FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Analýza obsahu uživatelských profilů ve webových prohlížečích
Student:	Stanislav Lepič
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2021/2022

Pokyny pro vypracování

1. Seznamte se s hlavními webovými prohlížeči a zpracujte rešerši údajů, které v rámci svých uživatelských profilů ukládají.
2. Porovnejte podobnosti a odlišnosti v přístupu jednotlivých prohlížečů. Zaměřte se při tom zejména na uložená hesla, případně další bezpečnostně citlivé položky.
3. Navrhněte vhodný formát souboru pro přenos údajů mezi profily různých prohlížečů.

4. Vytvořte základ aplikace pro off-line přenos profilových údajů mezi prohlížeči. Soustřeďte se na snadnou rozšiřitelnost o další prohlížeče a další položky profilu.

5. Otestujte přenos profilu mezi jednotlivými prohlížeči, v případě dílčích potíží diskutujte jejich příčiny a možná řešení.

6. Diskutujte své výsledky.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 29. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Analýza obsahu uživatelských profilů ve webových prohlížečích

Stanislav Lepič

Katedra počítačových systémů
Vedoucí práce: Ing. Josef Kokeš

13. května 2021

Poděkování

Děkuji svému vedoucímu Ing. Josefu Kokešovi za všechny cenné rady a vedení při vytváření této práce. Děkuji všem svým blízkým, kteří mě ve studiu podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2021 . . . . . .

Odkaz na tuto práci

Lepič, Stanislav. *Analýza obsahu uživatelských profilů ve webových prohlížečích*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato práce se zabývá převodem uživatelských dat mezi prohlížeči, a to jak v rámci jednoho, tak i několika různých zařízení. Práce analyzuje způsob ukládání uživatelských dat a jejich šifrování se zaměřením na prohlížeče Google Chrome a Mozilla Firefox. Na základě této analýzy je následně navržen přenosový soubor ve formátu JSON, který využívá vytvořený program pro demonstraci funkčnosti mezi dvěma zvolenými prohlížeči. Program je vytvořen v jazyce C/C++ a umožňuje převod v rámci jednoho zařízení, offline převod mezi zařízeními, nebo pouze extrakci dat z prohlížeče. Při návrhu programu byl kladen důraz na snadné ovládání a rozšiřitelnost o další prohlížeče, nebo druhy dat.

Klíčová slova webový prohlížeč, převod dat, uživatelské údaje, přístupová hesla, šifrování, Mozilla Firefox, Google Chrome, Windows

Abstract

This thesis deals with the transfer of user data between browsers, both within one and several different devices. The work analyzes how user data is stored and encrypted, focusing on Google Chrome and Mozilla Firefox browsers. Based on this analysis, a transfer file in JSON format is then designed, which uses the created program to demonstrate the functionality between the two selected browsers. The program is created in C/C ++ and allows conversion within one device, offline conversion between devices or just extracting data from the browser. When designing the program, emphasis was placed on easy operation and extensibility with other browsers or data types.

Keywords web browser, data transfer, user data, passwords, encryption, Mozilla Firefox, Google Chrome, Windows

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Webové prohlížeče dnes patří mezi nejpoužívanější programy vůbec. Pro drtivou většinu uživatelů prohlížeč znamená internet, na kterém denně využívají mnoho nejrůznější služeb, pro něž je třeba si pamatovat velké množství přihlašovacích údajů. Z tohoto důvodu je možné nechat prohlížeč tyto údaje zapamatovat a uživatelé této možnosti rádi využívají. Webové prohlížeče se však neustále vyvíjí, a to nejen ve způsobu, kterým všechna uživatelská data ukládají, ale hlavně v nabízených funkcích. Problém pak nastává, pokud se uživatel rozhodne přejít na jiný prohlížeč nebo chce svá data převést na jiné zařízení. Většina prohlížečů nabízí určitá řešení, ovšem ta skrývají nejrůznější úskalí.

Práce se zabývá analýzou uživatelských profilů ve webových prohlížečích, tedy otázkou jaká data a v jakém formátu prohlížeče ukládají. Na základě porovnání těchto dat je navržen univerzální formát, který slouží jako mezikrok pro jejich převod. Tento formát je následně využit při tvorbě aplikace.

Výsledek této práce je určen všem uživatelům, kteří řeší přenos dat mezi webovými prohlížeči, ale stávající řešení jim nedostačují. Dále pak těm, kteří přechází na nové zařízení, ale např. nemají zájem vytvářet si v prohlížeči speciální účet pro online synchronizaci dat mezi zařízeními.

Analytická část se zabývá principy ukládání uživatelských údajů v jednotlivých prohlížečích. Jsou zde diskutovány stávající možnosti převodu dat a jejich nedokonalosti. Dále jsou zde popisovány soubory a formáty, ve kterých jsou data ukládána, a jaké bezpečností mechanismy a šifrovací algoritmy jsou využívány. Analýza se zaměřuje na přihlašovací údaje, historii, záložky a soubory cookies. U těchto dat je porovnáváno, ze kterých údajů se skládají, na základě čehož je následně navržen formát pro jejich přenos.

Praktická část se věnuje tvorbě desktopové aplikace v jazyce C/C++ určené pro systém Windows. Funkčnost je demonstrována na dvou prohlížečích – Google Chrome a Mozilla Firefox. Vývoj je zaměřen hlavně na snadnou rozšiřitelnost pro další prohlížeče a jednoduché ovládání.

Kapitola 1

Analýza stávajícího řešení

Analýza převodu dat se zaměřuje na čtyři aktuálně nejpoužívanější prohlížeče pro OS Windows 10, a to na Google Chrome, Microsoft Edge a Operu, založených na jádru Chromium, a také na Mozillu Firefox založenou na jádru Gecko. Značné zastoupení na poli prohlížečů má také Safari, ale jeho vývoj byl pro OS Windows v roce 2012 ukončen. [1] Stav je popisován k akutálním verzím prohlížečů, tedy Google Chrome 90.0.4430, Mozilla Firefox 88.0, Microsoft Edge 90.0.818.46 a Opera 75.0.3969.218.

1.1 Převod mezi prohlížeči v rámci jednoho zařízení

Pro převod uživatelských dat mezi prohlížeči aktuálně neexistuje žádné univerzální řešení. Google Chrome, Mozilla Firefox a Microsoft Edge nabízí možnost převodu dat mezi sebou, nebo také import ze zastaralé verze prohlížeče Microsoft Edge (před jeho přechodem na jádro Chromium), případně z Internet Exploreru. Import dat z Opery však nenabízí ani jeden. Opera pak umožňuje import z ostatních sledovaných prohlížečů a z Internet Exploreru.

Problém při převodu přihlašovacích údajů z Mozilly Firefox do jiného prohlížeče nastává ve chvíli, kdy má uživatel nastaveno tzv. hlavní heslo. To slouží jako přidaná vrstva bezpečnosti a je vyžadováno ve chvíli, kdy má Mozilla Firefox poskytnout uložené přihlašovací údaje. Všechny analyzované prohlížeče založené na Chromium si s hlavním heslem nedokážou poradit a funkce importu zde tak nefunguje. Problém je o to horší, že se všechny prohlížeče chovají jakoby přenos bez problému proběhl, včetně vypsání hlášky potvrzující úspěšné přenesení. Řešením může být dočasné odebrání hlavního hesla a provedení importu. Běžný uživatel však nemá možnost se ze strany prohlížeče o tomto problému, nebo dokonce jeho řešení dozvědět.

1.2 Online převod dat mezi zařízeními

Všechny čtyři výše jmenované prohlížeče nabízejí synchronizaci uživatelských dat pomocí uživatelského účtu. Google Chrome využívá svůj Google účet. Mozilla Firefox tuto službu nazývá Firefox Sync a vyžaduje zřízení speciálního účtu. Nejvíce možností nabízí Microsoft Edge, kde je možno využít pracovní nebo školní účet od Microsoftu, případně běžný Microsoft účet, nebo dokonce Skype účet. Opera pak také nabízí vlastní službu Opera Sync, kde synchronizace probíhá pomocí vlastního účtu. Navíc však podporuje synchronizaci mezi desktopovou a mobilní verzí prohlížeče, např. naskenováním QR kódu.

