FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Bezpečnostní analýza ukládání hesel v prohlížeči Mozilla Firefox
Student:	Vladimir Dmitriev
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 se stavem podpory ukládání hesel v aktuálních webových prohlížečích.
2. Nastudujte dostupné informace o způsobu ukládání hesel v prohlížečích Google Chrome, Mozilla Firefox, Microsoft Edge.
3. Porovnejte používané techniky mezi sebou a srovnejte je také s aktuálními best practices.
4. Zaměřte se na Mozilla Firefox. Ověřte shodu dokumentovaných vlastností se skutečnou realizací správce hesel v prohlížeči.

5. Navrhněte a implementujte nástroj pro export a import uložených hesel v profilu prohlížeče.
6. Otestujte tento nástroj na několika podporovaných platformách. Diskutujte své výsledky.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 27. ledna 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Bezpečnostní analýza ukládání hesel v prohlížeči Mozilla Firefox

Vladimir Dmitriev

Katedra informační bezpečnosti
Vedoucí práce: Ing. Josef Kokeš

5. května 2021

Poděkování

Rád bych poděkoval vedoucímu mé práce Ing. Josefu Kokešovi za vedení a rady při vypracování. Dále bych rád poděkoval rodině za veškerou podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů 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 předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 5. května 2021 . . . . . .

Odkaz na tuto práci

Dmitriev, Vladimir. *Bezpečnostní analýza ukládání hesel v prohlížeči Mozilla Firefox*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

Abstrakt

Tato práce je zaměřena na seznámení s postupy, jakými webové prohlížeče zabezpečují uložená prostřednictvím správce hesel data. Jedním z cílů je porovnání „best practicies“ pro danou problematiku s řešeními použitými v prohlížečích. Druhá polovina práce se víc zaměřuje na Mozilla Firefox a jeho implementaci zpracování hesel a jejich uložení. Dále je na základě získaných informací vytvořen nástroj pro import a export hesel pro profily Firefox.

Klíčová slova Network Security Services (NSS), Mozilla Firefox, heslo, prolomení hesla, útok hrubou silou, správce hesel

Abstract

This work is focused on getting acquainted with the methods by which web browsers secure data stored in the password managers. One of the goals is to compare „Best practices“ for this problem with solutions which are used. The second half of the work is more focused on Mozilla Firefox and its implementation of password processing and storage. Based on the obtained information a tool for import and export of passwords for Firefox profiles is created.

Keywords Network Security Services (NSS), Mozilla Firefox, password, password cracking, brute-force attack, password manager

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V roce 2004 NIST vydal návod, který obsahoval i doporučení k nastavení politik k heslům uživatelů [1]. Za účinné bylo považováno použití symbolů z různých množin (speciální, číslice, velká písmena), expirace hesla za 3-6 měsíců. Tato doporučení ale vedla ke kontraproduktivním výsledkům. Místo komplexních hesel vznikala např. spojení starých hesel s minimální množinou požadovaných symbolů na konci. V případě opravdu poctivě vytvořených hesel to často končilo použitím stejného hesla které pro celou řadu služeb nebo jeho vytištěním na papír vedle počítače [2].

V roce 2017 byly publikovány nové instrukce, zmírňují požadavky ohledně komplexnosti a pravidelné výměny. Zmiňují také, že přehnané požadavky nesou většinou spíš frustraci uživatelům než nějaký užitek [2]. Pro uživatele ale stejně zůstává obtížnou úlohou zapamatovat si přihlašovací údaje ke všem použitým servisům, i když budou redukovány požadavky na heslo a interval výměny.

Správce hesel může být pomocníkem pro uživatele v této situaci. Pamatovaní celého seznamu hesel se zmenší na jedno hlavní. Otázkou ale zůstává míra ochrany takových údajů.

V dané práci se zabývám pouze správci hesel, které jsou vestavěny do webových prohlížečů. Oddělené nástroje jako jsou LastPast, Dashlane a další probrány nebudou (stejně jako i synchronizační prvky, které dál zmíněné prohlížeče nabízejí).

V první kapitole se zaměřuji na technologie použité webovými prohlížeči k uchování uložených přihlašovacích údajů.
V druhé se dívám na „best practices“ od NIST a porovnávám to s realitou ve vybraných prohlížecích.
V třetí kapitole se soustředím na implementaci správce přihlašovacích údajů v Mozilla Firefox a porovnávám ji s dokumentovanými vlastnostmi.
V čtvrté implementuji nástroj, který dokáže importovat a exportovat heslo do/z Mozilla Firefox zpracováním samotného souboru s uloženými daty.

Kapitola 1

Ukládání hesel v aktuálních webových prohlížečích

V této kapitole se zaměřím na postupy, které byly použity v rámci realizace správce hesel nejvíce používanými webovými prohlížeci (Google Chrome, Safari, Microsoft Edge a další).

1.1 Google Chrome

Google Chrome je nejvíc používaný webový prohlížeč [3], založený na projektu Chromium (bezplatný a open-source projekt sponzorovaný Google). Na různých operačních systémech používá různé nástroje k zpracování šifrovacího klíče.

Na strojích s Windows je to Data Protection API (DPAPI), jehož popis je uveden v kapitole 1.6.1, nástroj, který umožňuje šifrovaní a dešifrování dat. Pro omezení přístupu k utajeným datům používá heslo k uživatelskému účtu Windows. Pomocí DPAPI Chrome šifruje náhodně generovaný 32 bajtový klíč, kóduje ho do Base64 formátu a umísťuje do souboru "Local State". Na uložené přihlašovací údaje aplikuje AES-256 GCM v kapitole 1.6.7 s použitím dříve vytvořeného klíče a IV (inicializační vektor) – 12 bajtů náhodně generovaných dat. Zpracované údaje pak uloženy do souboru „Login Data“. (Před verzi 80 byla pomocí DPAPI zpracována všechna hesla) [4].

V prostředí macOS je použit svazek klíčů – Keychain, jehož popis je uveden v kapitole 1.6.4. Do roku 2015 Google Chrome ukládal všechny přihlašovací údaje do svazku, ale od verze OS X 10.9 byl přidán iCload Keychain a přepracována politika přístupu k uloženým klíčům [5]. Kvůli těmto změnám prohlížeč teď ukládá pouze 1 šifrovací klíč ve svazku a zpracovaná data v souboru „Login Data“. Klíč pro šifrování/dešifrování generován samotným prohlížečem a

data jsou podobně jako na Windows šifrována AES-256 GCM.

Na Linuxu Chrome aplikuje GNOME Keyring (Seahorse) (s popisem v kapitole 1.6.5) nebo KWallet k uložení šifrovacího klíče a šifrovaná data ukládá do „Login Data“ v složce uživatelského profilu Google Chrome [5]. (Podobně jako v případě macOS, v GNOME Keyring byla uložena všechna hesla, ale z technických důvodů to bylo změněno). V případě absence GNOME Keyring a Kwallet bude použit konstantní klíč pro šifrování [5].

1.2 Vivaldi a Opera

Jsou postaveny na bázi Chromium. Na stanicích s OS Windows, podobně jako Google Chrome, používají DPAPI [6] k ochraně šifrovacího klíče a šifrují údaje pomocí AES-256 GCM. (V minulých verzích byla pomocí DPAPI šifrována všechna hesla).
Na Linuxu jsou použity Gnome Keyring a pro macOS Keychain, podobným způsobem jako u Google Chrome.

1.3 Microsoft Edge

Microsoft Edge stejně jako i jeho předchůdce – Microsoft Edge Legacy, pro uložení přihlašovacích údajů uživatele používal „Credential Manager“ (speciální nástroj pro Windows, do kterého se ukládala hesla k webovým účtům a hesla pro další Windows aplikace). Ale s přechodem na bázi Chromium od verze 76 aplikoval DPAPI k šifrování všech uložených hesel. Teď je použitím DPAPI zpracován jenom šifrovací klíč pro AES-256 GCM a data ukládá do složky „Application Data“ [7].

Na macOS jsou údaje uloženy v souboru „Login Data“ ve složce profilu, v šifrovaném stavu a klíč je uložen v Keychainu uživatele.

1.4 Safari

K uložení hesel Safari využívá systémem nabízeny nástroj – Keychain. Přistup k nim uživatel pak dostává pomocí hesla k profilu operačnícho systému. Samotný Keychain aplikuje AES-GCM-256 pro šifrování a SQL soubor pro uložení dat.

1.5 Mozilla Firefox

V případě Mozilla Firefox není použit žádný z vestavěných nástrojů operačního systému pro šifrovaní nebo uložení klíčů. Namísto toho má sbírku knihovnen

Tabulka 1.1: DPAPI algoritmy a počet iterací pro různé verze Windows [12]

OS	Šifrovací algoritmy	Hašovací algoritmy	Počet iterací PBKDF2
Windows 2000	RC4	SHA1	1
Windows XP	3DES	SHA1	4000
Windows Vista	3DES	SHA1	24 000
Windows 7	AES-256-CBC	SHA512	5600
Windows 8	AES-256-CBC	SHA512	8 000
Windows 10	AES-256-CBC	SHA512	8 000

NSS, která implementuje šifrování, uložení privátních klíčů a zašifrovaných dat v speciálních souborech. Na rozdíl např. od Google Chrome má uživatel možnost nastavit hlavní heslo, které bude použito k šifrování jeho uložených přihlašovacích údajů [8, 9, 10].

1.6 Prohlížeči použité prostředky a algoritmy

V dalších podkapitolách jsou podrobněji rozebrány některé prostředky a algoritmy zmíněné v popisu jednotlivých prohlížečů.

1.6.1 Data Protection API

Od Microsoft Windows 2000 operační systém poskytuje pro uživatele a aplikace Data Protection Application-Programming Interface(DPAPI) bez potřeby stahování dodatečných knihoven. API obsahuje 2 funkce *CryptProtectData()* a *CryptUnprotectData()* s různými nastaveními, která modifikují jejich chování [11].

Veřejné rozhraní DPAPI je částí knihovny Crypto32.dll, která je v CryptoAPI, a součásti všech Windows systémů. Když aplikace volá funkce pro šifrování nebo dešifrování poskytované DPAPI, ta udělá RPC zavolání (Remote procedure call), který ale zůstane na daném počítači. Pak budou zavolány privátní funkce z CryptoAPI (*Crypto32.dll*), které použijí kontext z LSA (*Local Security Authority*) k vybrané operaci.

Pro zabránění v přístupu jedné aplikace používající DPAPI k datům jiné v dešifrovaném stavu API umožňuje předat doplňující klíč (*secret*), který pak bude použit při zpracování dat.

Sám DPAPI generuje hlavní heslo (*Master key*). K tomu na začátku provede odvození klíče z hesla (Password-Based Key Derivation, popsán v standardu PKCS#5 1.6.8) PBKDF2 s SHA512 s 8 000 iteracemi pro Windows 10 [12, 13] (pro další verze v tabulce 1.6.1). Z toho vzniklý klíč je použit k šifrování

hlavního hesla pomocí AES-256-CBC v případě Windows 10, který je pak uložen ve složce profilu uživatele.

