FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Bezpečnostní analýza programu Passbolt
Student:	Radek Smejkal
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Seznamte se s problematikou správců hesel.
Zaměřte se na program Passbolt (https://www.passbolt.com), nastudujte jeho možnosti a porovnejte ho s vybranými konkurenčními programy.
Proveďte analýzu uživatelského prostředí. Vyberte několik míst, která mohou mít významnější bezpečnostní dopady.
Prostudujte zdrojový kód aplikace se zaměřením na nalezená riziková místa; analyzujte také rozšíření do prohlížeče. Vyhodnoťte, zda je implementace bezpečná.
U případných nalezených zranitelností zdokumentujte jejich příčiny, vyhodnoťte míru rizika a velikost dopadů a navrhněte nápravu.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Pavel Tvrdík, CSc.
vedoucí katedry

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

V Praze dne 24. ledna 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Bezpečnostní analýza programu Passbolt

Radek Smejkal

Katedra počítačových systémů
Vedoucí práce: Ing. Josef Kokeš

10. května 2019

Poděkování

Rád bych poděkoval svému vedoucímu práce panu Ing. Josefu Kokešovi za jeho cenné rady, připomínky a nápady v průběhu vzniku této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl 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.

V Praze dne 10. května 2019 .....

Odkaz na tuto práci

Smejkal, Radek. *Bezpečnostní analýza programu Passbolt.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá problematikou hesel a jejich bezpečným ukládáním za použití správců hesel. Zároveň poskytuje stručný přehled současných správců hesel se zaměřením na bezpečnostní funkcionality a jejich vzájemné porovnání.

Práce zpracovává bezpečnostní analýzu správce hesel Passbolt, který se zaměřuje na podporu týmové spolupráce a sdílení hesel. Potenciálně riziková místa jsou hlouběji zkoumána přímo ve zdrojovém kódu aplikace. Analýza zahrnuje také monitorování a následný průzkum síťového provozu.

Během analýzy nebyla nalezena žádná zranitelnost přímo ohrožující utajení či integritu ukládaných dat. Analýza nicméně objevila několik situací, které jistým způsobem mohou vést ke snížení bezpečnosti nebo i k úplné kompromitaci účtu uživatele. Po vyhodnocení závažnosti jednotlivých nálezů jsou navržena možná opatření k eliminaci těchto zranitelností.

Klíčová slova správce hesel, bezpečnostní analýza, IT bezpečnost, šifrování, Passbolt, ukládání hesel

Abstract

This work focuses on password management and secure password storage using password managers. It also provides a brief overview of current password managers, focusing on security features and their comparison.

The aim of this work is to perform a security analysis of the selected password manager Passbolt, which was designed to support team collaboration and password sharing. The potentially risky places found are further examined directly in the application's source code. The analysis also includes monitoring and subsequent network traffic exploration.

During the analysis, no vulnerability directly endangering the confidentiality or integrity of the stored data was found. However, the analysis has identified several situations that may in some ways lead to a reduction of security or even a complete compromise of the user's account. After evaluating the severity of each finding, possible measures to eliminate these vulnerabilities are suggested.

Keywords password manager, security analysis, IT security, encryption, Passbolt, password storage

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Používání internetových služeb a ukládání citlivých dat online představuje možné riziko neoprávněného přístupu cizích osob. Jako základní faktor ověření identity uživatele se používají hesla nebo heslovité fráze (tzv. passphrase). S rostoucím počtem různých uživatelských účtů přidružených k používaným službám lze jednoduše ztratit kontrolu nad správou hesel, potřebnou pro správné zabezpečení.

Aplikace umožňující bezpečné uložení přístupových údajů s jejich pohodlným opětovným použitím se nazývají správci hesel (z ang. password manager). Správci hesel by měly disponovat velmi kvalitním zabezpečením vzhledem k druhu a počtu ukládaných údajů.

V některých případech dochází také k potřebě sdílení přístupových údajů. Tento koncept sice není obecně doporučovaný, nicméně existují situace, které se bez něj neobejdou. Jedná se např. o hesla do externích služeb v rámci společnosti nebo o přístupy do systémů povolujících pouze jediného uživatele. Na to by měly správci hesel pamatovat a zprostředkovat bezpečný proces sdílení hesel.

Pro svou práci jsem si zvolil analýzu programu Passbolt, který se zaměřuje právě na podporu týmové spolupráce a na možnosti sdílení hesel. Jeho zaměření ho směřuje do komerční oblasti, a proto není příliš známý u běžných uživatelů.

Tato práce se skládá ze dvou částí, teoretické a praktické. Cílem první části práce je seznámení čtenáře s problematikou hesel, jejich bezpečným ukládáním a zvyšující se potřebou správců hesel nezbytných pro bezpečnou správu hesel. Práce poskytuje čtenáři základní přehled o současných možnostech správců hesel, jejich hlavních rozdílech a funkcionalitách se zaměřením na bezpečnost. Kapitola 2 detailně popisuje vybranou aplikaci Passbolt, její výhody, nevýhody a především odlišnosti od ostatních správců hesel.

Praktická část práce z bezpečnostního pohledu analyzuje uživatelské rozhraní, síťový provoz i zdrojový kód aplikace včetně doplňku do prohlížeče. První část analýzy vybírá potenciálně riziková místa nebo místa s velkým vlivem na bezpečnost dat v souvislosti s účelem správce hesel. Navazující části hlouběji zkoumají vybraná místa v kódu, dokumentují nalezené zranitelnosti pro následné vyhodnocení možných rizik, určení velikosti dopadů a případné navrhnutí opravy.

Kapitola 1

Nakládání s hesly a porovnání současných řešení

V dnešní době online technologií lidé přirozeně využívají služby, které jim obchodní společnosti nabízí prostřednictvím internetu. Aby služba rozlišila, s kým komunikuje, a aby zabránila neoprávněnému přístupu, potřebuje způsob ověřování identity druhé strany.

Právě autentizační způsoby brání útočníkům v přístupu do emailů, internetových bankovnictví a dalších účtů. Mimo klasických hesel existují i jiné způsoby elektronického ověření identity, nicméně se používají méně kvůli větší náročnosti technického provedení. Další faktory se využívají stále častěji, ovšem až v druhém nebo dalším kroku. Rozhodně není dobré používat slabé heslo a spoléhát se pouze na druhý faktor.