1.3 Offline převod dat mezi zařízeními

Offline převodem dat myslíme možnost vyexportovat data z prohlížeče do souboru, přenést ho na jiné zařízení a opět importovat zpět. Existuje velká řada programů a rozšíření pro prohlížeče nabízející pouze export dat, to ovšem není cílem této práce.

1.3.1 Přihlašovací údaje

Mozilla Firefox nabízí export přihlašovacích údajů do čitelné podoby ve formě *CSV* souboru. Tento soubor je pak možné importovat zpět, nicméně z důvodu výkonnostních problémů je tato možnost dočasně skryta. Pro aktivaci importu tohoto souboru je tedy třeba ručně přepnout příslušnou volbu v nastavení. [2]

Prohlížeče založené na Chromiu shodně umožňují export do velmi jednoduchého *CSV* souboru obsahujícího pouze jméno stránky, url, přihlašovací jméno a heslo. Import zpět do prohlížeče je v nich pak možné zapnout v experimentálních funkcích. Díky tomu tedy lze mezi těmito prohlížeči volně přenášet přihlašovací údaje offline.

1.3.2 Záložky

Záložky je možné ve všech sledovaných prohlížečích exportovat do formátu *HTML* a z tohoto formátu je do libovolného prohlížeče importovat zpět. Jsou to tedy jediná data, která se ukládají ve všech prohlížečích do stejného formátu, díky čemuž je možné s nimi snadno manipulovat.

1.3.3 Historie a cookies

Žádný ze sledovaných prohlížečů nenabízí možnost exportu nebo importu historie nebo souborů cookies.

Kapitola 2

Analýza souborů

Struktura uživatelských dat, kterými myslíme přihlašovací údaje, historii, záložky a soubory cookies, je mezi prohlížeči s různými jádry poměrně odlišná, a to hlavně v podrobnostech, které pro jednotlivá data ukládají. Pro demonstraci funkčnosti byl vybrán aktuálně nejrozšířenější webový prohlížeč stavějící na jádru Chromium – Google Chrome, jehož aktuální zastoupení na poli prohlížečů se odhaduje na 64 %. [3] Jako druhý prohlížeč byla zvolena Mozilla Firefox, která z nejpoužívanějších prohlížečů pro OS Windows jako jediná staví na odlišném jádru jménem Gecko. Prohlížeče Microsoft Edge a Opera nebyly vyhodnoceny jako vhodné, neboť staví na jádru Chromium, a struktura jejich uložených údajů je tak v mnoha ohledech stejná jako v Google Chrome.

Všechny analyzované databázové soubory jsou ve formátu *sqlite*. Pro práci s nimi byl použit nástroj *DB Browser for SQLite*[a]. Pro práci se soubory ve formátu *JSON* byl použit program *JSON Viewer*[b]. Data ve formátu *ASN.1* byla zkoumána pomocí online nástroje *ASN.1 JavaScript decoder*[c].

Při popisu konkrétních databázových tabulek jsou vynechány sloupce, jež obsahují pro tuto práci nezajímavé údaje, nebo jejichž účel není znám.

2.1 Časové formáty

Ve zkoumaných souborech můžeme nalézt dva typy časových záznamů.

- *Unix time* – čas vyjádřený počtem sekund od 1. 1. 1970, Mozilla Firefox ale převážně využívá rozšířený formát na mikrosekundy
- *FILETIME* – čas vyjádřený počtem mikrosekund od 1. 1. 1601

[a] `https://sqlitebrowser.org/`
[b] `https://www.mitec.cz/jsonv.html`
[c] `https://lapo.it/asn1js`

2.2 Google Chrome

Všechny čtyři soubory, které podléhají hlubší analýze, mají v prohlížečích založených na Chromiu podobnou strukturu. V případě záložek se liší Opera, která ukládá pro každou z nich množství metadat a celková struktura souboru je komplikovanější. V *Cookies* databázi ukládá dva sloupce navíc Microsoft Edge. Ty pravděpodobně souvisí s jeho přechodem na novější verzi s Chromium jádrem. Přihlašovací údaje ukládají oproti Google Chromu oba prohlížeče mírně odlišně. Zatímco Microsoft Edge obsahuje navíc databázovou tabulku s úniky dat, Opera rozdílně vytváří databázové tabulky s metadaty. Opera také ukládá větší množství dat souvisejících s historií návštěv jednotlivých webových stránek, jako jsou například celková délka návštěv nebo jejich počet.

V Google Chromu, který je v centru našeho zájmu, jsou soubory ve složce *C:\Users\<uživatel>\AppData\Local\Google\Chrome\User Data\Default*. Ta obsahuje i několik dalších databází, jako například *Top Sites* ukládající často navštěvované stránky, nebo *Web Data*, kde jsou uloženy např. údaje, které uživatel vyplnil do formulářů, nebo čísla kreditních karet.

2.2.1 DPAPI

Pro uložení šifrovacího klíče využívá Google Chrome *Data Protection Application Programming Interface (DPAPI)*. To bylo poprvé představeno při vydání OS Windows 2000 a dodnes slouží, jako hlavní systém ochrany pro OS Windows. *DPAPI* poskytuje pouze dvě funkce, a to *CryptUnprotectData* a *CryptProtectData*. Díky této jednoduchosti a léty prověřenému návrhu, který je stále vylepšován se *DPAPI* těší značné oblibě. Mimo Google Chromu ho využívá řada další programů, jako například Skype, Outlook nebo Microsoft Edge.

Ačkoliv *DPAPI* poskytuje pouze dvě jednoduché funkce, na pozadí se jedná o komplikovaný proces, pro který neexistuje žádná oficiální dokumentace. Stejně tak *BLOB*, který funkce *CryptProtectData* poskytuje jako výsledek šifrování, není oficiálně zdokumentován. Existují však skupiny, které pomocí reverzního inženýrství popsali principy fungování *DPAPI* včetně struktury datového *BLOBu*.[4]

Od OS Windows 7 využívá *DPAPI* pro šifrování *AES-256* v módu *CBC*. Jako hašovací algoritmus *SHA512* a funkci pro odvození klíče PBKDF2. Parametry těchto algoritmů, jako například počet iterací může uživatel s administrátorskými oprávněními nastavit v příslušných registrech.

Špatnou konfigurací *DPAPI*, však může být bezpečnost oslabena. Konkrétně jde o případ, kdy je šifrování provedeno s příznakem *CRYPTPROTECT_LOCAL_MACHINE*, což zapříčiní, že zašifrovaná hodnota není závislá na hesle uživatele. Další možností je *DPAPI* nakonfigurovat, aby pra-

coval v režimu kompatibility pro OS Windows 2000, kde je pomocí známe zranitelnosti snadné dešifrovat *BLOB* bez uživatelského hesla.