Generované hlavní heslo ale není použito rovnou k šifrování dat. Namísto toho bude vytvořen symetrický klíč pro *CryptProtectData()* a *CryptUnprotectData()* pomocí hlavního hesla (Master key), náhodných dat a doplňujícího klíče (*pOptionalEntropy*), který může dodat aplikace [14]. Tento symetrický klíč systém neukládá, místo toho uloží náhodná data použité při generování symetrického klíče do BLOB struktury k zašifrovaným datům. Kdy je BLOB předán zpátky DPAPI, náhodná data jsou znovu využita k odvození klíče a následnému dešifrování. Taková možnost v Chromium-based prohlížečích není použita. Zkontrolovat to můžeme pomocí funkce *CryptUnprotectData*, které na vstup dáme zašifrované heslo z „Local State“ souboru a pro parametr *pOptionalEntropy* nic nepřidáme. Dešifrováním dat z „Login Data“ pomocí AES-256-GCM s získaným klíčem dostaneme uložená hesla [15].

Generované hlavní heslo má hard-coded expirační čas – 3 měsíce. Po vygenerování nového hlavního hesla bude staré uloženo do souboru v složce profilu uživatele, který je ochráněn uživatelským heslem k systému. Kromě toho je ke každému použitému hlavnímu heslu generován Globally Unique Identifier (*GUID*). GUID je také součástí BLOB, který při žádosti o dešifrování předá aplikace do DPAPI a podle toho bude použito odpovídající hlavní heslo [11].

DPAPI má propojení s procesem výměny hesla a v okamžiku jeho změny uživatelem dešifruje a zároveň šifruje novým heslem uložená stará hlavní hesla (použitá k šifrování). Systém také má soubor „Credential History“, do kterého ukládá stará hesla k uživatelskému účtu. Pokud je nutno, DPAPI zkusí otevřít „Credential History“ pomocí aktuálního hesla a aplikuje stará hesla k dešifrování hlavních hesel.

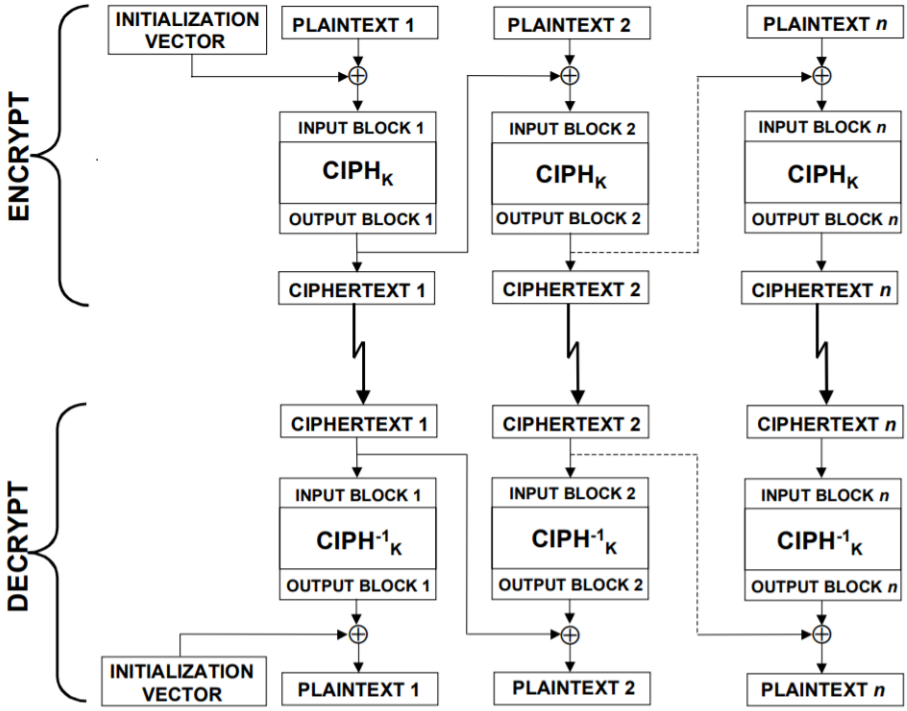
1.6.2 PBKDF2

Password-Based Key Derivation Function 2 (PBKDF2) je funkce odvození klíče z hesla. Pro její fungování musíme vybrat pseudo-náhodnou funkci, která použije jako vstup heslo nebo heslovou frázi (*passphrase*), sůl a počet iterací. Vygenerováný klíč pak může být použit jako kryptografický v dalších operacích. Zvýšením počtu iterací se zvětšuje čas, který by případný útočník musel věnovat prolomení hesla – key stretching. Použitím soli ve funkci se snižuje možnost aplikace předem zpracovaných hašů (*rainbow tables*) pro útoky [16].

Tento proces je použit např. v DPAPI při generování klíče šifrování dat (*Master key*), v NSS knihovně k generování klíče z hlavního hesla pro šifrováni uloženého 3DES klíče. Podle standardu od NIST pro rok 2010 by minimální počet iterací v PBKDF2 měl být 1 000 [16], pro rok 2017 už byla doporučená hodnota 10 000 [2]. Pro sůl podle RFC [17] by měla minimální délka být 64 bitů, podle NIST 128 bitů. Na rozdíl od PBKDF1, PBKDF2 může produkovat klíče s délkou větší než 160 bitů.

Slabinou PBKDF2 ale je, že může být implementována s malou velikostí RAM – není paměťově náročná a kvůli tomu útok hrubou silou např. pomocí GPU může být relativně levným. Alternativou pro PBKDF2 NIST uvádí algoritmus Baloon (paměťově a časově náročný), ale také doplňuje, že momentálně je PBKDF2 široce používaná a nevidí striktní požadavek na paměťovou náročnost jako praktický, a proto je to jenom doporučení. Za důležitější považuje použití pouze doporučených jednosměrných funkcí než nucené použití paměťově náročné funkce.[2, 16]

1.6.3 CBC šifrovací režim



Obrázek 1.1: CBC režim, šifrovací a dešifrovací schémata [18]

Cipher Block Chaining (CBC) – řetězení šifrových bloků – je režim blokové šifry. Během procesu je každý blok před samotným šifrováním ještě xorován s předchozím již šifrovaným blokem, jak je vidět na obrázku 1.6.3. První blok je xorován s inicializačním vektorem (IV). Tento režim je použit např. pro šifrování dat v DPAPI nebo v NSS knihovně, Gnome Keychain. Použitý IV nemusí být utajován, ale nesmí být předvídatelný. [18]

Daný režim je široce používán. Má ale i nevýhody plynoucí ze závislosti šifrování bloků na předchozích zpracovávaných částech. Šifrovaní na rozdíl

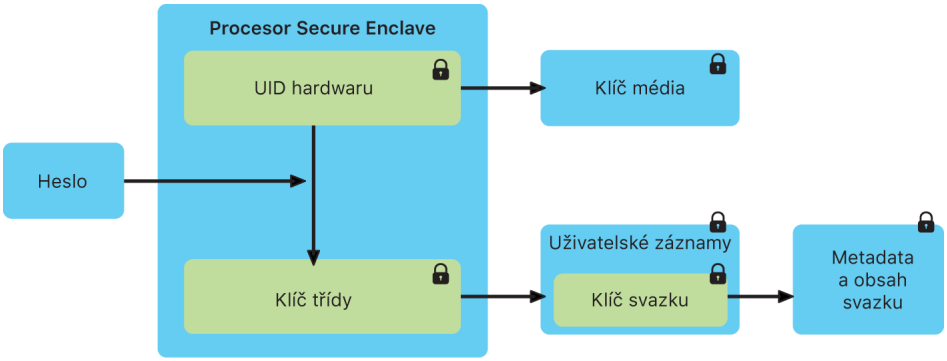
od dešifrování nelze paralelizovat a při poškození šifrovaného bloku přímo následující blok dešifrovat nejde.

1.6.4 Keychain

Pro bezpečné uchování hesel, klíčů nebo přihlašovacích tokenů v systémech iOS a iPadOS existuje speciální nástroj – svazek klíčů (*Keychain*). Položky uložené v svazku jsou šifrovány dvěma různými klíči pomocí AES-256-GCM: klíč pro tabulky metadat a řádkový klíč pro tajné hodnoty. Oddělený klíč pro metadata je použit za účelem zrychlení hledání. Pro jeho zabezpečení je použit modul Secure Enclave, ale uchovává se v mezipaměti obyčejného procesoru, pro rychlé zpracování dotazů na svazek klíčů. Pro použití druhého klíče ale směřuje požadovek vždy přes modul Secure Enclave [19].

Samotný svazek představuje SQLite databázi na disku přístup procesů a aplikací k svazku řídí démon securityd. Keychain také nabízí možnost sdílení přístupu k položkám ve svazku mezi aplikacemi pocházejícími od stejného vývojáře.

V případě Secure Enclave uživatel nikdy nedostává samotný klíč šifrování, místo toho aplikace nařídí Secure Enclave na začátku vytvořit nový klíč a uložit ho. Pak dostáváme pouze vystup operací jako např. šifrovaná data.



Obrázek 1.2: Procesor Secure Enclave, použití uživatelského hesla a UID klíče [19]

Secure Enclave v sobě má také unikátní UID 1.6.4, který je různý pro různá zařízení a nemá spojení s žádným dalším identifikátorem na stroji. Při spuštění zařízení bude vytvořen klíč pro šifrování dat (*ephemeral memory protection key*), které Secure Enclave potřebuje uložit na disk. Generování aplikuje odvozovací algoritmus PBKDF2 s AES jako pseudo-náhodnou funkcí a vstupem je uživatelské heslo a UID [20]. Od verze A11 je tento UID generován pomocí Secure Enclave TRNG a pak zapsán do Secure Enclave.

Součástmi Secure Enclave jsou kromě Boot ROM (který poskytuje root of trust pro Secure Enclave), také TRNG pro generování náhodných dat založený

na kruhovém oscilátoru (výstup je ještě dále zpracován pomocí CTR_DRBG – algoritmus založený na blokové šifře v CTR režimu), hardwarový blok AES poskytující symetrické šifrování a navržený pro odolání únikům dat, které lze použít v jednoduché odběrové analýze (SPA) a diferenciální odběrové analýze (DPA).

1.6.5 Gnome Keyring

Kolekce komponent pro Linuxové distribuce, které uchovávají citlivá data jako hesla, klíče, certifikáty a zprostředkují aplikacím. Keyring je integrován s PAM (*Pluggable authentication module*) a PKCS#11. Gnome Keyring nepodporuje uložení dat na externím médiu, ale nabízí podporu smart karet [21].

Defaultně bude keyring otevřen po přihlášení do OS, ale jde vytvořit oddělený keyring a pro něj nastavit hlavní heslo, které bude použito pro zabezpečení hesel a privátních klíčů.

Hlavní heslo (nebo heslo k OS) je iterativně hašováno pomocí SHA-256 a výsledek pak je použit jako klíč pro šifrovaní obsahu keyringu. Počet iterací použitých v tomto kroku je mezi 1000 a 2000. Uložená data jsou pak šifrována AES-128 [21].