1.1 Základní faktory autentizace

Způsoby ověření identity uživatele se dělí na tyto možné faktory, pomocí kterých aplikace uděluje přístupy do příslušných účtů.

Znalost – Tajná informace, kterou zná pouze vlastník. To zahrnuje právě hesla, bezpečnostní fráze, různé kódy nebo i PINy. [1]

Vlastnictví – Fyzické vlastnictví unikátního zařízení. Zástupcem tohoto faktoru jsou např. HW klíče [1]. Zahrnují se sem i jednorázové SMS kódy, které de facto nejsou spojené přímo s fyzickým zařízením a přístup k těmto zprávám mají i další aplikace telefonu. Navíc vyžadují službu třetí strany a bohužel nelze vyloučit její kompromitaci či odposlech. Do této kategorie patří také mobilní autentizační aplikace, navázané na jediný telefon. Tyto aplikace, narozdíl od SMS zpráv, mohou zaručit šifrovanou komunikaci a unikátnost použitého zařízení, proto zajišťují rozhodně kvalitnější ověření než SMS kódy. [2]

Biometrie – Jedinečná biologická charakteristika člověka. Použití tohoto faktoru komplikují větší technické potíže než např. u hesel. Z biometrických faktorů se používá otisk prstu, sken sítnice, sken duhovky, otisk dlaně nebo rozpoznání obličeje. [3]

O vícefaktorové autentizaci mluvíme tehdy, pokud autentizační proces využívá alespoň dva z těchto základních faktorů [1]. Služba může navíc zohlednit také další identifikátory, např. přibližnou geolokaci či IP adresu. Kvůli jednoduchému způsobu podvrhnutí a jiným přítěžím (cestování apod.) se určitě nedají tyto rozšiřující faktory použít jako primární zdroj ověření. Nicméně často dokáží upozornit v případě neoprávněných pokusů o přihlášení a tím včas varovat skutečného majitele účtu. [3]

1.2 Přístup uživatelů k autentizaci

Studiem chování uživatelů, jejich návyků a přístupu k zabezpečení nebo i analýzou uniklých přístupových údajů se zabývá nejedna studie. Podle webu `blog.dashlane.com`, jednoho ze správců hesel, vlastní průměrný uživatel 90 online účtů. Pro americké občany tento průměr dosahuje až 130 účtů. Článek byl zveřejněn v roce 2015 a s technologickým růstem nelze předpokládat snížení těchto statistik. [4]

Při tak velkém počtu různých účtů není překvapivé, že si lidé často hesla nepamatují, vymýšlejí slabá hesla, zapisují si je na veřejně dostupné lístečky nebo dokonce opakují stejná hesla pro více účtů. Poměr uživatelů opakujících stejná hesla pro více účtů činí až kolem 60 % [5]. U některých dotazníkových studií toto číslo dokonce přerůstá přes 70 % [6].

Úniky hesel analyzuje např. Troy Hunt na webu `haveibeenpwned.com`. Jeho služba poskytuje především bezplatné ověření, zda není účet uživatele evidován v nějakém úniku informací. V současnosti web obsahuje evidenci více než 7 700 000 000 uniklých účtů z více než 91 000 úniků [7]. Tyto úniky vedly k seznamu unikátních hesel přesahující počet 500 000 000 [8]. Ve srovnání s počtem lidí na internetu, který čítá přibližně 4 300 000 000 [9], to opravdu není málo.

V roce 2017 úniky přístupových údajů analyzovala také společnost Google, která se zaměřila na identifikaci a detekci jejich původu. Analyzovala přibližně 2 000 000 000 hesel a následně srovnala nejčastější zdroje uniklých údajů, mezi které patří phishing a malware. Studie zkoumala také cíle těchto útoků. Podle výsledků malware v podobě keyloggeru útočí hlavně na emailové účty, hesla, informace o zařízení a uživateli, zatímco phishingové stránky cílí na údaje kreditních karet a emaily, na hesla pak o něco méně. [10]

LastPass v roce 2018 analyzoval 43 000 svých klientských podniků. Do porovnávacích kritérií zahrnul např. průměrnou délku hesel, počet slabých hesel, počet opakujících se hesel, průměrnou sílu hesla, sílu sdílených hesel nebo

použití vícefaktorové autentizace. Výsledky srovnávají bezpečnostní úroveň podniků podle velikosti i odvětví. Nejlepších výsledků průměrně dosahují organizace z bankovních a technických odvětví. Také vedou v použití vícefaktorové autentizace, kterou ostatní odvětví využívají jen zřídka. [11]

Všechny průzkumy dochází ke stejnému závěru. Lidé si nedokážou zapamatovat bezpečná hesla. S každou další registrací si musí zapamatovat další komplikovaný údaj. Proto při registraci volí velmi podobná hesla jako do jiné služby nebo použijí slabé a lehce předvídatelné heslo. Většina uživatelů si ani neuvědomí rizika spojená s použitím stejných či obdobných přístupových údajů pro více účtů, protože se domnívají, že nemohou být cílem útoku. [12]

Ideální řešení této problematiky nabízí tzv. správci hesel. Tyto programy bezpečně uchovávají všechny přístupové údaje ke všem uloženým uživatelským účtům v zašifrované databázi. K opětovnému přístupu k nim stačí uživateli jedno silné heslo, označované jako hlavní (master password). Použití takové aplikace umožňuje volit dostatečně silná a unikátní hesla pro každou službu bez nutnosti jejich pamatování. Hlavní heslo by však mělo být dostatečně silné. V případě prolomení hlavního hesla útočník získává přístup do všech uložených účtů. U lokálních správců musí útočník navíc získat šifrovanou databázi z počítače uživatele. Opatrní uživatelé, kteří nechtějí uložit všechny své přístupové údaje pod jedno hlavní heslo, mohou rozdělit účty do skupin např. podle jejich důležitosti nebo oblasti využití. Z každé skupiny poté vytvoří samostatné databáze, ovšem nezbytně s použitím zcela odlišných a dostatečně silných hlavních hesel.

