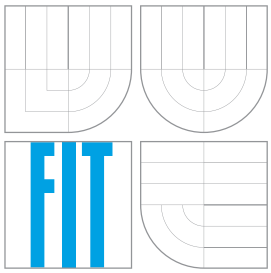
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

FORENZNÍ ANALÝZA WEBOVÝCH PROHLÍŽEČŮ

FORENSIC ANALYSIS OF WEB BROWSERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DANIEL DUŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL OČENÁŠEK, Ph.D.

BRNO 2016

Zadání bakalářské práce/13984/2015/xdusek21

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Dušek Daniel**
Obor: Informační technologie
Téma: **Forenzní analýza webových prohlížečů**
Forensic Analysis of Web Browsers
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s technikami, postupy a softwarem pro forenzní analýzu (FA). Zaměřte se na způsoby a metody, které mohou forenzní vyšetřování ohrozit.
2. Navrhněte a implementujte nástroj, který dokáže provést z hlediska FA bezpečné mazání citlivých údajů a souborů. Volitelně navrhněte/implementujte další funkcionalitu, která znemožní či ztíží možnost FA. V obou dvou případech se zaměřte na běžně používané souborové systémy.
3. Seznamte se s technikami používanými pro jednoznačnou indentifikaci (fingerprinting) uživatele při používání webového prohlížeče. Pokuste se odhadnout, jaké techniky a postupy budou použitelné v budoucnosti.
4. Navrhněte a implementujte teoretické i praktické postupy použitelné k co nejlepší identifikaci konkrétního uživatele skrze informace poskytnuté prohlížečem.
5. Implementované nástroje otestuje, zhodnoťte dosažené výsledky a navrhněte další rozšíření.

Literatura:

- Phillips Amelia, Nelson Bill, Enfinger Frank, Steuart Chris: Guide to Computer Forensics and Investigations, Course Technology, 2003.
- Prosise Chris, Mandia Kevin: Incident Resoponse and Computer Forensics, Osborne McGraw-Hill 2002.
- Brown Christopher LT: Computer Evidence : Collection Preservation, Charles River Media, 2005.
- Stallings, W.: Cryptography and network security: principles and practice. 2nd ed., Prentice Hall, Upper Saddle River, 1999.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 - 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Očenášek Pavel, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce řeší návrh a implementaci dvojice nástrojů souvisejících s forenzní analýzou webových prohlížečů Google Chrome a Google Chromium. První nástroj si klade za cíl bezpečně mazat citlivá data a údaje z prohlížeče, zatímco druhý nástroj cílí na generování otisku uživatele založeného právě na citlivých datech a údajích v prohlížeči zanechaných. V rámci řešení tohoto problému byla vytvořena doposud neexistující dokumentace souborových databází cílových prohlížečů. Implementace nástroje pro bezpečné mazání je realizována pomocí opakovaného přepisu citlivých dat náhodnými hodnotami a následným odstraněním těchto dat. Nástroj pro jednoznačnou identifikaci využívá poznatků získaných v rámci řešení této práce pro vytvoření co nejlepšího možného otisku uživatele používajícího prohlížeč. V práci je blíže popsáno výsledné testování implementovaných nástrojů. Čtenář této práce může těžit z poznatků získaných o databázových tabulkách jednotlivých souborových databází aplikací Google Chrome a Google Chromium a významu jejich sloupců.

Abstract

This work deals with design and implementation of two tools related to web browser forensics. First tool implements functionality to safely delete user's private browsing data stored by applications Google Chrome and Google Chromium. The second tool on the other hand targets on specific user identification based on his browsing activity. In course of design and implementation, the untill-now-inexistent documentation for Chrome's database was created. Implementation of secure deletion is realized using multiple overwrites of targeted files. Specific user identification tool uses researched columns' value meanings in order to identify specific user. Reader can benefit from summarized knowledge of Chrome's database structures and column value meanings.

Klíčová slova

Forenzní analýza prohlížečů, Google Chrome & Google Chromium forenzika, Bezpečné mazání citlivých dat, Jednoznačná identifikace uživatele

Keywords

Web browser forensics, Google Chrome & Google Chromium forensics, Secure sensitive data deletion, Specific user identification

Citace

DUŠEK, Daniel. *Forenzní analýza webových prohlížečů*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Očenášek Pavel.

Forenzní analýza webových prohlížečů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Očenáška, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Daniel Dušek
16. května 2016

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Pavlu Očenáškovi, Ph.D. za poskytnuté rady, věcné připomínky a odborné vedení práce.

Velké díky patří celé mé rodině za nikdy nekončící podporu poskytovanou po celou dobu mého studia.

V poslední řadě bych rád poděkoval přítelkyni Anně, která mi byla po dobu psaní práce silnou morální oporou.

Obsah

Kapitola 1

Úvod

Počátek 21. století a jeho pokračování je spojováno s masivním a rychlým rozšířením výpočetní techniky a Internetu i mezi běžnou a neodbornou veřejností. Výpočetní technika a Internet se staly takřka každodenní a nedílnou součástí našeho života. Téměř každý člověk dnes nosí v kapse telefon, vlastní počítač a připojuje se k Internetu. Toto rozšíření udělalo naše životy v mnohých ohledech mnohem jednodušší.

Se začleněním výpočetní techniky do našich životů se bohužel začala objevovat i nová forma kriminality — počítačová kriminalita, nebo také kybernetická kriminalita – zacílená a páchaná právě prostřednictvím zařízení, která lidé denně používají. Vznik nového odvětví kriminality vedl i ke vzniku nového podoboru forenzní vědy — počítačové forenziky.

V této práci se autor zaměřuje na vytvoření nástroje pro bezpečné mazání citlivých souborů a údajů z webového prohlížeče z hlediska forenzní analýzy a na další způsoby, kterými lze ztížit či znemožnit forenzní analýzu úplně.

V souvislosti s vytvářením tohoto nástroje se v práci dále autor zabývá metodami a přístupy, které forenzní analýzu počítače ohrožují. Znalost těchto metod je klíčová k správnému uchopení návrhu a implementace.

Cílem tohoto nástroje ovšem není dát útočníkovi do rukou prostředek, kterým po sobě zamete stopy, nýbrž dát běžnému uživateli možnost, jak se bránit scénáři, kdy útočník použije metody forenzní analýzy k tomu, aby o něm získal důvěrné informace.

Dále je pak práce zaměřena na metody identifikace konkrétního uživatele prostřednictvím prohlížeče *Google Chrome* a reálnou implementaci aplikace, která tyto metody spojuje a aplikuje.

Blíže se práce věnuje možnostem identifikace uživatele prostřednictvím skenování souborového systému a vyhledáváním citlivých informací, které prohlížeč ukládá o uživatelově činnosti.

Nakonec jsou vytvořené nástroje otestovány a představeny v podobě aplikace *SD4Gen*, která je produktem této práce. Dále jsou také zhodnoceny výsledky, ke kterým autor dospěl.

Veškerá výše popsaná funkcionalita je testována a určena pro běžně používané souborové systémy linuxových distribucí bez žurnálování, *FAT32* a *ext2*, popřípadě *ext3*, kde je žurnálování zapnuté v základním nastavení [7].

Kapitola 2

Počítačová forenzika

Jak již bylo nastíněno v úvodu, s masivním rozšířením výpočetní techniky došlo také ke vzniku nového odvětví kriminality a v reakci na to i ke vzniku nového podoboru forenzní analýzy, **počítačové forenziky**.

2.1 Obecný postup forenzního vyšetřování – DFRW model

Se vzrůstajícím počtem případů kyberkriminality vznikla také poptávka po obecně použitelném modelu forenzního vyšetřování. Jedním z nejznámějších a všestranně nejpoužitelnějších modelů [10] je Digital Forensics Workshop [15] model, který sestává ze sedmi klíčových fází:

- **Identifikace** (*identification*) – počáteční identifikace, že situace vyžaduje forenzní vyšetřování,
- **uchování** (*preservation*) – zajištění, že data budou získána forenzním způsobem a že veškeré provedené akce budou dobře dokumentovány,
- **sběr dat** (*collection*) – použití ověřených softwarových a hardwarových prostředků ke sběru důkazů,
- **vyšetřování** (*examination*) – určení relevantních dat pomocí filtrovacích a extrahovacích technik,
- **analýza** (*analysis*) – porozumění časové posloupnosti událostí a spojení dat s cílem vytvořit si kompletní představu o incidentu,
- **prezentace** (*presentation*) – dokumentování a prezentace výsledků procesu vhodným způsobem,
- **rozhodnutí** (*decision*) – při soudním vyšetřování jde o rozhodnutí, zda jsou důkazy dostačující k vedení trestního řízení; v rámci vyšetřování v organizaci je v tomto bodě rozhodováno, zda budou provedeny akce vůči pachateli.

V závorkách za jménem každé fáze je také uveden anglický název dané fáze, aby bylo předejito možnému zmatení jednotlivých fází při překladu. Dostupná česká literatura se s názvy a leckdy i počtem fází poměrně liší.

2.2 Metody a postupy ohrožující forenzní vyšetřování

Jak vyplývá z druhé a třetí fáze DFRW modelu, veškerá data, která zároveň slouží jako potenciální důkaz, je třeba zajistit bezpečným způsobem. Během těchto dvou fází je také nejpravděpodobnější, že dojde k pochybení, které může mít vliv na vyšetřování.

Eric Shirk v roce 2007 napsal článek [16] zabývající se riziky a možnými pochybeními při provádění takzvané „*udělej si to sám*“ počítačové forenziky. Kromě popisu běžných pochybení pramenících z nedostatečné znalosti personálu zajišťujícího místo činu před příchodem forenzního specialisty se v článku věnuje i metodám, které sám nezkušený forenzní specialista může provést chybně a ohrozit tím forenzní vyšetřování.

Mezi rizikové přístupy relevantní k cílům této práce patří:

- **Zničení a manipulace** — jedná se o zásahy, ke kterým dojde před příjezdem specialisty: IT tým vypne počítač, „nakoukne“ do počítače a hledá sám důkazy,
- **Podřadné metody, nástroje a trénink** — používání nevalidovaných forenzních nástrojů, nedostatečná znalost operačních systémů, metody prováděné v „živém prostředí“.

Zmíněné přístupy se týkají poškozování a zanášení nerelevantních dat a metadat do prostředí, které podléhá forenznímu vyšetřování. Tyto akce mohou být komplikací pro forenzní vyšetřování a mohou ohrozit použitelnost důkazů. Právě na poškozování dat se soustředí část této práce, jejíž cílem je bezpečně mazat citlivé soubory z prohlížeče.

2.3 Forenzní analýza webových prohlížečů

Prakticky každému případu kyberkriminality musí předcházet fáze, kdy pachatel získává informace a zkušenosti potřebné k tomu, aby byl schopen zločin vykonat. Ve většině případů je pravděpodobné, že tyto zkušenosti a informace získá prostřednictvím Internetu, konkrétněji, skrze webový prohlížeč.

Článek od kolektivu autorů Junghoon Oh, Seungbong Lee a Sangjin Lee [14] se shoduje s mnoha dalšími dostupnými články z Internetu na místech, kterým je třeba věnovat zvýšenou pozornost při forenzní analýze webového prohlížeče:

- Historie,
- cookies,
- stažené soubory,
- data webových formulářů,
- uložené přihlašovací údaje.

Pouhou analýzou stránek, které pachatel procházel, lze vyvodit závěry o tom, co bylo jeho úmyslem. Zde se také s úspěchem dá uplatnit analýza parametrů adres z vyhledávačů.

Vyhledávač Google například předává vyhledávaná klíčová slova v adrese, kterou je specialista v rámci forenzního vyšetřování schopný analyzovat:

```
https://www.google.com/?#q=How+to+hack+Facebook
```

Už od prvního pohledu lze určit, co konkrétní osoba vyhledávala, a co je důležitější, co bylo jejím úmyslem. Vyhledávaná klíčová slova poskytují specialistovi unikátní pohled do mysli pachatele a mohou ve výsledku vést k jeho dopadení.

Kromě vyhledávací fáze pachatel používá webový prohlížeč i k osobním záležitostem, jako je komunikace s přáteli, vyřizování emailů či různé volnočasové aktivity. Tyto informace mohou být pro forenzní vyšetřování taktéž podstatné — pachatel může kupříkladu někomu prostřednictvím sociální sítě sdělit své úmysly, nebo může být k trestné činnosti naveden. Zde pak přichází na řadu analýza prohlížeče, seznamu stažených souborů, či samotných stažených souborů.

Kapitola 3

Bezpečné mazání citlivých dat a údajů

Jedním z cílů této práce je vytvořit nástroj, který dovede provést z hlediska forenzní analýzy bezpečné mazání citlivých údajů a souborů z webového prohlížeče. Tato kapitola zahrnuje jak význam slovního spojení „bezpečné mazání“ — a to především z pohledu počítačové forenziky — tak návrh nástroje, který toto mazání provádí, stejně tak jako i jeho implementaci.

3.1 Bezpečné mazání z pohledu počítačové forenziky

Pro pochopení významu forenzně bezpečného mazání dat je nejprve třeba si uvědomit, jakým způsobem jsou data mazána běžným způsobem, z pohledu souborového systému. Je poměrně běžnou praxí, že dnešní souborové systémy při žádosti o smazání souboru vůbec soubor fyzicky neodstraní — pouze odeberou záznam o mazaném souboru z kořenového adresáře souborového sytému. Data obsažená v souboru zůstávají nedotčena, avšak místo, na kterém v paměti leží je označeno jako prázdné a lze do něj zapisovat nové hodnoty [12]. Dokud nejsou data přepsána, lze k nim přistupovat, popřípadě je i číst a obnovovat [11].