1.6.6 Network Security Service

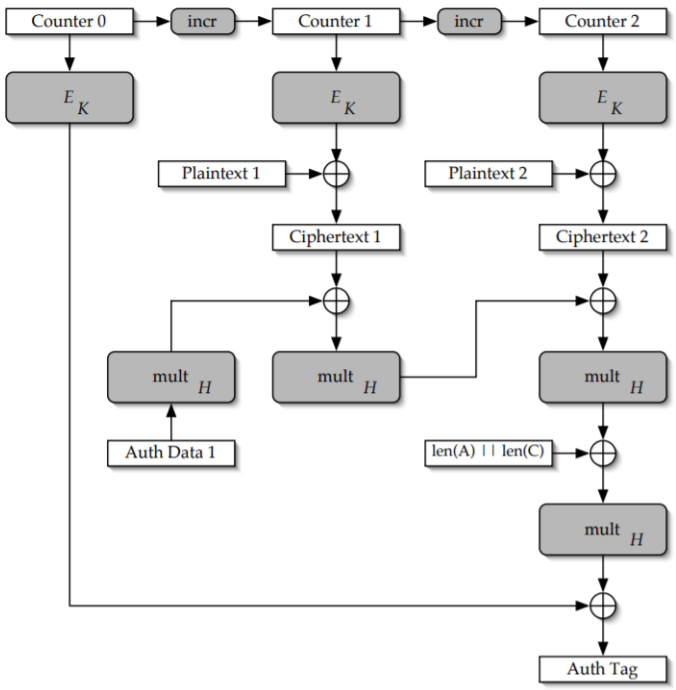
Network Security Service (NSS) je sbírka knihoven, API a služeb pro podporu cross-platformního vývoje zabezpečených klientských a serverových aplikací. Nabízí celou open-source implementaci knihoven použitých v Mozilla Firefox, Opera (SSL), serverových produktech od Red Hat a jiných.

NSS podporuje různé kryptografické standardy jako např. PKCS#5 (11, 12 atd.), TLS, S-MIME, SSL v2 a v3 a další [8].

1.6.7 GCM šifrovácí schéma

Jeden z režimů blokových šifer, který zároveň zajišťuje autentizaci dat (garantuje integritu dat) a kromě Keychain je také použit v SSH, TLS 1.2/1.3 a Google Chrome. Tento režim používá čítač a násobení v konečném (Galoisově) tělese. IV, který v procesu zpracuje, může mít různou délku, což zjednodušuje různým aplikacím jeho generování. Do procesu šifrování lze přidat doplňující autentizovaná data (AAD). Ta budou autentifikována, ale nebudou šifrována. Tento algoritmus může být použit pro data, které chceme ověřit, ale potřebujeme nechat nešifrovaná – např. pro síťovou komunikaci to mohou být adresy, porty, verze protokolů atd. GCM může pracovat i jako MAC na data v AAD, kdy délka šifrovaných dat bude 0 [22].

Pro popisovaný případ šifrování na obrázku 1.6.7 bude platit, že dodatečná data A mají délku m bloků a poslední z nich má v bitů. Otevřený text P čítá n bloků a poslední obsahuje u bitů. Použitá funkce *incr* pracuje s nejpravějšími



Obrázek 1.3: GCM režim, šifrovací schéma, E_K – bloková šifra, $mult_H$ – násobení v Galoisově tělese s podklíčem H, *incr* – funkce inkrementace hodnoty čítače, *AuthTag* – jeden z výsledků šifrování GCM potřebný k autentifikaci dat [22]

32 bity vstupu jako s nezáporným číslem s LSB (least significant bit) vpravo a zvětšuje hodnotu v modulu 2^{32}. Počáteční hodnota čítače Y_0 bude IV doplněný o 31 nulové bity a 1 jedničkový na konci, pokud délka IV je 96 bitů. Pro jinou délku IV bude poslán na vstup funkce GHASH.

E_K ve schématu 1.6.7 značí použití blokové šifry s klíčem K. Podobně jako pro CTR režim, jsou bloky otevřeného textu xorovány s šifrovanými hodnotami čítače. Kromě šifrového textu bude výsledkem GCM Auth Tag – hodnota $GHASH_{1+m+n}$ xorovaná s $E_K(Y_0)$. V závislosti na počtů bloků A, C podle funkce na obrázku 1.6.7 bude provedeno xorování s výsledkem minulé iterace a násobení v Galoisově tělese podklíčem H (počateční hodnotou X_0 pro GHASH je nula a podklíč H získáme šifrováním bloku nul v E_K). Pro Auth Tag může být vybrán nejaký počet bitů t z výsledku xorování $GHASH_{1+m+n}$ a zašifrovanou první hodnou čítače ($E_K(Y_0)$) [22].

1.6.8 Public Key Cryptographic Standards

Public Key Cryptographic Standards (PKCS) – skupina specifikací pro kryptografii s veřejným klíčem vypracovaná RSA Security a často použitá v praxi

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^* \| 0^{128-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_i) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_m^* \| 0^{128-u})) \cdot H & \text{for } i = m+n \\ (X_{m+n} \oplus (\text{len}(A) \| \text{len}(C))) \cdot H & \text{for } i = m+n+1. \end{cases}$$

$$H = E(K, 0^{128})$$

$$Y_0 = \begin{cases} IV \| 0^{31} 1 & \text{if len}(IV) = 96 \\ \text{GHASH}(H, \{\}, IV) & \text{otherwise.} \end{cases}$$

$$Y_i = \text{incr}(Y_{i-1}) \text{ for } i = 1, \ldots, n$$

$$C_i = P_i \oplus E(K, Y_i) \text{ for } i = 1, \ldots, n-1$$

$$C_n^* = P_n^* \oplus \text{MSB}_u(E(K, Y_n))$$

$$T = \text{MSB}_t(\text{GHASH}(H, A, C) \oplus E(K, Y_0))$$

Obrázek 1.4: Popis funkce GHASH pro GCM režim, X_i – mezivysledky funkce GHASH, H – podklíč, A_i – autentifikovaná, ale nešifrována data (m bloků), C_i bloky zašifrovaných dat (n bloků), fukce MSB_t vratí t nejvýznamějších bitů, P_i – bloky otevřeného textu [22]

a periodicky doplňovaná.

Standardy zpracovávají různé algoritmy a postupy. Např. PKCS#5 se zabývá šifrováním na základě hesla (PBE – Password-based Encryption), kdy je použita funkce pro odvození klíče z hesla nebo heslové fráze pro další šifrovaní. PKCS#11 standardizuje rozhraní pro uložení klíčů v specializovaném hardwaru (čipové karty, bezpečnostní moduly, tokény a další) nebo získání klíčů z nich. PKCS#12 specifikuje formát souborů, které lze použit pro ukládaní nebo předání privátního klíče s certifikáty [23].

PKCS#5 standard je použit v DPAPI a NSS při generování klíčů z hesla k profilu nebo hlavního hesla. NSS také implementuje rozhraní PKCS#11 a PKCS#12 pro export/import klíčů a certifikatů do/z speciálních souborů a NSS databází. Speciální soubory a databáze v případě NSS jsou SQLite soubory cert9.db, key4.db a pkcs11.txt (starou verzí je cert8.db, key3.db a secmod.db, které používaly Berkeley DB format) [24].

Kapitola 2

Porovnání použitých technik a „best practices“

NIST publikoval v roce 2017 návod [2], který obsahuje „best practices“ pro politiky vytvoření a uložení hesel. V této kapitole se zaměřím na to, jaká z těch doporučení splňují zvolené vestavěné do webových prohlížečů správce hesel.

2.1 Složení hesla

Minimální délka vybraného uživatelem hesla by měla být 8 symbolů, generované strojem 6 a maximální délka alespoň 64 symbolů. Doporučená je i podpora všech ASCII znaků a kontrola existence hesla v slovnících. Na rozdíl od návodu z roku 2004 nejsou požadavky na složitost (speciální znaky) [2].

Nucení k dodržení těchto doporučení je spíš starostí služeb, kde uživatel určité heslo použije, nežli správce hesel. Ale v jejich kompetenci by bylo při vyplnění přihlašovacích údajů ukazovat semafor vhodnosti tohoto hesla.

Mozilla Firefox má takový semafor pouze při nastavení hlavního hesla [25] a na macOS při vytvoření speciálního klíče pro Keychain [26] (které se bude lišit od hesla k systému) ukáže semafor „kvality hesla“, který ocení silu uživatelského hesla, ale další omezení mít nebude – „123“ i „password“ můžeme nastavit (jenom nedovolí nastavit prázdné heslo).

V případě např. Google Chromu, Microsoft Edge nebo Vivaldi, kde přistup k heslům zabezpečen přes nástroje operačního systému – DPAPI, Keychain (pokud nenastavíme dodatečné heslo) [21], je politika vytvoření hesel na tvůrci OS.

2.2 Zabránění offline útoku

Pro zvýšeni ceny útoku hrubou sílou na hesla uživatelů musí být použita funkce pro odvození klíče z hesla se solí (pro ztíženi použití předem zpracované tabulky s haši) a určitým počtem iterací (pro zvýšení času zpracování). NIST doporučuje použití paměťově a časově náročné funkce pro odvození klíče, jako např. Balloon. Větší důraz ale klade na možnost použití schválených jednosměrných funkci (HMAC, SHA-3, CMAC a další) v odvozovacím algoritmu a tím připouští použití PBKDF2 [2].

Délka soli pro odvozovací funkce by měla být minimálně 32 bity a má být generována náhodně, počet iterací pro PBKDF2 by měl být nastavován individuálně v závislosti na výkonu počítače; NIST uvádí nyní často použitou hodnotu 10 000 iterací [2].

V případě DPAPI ve Windows a NSS knihovny pro Firefox je použita právě PBKDF2. DPAPI aplikuje pro Windows 8/10 SHA-512 a 8 000 iterací. NSS jako defaultní hodnotu využívá 10 000 [27] uvedených v návodu a délku soli 32 bajtů pro SHA-256 v PBKDF2.

2.3 Rate limiting a nápovědy

Podle NIST by neměly být použity „hints“ (nápovědy k nastavenému uživatelem heslu, jako např. „Jméno prvního mazlíčka?“). Odpovědi k nim útočník by mohl získat prostřednictvím sociálních sítí nebo pomocí sociálního inženýrství. Pro prohlížeče založené na Chromium a používající Keychain nebo DPAPI není ani logické místo na jejich použití, když přístupem k přihlašovacím údajům je heslo k OS. V případě Firefox žádné nápovědy k hlavnímu heslu ani možnost jeho obnovení při zapomenutí není.

Rate limiting – ohraničení nepovedených autentizaci za nějaký interval, nemá moc smyslu, když máme lokálně přístup k souborům, kde jsou uložena data („Login Data“, „key4.db“, „logins.json“, „Application Data“).

2.4 Kontrola úniku hesel

I když správce hesel (prohlížečů uvedených dřív) neprovádí kontroly uložených přihlašovacích údajů, některé nabízí nástroje, které upozorní uživatele na kompromitovaná hesla (provedou kontrolu pomocí databáze „Have I Been Pwned“). Pro Google Chrome je nástroj Password Checkup [28], pro Mozilla Firefox – Firefox Monitor [29].

