FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Bezpečnostní analýza programu KeePassXC
Student:	Michal Kavan
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

1) Seznamte se s problematikou bezpečné práce s hesly.
2) Proveďte rešerši známých programů pro správu hesel.
3) Zaměřte se na program KeePassXC (https://keepassxc.org). Vyhodnoťte jeho uživatelské prostředí ve vztahu k bezpečnosti práce s hesly. Navrhněte potenciální vektory útoku.
4) Prostudujte zdrojový kód aplikace zadané vedoucím práce vzhledem k zvoleným útočným vektorům. Bude-li to vhodné, otestujte je pomocí vhodných testovacích nástrojů.
5) Nalezené zranitelnosti zdokumentujte, vyhodnoťte jejich závažnost a navrhněte opatření k nápravě.
6) Diskutujte svoje zjištění.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 19. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Bezpečnostní analýza programu KeePassXC

Michal Kavan

Katedra počítačových systémů
Vedoucí práce: Ing. Josef Kokeš

13. května 2018

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Josefu Kokešovi za pozitivní přístup a cenné rady při kompletaci textu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. § 46 odst. 6 tohoto zákona 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 13. května 2018 .....

Odkaz na tuto práci

Kavan, Michal. *Bezpečnostní analýza programu KeePassXC*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá problematikou bezpečné práce s hesly a jejich ukládání. Poskytuje zároveň přehled současných řešení pro správu hesel. Cílem práce je provedení analýzy uživatelského prostředí správce hesel KeePassXC s ohledem na možná bezpečnostní rizika. Nalezená rizika jsou blíže analyzována přímo ve zdrojovém kódu programu. Při analýze bylo nalezeno jedno místo, kdy program neodpovídá standardům pro správnou práci s hesly. Po zhodnocení závažnosti této nalezené zranitelnosti byla navržená možná opatření vedoucí k nápravě.

Klíčová slova bezpečnostní analýza programu, správce hesel, KeePassXC, ukládání hesel, šifrování

Abstract

The main focus of this thesis is password management and secure password storage. It provides an overview of current password management solutions. The goal of this thesis is to perform a security assessment of KeePassXC password manager. Security assessment starts with a review of graphical interface and suspicious items are further analysed in the source code. The analysis confirmed one suspicious place where program is not compliant with current security standards. Severity of this vulnerability has been evaluated and possible fixes have been suggested.

Keywords application security assessment, password manager, KeePassXC, password storage, encryption

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Počet internetových služeb a jejich uživatelů neustále stoupá. Mnoho informací, které uživatelé na internetu ukládají, přitom obsahuje soukromou komunikaci nebo soukromá data. Přístup k těmto datům je nejčastěji řízen pomocí uživatelských hesel, která jsou poté jedinou bariérou dělící útočníky od těchto dat. Zabezpečení a správa hesel se stává důležitou problematikou při pohybu uživatelů na internetu.

Správu hesel a nakládání s nimi usnadňují aplikace, které nazýváme správci hesel. Správce hesel umožňuje bezpečné uložení a správu všech hesel uživatele. Jednou z těchto aplikací je i aplikace KeePassXC, kterou se tato práce dále zabývá.

Práce má za cíl přiblížit čtenáři problematiku bezpečného ukládání hesel. Výstup práce poskytne přehled současných řešení a pomůže uživatelům programu KeePassXC, kteří chtějí porozumět zabezpečení tohoto programu.

Téma jsem si zvolil, protože správce hesel využívá stále více uživatelů. Vzhledem k povaze dat, která tyto programy spravují, je bezpečnost těchto aplikací důležitým hlediskem. Program KeePassXC jsem pro analýzu zvolil, protože dle oficiální dokumentace nebyl bezpečnostní audit tohoto programu prozatím proveden a používání této aplikace tak může představovat pro uživatele bezpečnostní riziko.

Práce na úvod seznámí uživatele s problematikou bezpečného ukládání hesel a poskytne přehled známých programů pro správu hesel. V kapitole 3 je blíže představena aplikace KeePassXC, které bude analyzovaná, a je v ní popsán postup bezpečnostní analýzy. V kapitole 4 je popsáno uživatelské prostředí programu ve vztahu k bezpečnosti práce s hesly. Záměrem je nalézt místa, která by mohla být použita při potenciálním útoku. Na tato místa se dále zaměřím v kapitole 5, kde je provedena analýza zdrojového kódu programu. Nalezené zranitelnosti jsou dále diskutovány v kapitole 6, kde je zhodnocena jejich závažnost a jsou navržena možná řešení.

Kapitola 1

Cíl práce

Cílem rešeršní části práce je seznámit čtenáře se základní problematikou bezpečné práce s hesly a provést rešerši známých programů pro správu hesel. Úkolem je vysvětlit čtenáři potřebu použití správců hesel a jejich fungování. Dále je cílem popsat typy správců hesel, jejich rozdíly, fungování a implementační detaily s ohledem na zabezpečení dat.

Praktická část práce má za cíl provést analýzu uživatelské rozhraní a zdrojového kódu zvolené aplikace. Při analýze uživatelského prostředí je úkolem se zaměřit na ty části uživatelského prostředí, které mají vliv na bezpečnost aplikace a nalézt potenciální vektory útoku. S ohledem na nalezené útočné vektory je úkolem prostudovat zdrojový kód aplikace pro hlubší posouzení nalezených zranitelností. Bude-li to vhodné, zdrojový kód bude otestován pomocí vhodných testovacích nástrojů. Konečným cílem analýzy je nalezené zranitelnosti zdokumentovat, vyhodnotit jejich závažnost a navrhnout opatření k jejich nápravě.

Kapitola 2

Problematika ukládání hesel a současná řešení

Hesla jsou základním stavebním kamenem pro řízení přístupu k informacím v digitálním světě. Hesla jsou používána k ověření identity, zabezpečení informací a zamezení neoprávněného přístupu k informacím či datům. Služby jako internetová bankovnictví, osobní či pracovní emaily, internetové obchody a komunikátory používají hesla často jako jedinou metodu pro zabránění přístupu neoprávněným osobám. Kvalitní heslo potom komplikuje pokusy útočníků, kteří se snaží ukrást citlivá data nebo peníze.

Studie provedená v roce 2007 zaměřující se na návyky uživatelů při používání hesel [1] uvádí, že průměrný uživatel vlastní přibližně 25 internetových účtů, které vyžadují heslo. Tato studie dále uvádí, že každý den průměrný uživatel zadá na internetu přibližně 8 hesel. Studie provedená v roce 2015 dokonce uvádí, že průměrný uživatel vlastní minimálně 90 internetových účtů [2]. Vzhledem k technologickému postupu a množství lidí aktivně využívající internet toto číslo pravděpodobně dále poroste. Zapamatování takového množství hesel, při dodržení požadované složitosti a unikátnosti pro každou ze služeb, je pro běžného uživatele prakticky nemožné [3].

Řešení této problematiky přináší programy nazývané správci hesel. Jedná se o programy, které dokáží uchovávat velké množství hesel v zašifrované databázi. Popularita správců hesel roste společně s počtem internetových účtů, které uživatelé vlastní. Představují totiž bezpečnostní schránku, která uchovává všechna hesla a uživatelská jména, která do ní byla vložena. Zašifrovanou databázi je poté schopen otevřít pouze ten uživatel, který zná tzv. hlavní heslo. Hlavní heslo se používá k zašifrování celé databáze a v případě prolomení této ochrany získává útočník přístup ke kompletní databázi hesel uživatele. Hlavní heslo je ideálním řešení pro uživatele, kteří mají několik hesel napříč účty, a je také jediným heslem, které je nutné si pamatovat. [4]

Hesla ovšem nejsou jediným mechanizmem pro řízení přístupu a k ověření identity uživatele. Přehled ostatních metod a jejich použití v reálném světě

nabízí následující kapitola.

2.1 Možnosti autentizace uživatele

Autentizace je proces ověření identity uživatele, kterým se zjišťuje, zda je uživatel opravdu tím, za koho se vydává [5]. To se nejčastěji provádí ověřením jedním nebo kombinací více z následujících ověřovacích faktorů:

2.1.1 Znalost

Jedná se o informaci, na kterou si je uživatel schopen vzpomenout. Klasickým příkladem je heslo nebo PIN. Rozšířenou technikou je ovšem i výběr obrázků podle vlastní preference nebo nakreslení gesta či znaku, které je známe především z mobilních telefonů. [5]

2.1.2 Vlastnictví

Tento faktor představuje to, co má uživatel ve svém vlastnictví. To zahrnuje například hardwarové tokeny, platební karty, mobilní telefony (aplikace využívající IMEI), SIM karty (ověření zpětným voláním a pomocí SMS OTP) a v zobecněné podobě i osobní doklady (pas, občanský a řidičský průkaz). [5]

2.1.3 Biometrie

Faktor zahrnuje charakteristiky svého nositele. Může se jednat o otisk prstu, rozšířený jak ve sféře notebooků, tak mobilních telefonů. Další možností je například sken očnice, rozpoznávání obličeje nebo hlasu. [5]

2.1.4 Vícefaktorová autentizace

Vícefaktorovou autentizací je proces, při kterém jsou ověřeny alespoň dva z ověřovacích faktorů. O takový typ autentizace se tedy například jedná, pokud je pro přihlášení nutné zadat heslo (Znalost) a ověřovací kód zaslaný přes SMS (Vlastnictví mobilního telefonu) [5].

Technická doporučení pro implementaci autentizace uživatele poskytuje mimo jiné například americký Národní institut standardů a technologie (NIST), který ve své publikaci SP 800-63B [6] vydává směrnici s technickými požadavky pro implementaci ověřovacích mechanismů u digitálních služeb používaných federálními úřady.

2.2 Potřeba správce hesel

Použití správce hesel snižuje riziko, které představuje používání slabých hesel a opakování stejného hesla pro více služeb. Pokud je stejné heslo užíváno u více

služeb, hrozí při prolomení jednoho z nich zneužití i dalších účtů, u kterých uživatel používá to samé heslo.

Výzkum provedený v roce 2011, který analyzoval více než 14000 hesel použitých na systémech UNIX [3], zjistil, že téměř 25% použitých hesel byla slova, které je možné zároveň nalézt ve slovníku, což umožňuje jejich snadné uhádnutí. Ideální heslo by mělo být snadno zapamatovatelné (použitelnost) a zároveň komplexní (bezpečnost). Bohužel tyto dvě podmínky se vzájemně vylučují.

Evropská agentura pro bezpečnost sítí a informací (ENISA) [7] doporučuje uživatelům pro správnou práci s hesly následující:

- Kombinaci více znaků v hesle (malá písmena, velká písmena, speciální znaky). [7]
- Používání dlouhých hesel. Hesla do délky 9 znaků lze prolomit v řádu sekund. Při použití více než 14 znaků v hesle již nehrozí jeho prolomení pomocí aktuálně dostupné výpočetní techniky. [7]
- Využití unikátního heslo pro každou webovou službu. [7]
- Použití správce hesel pro vygenerování náhodného hesla. [7]
- Pokud je použití náhodného heslo nepraktické, uživatel by měl využít dlouhou frázi. [7]

Mnohé internetové služby již po uživatelích požadují, aby jejich heslo splňovalo určité požadavky, jako je například minimální počet znaků nebo využití speciálních znaků.

Správci hesel byli vytvořeni za účelem usnadnění práce s hesly běžnému uživateli. Hlavní funkcí těchto aplikací je ukládání uživatelských jmen a hesel, díky kterému si uživatel nebude muset pamatovat vysoký počet komplexních hesel. Další přidanou hodnotou správců je funkce předvyplňování webových formulářů a ukládání URL adres. [8]

Používáním správce hesel se snižuje zranitelnost uživatele proti typickým útokům jako je phishing, při kterém útočník podvrhne webovou stránku a vyžádá od uživatele zadání přihlašovacích údajů. Falešná webová stránka ovládaná útočníkem přihlašovací údaje shromažďuje a umožnuje útočníkovi jejich pozdější využití nebo prodej. Správce hesel snižuje riziko phishingu tím, že při navštívení phishingové stránky správce hesel nevyplní heslo nebo nepřihlásí uživatele automaticky, jak by k tomu došlo, kdyby se URL adresa shodovala s tou uloženou v databázi. [4]

Správce hesel rovněž poskytuje ochranu před škodlivými programy odposlouchávajícími stisky kláves na uživatelově zařízení. Při použití správce totiž nedochází k manuálním psaní hesel na klávesnici, ale k automatickému kopírování. Tím dochází ke snížení rizika, že jsou stisky kláves ukládány škodlivým programem. [4]

2.3 Fungování správce hesel

Existuje mnoho variant správců hesel. Některé z nich jsou součástí široce rozšířených webových prohlížečů, jiné existují jako samostatné aplikace a v neposlední řadě existují správci fungující kompletně v online úložišti. Všechny varianty ovšem mají určité společné rysy.

Bezpečnost databáze spočívá v použití silného přihlašovacího mechanismu do databáze. Po úspěšném přihlášení je možné zobrazit všechna uživatelská jména, hesla, URL a další data uložená v databázi. Pro přihlášení by se proto mělo využívat vícefaktorové autentizace. V závislosti na implementaci správce hesel je potom jeden či více autentizačních faktorů využito pro zašifrování samotné databáze. [8]

Nejčastěji je pro zašifrování databáze používáno hlavní heslo. Šifrování je důležitá vlastnost správce hesel a existuje mnoho šifrovacích algoritmů, které jsou využívány s ohledem na to, o jaký typ správce se jedná. Algoritmus a postup šifrování u jednotlivých programů budou představeny v následující sekci.