Forenzně bezpečné mazání se tedy musí postarat i o zbytková data, která po odstranění souborovým systémem zůstanou přístupná. V nástroji navrhovaném a implementovaném v rámci této práce je použita strategie, při které dochází k opakovanému přepisování místa, kde je uložen mazaný soubor, náhodně generovanými daty. Po několikanásobném přepsání obsahu souboru je pak dále postupně přepisován název souboru, čímž se dociluje vymazání zbývajících záznamů a metadat o souboru ze souborového systému [6]. Tato strategie by měla být použitelná i při snaze o obnovení a čtení dat pomocí drahého hardwarového vybavení [11]

3.2 Návrh nástroje

Hlavní funkcionalita nástroje by měla být naimplementována jako konzolová aplikace spustitelná z příkazové řádky, kde její činnost bude určena parametry a jejich hodnotami. Nástroj by měl cílit na běžně používané souborové systémy a měl by umožňovat bezpečné smazání citlivých údajů a dat, které po sobě uživatel zanechá v prohlížeči při jeho používání.

Dalším požadavkem na výsledný nástroj je možnost jeho propojení s funkcionalitou poskytovanou nástrojem navrženým a implementovaným v druhé části této práce. Tento

požadavek bylo nutné zohlednit již v rané fázi návrhu — následkem zohlednění tohoto požadavku jsou parametry pro nástroj pro bezpečné mazání citlivých dat a pro nástroj pro identifikaci konkrétního uživatele v průniku.

V této kapitole je kromě návrhu, struktury a implementace nástroje pro bezpečné mazání dat popisována i společná část pro oba dva nástroje implementované v rámci této práce, která slouží jako vstupní bod celého nástroje.

3.2.1 Volba cílové platformy a software

Tato práce se zabývá návrhem a implementací nástroje pro bezpečné mazání citlivých dat a údajů pro souborové systémy bez žurnálování, používané běžnými linuxovými distribucemi – tedy *FAT32*, *ext2* a *ext3* s vypnutým žurnálováním. Volba je ovlivněna faktem, že selektivní bezpečné mazání souborů je proveditelné pouze na souborových systémech bez žurnálování, nebo na takových systémech, kde je žurnálování vypnuto [6].

Volba cílového prohlížeče, nad nímž tento nástroj operuje, je částečně odvislá od volby souborových systémů. Vzhledem k nátuře vybraných souborových systémů byl vybrán prohlížeč *Google Chrome*, zejména pro svou dnešní rozšířenost, která v době psaní práce dosahuje až 68% podílu na trhu [4]. V užším zvažovaném výběru byl i prohlížeč *Firefox*, avšak vzhledem ke zmíněnému podílu na trhu nakonec nebyl vybrán.

3.2.2 Volba implementačního jazyka

Jako implementační jazyk byl zvolen jazyk C++, zejména proto, že se jedná o jazyk nižší úrovně. Díky této nižší úrovni je možné provádět operace jako čtení a zápis přímo na fyzická umístění na disku [5] [13].

Jazyk C++ dále poskytuje možnost využití objektového přístupu, což usnadňuje abstrakci reality a zapouzdření související funkcionality do tříd dle jejich logického významu [13].

Jiným zvažovaným implementačním jazykem byl čistý jazyk C ze stejných důvodů jako výše zmíněný C++, avšak absence možnosti objektově orientovaného přístupu v tomto jazyce podpořila autorovo rozhodnutí nevyužít ho.

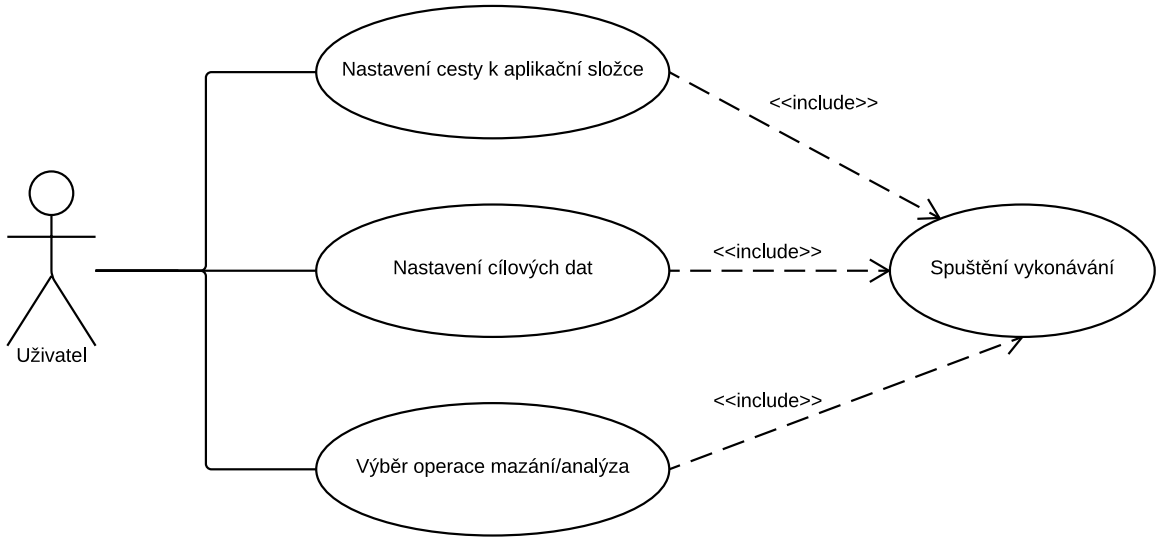
3.2.3 Diagram případů použití

Diagram případů použití zavádí pouze jednu roli, a to uživatele. Neexistuje zde žádná speciální role pro správce či administrátora. Uživatel má k dispozici celkem 3 operace, které může provést.

První operací je zadání cesty ke kořenové složce aplikační složky prohlížeče s uživatelskými daty. Postup vyhledání a zadání cesty ke správné složce je popsán v manuálu k nástroji přiloženém v sekci přílohy této práce. Další operací, kterou uživatel může provést, je nastavení cíle dat, nad kterými bude vybraná operace probíhat. Způsob tohoto nastavení a možné volby jsou taktéž popsány v manuálu v sekci přílohy. Třetí a poslední operací je pak výběr operace, která bude prováděna — uživatel má na výběr celkem ze dvou možností: bezpečné mazání dat a analýza existujících dat. Volba je blíže opět popsána v manuálu k nástroji v sekci přílohy.

Veškeré nastavení provedené výše popsanými případy je pak spuštěno povinnou operací „Spuštění vykonávaní“.

V konzolové aplikaci jsou jednotlivé případy použití zastoupeny použitím správných parametrů spuštění nástroje, stisknutí klávesy „*Enter*“ pak zastupuje operaci „Spuštění vykonávaní“.



Obrázek 3.1: Diagram případů použití aplikace SD4Gen

3.2.4 Parametry spuštění

Jak již bylo zmíněno, parametry spuštění nástroje jsou společné jak pro nástroj pro bezpečné mazání citlivých dat z prohlížeče, tak pro nástroj pro jednoznačnou identifikaci uživatele, neboť jsou implementovány dohromady, jako jedna aplikace. Očividným parametrem spuštění nástroje je tedy výběr mezi režimem mazání dat a analýzou existujících dat.

Dále existuje parametr, který ovlivní, jaká data budou mazána, respektive analyzována. Zde je třeba zohlednit, že hodnoty tohoto parametru jsou nezávislé na tom, jaký režim práce nástroje byl zvolen. Cílová data jsou stejná, ať už jde o mazání, nebo analýzu.

Nezbytným parametrem je také parametr určující adresář, ve kterém se nachází data, nad kterými má nástroj operovat. Parametry spuštění jsou uvedeny v tabulce 3.1.

Parametr (dlouhá verze parametru)	**Účel**
-h (--help)	Zobrazí nápovědu aplikace
-a (--action)	Nastaví režim běhu aplikace — mazání vs. analýza, hodnota `deletion`
-t (--target)	Výběr dat nad kterými aplikace pracuje, možné hodnoty `all`, `history`, `webdata`, `logindata`, `cookies` a `purge`
-p (--path)	Určuje cestu ke složce aplikačních dat prohlížeče

Tabulka 3.1: Seznam použitelných parametrů a jejich hodnot pro aplikaci SD4Gen v režimu deletion

3.3 Implementace nástroje

V následující části textu je popsána implementace nástroje pro bezpečné mazání citlivých dat, společně s implementací společné části s nástrojem pro jednoznačnou identifikaci uživatele. Jsou zde popsány implementace bezpečného mazání jednotlivých cílů, všech cílů a cíle označeného speciální hodnotou *purge*, dále je zde popsáno využití objektového přístupu, jsou zde zdokumentovány návratové kódy konce aplikace a ukázkové výstupy. V závěru této sekce jsou pak uvedeny metriky kódu.

3.3.1 Společná část — vstupní bod aplikace

Vstupní bod celé aplikace zahrnující oba dva nástroje implementované v rámci této práce je soubor `Main.cpp`. Soubor obsahuje funkce pro zpracování parametrů a jejich argumentů, které jsou zadávány při spouštění výsledné aplikace, funkci zobrazující nápovědu aplikace, dvě funkce volané dle zvoleného režimu aplikace (mazání, analýza) a logiku určující zda bude volána část provádějící bezpečné mazání, nebo analýzu.

Funkce zpracující parametry a jejich argumenty se soustředí na kontrolu, zda zadané parametry aplikace jsou přípustné a zda argumenty jednotlivých parametrů jsou ve správném tvaru. Existují oddělené funkce pro kontrolu zadaných argumentů přepínačů -t (--target), -a (--action). Tyto funkce jsou pak volány z příslušných částí funkce, která zpracovává parametry s využitím `getopt` knihovny. Veškeré parametry a jejich přepínače jsou citlivé na velikost písmen a očekávají, že jejich hodnoty — s výjimkou cesty k adresáři (parametr -p/--path) — budou pouze malá písmena.

Dle nastavené prováděné operace (přepínač -a/--action) se pak uvnitř těla hlavní funkce `main` rozhodne, zda bude instanciována sada tříd související s bezpečným mazáním, nebo jednoznačnou identifikací. Instanciace a volání příslušných metod vytvořených objektů se odehrává uvnitř funkce pro provedení vybrané operace (`ExecuteDeletion()` a `ExecuteFingerPrinting()`).

3.3.2 Použití objektově orientovaného přístupu

Zapouzdření a oddělení logicky souvisejících částí kódu od kódu vstupního bodu aplikace bylo dosaženo vytvořením třídy `DataDeletion`, která je deklarována a implementována v souborech `DataDeletion.h` a `DataDeletion.cpp`.

Tato třída obsahuje veřejné i privátní metody pro mazání souborů obsahujících citlivá data, která si ukládá prohlížeč o uživateli na základě jeho aktivity. Dále je zde privátní členská proměnná typu řetězec, do které se přes konstruktor třídy předává cesta ke složce s aplikačními daty, která je přijata jako argument přepínače -p/--path.

Instance třídy je vytvořena ve funkci `ExecuteDeletion()` volané ze souboru `Main.cpp` v případě, že je parametrem -a/--action vybrána volba bezpečného mazání. V rámci této funkce jsou volány veřejné metody této třídy v závislosti na zvolených cílech (přepínač -t/--target). Po dokončení práce je instance třídy zrušena, aby bylo předejito neuvolnění paměťových prostředků.

3.3.3 Bezpečné mazání jednotlivých cílů

V sekci 2.3 bylo dle poznatků z článku [14] uvedeno, že existuje několik míst, do kterých je významné se dívat, pokud se snažíme zjistit informace o uživateli, který prohlížeč použí-

val. Tato místa lze selektivně vybrat pomocí argumentu parametru -t/--target a bezpečně vymazat. Možné hodnoty tohoto přepínače jsou uvedeny v tabulce 3.1.

Tato místa jsou v aplikaci *Google Chrome* reprezentována souborovými *sqlite3* databázemi s příznačnými názvy, nacházejícími se ve složce aplikačních dat prohlížeče *Google Chrome* [2].

Metody třídy `DataDeletion` pak cílí na jednotlivá z těchto míst a provádí jejich bezpečné mazání pomocí volání systémové funkce `shred` s parametrem `-u`. Toto volání způsobí, že cílový soubor je v paměti celkem 3x přepsán náhodnými bajtovými hodnotami, následně je soubor postupně přejmenováván a smazán [6]. Trojnásobný přepis je dle Gutmanna dostačující k znemožnění obnovy a čtení souboru, a to i za pomoci drahého hardwarového vybavení [11].

Vlastním výzkumem jsem zjistil, že prohlížeč *Google Chrome* si v minulosti vytvářel vlastní žurnálovací soubor pro důležité databáze, a to včetně těch, na které cílí implementovaný nástroj. Tato skutečnost je v nástroji reflektována tím, že metody, které mažou databázové soubory, se navíc ještě dívají, zda existuje žurnál mazané databáze a v případě, že ano, mažou i ten.