2.2.2 Šifrování

Google Chrome šifruje uložená uživatelská hesla a hodnoty souborů cookies. Využívá k tomu šifru *AES-256* v *GCM* módu. Šifrovací klíč je uložen v *JSON* souboru *Local State*, ve kterém je umístěn v objektu *os_crypt* obsahující jedinou hodnotu *encrypted_key*, což je náš hledaný klíč. Tento soubor se nachází v adresáři *C:\Users\<uživatel>\AppData\Local\Google\Chrome\User Data*. Na začátku zašifrovaného klíče se nachází řetězec *„DPAPI“* a celá hodnota je následně zakódována pomocí Base64. Pro dešifrování klíče je nutné použít funkci *CryptUnprotectData* z Windows knihovny *Crypt32*. Vzhledem k použití této funkce je možné klíč dešifrovat pouze ze stejného počítače a uživatelského účtu, ze kterého byl zašifrován. [4], [5]

Formát zašifrované hodnoty hesla nebo cookie je zobrazen na obrázku 2.1. První tři bajty obsahují řetězec *„v10“*, následovaný dvanácti bajty inicializačního vektoru, dále se zde nachází samotná zašifrovaná hodnota proměnné délky a nakonec je zde šestnáct bajtů ověřovacího tagu.

```
struct WebPassword
{
 BYTE signature[3] = "v10";
 BYTE iv[12];
 BYTE encPassword[...]
 BYTE tag[16]
}
```

Obrázek 2.1: Struktura zašifrovaného údaje v Google Chromu [6]

2.2.3 Přihlašovací údaje

Přihlašovací údaje jsou uloženy v databázi *Login Data*. Mimo nejvýznamnější databázové tabulky *logins* (tabulka 2.1) zde můžeme najít také databázovou tabulku *stats*. Pokud se uživatel rozhodne dialog nabízející uložení přihlašovacích údajů uzavřít, je doména, přihlašovací jméno, celkový počet těchto uzavření a čas poslední návštěvy uložen právě do této databázové tabulky. [7]

Login Data.logins	
Název sloupce	Význam
origin_url	URL adresa, kde se nachází formulář
action_url	URL adresa, kde je umístěn přihlašovací skript
username_element	název pole kam bylo jméno zadáno
password_element	název pole kam bylo heslo zadáno
username_value	přihlašovací jméno
password_value	zašifrované přihlašovací heslo
date_created	čas vytvoření záznamu (FILETIME)
blacklisted_by_user	uživatel zakázal zapamatování hesla
times_used	počet použití záznamu
date_last_used	čas posledního použití záznamu (FILETIME)

Tabulka 2.1: Tabulka vybraných sloupců databázové tabulky *logins* [8]

2.2.4 Historie

Historii navštívených stránek nalezneme v databázovém souboru *History*. Databáze, kromě několika tabulek s vnitřními daty prohlížeče, obsahuje tabulku *Downloads*, ve které můžeme najít záznamy o stahovaných souborech, nebo také tabulku *keyword_search_terms* obsahující všechny, uživatelem ručně zadané výrazy do adresního řádku. Práce se však zaměřuje pouze na historii prohlížení. Tu Google Chrome ukládá pouze po dobu tří měsíců. Je možné přímo do databáze vložit i starší záznamy, nicméně prohlížeč historie na to není připraven a má problémy s jejich zobrazením. Abychom získali kompletní informace o historii, musíme „složit“ data ze dvou databázových tabulek, a to *urls* a *visits*. Jejich sloupce jsou rozepsány v tabulkách 2.3 a 2.4.

Přehled o importovaných záznamech můžeme získat z databázové tabulky *visit_source*. Ta obsahuje pouze dva sloupce, kde *id* představuje cizí klíč do tabulky *visits* a sloupec *source* obsahuje typ importu (tabulka 2.2). Zvláštností je typ importu s hodnotou *1*, který reprezentuje vytvoření záznamu uživatelem. Vzhledem k tomu, že by zde měly být uloženy pouze importované záznamy, můžeme předpokládat, že tato hodnota není prohlížečem využívána. [9]

Zvláštní pozornost je věnována sloupci *transition* (tabulka 2.5) z databázové tabulky *visits*. Ten v sobě ukládá číselnou hodnotu představující typ přechodu při konkrétní návštěvě stránky. Po převedení této hodnoty do hexadecimální podoby a aplikování bitové operace AND s maskou *0xFF* získáme typ přechodu. Při aplikaci „opačné“ masky *0xFFFFFF00* získáme druhou část informace tzv. *kvalifikátor*. Podrobný popis všech hodnot můžeme nalézt v komentářích zdrojového kódu. [11]

History.visit_source.source	
Hodnota	Význam
0	synchronizováno z jiného zařízení
1	navštíveno uživatelem
2	přidáno rozšířením
3	importováno z Mozilly Firefox
4	importováno z Internet Exploreru
5	importováno ze Safari

Tabulka 2.2: Tabulka typů importu ve sloupci *source* [10]

History.urls	
Název sloupce	Význam
id	identifikátor záznamu
url	URL adresa navštívené stránky
title	titulek stránky z HTML tagu <title>
visit_count	počet návštěv stránky
last_visit_time	čas posledního návštěvy (FILETIME)

Tabulka 2.3: Tabulka vybraných sloupců databázové tabulky *urls*

History.visits	
Název sloupce	Význam
id	identifikátor záznamu
url	cizí klíč do tabulky *urls*
visit_time	čas návštěvy (FILETIME)
from_visit	identifikátor zdrojové návštěvy ze sloupce *id*
transition	typ přechodu
visit_duration	délka návštěvy v mikrosekundách

Tabulka 2.4: Tabulka vybraných sloupců databázové tabulky *visits* [8]

2.2.5 Záložky

Soubor záložek ve formátu *JSON* obsahuje dva objekty nejvyšší úrovně, a to *checksum*, kde je uložen kontrolní součet, a *roots*, obsahující tři další objekty rozdělující záložky podle typu. Všechny tři objekty obsahují stejnou strukturu, jež je vidět na obrázku 2.2. Prvním atributem je pole *children*, ve kterém jsou uloženy objekty reprezentující konkrétní záložky. Všechny tyto objekty mají také stejnou strukturu (obrázek 2.3) a můžou obsahovat další pole *children*, což reflektuje adresářovou strukturu, kterou si uživatel v prohlížeči vytvořil. Čas je opět uložen ve formátu *FILETIME* a *guid* je standardní *UUID* ve verzi 4, tedy náhodně generovaný. V případě změny obsahu souboru se hodnota *checksum* sama přepočítá při dalším spuštění Google Chromu. Při práci s tímto

History.visits.transition			
typ		**kvalifikátor**	
Hod.	Význam	Hodnota	Význam
1	Typed	0x00800000	Blocked
2	Auto Bookmark	0x01000000	Forward back
3	Auto Subframe	0x02000000	From Address bar
4	Manual Subframe	0x04000000	Home Page
5	Omnibar Generated	0x08000000	From API
6	Top Level / Start page	0x10000000	Chain start
7	Form Submit	0x20000000	Chain end
8	Reload / Restore záznamu	0x40000000	Client redirect
9	Keyword záznamu	0x80000000	Server redirect
10	Keyword Generated		

Tabulka 2.5: Tabulka hodnot po použití bitových masek na sloupec *transition* [12]

souborem je třeba mít na paměti, že se vytvoří až ve chvíli, kdy uživatel přidá první záložku.

```
{
   "checksum": "",
   "roots": {
      "bookmark_bar": {
        "children": [...],
        "date_added": "13262742847379952",
        "date_modified": "13262827577738200",
        "guid": "00000000-0000-4000-a000-000000000002",
        "id": "1",
        "name": "Bookmarks bar",
        "type": "folder"
         }
      "other": {...}
      "synced": {...}
    }
}
```

Obrázek 2.2: Struktura souboru záložek v Google Chromu

```
{
    "date_added": "13011557675000000",
    "guid": "2ecaa455-1bb6-4bf7-908e-6eebb07ecead",
    "id": "19",
    "name": "Google",
    "type": "url",
    "url": "https://www.google.cz/"
}
```

Obrázek 2.3: Struktura záložky v Google Chromu

2.2.6 Cookies

V databázi *Cookies* nalezneme dvě databázové tabulky. Důležitá je však pouze databázová tabulka *cookies*, jejíž popis se nachází v tabulce 2.6. Zde se nachází všechny soubory cookies, které prohlížeče na straně uživatele uložily. Zajímavostí je, že Google Chrome, na rozdíl od Mozilly Firefox, v této databázi zobrazuje i *session cookies*.