Správce hesel lze rozdělit do následujících kategorií [9]:

- Lokální správci hesel
- Cloudové služby
- Správci hesel v prohlížečích

Pro bližší prozkoumání fungování jednotlivých správců hesel byli vybráni ti nejrozšířenější v jednotlivých kategoriích. Lokální správce KeePassXC, který je v dalších částech práce blíže zkoumán, byl doplněn programy KeePass a Password Safe, které jsou také multiplatformními a zároveň open-source programy [10]. Za zástupce kategorie cloudových správců hesel byly vybrány služby LastPass, Dashlane a 1Password, které patří dle serveru Everplans k těm nejpopulárnějším [11]. Nejoblíbenější prohlížeče byly vybrány na základě aktuálního podílu na trhu v březnu roku 2018 podle serveru NetMarketShare [12]. Seznam všech později představených správců hesel lze vidět na obrázku 2.1.

2.4 Lokální správci hesel

Kategorie lokálních správců zahrnuje aplikace, které mají grafické rozhraní umožňující uživateli jednoduchou správu jeho účtů a hesel. Zašifrovaná databáze je poté uložena na lokálním disku. Lokální uložení zašifrovaných dat dává uživateli možnost s databází jakkoliv nakládat a přesouvat mezi počítači například na USB disku. Databáze ovšem pro uživatele není tak přístupná, jako kdyby byla uložená na online úložišti. Možnost přesouvat databázi a uchovávat

Tabulka 2.1: Nejrozšířenější správci hesel

Typ	Název
Lokální	KeePassXC
Lokální	KeePass
Lokální	Password Safe
Cloudový	LastPass
Cloudový	Dashlane
Cloudový	1Password
Prohlížeč	Google Chrome
Prohlížeč	Mozilla FireFox
Prohlížeč	Microsoft Internet Explorer

ji dle potřeby může představovat zároveň bezpečností výhodu i bezpečnostní hrozbu [13].

Při lokálním ukládání zašifrovaných dat nemá aplikace důvod přenášet citlivá data po internetu, čímž se snižuje bezpečnostní riziko spojené s možným odposloucháváním přenášených dat. Zároveň nehrozí nebezpečí, že uložená data budou prozrazena při napadení cloudového poskytovatele. Existující bezpečnostní analýzy se zaměřovaly na testování formátu databáze a způsobu, jakým s nimi lokální správci pracují [4]. Tato studie vyvrátila, že by architektura analyzovaných formátů databází představovala bezpečnostní riziko.

Hlavním rysem společným pro všechny lokální správce hesel je možnost stažení instalačního programu nebo stažení zkomprimované složky obsahující všechny soubory potřebné ke spuštění programu na jakémkoliv počítači. To dodává uživateli potřebnou přenositelnost programu, která se hodí, například když není dostupné připojení k internetu. [8]

Další vlastností umožňující přenositelnost je možnost importovaní a exportování databáze. Při exportování databáze se celá nebo jen část uložených dat dešifruje a uloží do souboru v čitelné formě. Nejčastěji je možné data exportovat alespoň ve formátu CSV. To umožňuje importování celé nebo jen části uložených dat například do databáze jiné aplikace.

Důležitou funkcionalitou je generování náhodných hesel pro jednotlivé účty. Použitím generátoru hesla se předchází situaci, kdy uživatel zvolí stejné nebo podobné heslo jako u jiného účtu. Některé programy umožňují také volbu konkrétní délky hesla a typů znaků, které budou v hesle použity.

Automatickým mazáním hesel zkopírovaných do schránky se předchází situacím, kdy uživatel své heslo nechtěně zkopíruje mimo přihlašovací formulář například několik minut po prvotním přihlášení. Aby se těmto situacím předešlo, je správce hesel schopen po několika sekundách po zkopírování hesla z databáze obsah kopírovací schránky smazat.

V následujících podkapitolách budou blíže popsáni jednotliví správci hesel se zaměřením na jejich vlastnosti z hlediska bezpečnosti a šifrování ukládaných

dat.

2.4.1 KeePassXC

KeePassXC je open-source správcem hesel, který vznikl v roce 2017 oddělením od populárního správce hesel KeePassX. Oddělení proběhlo za účelem rychlejšího vývoje aplikace a přidávání nových funkcionalit [8]. Právě přidávání nových funkcionalit často přináší potenciální bezpečnostní mezery ve vyvíjeném softwaru [14]. Vzhledem k charakteru této aplikace je její bezpečnost a bezpečné ukládání hesel zásadní.

Databáze hesel Pro bližší pochopení fungování programu KeePassXC je důležité porozumět formátu KDBX, který tento program používá jako hlavní mechanismus pro ukládání všech zašifrovaných dat. Program podporuje formát databáze KDBX3.1 a KDBX4. [15]

Formát databáze KDBX se skládá ze dvou částí, kterými jsou nezašifrovaná hlavička a zašifrovaná databáze ve formátu XML. Nezašifrovaná hlavička obsahuje informace o souboru a data nutná k dešifrování celé databáze. Prvních 8 bajtů souboru obsahuje tzv. „magic bytes“, což je specifická posloupnost bitů, která napomáhá rozpoznání formátu souboru. Nezašifrovaná hlavička obsahuje dále verzi souboru a další nepovinné položky. Mezi takové položky patří například identifikátor použitého šifrovacího algoritmu, informace, zda jsou šifrovaná data komprimovaná, sůl použitá v algoritmu pro odvození klíče a počet iterací funkce pro odvození klíče. [15]

Hlavní rozdíl mezi verzemi KDBX3.1 a KDBX4 je nový způsob ověřování integrity nezašifrovaných dat v hlavičce. Verze KDBX4 využívá hašovací funkce HMAC-SHA-256. Starší formát databáze KDBX3.1 používal pro ověřování dat v hlavičce hašovací funkci SHA256 a haš uložil v zašifrované části databáze. [15]

Funkce pro odvození klíče (překlad z anglického „key derivation function“) je dalším důležitým mechanizmem pro zabezpečení databáze. Cílem funkce je vytvoření odvozeného klíče (OK) o určité délce. Funkce přitom používá jako parametry heslo, sůl a počet iterací. Jako heslo je použito hlavní heslo do databáze, případně v kombinaci s dalšími ověřovacími faktory pro přístup do databáze. Funkce pro odvození klíčů jsou záměrně pomalé, aby tím znemožnily slovníkové útoky nebo útoky hrubou silou. Rychlost funkce lze ovlivnit počtem iterací, které funkce vykoná. [16]

$$OK = funkce(Heslo, Sul, PocetIteraci)$$

Zašifrovaná část obsahuje všechna data uložená v databázi včetně uživatelských jmen, URL adres nebo poznámek uživatele. Tím se tento formát odlišuje od jiných typů správců hesel, které například šifrují pouze hesla. [17]

Šifrování databáze Šifrování databáze umožňuje program KeePassXC pomocí 3 šifrovacích algoritmů, kterými jsou AES/Rijndael, Twofish a ChaCha20. Všechny algoritmy využívají klíč délky 256 bitů, který je vytvořený funkcí pro odvození klíče. Funkci pro odvození klíče je možné zvolit z možností AES-KDF a Argon2. [8]

2.4.2 KeePass 2.x verze 2.38

KeePass je program s otevřeným zdrojovým kódem, který je vyvíjen od roku 2003 a je dostupný na operačních systémech Windows, Linux a Apple Mac OS. Během let získal mnohá ocenění, která lze nalézt na oficiálním stránkách programu. [15]

Pro KeePass je díky široké uživatelské základně dostupné velké množství pluginů. Jedná se přitom například o pluginy podporující exportování a importování dat v dodatečných formátech, umožňující zálohu databáze nebo podporující další algoritmy pro šifrování databáze. [13]

Jako jediný ze zkoumaných lokálních správců hesel podporuje KeePass přihlášení do databáze pomocí uživatelského účtu na operačním systému Windows. Databázi je poté možné otevřít, pouze pokud je otevírána pod uživatelským účtem, pod kterým byla vytvořena. Uživatelům tato funkce usnadňuje práci, protože nemusí zadávat heslo pro odemčení databáze hesel, pokud se již přihlásil do svého Windows účtu. Pokud ovšem dojde ke smazání uživatelského účtu na operačním systému, je přístup k databázi ztracen a nestačí vytvořit uživatelský účet se stejným jménem a heslem. [15]

Šifrování databáze KeePass verze 2.x provádí šifrování databáze pomocí algoritmu AES/Rijndael s délkou klíče 256 bitů a použitím módu CBC. Klíč pro šifrování je vytvořen hašovacím algoritmem SHA-256, který klíč vytvoří použitím jednoho či více následujících faktorů: hlavní heslo databáze, soubor s klíčem, klíč pluginu nebo klíč uživatelského účtu Windows. [15]

KeePass používá další mechanizmy pro zlepšení bezpečnosti. Jedná se například o funkci pro odvození klíče AES-KDF. Funkce AES-KDF přijímá parametry, kterými jsou heslo, sůl a počet iterací. Výstupem této funkce je odvozený klíč, který je použit pro zašifrování databáze. Ochrana před slovníkovými útoky a útoky hrubou silou spočívá v počtu iterací, které musí funkce provést, čímž je ovlivněna doba provádění této funkce. Hádání velkého množství možných klíčů se tím stává pomalejší a neefektivní. [16]

2.4.3 Password Safe

Password Safe je volně dostupným správcem hesel s otevřeným kódem. Projekt Password Safe byl založen bezpečnostním expertem Brucem Schneierem v roce 2002. Program se snaží poskytovat všechny potřebné funkce při zachování nízké velikosti výsledného programu. Stejně jako ostatní podobné pro-

gramy i Password Safe umožňuje importování a exportování dat, generování náhodných hesel a mazání kopírovací schránky. [18]

Hlavní odlišností od programů KeePass a KeePassXC je použití jiného typu databáze pro ukládání hesel. Password Safe využívá formátu V3 [18], který byl blíže analyzován ve studii zkoumající různé formáty databází [4] a označen za bezpečný.

Šifrování databáze program Password Safe provádí pomocí algoritmu Twofish s délkou klíče 256 bitů. Výsledný klíč se vytváří pomocí algoritmu PBKDF2 a hašovací funkce SHA-256 s nastavitelným množstvím iterací. [18]

2.5 Cloudové služby

Hlavním znakem tohoto typu správce hesel je centrální ukládání databáze hesel v online úložišti na serveru poskytovatele služby. Existuje několik možností, jak přistoupit k uživatelově databázi [9]:

- Lokální aplikace připojující se na server poskytovatele. Může se jednat o desktopovou nebo mobilní aplikaci.
- Webová aplikace
- Rozšíření do prohlížeče

Tento způsob ukládání hesel umožňuje uživatelům mít přístup k databázi hesel z více zařízení při zajištění neustále synchronizace databáze hesel. Díky této vlastnosti jsou cloudoví poskytovatelé správců hesel stále více vyhledávaní uživatelskou komunitou [9]. Stejně jako ostatní aplikace umožňují i cloudové služby generování nových hesel, což usnadňuje generaci komplexních a unikátních hesel pro různé internetové služby.

Cloudoví poskytovatelé často nabízejí více variant správců. Jedná se často o lokální aplikace, rozšíření prohlížečů ale i samotné webové aplikace. Všechny varianty poté využívají tu stejnou databázi uloženou v cloudovém úložišti. Často používaná jsou již zmiňovaná rozšíření do prohlížečů. Ta totiž umožňují automatické rozpoznání, kdy uživatel navštíví určitou službu a automaticky vyplní přihlašovací formulář nebo uživatele na dané stránce přímo automaticky přihlásí. [9]

Šifrování dat je hlavní prioritou správců hesel a různé služby řeší šifrování dat jiným způsobem. Může se například lišit, zda program šifruje pouze hesla nebo všechna data v databázi. Například LastPass šifruje uživatelská jména i hesla [19]. Naopak správce hesel PasswordBox šifruje pouze hesla k jednotlivým účtům [17].