3.3.4 Bezpečné mazání všech cílů a speciální cíl „purge“

Jedním z povolených cílů nástroje pro bezpečné mazání citlivých uživatelských dat z prohlížeče je cíl `all`. Tento přepínač maže všechny dostupné dílčí cíle, které uživatel může specifikovat, ale navíc maže ještě uložený soubor s profilovým obrázkem uživatele prohlížeče. Mazání všech cílů je implementováno metodou, která postupně volá všechny metody mazající dílčí cíle a metody pro bezpečné smazání profilového obrázku. Toto mazání probíhá stejně jako bezpečné mazání popsané výše.

Speciální cíl přepínače -t/--target `purge` nevolá žádnou z výše popsaných metod. Cílem tohoto přepínače je umožnit uživateli smazat kompletně uživatelskou složku, kterou si o něm prohlížeč *Google Chrome* vytvořil a docílit tak úplného smazání všech stop o uživatelské aktivitě. Je-li předán přepínači cíl `purge` je volána stejnojmenná, avšak kapitálkovaná metoda třídy `DataDeletion`, metoda `Purge`. Tato metoda provádí opět volání systémové funkce `shred`, ale až nad výsledky rekurzivního volání funkce `find`, která vrátí všechny soubory v složce předané parametrem -p/--path. Následně je opět zavolána systémová funkce, tentokrát však funkce `rm` s parametry `-rf`, taktéž nad složkou předanou parametrem. Tímto je docíleno absolutního smazání stop po uživatelské aktivitě.

3.3.5 Návratové kódy konce aplikace

Proběhnou-li veškeré operace prováděné aplikací a jednotlivými nástroji bez problémů, aplikace vždy vrací hodnotu `0`. Dojde-li však k chybě za běhu některého z nástrojů, či samotné aplikace, končí aplikace s číslem odlišným od 0, které reprezentuje konkrétní chybu. Možné návratové kódy chyb společné části aplikace pro oba nástroje a chybové kódy nástroje pro bezpečné mazání jsou znázorněny v tabulce 3.2

3.3.6 Ukázkové výstupy nástroje pro bezpečné mazání

Zde jsou uvedeny ukázkové výstupy, které mohou být vyprodukovány nástrojem pro bezpečné mazání citlivých dat, včetně ukázky spuštění nástroje a krátkého popisku významu výstupu.

Kód	Důvod
0	Aplikace proběhla bez problémů
1	Špatné přepínače -a či -o
2	Chybějící cesta k aplikační složce (-p přepínač)
3	Zvolena nesprávný cíl (-t)
4	Bezpečné mazání selhalo

Tabulka 3.2: Návratové kódy konce aplikace SD4Gen pro nástroj bezpečného mazání

```
user@domain:$ ./SD4Gen -a deletion -t all -p ../google-chrome/Default/
Deletion completed successfuly.
```

Kód 3.2: Spuštění nástroje bezpečného mazání nad složkou **obsahující** soubory k mazání, včetně žurnálovacích souborů

```
user@domain:$ ./SD4Gen -a deletion -t all -p ../google-chrome/Default/
History database file does not exist, no deletion executed.
No History-journal file to be deleted.
Web Data database file does not exist, no deletion executed.
No Web Data-journal file to be deleted.
Login Data database file does not exist, no deletion executed.
No Login Data-journal file to be deleted.
Cookies database file does not exist, no deletion executed.
No Cookies-journal file to be deleted.
Deletion completed successfuly.
```

Kód 3.3: Spuštění nástroje bezpečného mazání nad složkou **neobsahující** žádné soubory k mazání

3.3.7 Metriky kódu

Závěrem této kapitoly jsou uvedeny metriky kódu zahrnující implementaci společné části aplikace a implementaci nástroje pro bezpečné mazání citlivých údajů a souborů.

Počet souborů se zdrojovými kódy: 5
Počet řádků kódu napříč soubory: 994
Velikost souborů: 30.4 kB

Kapitola 4

Metody jednoznačné identifikace uživatele

Tato kapitola pojednává o metodách a technikách používaných pro jednoznačnou identifikaci uživatele na základě jeho aktivity při používání webového prohlížeče. V závěru této kapitoly je pak uveden odhad, jakým směrem se v budoucnosti budou dále tyto postupy a techniky vyvíjet a ubírat.

V době psaní této práce neexistuje žádná uznávaná ucelená metoda jednoznačné identifikace uživatele, vždy jde o posloupnost kroků analyzujících stopy, které po sobě uživatel v prohlížeči zanechal. Samotná identifikace pravděpodobného uživatele je až kognitivní proces provedený vyšetřovatelem [14].

4.1 Reálná a virtuální identita

Mluvíme-li o jednoznačné identifikaci uživatele, dosti pravděpodobně mluvíme o zjištění, o jakého konkrétního jedince lidské společnosti jde, a tedy mluvíme o reálné, světské, identitě konkrétního uživatele. Zajímá nás reálné jméno uživatele, jeho adresa, místo pobytu, telefonní číslo – informace, které nás dovedou ke konkrétní osobě.

Je ovšem nutné si uvědomit, že informace, které o identitě uživatele můžeme získat z jeho webového prohlížeče, nebudou ve většině případů takto specifické. Informace, které získáme, budou pravděpodobně reprezentovat pouze virtuální identitu daného uživatele. Virtuální identita se nebude stoprocentně shodovat s reálnou identitou uživatele, avšak je pravděpodobné, že poskytne vodítko, které potenciálního vyšetřovatele dovede k reálné identitě uživatele. V kontextu této práce bude pojem virtuální identita používán ve smyslu množiny informací, které je možné o uživateli zjistit analýzou jeho aktivity ve webovém prohlížeči.

Typickými informacemi, které je možné získat z virtuální identity uživatele, jsou například oblíbené stránky, koníčky, přezdívky či emailová adresa. Tyto informace jsou většinou spojeny s pohybem uživatele po Internetu.

Z těchto informací by ne vždy muselo být patrné, k jaké reálné identitě se virtuální identita váže. Naštěstí dnes žijeme v době, kdy je zcela běžné vlastnit například Facebookový účet, vlastnit email reprezentující občanské jméno, případně nakupovat v internetových obchodech. Z informací, které uživatel poskytne zmíněným stránkám, a tím pádem je zároveň přidá do informací patřících k jeho virtuální identitě, už je mnohem lépe možné udělat si obrázek o tom, komu ona virtuální identita patří.

4.2 Proces analýzy zanechaných stop

Pro výsledný odhad a určení nejpravděpodobnějšího uživatele, který používal prohlížeč, je třeba provést nejprve analýzu stop, které po sobě zanechal. Tyto stopy jsou typicky zanechávany v souborech, do kterých si je odkládá prohlížeč.

Jedna z populárních metodologií analýzy zanechaných stop doporučuje rozdělení celkové analýzy na logické části, čímž cílí na logické a chronologické posloupnosti všech druhů analýzy, což ve výsledku vede k lépe interpretovatelným výsledkům [14]:

- **Integrovaná analýza** - vyšetřovatel by měl provést analýzu všech prohlížečů uložených na analyzovaném systému, aby nedošlo ke ztrátě dat,
- **analýza časové osy** – vyšetřovatel by měl sestavit časovou osu, ze které bude možné sledovat, jakým způsobem docházelo k páchání vyšetřovaného skutku; v časové ose nesmí být opomenuta ‚studijní fáze', kdy pachatel získával informace potřebné k provedení trestného činu,
- **analýza vyhledávací historie** – mimo zachycení pohybu pachatele po stránkách v čase je také třeba analyzovat jím vyhledávané termíny, neboť právě tyto vyhledávané termíny dávají vyšetřovateli unikátní náhled do mysli pachatele; toto téma je již nastíněno v sekci 2.3,
- **analýza zakódovaných adres** – dále je třeba zohlednit fakt, že pro navštívené stránky a vyhledávané výrazy v jiných jazycích než je angličtina, mohou být data obsahující znaky specifické pro daný jazyk, zakódována v nečitelné podobě; na toto je třeba pamatovat a provést analýzu i takových dat,
- **analýza uživatelské aktivity** – vyšetřovatel musí také prověřit stránky, které uživatel navštívil a vyšetřovatel je subjektivně vyhodnotil jako důležité; jde o časově velice náročný proces, avšak nezbytný k plnému porozumění nátuře vykonaného činu.

Výše popsaná metodologie je velmi zatížená na lidského vyšetřovatele a obsahuje mnoho míst, kde by právě lidský faktor mohl selhat – například vyhodnocení, které adresy jsou skutečně důležité, a které ne.

4.3 Techniky použitelné v budoucnosti

Jedním z cílů této práce je odhadnout, jaké postupy a techniky vedoucí k jednoznačné identifikaci uživatele budou použitelné v budoucnosti.

V posledních letech se začal rozvíjet trend hostování uživatelských dat na serverech patřících provozovateli služby, případně pronajatých provozovateli služby. Autor této práce se domnívá, že v budoucnosti budou i data, jako jsou stopy identifikující uživatele, ukládána do tohoto prostoru. Dle autora této práce má k tomuto přístupu nejlépe našlápnuto společnost *Google a.s.*, která má již nyní provázaný prohlížeč *Google Chrome* s uživatelským účtem, který je dále vázaný k reálné identitě uživatele.

Dalším odhadem je, že výše popsaná metodologie bude z velké části automatizována — například místo vyšetřovatele posuzujícího, zda je důležité konkrétní stránku prověřit, bude každá stránka prověřena automaticky programem provádějícím analýzu stop zanechaných uživatelem. Tím bude vyloučeno riziko chyby lidského faktoru.

Kapitola 5

Identifikace konkrétního uživatele aplikace Google Chrome

Druhým cílem této práce je navrhnout a naimplementovat nástroj, který spojí praktické i teoretické postupy použitelné k co nejlepší identifikaci konkrétního uživatele skrze informace poskytnuté prohlížečem.

5.1 Analýza relevantních dat

Data použitelná pro jednoznačnou identifikaci uživatele jsou k nalezení ve složce s uživatelskými daty aplikace Google Chrome. V této složce jsou přítomny soubory obsahující databáze informací o uživatelské aktivitě, v kontextu této práce jsou to jmenovitě soubory: History, Login Data, Web Data, Cookies. Mimo to je zde navíc uložen soubor Google Profile Picture.png, který může obsahovat uloženou podobiznu uživatele v případě, že byl prohlížeč propojen s účtem Google.

Databáze uložené v souborech jsou ve formátu *sqlite3* [2] a k jejich prohlížení může být použit například volně dostupný nástroj *DB Browser for SQLite*[1]. V procesu vzniku práce byl použit právě tento nástroj k zjištění struktury databází uložených v souborech a zkoumání reálné ukládaných hodnot aplikací *Google Chrome*.

V následujících podsekcích jsou popsány datové typy použité pro reprezentaci času v databázích, jednotlivé databáze, jejich tabulky a sloupce, jejichž hodnoty byly shledány jako relevantní k předmětu této práce. Proces vyhodnocení, zda jsou data relevantní k jednoznačné identifikaci uživatele, byl inspirován a validován vůči poznatkům zmíněným v článcích [9] [8].

5.1.1 Časové formáty použité pro reprezentaci času v databázích

Při práci s výše zmíněnými databázemi bylo odhaleno, že pro reprezentaci času uvnitř databází a aplikace jsou použity celkem tři typy reprezentace času:

- *Unixový čas* — udávaný v počtu sekund uplynulých od 1. 1. 1970, rozpoznatelný dle typického počtu 10 digitů,
- *Windows FILETIME* — udávaný v počtu mikrosekund od 1.1. 1601, rozpoznatelný dle typického počtu 18 digitů,

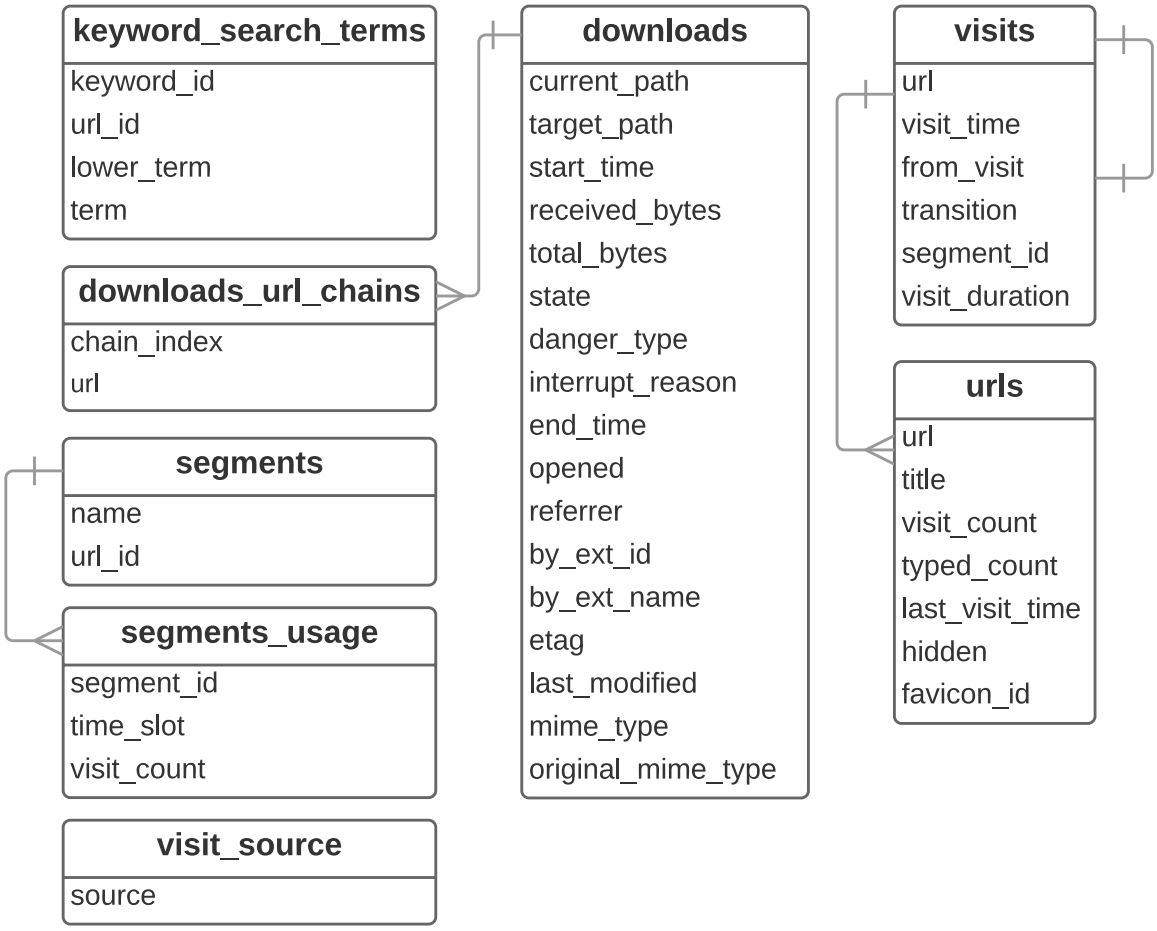
[1] Nástroj DB Browser for SQLite dostupný z `http://sqlitebrowser.org/`

- *CTime* — udávaný jako řetězec anglicky čitelného času, např.: 10:15 PM March 19, 1999.