Cookies.cookies	
Název sloupce	Význam
creation_utc	čas vytvoření záznamu (FILETIME)
host_key	doména, pro kterou je cookie vytvořena
name	název cookie
value	hodnota cookie (dnes již nevyužito)
path	konkrétní cesta v URL
expires_utc	čas expirace cookie (FILETIME)
is_secure	cookie je přenášena přes zabezpečené kanály
is_httponly	cookie může využívat pouze server, kde vznikla
last_access_utc	čas posledního použití cookie (FILETIME)
encrypted_value	zašifrovaná hodnota cookie
samesite	příznak omezení cookie na konkrétní web
source_port	port, po kterém byla cookie přenesena

Tabulka 2.6: Tabulka vybraných sloupců databázové tabulky *cookies*

2.3 Mozilla Firefox

Všechny čtyři soubory, které podléhají hlubší analýze, jsou uloženy ve složce C:\Users\<uživatel>\AppData\Roaming\Mozilla\Firefox\Profiles\<profil>. Pokud disponujeme starší instalací prohlížeče, můžeme zde nalézt i zastaralé databázové soubory neobsahující žádný data. Jako příklady uveďme databázi *signons.sqlite*, která byla dříve využívána jako databáze přihlašovacích údajů nebo *key3.db*, jež je předchůdcem dále popisované databáze *key4.db*. V Mozille Firefox je čas ve formátu *Unix time* uvažován vždy v mikrosekundách, pokud není řečeno jinak.

2.3.1 Šifrování

Mozilla Firefox využívá pro práci s přihlašovacími údaji funkce z knihovny *NSS*. Pro uložení vnitřních dat nutných k šifrování je stěžejní databáze *key4.db*. Ta obsahuje dvě databázové tabulky – *metaData* a *nssPrivate*.

V tabulce *metaData* je pro nás významný řádek obsahující ve sloupci *id* řetězec *„password“*. V jeho dvou zbývajících sloupcích jsou hodnoty uloženy jako *BLOB*. Sloupec *item1* obsahuje hodnotu, kterou nazýváme *globalSalt*. Ve sloupci *item2* pak nalezneme *ASN.1* strukturu (obrázek 2.4) obsahující mimo jiné zašifrovaný řetězec *„password-check\x02\x02“*. Ten slouží jako kontrola nastavení tzv. hlavního hesla, kterým si uživatel může zabezpečit své přihlašovací údaje. Pokud není toto hlavní heslo uživatelem nastaveno, jeho hodnotu představuje prázdný řetězec.

Pro provedení této kontroly je nejprve nutné za hodnotu *globalSalt* připojit hlavní heslo a výsledný řetězec zahašovat pomocí funkce *SHA1*. Výsledek této operace poslouží jako heslo pro funkci *PBKDF2_HMAC_SHA256*. Další potřebné argumenty pro tuto funkci jsou *entrySalt* představující kryptografickou sůl, *iterationCount* určující počet iterací (standardně 1) a *keyLength* popisující délku klíče (standardně 32 bajtů). Umístění těchto hodnot můžeme vidět ve struktuře na obrázku 2.4. Konečně pro dešifrování výsledného řetězce použijeme *AES-256* v *CBC* módu, kde klíč představuje vypočtená hodnota z funkce *PBKDF2_HMAC_SHA256* a inicializační vektor hodnota *iv*, kterou můžeme opět najít ve struktuře na obrázku 2.4. [13] Před hodnotu inicializačního vektoru je navíc nutné, z interních důvodů, připojit hexadecimální hodnotu *\x04\x0E*. [14] Výsledným řetězcem musí být hodnota *„password-check\x02\x02“*. V opačném případě bylo nesprávně zadáno hlavní heslo, kterým uživatel přihlašovací údaje uzamkl. Příslušná funkce z knihovny *NSS* pro tuto operaci je *PK11_CheckUserPassword*.

Databázová tabulka *nssPrivate* obsahuje jediný řádek s mnoha sloupci. Významnou hodnotu uchovává sloupec *a11*, jehož struktura je, s výjimkou délky zašifrovaného řetězce, totožná se strukturou popsanou na obrázku 2.4. Ze sloupce *a11* je ekvivalentní postupem popsaným v předchozím odstavci možné získat hodnotu, kde prvních 24 bajtů představuje šifrovací klíč pro

jednotlivé přihlašovací údaje. Přihlašovací jména i hesla, která jsou v souboru *logins.json* zakódovaná do Base64, mají strukturu popsanou na obrázku 2.5. Před pokusem o dešifrování je vhodné zkontrolovat 16bajtovou hexadecimální hodnotu *F8000000000000000000000000000001*, která se nachází v datové struktuře „nejvýše". Tato hodnota je, jako *BLOB*, také uložena v *nssPrivate* ve sloupci *a102*. Pro získání údaje musíme dešifrovat *encryptedValue* z obrázku 2.5, pro což je použita šifra *3DES*, kde inicializační vektor představuje *iv* (taktéž z obrázku 2.5) a klíč získáme výše popsaným postupem. Závěrem nesmíme zapomenout odstranit zarovnání definované ve standardu *PKCS#7*. [15] Příslušná funkce, z knihovny *NSS*, je pro šifrování *PK11SDR_Encrypt* a pro dešifrování *PK11SDR_Decrypt*.

```
SEQUENCE (2 elem)
 SEQUENCE (2 elem)
  OBJECT IDENTIFIER 1.2.840.113549.1.5.13 pkcs5PBES2
  SEQUENCE (2 elem)
   SEQUENCE (2 elem)
    OBJECT IDENTIFIER 1.2.840.113549.1.5.12 pkcs5PBKDF2
    SEQUENCE (4 elem)
      OCTET STRING (32 byte) //entrySalt
      INTEGER 1 //iterationCount
      INTEGER 32 //keyLength
      SEQUENCE (1 elem)
       OBJECT IDENTIFIER 1.2.840.113549.2.9 hmacWithSHA256
   SEQUENCE (2 elem)
    OBJECT IDENTIFIER 2.16.840.1.101.3.4.1.42 aes256-CBC
    OCTET STRING (14 byte) //iv
 OCTET STRING (16 byte) //encrypted password-check\x02\x02
```

Obrázek 2.4: ASN.1 struktura obsahující kontrolní řetězec v Mozille Firefox

```
SEQUENCE (3 elem)
 OCTET STRING (16 byte) F8000000000000000000000000000001
 SEQUENCE (2 elem)
  OBJECT IDENTIFIER 1.2.840.113549.3.7 des-EDE3-CBC
  OCTET STRING (8 byte) //iv
 OCTET STRING (? byte) //encryptedValue
```

Obrázek 2.5: ASN.1 struktura přihlašovacího údaje v Mozille Firefox

2.3.2 Přihlašovací údaje

Soubor *logins.json* obsahuje přihlašovací údaje, které mají zašifrované nejen heslo, ale na rozdíl od Google Chromu i přihlašovací jméno. Stejně jako v případě souboru *Bookmarks* v Google Chromu je tento soubor vytvořen až ve chvíli, kdy uživatel uloží první přihlašovací údaje. Na obrázku 2.6 můžeme vidět celou strukturu tohoto souboru, na obrázku 2.7 pak ukázku jednoho údaje. U záznamů, jejichž význam byl zjištěn se nacházejí komentáře. Zašifrované údaje jsou ve formátu *ASN.1* a následně zakódované do Base64. Čas je v tomto souboru v milisekundách.

```
{
  "nextId": 102, //identifikátor pro nový záznam
  "logins": [], //pole přihlašovacích údajů
  "version": 3, //verze tohoto souboru
  //záznamy se slabými hesly
  "potentiallyVulnerablePasswords": [],
  //záznamy vyskytující se v úniku dat
  "dismissedBreachAlertsByLoginGUID": {}
}
```

Obrázek 2.6: Struktura souboru přihlašovacích údajů v Mozille Firefox

```
{
  "id": //identifikátor záznamu
  "hostname": //adresa webu
  "httpRealm": null,
  "formSubmitURL": //adresa, kde byly údaje zadány
  "usernameField": //název formulářového pole pro jméno
  "passwordField": //název formulářového pole pro heslo
  "encryptedUsername": //zašifrované jméno
  "encryptedPassword": //zašifrované heslo
  "guid": //UUID verze 4
  "encType": 1,
  "timeCreated": //čas vytvoření (Unix time)
  "timeLastUsed": //čas posledního použití hesla (Unix time)
  "timePasswordChanged": //čas změny hesla (Unix time)
  "timesUsed": //počet použití hesla
}
```