2.5 Generátor hesel

Schopnost generování náhodných, unikátních a komplexních hesel je požadovanou vlastnosti pro správce hesel podle NIST. Generátor hesel jako dostupnou funkci mají Vivaldi, Google Chrome a Mozilla Firefox (z seznamu prohlížečů uvedených v 1. kapitole). Splňují požadavek na minimální délku hesla 8 symbolů (generují hesla délkou 15). Firefox, Vivaldi a Chrome mají podobné generátory (s tím, že Firefox a Vivaldi zakládají generátory na verzi generátoru z Chromium [30, 31]). Generované heslo obsahuje symboly z malé/velké abecedy a číslic (symboly I, i, O, o, 0, 1 byly odstraněny kvůli vizuální podobnosti) a použije Fisher–Yates shuffle pro vytvoření náhodné permutace pro heslo.)

2.6 Použití správce hesel

Dokumentu SP 800-63B [2] neobsahuje přímé doporučení od NIST pro použití správce hesel, ale jenom doporučuje povolení funkce vložení hesla, aby uživatel mohl použit správce hesel, pokud o to bude mít zájem [32].

NIST souhlasí s tím, že správce hesel jako samostatné aplikace i vestavěné ve webových prohlížecích mohou nabídnout dobré zabezpečení pro hesla k online službám. Jejich větší bezpečnost pochází z existence generátoru silných hesel, zabezpečeného a šifrovaného úložiště. Samozřejmě velkým přínosem pro uživatele je i aplikace hlavního hesla místo pamatovaní velkého seznamu hesel k různým službám.

Upozorňuje ale i na to, že správce hesel se stává cílem útočníků, kvůli možnosti získat všechna hesla (i další citlivá data) uživatele, a proto je zabezpečení těchto úložišť důležité.

Pro hlavní heslo pro správce hesel radí použit delší heslovou frázi (passphrase) a postarat se o jeho bezpečnost. Pro vytvoření hesel k použitým slůžbám doporučuje vybírat správce hesel s integrovaným generátorem silných hesel a zároveň varuje před správci, které nabízí funkci obnovení hlavního hesla při zapomenutí. A použít vícefaktorovou autentizaci, pokud správce hesel nabízí takovou možnost.

2.7 Výsledek

Část vybraných prohlížečů pro ochranu dat používá prostředky OS. DPAPI splňuje požadavky s použitím odvozovací funkce, doporučené hašovací funkce. NSS také odvozuje klíč z hlavního hesla pomocí PBKDF2 a splňuje i nárok na hašovací funkci i na počty iterací/délku soli. Pro případ Google Chrome na macOS je generován náhodný šifrovací klíč šifrování, který je pak uložen do Keychain. V případě Linuxu lze využit Gnome Keyring, které pouze hašuje heslo od profilu OS, ale v situaci, kdy nebude mít možnost použít Gnome Keyring tak to zašifruje konstantním heslem.

Mozilla Firefox a Google Chrome také splňují doporučení NIST na upozornění uživatele při nalezení přihlašovacích údajů v „Have I Been Pwned“ databázi. Žádný z nich nebude nutit dodržet politiku při uložení hesel, pouze u Mozilly a pro případ Keychain, kdy změníme heslo k němu, nám bude ukázáno, jak bezpečné je naše heslo podle webového prohlížeče.

Nepodařilo se také najít expirační čas, po kterém by byly měněný klíče pro Google Chrome nebo Mozilla Firefox. Pro hlavní heslo u Firefox také nejsou nastaveny doby platnosti nebo funkce na nucenou výměnu šifrovacího klíče [33].

Kapitola 3

Správce hesel Mozilla Firefox

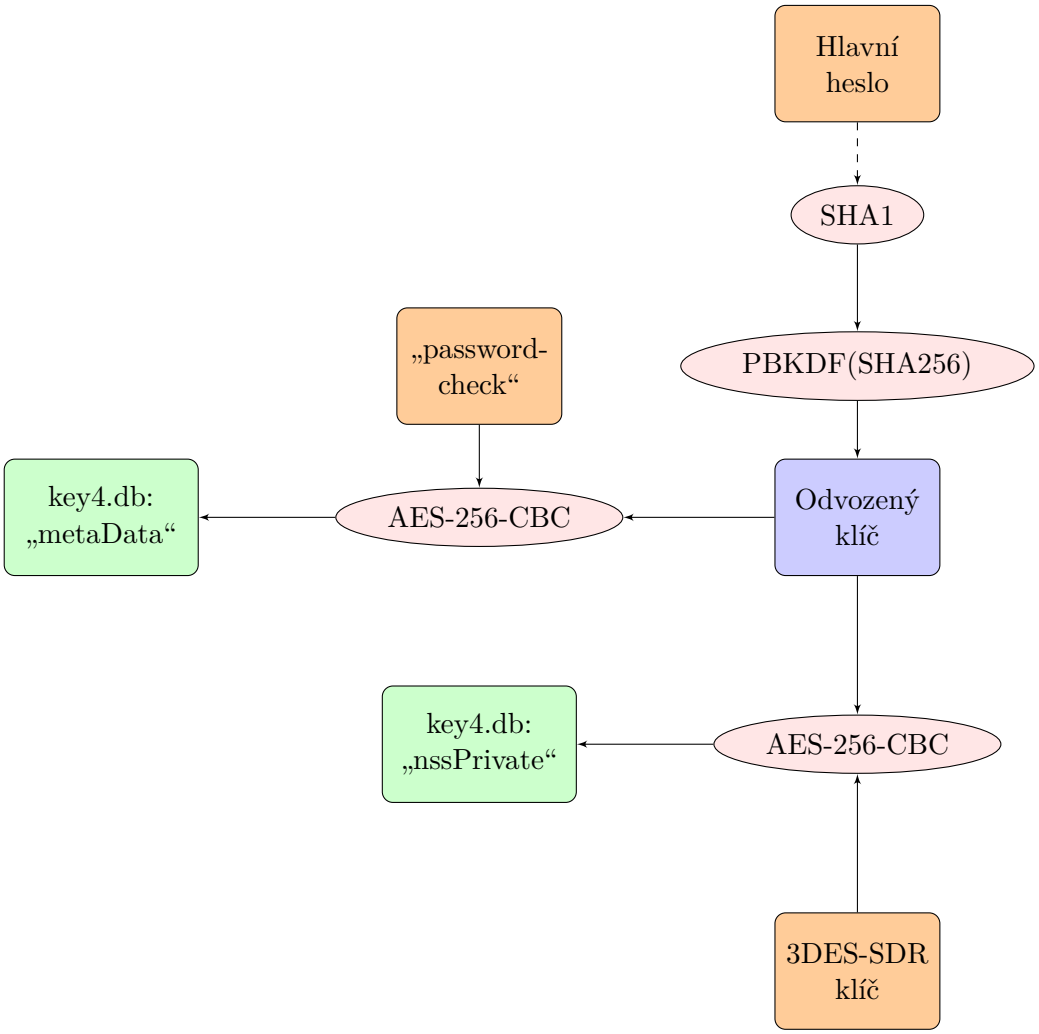
Tato kapitola je zaměřená na dokumentované vlastnosti správce hesel v Mozilla Firefox a jejich porovnání s implementací.

3.1 Zpracování hlavního hesla

Mozilla Firefox používá sbírku knihoven NSS k zpracovaní hesel, certifikátů, tokenů, šifrování dat. Pro uložení dat má speciální soubory. Soubor „key4.db“ je SQLite databáze pro klíče. Druhý soubor – „logins.json“ obsahuje informace jako jsou hostname, zašifrované heslo, zašifrované uživatelské jméno, GUID, čas vytvoření / posledního použití. Hesla a uživatelská jména jsou v ASN1 formátu a kódování Base64.

Uložené přihlašovací údaje jsou šifrovány pomocí 3DES-EDE-CBC (daný režim je krátce popsán v kapitole 1.6.3), prohlížeč sám generuje klíč [34]. Tento klíč před uložením je šifrován pomocí AES-256-CBC. Klíč pro AES vzniká použitím funkce odvození klíče z hlavního hesla (Master password). NSS aplikuje PBKDF2 s pseudonáhodnou funkcí SHA256 [35] a navíc SHA1 před tím [36, 37]. Dohromady musí odvozovací funkce dostat na vstup hlavní heslo, sůl a počet iterací. Standardně NSS (a tím pádem i Mozilla Firefox) používá 10 000 iterací [27]. Výchozí hodnotou hlavního hesla je prázdný řetězec – „“ [36] a podle statistiky od Mozilla přibližně jen 0,5 % uživatelů má nastaveno vlastní hlavní heslo [38].

NSS v „key4.db“ ukládá zašifrovaný 3DES klíč, sůl pro SHA1 a SHA256, počet iterací. K ověřeni platnosti uživatelem zadaného hesla NSS zašifruje ověřovací řetězec „password-check“ pomocí odvozeného klíče a také uloží do „key4.db“ s dalšími informacemi [39]. Sůl pro SHA1 v obou případech (3DES klíč a ověřovací řetězec) je stejná, pro SHA256 se liší [37]. Celý postup zpracování hlavního hesla je znázorněn na obrázku 3.1.



Obrázek 3.1: Schéma zpracování hlavního hesla, „metaData“ a „nssPrivate“ – tabulky v sqlite3 souboru „key4.db“, SDR key – 3DES klíč generovány prohlížečem [37]

3.2 Uložení dat

Data potřebná pro dešifrování přihlašovacích údajů jsou uložena v souboru „key4.db“ v tabulkách „metaData“ a „nssPrivate“. V první je zašifrovaný ověřovací řetězec „password-check“, ve druhé je mimo jiné uložen šifrovaný 3DES klíč.

Struktura získaná z „metaData“ bude obsahovat 2 části: „item1“ – globální sůl (pro SHA1) a „item2“ – část v ASN1 formátu. Po postupném dekódovaní podle ASN1 označení (použité typy jsou uvedeny v tabulce 3.2) vznikne struktura jako ve výpisu 3.1. Součástí dat jsou také OIDs (Object Identifiers), které

Listing 3.1: Dekódována položka „item2“ z tabulky „metaData“ se standardními hodnotami pro Mozilla Firefox verze 73 a výšší

```
SEQUENCE (A)
   A.SEQUENCE (B)
      B.OBJECTIDENTIFIER  1.2.840.113549.1.5.13
      B.SEQUENCE (C)
         C.SEQUENCE (D)
            D.OBJECTIDENTIFIER 1.2.840.113549.1.5.12
            D.SEQUENCE (E)
               E.OCTETSTRING salt, 32 bytes
               E.INTEGER iteration count
               E.INTEGER length of the key
               E.SEQUENCE (F)
                  F.OBJECTIDENTIFIER  1.2.840.113549.2.9
         C.SEQUENCE (G)
            G.OBJECTIDENTIFIER  2.16.840.1.101.3.4.1.42
            G.OCTETSTRING IV, 14 bytes
   A.OCTETSTRING cipher text, 16 bytes
```

ukazují na použité algoritmy (v tabulce 3.1 k nim jsou uvedeny).