2.5.1 LastPass

LastPass pro šifrování databáze využívá algoritmu AES s délkou klíče 256 bitů. Pro odvození finálního klíče používá program LastPass funkci PBKDF2 a hašovací funkci SHA-256. Databáze je vždy šifrována na koncovém zařízení. Díky tomu jsou vždy přenášena data v zašifrovaném formátu a nehrozí odposlechnutí dat v otevřeném textu. V minulosti byl LastPass postižen několika bezpečnostními incidenty, kdy bezpečnostní experti zjistili analomální tok dat ze sítě LastPass indikující možný únik dat, který se ovšem nepodařilo potvrdit. V roce 2017 tým bezpečnostních analytiků společnosti Google zjistil bezpečnostní trhlinu v rozšíření do prohlížeče od společnosti LastPass. Společnost chybu během několika dní opravila a na chybu uživatele upozornila. [20]

2.5.2 Dashlane

Dashlane stejně jako LastPass nabízí uživatelům základní funkce, kterými jsou správa hesel, automatické vyplňování formulářů nebo generování náhodných hesel. Pokročilejší funcionalitou tohoto správce hesel je možnost automatické změny hesla u nejznámějších online účtů, jakými jsou například Facebook, Twitter nebo Dropbox. [21]

Pro šifrování databáze program Dashlane využívá algoritmus AES s délkou klíče 256 bitů a funkci pro odvození klíče PBKDF2. Dashlane v prosinci roku 2017 vydal detailní publikaci popisující zabezpečení jednotivých komponent služby. [22]

2.5.3 1Password

1Password prezentuje svoji aplikaci jako nejbezpečnějšího správce hesel. Aplikace umožňuje správu hesel a dalších citlivých informací, kterými mohou být například licenční klíč nebo údaje o kreditních kartách. Aplikace se sice prezentuje jako nejbezpečnější správce hesel, ale z hlediska šifrování používá stejný šifrovací algoritmus jako aplikace LastPass a Dashlane. Pro šifrování databáze využívá algoritmu AES s délkou klíče 256 bitů. Pro odvození finálního klíče používá program 1Password funkci PBKDF2 a hašovací funkci SHA-256. Ze všech tří zmíněných cloudových služeb nabízí 1Password nejlepší technickou dokumentaci a popis zabezpečení programu. [23]

2.6 Správci hesel v prohlížečích

Většina nejběžněji používaných prohlížečů jako je Google Chrome, Mozilla FireFox, Microsoft Internet Explorer, Safari nebo Opera používají správu hesel přímo zabudovanou v programu. Každý prohlížeč má určitý způsob ukládání hesel, který je závislý na operačním systému, na kterém je program spuštěn.

Správci hesel v prohlížečích mají omezezené funkce oproti ostatním typům. Například prohlížeče nenabízejí komplexní generátory nových hesel. [24]

Na operačním systému Windows prohlížeče Google Chrome a Microsoft Internet Explorer využívají k šifrování „Windows Data Protection API“. Pro šifrování je u „Windows Data Protection API“ používáno heslo k uživatelově Windows účtu. Díky tomu nejsou data čitelná v otevřeném textu a přitom si uživatel nemusí pamatovat hlavní heslo k databázi. [24]

Prohlížeč Mozilla FireFox má správce hesel zabudovaný přímo v prohlížeči. Uživatel si může zvolit vlastní hlavní heslo databáze a pro šifrování lokálně uložených dat prohlížeč využívá šifrovací algoritmus 3DES. Pokud uživatel hlavní heslo nezvolí, jsou jeho hesla stále šifrovaná, ale klíč k rozšifrování je dostupný v lokálním souboru *key4.db*. Jako funkci pro odvození klíče prohlížeč používá hašovací funkci SHA-1, která ale již není považována za bezpečnou. [25]

Na zařízeních s operačním systémem MAC OSX prohlížeče spoléhají na šifrování pomocí „Keychain service“. Stejně jako u systému Windows je pro šifrování použito heslo k uživatelovu účtu do systému. [24]

Kapitola 3

Postup bezpečnostní analýzy

Bezpečnostní analýza programu KeePassXC je rozdělena na dvě na sebe navazující části. V první části bude provedena analýza uživatelského prostředí aplikace. Budou zkontrolována výchozí nastavení programu z hlediska bezpečnosti, možnosti autentizace a postupu vytvoření nové databáze. Snahou bude nalézt taková místa v programu, která odporují základním bezpečnostním principům nebo dokonce představují bezpečnostní riziko pro běžného uživatele. Bližšímu zkoumání budou podrobeny i funkce importování a exportování dat, které manipulují s daty v nezašifrované podobě. Všechna podezřelá místa nalezená během analýzy uživatelského prostředí budou označena kódem Z1-Zx, aby je později bylo možné zkontrolovat přímo ve zdrojovém kódu programu.

V druhé části bezpečnostní analýzy budou podrobena hlubší analýze podezřelá místa nalezená během analýzy uživatelského prostředí a bude rozhodnuto, zda představují bezpečnostní riziko pro uživatele. Pro analýzu zdrojového kódu bude využito i automatizovaných nástrojů, kterými provedeme statickou analýzu zdrojového kódu. Vzhledem k tomu, že KeePassXC dle oficiální dokumentace využívá několik externích knihoven, bude prozkoumáno které knihovny jsou skutečně využívány a zda použití některých z nich nepředstavuje bezpečnostní riziko. V poslední části budou zkoumány přímo používané šifrovací funkce a jejich implementace ve zdrojovém kódu. Nalezené zranitelnosti budou označeny kódy V1-Vn, aby v další části práce bylo možné posoudit jejich závažnost a mohla být navržena bezpečnostní opatření.

3.1 Projekt KeePassXC

Pro bezpečnostní analýzu byla zvolena nejnovější dostupná verze programu, kterou byla verze 2.3.0. Zdrojový kód aplikace je napsaný v jazyce C++ a pro grafické rozhraní je použita knihovna Qt5. Součástí programu je také větší množství externích knihoven, které jsou dále zkoumány v jedná z nadcházejících kapitol.

Pro systém Windows nabízí KeePassXC přenositelnou verzi programu a instalační balíček. Při stažení přenositelné verze jsou všechny potřebné soubory pro spuštění programy součástí zkomprimovaného souboru. Program lze poté kdykoliv spustit třeba z přenositelného média bez nutnosti instalace. Instalační balíček po spuštění rozbalí všechny potřebné soubory na lokálním počítači a vytvoří potřebné záznamy v registrech.

3.2 Bezpečnostní princip SD^3

Analýza se bude řídit bezpečnostními principy označovaným SD^3. Označení SD^3 pochází z anglických názvů jednotlivých principů, kterými jsou *Secure by Design*, *Secure by Default* a *Secure in Deployment* neboli zabezpečení při vývoji, zabezpečení při výchozím nastavení a zabezpečení při použití. Jedná se o množinu strategií, které pomáhají při dosahování krátkodobých a dlouhodobých cílů v bezpečnosti. [14]

Secure by Desing – Zabezpečení při vývoji Tato strategie popisuje princip, při kterém jsou již v průběhu vývoje podniknuty odpovídající kroky pro bezpečný celkový návrh produktu. Může se jednat například o naplánování běhu aplikace s minimálními oprávněními, dodržování programovacích konvencí, vyřazení nepoužívaného kódu a funkcí, pravidelné penetrační testování produktu nebo naplánování periodických bezpečnostních školení pro vývojáře. [14]

Secure by Default – Zabezpečení při výchozím nastavení Strategie vystihuje princip, při kterém je aplikace už ve svém výchozím nastavení dostatečně bezpečná. Toho lze dosáhnout například instalací jen těch služeb a funkcí, které jsou skutečně využívány, nebo vzdáním se oprávnění, která nejsou vyžadována. [14]

Secure in Deployment – Zabezpečení při použití Tento princip znamená, že je produkt i po své instalaci snadno udržovatelný a tím umožňuje v budoucnu změny, které neohrozí jeho bezpečnost. K produktu by měla být dostupná dostatečná dokumentace a uživatelům by měly být na obrazovce poskytnuty tipy a varovná hlášení. [14]

3.3 Doporučení pro implementaci autentizace

Během bezpečnostní analýzy bude porovnáno, zda program KeePassXC splňuje směrnici SP 800-63B institutu NIST [6] pro programy implementující autentizaci uživatelů. Pro jednotlivé typy ověřování bude zkoumáno, zda program KeePassXC splňuje normativní požadavky stanovené tímto úřadem.

3.3.1 Zapamatovatelné heslo

Do této kategorie v angličtině nazvané „Memorized Secrets“ patří ověřovací faktory, kterými jsou například heslo a nebo PIN. Takové heslo musí mít dostatečnou složitost a délku, aby bylo pro útočníka obtížné ho uhádnout, ale které zároveň musí být zapomatovatelné pro uživatele. [6]

Konkrétně klade norma při použití hesla několik požadavků. Mezi ty nejdůležitejší patří požadavek na délku hesla, které by mělo mít alespoň 8 znaků pro hesla zvolená uživatelem a 6 znaků pokud je heslo náhodně vygenerované. Není přitom kladen žádný požadavek na složení znaků hesla a může se například jednat o heslo složené čistě z čísel. Pokud se přitom heslo objevuje na seznamu známých hesel nebo na seznamu hesel uniklých z jiných služeb, program by po uživateli měl požadovat zadání hesla jiného. [6]

Mezi známá hesla se řadí hesla uniklá při prolomení databází. To se v minulosti přihodilo například u společností Yahoo nebo Equifax [26]. Dále mezi známá hesla řadíme slovníková slova jakéhokoliv jazyka a opakující se znaky nebo posloupné sekvence. Mezi sekvence řadíme například „aaaaaa“ nebo „abcd12345“. V hesle by také neměl být použit název služby, u které je heslo použito, případně heslo úzce spojené s touto službou. [6]

Dle normy SP 800-63B by měl program při vytváření hesla uživateli poskytnout informaci o síle hesla, aby uživatel zvolil heslo dostatečně silné. Uživatel by měl mít zároveň možnost si heslo na omezenou dobu zobrazit v neskryté podobě. [6]

Při ukládání by mělo být heslo uloženo tak, aby bylo odolné vůči útokům hrubou silou. Služba by měla zajistit, aby byl ukládán pouze haš hesla odvozený pomocí funkce pro odvození hesla za využití soli. Mezi vhodné hašovací funkce přitom norma řadí funkce *PBKDF2* a *Balloon*.

3.3.2 Sdílené tajemství

Jedná o typ zabezpečení, při kterém je pro ověření využito dat, ke kterým má přístup pouze uživatel a ověřovací autorita. Jedná o fyzický nebo elektronický typ dat, který je použit při ověření identity uživatele. Není přitom nutné, aby si uživatel tato data pamatoval. Nejčastěji je v této kategorii používán obnovovací klíč, který by pro normálního uživatele nebylo možné si zapomatovat, ale uživatel k němu má přístup a může vlastnictvím tohoto záznamu ověřit svoji identitu. [6]

Při vytváření sdíleného tajemství by měl být využit schválený generátor náhodných bitů. Vygenerované tajemství by mělo mít entropii alespoň 20 bitů a při přenosu od uživatele k ověřovací autoritě musí být přenos dat zabezpečen. Pokud má přitom tajemství entropii menší než 64 bitů, musí být implementována ochrana před útoky hrubou silou. Taková ochrana by neměla celkem povolit více než 100 neúspěšných pokusů o ověření, nebo by po určitém počtu neúspěšných pokusů měla po určitou dobu ověřování zabránit. [6]

Přesné požadavky pro správnou implementaci náhodného generátoru NIST specifikuje v publikaci SP 800-90A [27] a SP 800-90B [28]. Jedná se o rozsáhlou publikaci popisující požadavky pro implementaci deterministického generátoru pseudonáhodných bitů. Pro běh deterministického generátoru je zásadní jeho vstup, kterým je inicializační hodnota, v angličtině známá jako „random seed“. Při použití stejné inicializační hodnoty vyprodukuje generátor vždy stejný výstup. Nepředvídatelnost této incializační hodnoty je proto důležitým faktorem.

3.3.3 Jednofaktorová kryptografická zařízení

Hardwarová zařízení umožňující autentizaci pomocí kryptografických operací za použití symetrických nebo asymetrických klíčů označujeme za jednofaktorová kryptografické zařízení neboli token. Tato zařízení je možné připojit do koncového zařízení (nejčastěji pomocí USB slotu). Po odeslání požadavku do zařízení se tento požadavek pomocí symetrického nebo asymetrického klíče podepíše a odpoveď odešle zpět do koncového zařízení. Koncové zařízení je poté schopné ověřit, zda je odpoveď podepsaná konkrétním kryptografickým tokenem. Ověření tohoto typu představuje autentikační faktor vlastnictví. [6]

Dle normy nesmí být možné data uložená na kryptografickém zařízení exportovat, protože by tím došlo k prozrazení klíče. Délka klíče uloženého na zařízení by měla být alespoň 112 bitů a požadavek odeslaný do zařízení k ověření by měl mít délku alespoň 64 bitů. Kryptografické zařízení by mělo po uživateli vyžadovat nějakou akci k tomu, aby zpracovalo požadavek. Nejčastěji se jedná o nutnost stisknutí tlačítka na zařízení. Tím je možné předejít nechtěnému ověření požadavků přicházejících například z nakažených koncových počítačů. [6]

3.4 Testovací prostředí

Pro analýzu byla zvolena verze programu KeePassXC pro operační systém Windows. Operační systém Windows 7 je dle statistik NetMarketShare [29] z února roku 2018 stále nejrozšířenější systém s podílem na trhu přes 44%. Právě proto byla tato verze programu podrobena bezpečnostní analýze, aby měla bezpečnostní přínos pro co největší množství uživatelů.

Veškeré testování programu probíhalo na virtuálním stroji s nainstalovanou verzí operačního systému Windows 7 Enterprise. Virtuálnímu stroji byla alokována 2 jádra procesoru Intel Core i5-4310M s frekvencí 2.7 GHz a velikost RAM paměti 2 GB. Výkon testovacího prostředí je relevantní především pro posouzení dostatečného počtu iterací prováděných derivačními funkcemi při odvození klíče pro šifrování dat.