Obrázek 2.7: Struktura přihlašovacího údaje v Mozille Firefox

2.3.3 Historie

Na rozdíl od Google Chromu je v Mozille Firefox historie uložena pouze v jedné databázové tabulce *moz_places* (tabulka 2.7), která se nachází v databázi *places.sqlite*. V té však existuje cizí klíč (fk) do tabulky *moz_origins*, která ukládá webové adresy bez *path* části a vytváří tak agregovaný záznam o všech návštěvách konkrétního webu. Pro tyto adresy Mozilla Firefox vypočítává hodnotu *frecency* reflektující počet návštěv tohoto webu. Pojem *frecency* je kombinací slov frequency a recency, tedy frekvence a aktuálnost. Výsledná hodnota je tedy kombinací těchto dvou parametrů. Podrobný postup výpočtu je popsán v dokumentaci na stránkách *MDN Web Docs*. [16] Doba po kterou jsou záznamy historie ukládány není nastavena na pevný čas jako v Google Chromu, ale záleží na počtu záznamů v databázi. Tuto hodnotu je možné zjistit a případně změnit v pokročilé konfiguraci prohlížeče, kterou je možné otevřít zadáním „*about:config*“ do adresního řádku prohlížeči. Hodnotu je zde možné nalézt pod názvem *places.history.expiration.transient_current_max_pages* [17]

places.sqlite.moz_places	
Název sloupce	Význam
id	identifikátor záznamu
url	URL adresa navštívené stránky *urls*
title	titulek stránky z HTML tagu <title>
rev_host	opačně zapsána URL adresa bez *path* části
visit_count	počet návštěv stránky
favicon_id	nepoužívaný *fk* do zrušené tabulky *moz_favicons*
frecency	hodnota zvyšující se podle počtu návštěv
last_visit_date	čas poslední návštěvy (Unix time)
guid	12znakový guid identifikátor
url_hash	haš vypočítaná z *url*
description	popis z HTML tagu <meta content>
preview_image_url	náhled z HTML tagu <meta property=”og:image”>
origin_id	fk do tabulky *moz_origins*

Tabulka 2.7: Tabulka vybraných sloupců databázové tabulky *moz_places*

2.3.4 Záložky

Záložky jsou, stejně jako historie, uloženy v databázi *places.sqlite*, konkrétně v databázové tabulce *moz_bookmarks* (tabulka 2.8). Každý záznam má svého rodiče a cizí klíč do *moz_places*, který svazuje záložky s konkrétní webovou stránkou. Rodiče jsou v prohlížeči zobrazeni jako složky, které představují jednotlivé typy záložek. Těch je celkem pět, a to *Nabídka záložek*, *Lišta záložek*, *Štítky*, *Ostatní záložky* a *Záložky z mobilu*. Všechny pak mají jako svého rodiče nastavený kořenový adresář *root*. Uživatel má v prohlížeči možnost vytvářet

další složky a ty „zavěsit“ pod kteréhokoliv rodiče, čímž si může vytvořit libovolně komplikovanou adresářovou strukturu.

places.sqlite.moz_bookmarks	
Název sloupce	Význam
id	identifikátor záznamu
type	typ záložky, složka nebo url
fk	cizí klíč do tabulky *moz_places*
parent	číslo rodiče
postion	pozice v seznamu záložek
title	název záložky
dateAdded	čas přidání záložky (Unix time)
lastModified	čas poslední návštěvy (Unix time)
guid	12znakový guid identifikátor

Tabulka 2.8: Tabulka vybraných sloupců databázové tabulky *moz_bookmarks*

2.3.5 Cookies

V samostatné databázi *cookies.sqlite*, která obsahuje pouze jedinou databázovou tabulku *moz_cookies*, jsou uloženy nešifrované soubory cookies. Popis vybraných sloupců se nachází v tabulce 2.9

cookies.sqlite.moz_cookies	
Název sloupce	Význam
id	identifikátor záznamu
name	název cookie
value	nešifrovaná hodnota cookie
host	doména, pro kterou je cookie vytvořena
path	konkrétní cesta v URL
expiry	čas expirace cookie (Unix time v sekundách)
lastAccessed	čas posledního použití cookie (Unix time)
creationTime	čas vytvoření cookie (Unix time)
isSecure	cookie je přenášena přes zabezpečené kanály
isHttponly	cookie může využívat pouze server, kde vznikla
sameSite	příznak omezení cookie na konkrténi web

Tabulka 2.9: Tabulka vybraných sloupců databázové tabulky *moz_cookies*

2.4 Srovnání z hlediska bezpečnosti

Oba prohlížeče využívají pro uložení uživatelských dat kombinaci *sqlite* databází a JSON souborů.

Mozilla Firefox využívá pro zašifrování hlavního hesla *AES-256-CBC*. Pro uložení jeho klíče je využita funkce *PBKDF2_HMAC_SHA256*, která však používá pouze jednu iteraci což je z hlediska bezpečnosti nedostatečné. Hlavní heslo navíc nemusí být vůbec nastaveno a šifrování hesel v takovém případě stojí na šifře *3DES* s délkou klíče 24 bajtů. Šifra *3DES* s touto délkou klíče je v tuto chvíli ještě stále dostatečně bezpečná, nicméně podle *NIST* bude v roce 2023 označena jako zastaralá. [18]

Google Chrome shodně používá pro uložení hlavního šifrovacího klíče *AES-256-CBC* (skrz *DPAPI*), avšak pro uložení jeho klíče, je použita funkce *PBKDF2_HMAC_SHA512* s minimálně 5600 iteracemi. [4] K zašifrování samotného hesla je pak použit algoritmus *AES-256-GCM*, který pomocí tzv. tagu ověřuje integritu zašifrované hodnoty, což je z hlediska bezpečnosti velmi vhodné.

Z tohoto důvodu je zabezpečení uživatelských dat v prohlížeči *Google Chrome* lepší než v *Mozille Firefox*.

Kapitola 3

Návrh přenosového souboru

Jako formát přenosového souboru byl, z důvodu jeho velmi jednoduché struktury, zvolen *JSON*. Na rozdíl například od formátu *sqlite* databáze, je možné přenosový soubor snadno otevřít a porozumět jeho obsahu, bez potřeby instalování dalších programů. To je vhodné pro extrakci dat ve chvíli, kdy je prohlížeč nefunkční, ale příslušné soubory nebyly poškozeny. Struktura jednotlivých údajů byla navržena na základě průniku ukládaných informací ve všech zkoumaných prohlížečích.

Soubor se skládá z jednoho hlavního JSON objektu nazvaného *data*, který obsahuje pole pro každý typ dat. V těchto polích jsou následně záznamy ukládány v příslušných polích a formátech, které jsou definovány v následujících podkapitolách. Na obrázku 3.1 můžeme vidět jeho strukturu. Pro bezproblémové fungování programu je nutno dodržet několik obecných pravidel pro tvorbu přenosového souboru:

- Všechny časové záznamy jsou ve formátu *Unix time* s přesností na mikrosekundy.
- Každá položka musí nabývat nějakou hodnotu, nemůže se tedy rovnat *NULL*.
- Chybějí hodnoty řetězců jsou nastaveny na prázdný řetězec, pokud není řečeno jinak.
- Chybějících časové záznamy jsou nastaveny na aktuální čas, pokud není řečeno jinak.
- Při zpracování přenosového souboru program musí očekávat rozšířenou variantu o další pole dat nad rámec definované struktury na obrázku 3.1.
- Způsoby nakládání s chybějícími záznamy, které jsou specifické pro konkrétní typ dat, jsou popsaný v příslušných sekcích.

```
{
  "data": {
    "credentials": [],
    "cookies": [],
    "history": [],
    "bookmarks": []
  }
}
```