Ve výpisu 3.1 je znázorněna struktura 2. části získaných z „metaData“ informací. Z nich plyne, že jsou použity PBES2 (Password Based Encryption Schemes) a PBKDF2 s HMAC-SHA256 pro derivaci klíče. V části pro PBKDF2 (E) jsou uložena data potřebná pro zpracování klíče: sůl délky 32 bajtů, počet iterací, délka klíče – 32 bajtů (potřebných pro AES-256) a samotný OID pro HMAC-SHA-256.

Další část (C) obsahuje OID pro AES-256-CBC a inicializační vektor k tomu. AES potřebuje IV délky bloku (16 bajtů), ale v dekódovaném OctetString jich je pouze 14. Před použitím NSS na začátek řetěze ještě 2 bajty – ”0x04”a ”0x0e”, aby to vypadalo jako data v ASN1 formátu – typ a délku (0x04 – OctetString (typ), 0x0e (14 dec – délka)). Příčinou je zpětná kompatibilita se starými verzemi NSS, které měly problém v dekódovací části [40].

V tabulce „nssPrivate“ je uložená struktura pro 3DES klíč. Ta je skoro stejná jako pro ověřovací řetězec, ale neobsahuje globální sůl (v SHA1 bude použita sůl z „metaData“ – „item1“).

3.3 Neshoda v počtu iterací

Během rozboru jsem zkusil napodobit útok hrubou silou. K tomu jsem použil mechanismus s šifrovanou ověřovací frází, kdy heslo projde přes odvozovací funkci a pak bude použito k dešifrování. Pokud výsledkem bude řetězec „password-check“ – správné heslo je nalezeno. Výsledkem kontroly bylo přibližně 750 000 hesel za minutu (12 500 za sekundu). Počet pokusů vypadal docela

Tabulka 3.1: Názvy algoritmů k identifikačním kódům použitým v „key4.db“

OIF code	algorithm
1.2.840.113549.1.5.13	PBES2 encryption scheme
1.2.840.113549.1.5.12	PBKDF2 key derivation algorithm
1.2.840.113549.2.9	HMAC-SHA-256 message authentication scheme
2.16.840.1.101.3.4.1.42	256-bit Advanced Encryption Standard (AES) algorithm with Cipher-Block Chaining (CBC) mode of operation

Tabulka 3.2: ASN1 typy, použité při dekódování uložené v „key4.db“ struktury [41]

Tag(hex)	Type
02	INTEGER
04	OCTET STRING
06	OBJECT IDENTIFIER
10 nebo 30	SEQUENCE

vysoký s ohledem na to, že jsem nepoužil např. GPU pro počítaní hašů.

Pro odvození klíče je standardně použito 10 000 iterací (podle dokumentace [27, 36]), ale jenom za podmínky, že uživatel nastaví vlastní heslo. Pro výchozí prázdný řetězec NSS nebude provádět 10 000 iterací, místo toho bude jen 1 [36]. Po ověření „key4.db“ s nastaveným heslem pro 84. verzi se však hodnota z nějakého důvodu rovnala 1.

Listing 3.2: Soubor „sftkpwd.c“ (verze 71.), použití 3DES pro šifrování

```
SECStatus
sftkdb_EncryptAttribute(PLArenaPool *arena, SECItem *passKey,
 SECItem *plainText, SECItem **cipherText)
{...
 cipherValue.alg = SEC_OID_PKCS12_PBE_WITH_SHA1_AND_TRI–
 PLE_DES_CBC;
 cipherValue.salt.len = SHA1_LENGTH;
 cipherValue.salt.data = saltData;
 ...}
```

Ještě na začátku hledání informací o uložení hesel v prohlížecích jsem objevil článek s popisem problému implementace odvozovací funkce z hesla v Mozilla Firefox. Tehdy byla používána 1 iterace hašovací funkce SHA1. Jako závěr byla zmíněna aktualizace mechanismu odvození. K funkci SHA1 byla

přidána aktuální PBKDF2 a 10 000 iterací. Podle autora článku to odpovídalo realitě pro verzi 72 [42].

Soubory „sftkdb.c“ a „sftkpwd.c“, ze kterých jsou uvedené výpisy, lze najít v příloze k práci a jsou přístupné na stránce Firefox [43].

Pro verzi 71 ještě nebyl změněn mechanismus zpracování hlavního hesla. Byla použita pouze funkce SHA1 a počet iterací nebyl udržován (byl stale 1). Pro 3DES klíč a ověřovací frázi stejně jako pro přihlašovací údaje byl použit 3DES-CBC-EDE. Ve výpisu 3.2 je část funkce, která je použita pro šifrování mimo jiné i 3DES klíče. Ta nevyžaduje počet iterací k zpracování dat a „cipherValue.alg“ ukazuje na použitý algoritmus.

Listing 3.3: Soubor „sftkdb.c“ (verze 71.), bez podmíněných bloků s proměnnou „NSS_DISABLE_DBM“

```
switch (dbType) {
    case NSS_DB_TYPE_LEGACY:
        crv = sftkdbCall_open(confdir, certPrefix, keyPrefix,
                    8, 3, flags, noCertDB ? NULL : &certSDB,
                                        noKeyDB ? NULL : &keySDB);
        break;
    case NSS_DB_TYPE_MULTIACCESS:
        crv = sftkdbCall_open(configdir, certPrefix,
                                        keyPrefix, 8, 3, flags,
                            noCertDB ? NULL : &certSDB,
                            noKeyDB ? NULL : &keySDB);
        break;
    case NSS_DB_TYPE_SQL:
    case NSS_DB_TYPE_EXTERN: /* SHOULD open a loadable db */
        crv = s_open(confdir, certPrefix, keyPrefix, 9, 4,
                    flags, noCertDB ? NULL : &certSDB,
                                noKeyDB ? NULL : &keySDB,
                                &newInit);
```

Ve výpisu 3.3 je segment funkce „sftk_DBInit ze souboru „sftkdb.c“ pro otevření databáze s zašifrovanými daty. Pro danou verzi neobsahuje žádný podmíněny blok „#ifndef“.

Od verze 72 se ukládá počet iterací pro SHA1. Hodnota bude stanovena na základě výsledku funkce „getPBEIterationCount“. Standardní hodnotou je 10 000 a lze je měnit pomocí nastavení globálních proměnných. Použití počtu iterací lze najít např. ve změněné funkci „sftkdb_EncryptAttribute“ ve výpisu 3.5, která teď jako parametr přijímá počet iterací, na rozdíl od verze 71.

Do souboru „sftkdb.c“ funkce „sftk_DBInit“ byla přidána proměnná „legacy“ (výpis 3.4), která na základě toho, jakou databázi se podaří otevřít, mění svou hodnotu. Standardně je s hodnotou „PR_TRUE“, ale pokud se nepodaří otevřít starou „key3.db“ a „cert8.db“, bude nastavena na „PR_FALSE“.

Hodnota dané proměnné pak bude zapsána do jiné – „usesLegacyStorage “, která pak ovlivní funkci změny/nastavení hesla.

Listing 3.4: Soubor „sftkdb.c“ (verze 72.), doplnění o novou proměnnou „legacy“.

```
 PRBool legacy = PR_TRUE;
...
    case NSS_DB_TYPE_SQL:
    case NSS_DB_TYPE_EXTERN: /* SHOULD open a loadable db */
       crv = s_open(confdir, certPrefix, keyPrefix, 9, 4,
                            flags, noCertDB ? NULL : &certSDB,
                             noKeyDB ? NULL : &keySDB, &newInit);
      legacy = PR_FALSE;
...
```

Listing 3.5: Soubor „sftkpwd.c“ (verze 72.), po přidaní použití iterací

```
...
static const int NSS_MP_PBE_ITERATION_COUNT = 10000;
...
SECStatus
sftkdb_EncryptAttribute(PLArenaPool *arena, SECItem *passKey,
 int iterationCount, SECItem *plainText, SECItem **cipherText)
{ ...
     cipherValue.alg = SEC_OID_PKCS12_PBE_WITH_SHA1_AND_TRI–
     PLE_DES_CBC;
     cipherValue.salt.len = SHA1_LENGTH;
     cipherValue.salt.data = saltData;
   ...}
```

Pro verzi 73 se algoritmem šifrování 3DES klíče (přihlašovací údaje) a ověřovací fráze stal AES-256-CBC s hašovací funkcí SHA256 pro PBKDF2. Ve funkci „sftkdb_EncryptAttribute“ z výpisu 3.6 teď na základě možnosti uložení metadat (legacy format neposkytne uložení počtu iterací) bude vybrán postup. Pro soubor „sftkdb.c“ nebyly provedeny žádné změny spojené s problémem (od verze 72).

Ve verzi 74 se v důsledku opravy [44, 45] a odstranění „lgglue“ knihovny z kompilace přidávají podmíněné bloky s proměnnou „NSS_DISABLE_DBM“. A dříve zmíněná část souboru „sftkdb.c“ teď vypadá jako výpis 3.7. Kontroly otevření starých formátů souborů nebudou provedeny, pokud je daná proměna nastavena na 1. Hodnota proměnné „legacy“ je standardně „PR_TRUE“, a jak lze vidět, může změnit hodnotu na „PR_FALSE“ pouze v dalším podmíněném bloku, do kterého se ale nedostane (může nabýt hodnoty „PR_FALSE“ pouze v daném místě). Hodnota se pak propaguje do proměnné „usesLegacyStorage“, která je použita v souboru

Listing 3.6: Soubor „sftkpwd.c“ (verze 73.), použití AES-256-CBC pro šifrování

```
SECStatus
sftkdb_EncryptAttribute(PLArenaPool *arena, SFTKDBHandle
*handle, SDB *db, SECItem *passKey, int iterationCount,
 CK_OBJECT_HANDLE id, CK_ATTRIBUTE_TYPE type,
 SECItem *plainText, SECItem **cipherText)
{...
 if (sftkdb_useLegacyEncryption(handle, db)) {
        cipherValue.alg = SEC_OID_PKCS12_PBE_WITH_SHA1_AND_TRI–
        PLE_DES_CBC;
        cipherValue.salt.len = SHA1_LENGTH;
        hashType = HASH_AlgSHA1;
    } else {
        cipherValue.alg = SEC_OID_AES_256_CBC;
        cipherValue.salt.len = SHA256_LENGTH;
        hashType = HASH_AlgSHA256;
    }
...}
```

„sftkpwd.c“. Ve výpisu 3.8 je uvedená část funkce „sftkdb_ChangePassword“, která v závislosti na hodnotě může použit 1 iteraci (funkce „sftk_isLegacyIterationCountAllowed()“ z výpisu vrátí „false“ pokud „NSS_ALLOW_LEGACY-_DBM_ITERATION_COUNT“ proměnná je nenastavena (standardně) nebo hodnotou není „0“). Ve výsledku pro odvození klíče byla pořad používána jen 1 iterace.