5.1.2 Databáze History

V databázi *History* jsou obsaženy především informace o prohlížecí aktivitě uživatele, jako jsou informace o navštívených stránkách, počet těchto návštěv, seznam všech navštívených adres, ručně zadané adresy, informace o vyhledávaných klíčových slovech či informace o stažených souborech.

Schéma *History* databáze (bez meta-tabulek) je znázorněno v obrázku 5.1. Níže znázorněné databázové tabulky *segments_usage*, *segments* a *visit_source* nejsou přímo relevantní k určení identity uživatele prohlížeče, slouží pouze pro interní účely aplikace *Google Chrome*. Tato informace byla zjištěna zkoumáním hodnot a jejich vztahu vůči ostatním tabulkám.



Obrázek 5.1: Znázornění databázového schématu *sqlite3* databáze *History* generovaného aplikací *Google Chrome*. Obrázek byl vytvořen prostřednictvím nástroje *Lucid Chart* dostupného online [1].

Následující tabulky obsahují podmnožinu sloupců databázových tabulek zajímavých z pohledu jednoznačné identifikace uživatele, s podrobnějším vysvětlením významů sloupců databázových tabulek, které jsou považovány za relevantní k předmětu této práce.

[1] Nástroj Lucid Chart dostupný z `http://lucidchart.com`

Tabulka *urls* obsahuje záznam o každé navštívené adrese webové stránky, kterou uživatel navštívil za období posledních devadesáti dnů. Mezi adresami uvedenými v této tabulce nejsou přítomny adresy zabezpečených webů, ani adresy navštívené v anonymním režimu [1].

History.urls		
Název sloupce	**Datový typ**	**Význam**
url	LONGVARCHAR	Adresa, na kterou uživatel přistoupil
title	LONGVARCHAR	Titulek adresy, na kterou uživatel přistoupil
visit_count	INTEGER	Počet návštěv dané adresy
last_visit_count	INTEGER	*FILETIME* čas posledního přístupu

Tabulka 5.1: Tabulka obsahuje podmnožinu sloupců databázové tabulky *urls* z databáze *History*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

Tabulka visits obsahuje sloupec *transition*; význam jeho hodnot byl v průběhu práce na analýze relevantních dat vyzkoumán prostřednictvím reverzního inženýrství — ze zdrojového kódu samotného Google Chrome. Význam těchto záznamů je zahrnut do tabulky 5.3.

History.visits		
Název sloupce	**Datový typ**	**Význam**
url	INTEGER	Cizí klíč do tabulky *History.urls*
visit_time	INTEGER	*FILETIME* čas návštěvy
from_visit	INTEGER	Odkaz na zdrojovou návštěvu
transition	INTEGER	Typ přechodu na tuto návštěvu
visit_duration	INTEGER	Počet mikrosekund strávených na této návštěvě

Tabulka 5.2: Tabulka obsahuje podmnožinu sloupců databázové tabulky *visits* z databáze *History*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

Hodnota uložená ve sloupci *transition* je nezáporného celočíselného 64 bitového typu, jejíž spodní bajt obsahuje informaci o tom, jakým způsobem bylo na adresu navigováno [2]. Pro získání hodnoty spodního bajtu byla provedena logická operace `AND` této hodnoty s bitovou maskou `0xFF`. Tím je docíleno „zahození“ horních bajtů a ponechání pouze jednoho spodního. Získanou hodnotu je pak možné již porovnat a přiřadit k hodnotám typů přechodů na návštěvu dostupným z *Google Chrome Extension* [3] dokumentace. Tabulka 5.3 obsahuje seznam a význam těchto hodnot.

V tabulce *keyword_search_terms* jsou obsaženy záznamy o vyhledávaných termínech v různých vyhledávacích prvcích na navštívených stránkách. Sloupec *term* obsahuje nemodifikované znění vyhledávaného dotazu, zatímco lower_term sloupec obsahuje vyhledávaný dotaz převedený na malá písmena.

Tabulka *downloads* obsahuje záznamy o stažených, popřípadě aktuálně stahovaných souborech a záznamy o stavu těchto stahování. Sloupce *state*, *danger_type* a *interrupt_reason* obsahují hodnoty, jejichž význam byl opět určen prostřednictvím reverzního inženýrství provedeného nad kódem *Google Chrome*, nebo empirickou simulací situace a zkoumáním reálně uložených hodnot. Hodnoty a možné výsledky jsou popsány v tabulkách 5.8, 5.6 a 5.7.

Hodnoty sloupce visits.transition	
Hodnota sloupce*	**Typ přechodu****
0	LINK
1	TYPED
2	BOOKMARK
3	AUTO_SUBFRAME
4	MANUAL_SUBFRAME
5	GENERATED
6	START_PAGE
7	FORM_SUBMIT
8	RELOAD
9	KEYWORD
10	KEYWORD_GENERATED

Tabulka 5.3: * Hodnota sloupce po aplikaci masky `AND 0xFF`, ** Typ přechodu dle dokumentace Google Chrome Extensions

History.keyword_search_terms		
Název sloupce	**Datový typ**	**Význam**
lower_term	LONGVARCHAR	Vyhledávaný dotaz, v malých písmenech
term	LONGVARCHAR	Vyhledávaný dotaz, originální znění

Tabulka 5.4: Tabulka obsahuje podmnožinu sloupců databázové tabulky *keyword_search_terms* z databáze *History*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

History.downloads		
Název sloupce	**Datový typ**	**Význam**
current_path	LONGVARCHAR	Aktuální cesta ke staženému souboru
target_path	LONGVARCHAR	Cesta ke stahovanému souboru v době stahování
start_time	INTEGER	*FILETIME* čas započetí stahování
received_bytes	INTEGER	Počet stažených bajtů
total_bytes	INTEGER	Počet bajtů celkem
state	INTEGER	Stav stahovaného záznamu
danger_type	INTEGER	Úroveň vyhodnoceného rizika souboru
interrupt_reason	INTEGER	Důvod přerušení stahování
end_time	INTEGER	*FILETIME* čas konce stahování
opened	INTEGER	Informace o tom, zda byl soubor otevřen
referrer	VARCHAR	Adresa, ze které bylo stahování započato
last_modified	VARCHAR	Čas poslední úpravy záznamu v tabulce
mime_type	VARCHAR	Typ mime stahovaného záznamu

Tabulka 5.5: Tabulka obsahuje podmnožinu sloupců databázové tabulky *downloads* z databáze *History*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

Sloupec *state* obsahuje číslenou hodnotu reprezentující stav, ve kterém se stahování na-

chází. Tyto hodnoty byly vyzkoumány empiricky — byly nasimulovány situace, ve kterých došlo k různým koncům stahování. Tyto situace byly zapsány, aby je bylo možné následně identifikovat v databázi, kde jejich výsledky aplikace *Google Chrome* zapsala. Tyto výsledky byly porovnány se zapsanými situacemi a zpětně ověřeny s hodnotami ze zdrojového kódu aplikace *Google Chrome*. Tímto postupem bylo zároveň ověřeno, že zvolená metoda empirického postupu byla v tomto případě aplikovatelná, a že dosažené výsledky mohou být považovány za věrohodné.

Hodnoty sloupce downloads.state	
Hodnota	**Význam**
0	Stahování právě probíhá
1	Stahování bylo dokončeno
2	Stahování bylo přerušeno
3	Stahování bylo zrušeno (vnější akce)
4	Stahování dosáhlo maximálního stavu (interní hodnota)

Tabulka 5.6: Tabulka obsahuje popis významů jednotlivých hodnot, které se mohou vyskytovat ve sloupci *state* uvnitř tabulky History.downloads.

Další sloupec databázové tabulky *downloads*, *danger_type* označuje úroveň, případně typ nebezpečí, které představuje stahovaný soubor. Hodnoty uvedené v tabulce 5.7 byly vyzkoumány aplikací metod reverzního inženýrství na kód aplikace *Google Chrome*.

Hodnoty sloupce downloads.danger_type	
Hodnota	**Význam**
0	Bez nebezpečí
1	Přípona nebezpečná pro systém (např.: *.pdf)
2	URL stahovaného souboru je nebezpečná
3	Obsah souboru je nebezpečný
4	Potenciálně nebezpečný soubor
5	Neznámý/neobvyklý obsah souboru
6	Soubor označen za nebezpečný, uživatel upozornění ignoruje
7	Soubor pochází od nebezpečného hostitele
8	Soubor je nebezpečný pro prohlížeč
9	Maximální nebezpečí (hodnota má pouze interní význam)

Tabulka 5.7: Tabulka obsahuje popis významů jednotlivých hodnot, které se mohou vyskytovat ve sloupci *danger_type* uvnitř tabulky History.downloads.

Sloupec databázové tabulky *interrupt_reason* obsahuje číselnou hodnotu reprezentující důvod, proč bylo dané stahování přerušeno. V případě, že stahování přerušeno nebylo, je hodnota v tomto sloupci rovna 0. Níže vyzkoumané hodnoty byly získány pomocí reverzního inženýrství nad zdrojovým kódem aplikace *Google Chrome*.

Hodnoty sloupce downloads.interrupt_reason	
Hodnota	**Význam**
0	Stahování nepřerušeno
1	Generická chyba operace nad souborem
2	Přístup k souboru zamítnut
3	Nedostatek volného místa na disku
5	Jméno souboru je příliš dlouhé
6	Soubor je příliš veliký pro systém
7	Soubor je nakažený virem
10	Soubor je používán
11	Soubor je zablokován lokálními bezpečnostními pravidly
12	Kontrola bezpečnosti souboru selhala
13	Obnovení stahování nebylo úspěšné
20	Generická síťová chyba
21	Síťová operace překročila svůj časový limit
22	Spojení ztraceno
23	Server ztratil připojení k internetu
24	Neplatný síťový požadavek
30	Server provedl neplatnou operaci
31	Nepodporovaný *range* požadavek
32	Požadavek nesplňuje serverové podmínky
33	Server nemá požadovaná data
40	Uživatel zrušil stahování
41	Uživatel vypnul prohlížeč
50	Prohlížeč havaroval (interní hodnota)

Tabulka 5.8: Tabulka obsahuje popis významů jednotlivých hodnot, které se mohou vyskytovat ve sloupci *interrupt_reason* uvnitř tabulky History.downloads.

Databázová tabulka *downloads_url_chains* úzce souvisí s databázovou tabulkou *downloads*. Tabulka obsahuje záznamy určující, z jakých adres byl soubor (resp. záznam v tabulce *downloads*) stahován po dobu, co stahování probíhalo. Tato informace byla zjištěna empirickými simulacemi odpovídajících situací.

History.downloads_url_chains		
Název sloupce	**Datový typ**	**Význam**
id	INTEGER	Cizí klíč do tabulky *downloads*
url	LONGVARCHAR	Adresa, ze které bylo stahováno
chain_index	INTEGER	Pořadí, ve kterém bylo stahováno ze záznamu

Tabulka 5.9: Tabulka obsahuje podmnožinu sloupců databázové tabulky *downloads* z databáze *History*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

5.1.3 Databáze Login Data

V *Login Data* databázi jsou obsažena přihlašovací data, která si uživatel dobrovolně uložil do prohlížeče, když prohlížeč nabízel, že si je zapamatuje. Databáze obsahuje jedinou relevantní tabulku, a to tabulku *logins*. Z tohoto důvodu zde není uveden obrázek databázového schématu. Tabulka 5.10 zobrazuje množinu sloupců databázové tabulky *logins*, která byla shledána být relevantní k předmětu této práce.

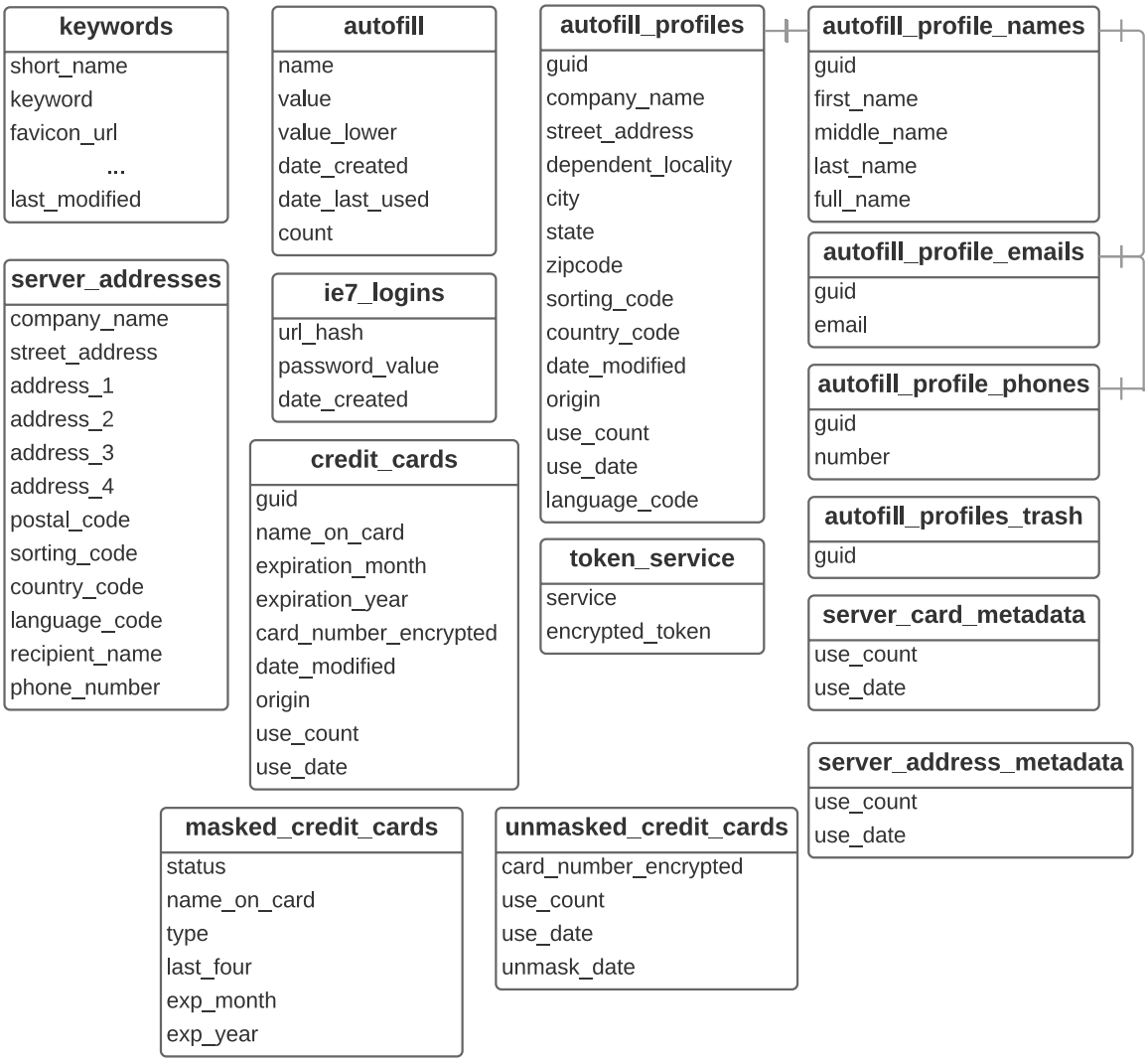
Hodnoty ukládané do sloupců databázové tabulky logins jsou v srozumitelném formátu a žádná z relevantních hodnot není uložena takovým způsobem, aby bylo nutné zjišťovat její význam. Výjimkou je pouze hodnota sloupce *password_value*, která je uložena jako zašifrovaný *blob* — uložená hesla tedy není možné snadnou cestou extrahovat. Na druhou stranu u této hodnoty nás zajímá spíše fakt, jestli je uložena, nebo není. Samotné znění hesla není pro identifikaci uživatele přímo relevantní.

Login Data.logins		
Název sloupce	**Datový typ**	**Význam**
origin_url	VARCHAR	Adresa, na které se vyskytuje formulář
action_url	VARCHAR	Adresa, na které je umístěn přihlašovací skript
username_value	VARCHAR	Zadané uživatelské jméno
password_value	BLOB	Zašifrované použité heslo
signon_realm	VARCHAR	Realm do kterého přihlášení platí
ssl_valid	INTEGER	Informace o tom, zda je podporováno SSL
date_created	INTEGER	FILETIME čas vytvoření záznamu
possible_usernames	BLOB	Další možná uživatelská jména
times_used	INTEGER	Počet použití tohoto záznamu
date_synced	INTEGER	FILETIME čas poslední synchronizace údajů
display_name	VARCHAR	Zobrazovaný název uložených údajů

Tabulka 5.10: Tabulka obsahuje podmnožinu sloupců databázové tabulky *logins* z databáze *Login Data*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

5.1.4 Databáze Web Data

V této databázi si prohlížeč drží informace o tom, jaká data zadal uživatel do textových políček napříč všemi stránkami, které navštívil. Databáze je mnohem komplexnější než ostatní analyzované databáze a obsahuje údaje, jako jsou například reálné adresy, které uživatel zadal na stránkách elektronických obchodů. Schéma databáze Web Data je k vidění na obrázku 5.2.



Obrázek 5.2: Znázornění databázového schématu *sqlite3* databáze *Web Data* generovaného aplikací *Google Chrome*. Obrázek byl vytvořen prostřednictvím nástroje *Lucid Chart* dostupného online [1].

Následující tabulky obsahují vždy množinu relevantních sloupců databázových tabulek, které vedou k jednoznačné identifikaci uživatele, jejich typ a stručný popis významu.

[1] Nástroj Lucid Chart dostupný z `http://lucidchart.com`

Web Data.autofill		
Název sloupce	**Datový typ**	**Význam**
name	VARCHAR	Název prvku pro automatické doplňování
value	VARCHAR	Obsah vyplněný do prvku
date_created	INTEGER	Unixový čas vytvoření záznamu
date_last_used	INTEGER	Unixový čas posledního použití záznamu
count	INTEGER	Počet použití záznamu

Tabulka 5.11: Tabulka obsahuje podmnožinu sloupců databázové tabulky *autofill* z databáze *Web Data*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

Web Data.autofill_profile_emails		
Název sloupce	**Datový typ**	**Význam**
guid	VARCHAR	Identifikátor profilu k němuž se hodnota váže
email	VARCHAR	Konkrétní email

Tabulka 5.12: Tabulka obsahuje podmnožinu sloupců databázové tabulky *autofill_profile_emails* z databáze *Web Data*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

Web Data.autofill_profile_names		
Název sloupce	**Datový typ**	**Význam**
guid	VARCHAR	Identifikátor profilu k němuž se hodnota váže
first_name	VARCHAR	Jméno vyplňované do položek pro jméno
middle_name	VARCHAR	Jméno vyplňované do položek pro střední jméno
last_name	VARCHAR	Jméno vyplňované do položek pro příjmení
full_name	VARCHAR	Jméno vyplňované do položek pro celé jméno

Tabulka 5.13: Tabulka obsahuje podmnožinu sloupců databázové tabulky *autofill_profile_names* z databáze *Web Data*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

Web Data.autofill_profile_phones		
Název sloupce	**Datový typ**	**Význam**
guid	VARCHAR	Identifikátor profilu k němuž se hodnota váže
number	VARCHAR	Telefonní číslo

Tabulka 5.14: Tabulka obsahuje podmnožinu sloupců databázové tabulky *autofill_profile_phones* z databáze *Web Data*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

5.1.5 Databáze Cookies

Databáze obsahuje informace o objektech *cookies*, které si navštívené webové stránky a aplikace prostřednictvím prohlížeče uložily k uživateli do počítače. Z těchto informací lze pak například určit, které stránky uživatel navštívil v tak daleké minulosti, že je prohlížeč *Google Chrome* již nedrží v paměti adres, na které bylo přistoupeno. Databáze opět ob-

Web Data.autofill_profiles		
Název sloupce	**Datový typ**	**Význam**
guid	VARCHAR	Identifikátor profilu k němuž se hodnota váže
company_name	VARCHAR	Název společnosti
street_address	VARCHAR	Adresa ulice
city	VARCHAR	Město
state	VARCHAR	Stát (kraj, okres)
zipcode	VARCHAR	Poštovní směrovací číslo
sorting_code	VARCHAR	Řadící kód (pro ČR nerelevantní)
country_code	VARCHAR	Kód země
date_modified	INTEGER	Unixový čas poslední změny záznamu
origin	VARCHAR	Adresa webu, ze kterého zaznamenané hodnoty pochází
use_count	INTEGER	Počet použití záznamu
use_date	INTEGER	Unixový čas posledního použití záznamu

Tabulka 5.15: Tabulka obsahuje podmnožinu sloupců databázové tabulky *autofill_profiles* z databáze *Web Data*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

sahuje pouze jednu tabulku, ve které se nachází relevantní data a proto zde není uvedeno plné schéma této databáze. Tabulka 5.16 obsahuje popis a osvětlení významů jednotlivých relevantních sloupců v tabulce cookies, která je obsažena v této databází.

Cookies.cookies		
Název sloupce	**Datový typ**	**Význam**
creation_utc	INTEGER	FILETIME čas vytvoření záznamu
host_key	VARCHAR	Doména, na které je záznam cookies platný
name	VARCHAR	Název konkrétního položky cookies
value	VARCHAR	Hodnota cookie (dnes již nevyužitý sloupec)
path	VARCHAR	Virtuální složka označující platnost cookies
expires_utc	INTEGER	FILETIME čas expirace záznamu
secure	INTEGER	Informace o tom, zda je cookie zabezpečená
last_access_utc	INTEGER	FILETIME čas posledního přístupu k záznamu
has_expires	INTEGER	Informace o tom, zda cookie má nastavenou expiraci

Tabulka 5.16: Tabulka obsahuje podmnožinu sloupců databázové tabulky *cookies* z databáze *Cookies*, zajímavou z hlediska jednoznačné identifikace konkrétního uživatele.

5.2 Návrh programu

Tato sekce se zaobírá kompletním návrhem nástroje pro jednoznačnou identifikaci konkrétního uživatele od fáze volby cílového prohlížeče a implementačního jazyka, přes parametry spouštění po formát výstupu programu.

Vzhledem k tomu, že návrh a implementace společné části finální aplikace již byla popsána v kapitole 3 — *Bezpečné mazání citlivých dat a údajů*, tato část bude práce pojednává výhradně o návrhu a implementaci nástroje pro jednoznačnou identifikaci uživatele a předpokládá čtenářovu znalost zmíněné kapitoly.

5.2.1 Volba cílového prohlížeče a implementačního jazyka

Cílovým prohlížečem, nad kterým nástroj pro identifikaci uživatele operuje je, stejně jako v kapitole 3 prohlížeč *Google Chrome*, pro svou aktuální rozšířenost na trhu [4], stejně jako pro svou multiplatformnost. Přesto, že finální aplikace je psána tak, aby běžela na linuxové distribuci Ubuntu 14.04, je aplikace schopná pracovat i nad daty extrahovanými z jiných systémů, například systému Windows 7.

Nástroj je implementován v jazyce C++ proto, aby ho bylo možné kombinovat s nástrojem vyvinutým v rámci kapitoly 3.

5.2.2 Využití tříd

Pro logické členění programu je vhodné separovat celý program do tříd podle toho, s čím pracují a jaké operace zapouzdřují. Navrhovaný nástroj extrahuje data ze čtyř databází, tedy je vhodné rozdělit program minimálně do čtyř tříd, kde každá z těchto tříd zapouzdřuje operace nad jednou z těchto databází.

Kromě tříd pro operace s konkrétními databázemi je také potřeba mít k dispozici třídu, která bude zaštiťovat samotnou práci s databázemi *sqlite* obecně — třídu, co bude zajišťovat hladké zasílání dotazů k vykonání, reprezentaci výsledků, připojování a odpojování databází a uvolňování použitých prostředků.

Logickou poslední částí, která je u navrhovaného nástroje potřeba, je třída prezentující výsledky pro koncové uživatele. Třída schovávající logiku v pozadí za generováním výstupních souborů v různých formátech.

Využití objektového přístupu v C++ se v kontextu této práce omezuje na využití tříd k zapouzdření související funkcionality a usnadnění orientace v kódu. V případě potřeby rozšířit v budoucnosti aplikaci by bylo možné naimplementovat abstraktní třídy, které by předepisovaly společné metody a vlastnosti tříd pracujících nad konkrétními databázemi.

Další možností rozšíření objektového přístupu by mohlo být neimplementovat zmíněnou abstraktní třídu jako abstraktní, ale jako třídu, která by zaštítila operace nad databázemi a přidala společné metody a vlastnosti pro třídy pracující již s konkrétními databázemi. Tato třída by pak byla těmito třídami děděna. Tímto řešením by pak veškerá funkcionalita pracující s databází byla odstíněna od zbytku kódu a využití objektového přístupu by bylo rozšířeno o dědičnost, teoreticky pak i polymorfismus.

5.2.3 Parametry spuštění

Nástroj pro jednoznačnou identifikaci uživatele spolupracuje s nástrojem pro bezpečné mazání citlivých údajů navrženým v kapitole 3 a proto zde existuje téměř stoprocentní průnik parametrů spuštění pro oba nástroje. Tabulka 5.17

V tabulce oproti tabulce 3.1 chybí u přepínače -t/--target hodnota `purge`, která z pohledu nástroje jednoznačné identifikace uživatele nemá význam.

Zde je nutné podotknout, že pro účel jednoznačné identifikace uživatele ani samotný přepínač -t/--target nemá zcela smysl. Vedlejším produktem, ale stejně důležitým, ne-li více, procesu jednoznačné identifikace uživatele prostřednictvím nástroje *SD4Gen* jest generování výstupních souborů analyzovaných databází, ať už v člověkem čitelné podobě, nebo strojově zpracovatelném formátu. Schůdným kompromisem pro vyřešení problému přebytečnosti přepínače -t/--target je možnost nesnažit se generovat otisk uživatele v případě omezených dat, nad kterými nástroje operuje. Nástroj v případě, že přepínač -t/--target

Parametr (dlouhá verze parametru)	Účel
-h (--help)	Zobrazí nápovědu aplikace
-a (--action)	Nastaví režim běhu aplikace — mazání vs. analýza, hodnota `fingerprinting`
-t (--target)	Výběr dat nad kterými aplikace pracuje, možné hodnoty `all`, `history`, `webdata`, `logindata`, a `cookies`.
-p (--path)	Určuje cestu ke složce aplikačních dat prohlížeče
-o (--output)	Určuje výstupní formát; hodnoty `xml` či `html`.

Tabulka 5.17: Seznam použitelných parametrů a jejich hodnot pro aplikaci SD4Gen v režimu *fingerprinting*

není nastaven na hodnotu `all` generuje na výstup pouze analýzu dat specifikovaných přepínačem -t/--target, ale nesnaží se na jejich základě generovat otisk uživatele.

Jak je napsáno níže v této kapitole, nástroj může generovat více výstupních formátů. Přepínač -o/--output umožňuje vybrat si, jaký formát to bude. Mezi podporovanými formáty jsou formáty `xml` a `html`.

Vzhledem k tomu, že nástroj je určen do rukou vyšetřovatele, a je pravděpodobné, že vyšetřovatel nebude pracovat ve standardním prostředí, kde by se nacházela složka s aplikačními daty na fixním místě v systému, parametrem -p/--path je nutno specifikovat cestu ke složce aplikačních dat. Toto může být obzvlášť vhodné pokud vyšetřovatel vyšetřuje více počítačů zároveň a má u sebe forenzně bezpečné bitové kopie aplikačních dat prohlížečů z různých počítačů. Právě díky tomuto parametru by bylo možné v popsaném případě napsat skript, který by analýzu více aplikačních složek automatizovat a nenutil by vyšetřovatele spouštět skript nad každou složkou zvlášť.

5.2.4 Výstupní formát

U volby výstupního formátu bylo nutné se zamyslet nad tím, jakým způsobem je vhodné prezentovat vyšetřovateli výsledky, kterých nástroj dosáhne. Finální formát musí podléhat následujícím, subjektivně zvoleným podmínkám:

- Snadná reprezentovatelnost napříč různými systémy,
- čitelnost člověkem,
- možnost efektivního strojového zpracování.

Sekundárním kritériem výběru výstupního formátu by měl být také marketingový potenciál — bylo by výhodné, kdyby nástroj produkoval výstup takovým způsobem, aby oslovil i méně technicky znalé zákazníky a byl pro ně přínosem.

5.3 Implementace Nástroje

Tato sekce zahrnuje popis využití objektově orientovaného přístupu při utváření praktické části této práce, je zde popsána role jednotlivých tříd, jejich metod, a zajímavých implementací. Dále jsou zde odděleně popsány metody využívané pro konverzi člověku nečitelných hodnot na hodnoty člověkem čitelné. Mimo to jsou zde popsány další zajímavé implementační techniky, které byly použity. Ke konci této kapitoly je také zmíněna dřívější bezpečnostní chyba aplikace *Google Chrome*, která byla nalezena autorem této práce a jejíž existence byla v rámci implementace nástroje zvážena.

5.3.1 Použití objektově orientovaného přístupu

Nástroj jednoznačné identifikace uživatele se stejně, jako nástroj pro bezpečné mazání dat, odděluje od vstupního bodu aplikace zapouzdřením své funkcionality do příslušných tříd, které jsou z vstupního bodu instanciovány a jejichž metody jsou následně volány.

Oproti nástroji pro bezpečné mazání dat je počet použitých tříd podstatně vyšší, stejně jako jejich účel. Existují zde celkem tři typy tříd:

- Pro přístup k databázi *sqlite* — `SQLiteDriver`,
- pro extrakci konkrétních dat z databází — `HistoryExtractor`, `LoginsExtractor`, `CookiesExtractor` a `WebDataExtractor`,
- a pro reprezentaci výsledků uživateli — `ReportGenerator`.

Třída `SQLiteDriver` je deklarována a definována v souborech `SQLiteDriver.h` a `SQLiteDriver.cpp`. V této třídě existují metody pro otevírání a uzavírání spojení se souborovou databází *sqlite3* a metoda `ExecSQLQuery` přijímající řetězec jako parametr. Řetězec by měl obsahovat validní SQL dotaz, který je následně poslán databázi k níž existuje aktivní spojení.

Všechny metody zmíněné třídy obsahují logiku, která v případě neúspěchu prováděné operace nastavuje interní proměnnou `zErrorMsg` na hodnotu odpovídající příčině neúspěchu. Kontrola úspěchu operace je prováděna skrze interní proměnnou `ResultCode`, která je nastavována na návratovou hodnotu funkcí dostupných z knihovny `sqlite3.h` při jejich volání.

Třída `HistoryExtractor`, která je deklarována a definována ve stejnojmenných *.h a *.cpp souborech již obsahuje metody pro extrakci konkrétních dat z databáze *History*. Metody této třídy volají opět některé z funkcí knihovny `sqlite3.h` nad výsledkem zaslaných SQL dotazů prostředictvím objektu třídy `SQLiteDriver`. Extrahovaná data z databáze jsou ukládána do vektorů (`std::vector<T>`) odpovídajících datových struktur definovaných v hlavičkovém souboru `HistoryTableTypes.h` a předávána dále metodám třídy `ReportGenerator`.

Datové struktury popsané v hlavičkovém souboru reprezentují jeden — velice často omezený — řádek databázové tabulky, ze které jsou data extrahována. Vektor těchto datových struktur pak reprezentuje určitou množinu řádků vybraných sloupců databázové tabulky, případně i spojení tabulek, z níž jsou data vybrána.

Mezi konkrétní data extrahovaná metodami této třídy patří uživatelem ručně zadané adresy do prohlížeče, vyhledávané termíny, seznam stažených souborů, seznamy řetězců url, ze kterých soubor byl stahován, všechny návštěvy v chronologickém pořadí a všechny uložené url.

Na stejném principu a se stejnými jmennými konvencemi, jako je popsáno výše, fungují i ostatní třídy cílící na extrakci dat z konkrétních souborových databází — jmenovitě `CookiesExtractor`, `LoginsExtractor` a `WebDataExtractor`.

Díky malému množství tabulek a možnosti extrahovat záznamy z databází *Login Data* a *Cookies* sestávají třídy `LoginsExtractor` a `CookiesExtractor` pouze z jedné metody, která extrahuje data právě z jejich tabulek.

Třída `ReportGenerator`, která je deklarována a definována ve stejnojmenných *.h a *.cpp souborech obsahuje metody určené k reprezentování extrahovaného obsahu koncovému uživateli. Je zde naimplementována funkcionalita, která umožní z vektorů odpovídajících datových struktur vytvářet člověkus srozumitelný výstup, případně strojově zpracovatelný *xml* výstup.

Jsou zde přítomny metody pro generování *html* a *xml* výstupu a metody, které konvertují člověku nečitelné formáty na formáty čitelné. Těmto metodám je věnovaná část 5.3.2. Dále je zde přítomna metoda, která na základě všech vyextrahovaných informací prezentuje vytvořený otisk identifikovaného uživatele.

5.3.2 Metody pro konverzi hodnot

Nebude určitě žádným překvapením, že některé hodnoty ukládané do databáze jsou ukládané v takovém formátu, že jim stroj rozumí, ale člověk ne. Typicky jsou to časové otisky ukládané jako 64 bitový neznaménkový celočíselný literál, či booleovská hodnota. V kontextu této práce to navíc mohou být hodnoty jako je *transition* extrahovaná z databáze *History*. Pro tyto hodnoty vznikla v procesu práce na nástroji potřeba po metodách, které by tyto hodnoty konvertovaly do čitelné podoby. Tato funkcionalita je implementována v rámci třídy `ReportGenerator` a tato část práce je věnována právě metodám tuto funkcionalitu implementujícím.

Třída obsahuje implementaci celkem sedmi konverzních metod. Nejčastěji používanou metodou je metoda `ConvertToReadableTime`, která na vstupu přebírá 64bitové neznaménkové celé číslo reprezentující počet mikrosekund od 1. 1. 1601. Metoda uvnitř provede převod na klasický unixový čas odečtením konstanty reprezentující rozdíl epoch a na výstup vrací řetězec obsahující *CTime* formát zaznamenaného času, tj. například: „*Sun Apr 10 11:47:20 2016*“.

Další velmi často používanou metodou je `DisplayNiceBoolean`. Tato metoda na svém vstupu přijímá celé číslo, u něhož předpokládá hodnotu `1`, která odpovídá booleovské hodnotě „pravda“, nebo hodnotu `0`, která odpovídá booleovské hodnotě „nepravda“. Tato metoda je volána v této práci v místech, kde je vhodnější reprezentovat pravdu a nepravdu jako „Yes“ a „No“, jsou toto i hodnoty, která metoda poskytuj na svůj výstup.

Při počítání doby, kterou uživatel strávil na stránce je výsledná doba dostána v počtu mikrosekund. Z tohoto důvodu také existuje metoda `ConvertToReadableSeconds`, která přijímá hodnotu v mikrosekundách a vrací hodnotu v sekundách.

Následují čtyři metody, které jsou použity k převodu hodnot uložených v databázi na hodnoty, čitelné člověku, jejichž význam byl vyzkoumán pro účely této práce prostřednictvím empirických postupů a reverzního inženýrství aplikovaného na zdrojový kód aplikace *Google Chrome*. Tyto metody konvertují své hodnoty na základě definovaných `enum` výrazů ve stejné třídě.

Metoda `ConvertToReadableTransition` konvertuje 64bitové neznaménkové celé číslo na řetězec reprezentující název typu přechodu z jedné návštěvy na jinou. Názvy těchto hodnot byly čerpány z Google Chrome dokumentace[1]pro vývojáře rozšíření. Na 64-bitové

číslo je zde nejprve aplikována logická operace `AND` s maskou `0xFF`, čímž se docílí ořezání hodnoty pouze na velikost jednoho bajtu. Pouze spodní bajt této hodnoty je potřeba pro určení typu přechodu a jeho názvu.

Metoda `ConvertToReadableInterruptReason` pracuje velice podobným způsobem, jako předchozí popsaná metoda. Není zde však třeba aplikovat logickou operaci s maskou — získané číslo z databáze již lze porovnat s odpovídajícím `enum` výrazem.

Stejný princip jako metoda `ConvertToReadableInterruptReason` je používán u zbylých dvou metod `ConvertToReadableState` a `ConvertToReadableDangerType`.

5.3.3 Využití „header guards“

Vzhledem k velkému počtu tříd a souborů, ve kterých jsou tyto třídy deklarovány a implementovány, a počtu souborů v nichž jsou deklarovány datové struktury používané těmito třídami, vznikla potřeba zabránit možným kolizím při překladu. Tyto kolize by potenciálně pramenily z špatného použití konstrukce `#include<X>`, kde by docházelo k cyklickým závislostem, nebo nedostatečné viditelnosti (následek zapomenutí `#include<X>`). Z tohoto důvodu v kódu aplikace byly zavedeny tzv. *header-guards*, které brání násobnému vložení stejného hlavičkového souboru.

Tato technika je založena na použití podmíněného definování a vložení knihovny. Existuje-li již konstanta s názvem odpovídajícím hlavičkovému souboru, který je plánováno vložit, soubor není vložen. Naopak, pokud neexistuje definice této konstanty, je soubor vložen a konstanta nadefinována. Tento kód je pak vložen do souborů, v nichž jsou příslušné hlavičkové soubory potřeba. Touto technikou je zabráněno jejich případným kolizím. Ukázka tohoto postupu je uveedena v kódu 5.3.

```
#ifndef LOGINSTABLETYPES_H
#define LOGINSTABLETYPES_H
#include "LoginsTableTypes.h"
#endif

#ifndef WEBDATATABLETYPES_H
#define WEBDATATABLETYPES_H
#include "WebDataTableTypes.h"
#endif
```

Kód 5.3: Příklad použití tzv. „header guards“ v kódu aplikace *SD4Gen*.