Obrázek 3.1: Základní struktura přenosového souboru

3.1 Přihlašovací údaje

Minimální data potřebná pro vytvoření záznamu přihlašovacího údaje jsou *url*, *username* a *password*, v opačném případě je záznam neúplný a neměl by se v přenosovém souboru vyskytovat. Způsob, kterým jsou přiřazeny údaje z obou prohlížečů do přenosového souboru, je popsán v tabulce 3.1. V případě chybějících názvů formulářových polí (*usernameField* a *passwordField*) není vhodné doplňovat jakékoliv standardní názvy, ale použít prázdný řetězec. Chybějící počet použití (*timeUsed*) je doplněn hodnotou *0*.

```
{
    "url": "https://www.test.cz/",
    "actionUrl": "https://login.test.cz/",
    "usernameField": "username",
    "passwordField": "password",
    "username": "testUsername",
    "password": "testPassword",
    "timeCreated": 1616195688290274,
    "timeLastUsed": 1616195676503711,
    "timesUsed": 0
}
```

Obrázek 3.2: Struktura přihlašovacího údaje v přenosovém souboru

Přiřazení přihlašovacích údajů		
Název údaje	Mozilla Firefox	Google Chrome
url	hostname	origin_url
actionUrl	formSubmitURL	action_url
usernameField	usernameField	username_element
passwordField	passwordField	password_element
username	encryptedUsername	username_value
password	encryptedPassword	password_value
timeCreated	timeCreated	date_created
timeLastUsed	timeLastUsed	date_last_used
timesUsed	timesUsed	times_used

Tabulka 3.1: Přiřazení hodnot přihlašovacích údajů z prohlížečů do přenosového souboru

3.2 Historie

Pro vytvoření záznamu historie postačí pouze webová adresa (*url*). Případný chybějící čas poslední návštěvy (*lastVisit*) by měl být nastaven na aktuální čas. Rozhodně není vhodné ponechat čas na nulové hodnotě, neboť Google Chrome pracuje pouze s historií mladší než tři měsíce a takovýto záznam by nezpracoval. Neznámý počet návštěv (*visitCount*) je vhodné nastavit na *0*.

```
{
    "url": "https://www.fit-wiki.cz/",
    "title": "obsah [FIT Wiki, FIT sobě]",
    "visitCount": 1,
    "lastVisit": 1619736690250475
}
```

Obrázek 3.3: Struktura záznamu historie v přenosovém souboru

Přiřazení historie		
Název údaje	Mozilla Firefox	Google Chrome
url	url	url
title	title	title
visitCount	visit_count	visit_count
lastVisit	last_visit_date	last_visit_time

Tabulka 3.2: Přiřazení hodnot historie z prohlížečů do přenosového souboru

3.3 Záložky

Pro přenesení záložky je třeba poskytnout její jméno (*name*) a webovou adresu (*url*). Pokud by jméno záložky nebylo přeneseno, může se zobrazení v různých prohlížečích chovat nestandardně. Mozilla Firefox je na takovouto situaci připravena a v případě prázdného pole *title* v databázové tabulce *moz_bookmarks* zobrazí namísto jména záložky její webovou adresu. Problém však nastává v Google Chromu, kde prázdný řetězec v libovolném klíči *title* zapsaném v JSON souboru *Bookmarks* způsobí zobrazení samotné ikony webu s prázdným názvem. To je z pohledu uživatele neakceptovatelné. Z tohoto důvodu je nutné název záložky vždy vyplnit a nezpracovat případné neúplné záznamy.

```
{
    "url": "https://www.fit-wiki.cz/",
    "name": "obsah [FIT Wiki, FIT sobě]",
    "dateAdded": 1618271115638109,
    "dateModified": 1618271115638109
}
```

Obrázek 3.4: Struktura záložky v přenosovém souboru

Přiřazení záložek		
Název údaje	Mozilla Firefox	Google Chrome
url	url (moz_places)	url
name	title (moz_bookmarks)	name
dateAdded	dateAdded (moz_bookmarks)	date_added
dateModified	lastModified (moz_bookmarks)	date_modified

Tabulka 3.3: Přiřazení hodnot záložek z prohlížečů do přenosového souboru

3.4 Cookies

K vytvoření záznamu souboru cookie v přenosovém souboru je nutné poskytnout minimálně název (*name*), hodnotu (*value*) a webovou adresu (*host*). Cesta (*path*), které konkrétní cookies náleží, bude v případě její absence nastavena na hodnotu „/“. Klíč *isSecure* indikuje, zda byla cookie přenesena přes protokol *HTTPS*. Pokud ano, je jeho hodnota nastavena na *1*, v opačném případě na *0*, což jsou jediné dvě hodnoty, kterých může nabývat. Stejných dvou hodnot může nabývat klíč *isHttpOnly*, určující, zda k obsahu cookie může přistupovat JavaScript (hodnota *0*), či nikoliv (hodnota *1*). V případě chybějících hodnot klíčů *isSecure* a *isHttpOnly* je nutné předpokládat z pohledu bezpeč-

nosti méně příznivou variantu a hodnoty nastavit na *0*. Klíč sameSite určuje kontext, ve kterém může být cookie dostupná. Připouštíme čtyři hodnoty, kterých může cookie nabývat. Hodnota *2* představuje příznak *Strict*, který omezuje odeslání cookie pouze na stejný web, který ji vytvořil. Podobně se chová cookie s příznakem *Lax* (hodnota *1*), která ale navíc dovoluje odeslání cookie i v případě, kdy uživatel na zdrojový web přechází například kliknutím na odkaz. Hodnota *0* pak znamená příznak *None*, který je vhodné nastavit ve chvíli, kdy je cookie cíleně poskytnuta jinému než zdrojovému webu. V případě příznaku *None* je však nutné nastavení parametru *isSecure* na hodnotu *1*. Pro klíč *sameSite* připouštíme však i hodnotu *-1*, kterou Google Chrome používá, pro nastavení nedefinované hodnoty. Mozilla Firefox však nedefinovanou hodnotu klíče *sameSite* nepodporuje, je tedy nutné případnou hodnotu *-1* převést na výchozí hodnotu *2*, tedy *Lax*. Tato výchozí hodnota a další informace jsou popsány např. na vývojářském webu Mozilly Firefox. [19]

```
{
    "name": "testName",
    "value": "27aae17feb2ce91b41cad473ea67",
    "host": "subpage.test.com",
    "path": "/",
    "expiry": 1644631653396221,
    "lastAccessed": 1620211211473279,
    "creationTime": 1613095653396221,
    "isSecure": 1,
    "isHttpOnly": 1,
    "sameSite": 0
 }
```

Obrázek 3.5: Struktura souboru cookie v přenosovém souboru

Přiřazení cookies		
Název údaje	Mozilla Firefox	Google Chrome
name	name	name
value	value	encrypted_value
host	host	host_key
path	path	path
expiry	expiry	expires_utc
lastAccessed	lastAccessed	last_access_utc
creationTime	creationTime	creation_utc
isSecure	isSecure	is_secure
isHttpOnly	isHttpOnly	is_httponly
sameSite	sameSite	samesite

Tabulka 3.4: Přiřazení hodnot cookies z prohlížečů do přenosového souboru

Kapitola 4

Implementace

Aplikace byla implementována v jazyce C/C++ a využívá knihovnu *OpenSSL* ve verzi 1.1.1i[d]. Vytvořena a následně testována byla na 64bitové verzi OS Windows 10. Pro práci se soubory ve formátu JSON byla využita knihovna *JSON for Modern C++*.[e] Pro kódování a dekódování ve formátu Base64 byla použita knihovna licencovaná společností Apple.[f] Aplikace také pro práci s databázemi využívá 64bitovou knihovnu *sqlite*.[g]

Program se skládá ze dvou základních souborů, a to *main.cpp* a *datatransfer.cpp* s hlavičkovým souborem *datatransfer.h*. Tento hlavičkový soubor je pak zahrnut do hlavičkových souborů pro implementaci převodu v jednotlivých prohlížečích. Konkrétně to jsou soubory *chrome.cpp* s hlavičkovým souborem *chrome.h* a *fireofx.cpp* s hlavičkovým souborem *firefox.h*.