S přechodem z jedné verze Mozilla Firefox na novější se neaplikuje žádný automaticky mechanismus aktualizace uložených dat v souboru „key4.db“. Pokud např. je nainstalována verze 71 a má nastavené hlavní heslo, tak bude použito šifrování 3DES, SHA1 a 1 iterace. Při instalaci verze 73, kde je použit správný počet iterací (10 000), nebudou data aktualizována. Struktura a hodnoty zůstanou od verze 71 – 3DES pro šifrování klíče a ověřovací fráze, 1 iterace při odvození. Taková situace není jen pro daný případ, minimálně pro aktualizace mezi verzemi 71-87 nebudou automaticky kontrolovány údaje. Tím může prohlížeč používat starší mechanismy a hodnoty – 3DES místo AES nebo 1 iteraci místo 10 000 pro nastavené hlavní heslo i po instalaci opravené verze.

Dřív v kapitole byla zmiňována hodnota 12 500 kontrol hesel za sekundu pro 1 iteraci. Po odstranění nadbytečných kontrol a přepisování proměnných se čas zlepšil a teď za minutu program zvládl zkontrolovat 1,6 milionu možných hesel – přibližně 27 000 tisíc za sekundu. Daný výsledek platí pro jednu spuštěnou instanci (1 vlákno) programu bez použití nějakých sofistikovanějších technik jako např. použití GPU při výpočtu hašů. Daný test byl prováděn na notebooku s procesorem Kaby Lake i7-7700HQ, 4 jádra, s zapnutou In-

Listing 3.7: Soubor „sftkdb.c“ (verze 74.), přidaní podmíněných bloků s proměnnou „NSS_DISABLE_DBM“

```
switch (dbType) {
#ifndef NSS_DISABLE_DBM
    case NSS_DB_TYPE_LEGACY:
        crv = sftkdbCall_open(confdir, certPrefix, keyPrefix,
                    8, 3, flags, noCertDB ? NULL : &certSDB,
                                        noKeyDB ? NULL : &keySDB);
        break;
    case NSS_DB_TYPE_MULTIACCESS:
        crv = sftkdbCall_open(configdir, certPrefix, keyPrefix,
                      8, 3, flags, noCertDB ? NULL : &certSDB,
                                        noKeyDB ? NULL : &keySDB);
        break;
#endif /* NSS_DISABLE_DBM */
    case NSS_DB_TYPE_SQL:
    case NSS_DB_TYPE_EXTERN: /* SHOULD open a loadable db */
        crv = s_open(confdir, certPrefix, keyPrefix, 9, 4,
                             flags, noCertDB ? NULL : &certSDB,
                             noKeyDB ? NULL : &keySDB, &newInit);

#ifndef NSS_DISABLE_DBM
     legacy = PR_FALSE;

}
```

tel Turbo Boost Technology, 16 GB RAM. Pro případ spuštění několika instancí zároveň byl nejlepší výsledek při 6 s průměrným počtem kontrol 134 000 za sekundu.

Pro lepší pochopení vztahu času a počtu iterací je vyplněna tabulka 3.3 pro 1 iteraci a 27 000 kontrol za sekundu. Pro situaci, kdy máme nastavený počet iterací na 10 000 se počet kontrol zmenšuje na přibližně 95 za sekundu – přibližně 280krát pomaleji. Pro minimálně dlouhé heslo podle NIST (8 symbolů) s použitou např. jen množinou malých písmen by kontrola všech možných hesel trvala 89 dnů, pokud je počet iterací 1. Pro případ 10 000 iterací se čas zvýší na 67 let podle tabulky 3.4.

3.4 Bugreport

Po několika dalších kontrolách jsem poslal hlášení o chybě s popisem problému použíti nesprávného počtu iterací. Do něj jsem doplnil příklady toho, jaká data jsou použitá během procesu odvození a pak uložena v „key4.db“. Kopie stránky z „bugzilla.mozilla.org“, kde hlášeni bylo vytvořeno, je v přiloze k práci. Část

Tabulka 3.3: Čas hledání hesel (27 000 kontrol za sekundu)

	Složení hesla			
Délka hesla	0–9	a–z	a–z, A–Z	0-9, a–z, A–Z
6 znaků	37 sekund	3 hodin	8 dnů	24 dnů
7 znaků	6 minut	3 dnů	1,2 let	4,1 roků
8 znaků	1 hodina	89 dnů	63 let	260 let
9 znaků	10 hodin	6 let	3,3 tisíc let	16,1 tisíc let
10 znaků	4,2 dnů	165 let	169 tisíc let	985 tisíc let
11 znaků	43 dnů	4,3 tisíce let	8,8 milionů let	61,1 milionů let
12 znaků	1,1 rok	112 tisíc let	459 milionů let	3,8 miliardy let
13 znaků	11 let	2,9 milionů let	23 miliardy let	234 miliardy let
14 znaků	117 let	75 milionů let	1,2 bilionů let	14 bilionů let
15 znaků	1,1 tisíc let	1,9 miliardy let	64 bilionů let	903 bilionů let

Tabulka 3.4: Čas hledání hesel (95 kontrol za sekundu)

	Složení hesla			
Délka hesla	0–9	a–z	a–z, A–Z	0-9, a–z, A–Z
6 znaků	3 hodiny	37 dnů	6,5 let	18 let
7 znaků	1,2 dnů	29 hodin	343 roků	1,1 tisíce let
8 znaků	12,1 dnů	69 let	17,8 tisíc let	72 tisíce let
9 znaků	121 dnů	1,8 tisíc let	927 tisíce let	4,5 milionů let
10 znaků	3 roků	47 tisíc let	48 milionů let	280 milionů let
11 znaků	33 roků	1,2 milionů let	2,5 miliardy let	17,3 miliardy let
12 znaků	333 let	31 milionů let	130 milard let	1 bilion let
13 znaků	3,3 tisíce let	828 milionů let	6,7 bilionů let	66 bilionů let
14 znaků	33 tisíce let	21,5 miliard let	352 bilionů let	4 biliardy let
15 znaků	333 tisíce let	559 miliard let	18 biliard let	256 biliard let

Listing 3.8: Soubor „sftkpwd.c“ (verze 74.), vliv proměnné „usesLegacyStorage“ na počet iterací při odvození klíče.

```
SECStatus
sftkdb_ChangePassword(SFTKDBHandle *keydb,
                      char *oldPin, char *newPin,
                      PRBool *tokenRemoved)
{...
    if (newPin && *newPin == 0) {
        iterationCount = 1;
    } else if (keydb->usesLegacyStorage
                    && !sftk_isLegacyIterationCountAllowed())
     {
        iterationCount = 1;
      }
...}
```

hlašení z webu je také na obrázku 3.4, kde lze najít název, číslo hlášení, stav, typ a další informace. Přístup k stránce s hlášením je omezen kvůli tomu, že je spojen s bezpečností aplikace.

Samotné hlášení bylo odeslano 27. 12. 2020. Za týden byl už objeven problém – chyba ve funkci „sftk_DBInit“ souboru „sftkdb.c“. Pokud je proměnná „NSS_DISABLE_DBM“ definována, což je standardně, tak vždy bude zvolena cesta zpracovaní jako pro legacy formát souborů. To pak vedlo na chybné použití 1 iterace v funkci „sftkdb_ChangePassword“ v souboru „sftkpswd.c“ při změně hlavního hesla uživatelem. Pro opravu byla připravena změna „sftk_DBInit“ a vytvořen test, aby znovu nedošlo k situaci použití menšího počtu iterací.

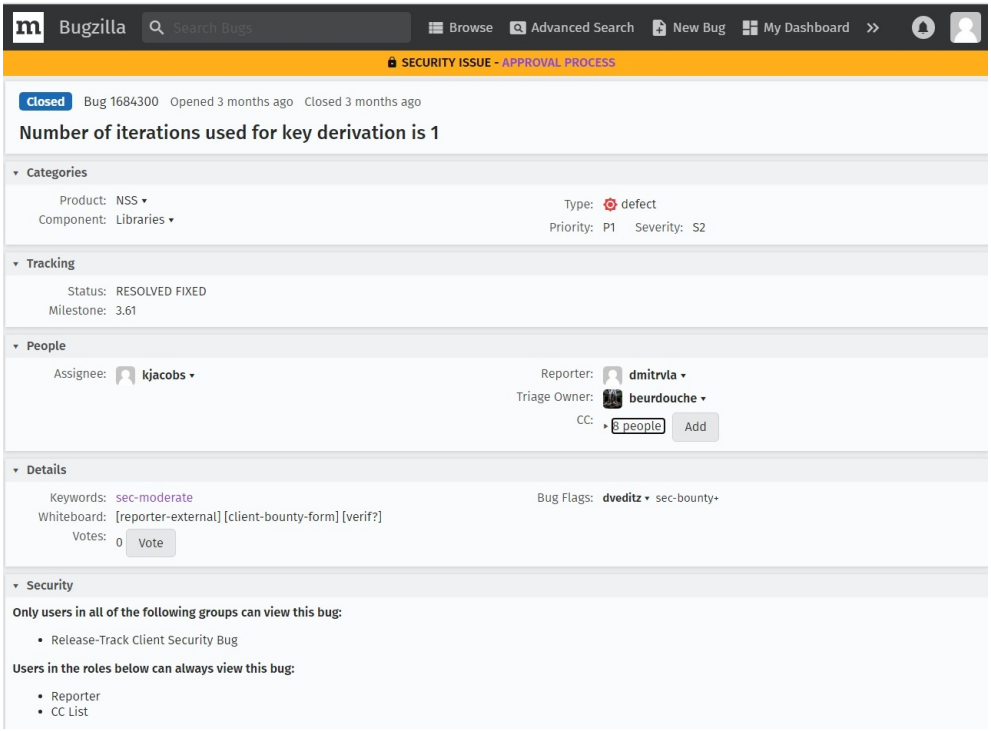
Milestone pro danou chybu byl nastaven na verzi 3.61 knihovny NSS, ve které to bylo opraveno. Danou verzi samotný prohlížeč začal používat až od verze 86, tedy 23. 2. 2021 [9].

3.5 86. verze

V případě odcizení souborů „key4.db“ a „logins.json“ by chyba uvedená dříve způsobila zkrácení času, který by případný útočník potřeboval pro prolomení hesel hrubou silou. Opravená verze 86 byla publikována 23. února roku 2021, skoro 2 měsíce po nahlášení chyby. Ve verzi 86 jsou 2 změny spojené s hlášením: samotná oprava části s „NSS_DISABLE_DBM“ a test, který musí zabránit opakování situace s nesprávným počtem iterací při odvození klíče.

Před změnou část kódu souboru „sftkdb.c“ funkce „sftk_DBInit“ vypadala jako ve výpisu 3.7.

Jak bylo zmíněno dřív, problém byl v tom, že NSS kvůli podmínce závislé



Obrázek 3.2: Screenshot stránky reportu z webu “bugzilla.mozilla.org“

na „NSS_DISABLE_DBM“ ani nemělo možnost použít správný počet iterací. Proměnná „legacy“ byla právě v podmíněném bloku „#ifndef NSS_DISABLE-_DBM“. Ve verzi souboru pro Mozilla Firefox 86 byla přemístěna možnost změny „legacy“ před tento blok jako ve výpisu 3.9.

Listing 3.9: Po opravě

```
#endif /* NSS_DISABLE_DBM */
    case NSS_DB_TYPE_SQL:
    case NSS_DB_TYPE_EXTERN: /* SHOULD open a loadable db */
        crv = s_open(confdir, certPrefix, keyPrefix, 9, 4,
                             flags, noCertDB ? NULL : &certSDB,
                             noKeyDB ? NULL : &keySDB, &newInit);
        legacy = PR_FALSE;

#ifndef NSS_DISABLE_DBM
```

Problém ale může zůstat i po nainstalování opravené verze 86 nebo vyšší. Pokud uživatel nepřenastaví své hlavní heslo bude použita dříve uložena hodnota a de-fakto oprava pro starší profil nenastane. *„Obyčejná aktualizace Firefox nemění počet iterací.“ [...] „Jen pokud použiješ změnu/nastavení hlavního hesla s novým kódem bude použit vyšší počet iterací.“* [46](překlad vlastní) I

když tento komentář z „bugzilla.mozilla.org“ je rok starý, odpovídá výsledkům kontrol při aktualizaci do různých verzí prohlížeče. Ve zdrojových kódech se také nepodařilo najít nějaký mechanismus, který by za to odpovídal.

Test pro kontrolu správného počtu použitých iterací v PBKDF2 byl přidán do souboru „softoken_gtest.cc“ ve složce „gtests“ pro NSS. Ten otevře databázi „key4.db“ a zkontroluje použitý PBE algoritmus a počet iterací s nastavením pro testování heslem „password“. Získané hodnoty budou porovnány s očekávanými bez dekódování z ASN1 formátu. Očekávanými výsledky jsou AES-256-CBC s HMAC-SHA256 a počet iterací rovny 10 000.

3.6 Zvýšení počtu iterací

Listing 3.10: Zvýšení počtu iterací nastavením proměnné prostředí.

```
terminal#
> export NSS_MIN_MP_PBE_ITERATION_COUNT=40001
> env | grep "NSS"
> firefox



/etc/environment
...
NSS_MIN_MP_PBE_ITERATION_COUNT="40001".
```

Při potřebě zvýšení počtu iterací v NSS lze použit proměnnou prostředí – „NSS_MIN_MP_PBE_ITERATION_COUNT“ [27]. Pro kontrolu funkce lze provést příkazy z výpisu 3.10. Nastaví to proměnnou na hodnotu 40 001 iterací, další příkaz vypíše hodnotu a název, pokud proměnná byla úspěšně nastavena. Následující příkaz spustí Firefox z daného terminálu. Ke změně použitého počtu při odvození klíče je potřeba přenastavit hlavní heslo (lze použit i stejné). K trvalému použití změněného počtu lze příkaz „export“ z výpisu přidat do souboru „ /.bashrc“ nebo nastavit hodnotu přes soubor „/etc/environment“ doplněním řádky z výpisu 3.10.

Kapitola 4

Implementace nástroje pro zpracování hesel

V této kapitole je popsána implementace programu pro zpracování přihlašovacích údajů uložených v Mozilla Firefox.

4.1 Již existující programy

Mozilla Firefox od verze 80 nabízí možnost importu hesel z „.csv“ souborů s určitým formátem a export od verze 79 [47]. Kromě toho existuje několik open-source programů, které nabízí funkce pro export uživatelem uložených hesel. Jsou to např. LaZagne [48] nebo ffpass [49].

4.2 Popis CLI programu

Moje implementace pro Python bez použití knihovny NSS má několik funkcí:

- Export uložených hesel do csv souboru
- Import ze souboru do profilu
- Export z jednoho a zároveň import do dalšího profilu
- Pokus o získání hlavního hesla hrubou silou

Program lze spustit v interaktivním režimu nebo zadat parametry, které budou použity během zpracování. Pro získaní seznamu dostupných argumentů je potřeba do terminálu zadat příkaz „python3 main.py -h“.

Při interaktivním režimu při výběru libovolné varianty bude možnost použit „profiles.ini“ soubor (standardní umístění nebo s explicitně zadanou cestou). Ze souborů získáný seznam profilů bude vypsán do terminálu, ze kterého bude

možno vybrat. Další možnosti je zadaní celé cesty k profilům. Ve výpisu 4.1 je ukázáno několik příkladů použití neinteraktivního režimu s argumenty.

Pro první 3 varianty je potřeba zadat hlavní heslo pro vybraný profil (v 3. případě je potřeba vybrat i profil, kam budou importovány údaje, a zadat i další hlavní heslo). Pro variantu s hledáním hesla je dvě možnosti: použit předem vytvořený soubor (každý řádek je jedním možným heslem), nebo vybrat délku možného hesla a program postupně bude generovat řetězce z množiny číslic a anglické malé/velké abecedy. Na základě počtu iterací použitých při odvození klíče se také změní frekvence výpisu. Pro 1-9999 iterací bude vypisovat počet již zkontrolovaných hesel jednou za 1 milion, pro 10 000 a víc to bude jednou za 10 000 zkontrolovaných hesel.

Listing 4.1: Příklady použití programu s argumenty.

```
#Extrahování údaju z profilu
> python3 main.py -e "~/firefox_profile"

#Extrahování z 1. do 2. profilu
> python3 main.py -ms "~/firefox_folder/first_profile"
\ -md "~/firefox_folder/second_profile"

#Hledání hesla s použitím předem připraveného souboru
> python3 main.py -f  "~/firefox_profile\ -d „~/namelist.txt"
```

4.3 Podrobnější popis implementace

Program funguje na Windows 10 a Linux (Debian 20.04.1). Mozilla Firefox nepoužívá nástroje nabízené přímo operačním systémem, jako jsou DPAPI nebo Keychain, proto byla potřeba výměny pouze cest k složkám instalace Firefox pro různé platformy.

Implementace se skládá z několika modulů:

main.py – rozcestník, který nabízí výběr mezi funkcemi exportu, importu a hledání hesla

support_functions.py – podpůrné funkce pro další moduly programu

- *decode_credentials_object* – dekódování dat z ASN1 a Base64 formatu uložených v souboru „logins.json“
- *decode_tuples* – dekódování uživatelských jmen a hesel ze seznamů položek (username, password, hostname)

- *decide_which_profile_ini* – výběr „profiles.ini“ ze standardní cesty nebo explicitně zadané uživatelem
- *decode_asn1_seqs* – dekódování struktury z key4.db, kontrola OIDs použitých algoritmů a vrácení potřebných dat pro odvození klíče a dešifrování ověřovací fráze a 3DES klíče
- *get_profiles_list* – načtení jmen profilů
- *add_prepared_logins* – přidaní importovaných údajů do výsledného seznamu [50]
- *encode_object_asn1* – zakóduje do ASN1 podoby předaná šifrovaná data (heslo nebo uživatelské jméno), IV a identifikátor použitého algoritmu. Použitá struktura je uvedena ve výpisu 4.2 [37].

Listing 4.2: ASN1 struktura uloženého hesla/uživatelského jména v logins.json

```
SEQUENCE (A)
    A.OCTETSTRING - key_id
    A.SEQUENCE (B)
        B.OBJECTIDENTIFIER - 1.2.840.113_549.3.7
        B.OCTETSTRING - IV
    A.OCTETSTRING - cipher text
```

crypto_functions.py – nabízí potřebné šifrovací funkce a funkce odvození klíče.

- *make_key_from_master_password* – pomocí SHA1 a PBKDF2 odvodí šifrovací klíč pro AES-256-CBC
- *decrypt_key* – dešifruje uloženou ověřovací frázi nebo 3DES klíč pomocí AES-256-CBC
- *check_master_password* – zkontroluje zadané hlavní heslo
- *get_decrypt_key* – řídí proces získání 3DES klíče z key4.db
- *decrypt_logins_3des* – dešifruje uživatelské jméno a heslo pomocí 3DES
- *encrypt_data_des3* – generuje IV a zašifruje předány objekt – heslo nebo uživatelské jméno

file_io.py – zpracování potřebných operací s načítáním nebo zápisem dat z a do „.json“ souboru.

- *get_logins_json* – načte celou strukturu ze souboru „logins.json“
- *read_import_logins* – načte hesla připravená pro import z „data.json“

- *get_login_tuples* – ze souboru „logins.json“ načte pouze zakódovaná uživatelská jména, zakódovaná hesla a hostname
- *write_to_json* – strukturu doplněnou o nová hesla zapíše zpátky do „logins.json“

check_data.py – načte soubor „key4.db“ a vypíše použitý algoritmus šifrování 3DES klíče a počet iterací.

arguments_mode.py – řídí funkce programu s vstupními argumenty

find_master_password_in_dict_file.py, **extract_from_one_to_another_profile.py**, **extract_from_one_to_another_profile.py**, **import_credentials_to_profile.py**, **extract_credentials_from_profile.py**, **check_data.py** představují implementaci funkcí exportu, importu, hledání hesla a kontroly použitého počtu iterací a šifrovacího algoritmu. Každá používá funkce z uvedených dříve modulů.

4.4 Výsledek

Použitím inforamcí ziskaných v druhé a třetí kapitole byl vytvořen program, který dokáže pracovat s 73. a novějšími verzemi prohlížeče Mozilla Firefox. Má interaktivní režím a režim s argumenty zadavanými do terminálu. Kromě funkcí exportu a importu program také nabízí možnost kontroly použitého šifrovacího algoritmu nebo počtu iterací při odvození hesla a také hledání hesla hrubou silou.

Závěr

Práce se zabývala správci hesel vestavěnými do webových prohlížečů. V první kapitole byly rozebrány postupy, které aplikují nejpoužívanější prohlížeče jako např. Google Chrome, Safari, Microsoft Edge, Mozilla Firefox atd. V druhé byly porovnány „best practices“ od NIST a postupy použité v prohlížečích. V třetí byla hlouběji rozebrána implementace zpracování a uložení hesel v Mozilla Firefox. V ní byla také během porovnání s dokumentací objevena chyba v počtu iterací použitých v procesu odvození klíče z hlavního hesla (pro kterou bylo odesláno hlášení o chybě). Během dalšího rozboru bylo také zjištěno, že dána chyba je přitomná od verze 74 (od 10 března 2020) a až do verze 85 (ve verzi 86 byla opravena). Zajímavým detailem také je, že při aktualizaci Mozilla Firefox není měněna struktura uložených dat. Pokud uživatel má nespravný počet iterací, při přestupu do verze 86 nebo vyšší nebude upozorněn a chyba se neopraví dokud uživatel nezmění hlavní heslo (kdy dojde ke kontrole počtu iterací a algoritmů). V čtvrté kapitole jsou popsány funkce implementace CLI programu pro export přihlašovacích údajů z profilů a import do lokálního správce hesel Mozilla Firefox.

Literatura

[1] Burr, W. E.; Dodson, D. F.; Polk, W. T.: NIST Special Publication 800-63: Digital Identity Guidelines Frequently Asked Questions [online]. 2004, [Cited 2021-04-06]. Dostupné z: https://csrc.nist.gov/CSRC/media/Publications/sp/800-63/ver-10/archive/2004-06-30/documents/sp800-63-v1-0.pdf

[2] National Institute of Standards and Technology (NIST): NIST Special Publication 800-63B [online]. 2017, [Cited 2021-04-06]. Dostupné z: https://pages.nist.gov/800-63-3/sp800-63b.html

[3] StatCounter: Browser Market Share Worldwide [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://gs.statcounter.com/browser-market-share

[4] Elcomsoft Co.Ltd.: Elcomsoft Internet Password Breaker 3.10 extracts Edge Chromium passwords, updates Chrome support: Elcomsoft Co.Ltd. [online]. 2020, [Cited 2021-03-14]. Dostupné z: https://www.elcomsoft.es/news/741.html

[5] Google LLC: Chrome Security FAQ [online]. 2019, [Cited 2021-03-14]. Dostupné z: https://chromium.googlesource.com/chromium/src/+/master/docs/security/faq.md#Does-the-Password-Manager-store-my-passwords-encrypted-on-disk

[6] XenArmor Security Solutions Pvt Ltd.: Password Secrets of Popular Web Browsers [online]. 2019, [Cited 2021-04-06]. Dostupné z: https://xenarmor.com/password-secrets-of-popular-web-browsers/

[7] Kirk, E.: Autofill Blog #2: Password Security [online]. Nov 2020, [Cited 2021-04-06]. Dostupné z: https://techcommunity.microsoft.com/t5/articles/autofill-blog-2-password-security/m-p/963847

[8] Mozilla Corporation: Overview of NSS [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS/Overview

[9] Mozilla Corporation: NSS:Release Versions [online]. [Cited 2021-04-06]. Dostupné z: https://wiki.mozilla.org/NSS:Release_Versions

[10] Jordan: How Browsers Store Your Passwords (and Why You Shouldn't Let Them) [online]. 2013, [Cited 2021-04-06]. Dostupné z: http://raidersec.blogspot.com/2013/06/how-browsers-store-your-passwords-and.html

[11] Microsoft Corporation: Windows Data Protection [online]. 2010, [Cited 2021-04-06]. Dostupné z: https://docs.microsoft.com/en-us/previous-versions/ms995355(v=msdn.10)

[12] Pegasystems Inc.: Encryption settings for Pega Robotic Automation [online]. Jan 2017. Dostupné z: https://community.pega.com/knowledgebase/articles/pega-rpa/encryption-settings-pega-robotic-automation

[13] Passcape Software: Password Recovery Software [online]. 2012, [Cited 2021-04-06]. Dostupné z: https://www.passcape.com/index.php?section=docsys&;cmd=details&;id=28

[14] Microsoft Corporation: CryptUnprotectData function (dpapi.h) - Win32 apps [online]. 2018, [Cited 2021-04-06]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/api/dpapi/nf-dpapi-cryptunprotectdata

[15] Khanna, A. S.: amandeepsinghkhanna/chrome_password_retriever [online]. Jul 2020, [Cited 2021-04-06]. Dostupné z: https://github.com/amandeepsinghkhanna/chrome_password_retriever

[16] Turan, M. S.; Barker, E.; Burr, W.; aj.: Recommendation for Password-Based Key Derivation [online]. 2010, [Cited 2021-04-06]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf

[17] Moriarty, K. M.; Kaliski, B.; Rusch, A.: PKCS #5: Password-Based Cryptography Specification Version 2.1 [online]. 2017, [Cited 2021-04-06]. Dostupné z: https://tools.ietf.org/html/rfc8018#appendix-C

[18] Dworkin, M.: Recommendation for Block Cipher Modes of Operation [online]. 2001, [Cited 2021-04-06]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

[19] Apple Inc.: Secure Enclave [online]. 2021, [Cited 2021-03-14]. Dostupné z: https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web

[20] Apple Inc.: Apple Platform Security [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://support.apple.com/en-au/guide/security/aside/sec7ca604c5e/1/web/1

[21] Graef, N. D.: Gnome Keyring Wiki [online]. 2020, [Cited 2021-04-06]. Dostupné z: https://wiki.gnome.org/Projects/GnomeKeyring/SecurityFAQ

[22] McGrew, D. A.; Viega, J.: The Galois/Counter Mode of Operation (GCM) [online]. 2004, [Cited 2021-04-06]. Dostupné z: https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf

[23] Yongge Wang, P.: Public-Key Cryptography Standards: PKCS [online]. 2012, [Cited 2021-04-06]. Dostupné z: https://webpages.uncc.edu/yonwang/papers/pkcs.pdf

[24] Mozilla Corporation: NSS SQLite-based DB [online]. 2018, [Cited 2021-04-06]. Dostupné z: https://wiki.mozilla.org/NSS_SQLite-based_DB

[25] Mozilla Corporation: Use a Primary Password to protect stored logins and passwords [online]. 2020, [Cited 2021-04-06]. Dostupné z: https://support.mozilla.org/en-US/kb/use-primary-password-protect-stored-logins

[26] Apple Inc.: If your Mac keeps asking for the login keychain password [online]. Aug 2017, [Cited 2021-04-06]. Dostupné z: https://support.apple.com/en-us/HT201609

[27] Engert, K.: Increase NSS MP KDF default iteration count, by default for modern key4 storage, optionally for legacy key3.db storage [online]. 2019, [Cited 2021-04-06]. Dostupné z: https://bugzilla.mozilla.org/show_bug.cgi?id=1562671#c26

[28] Tuerk, A.: To stay secure online, Password Checkup has your back [online]. Oct 2019, [Cited 2021-04-06]. Dostupné z: https://blog.google/technology/safety-security/password-checkup/

[29] Mozilla Corporation: Find out if you've been part of a data breach [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://monitor.firefox.com/

[30] Mozilla Corporation: PasswordGenerator.jsm [online]. 2019, [Cited 2021-04-06]. Dostupné z: https://searchfox.org/mozilla-central/source/toolkit/components/passwordmgr/PasswordGenerator.jsm

[31] Google LLC: password_generator.cc [online]. 2021, [Cited 2021-04-13]. Dostupné z: https://github.com/chromium/chromium/blob/master/components/password_manager/core/browser/generation/password_generator.cc

[32] National Institute of Standards and Technology (NIST): NIST SP 800-63 Digital Identity Guidelines-FAQ [online]. 2020, [Cited 2021-04-06]. Dostupné z: https://pages.nist.gov/800-63-FAQ/#q-b12

[33] Google LLC: Chromium source code [online]. 2021. Dostupné z: https://github.com/chromium/chromium/tree/master/components/os_crypt

[34] Bolyard, N. B.: Master Password method [online]. 2007, [Cited 2021-04-06]. Dostupné z: https://groups.google.com/g/mozilla.dev.tech.crypto/c/KK5MPXw3hw4/m/yO6ct7PN9hkJ

[35] Mozilla Corporation: lowpbe.c [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://searchfox.org/mozilla-central/source/security/nss/lib/softoken/lowpbe.c

[36] Mozilla Corporation: sftkpwd.c [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://searchfox.org/mozilla-central/source/security/nss/lib/softoken/sftkpwd.c

[37] Kruglov, G.: Fennec logins & password data import #4067 [online]. 2019, [Cited 2021-04-06]. Dostupné z: https://github.com/mozilla-mobile/android-components/issues/4067

[38] Mozilla Corporation: Measurement Dashboard [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://telemetry.mozilla.org/new-pipeline/dist.html#!cumulative=0&end_date=2021-03-22&include_spill=0&keys=__none__!__none__!__none__&max_channel_version=nightly%252F88&measure=MASTER_PASSWORD_ENABLED&min_channel_version=nightly%252F55&processType=*&product=Firefox&sanitize=1&sort_by_value=0&sort_keys=submissions&start_date=2021-02-22&table=0&trim=1&use_submission_date=0

[39] Mozilla Corporation: keydb.c [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://searchfox.org/mozilla-central/source/security/nss/lib/softoken/legacydb/keydb.c

[40] Relyea, R.: [key4.db] IV size for aes256-CBC [online]. 2020, [Cited 2021-04-06]. Dostupné z: https://groups.google.com/g/mozilla.dev.tech.crypto/c/IdEJ7ZZy2Lo

[41] Cassel, L. N.; Austing, R. H.: Computer Networks and Open Systems An Application Development Perspective [online]. 2000, [Cited 2021-04-13]. Dostupné z: https://www.obj-sys.com/asn1tutorial/node10.html

[42] Palant, W.: Master password in Firefox or Thunderbird? Do not bother! [online]. Mar 2018, [Cited 2021-04-06]. Dostupné z: https://palant.info/2018/03/10/master-password-in-firefox-or-thunderbird-do-not-bother/

[43] Mozilla Corporation: Index of /pub/firefox/releases/ [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://ftp.mozilla.org/pub/firefox/releases/

[44] J.C.Jones: Crash in [@ RtlpWaitOnCriticalSection — RtlpDeCommitFreeBlock — nssutil3.dll — nssdbm3.dll — sftkdb_Update] [online]. 2020, [Cited 2021-04-13]. Dostupné z: https://bugzilla.mozilla.org/show_bug.cgi?id=1609673

[45] J.C.Jones: Bug 1609673 - Conditionally compile out all libnssdbm glue [online]. 2020, [Cited 2021-04-13]. Dostupné z: https://github.com/nss-dev/nss/commit/c046b1309bfdf11742c656d1370abdd87c6a4766#diff-7874a6c375e79f346f11d9dd0f7fa4b174542efa7b8039c104d6fe596150ca14

[46] Engert, K.: Having a Master Password makes Firefox password manager (both Lockwise and legacy) slow... [online]. 2020, [Cited 2021-04-06]. Dostupné z: https://bugzilla.mozilla.org/show_bug.cgi?id=1606992

[47] Mozilla Corporation: Import login data from a file [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://support.mozilla.org/en-US/kb/import-login-data-file

[48] AlessandroZ: AlessandroZ/LaZagne [online]. Mar 2021, [Cited 2021-04-06]. Dostupné z: https://github.com/AlessandroZ/LaZagne

[49] Louis, A.: louisabraham/ffpass [online]. Mar 2021, [Cited 2021-04-06]. Dostupné z: https://github.com/louisabraham/ffpass

[50] Mozilla Corporation: storage-json.js [online]. 2021, [Cited 2021-04-06]. Dostupné z: https://searchfox.org/mozilla-central/source/toolkit/components/passwordmgr/storage-json.js

Příloha A

Seznam použitých zkratek

AAD Additional Authenticated Data

AES Advanced Encryption Standard

ASN.1 Abstract Syntax Notation One

CBC Cipher Block Chaining

CLI Command Line Interface

CMAC Cipher-based Message Authentication Code

CTR Counter Mode

DPAPI Data Protection Application Programming Interface

GCM Galois/Counter Mode

GUI Graphical User Interface

GUID Globally Unique Identifier

HMAC Hash-based Message Authentication Code

IV Initialization Vector

LSB Least Significant Bit

MAC Message Authentication Code

NIST National Institute of Standards and Technology

NSS Network Security Services

OID Object Identifier

PBE Password-Based Encryption

PBKDF2 Password-Based Key Derivation Function 2

PKCS Public Key Cryptographic Standards

SHA Secure Hash Algorithm

SSL Secure Sockets Layer

3DES Triple DES (Data Encryption Standard)

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `bugreport` ..... složka s kopií hlášení chyby z bugzilla
- `old_versions` ..... soubory „sftkdb.c“ a „sftkpwd.c“ z Mozilla Firefox (71-74)