1.3 Doporučené požadavky na heslo

Aktuální standardy Evropské agentury pro bezpečnost sítí a informací (ENISA) [13] a Národního institutu standardů a technologií (NIST) [14] doporučují tyto parametry hesel:

- **Udržení v tajnosti** – heslo není sdílené tajemství a jeho prozrazení i věrohodné osobě zvyšuje riziko zneužití.
- **Používání delších hesel namísto komplexních** – krátké heslo, obsahující znaky ze všech skupin, trvá prolomit výrazně kratší čas než dlouhé heslo jen se znaky malé abecedy.
- **Unikátnost** – Opakování hesel pro více účtů vede k velkému riziku v případě úniku jednoho z nich. Rovněž se doporučuje vyvarovat často používaným a obecně známým heslům.
- **Spojení s uživatelem** – Heslo **nemá obsahovat** jakékoli informace uživateli blízké (např. datum narození, jméno mazlíčka, zájmy…). Tyto informace se snadno dohledávají a výrazně zvyšují pravděpodobnost kompromitace účtu.

- **Správce hesel a generovaná hesla** – Správci hesel umožňují vygenerovat dlouhá náhodná hesla nesouvisející s uživatelem. Následné uložení a používání nevyžaduje nutnost pamatování.
- **Passphrase** – Bezpečnější a přitom překvapivě snazší na zapamatování jsou tzv. passphrase (heslovité fráze o několika náhodných slovech). Ideálně využitelné jako hlavní heslo k přístupu do správce hesel.
- **Druhý faktor** – Kombinace faktorů velmi komplikuje útočníkovi situaci a tím výrazně snižuje riziko kompromitace.

1.4 Existující správci hesel

V dnešní době lze na trhu nalést poměrně velké množství programů, které nabízejí správu přístupových údajů. Každý z nich obsahuje různé množství funkcionalit a bezpečnostních prvků. Mezi nejimplementovanější funkce patří kromě bezpečného šifrování např. generování náhodných hesel, automatické vyplňování webových formulářů nebo možnost synchronizace databáze.

1.4.1 Lokální správci hesel

Lokální správci hesel ukládají data do jednoho či více souborů na disk v počítači uživatele. Z toho vyplývá jejich obecná výhoda: odolnost proti phishingu a jiným online útokům. Nevýhody může představovat horší možnost zálohování databáze, integrace s webovým prohlížečem a podpora vyplňování formulářů, synchronizace databáze mezi více zařízení (obvykle se využívá další služba). V neposlední řadě existuje také riziko při napadení počítače malwarem.

KeePass

KeePass patří mezi tradiční lokální opensource správce hesel. Na trhu působí již od roku 2003 a nabízí i pokročilé funkce. Data ukládá do jediného souboru a ve výchozím nastavení vše šifruje šifrou AES s délkou klíče 256 bitů. Šifrovací klíč odvozuje z hlavního hesla (a případného přístupového souboru) pomocí hashovací funkce SHA-256 a opakovaného aplikování funkcí AES-KDF (popř. Argon2) spolu s náhodně generovanou solí. [15, 16]

Mezi neobvyklou funkcionalitu, kterou nenajdeme u jiných správců hesel, patří ochrana proti keyloggerům. Ochrana spočívá v použití systémové funkce *Secure Desktop* pro dialogové okno hlavního hesla. Tento režim chrání před ostatními procesy mimo toto prostředí. [17] Další část ochrany simuluje falešné stisky kláves. Pro ostatní procesy by tato simulace měla být k nerozeznání od kláves skutečné klávesnice. Poslední část ochrany mlží kopírování do schránky. *KeePass* při kopírování do schránky náhodně rozděluje kopírovaný vstup na

několik prokládaných částí. Ty postupně kopíruje do schánky a přeskakuje již zkopírované znaky vkládáním stisků levé a pravé šipky. Tato ochrana není sice úplně 100%, ale rozhodně není zbytečná. Bohužel funguje pouze pod systémem Windows a ve výchozím nastavení navíc není aktivovaná (podle výrobce kvůli možným potížím s kompatibilitou). [18]

Pass

Pass, unixový zástupce čistě lokálního správce hesel pod opensource licencí GPLv2+, dodržuje princip jednoduchosti návrhu KISS a filozofii Unixu. Každý záznam představuje jeden textový soubor obsahující heslo i jakékoli další části záznamu. *Pass* šifruje tyto soubory samostatně pomocí *GnuPG* a veřejného klíče. Úspěšné dešifrování umožňuje pouze soukromý klíč uživatele. Ovládání programu probíhá z příkazové řádky, ale také k němu lze nalézt i řadu GUI rozšíření od jiných autorů, jak tomu v opensource komunitách bývá. V kombinaci se systémem git umožní verzování nebo i zálohu a synchronizaci v případě vzdáleného repozitáře. [19]

1.4.2 Cloudoví správci hesel

Použití cloudového správce hesel přináší uživatelům mnoho výhod. Poskytovatel služby může automaticky obstarávat zálohy a případně i jejich verzování. Všichni uživatelé jistě ocení pohodlnou synchronizaci mezi více zařízeními nebo technicky snazší a bezpečnější automatické vyplňování formulářů. S tím se přirozeně spojuje i několik nevýhod. Použití služby nutně vyžaduje přístup k internetu. Protože většinou ukládáme hesla k online účtům, není to velké omezení. Avšak dostupnost z internetu výrazně zvyšuje riziko útoku. Buď na servery poskytovatele za účelem odcizení databáze a následného prolamování šifrování, nebo i jen za účelem způsobit nedostupnost služby. Cílem útoku mohou být také samotní uživatelé. Útočník se může pokusit vylákat z uživatelů hlavní heslo např. pomocí phishingu.

LastPass

LastPass používá přibližně 13,5 milionů lidí a 43 000 firem [20]. Zakládá si na dobré bezpečnosti a velkém množství funkcí jako automatické vyplňování formulářů, generování náhodných hesel, audit hesel, vícefaktorová autentizace, ochrana automatického vyplnování a přihlášení v případě podezření na phishing podle shody URL adresy s uloženou adresou... Program nabízí několik placených variant pro jednotlivce i společnosti. Tomu také odpovídají cenové nabídky rozšiřujících funkcí, jako např. integrace s Active Directory, sdílení hesel, vynucení různých bezpečnostních politik pro firemní hesla nebo souhrné reporty. K dispozici je i bezplatná varianta s omezeným množstvím funkcionalit. [21]

Šifrovací klíč *LastPass* odvozuje z uživatelova hlavního hesla pomocí funkce PBKDF2-SHA256, kterou opakuje nejméně 100 000 krát. Samotné šifrování probíhá pomocí šifry AES-256. K autentizaci uživatele používá dále hashovaný šifrovací klíč odvozený z hlavního hesla. Všechna data by měla být přenášena šifrovaně a k dešifrování by mělo dojít až u koncového uživatele. [22]