4.1 Návrh základu aplikace

Konzolová aplikace **BrowserDataTransfer** má uživatelské rozhraní implementováno ve výchozím souboru *main.cpp*. Na základě uživatelského vstupu je následně naplněna strukturu ukazatelů na funkce (obrázek 4.1), implementací funkcí z požadovaného prohlížeče. Použití této struktury definované v souboru *datatransfer.h* zajišťuje snadné rozšíření o další prohlížeče. Je však nutné implementovat tuto přesnou sadu funkcí pro každý prohlížeč a u všech funkcí striktně dodržet předepsané vstupní parametry a návratové hodnoty.

V souboru *datatransfer.cpp* jsou implementovány funkce, které jsou určeny jako univerzální funkce použitelné pro všechny prohlížeče. Najdeme zde běžné funkce pro kontrolu existence souborů nebo kontrolu, zda soubor obsahuje validní JSON. Dále pak funkce zajišťující převod mezi časovými formáty a funkci

[d] `https://www.openssl.org/source/`
[e] `https://github.com/nlohmann/json`
[f] `https://opensource.apple.com/`
[g] `https://www.sqlite.org/download.html`

generující UUID ve verzi 4. Je zde také implementováno šifrování a dešifrování symetrickou šifrou AES v módu GCM s 256bitovým klíčem. Nakonec je zde možné najít funkce *toTransferFile* a *fromTransferFile* obstarávající načítání a ukládání přenosového souboru a volání funkcí pro jednotlivé typy dat.

```
typedef struct
{
    int (*credentialsToTransferFile)(json& file, string path);
    int (*credentialsFromTransferFile)(json& file);
    int (*cookiesToTransferFile)(json& file, string path);
    int (*cookiesFromTransferFile)(json& file);
    int (*historyToTransferFile)(json& file, string path);
    int (*historyFromTransferFile)(json& file);
    int (*bookmarksToTransferFile)(json& file, string path);
    int (*bookmarksFromTransferFile)(json& file);
} structPointerStructure;
```

Obrázek 4.1: Zjednodušená struktura ukazatelů na funkce pro převod dat

4.2 Implementace pro Google Chrome

Vyjma výše zmíněné sady funkcí pro přenos dat je v *chrome.cpp* implementována funkce *chromeGetMasterKey*, která vyhledá a pomocí *CryptUnprotectData* dešifruje hlavní šifrovací klíč. Ten je následně předáván do funkcí *chromeDecrypt* a *chromeEncrypt*, které jsou zodpovědné za zpracování a opětovné sestavení datového BLOBu, jehož struktura je zobrazena na obrázku 2.1. Funkce *chromeEncrypt* je navíc zodpovědná za vygenerování 12bajtového inicializačního vektoru pro funkci *Aes256GcmEncrypt*. Toto generování je zajištěno voláním *RAND_bytes* z knihovny OpenSSL. Funkce *RAND_bytes* zajišťuje generování kryptograficky silných pseuonáhodných dat. Z hlediska bezpečnosti byl tedy tento způsob vyhodnocen jako vhodný.

4.3 Implementace pro Mozillu Firefox

Ve zdrojových kódech Mozilly Firefox umístěných v souboru *firefox.cpp* jsou pro šifrování a dešifrování uživatelských údajů využity dynamicky načtené funkce knihovny *NSS* z instalačního adresáře Mozilly Firefox. Stejným způsobem je řešena funkce *firefoxCheckMasterPassword* starající se o kontrolu nastavení hlavního hesla. Toto řešení je funkční, nicméně vhodnější by byl program fungující samostatně bez závislosti na knihovně *NSS*. To bylo původním záměrem, bohužel nebyla nalezena vhodná knihovna pro jazyk C/C++, která

by dokázala jednoduše pracovat s datovými *BLOBy* ve formátu *ASN.1*. Alternativou by byla vlastní implementace pro zpracování dat ve formátu *ASN.1*, to je ovšem velmi časově náročně. Nejen z tohoto pohledu by se jako vhodnější implementační jazyk jevil *Python*, pro který existuje velmi jednoduchá a funkční knihovna *pyasn1*.

4.4 Uživatelské rozhraní

Program nevyužívá systém přepínačů, jak je u mnoha podobných programů zvykem. Namísto toho uživatele provádí nastavením potřebných parametrů krok za krokem. Po prvotním upozornění na nutnost uzavření prohlížečů je uživatel vyzván, aby číselnou volbou zvolil operaci, kterou má program vykonat. Na výběr je přenos dat, který data převede mezi prohlížeči přímo a přenosový soubor smaže, dále pak vytvoření přenosového souboru a konečně načtení přenosového souboru. Pokud uživatel zvolí možnost přímého přenosu, je vyzván, aby zadal zdrojový a cílový prohlížeč. V opačném případě se program zeptá pouze na zdrojový prohlížeč pro vytvoření přenosového souboru nebo na cílový prohlížeč v případě jeho načtení. Pokud nebyl zvolen přímý převod, uživatel je vyzván, aby zadal cestu k přenosovému souboru, nebo využil výchozí cesty do aktuálního adresáře k souboru *transferFile.json*. Dále je nutné zvolit, která data mají být převedena. Na výběr jsou přihlašovací údaje, záložky, soubory cookies a historie, případně všechna data dohromady. Uživatel si však může vybrat i kombinaci těchto možností zadáním několika čísel najednou v libovolném pořadí. Pokud probíhá přenos směrem z prohlížeče, program se ještě zeptá, zda si uživatel přeje zdrojové soubory prohlížeče vyhledat automaticky, nebo k nim zadat cesty ručně. Jestliže program při pokusu o nakládání s přihlašovacími údaji v Mozille Firefox zjistí, že jsou uzamčeny hlavním heslem, vyzve uživatel k jeho zadání.

4.5 Testování

Při testování přenosu přihlašovacích údajů byl zjištěn výskyt duplicit, ačkoliv program měl být pro tyto případy ošetřen. Duplicity nastávaly pouze v případě, kdy byly do prohlížeče přenášeny přihlašovací údaje, které zde již byly uloženy. Opakování přenosu však další duplicity již nevytvářelo. Porovnáním duplicitních údajů bylo zjištěno, že u některých adres webů se nachází lomítko na jejich konci a u jiných ne. Google Chrome totiž ukládá, na rozdíl od Mozilly Firefox, adresu webu i s lomítkem na konci. Na základě tohoto zjištění byl program upraven tak, aby se v přenosovém souboru, v záznamech přihlašovacích údajů nacházely webové adresy s lomítkem na konci. V případě přenosu směrem z Mozilly Fireofx je tak lomítko na konec adresy přidáno a v opačném směru zase odebráno.

Při přenosu do databázové tabulky *moz_places*, která ukládá historii v Mozille Firefox, byl taktéž zjištěn výskyt duplicit. Problém byl způsoben chybějící kontrolou na straně programu. *SQL* příkaz vkládající záznamy do databáze sice obsahoval příznak *IGNORE*, ovšem tabulka *moz_places* nemá *UNIQUE* omezení, a tak příznak *IGNORE* neměl žádný efekt.

Po těchto zjištěních a opravě dalších drobných chyb je přenos plně funkční pro oba prohlížeče. Značnou nevýhodou, kterou je třeba zmínit, je pomalý přenos při velkých objemech dat. Rychlost je však daní za vytváření a následné zpracování přenosového souboru, díky kterému je možné převádět data i mezi zařízeními.

Závěr

Cílem práce bylo analyzovat stávající řešení převodu uživatelských dat mezi prohlížeči, zaměřit se na nevhodně fungující nebo chybějící funkcionalitu a porovnat způsob, kterým prohlížeče data ukládají. Možnosti převodu byly popsány u čtyř nejpoužívanějších webových prohlížečů pro OS Windows 10. Z těchto čtyř byly následně pro analýzu souborů a demonstraci v praktické části vybrány dva prohlížeče – Google Chrome a Mozilla Firefox. Výběr proběhl podle popularity a jádra, na kterém jsou prohlížeče postaveny. Na základě analýzy uložených dat byla navržena univerzální struktura v souboru formátu *JSON*, která je využívána v implementační části. Vytvořený program zvládne vytvoření i načtení přenosového souboru, díky čemuž je možné přenášet data volně mezi zařízeními. Dokáže také provést přenos v rámci jednoho zařízení, kde soubory automaticky vyhledá. Je možné zvolit cestu, kam bude přenosový soubor uložen, a stejně tak cesty souborů, ze kterých budou data přenášena. To může být vhodné v případě, kdy se soubory nenachází na standardních místech. Alternativně může být program použit pro záchranu dat z nefunkčního prohlížeče. Tyto změny byly taktéž zaznamenány do dokumentace přenosového souboru.