5.3.4 Bezpečnostní chyba aplikace Google Chrome

Okolo roku 2013 se autor této práce věnoval bližšímu průzkumu aplikačních dat aplikace *Google Chrome*. V rámci tohoto průzkumu byla objevena potenciální bezpečnostní díra ve způsobu, kterým prohlížeč žurnáloval důležité informace, včetně databází, které jsou v rámci této práce analyzovány. Chyba způsobovala, že když uživatel smazal skrze grafické rozhraní prohlížeče svou historii, žurnálový soubor historie zůstal přítomen včetně záznamů, které měly být odstraněny. Chyba se projevovala i při mazání pomocí dalšího, externího software,

[1]Google Chrome dokumentace pro vývojáře rozšíření dostupná z: `https://developer.chrome.com/extensions`

jako je CCleaner[1]. Tato informace byla získána od uživatele webového portálu reddit.com[2] *HeloRising*. V současnosti je tato chyba již opravena — otestováno autorem této práce.

Pokud by se ovšem uživatel nástroje implementovaného v rámci této práce dostal k verzi prohlížeče, která touto chybou stále trpěla, je možné provést i analýzu tohoto souboru pro získání nepozměněné historie prohlížení.

5.3.5 Ukázkové výstupy

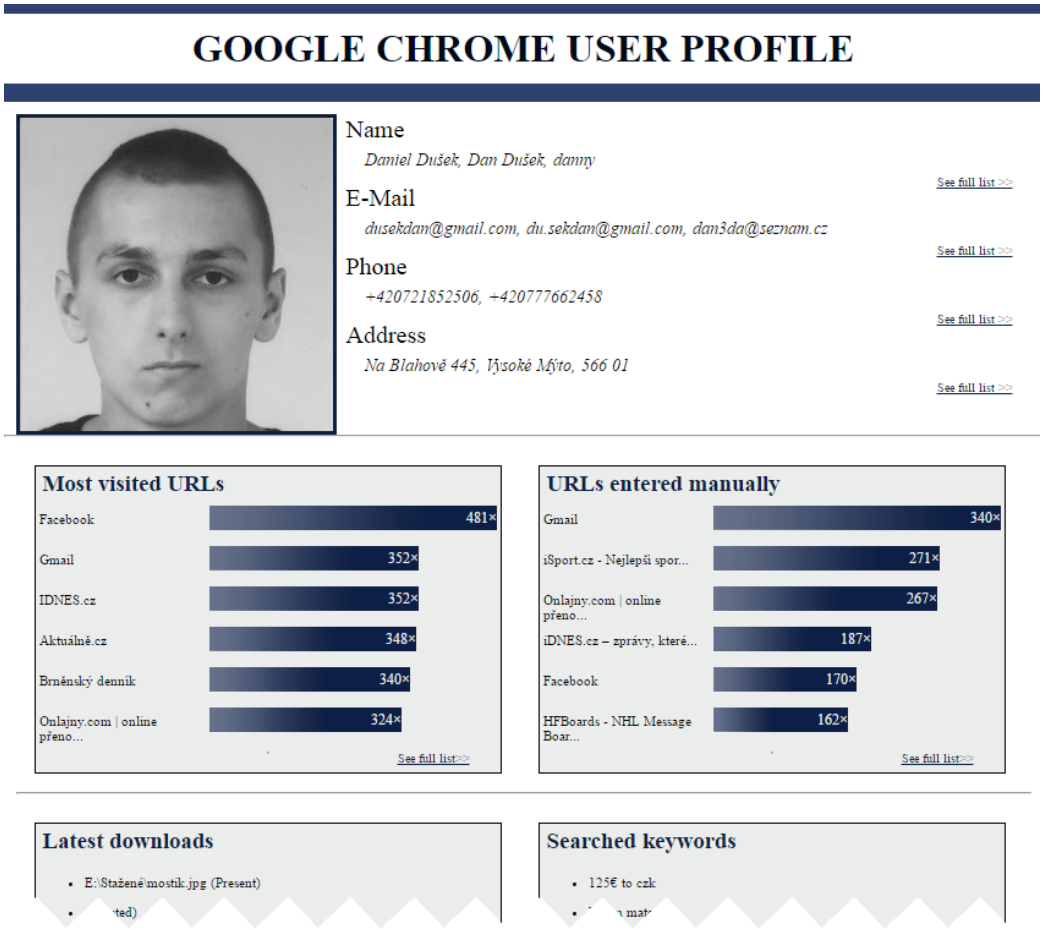
Tato část obsahuje ukázky výstupů, které mohou být vyprodukovány spuštěním aplikace v režimu *fingerprinting*.

```
<?xml version="1.0" encoding="UTF-8"?>
<cookies>
<Record>
<HostKey>.180481574.log.optimizely.com</HostKey>
<Name>end_user_id</Name>
<Value>[Encrypted]</Value>
<Path>/</Path>
...
<LastAccessUtc>Sun Apr 10 11:47:20 2016</LastAccessUtc>
</Record>
<Record>
...
</Record>
...
</cookies>
```

Kód 5.4: Zkrácený ukázkový xml výstup pro analyzovanou databázi *Cookies*.

Mimo *xml* výstup produkovaný příslušným nastavením přepínače -o/--output, lze v této části najít také produkovaný výstup při nastavení přepínače na HTML. Obrázek 5.5 zobrazuje možný vzhled výstupu programu při volání programu v režimu pro jednoznačnou identifikaci uživatele.

[1] *CCleaner* software, ke stažení na `https://www.piriform.com/ccleaner/download`
[2] *reddit: the front page of the internet*, dostupné z `http://reddit.com`



Obrázek 5.5: Ukázkový výstup nástroje pro jednoznačnou identifikaci uživatele.

5.3.6 Metriky kódu

Závěrem této kapitoly jsou uvedeny základní metriky kódu souborů týkajících se implementace nástroje pro jednoznačnou identifikaci uživatele.

Počet souborů se zdrojovými kódy: 28
Počet řádků kódu napříč soubory: 4 444
Velikost souborů: 92.4 kB

Kapitola 6

Testování aplikace SD4Gen

Kapitola pojednává o testování obou vyvinutých nástrojů. První část této kapitoly je věnována testování nástroje pro bezpečné mazání citlivých dat, a to na různých souborových systémech, na kterých by nástroj měl pracovat. Druhá část kapitoly pak hovoří o testování nástroje pro jednoznačnou identifikaci uživatele.

Proces vývoje obou nástrojů byl v jejich počátku veden stylem „*test-driven development*“, kdy nejprve byly napsány testy na ještě neimplementované části aplikace a až následně byly části pokryté testy implementovány. Tento postup byl časem vyměněn za programování částí na základě akceptačních kritérií. Důvodem této změny byla jednak časová tíseň a druhak vzrustající obtížnost pokrývání nové funkcionality testy před samotnou implementací.

6.1 Testování bezpečného mazání

Aplikace bezpečného mazání využívá existující funkcionality systémové funkce `shred`. Funkce je schopná dle dokumentace operovat úspěšně nad souborovými systémy, které nepoužívají žurnálování. V kontextu této práce jsou zajímavé souborové systémy *FAT32, ext2* a *ext3 s základním nastavení žurnálování*. Vzhledem k tomu, že tato funkce je dobře dokumentována a je součástí tzv. *GNU Coreutils*, je možné předpokládat, že slibovaná funkcionalita je opravdu doručena a nemá význam ji dále testovat.

Co však testovat má smysl, je správná integrace této funkce do nástroje pro bezpečné mazání citlivých dat. V této sekci je popsán postup a výsledek testování jednotlivého selektivního mazání údajů, mazání všech cílů a mazání s přepínačem *purge*. Následně je testováno použití nástroje k jiným účelům, než je zamýšlené použití.

6.1.1 Testování selektivního mazání

Pro testování selektivního mazání byla vykopírována adresářová struktura z aplikační složky *Google Chrome* a umístěna do stejného adresáře, ve kterém se nachazí spustitelný soubor *SD4Gen*. V kódu, kde dochází k volání systémové funkce *shred* byl přidán přepínač `-v`, který způsobí, že funkce průběžně hlásí jak postupuje se svou činností.

Dalším krokem bylo postupné spouštění aplikace *SD4Gen* s argumentem přepínače -p/--path nastaveným na cestu k vykopírovanému adresáři, akcí nastavenou na hodnotu `deletion` a s každým spuštěním jiným nastaveným cílem. Po každém spuštění byla provedena kontrola výstupu funkce *shred* a byla manuálně kontrolována přítomnost souboru, který byl nastaven cílem.

Ve všech případech výstup odpovídal předpokladu. Aplikace tímto testem tedy prošla.

6.1.2 Testování mazání všech cílů

Pro tento test došlo ke stejné přípravě prostředí, jako v předchozím případě — adresářová struktura byla znovu překopírována do složky s aplikací *SD4Gen*. Rozdílem oproti předchozímu testu byl argument přepínače určujícího cíl. Zde byla použita hodnota `all`. Hodnota by měla způsobit smazání všech souborů, které jde mazat selektivně, ale najednou. Mimo souborů mazatelných selektivně by nástroj měl nyní smazat i soubor „*Google Profile Picture.png*“ nacházející se ve stejném adresáři.

Opět byly zkontrolovány výstupy funkce *shred* a nepřítomnost souborů v jejich původních umístěních. Aplikace opět tímto testem prošla.

Pokročilejší verzí tohoto scénáře byla úprava aplikační složky tak, že některé databáze měly svůj žurnálový soubor a některé ne. Některé databáze byly odstraněny a některé ponechány. Nástroj se v tomto případě pokusil smazat existující soubory a u neexistujících nahlásil jejich absenci. Tímto rozšířeným testem aplikace taktéž prošla.

6.1.3 Testování s přepínačem cíle *purge*

Otestování přepínače cíle s hodnotou `purge` vycházelo ze stejného prostředí jako předchozí testy. Předpokladem výstupu této operace je odstraněná celá adresářová struktura, nad kterou je nástroj volán.

Po spuštění nástroje nad připraveným prostředím byl opět zkontrolován výstup *shred* funkce a existence adresářové struktury v příslušném místě. Aplikace tímto testem opět prošla. Je však nutné poznamenat, že tato operace trvá delší dobu, než všechny výše testované případy dohromady.

6.1.4 Testování nezamýšleného použití nástroje

Během testování přepínače s cílem nastaveným na hodnotu `purge` bylo odhaleno chování, které nebylo při původním návrhu zamýšleno. Při této hodnotě přepínače cíle a **jakékoliv** cestě poskytnuté přepínači -p/--path je rekurzivně smazána poskytnutá cesta. Toto chování je výsledkem toho, že nástroj nekontroluje, zda opravdu pracuje nad adresářovou strukturou aplikační složky *Google Chrome*.

Ačkoliv by se toto mohlo zdát jako slabina a chyba nástroje, nemusí tomu stoprocentně tak opravdu být. V případě, že uživatel již zkoušel smazat aplikační složku sám a ručně, ale nebyl úspěšný, může být nemožné poznat, zda jde o aplikační složku aplikace *Google Chrome*. V takovém případě by nástroj při rozpoznání, že nejde o podporovanou adresářovou strukturu neumožnil uživateli složku přemazat. Toto je dle autorova přesvědčení nežádoucí chování.

6.2 Testování nástroje pro jednoznačnou identifikaci

U nástroje pro jednoznačnou identifikaci uživatele je možné testovat hned několik podporovaných chování a scénářů. Začátkem této části práce se hovoří o možnosti analyzovat aplikační složky aplikace *Google Chrome* generované na systému Windows a složky generované pod systémem Linux. Následně jsou zmíněny testy a jejich výsledky při testování nad reálnými daty existujících uživatelů, závěru této části se pak nachází test na uměle vytvořených datech.

V tento moment je nutno podotknout, že získání reálných dat pro testování a demonstraci je poměrně obtížný úkol. Tento fakt je způsoben náturou dat, které aplikace pro otestování své činnosti vyžaduje. Jde o soukromé informace o prohlížení konkrétního uživatele a mnohdy i o jeho důvěrná data jako jsou jméno, telefonní číslo, email a další.

6.2.1 Testování na aplikační složce generované ve Windows

Pro otestování schopnosti nástroje operovat nad aplikační složkou generovanou pod systémem Windows je dostatečným testem úspěšné spuštění nástroje nad touto složkou. V případě, že celý proces proběhne v pořádku, lze předpokládat, že nástroj je schopen analyzovat i složky generované aplikací pod Windows.

Níže popsaný scénář 6.2.4 byl proveden právě nad složkou generovanou na systému Windows. Proces analýzy zajímavých dat proběhl v pořádku a proto lze tvrdit, že aplikace prošla tímto testem.

6.2.2 Testování na aplikační složce generované v Linuxu

Testování na aplikační složce generované v linuxovém systému (konkrétně šlo o systém Ubuntu 14.04) bylo prováděno v průběhu celého vývoje nástroje, neboť vývoj probíhal právě na linuxovém systému. Za stejných podmínek pro splnění testu, jako byly uvedeny u testu 6.2.1, je možné tvrdit, že aplikace tímto testem opět prošla.

V rámci tohoto testu bylo prokázáno, že aplikace je schopná pracovat jak nad prohlížečem *Google Chrome*, tak nad jeho open-source alternativou *Google Chromium*.

6.2.3 Test nad reálnými daty — Uživatel 1

Kromě testování, zda aplikace proběhne správně a správně skončí nad daty z různých systémů, je dalším kritériem k otestování schopnost nástroje identifikovat konkrétního uživatele.

Aplikace byla spuštěna v režimu `fingerprinting` nad aplikační složkou obsahující data uživatele. Výstupem tohoto testu je několik HTML souborů popisujících nejpravděpodobnějšího uživatele prohlížeče, z něhož data pochází. Tento test správně identifikoval uživatele Tomáš Kaplan, z jehož prohlížeče složka aplikačních dat pocházela.

Pro tento test byly získány data od pana Tomáše Kaplana, kterému tímto autor práce děkuje.

6.2.4 Test nad reálnými daty — Uživatel 2

Za stejných podmínek a okolností jako v předchozím případě byl proveden test nad reálnými daty poskytnutými panem Davidem Říhou, kterému taktéž tímto autor práce děkuje. Naneštěstí bylo nutno velkou část dat zcenzurovat, a to vzhledem k prohlížecím návykům původce dat.

Nástroj byl opět schopný identifikovat a vyprodukovat očekávaný výstup identifikující původce dat.

6.2.5 Test nad reálnými daty — Uživatel 3

Pro reálný nedostatek dat byl autor práce nakonec nucen otestovat nástroj i nad vlastními daty. Aby test nevycházel ze stejných dat, nad kterými bylo vyvíjeno, jako kompromis byla provedena extrakce dat z prohlížeče umístěného na počítači, ze kterého autor vykonává

výhradně pracovní záležitosti. Pro tato data základní test na otisk uživatele z jeho aktivity v prohlížeči opět prošel.

Jako rozšířený test byla provedena extrakce do formátu XML, kdy není generován HTML otisk uživatele, ale XML výstup obsahující data vyextrahovaná z databází. Tato extrakce proběhla v pořádku a výstupy odpovídaly očekávání. Aplikace tímto testem prošla.

6.2.6 Test nad uměle vytvořenými daty

V poslední řadě bylo třeba otestovat chování programu při hraničních hodnotách. Pro tento účel byla simulávana situace při které více lidí používá stejný prohlížeč a následně jsou jeho data analyzována.

Tento test odhalil, že aplikace v takovém případě není schopná jednoznačně určit, která konkrétní osoba používala prohlížeč a nedojde k jednoznačné identifikaci. Výsledný otisk obsahuje jména nejaktivnějších uživatelů a podpůrná data (mimo hlavní stránku otisku) obsahují pak kompletní seznam aktivity všech uživatelů. Přestože nedojde i jednoznačné identifikaci, analyzovaná data mají stále hodnotu, neboť obsahují záznam o uživatelské činnosti uvnitř prohlížeče.

Celkovým závěrem tohoto testu je, že aplikace není schopná identifikovat konkrétního uživatele, pokud data vyprodukovaná prohlížečem obsahují informace o více uživatelích, kterými byl používán.

6.3 Zhodnocení výsledků testování

Nástroje implementované v aplikaci *SD4Gen* byly testovány odděleně a nad reálnými i uměle vytvořenými daty.

Výsledky testů provedených na nástroji pro bezpečné mazání citlivých dat byly výborné a naznačují, že tento nástroj je použitelný pro účel k němuž byl vyvinut. Testování nástroje mimo jiné odhalilo, že ho je možné použít i pro jinou, než originálně zamýšlenou činnost, což je autorem chápáno spíše jako vyšší flexibilita nástroje, než jako jeho chyba.

Testy provedené nad nástrojem pro jednoznačnou identifikaci uživatele byly ovlivněny nepříjemným faktorem, a to nedostatkem reálných dat. Tento problém není možné jednoduchým způsobem řešit, neboť řešení zahrnuje žádání testovacích subjektů o jejich soukromé a důvěrné informace, přičemž možnost anonymního poskytnutí těchto dat je vyloučená.

I přes nedostatek dat pro testování nástroje pro jednoznačnou identifikaci uživatele, bylo možné tyto testy díky ochotě přátel autora práce provést. Z provedených testů vyplývá, že aplikace je schopná identifikovat uživatele, pokud jeho aplikační data obsahují dostatek identifikujících údajů. Rozšířeným testem nad uměle vytvořenými daty bylo dále odhaleno, že nástroj neidentifikuje úspěšně konkrétního uživatele, pokud prohlížeč používalo více lidí a stopy jejich aktivity jsou přítomny v aplikačních datech.

Z výsledků testování obou nástrojů aplikace *SD4Gen* vyplývá, že aplikace je použitelná pro účel, pro který byla navržena.

Kapitola 7

Závěr

Práce se zabývala návrhem a implementací nástroje pro forenzně bezpečné mazání citlivých údajů a souborů z webového prohlížeče *Google Chrome*, a návrhem a implementací nástroje pro jednoznačnou identifikaci uživatele na základě jeho aktivity v tomto prohlížeči. Nástroje byly spojeny v aplikaci *SD4Gen*, která umožňuje provádět obě požadované operace. Součástí této práce je seznámení s technikami a metodami, které ohrožují forenzní vyšetřování, a také odhad, jaké techniky jednoznačné identifikace uživatele budou použitelné v budoucnosti.

Hlavní přínos této práce spočívá v implementaci nástroje, který kromě extrakce důležitých dat pro případné forenzní vyšetřování, produkuje otisk pravděpodobného uživatele v člověku čitelné podobě. Tato produkovaná člověku čitelná forma usnadňuje prvotní odhad identity podezřelého — vyšetřovatel se tak nemusí nejprve probírat masou vyextrahovaných dat, aby určil potenciálního pachatele. Dalším přínosem této práce je zdokumentování hodnot ukládaných do databází generovaných prohlížečem *Google Chrome* a vyzkoumání významů těchto hodnot. V poslední řadě pak přináší běžnému uživateli možnost neobnovitelně mazat citlivá data z prohlížeče.

V průběhu tvorby této práce byla také odhalena bezpečnostní díra v aplikaci *Google Chrome*. Tato chyba je v současné době již opravena, avšak je v implementaci aplikace zohledněna a je jí využito.

Testování nástrojů implementovaných touto prací prokázalo, že je možné je použít pro forenzně bezpečné mazání citlivých dat a údajů z prohlížečů *Google Chrome* a *Google Chromium*. Dále bylo prokázáno, že je možné pomocí nástroje pro jednoznačnou identifikaci uživatele identifikovat konkrétního uživatele, za předpokladu, že po sobě zanechal dostatek stop při používání prohlížeče.

Předpokládaný budoucí vývoj této aplikace rozšíří repertoár podporovaných prohlížečů z *Google Chrome* a open-source alternativy *Google Chromium* o další prohlížeče, jako jsou například *Firefox* popřípadě *Safari*. Jinou možností budoucího vývoje je pak zavedení podpory nových výstupních formátů a umožnění identifikace více uživatelů používajících jeden prohlížeč.

Tato práce byla prezentována na Studentské konferenci inovací, technologií a vědy v IT, Excel@FIT 2016.

Literatura

[1] Delete your browsing history — Computer — Chrome Help [online]. Dostupné z https://support.google.com/chrome/answer/95589, 2009 [cit. 2016-01-23].

[2] SANS Digital Forensics and Incident Response Blog | Google Chrome Forensics | SANS Institute [online]. Dostupné z https://digital-forensics.sans.org/blog/2010/01/21/google-chrome-forensics, 2010-01-21 [cit. 2016-04-10].

[3] chrome.history — Google Chrome [online]. Dostupné z https://developer.chrome.com/extensions/history, 2011 [cit. 2016-01-23].

[4] Browser statistics [online]. Dostupné z http://www.w3schools.com/browsers/browsers_stats.asp, 2015 [cit. 2016-04-10].

[5] The GNU C Reference Manual [online]. Dostupné z http://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html, 2015 [cit. 2016-04-10].

[6] GNU Coreutils: shred invocation [online]. Dostupné z https://www.gnu.org/software/coreutils/manual/html_node/shred-invocation.html, 2015 [cit. 2016-05-12].

[7] Ubuntu documentation [online]. Dostupné z https://help.ubuntu.com/community/LinuxFilesystemsExplained, 2016 [cit. 2016-01-23].

[8] Baker, A.: Top 16 Pieces of Your Information Identity Thieves Crave [online]. Dostupné z http://www.manvsdebt.com/top-16-pieces-of-your-information.identity-thieves-crave/, August 2009-08-01 [cit. 2016-04-10].

[9] Bechar-Israeli, H.: FROM Bonehead TO cLoNehEAd: NICKNAMES, PLAY, AND IDENTITY ON INTERNET RELAY CHAT [online]. Dostupné z https://www.mediensprache.net/archiv/pubs/2035.html, 2009-08-01 [cit. 2016-04-10].

[10] Clarke, N.: *Computer forensics a pocket guide.* Ely: IT Governance Pub, 2010 [cit. 2016-01-23], ISBN 978-184-9280-402.

[11] Gutmann, P.: Secure Deletion of Data from Magnetic and Solid-State Memory. In *Proceedings of Sixth USENIX Security Symposium: Focusing on Applications of Cryptography*, editace T. editor, 6, USENIX, San Jose, California: USENIX, 1996 [cit. 2016-05-12], s. 77–90.

[12] Hughes, G. F.; Coughlin, T.; Commins, D. M.: Disposal of Disk and Tape Data by Secure Sanitization. *IEEE Security & Privacy*, ročník 7, č. 4, 2009 [cit. 2016-01-23]: s. 29–34, ISSN 1540-7993, doi:10.1109/MSP.2009.89.

[13] Lischner, R.: *C++ In a Nutshell. A Language & Library Reference*. Sebastopol, CA: O'Reilly Media, 5 2003 [cit. 2016-04-10], ISBN 978-0-596-00298-5.

[14] Oh, J.; Lee, S.; Lee, S.: Advanced evidence collection and analysis of web browser activity. *Digital Investigation*, ročník 8, Supplement, 2011 [cit. 2016-01-23]: s. S62 – S70, ISSN 1742-2876, doi:10.1016/j.diin.2011.05.008, the Proceedings of the Eleventh Annual {DFRWS} Conference11th Annual Digital Forensics Research Conference.

[15] Palmer, G.: Final Ed. Technical Report T-001-01 [online]. Air Force Research Laboratory, Rome Research Site. Dostupné z http://www.dfrws.org/2001/dfrws-rm-final.pdf, 2008-11-01 [cit. 2016-01-23].

[16] Shirk, E.: Law Practice Today: The Dangers of Do-It-Yourself Computer Forensics [online]. Dostupné z https://apps.americanbar.org/lpm/lpt/articles/tch11071.shtml, 2007 [cit. 2016-01-23].

Přílohy

Seznam příloh

Příloha A

Obsah DVD

Na přiloženém optickém médiu je k dispozici virtuální stroj pro virtualbox s předinstalovaným Ubuntu 14.04, testovacími daty, aplikací SD4Gen a jejími zdrojovými kódy. Uživatelské jméno pro přístup do systému je „prezentace“, heslo „prezentace2016“.

Mimo obraz virtuálního stroje je na CD soubor ReadMe.txt obsahující informace pro práci se soubory obsaženými na médiu.

Příloha B

Manuál k použití aplikace SD4Gen

Aplikace je určena k běhu pod operačním systémem Linux, testováno a optimalizováno pro Ubuntu 14.04. Na přiloženém DVD k této práci by měl být dostupný obraz tohoto systému s nainstalovanými prostředky k tomu, aby aplikace bezproblémově běžela.

B.1 Kompilace a překlad

Na ploše obrazu systému z přiloženého DVD je přítomna složka *Sources*, ve které se nachází zdrojové kódy aplikace. Pro překlad je nutné otevřít terminál v této složce a příkazem `make` přeložit projekt.

B.2 Použití nástroje

Program je třeba spouštět na *vykopírované* složce „*Default*“ ze složky s aplikačními daty prohlížeče *Google Chrome*. Program by neměl být spouštěn přímo nad „živou“ složkou aplikačních dat.

K dispozici je několik přepínačů, se kterými lze program spustit. Hlavním přepínačem je přepínač -a/--action, který může nabývat dvou hodnot `deletion` a `fingerprinting`. První hodnota říká, že program bude puštěn v režimu mazání citlivých dat, druhá hodnota pouští program v režimu identifikace konkrétního uživatele.

Dalším přepínačem je přepínač -t/--target, který může nabývat několika hodnot, které určují cíl operace zvolené přepínačem -a:

- `history` — cílem je soubor databáze *History*; přepínač je dostupný pro oba dva režimy aplikace,
- `webdata` — cílem je soubor databáze *Web Data*; přepínač je dostupný pro oba dva režimy aplikace,
- `cookies` — cílem je soubor databáze *Cookies*; přepínač je dostupný pro oba dva režimy aplikace,
- `logindata` — cílem je soubor databáze *Login Data*; přepínač je dostupný pro oba dva režimy aplikace,
- `purge` — cílem je celá adresářová struktura uživatelských dat předaná parametrem -p/--path; tento přepínač je dostupný pouze v režimu bezpečného mazání dat.

Přepínač -p/--path slouží k nastavení cesty ke složce, jejíž data budou analyzovány/mazány. Název složky, která je za posledním lomítkem v adrese by měl být „*Default*“.

Poslední dostupný přepínač je -o/--output, kterým lze nastavit, zda má být produktem pokusu o jednoznačnou identifikaci HTML či XML výstup. Povolené hodnoty jsou `html` a `xml`.

Příklad spuštění programu pro bezpečné mazání:

```
./SD4Gen -a deletion -t purge -p ../google-chrome/Default
```

Příklad spuštění programu pro jednoznačnou identifikaci uživatele:

```
./SD4Gen -a fingerprinting -t all -p ../google-chrome/Default
```

Generovaný výstup programu je pak k nalezení ve složce „*Out*“ ve stejném adresáři jako je celý projekt.