1Password

1Password používá velmi podobný princip zabezpečení jako *LastPass* a poskytuje obdobnou sadu funkcí. Navíc umožňuje hlavní heslo zkombinovat s klíčem o délce 128 bitů, který má k dispozici pouze vlastník v podobě souboru. Tento klíč přidává značnou míru entropie k hlavnímu heslu. Šifrovací klíč *1Password* odvozuje z této dvojce pomocí funkce PBKDF2 a data šifruje taktéž funkcí AES-256, stejně jako *LastPass*. I když se jedná o uzavřený software, zveřejňuje podrobnou technickou dokumentaci zaměřenou na zabezpečení aplikace. [23]

1.4.3 Ostatní správci hesel

Mimo výše zmíněných lze nalézt mnoho dalších obdobných aplikací. Většina z nich používá stejný či velmi podobný systém zabezpečení a nabízí obdobné funkcionality v různých cenových variantách. Z neobvyklých funkcionalit stojí za zmínku podpora autentizace otiskem prstu na mobilních telefonech s příslušným hardwarem u správce *StickyPassword* [24] nebo monitorování a upozornění na bezpečnostní úniky u správce *Dashlane* [25].

V neposlední řadě tu jsou funkce pro pamatování hesel integrované přímo do webových prohlížečů jako Google Chrome, Mozilla Firefox, Microsoft internet Explorer. Jedná se o vedlejší funkci, u které bohužel není příliš dbáno na bezpečnost. Hesla ukládají do lokálních databází (např. SQLite), avšak často nepoužívají bezpečné šifrování nebo nepoužívají žádné. [26]

Pokud šifrování použijí, šifrovací klíč odvozují buď z přihlášení uživatele pomocí Windows API, ke kterému má případný malware přístup také, nebo šifrovací klíč uloží do souboru, který je pro malware opět čitelný. Mozilla Firefox nabízí volbu hlavního hesla, bohužel ve výchozím nastavení není vynucovaná. Šifrovací klíč odvozuje pomocí funkce SHA-1, která pro účely spojené s bezpečností již není považovaná za dostatečnou. [26, 27]

Záměrně tyto funkce pro uložení hesel ve webových prohlížečích neoznačuji za správce hesel, protože nesplňují potřebnou úroveň zabezpečení, což je hlavní účel správců hesel. Také většinou neposkytují potřebné funkcionality správce hesel jako generování náhodných silných hesel.

Pro svou bezpečnostní analýzu jsem si vybral správce hesel *Passbolt*. Jeho představením a detailním popisem se zabývá nasledující kapitola.

Kapitola 2

Představení správce hesel Passbolt

Passbolt je opensource správce hesel založený na architektuře client-server a zaměřený na týmovou spolupráci. Umožňuje pohodlné sdílení hesel s dalšími uživateli nebo skupinami uživatelů, proto ho ocení především středně velké společnosti, pro které je primárně určen. [28, 29]

První verze se na trhu objevila v roce 2016 a ve srovnání s ostatními správci hesel patří mezi nováčky. To potvrzuje také menší množství funkcí, které obsahuje. Ačkoli zatím neumí tolik jako jiní správci hesel, funkcionality časem přibývají a již nyní se dá plnohodnotně a pohodlně použít. Cenové varianty produktu se liší hlavně rozsahem podpory ze strany výrobce a několika prémiovými funkcemi [30, 31, 32], např.

- importy a exporty hesel (ve formátech csv a kdbx)
- synchronizace uživatelů s LDAP serverem nebo s Active Directory
- tmavý design doplňku prohlížeče
- vícefaktorová autentizace pomocí HW klíče Yubikey nebo aplikace Duo

Pro bezpečnostní analýzu byla vybrána bezplatná Community edice ve verzi v2.7.1 a tyto dvě klíčové části:

passbolt_api – Serverová část psaná v jazyce PHP.

passbolt_browser_extension – Klientská část v podobě doplňku do prohlížeče v jazyce JavaScript.

Mimo tyto základní části *Passbolt* nabízí také volitelné **passbolt_cli** jako rozhraní příkazové řádky pro serverovou část nebo **passbolt_docker** pro spuštění v prostředí dockeru. [33]

Pro ukládání dat *Passbolt* podporuje dva podobné druhy databází, *Mysql* a *MariaDB*. Z bezpečnostního pohledu jsou důležité tabulky obsahující citlivé údaje jako jména, hesla a údaje o uživateli. Tyto údaje jsou zde myšleny v souvislosti s účelem správce hesel, nikoli pro přihlášení uživatele do aplikace. Jedná se o tabulky:

users – Identifikuje uživatele.

gpgkeys – Uchovává veřejné klíče každého uživatele.

secrets – Obsahuje šifrovaná hesla vytvořená uživatelem spolu s vazbou na `id_uživatele` a tabulku `resources`.

resources – Obsahuje zbylá data záznamu vytvořeného uživatelem (tj. název, přihlašovací jméno, URL adresa, popis). Tyto informace nejsou nijak šifrované.

permission – Slouží pro uložení práv ke každému záznamu tabulky `resources`. V případě sdílení reguluje práva uživatelů díky vazbě na tabulku `users`.

2.1 Odlišnosti od ostatních správců hesel

Každý správce hesel se vždy více či méně odlišuje od ostatních. *Passbolt* se od většiny správců hesel liší hned v několika ohledech.

První zásadní odlišnost spočívá ve volbě druhu kryptografie. Na rozdíl od většiny správců hesel, kteří používají symetrickou šifru AES a šifrovací klíč odvozují od hlavního hesla účtu, *Passbolt* pro účely autentizace, šifrování i sdílení používá principy asymetrické kryptografie (tedy veřejné a soukromé klíče). [28] Celý princip staví na faktu, že soukromý klíč může znát pouze vlastník a nikdy by neměl být sdílen nebo posílán přes síť další osobě. A to ani na server aplikace *Passbolt*! Samotné soukromé klíče se navíc šifrují symetrickou šifrou, pro případ odcizení. Passphrase, ze které se odvozuje klíč pro dešifrování soukromého klíče, poté slouží jako hlavní heslo do aplikace. *Passbolt* není jediný, kdo využívá pro ukládání hesel asymetrickou kryptografii. Dalším zástupcem je výše zmíněný lokální unixový správce hesel *Pass*.