Všechny vytyčené body práce byly splněny. Výsledkem je analytická část popisující data a použité šifrování. Výstupem praktické části je funkční program v jazyce C/C++ se snadným ovládáním. Návrh je zároveň zaměřen na snadnou rozšiřitelnost, tedy je možné v budoucnu doplnit implementace pro další webové prohlížeče.

Literatura

[1] DILGER, Daniel Eran. Apple apparently kills Windows PC support in Safari 6.0. [online]. 2012 [cit. 2021-04-19]. Dostupné z: https://appleinsider.com/articles/12/07/25/apple_kills_windows_pc_support_in_safari_60

[2] MOZILLA CORPORATION. Import login data from a file [online]. [cit. 2021-04-19]. Dostupné z: https://support.mozilla.org/en-US/kb/import-login-data-file

[3] STATCOUNTER. Browser Market Share Worldwide Mar 2020 - Mar 2021. [online]. [cit. 2021-04-20]. Dostupné z: https://gs.statcounter.com/browser-market-share

[4] PASSCAPE SOFTWARE. DPAPI Secrets. Security analysis and data recovery in DPAPI [online]. [cit. 2021-4-25]. Dostupné z: https://www.passcape.com/index.php?section=docsys&cmd=details&id=28

[5] MICROSOFT CORPORATION. CryptProtectData function [online]. 2018 [cit. 2021-4-25]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/api/dpapi/nf-dpapi-cryptprotectdata

[6] XENARMOR GLOBAL SECURITY SOLUTIONS PVT LTD. How to Recover Saved Passwords in Google Chrome [online]. 2019 [cit. 2021-4-21]. Dostupné z: https://xenarmor.com/how-to-recover-saved-passwords-google-chrome/

[7] BILLINGSLEY, Alex a James BILLINGSLEY. FOXTON FORENSICS. Analysing Chrome login data [online]. 2019 [cit. 2021-5-5]. Dostupné z: https://www.foxtonforensics.com/blog/post/analysing-chrome-login-data

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

[9] BILLINGSLEY, Alex a James BILLINGSLEY. FOXTON FORENSICS. Analysing synchronised browser history [online]. 2019 [cit. 2021-5-5]. Dostupné z: https://www.foxtonforensics.com/blog/post/analysing-synchronised-browser-history

[10] GOOGLE. History_types.h [online]. 2014 [cit. 2021-5-5]. Dostupné z: https://chromium.googlesource.com/chromium/src/+/master/components/history/core/browser/history_types.h

[11] GOOGLE. Page_transition_types.h [online]. 2012 [cit. 2021-5-5]. Dostupné z: https://source.chromium.org/chromium/chromium/src/+/master:ui/base/page_transition_types.h

[12] BENSON, Ryan. Chrome Transition Values [online]. 2014 [cit. 2021-5-5]. Dostupnéz:https://dfir.blog/chrome-transition-values/

[13] CLÉVY, Laurent. firepwd [online]. 2021 [cit. 2021-5-4]. Dostupné z: https://github.com/lclevy/firepwd/blob/master/firepwd.py

[14] RELYEA, Robert. Bug 1585189 - Changed the algorithm used to encrypt NSS database entries, from 3DES to AES256. [online]. 2019 [cit. 2021-5-4]. Dostupné z: https://hg.mozilla.org/projects/nss/rev/fc636973ad06392d11597620b602779b4af312f6#l6.49

[15] CLÉVY, Laurent. Mozilla password-base encryption [online]. 2015 [cit. 2021-5-4]. Dostupné z: https://github.com/lclevy/firepwd/blob/master/mozilla_pbe.pdf

[16] MOZILLA CORPORATION. Frecency algorithm [online]. 2019 [cit. 2021-5-8]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/Places/Frecency_algorithm

[17] MOZILLA CORPORATION. Places Expiration [online]. [cit. 2021-5-11]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/Places/Places_Expiration

[18] NIST. NIST Special Publication 800-131A [online]. [cit. 2021-5-13]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf

[19] MOZILLA CORPORATION. Using HTTP cookies [online]. [cit. 2021-5-12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

[20] MICROSOFT CORPORATION. The latest supported Visual C++ downloads [online]. [cit. 2021-5-12]. Dostupné z: https://support.microsoft.com/en-us/topic/the-latest-supported-visual-c-downloads-2647da03-1eea-4433-9aff-95f26a218cc0

Příloha A

Seznam použitých zkratek

3DES Triple Data Encryption Standard

AES Advanced Encryption Standard

API Application Programming Interface

ASN.1 Abstract Syntax Notation One

BLOB Binary Large OBject

CBC Cipher Block Chaining

CSV Comma-separated values

DPAPI Data Protection Application Programming Interface

fk Foreign key

GCM Galois/Counter Mode

GUID Globally Unique Identifier

HTTPS Hypertext Transfer Protocol Secure

HTML Hypertext Markup Language

JSON JavaScript Object Notation

NIST National Institute of Standards and Technology

NSS Network Security Services

OS Operační systém

PBKDF2 Password-Based Key Derivation Function 2

PKCS Public Key Cryptographic Standards

SHA Secure Hash Algorithm

SQL Structured Query Language

UUID Universally Unique Identifier

Příloha B

Uživatelská příručka

Aplikace **BrowserDataTransfer** je určena pro OS Windows 10. Pro spuštění je vyžadován balík *Microsoft Visual C++ Redistributable pro Visual Studio 2015, 2017 a 2019* dostupný ze stránek firmy Microsoft.[h] Dále je třeba, aby se ve stejném adresáři nacházela 64bitová knihovna *sqlite3.dll*, která je k dispozici na přiloženém médiu v adresáři *exe*, případně je možné ji získat ze stránek projektu *SQLite*.[i]

Pro správné fungování převodu pro Mozillu Firefox je vyžadováno, aby na zařízení, kde převod probíhá, byla Mozilla Firefox nainstalována. Pro správné fungování dešifrování dat z Google Chromu je nutné, aby tato operace probíhala na zařízení a uživatelském účtě, kde byla data zašifrována. Přenesení dat z Google Chromu na jiné zařízení a následný pokus o jejich dešifrování vždy skončí neúspěchem.

Před spuštěním programu musí být dotčené prohlížeče ukončeny. Přenosový soubor obsahuje velké množství citlivých údajů, je třeba s ním nakládat velmi opatrně a po práci ho bezpečně smazat!

[h] `https://aka.ms/vs/16/release/vc_redist.x64.exe`
[i] `https://www.sqlite.org/download.html`

Příloha C

Ukázka běhu aplikace

```
Make sure browsers are closed!!!

1) Transfer Data
2) Create transfer file
3) Load data from transfer file
Select method: 2

1) Set file name with path
2) Use default name (transferFile.json)
Select method: 1
Enter the path for transfer file: myTransferFile.json

1) Mozilla Firefox
2) Google Chrome
Select a source browser: 1

1) Credentials
2) Bookmarks
3) Cookies
4) History
5) All
Select the data for transfer: 132

1) Set paths of browsers files
2) Use default paths
Select method: 2
```

```
Data transfer in progress...
Enter Firefox master password:
Password is correct.
Transfer done.
```

Obrázek C.0: Ukázka běhu aplikace pro vytvoření přenosového souboru obsahujícího přihlašovací údaje, záložky a soubory cookies z Mozilly Firefox

Příloha D

Obsah přiloženého CD

- `BP-Stanislav-Lepic-2021.pdf` ..... text práce ve formátu PDF
- `ctiMe.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou aplikací včetně knihoven
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX