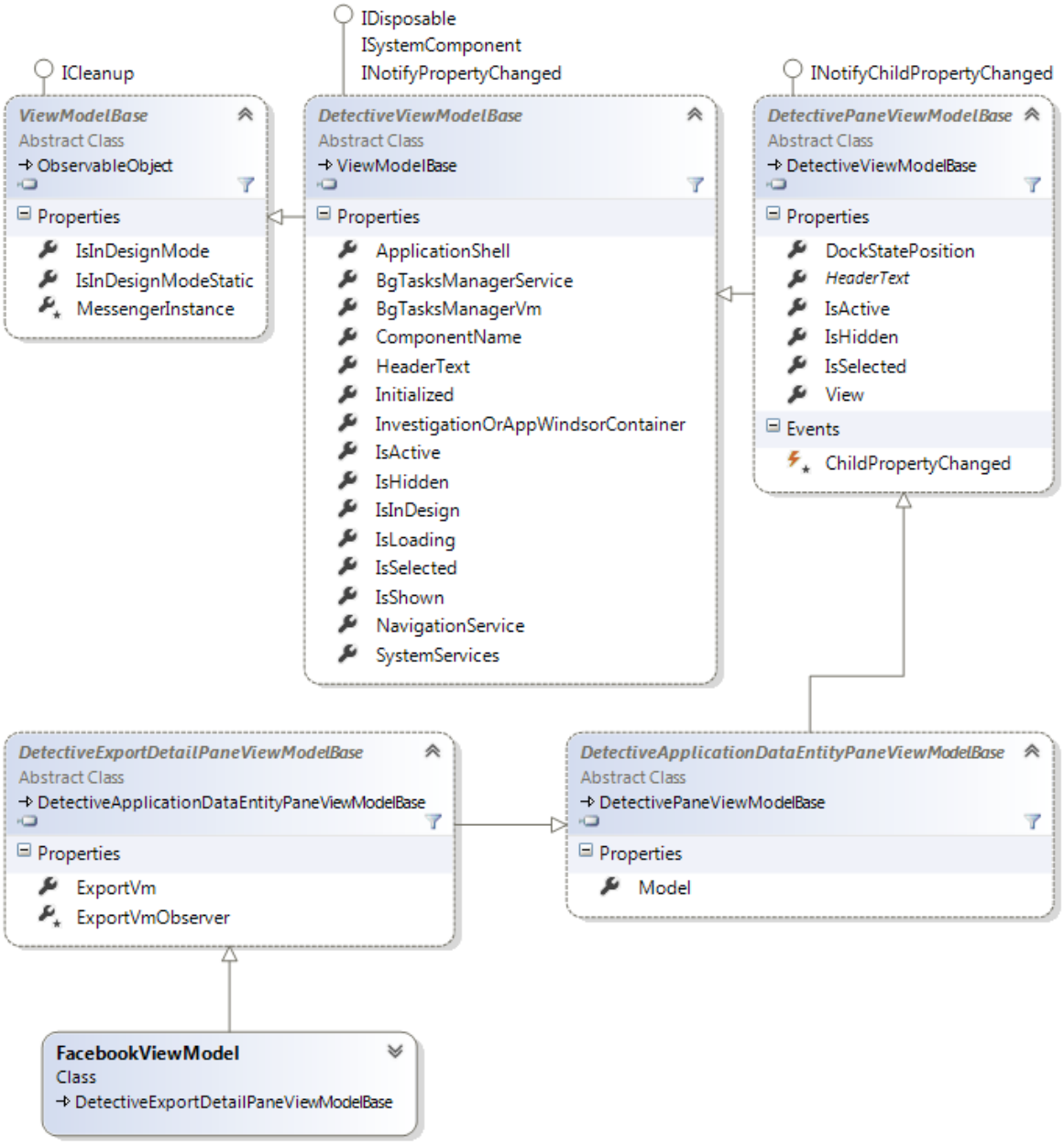
Obrázek B.1: Flowchart identifikace Facebook objektu

Příloha C

Diagramy tříd



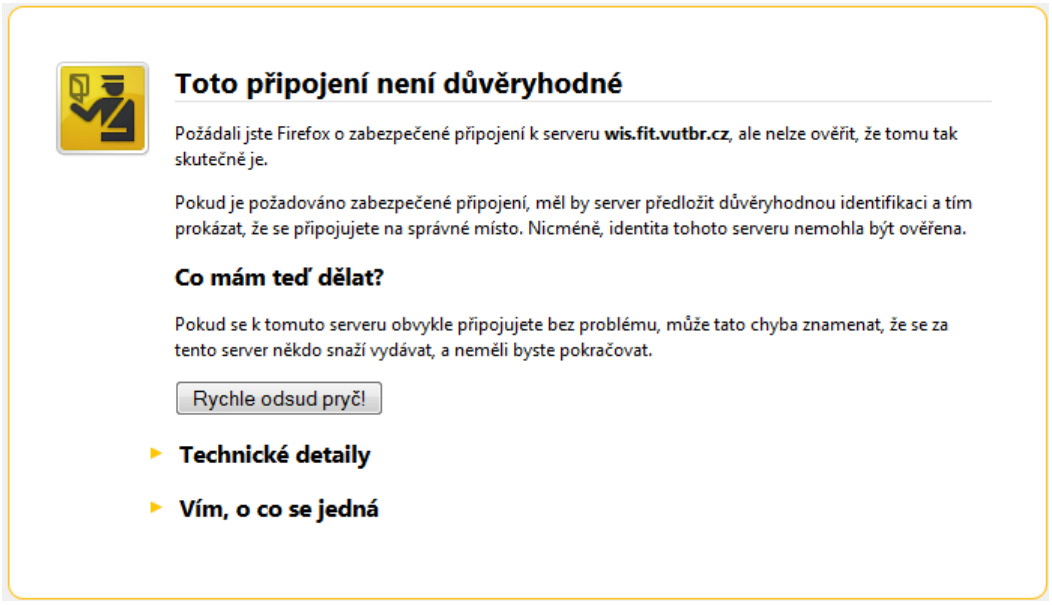
Obrázek C.1: Diagram třídy FacebookDataMining.



Obrázek C.2: Diagram třídy FacebookViewModel.

Příloha D

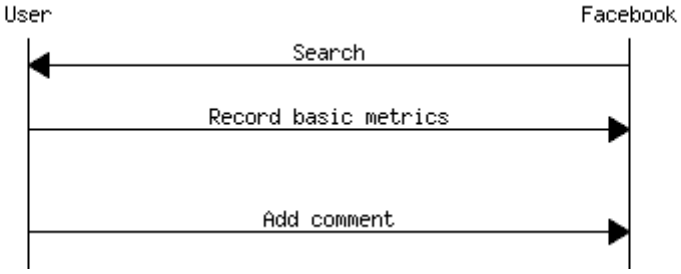
Varování proti podvrhnutí certifikátu



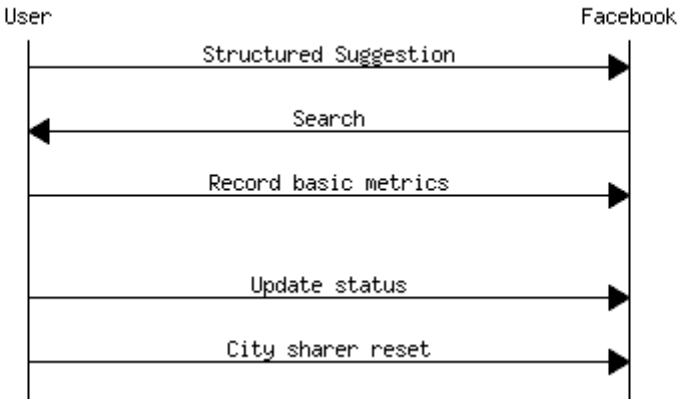
Obrázek D.1: Firefox oznámení o nevěrohodném certifikátu

Příloha E

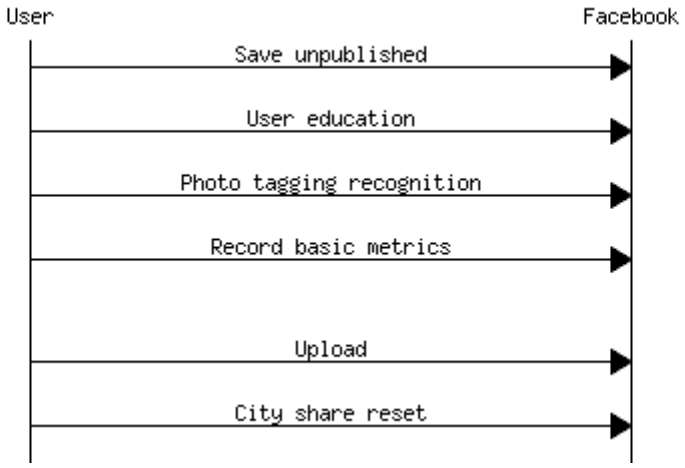
Sekvenční diagramy Facebook protokolu



Obrázek E.1: Komentář

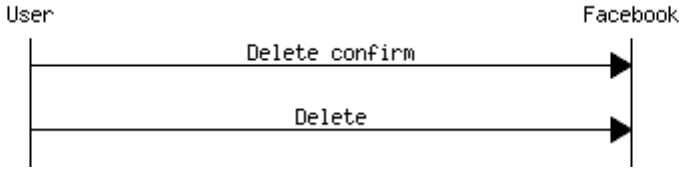


Obrázek E.2: Přidání stavu



Obrázek E.3: Přidání fotky

Obrázek E.4: Přidání se do skupiny



Obrázek E.5: Smazání komentáře

Příloha F

JSON chat zprávy na Facebooku

Data ze dne 27. 11. 2014.

```
message_batch[0][action_type]: ma-type:user-generated-message
message_batch[0][author]: fbid:100007717846239
message_batch[0][timestamp]: 1415172865981
message_batch[0][timestamp_absolute]: Today
message_batch[0][timestamp_relative]: 8:34am
message_batch[0][timestamp_time_passed]: 0
message_batch[0][is_unread]: false
message_batch[0][is_cleared]: false
message_batch[0][is_forward]: false
message_batch[0][is_filtered_content]: false
message_batch[0][is_spoof_warning]: false
message_batch[0][source]: source:chat:web
message_batch[0][source_tags][0]: source:chat
message_batch[0][body]: TESTOVA ZPRAVA
message_batch[0][has_attachment]: false
message_batch[0][html_body]: false
message_batch[0][specific_to_list][0]: fbid:765374730
message_batch[0][specific_to_list][1]: fbid:100007717846239
message_batch[0][signatureID]: 2d3706c9
message_batch[0][ui_push_phase]: V3
message_batch[0][status]: 0
message_batch[0][auto_retry_cnt]: 0
message_batch[0][manual_retry_cnt]: 0
message_batch[0][client_thread_id]: user:765374730
client: mercury
__user: 100007717846239
__a: 1
__dyn:7n8ajEBQcmdzpQ9UoHFaeFxq9ACxO4oKAdBGeqrWoBzEy78S8w
__req: r
fb_dtsg: AQHPKHwtfsjC
ttstamp: 265817280757211911610211510667
__rev: 1481734
```