Použití asymetrické kryptografie usnadňuje zaměření aplikace *Passbolt*, tedy sdílení hesel v týmu, a také se snáze realizuje, než při použití čistě symetrického šifrování. Server jen zprostředkuje veřejné klíče uživatelů, se kterými chceme heslo sdílet. Zašifrované heslo se poté odešle a uloží v databázi serveru. Při požadavku na dané heslo server odešle záznam z databáze a dešifrování proběhne opět na počítači koncového uživatele. Hlavní heslo ani soukromý klíč by proto nikdy neměly být odeslány, a to ani šifrovaně. [34]

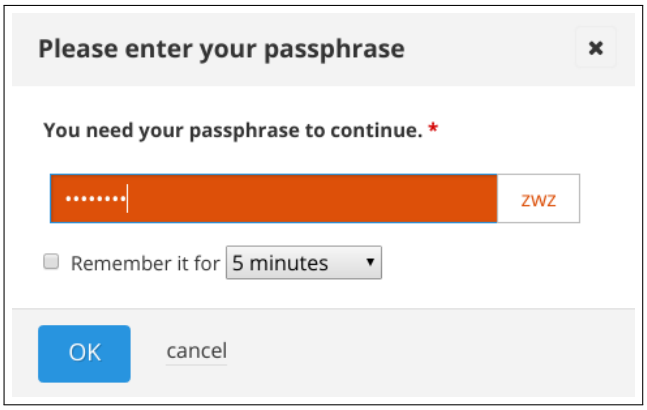
Druhou podstatnou odlišnost proti cloudovým správcům tvoří otevřenost kódu a hlavně možnost vlastního hostingu. To ocení právě společnosti, které si umístí svůj *Passbolt* server do vlastní sítě a omezí k němu přístupy z venčí. Tím

přidají další vrstvu zabezpečení, kterou musí potenciální útočník překonat. Pro zaměstnance může být stále dostupný i odjinud prostřednictvím VPN. Menší týmy bez vlastních serverů mohou využít hosting od výrobce. [32, 35]

Další neobvyklou odlišnost představuje možnost vlastního rozšiřování pomocí REST JSON API. Bohužel dokumentace API není k verzi v2.7.1 zatím dokončená, a tak použití není zrovna nejsnadnější. [28]

Poslední zmíněnou odlišnost reprezentuje způsob autentizace. *Passbolt* nepoužívá klasické ověření jménem a heslem, protože pro uživatele by to znamenalo další heslo navíc k zapamatování (kromě hesla k soukromému klíči). Místo toho využívá *GPGAuth* protokol, který zakládá na obousměrné výměně šifrovaných, elektronicky podepsaných náhodně generovaných tokenů pomocí veřejných a soukromých klíčů serveru a uživatele. Toto schéma detailně popisuje následující sekce 2.2. [36]

Zajímavou nadstavbou bezpečnosti autentizace tvoří tzv. bezpečnostní barva a kód, které se zobrazí při každém zadávání hlavního hesla. Tato ochrana má zřetelně upozorňovat uživatele na falešnou stránku resp. dialog hlavního hesla. Podvodné stránky nemohou znát správnou barvu a kód, protože tyto údaje uživatel volí při inicializaci doplňku prohlížeče, ukládají se pouze lokálně a platí po celou dobu až do resetu doplňku prohlížeče. Ukázku zobrazuje obrázek 2.1. [37]



Obrázek 2.1: Dialog s bezpečnostní barvou a kódem

2.2 Autentizační protokoly

Následující sekce popisuje dva možné protokoly využívané při elektronickém ověření identity. Také srovnává jejich výhody, nevýhody a potenciální bezpečnostní rizika.

2.2.1 Ověření hesla proti databázi

Jedná se o webový formulář, ve kterém uživatel vyplní své přihlašovací jméno a heslo. Údaje se odešlou na server, v databázi se vyhlédá záznam se jménem, odeslané heslo se poté zahashuje s příslušnou solí a pokud se výsledek shoduje s uloženým záznamem, ověření proběhlo úspěšně. Server vygeneruje unikátní session token, uloží ho a odešle klientovi v podobě cookie. Tím je klient přihlášen až do konce platnosti session tokenu. Pokud se zahashované heslo neshoduje s uloženým v databázi, služba odešle chybovou hlášku „Nesprávné uživatelské jméno nebo heslo“. Přihlášené uživatele server identifikuje právě podle platných session tokenů, které jsou součastí každého zaslaného požadavku. Pokud se blíží konec platnosti tokenu, služba může vygenerovat nový, nebo počká na konec platnosti tokenu a uživatel se musí znovu ověřit zadáním hesla. [36] Způsob autentizace proti databázi znázorňuje schéma 2.2.

Tento způsob ověření lze jednoduše implementovat a klade malé nároky na uživatele. Navzdory tomu obsahuje řadu potenciálních bezpečnostních i technických problémů.

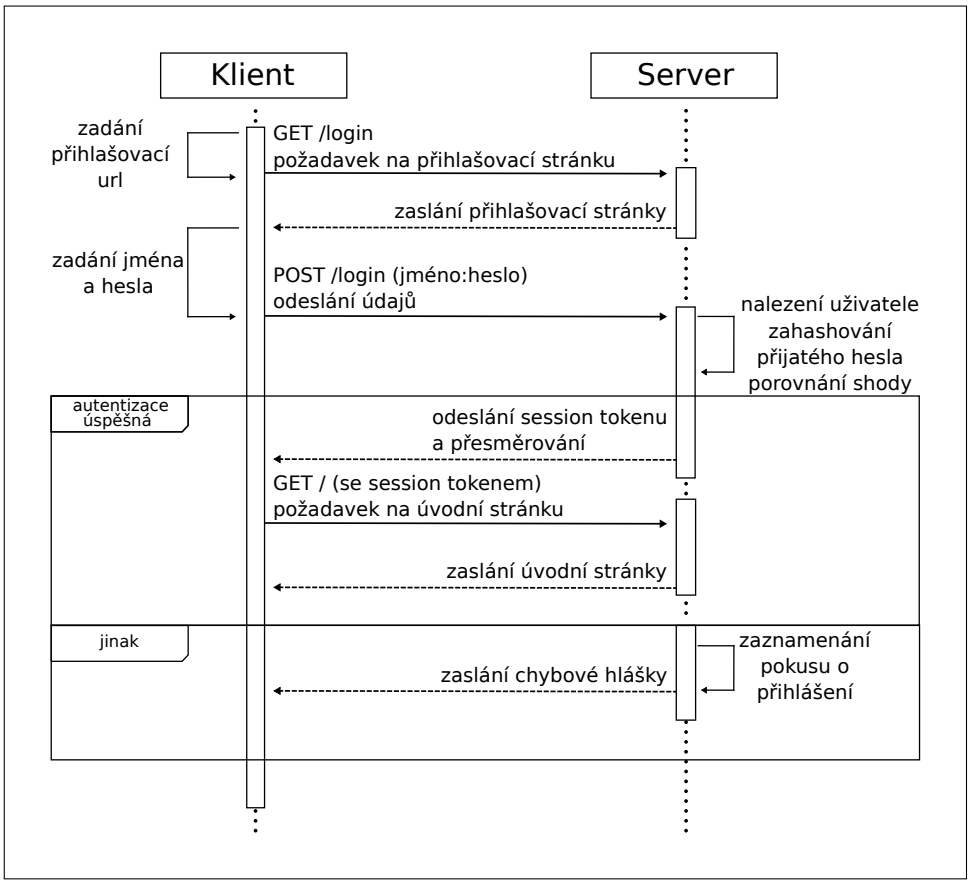
Za prvé uživatel posílá své přihlašovací údaje přes internet na server služby. I když bude komunikace mezi klientem a serverem šifrovaná, není odolná proti phishingovým útokům popř. útokům typu man in the middle. Phishing má za cíl zmatení uživatele vytvořením vzhledově totožné stránky. Pokud uživatel na takovou stránku přijde a pokusí se přihlásit, jeho údaje se odešlou na server útočníka, místo serveru služby, ke které se chtěl přihlásit. Tím útočník získá přihlašovací údaje, i když bude komunikace šifrovaná. [36]

Pro větší nenápadnost může útočník následně simulovat skutečnou službu a údaje uživatele jí předat za účelem získání session tokenu, který přepošle zpět původnímu uživateli. Ten poté nemusí mít ani tušení, že prozradil své heslo někomu jinému. Ochranou před tímto útokem je použití certifikátu na straně služby a jeho kontrola na straně uživatele. Bohužel mnoho uživatelů se v této problematice nevyzná a nepovažují ji za důležitou. Útočníci navíc volí velmi lehce zaměnitelné URL adresy, aby lidé na první podhled nepojali podezření, že se přihlašují k nesprávném serveru.

Další problém nastává při ukládání hesla resp. hashe hesla na serveru služby. V případě úniku databáze ze serveru útočník získává velké množství zahashovaých hesel. Z hashe není možné zpětně odvodit původní řetězec díky jednosměrnosti hashovací funkce, ale útočník stále může zkoušet hashovat různá hesla a porovnávat shodu s uniklou databází. Protože uživatelé mnohdy

volí slabá a lehce předvídatelná hesla, často je útočník uhodne. K prolamování lze použít také dopředu předpočítané hashe (tzv. Rainbow tables). Použití dostatečně dlouhé a náhodné soli zvětšuje množství dat v rainbow tabulce pro každý záznam a současné kapacitní omezení disků brání v uložení takového množství dat. [38]

V neposlední řadě narazíme na problém s resetem hesla. Podle empirických výsledků uživatelé hesla přirozeně zapomínají a často vyžadovaná komplexní hesla zapomínají ještě častěji [39]. Dle mého názoru by lidem vadilo, kdyby jim byl odepřen přístup ke službě, byť oprávněný, v případě zapomenutí hesla. Proto služby musí implementovat postup na obnovu zapomenutých hesel, nejčastěji dočasným odkazem na webový formulář zaslaným na email. Z toho vyplývá, že pokud se útočník dostane k takto zaslanému odkazu v emailu, který může dokonce sám iniciovat, získá plný přístup ke službě. Pokud navíc má přístup do emailové schránky uživatele, umožní mu přístup ke všem službám registrovaných na tento email.



Obrázek 2.2: Schéma ověření proti databázi [40]

2.2.2 GPGAuth protokol

Protokol *GPGAuth* si klade za cíl odstranit nedostatky autentizace pomocí hesla. Funguje na základě oboustranné výměny náhodně generovaného tokenu zašifrovaného a podepsaného pomocí asymetrické kryptografie. [36]

V prvním kroku se autentizuje server služby. Klient zahajuje komunikaci zasláním svého otisku klíče spolu s náhodnou zprávou, kterou zašifruje veřejným klíčem serveru. Server zprávu dešifruje a pošle zpět. Klient ověří, zda se přijatá zpráva shoduje s dříve vygenerovanou. Tato první část *GPGAuth* protokolu není povinná, přesto doporučovaná. Ověřuje, jestli uživatel komunikuje se stejným serverem jako obvykle. Slabinou může být možný man in the middle útok. Tedy v případě narušení SSL/TLS komunikace bude útočník schopen číst obsahy packetů a autentizace serveru přeto proběhne úspěšně. Doplněk do prohlížeče aplikace *Passbolt* proto striktně kontroluje URL adresu (včetně protokolu). Z toho důvodu nejde jednoduše narušit SSL/TLS komunikace vzhledově stejným webem s podobnou URL adresou. [36]

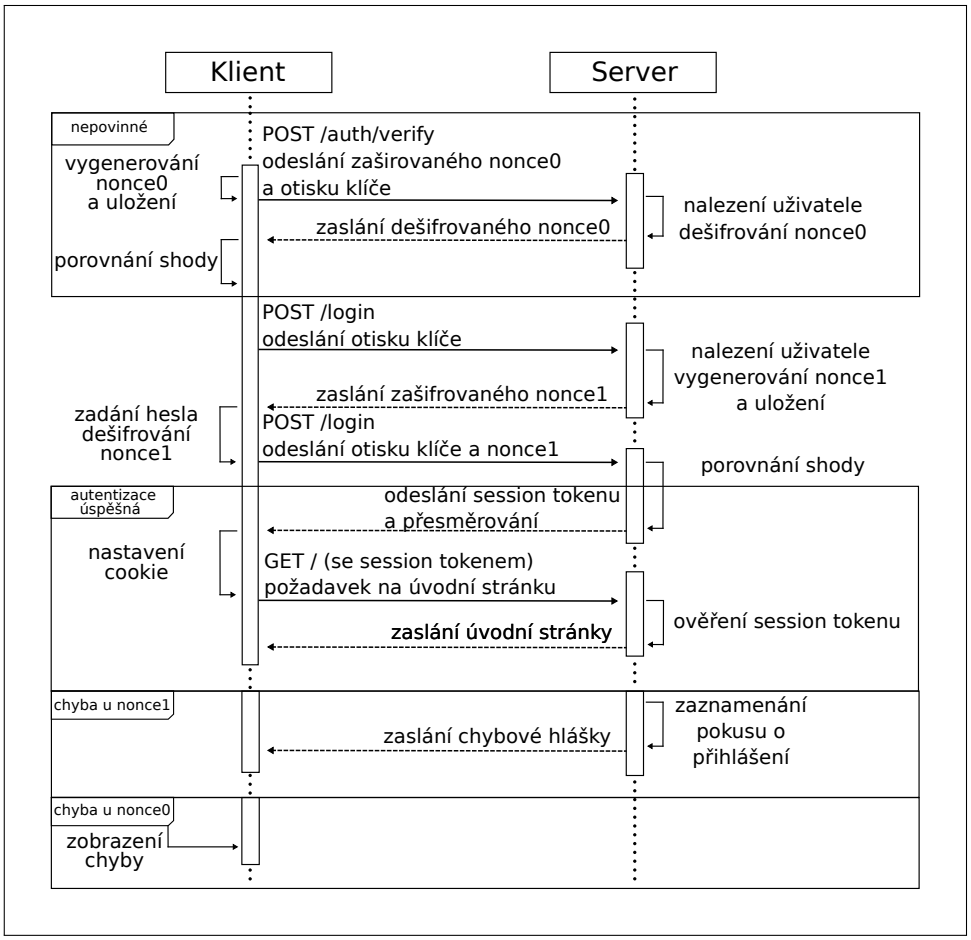
Samotné přihlášení zahajuje klient zasláním požadavku na přihlašovací stránku spolu s otiskem svého klíče. Služba podle otisku klíče ověří, zda uživatel existuje, a vygeneruje náhodný token. Tento token si uloží v čitelné podobě a poté ho zašifruje pomocí veřejného klíče uživatele uloženého v databázi. Kvůli autentičnosti ho navíc podepíše svým soukromým klíčem a odešle zpět. Uživatel zadá heslo ke svému soukromému klíči pro dešifrování doručeného tokenu. Po dešifrování a ověření podpisu veřejným klíčem serveru klient odešle dešifrovaný token zpět spolu s otiskem svého klíče. Služba nakonec porovná dešifrovaný token s dříve uloženým. Pokud se shodují, vygeneruje session token, odešle ho klientovi a uživatel je přihlášen. V případě neshody odešle chybovou hlášku. [36] Schéma protokolu lze vidět na obrázku 2.3.

Největší výhodou tohoto protokolu je odolnost proti phishingu. Právě phishing se zdá být v současnosti významný problém [41]. Mnoho uživatelů se nevyzná v oblasti bezpečnosti a nemají důvod nevěřit stránkám, které vypadají identicky s originálem. U ostatních cloudových správců hesel se ochrana řeší pomocí certifikátů, ty *Passbolt* používá samozřejmě také. Pokud útočník získá certifikát od některé důvěryhodné certifikační autority, která má svůj certifikát automaticky zařazený do prohlížečů, a zvolí podobnou URL adresu jako adresa služby, uživatel to s vysokou pravděpodobností ani nezaregistruje. Další vrstvou ochrany proti phishingu tvoří výše zmíněná bezpečnostní barva a kód.

Hlavní heslo uživatele není nikdy posíláno přes internet, stejně tak jako soukromý klíč. Hesla dokonce ani nemusí být uložena na straně serveru v žádné podobě. A i v případě, že útočník vytvoří falešný web a vyláká z uživatele hlavní heslo, bez jeho soukromého klíče nemá žádnou hodnotu. Tento princip osobně považuji za velkou výhodu, ale nese s sebou také riziko zapomenutí hesla. Tím, že soukromý klíč i heslo vlastní pouze uživatel, server ho nikdy nezná. V případě zapomenutí hesla či ztráty soukromého klíče účet nepůjde

obnovit. Nutnost pečlivého uschování soukromého klíče a hlavního hesla se stává plnou zodpovědností uživatele, který by měl mít např. externí zálohu pro případ poruchy disku či celého počítače. Velké riziko představuje také odcizení soukromého klíče. V případě prolomení hesla k soukromému klíči útočník může dešifrovat všechna data uživatele. V současnosti není implementovaný způsob pro revokaci soukromého klíče, proto by jediným řešením bylo smazání účtu spolu se všemi daty ze strany administrátora. Zálohu šifrovaných dat před smazáním účtu lze sice provést, nicméně identita uživatele by měla být ověřena nekompromitovanou cestou, např. fyzicky v případě firemního hostingu.

Další výhodu tvoří síla generovaného tokenu. Náhodné dostatečně dlouhé tokeny jsou pro útočníka nereálné na uhodnutí. Navíc síla šifrovaného hesla nijak nesouvisí se složitostí hlavního hesla. Když uživatel zvolí slabé heslo ke svému soukromému klíči, neovlivní to bezpečnost jeho uložených hesel ani v případě úniku dat ze serveru. Přesto se nedoporučuje používat slabá hlavní hesla např. pro případ úniku soukromého klíč uživatele z jeho počítače. [36]



Obrázek 2.3: Schéma GPGAuth protokolu [42]

Kapitola 3

Zvolený postup analýzy

Provedená bezpečnostní analýza programu *Passbolt* se skládá ze tří na sebe navazujících částí. V první části, analýze uživatelského rozhraní, se zkoumá aplikace z pohledu uživatele běžným použitím. Má za úkol nalézt taková místa, která hrají klíčovou roli v zabezpečení aplikace, nejsou v souladu s principy správné bezpečnosti, představují možné bezpečnostní riziko nebo nejsou plně srozumitelná pro uživatele z pohledu bezpečnosti a mohla by vést k použití slabšího zabezpečení. Nalezená podezřelá místa jsou označena kódem `Z1-Zn` a evidována pro hlubší analýzu ve zdrojovém kódu.

Druhá fáze bezpečnostní analýzy se zabývá monitorováním a následným průzkumem síťového provozu. Zde je kladen důraz na kontrolu dodržení postupů popsaných v dokumentaci a na kontrolu dat posílaných přes síť. Pro posouzení zranitelností a možného rizika se vybraná místa označují kódem `N1-Nn`.