Pro různé fáze testování byly použity jiné varianty programu. Použití přenositelné verze programu umožňuje jednoduchý návrat do výchozího stavu

a proto byla tato verze použita například při zkoumání změny chování programu při změně konfigurace.

3.5 Použité nástroje

Pro analýzu uživatelského rozhraní a zdrojového kódu programu bylo zvoleno několik nástrojů, které umožňují automatickou detekci možných míst obsahujících zranitelnosti. Výhodou využití takovýchto nástrojů je především jejich rychlost a možnost analýzy velkého množství kódu v krátkém čase. Nástroje pro automatickou analýzu umožňují detekci míst zranitelných pro typy útoku jako je přetečení bufferu nebo SQL Injection. Výstup programu přitom přesně určí zranitelné části kódu a je možné se na ně dále zaměřit. [30]

Některá zranitelná místa ovšem není možné nalézt pomocí automatizovaných nástrojů. Jedná se především o zranitelnosti při procesu autentizace, chyby v řízení přístupu pomocí access listů nebo nebezpečné použití šifrovacích funkcí. Nevýhodou je dále vysoká míra falešných detekcí a nemožnost nalezení chyb v konfiguraci, pokud se nachází v externím konfiguračním souboru.[30]

3.5.1 Cppcheck

Cppcheck je automatickým nástrojem pro statickou analýzu zdrojového kódu v programovacích jazycích C a C++. Cílem tohoto nástroje je nalezení chyb ve zdrojovém kódu, které kompilátor není schopen odhalit. Cppcheck je schopen ve zdrojovém kódu odhalit nedefinované chování programu jako je například použití neinicializovaných proměnných, použití indexů mimo alokované rozhraní, použití potenciálně nebezpečných funkcí nebo špatná správa paměti. Cppcheck je schopen odhalit takové chyby ve zdrojovém kódu, které mohou vést například k přetečení bufferu. [31]

K analýze byla použita nejnovější dostupná verze programu 1.83. Cppcheck ve svém výstupu používá několik úrovní zpráv v závislosti na závažnosti nalezené chyby [31]:

Error Chyby v kódu, které mohou představovat bezpečnostní riziko

Warning Doporučení pro předcházení chybám v kódu

Style Upozornění na stylistické nedostatky (nepoužité funkce, nepoužité proměnné, duplicitní kód)

Portability Varování před nekompatibilitou mezi 32-bit a 64-bit kódem

Performance Doporučení pro zrychlení kódu

Information Problémy konfigurace

3.5.2 Sysinternals Suite

Sysinternals Suite je balíčkem nástrojů vyvinutých skupinou Sysinternals pracující pod společností Windows. Balíček nabízí prostředky pro široký záběr možných úkonů při správě počítače a operačního systému. [32]

Z balíčku byl využit nástroj Process Monitor, který se specializuje na zjištění podrobných informací o právě běžících procesech a všech důležitých činnostech, které tyto procesy vykonávají. Je pomocí něho možné odhalit nejen standardní informace, ale například také to, které klíče registru jsou měněny, jaké operace jednotlivé procesy provádějí se soubory apod. Utilita umožňuje pokročilé filtrování výpisu, aby bylo možné zobrazit pouze specifické události. [32]

3.5.3 Dependency Walker

Dependency Walker je bezplatný nástroj, který dokáže oskenovat 32-bit nebo 64-bit Windows modul jakými jsou EXE, DLL, OCX, SCR soubory, a zobrazit všechny moduly, na nichž tento modul závisí. U každé nalezené závislosti aplikace zobrazí všechny funkce, které jsou exportované daným modulem a dokáže zobrazit i ty aktuálně používané. Zobrazit je také možné detailní informace o jednotlivých souborech, jako je úplná cesta k souboru, jeho verze, debugovací informace a další. [33]

3.5.4 WireShark

Aplikace Wireshark je protokolový analyzér a paketový sniffer. Mezi jeho nejčastější použití patří analýza a ladění problémů v počítačových sítích, vývoj softwaru, vývoj komunikačních protokolů a studium síťové komunikace. Pomocí programu WireShark je možné zjistit, jaký síťový provoz je generován koncovým zařízením a jaká data tento provoz obsahuje. [34]

Kapitola 4

Analýza uživatelského prostředí aplikace

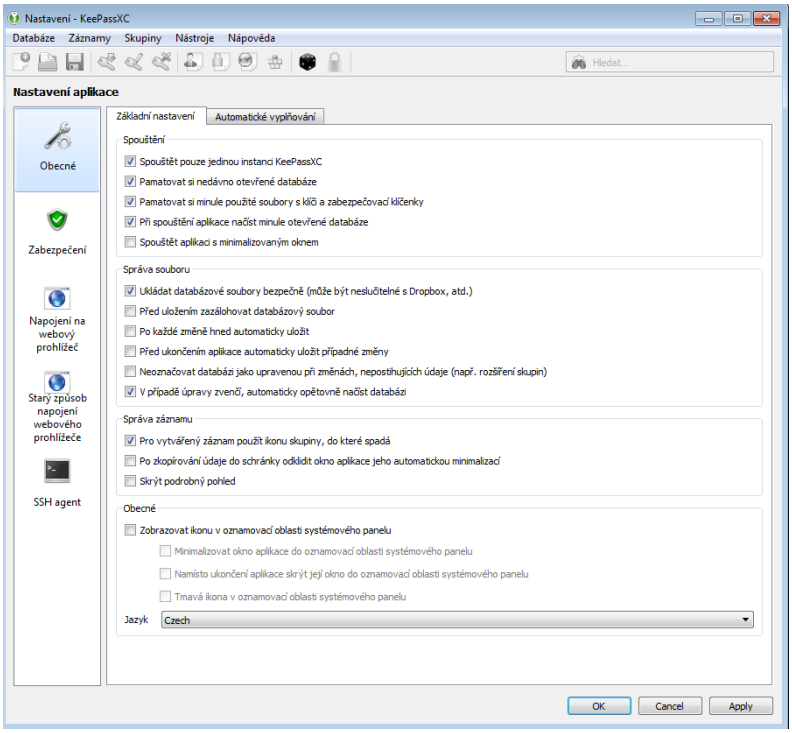
Při použití správce hesel uživatel očekává stoprocentní zabezpečení svých hesel za jakýchkoliv okolností. Při použití lokálního správce hesel by nemělo docházet k přenosu hesel po síti. Největší riziko z hlediska bezpečnosti představuje neautorizovaný přístup k databázi a uložení nezašifrovaných nebo nedostatečně zašifrovaných hesel na lokálním disku. Následující kapitola se tedy zaměřuje na uživatelské prostředí aplikace a jeho potenciální zranitelnosti.

4.1 Výchozí hodnoty aplikace

Většina uživatelů pracující se správci hesel nemá hlubší znalosti o šifrování hesel a spoléhá tedy na výchozí hodnoty zvoleného programu. Nevhodně vybrané hodnoty mohou potom představovat bezpečností riziko pro uživatele. V této kapitole budou prověřeny výchozí hodnoty aplikace při zakládání nové databáze hesel a výchozí nastavení použité při zašifrování databáze.

Pro analýzu výchozích hodnot aplikace byla použita přenositelná verze programu. Po prvním spuštění program nerozpoznal databázi, která by již byla otevřena při předchozím spuštění, a proto program nabídl možnost vytvoření nové databáze, otevření stávající, importování databáze KeePass verze 1 nebo importování dat z CSV souboru.

Před samotným vytvořením první databáze hesel bylo nejdříve prozkoumáno globální nastavení samotného programu. Základní nastavení programu se nachází pod záložkou „Obecné“, jak je patrné z obrázku 4.1. Obecné nastavení umožňuje měnit konfiguraci, která souvisí například se zobrazením grafického rozhraní po spuštění, možnost spuštění více instancí programu nebo volbu jazyka. Z bezpečnostního hlediska je zajímavá především sekce, ve které lze nastavit *Správu souboru*, kterým je v tomto případě zašifrovaný soubor obsahující všechna hesla.

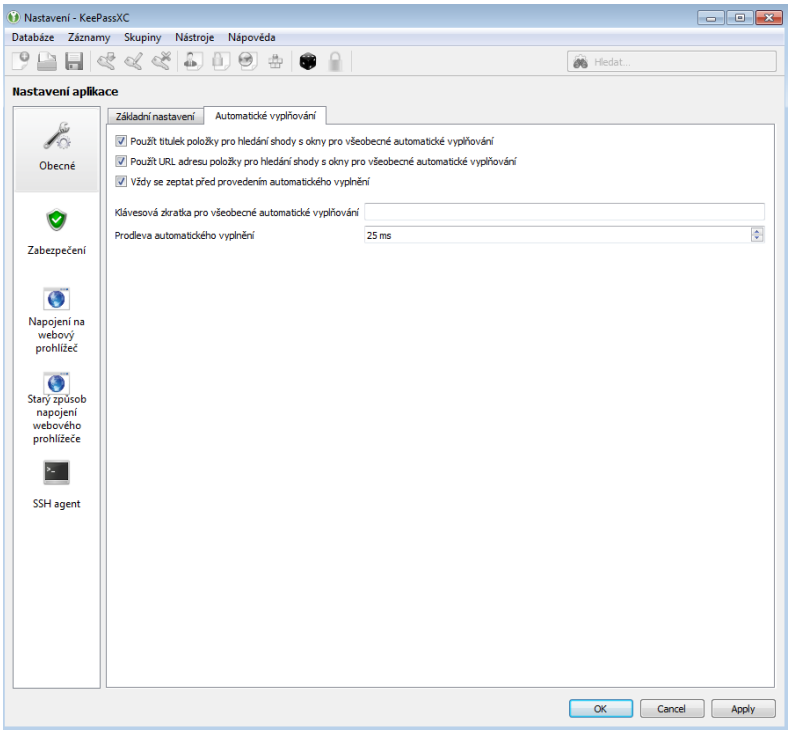


Obrázek 4.1: Výchozí nastavení – obecné nastavení

Jednou z konfiguračních možností v sekci *Správa souboru* je i možnost „Ukládat databázové soubory bezpečně (může být neslučitelné s Dropboxem, atd.)“. V oficiální dokumentaci programu není tato konfigurační možnost blíže popsána. Uživatel programu předpokládá, že by měla být databáze ukládána bezpečně vždy. Ve výchozím nastavení je funkce povolena a data by měla být v bezpečí, ale vzhledem k nejasnému významu této konfigurační možnosti a jejímu fungování byla označena za podezřelou s označením **Z1** a bude podrobena bližšímu zkoumání při analýze zdrojového kódu.

Z pohledu bezpečnosti je nejdůležitější položkou nastavení záložka *Zabezpečení*, která je znázorněna na obrázku 4.2. Z nabízených konfiguračních možností vzbuzuje otázky položka „Zamknout databáze, když je zamčena relace nebo je zavřeno víko notebooku“, která je ve výchozím nastavení zapnutá, ale oficiální dokumentace nevysvětluje, které signály operačního systému zamknutí databáze spouští. Chování této funkce bude blíže prozkoumáno ve zdrojovém kódu pod označením **Z2**.

Pro testování dalších výchozích hodnot je nutné vytvořit testovací databázi a zkontrolovat její nastavení. které je zachyceno na obrázku 4.3. Konfigurace databáze umožňuje nastavení použité šifrovací funkce, funkce pro odvození klíče a počet iterací šifrovacího algoritmu. Počet iterací šifrovacího algoritmu je ve výchozím nastavení stanoven na 1 milion. Posouzení správnosti této



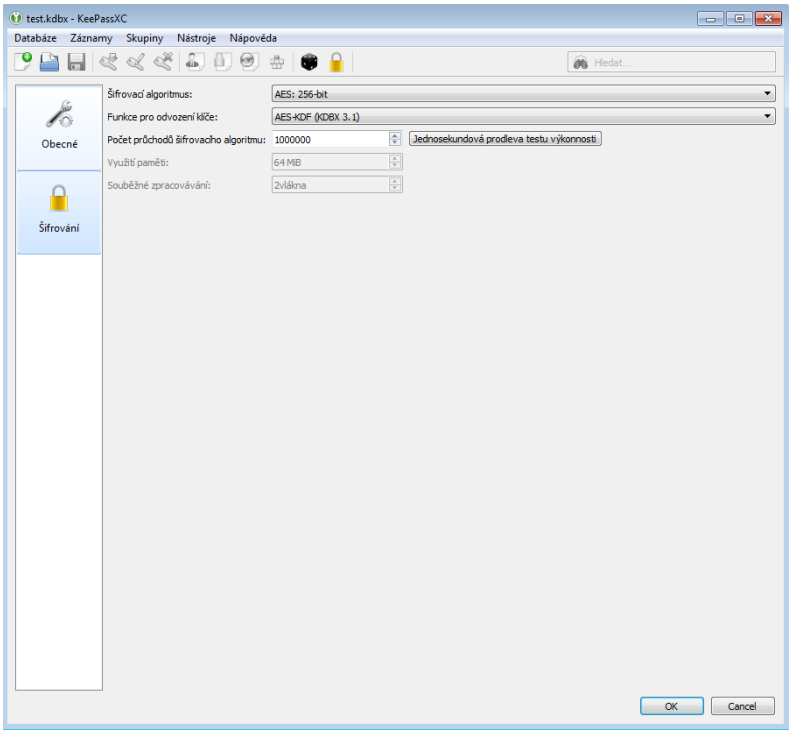
Obrázek 4.2: Výchozí nastavení – zabezpečení

hodnoty je velice individuální vzhledem k tomu, že je závislá na výkonnosti systému, na kterém je program spuštěn.

Program umožňuje provedení testu výkonnosti systému, který zjistí, kolik průchodů je možné na testovaném stroji provést za jednu sekundu. V testovacím prostředí se tato hodnota pohybovala v rozmezí 10-20 milionů v závislosti na momentovém zatížení systému. Dle správce úloh byla při testu využita obě jádra testovaného systému. Ze zjištěných hodnot vyplývá, že by na testovaném stroji mělo být možné se pokusit uhodnout 10-20 hesel za sekundu. Při průměrné rychlosti 15 pokusů za sekundu by trvalo přibližně půl roku uhodnout všechny možné kombinace 6 znakového hesla obsahující malá písmena anglické abecedy. Pokud by byla uvažována pouze slova obsažená v anglickém slovníku, kterých je zhruba 171 tisíc [35], pak by uhádnutí všech možných variant trvalo přibližně 3 hodiny.

Testování ovšem probíhalo na virtuálním stroji, které většinou poskytují nižší výkon než běžné počítače, na kterých by byly výpočty rychlejší. Pokud bylo k prolamování použito speciálně upraveného hardwaru, byly by výsledky řádově odlišné.

Pokud by tedy heslo splňovalo požadavky NIST [6] a mělo alespoň 6 náhodných znaků, lze považovat množství průchodů šifrovacím algoritmem za dostatečné. Při použití slovníkového slova jako hesla by jeho uhodnutí netrvalo



Obrázek 4.3: Výchozí nastavení – databáze

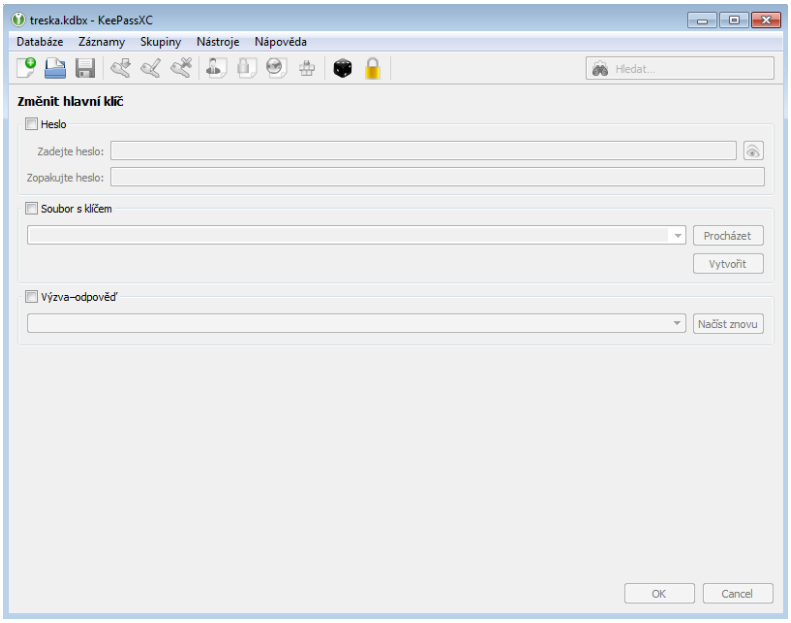
příliš dlouho.

Poslední zkoumaným nastavením programu byla výchozí konfigurace při vytváření nového záznamu do databáze hesel a nastavení generátoru náhodných hesel. Při vytváření nového záznamu do databáze nebyla nalezena žádná konfigurační možnost, která by měla vztah k zabezpečení přidávaného záznamu. Generátor náhodných hesel ovšem je důležitou součásti programu a jeho výchozí konfigurace by měla odpovídat aktuálním doporučením.

Jako výchozí hodnotu volí generátor náhodných hesel délku hesla o 16 znacích. Mezi vybírané znaky jsou ve výchozím nastavení voleny velká a malá písmena anglické abecedy a číslice 0-9. Generátor zároveň umožnuje zvolit kategorie speciálních znaků a znaky z rozšířené sady ASCII. Speciální znaky ve výchozím nastavení nejsou vybrány. Délka hesla i typy znaků vybrané ve výchozím nastavení ale odpovídají požadavkům NIST [6].

4.2 Vytvoření nové databáze a možnosti autentizace

Aplikace nabízí tři varianty autentizace pro otevření zašifrované databáze hesel, jak lze vidět na obrázku 4.4.



Obrázek 4.4: Okno pro vytvoření databáze

Heslo je základním autentizačním nástrojem a pro většinu uživatelů bude také jedinou použitou ověřovací metodou. Program KeePassXC při vytváření nové databáze volí tuto možnost jako výchozí.

Soubor s klíčem představuje autentizační faktor *Vlastnictví*, kdy uživatel poskytne cestu k souboru, který slouží jako součást klíče pro rozšifrováni databáze.

Výzva-odpověď je způsob ověření, který implementuje využití kryptografického tokenu YubiKey. Podpora tohoto způsobu ověřování byla do programu přidána až ve verzi 2.2.0, která byla zveřejněna v červnu roku 2017 [8]. Kryptografický token je zařízení podobné flashdisku, které lze zasunout do USB slotu a po stisknutí tlačítka na tokenu je provedena „akce“. Tento typ ověření využívá princip symetrické kryptografie, kdy se pomocí tajného klíče zašifruje výzva odeslaná do tokenu a koncové zařízení použije odpoveď přijatou z tokenu k zašifrování a dešifrování databáze.

Při vytvoření nové databáze je možné vybrat jakoukoliv kombinaci z autentizačních metod. Je přitom nutné vybrat alespoň jednu z variant, jinak aplikace neumožňuje novou databázi vytvořit. Pro zhodnocení bezpečnosti při vytváření nové databáze budou využita doporučení institutu NIST pro implementaci ověřování a bude porovnáno, zda program KeePassXC tato doporučení nějak zohledňuje.

Pro otestování autentizace pomocí hesla byla vytvořena nová databáze za použití několika hesel různých délek a se složením různých typů znaků. Byla použita hesla krátká, dlouhá, obsahující pouze speciální znaky, nejpoužívanější slovníková slova jako „heslo“ nebo „password“, pouze číslice, pouze bílé znaky jako například mezera nebo tabulátor.

Program vygeneroval varování pro jediný vstup, kterým bylo zadání prázdného řetězce. Program ale po ujištění, že opravdu chceme použít prázdný jako heslo, databázi vytvořil. Pro žádný jiný vstup nebylo vygenerováno varování a nebo chybová hláška a databáze byla úspěšně vytvořena. Při použití velmi dlouhého hesla (v řádu stovek tisíc znaků) nebylo opět vygenerováno žádné varování, ale při zobrazení hesla bylo viditelných pouze prvních 32851 znaků a další znaky nešly přidat. Toto omezení může být způsobené grafickým rozhraním, který program KeePassXC využívá, nicméně fakt vzbuzuje otázku, zda opravdu bylo pro šifrování použito celé původně vložené heslo nebo jen prvních 32851 znaků.

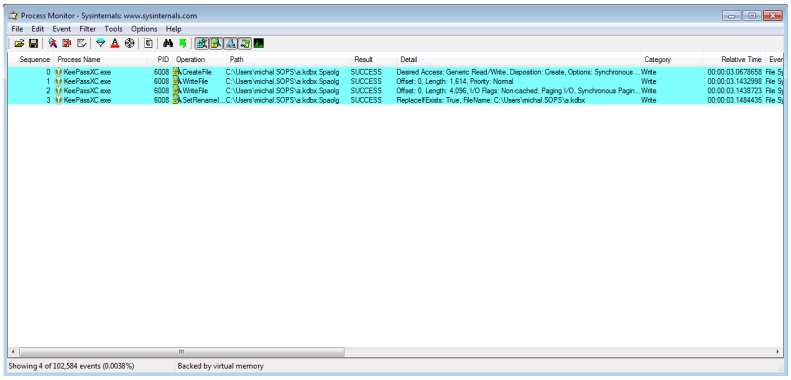
Zjištění, že program nevyžaduje po uživateli zvolení hesla o délce alespoň 8 znaků a nezobrazuje entropii zvoleného hesla, je v rozporu se směrnicí institutu NIST [6]. Dále fakt, že textové pole ignorovalo část vstupu (i když nelogicky dlouhého) bez varování je proti běžné programátorské praxi. Z tohoto důvodu bude prozkoumáno přesné chování programu při vytváření databáze ve zdrojovém kódu pod označením **Z3**.

V další části analýzy bylo zkoumáno chování programu při vytváření nové databáze při použití souboru s klíčem. Program byl otestován použitím prázdného souboru a extrémně dlouhého souboru. Program zároveň umožňoval vygenerování nového klíče. Po zvolení cesty k souboru s klíčem byl vytvořen soubor o velikosti 128 bajtů.

Při použití souboru s nulovou velikostí program vygeneroval chybové hlášení a upozornil nás, že soubor nelze použít jako soubor s klíčem. Ostatní soubory o velikosti 1B až 5GB byly programem přijaty a databáze byla úspěšně vytvořena.

Stejně jako při zadání slabého hesla program nevygeneroval žádné varování. Při použití malého souboru s nízkou entropií nezobrazil, jakou entropii soubor má a zda je vhodný pro šifrování databáze. Soubor s klíčem, který byl vygenerován, má velikost 128 bytů a dle směrnic NIST by tedy měl při vygenerování správným generátorem mít dostatečnou entropii. Právě generování náhodných bitů je ovšem kritické a bude dále zkoumáno ve zdrojovém kódu pod označením **Z4**.

Poslední ověřovací faktor s názvem *Výzva-odpověď* umožňuje použití kryptografického tokenu YubiKey a bez dostupného tokenu nebylo možné jeho použití v uživatelském rozhraní otestovat. Ověření fungování tohoto faktoru bude ověřeno tedy alespoň ve zdrojovém kódu pod označením **Z5**.



Obrázek 4.5: Process Monitor při sloučení databází

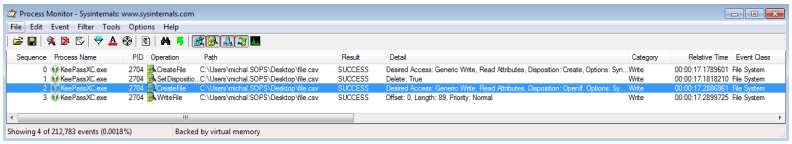
4.3 Možnosti importování a exportování dat

Při exportování a importování databáze uložených hesel dochází k práci s daty v otevřeném textu a nesprávná manipulace s těmito daty může vést k úniku dat. Další funkcí programu, při které může docházet k práci s daty v nezašifrovaném formátu, patří možnost slučování více databází, a proto bude prověřeno fungování všech těchto funkcí. Pro analýzu chování programu při práci se soubory byl využit program Process Monitor, který umožňuje všechny prováděné akce detailně sledovat.

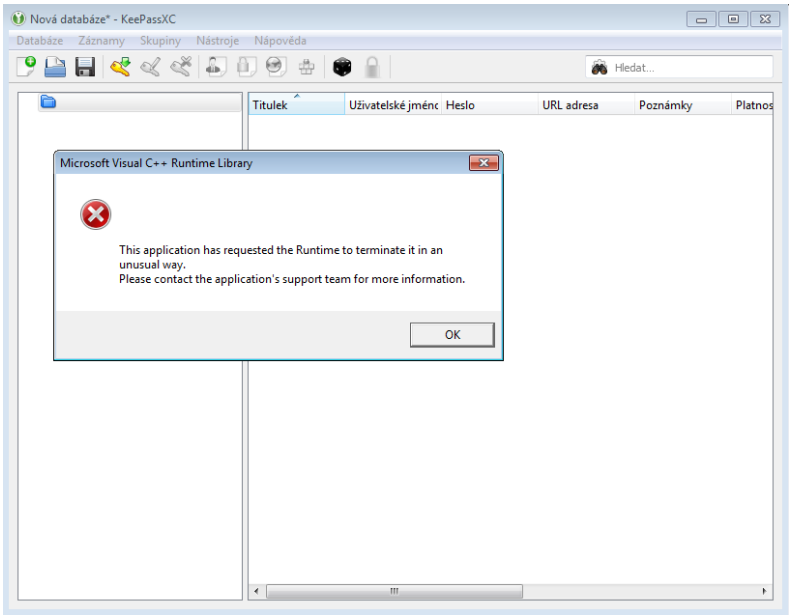
Pro otestování sloučení dvou databází byly vytvořeny databáze *a.kdbx* a *b.kdbx* a pomocí Process Monitoru bylo sledováno chování programu. Nejdříve byla otevřena databáze *a.kdbx* a do té byla sloučena databáze *b.kdbx*. Jak lze vidět na obrázku 4.5, nebyly vytvořeny žádné soubory mimo složku s výslednou databází. Při sloučení databází byl vytvořen nový dočasný soubor *a.kdbx.Spaolg*, který při uložení původní databázi *a.kdbx* přepsal. To zaručuje zachování původních verzí databází i nové sloučené databáze až do chvíle finálního uložení.

Exportování databáze proběhlo tak, jak by uživatel očekával, a byl vytvořen pouze jeden CSV soubor s exportovanými daty. Při testu byly i správně zpracovány speciální znaky dle normy RFC 4180 [36] pro soubory CSV. Při importování dat ze souboru KDBX1 i CSV program správně zpracoval prázdné soubory a KDBX soubor s neplatnou hlavičkou. Soubory vytvořené při exportování databáze lze vidět na obrázku 4.6.

Problém nastal pouze při pokusu o importování velkého testovacího souboru, který měl velikost okolo 5GB. Jak je možné vidět na obrázku 4.7, program nezvládl soubor zpracovat a spadl. Neočekávaný pád programu by samozřejmě neměl u žádné produkční aplikace nastat a jakákoliv výjimka by měla být zachycena. Tato chyba ovšem neohrožuje zabezpečení uživatelových dat a nastala pouze při pokusu o importování velmi velkého souboru.



Obrázek 4.6: Process Monitor při exportování databáze



Obrázek 4.7: Pád programu při importování velkého souboru

4.4 Síťový provoz

Dle oficiální dokumentace program KeePassXC běžně nevyužívá síťového připojení, protože nepodporuje online aktualizace a ani nenahrává databázi na internetové úložiště. Je dokonce možné zdrojový kód zkompilovat bez podpory pro síťovou komunikaci.

Určité funkce programu však síťové připojení vyžadují, ale ve výchozím nastavení jsou všechny vypnuty. Jde konkrétně o možnost stahování favicon webových stránek a použití těchto ikon v grafickém rozhraní programu u jednotlivých záznamů.

Pro zachycení provozu byly použity programy Process Monitor a Wire-Shark. Při výchozím nastavení opravdu nebyl programem generován žádný síťový provoz.

Tabulka 4.1: Potenciálně zranitelná místa nalezená při analýze uživatelského rozhraní

Označení	Popis
Z1	Konfigurační možnost bezpečného ukládání databáze
Z2	Zamčení databáze při zavření víka a ukončení relace
Z3	Kontrola hesla při vytváření nové databáze
Z4	Analýza náhodného generátoru
Z5	Šifrování databáze pomocí faktoru Výzva-odpověď

4.5 Shrnutí

Při analýze uživatelského rozhraní bylo nalezeno několik míst, které by mohla představovat bezpečnostní riziko a budou dále zkoumána přímo ve zdrojovém kódu aplikace. Přehled všech míst a jejich popis lze nalézt v tabulce 4.1.

Kapitola 5

Analýza zdrojového kódu aplikace

V první fázi analýzy zdrojového kódu byly prozkoumány používané šifrovací funkce, jejich reputace a implementace ve zdrojovém kódu. V druhé fázi byla analyzována potenciálně zranitelná místa, která byla nalezena při analýze uživatelského rozhraní programu. Na závěr analýzy zdrojového kódu byly použity automatické nástroje pro kontrolu kompletního kódu programu.

5.1 Šifrovací funkce

Bezpečné ukládání uživatelových hesel a ostatních dat je hlavním úkolem správce hesel. Při analýze uživatelského rozhraní bylo ověřeno, že program při manipulaci s daty nevytváří kopie souborů, která by obsahovala data v otevřeném textu. Je ale také nutné ověřit, zda se za zabezpečná dají považovat i data v zašifrované databázi. Hlavní roli přitom hrají algoritmy, kterými jsou data šifrována. V této kapitole ověříme, zda použité šifrovací algoritmy odpovídají aktuálním standardům. Neméně důležitou částí je také analýza samotné implementace těchto algoritmů ve zdrojovém kódu.

Při analýze uživatelského rozhraní bylo zjištěno, že program umožňuje šifrování pomocí algoritmů AES, Twofish a ChaCha20. Všechny algoritmy používají klíč o délce 256 bitů. Algoritmus AES je jedním z šifrovacích algoritmů schváleným NIST [37] a je považován za standard pokročilého šifrování. Algoritmus Twofish byl jedním z pěti finalistů pří výběru nového standardu pokročilého šifrování, který proběhl v letech 1997 až 2000 [38]. Při výběru byl tento algoritmus posouzen jako bezpečný a nebyly u něj nalezeny žádné útočné vektory. V soutěži neuspěl především díky složitosti své implementace [38]. Algoritmus ChaCha20 je na rozdíl od šifer AES a Twofish proudovou šifrou a vyniká především svou rychlostí. ChaCha20 je považován za alternativu k AES [39]. Bezpečnost tohoto algoritmu potvrdila i bezpečnostní studie

provedená v roce 2017 japonským výzkumným institutem KDDI [40].

Pro vytvoření šifrovacího klíče program umožňuje volbu jedné ze dvou funkcí pro odvození klíče. Varianta AES-KDF je standardem doporučovaným intitutem NIST [37]. Druhou variantou je použití funkce Argon2, která si získala popularitu po svém vítězství v soutěži „Password Hashing Competition" v roce 2015 [41]. U obou funkcí lze tedy považovat jejich zabezpečení za dostatečné.

V dalším kroku byla prověřena implementace šifrovacích funkcí přímo ve zdrojovém kódu programu. Pro nalezení výchozího bodu pro analýzu bylo nalezeno místo v kódu, které zachycuje požadavek na uložení databáze, čímž by mělo být zároveň spuštěno šifrování dat. Celý proces šifrování a zápis dat do souboru zajišťují třídy *Kdbx4Writer* a *Kdbx3Writer* v závislosti na verzi formátu výsledného souboru. Obě třídy mají velmi podobný kód a liší se především v implementaci ověřování integrity dat v hlavičce. Detailnější analýze byla podrobena především verze *Kdbx4Writer*, která podporuje nejnovější funkce a způsoby ověřování.

Pro vytvoření inicializačního vektoru (IV) a soli je použit stejný náhodný generátor, jehož funkčnost a bezpečnost je analyzována v podkapitole 5.2.4. Nový IV a sůl jsou generovány při každém běhu šifrovací funkce a nejsou závislé na instanci běhu programu. Tím nedochází k použití předvídatelných hodnot u použitých funkcí.

Samotné algoritmy jsou implementovány knihovnou `Libgcrypt`. Ve všech analyzovaných metodách byl dodržen referenční postup algoritmů AES, Twofish a ChaCha20. Nebylo nalezeno místo představující bezpečnostní hrozbu pro uživatele programu.

Detailní popis postupu šifrování s komentáři k jednotlivým krokům byl přidán přímo do zdrojových kódu *Kdbx4Writer_komentovane.cpp*, *SymmetricCipher_komentovane.cpp* a *SymmetricCipherGcrypt_komentovane.cpp*. Všechny zakomentované soubory jsou přílohou této práce.

5.2 Analýza potenciálně zranitelných míst

V následující podkapitole byla blíže prozkoumána potenciálně zranitelná místa, která byla nalezena při analýze uživatelského rozhraní. Nejdříve byly ve zdrojovém kódu přesně lokalizovány ty části kódu, které zajišťují danou funkcionalitu v uživatelském rozhraní. Poté byla nalezena všechna použití těchto funkcí v programu a bylo zhodnoceno, zda představují bezpečnostní riziko pro uživatele.

5.2.1 Z1 – Bezpečné ukládání databáze

Obecné nastavení programu umožňovalo konfigurační možnost bezpečného ukládání databáze. Uživatel programu předpokládá, že jsou jeho data uklá-

```
<item>
 <widget class="QCheckBox" name="useAtomicSavesCheckBox">
  <property name="text">
   <string>Safely save database files (may be incompatible with Dropbox, etc)</string>
  </property>
  <property name="checked">
   <bool>true</bool>
  </property>
 </widget>
</item>
```

Obrázek 5.1: Možnost bezpečného ukládání databáze

dána bezpečně za jakýchkoliv okolností. Přesný význam této konfigurační možnosti byl prozkoumán přímo ve zdrojovém kódu.

Ve zdrojovém souboru *SettingsWidgetGeneral.ui* grafického rozhraní bylo zjištěno, že se jedná o konfigurační možnost s názvem *useAtomicSavesCheckBox* jak lze vidět na obrázku 5.1. Pomocí příkazu *grep -r* bylo zjištěno další použití tohoto nastavení v ostatních souborech zdrojového kódu.

Proměnná *useAtomicSaves* se vyskytuje především ve třídě *Settings* a *Config*, které se starají o nastavení výchozí hodnoty a ukládání konfigurace do konfiguračního souboru. Nastavení je použito na jediném místě, kdy tato proměnná typu bool vstupuje jako parametr metody *saveToFile* třídy *Database*.

Do metody *saveToFile*, která je zobrazena na obrázku 5.2, vstupuje možnost bezpečného ukládání databáze jako parametr *atomic*. Tento parametr určuje způsob, jakým bude konfigurační soubor uložen. Pokud je předána hodnota *true*, je pro uložení souboru použita třída *QSaveFile*. Tato třída podporuje atomické zapisování do souboru a nehrozí tak ztráta dat pokud by došlo k neočekávanému pádu programu v nevhodnou chvíli. K tomu může dojít, pokud není tato konfigurační možnost zapnutá. Dle zdrojového kódu je totiž při vypnuté možnosti bezpečného ukládání databáze využívána třída *QTemporaryFile*, která nepodporuje atomické operace.

Při analýze bylo zjištěno, že konfigurační možnost bezpečného ukládání databáze nesouvisí se samotným zabezpečením uživatelových dat, ale ovlivňuje způsob, jakým program manipuluje s dočasnými soubory. Z tohoto důvodu nebyla shledána tato konfigurační možnost hrozbou pro uživatele.

5.2.2 Z2 – Zamykání databáze

Druhým zkoumaným místem byla možnost uzamknutí databáze při zavření víka notebooku nebo při ukončení uživatelské relace. Při analýze bylo postupováno stejně jako v předchozím bodu. Po nalezení přepínače v kódu grafického rozhraní byly nalezeny ostatní zdrojové kódy používající tuto konfiguraci.

Konfigurační možnost uzamknutí databáze po zavření víka notebooku je využívána pouze ve třídě MainWindow, která je implementována v souboru *MainWindow.cpp*. Po zachycení signálu *screenLocked* je spuštěna metoda *handleScreenLock*, která se stará o uzamknutí databáze, pokud je tato konfigurační možnost zapnutá. Pro bližší pochopení fungování této funkcionality bylo prozkoumáno, jak se signál *screenLocked* vyvolává.

```
QString Database::saveToFile(QString filePath, bool atomic, bool backup)
{
    QString error;
    if (atomic) {
        QSaveFile saveFile(filePath);
        if (saveFile.open(QIODevice::WriteOnly)) {
            // write the database to the file
            error = writeDatabase(&saveFile);
            if (!error.isEmpty()) {
                return error;
            }

            if (backup) {
                backupDatabase(filePath);
            }

            if (saveFile.commit()) {
                // successfully saved database file
                return {};
            }
        }
        error = saveFile.errorString();
    } else {
        QTemporaryFile tempFile;
        if (tempFile.open()) {
            // write the database to the file
            error = writeDatabase(&tempFile);
            if (!error.isEmpty()) {
                return error;
            }

            tempFile.close(); // flush to disk

            if (backup) {
                backupDatabase(filePath);
            }

            // Delete the original db and move the temp file in place
            QFile::remove(filePath);
            if (tempFile.rename(filePath)) {
                // successfully saved database file
                tempFile.setAutoRemove(false);
                return {};
            }
        }
        error = tempFile.errorString();
    }
    // Saving failed
    return error;
}
```

Obrázek 5.2: Metoda ukládající databázi do souboru

Signál *screenLocked* je vyvoláván několika třídami v závislosti na operačním systému, na kterém je program spuštěn. O vyvolání signálu v námi analyzované verzi systému se stará třída implementovaná v souboru *ScreenLockListenerWin.cpp*. Tato třída registruje zachytávání Windows zpráv s názvem WM_POWERBROADCAST a WM_WTSSESSION_CHANGE.

U zprávy WM_POWERBROADCAST se jednalo konkrétně o typ změny GUID_LIDSWITCH_STATE_CHANGE a PBT_APMSUSPEND a u zpráv WM_WTSSESSION_CHANGE o typ WTS_CONSOLE_DISCONNECT a WTS_SESSION_LOCK.

Zachytáváním těchto zpráv program detekuje uzamknutí Windows relace uživatele a změny na úrovni systému, kterými je uspání systému a uzavření víka notebooku. Implementace kódu tedy odpovídá očekávanému chování pro-

```
connect(m_ui->buttonBox, SIGNAL(accepted()), SLOT(generateKey()));
```

Obrázek 5.3: Potvrzení změny nastavení hesla databáze

```
void ChangeMasterKeyWidget::generateKey()
{
    m_key.clear();

    if (m_ui->passwordGroup->isChecked()) {
        if (m_ui->enterPasswordEdit->text() == m_ui->repeatPasswordEdit->text()) {
            if (m_ui->enterPasswordEdit->text().isEmpty()) {
                if (MessageBox::warning(this, tr("Empty password"),
                                        tr("Do you really want to use an empty string as password?"),
                                        QMessageBox::Yes | QMessageBox::No) != QMessageBox::Yes) {
                    return;
                }
            }
            m_key.addKey(PasswordKey(m_ui->enterPasswordEdit->text()));
        }
        else {
            m_ui->messageWidget->showMessage(tr("Different passwords supplied."), MessageWidget::Error);
            m_ui->enterPasswordEdit->setText("");
            m_ui->repeatPasswordEdit->setText("");
            return;
        }
    }
```

Obrázek 5.4: Kontrola hlavního hesla databáze

gramu a nejedná se o bezpečnostní hrozbu pro uživatele.

5.2.3 Z3 – Kontrola hlavního hesla

V následujícím bodě byla blíže zkoumána funkce vytvoření nové databáze pomocí hesla jako jediného ověřovacího mechanismu. Dle směrnice NIST [6] by měl program při vyváření nového hesla požadovat heslo alespoň o délce 8 znaků. Zároveň by měl progam uživatele upozornit, pokud se jedná o slabé heslo. Přímo ve zdrojovém kódu programu zjistíme, zda této normě program odpovídá.

Stejně jako v předchozích bodech byl i zde nejdříve nalezen zdrojový kód uživatelského rozhraní, který zajišťuje vytváření nové databáze. Jedná se konkrétně z o třídu *ChangeMasterKeyWidget*. Tato třída implementuje možnost jak změny hlavního hesla, tak vytvoření nové databáze. Potvrzením okna pro změnu hesla je spuštěna metoda *generateKey()*, jak je vidět na obrázku 5.3.

Při bližším pohledu na metodu *generateKey()*, která je zobrazena na obrázku 5.4, je vidět, že heslo je kontrolováno pouze na výskyt prázdného řetězce a shodu prvního a opakovaného hesla. Při zadání prázdného řetězce do textového pole pro vložení hesla je vyvoláno varování a po jeho potvrzení je toto heslo přijato.

Pokud je heslo akceptováno, je incializován objekt *PasswordKey* a ten je přidán metodou *addKey()* do proměnné *m_key* datového typu *CompositeKey*. Při analýze kódu objektu *PasswordKey* a *CompositeKey* nebylo zjištěno, že by program prováděl další kontroly zadaných hodnot.

Při analýze vytváření nové databáze za pomocí nového hesla bylo zjištěno, že program nevyžaduje jakoukoliv komplexnost hesla a ani nekontroluje jeho

```
void RandomBackendGcrypt::randomize(void* data, int len)
{
    Q_ASSERT(Crypto::initalized());

    gcry_randomize(data, len, GCRY_STRONG_RANDOM);
}
```

Obrázek 5.5: Generování náhodných dat

délku. Při analýze uživatelského rozhraní zároveň nebyl nalezen ukazatel entropie hesla, který by uživateli napověděl, zda je jím zvolené heslo bezpečné. Tato zjištění odporují normě NIST [6]. V následující kapitole bude zhodnocena závažnost této skutečnosti pod označením **V1** a bude navrženo možné řešení.

5.2.4 Z4 – Náhodný generátor

Nevhodné implementace náhodného generátoru pro vytváření souboru s klíčem může vést k předvídatelným výsledkům. Jak bylo vysvětleno v podkapitole 3.3.2, pokud známe zdrojový kód, kterým se data generují a je možné odhadnout inicializační hodnotu (random seed), je možné vypočítat výsledný klíč. Pokud se například jako inicializační hodnota využívá hodnota odvozená od aktuálního času a je známo, kdy přibližně byl algoritmus spuštěn, je velice limitován počet možných variant výsledného klíče. [42]

Při analýze generátoru náhodného klíče byla zkoumána funkce spuštěná při stisknutí tlačítka *Vytvořit* v grafickém rozhraní. Po zadání cesty k souboru je tato cesta předána jako parametr metodě *create()* objektu *FileKey*. Tato metoda využívá volání metody *randomArray()* s integer parametrem o hodnotě 128 a postupně je voláno několik dalších podpůrných metod. Samotné generování náhodných dat probíhá při volání metody *randomize()*, které spouští metodu *gcry_randomize()* z externí knihovny `Libgcrypt`, jak je vidět na obrázku 5.5.

`Libgcrypt` je open-source knihovnou implementující velké množství kryptografických operací. Dle oficiální dokumentace funkce *gcry_randomize()* využívá na systému Windows pro generování náhodných bajtů aktuální vlastnosti systému [43]. Při nahlédnutí do zdrojového kódu souboru *rndw32.c* implementujícího generátor náhodných dat na systému Windows lze vidět, že pro vytvoření náhodné hodnoty jsou kombinována data a statistiky například o zápisu na pevný disk, aktuálním síťovém toku, stavu napájení, aktivním okně, pozici ukazatele myši nebo uplynulého času od zapnutí systému [44].

Knihovna `Libgcrypt` tedy pro generování náhodných dat používá dostatek náhodných hodnot a tento generátor je možné považovat za bezpečný. Velikost souboru s klíčem (128 bajtů), který program KeePassXC generuje, se dá také považovat za dostatečný vzhledem k tomu, že takový soubor má vyšší entropii než 64 bitů doporučených dle normy NIST [6].

```
int yk_challenge_response(YK_KEY *yk, uint8_t yk_cmd,
                          int may_block,
                          unsigned int challenge_len,
                          const unsigned char *challenge,
                          unsigned int response_len,
                          unsigned char *response)
```

Obrázek 5.6: Funkce přijímající výzvu na odpověď tokenu

5.2.5 Z5 – Faktor výzva-odpověď

Pro ověření přístupu do databáze umožnuje program KeePassXC použití kryptografického tokenu „Výzva-odpověď“. Fungování tohoto způsobu ověřování je implementováno ve třídách *YkChallengeResponseKey* a *YubiKey*, které byly blíže zkoumány.

Třída *YubiKey* představuje samotný token a umožňuje získání jeho sériového čísla, inicializaci nového tokenu, detekování připojených tokenů a samotnou výzvu. Ve všech metodách využívá kód programu KeePassXC volání funkcí knihovny `Libyubikey`. Jedná se o knihovnu s otevřeným zdrojovým kódem dostupnou na stránkách firmy Yubico [45]. Tato knihovna zajišťuje interakci přímo s kryptografickým tokenem.

Nejzásadnější funkcí je funkce *yk_challenge_response*, kterou lze vidět na obrázku 5.6. Funkce předá tokenu výzvu uloženou pod ukazatelem *challenge* a kryptografický token uloží odpověď na tuto výzvu do pole pod ukazatelem *response*. Odpoveď je výstupem funkce HMAC-SHA1, kterou kryptografický token provádí pomocí svého tajného klíče.

Bezpečnost v tomto případ ovlivňuje především délka výzvy a tím možný počet odpovědí tokenu. Ve zdrojovém kódu bylo zjištěno, že jako výzva se u databáze vždy používá hlavní sůl, která je zároveň i součástí nešifrované části hlavičky databáze. Jak byl zjištěno při analýze šifrovacích funkcí v sekci 5.1, tak délka soli je 32 bajtů. Délka výzvy tedy splňuje normu NIST [6], které vyžaduje délku výzvy alespoň 8 bajtů.

Důležitou částí je také implementace kryptografické šifrovací šifry HMAC přímo v kryptografickém tokenu. Nejnovější verze tokenu YubiKey 4 má ovšem uzavřený kód a přesnou implementaci této funkce není bez pokročilejších technik reverzního inženýrství možné ověřit.

5.3 Statická analýza zdrojového kódu

Statická analýza kódu byla provedena pomocí programu Cppcheck. Pomocí tohoto programu byla provedena analýza všech zdrojových kódů programu ve složce *src/*. Ve výchozím nastavení, bez použití dodatečných přepínačů, program Cppcheck zobrazuje pouze zprávy na úrovni *error*. Tato úroveň označuje

```
(warning) Call of pure virtual function 'readLength' in constructor.
(warning) Member variable 'BenchmarkThread::m_rounds' is not initialized in the constructor.
(warning) Member variable 'Request::m_sortSelection' is not initialized in the constructor.
(warning) Member variable 'BinaryStream::m_device' is not initialized in the constructor.
(warning) Member variable 'SymmetricCipherStream::m_streamCipher' is not initialized in the constructor.
(warning) Member variable 'QtIOCompressorPrivate::manageDevice' is not initialized in the constructor.
```

Obrázek 5.7: Varování vygenerovaná programem Cppcheck

chyby v programu, které mohou představovat bezpečnostní riziko. Program nenalezl v kódu žádné chyby na této úrovni.

Pro zapnutí dodatečných kontrol byl použit přepínač *–enable=all*, který umožnuje spuštění těchto kontrol. Program varoval před nalezením nepoužitých funkcí, duplicitních podmínek v podmínkách nebo přepsání hodnot proměnných před jejich použitím. Nalezeno bylo i šest upozornění na úrovní *warning*, které jsou vidět na obrázku 5.7.

Jedno z varování upozorňuje před použitím abstraktní funkce v konstruktoru objektu. Zbylá varování upozorňují před inicializací proměnné mimo konstruktor, což může vést k nedefinovanému chování programu. Po bližším zkoumání bylo zjištěno, že nalezené výsledky nepředstavují bezpečnostní riziko. Detekovaná varování se týkala spíše stylových nedostatků ve zdrojovém kódu, které sice mohou snižovat důvěryhodnost programu, ale nemají vliv na zabezpečení dat uživatelů.

5.4 Použité knihovny

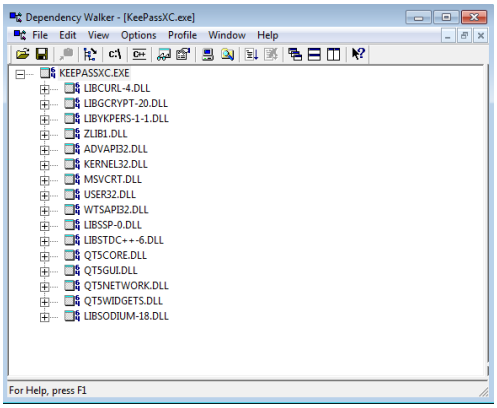
Zranitelnosti knihoven, které jsou v aplikaci používány, mohou být pro výsledný produkt stejně nebezpečné jako zranitelnosti v samotném kódu. Nebezpečné může být i použití knihoven z neznámých zdrojů, které mohou obsahovat škodlivý kód. Externí knihovny jsou v programu KeePassXC použity pro šifrování dat nebo pro při implementaci ověřovacího faktoru *Výzva-odpověď*, který je jeden ze tří autentizačních faktorů, který program KeePassXC podporuje.

Pro zjištění knihoven importovaných programem KeePassXC byl použit nástroj Dependency Walker. Po zadání cesty ke spustitelnému souboru program Dependency Walker zobrazil závislost programu KeePassXC na celkem 16 knihovnách, které lze vidět na obrázku 5.8.

Všechny importované knihovny a důvod jejich použití jsou vysvětleny v dokumentaci. Nebylo odhaleno importování žádné další knihovny u které by nebylo jasné její použití. Analýza jednotlivých knihoven nespadala svou náročností do rozsahu této práce.

5.5 Shrnutí

Analýza zdrojového kódu potvrdila zranitelnost, která může představovat bezpečnostní hrozbu pro uživatele. V následující kapitole 6 bude vyhodnocena její



Obrázek 5.8: Knihovny importované programem KeePassXC

Tabulka 5.1: Zranitelnost nalezená při analýze zdrojového kódu

Označení	Popis
V1	Kontrola hesla při vytváření nové databáze

závažnost a budou navržena možná opatření k její nápravě.

Kapitola 6

Nalezené zranitelnosti a navržená řešení

Pro vyhodnocení závažnosti bylo využito metodologie CVSS pro výpočet závažnosti zranitelností na počítačových systémech. Konkrétně byl vyžit kalkulátor dostupný na internetových stránkách NIST [46].

Při analýze zdrojového kódu byla u funkce zajišťující vytvoření databáze a změnu hlavního hesla zjištěno, že implementace neodpovídá normě amerického Národního institutu standardů a technologie (NIST). Program nijak uživatele neupozorňuje na nízkou entropii hesla a ani uživatele nevaruje před heslem slabým nebo obecně známým.

Závažnost tohoto chybějícího upozornění byla vyhodnocena za středně závažnou. Vzhledem k tomu, že hlavní heslo do databáze může být jediným mechanizmem pro přístup ke všem účtům uživatele, mělo by toto heslo splňovat bezpečnostní doporučení. Uživatele používající správce hesel očekává, že takový program bude mít bezpečnost jako hlavní prioritu. Pokud program uživatele na nebezpečné heslo neupozorní, mohl by uživatel nabýt dojmu, že jeho data jsou chráněná dostatečně.

Program KeePassXC již obsahuje generátor hesel, který zobrazuje přibližnout entropii hesla, a potřebný kód je tedy již v programu implementovaný v souboru *zxcvbn.c*. Řešením by tedy bylo tento ukazatel zobrazit i při vytváření nové databáze nebo změně hesla. Dalším možným opatřením je přidání seznamu známých hesel jako součást instalace programu. Při vytváření databáze a změně hesla by proběhla kontrola, zda heslo zvolené uživatelem není obsažené na seznamu hesel známých. Výše zmíněné změny vyžadují velké úpravy do současného zdrojového kódu. Jako jednoduché řešení se ale nabízí upozornit uživatele na příliš krátké heslo. Takové řešení by vyžadovalo přidání několika řádků do zdrojového kódu souboru *zxcvbn.c*. Navržená změna je vidět na obrázku 6.1.

```
if (m_ui->enterPasswordEdit->text().length() < 8)
{
  if (MessageBox::warning(this, tr("Slabe heslo"),
   tr("Heslo neni dostatecne dlouhe. Opravdu ho chcete zvolit?"),
   QMessageBox::Yes | QMessageBox::No) != QMessageBox::Yes)
   {
     return;
   }
}
```

Obrázek 6.1: Návrh řešení zranitelnosti V1

Závěr

Cílem bakalářské práce bylo bylo podat ucelený pohled na téma bezpečné práce s hesly a seznámit čtenáře s nejznámějšími programy pro správu hesel. Literární rešerše sloužila k lepšímu pochopení fungování jednotlivých typů správců hesel a jejich fungování z pohledu zabezpečení ukládaných dat. V praktické části bylo úkolem provést bezpečnostní analýzu konkrétního správce hesel. Hlavním účelem bylo nalezení potenciálních útočných vektorů, pomocí kterých by bylo možné ohrozit bezpečnost zkoumaného programu.

Postupně bylo zkoumáno uživatelské rozhraní a zdrojový kód programu tak, aby byla v první části analýzy určena místa v uživatelském rozhraní, která by mohla skrývat hrozbu pro uživatele. Ve druhé části byla potenciálně zranitelná místa blíže analyzována přímo ve zdrojovém kódu a byla posouzena jejich bezpečnostní závažnost.

Při analýze uživatelského prostředí bylo nalezeno pět míst, která buď přímo neodpovídala doporučením pro práci s hesly nebo se jednalo o funkce a konfigurační možnosti vzbuzující pochybnosti, zda nepředstavují riziko pro uživatele. Všech pět rizikových míst bylo podrobeno bližší analýze ve zdrojovém kódu programu a byl ověřen jejich přesný účel a chování.

U tří z pěti zkoumaných míst bylo podezření vyhodnoceno jako neopodstatněné a bylo zjištěno, že se jedná o standardní chování programu. V jednom případě analýza zdrojového kódu potvrdila, že chování programu neodpovídá běžným standardům pro práci s hesly. Vzhledem k tomu, že se jednalo o část programu zajišťující kompletní šifrování databáze, byl tento nedostatek označen za středně závažný.

Dále bylo zjištěno, že program nesprávně zpracovává soubory při importování dat, což způsobuje neočekávaný pád programu. V tomto případě se ale jednalo o chybu, která nepředstavovala bezpečnostní riziko pro uživatele.

Výsledky provedené analýzy poukázaly na místa ve zdrojovém kódu, která mohou přímo ohrozit bezpečnost ukládaných dat uživatelů. Jedná se ale o chyby, které lze poměrně lehce odstranit, a proto byla navržena možná řešení zajišťující nápravu. V budoucnosti by bylo možné bezpečnostní analýzu rozšířit

o bližší prozkoumání použitých knihoven, které v některých případech zajišťují kompletní fungování některých částí programu. Tato analýza by ale vyžadovala pokročilou znalost reverzního inženýrství a tak by ji bylo možné zrealizovat například v rámci diplomové práce.

Literatura

[1] Florencio, D.; Herley, C.: A Large-scale Study of Web Password Habits. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, New York, NY, USA: ACM, Květen 2007, ISBN 978-1-59593-654-7, s. 657–666, doi:10.1145/1242572.1242661. Dostupné z: http://doi.acm.org/10.1145/1242572.1242661

[2] Brash, T. L.: Online Overload – It's Worse Than You Thought. Červenec 2015, [cit. 2018-05-08]. Dostupné z: https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/

[3] Kumar, N.: Password In Practice: An Usability Survey. *Journal of Global Research in Computer Science [online]*, Květen 2011, [cit. 2018-03-18]. Dostupné z: http://www.jgrcs.info/index.php/jgrcs/article/view/96/96

[4] Gasti, P.; Rasmussen, K. B.: On the Security of Password Manager Database Formats. *Lecture Notes in Computer Science [online]*, 2012, [cit. 2018-03-19]. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-642-33167-1_44

[5] Turner, D.: Digital Authentication - the basics. *Cryptomathic*, 2016, [cit. 2018-03-25]. Dostupné z: https://www.cryptomathic.com/news-events/blog/digital-authentication-the-basics

[6] National Institute of Standards and Technology: *National Institute of Standards and Technology Special Publication 800-63B*. Červen 2017, [cit. 2018-04-24]. Dostupné z: https://doi.org/10.6028/NIST.SP.800-63b

[7] European Union Agency for Network and Information Security: *Authentication Methods*. [cit. 2018-03-18]. Dostupné z: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/authentication-methods

[8] KeePassXC Team: *KeePassXC - Documentation and FAQ*. [cit. 2018-04-19]. Dostupné z: https://keepassxc.org/docs/

[9] Zhao, R.; Yue, C.: Toward a secure and usable cloud-based password manager for web browsers. *Computers & Security [online]*, Říjen 2014, [cit. 2018-03-25]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167404814001059

[10] Krishna, V.: 5 Best Password Manager That Stores Locally. [cit. 2018-04-19]. Dostupné z: https://techwiser.com/password-manager-that-stores-locally/

[11] Everplans: *The Four Most Popular Password Managers*. [cit. 2018-04-19]. Dostupné z: https://www.everplans.com/articles/the-four-most-popular-password-managers

[12] NetMarketShare: *Browser Market Share*. [cit. 2018-03-25]. Dostupné z: https://www.netmarketshare.com/browser-market-share.aspx

[13] Reichl, D.: KeePass Features. [cit. 2018-03-25]. Dostupné z: https://keepass.info/features.html

[14] Howard, M.; LeBlanc, D.: *Bezpečný kód*. Computer Press, a.s., první vydání, ISBN 978-80-251-2050-7.

[15] Reichl, D.: KeePass Help Center. [cit. 2018-03-25]. Dostupné z: https://keepass.info/help/base/index.html

[16] Manber, U.: A Simple Scheme to Make Passwords Based on One-Way Functions Much Harder to Crack. *Computers & Security [online]*, Únor 1999, [cit. 2018-03-27]. Dostupné z: https://www.sciencedirect.com/science/article/pii/016740489600003X

[17] Li, Z.; He, W.; Akhawe, D.; aj.: The Emperor's New Password Manager: Security Analysis of Web-based Password Managers. Srpen 2014, [cit. 2018-04-20]. Dostupné z: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-li-zhiwei.pdf

[18] Password Safe: *Password Safe - Frequently Asked Questions*. [cit. 2018-04-20]. Dostupné z: https://pwsafe.org/faq.shtml

[19] LogMeIn, Inc: *LastPass Security*. [cit. 2018-04-20]. Dostupné z: https://lastpass.com/support_security.php

[20] LogMeIn, Inc: *Security Update for the LastPass Extension*. [cit. 2018-04-21]. Dostupné z: https://blog.lastpass.com/2017/03/security-update-for-the-lastpass-extension.html/

[21] Dashlane, Inc.: *The password manager, perfected.* [cit. 2018-04-19]. Dostupné z: https://www.dashlane.com/features/password-manager

[22] Dashlane, Inc.: *How many words are there in the English language?* Prosinec 2017, [cit. 2018-04-19]. Dostupné z: https://www.dashlane.com/download/Dashlane_SecurityWhitePaper_December2017.pdf

[23] AgileBits, Inc: *1Password Security Design.* Duben 2017, [cit. 2018-04-23]. Dostupné z: https://1password.com/files/1Password%20for%20Teams%20White%20Paper.pdf

[24] McCarney, D.: *Password Managers: Comparative evaluation, design, implementation and empirical analysis.* Diplomová práce, Carleton University, Duben 2013.

[25] Palant, W.: Master password in Firefox or Thunderbird? Do not bother! Březen 2018, [cit. 2018-05-12]. Dostupné z: https://palant.de/2018/03/10/master-password-in-firefox-or-thunderbird-do-not-bother

[26] Armerding, T.: The 17 biggest data breaches of the 21st century. Leden 2018, [cit. 2018-05-04]. Dostupné z: https://www.csoonline.com/article/2130877/data-breach/the-biggest-data-breaches-of-the-21st-century.html

[27] National Institute of Standards and Technology: *National Institute of Standards and Technology Special Publication 800-90A*. Červen 2015, [cit. 2018-05-02]. Dostupné z: http://dx.doi.org/10.6028/NIST.SP.800-90Ar1

[28] National Institute of Standards and Technology: *National Institute of Standards and Technology Special Publication 800-90B.* Leden 2018, [cit. 2018-05-02]. Dostupné z: https://doi.org/10.6028/NIST.SP.800-90B

[29] NetMarketShare: *Operating System Market Share.* [cit. 2018-02-27]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx

[30] Open Web Application Security Project: *Source Code Analysis Tools.* [cit. 2018-04-07]. Dostupné z: https://www.owasp.org/index.php/Source_Code_Analysis_Tools

[31] Marjamaki, D.: *Cppcheck - A tool for static C/C++ code analysis.* [cit. 2018-04-07]. Dostupné z: https://sourceforge.net/p/cppcheck/wiki/Home/

[32] Russinovich, M.: Windows Sysinternals. [online], [cit. 2018-04-24]. Dostupné z: https://docs.microsoft.com/en-us/sysinternals/

[33] Miller, S.: Dependency Walker 2.2. [online], [cit. 2018-04-24]. Dostupné z: http://www.dependencywalker.com/

[34] Wireshark Foundation: *About Wireshark*. [cit. 2018-04-25]. Dostupné z: https://www.wireshark.org/

[35] Oxford University Press: *How many words are there in the English language?* [cit. 2018-05-07]. Dostupné z: https://en.oxforddictionaries.com/explore/how-many-words-are-there-in-the-english-language/

[36] Shafranovich, Y.: *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. [cit. 2018-04-25]. Dostupné z: https://tools.ietf.org/html/rfc4180

[37] National Institute of Standards and Technology: *National Institute of Standards and Technology Special Publication 800-175B*. Srpen 2016, [cit. 2018-05-05]. Dostupné z: http://dx.doi.org/10.6028/NIST.SP.800-175B

[38] National Institute of Standards and Technology: *Cryptographic Standards and Guidelines*. Prosinec 2016, [cit. 2018-05-06]. Dostupné z: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development

[39] Nir, Y.: *ChaCha20 and Poly1305 for IETF Protocols*. [cit. 2018-05-06]. Dostupné z: https://tools.ietf.org/html/rfc7539

[40] KDDI Research, Inc.: *Security Analysis of ChaCha20-Poly1305 AEAD*. Únor 2017, [cit. 2018-05-06]. Dostupné z: http://www.cryptrec.go.jp/estimation/cryptrec-ex-2601-2016.pdf

[41] Password Hashing Competition: *Password Hashing Competition and our recommendation for hashing passwords*. [cit. 2018-05-06]. Dostupné z: https://password-hashing.net/

[42] Garfinkel, S.; Spafford, G.; Schwartz, A.: *Practical Unix & Internet Security*. O'Reilly Media, třetí vydání, 2003, ISBN 978-0596003234.

[43] GnuPG: *Random-Number Subsystem Architecture [online]*. [cit. 2018-05-05]. Dostupné z: https://www.gnupg.org/documentation/manuals/gcrypt/Random_002dNumber-Subsystem-Architecture.html

[44] GnuPG: *W32 entropy gatherer [online]*. [cit. 2018-05-05]. Dostupné z: https://github.com/gpg/libgcrypt/blob/master/random/rndw32.c

[45] Yubico: *YubiKey Personalization*. [cit. 2018-05-06]. Dostupné z: https://developers.yubico.com/yubikey-personalization/

[46] National Institute of Standards and Technology: *Common Vulnerability Scoring System Calculator*. [cit. 2018-05-12]. Dostupné z: `https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator`

Příloha A

Seznam použitých zkratek

3DES Tripple Data Encryption Standard

AES Advanced Encryption Standard

API Application Programming Interface

CBC Cipher Block Chaining

CSV Comma Seperated Values

ENISA European Union Agency for Network and Information Security

HMAC Keyed-Hash Message Authentication Code

IV Inicializační Vektor

NIST National Institute of Standards and Technology

PBKDF2 Password-Based Key Derivation Function 2

PIN Personal Identification Number

RAM Random Access Memory

SMS Short Message Service

URL Uniform Resource Locator

USB Universal Serial Bus

XML Extensible Markup Language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `keepassxc_source` ..... zdrojové kódy programu KeePassXC
  - `BP_Kavan_Michal_2018.tex` .. zdrojová forma práce ve formátu LaTeX
  - `img` ..... adresář s obrázky použitými v textu
- `text` ..... text práce
  - `BP_Kavan_Michal_2018.pdf` ..... text práce ve formátu PDF
- `output` ..... adresář s výstupy programů a snímky obrazovky
  - `cppcheck` ..... adresář s výstupy programu cppcheck
  - `import_export_slouceni` ..... výstupy programu Process Monitor
  - `pouzite_knihovny` ..... výstup programu Dependency Walker
  - `sifrovani` ..... komentované soubory zajišťující šifrování
  - `vychozi_nastaveni` ..... snímky výchozího nastavení programu
  - `cvss` ..... vyhodnocení závažnosti zranitelností