V poslední části analýzy se zkoumá fungování aplikace ve zdrojovém kódu. Velkou část tvoří analýza podezřelých míst vybraných v analýze uživatelského rozhraní. Tato místa jsou zkoumána především ve zdrojovém kódu doplňku do prohlížeče, protože právě ten zajišťuje všechny kryptografické operace nad uživatelskými daty. Kód externích knihoven již není součástí analýzy, nicméně do analýzy patří ověření, jaké šifrovací knihovny aplikace používá a zda neobsahují hlášené zranitelnosti. Nalezené chyby a zranitelnosti se evidují pod označením `V1-Vn`. Toto označení může být v odůvodněných případech přiděleno jakékoli situaci ve všech částech analýzy a zhodnocení rizika nemusí nutně předcházet analýza ve zdrojovém kódu.

Závěr analýzy hodnotí velikosti dopadů, posouzuje možná rizika a navrhuje případná řešení nalezených zranitelností a chyb. Závěr rovněž diskutuje výsledky analýzy s autory aplikace.

3.1 Testovací prostředí

Pro účely testování byl použit virtualizační nástroj VirtualBox. V něm byly vytvořeny dva virtuální stroje. Jeden s operačním systémem Debian 9, kde byl nainstalován server aplikace Passbolt ve verzi v2.7.1.

Klientskou stranu tvořil druhý virtuální stroj se systémem Lubuntu 18.10 a instalovým doplňkem do prohlížeče. Aplikace *Passbolt* v současné době podporuje prohlížeče Chrome a Firefox. Oba mají k dispozici verze pro více operačních systémů. Virtuální stroje byly umísteny do společné, úplně oddělené virtuální sítě.

Síťová analýza byla realizována použitím nástroje pro pokročilou analýzu komunikačních protokolů a síťového provozu Wireshark. Ten umožňuje zachytávání paketů na zvolených síťových kartách v reálném čase, jejich ukládání a následný průzkum zachyceného provozu. Nabízí funkcionality pro analýzu provozu jako filtrování paketů podle mnoha faktorů, spojování rámců, dohledání paketů s požadavkem/odpovědí... Podporuje také mnoho protokolů, které umí dekódovat, se správným klíčem popř. dešifrovat a následně data exportovat (např. HTML soubory, obrázky...). [43]

Kapitola 4

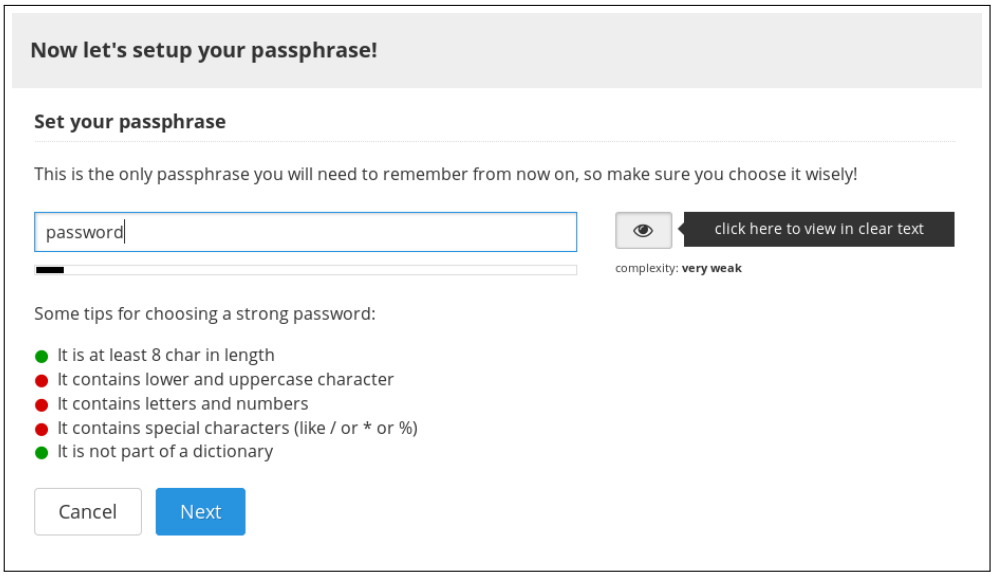
Analýza uživatelského rozhraní

Motivace uživatelů pro používání správců hesel spočívá především v zjednodušení pamatování hesel a zajištění jejich bezpečnosti. Očekává se, že tyto aplikace nebudou vyžadovat pokročilé znalosti v oboru a budou jednoduché na použití. Kvůli předpokladu cílové skupiny uživatelů různých technických znalostí a dovedností musí být nezbytně zajištěno dostatečné zabezpečení dat již ve výchozím nastavení, protože mnoho uživatelů použije právě výchozí nastavení.

4.1 Inicializace a obnova doplňku prohlížeče

Použití aplikace *Passbolt* začíná nainstalováním a nastavením doplňku prohlížeče. Poté, co administrátor vytvoří uživateli účet, server mu odešle emailem unikátní odkaz pro nastavení jeho doplňku prohlížeče a propojení se serverem. Tento proces slouží ke vzájemné výměně veřejných klíčů. Aby se předešlo útokům podvodných webů, aplikace vyzvývá ke kontrole identity serveru. Při každém pokusu o přihlášení doplněk prohlížeče kontroluje identitu protistrany pomocí těchto údajů.

Následuje vygenerování soukromého a veřejného klíče. V současné verzi nelze měnit parametry generovaného RSA klíče, který tak má fixní délku nastavenou na 2048 bitů. U zadávání hlavního hesla se zobrazuje informativní měřič entropie a obsaženost různých skupin znaků (velkých a malých znaků, číslic a speciálních symbolů). Tyto údaje mají pouze informativní charakter a lze zadat jakékoli heslo s alespoň 8 znaky. To vyhovuje doporučení NIST 800-63B z roku 2017 [14]. Heslo také prochází porovnáním proti databázi uniklých hesel prostřednictvím API stránky `https://api.pwnedpasswords.com/range`. Najde-li se shoda, opět se jedná o čistě informativní údaj, který na sebe bohužel nijak neupozorňuje a lze ho snadno přehlédnout. Standart NIST slovníkovou kontrolu plně doporučuje, ovšem v případě nalezené shody by aplikace měla informovat uživatele a nechat ho zvolit jiné heslo [14].



Obrázek 4.1: Zavádějící slovníková kontrola

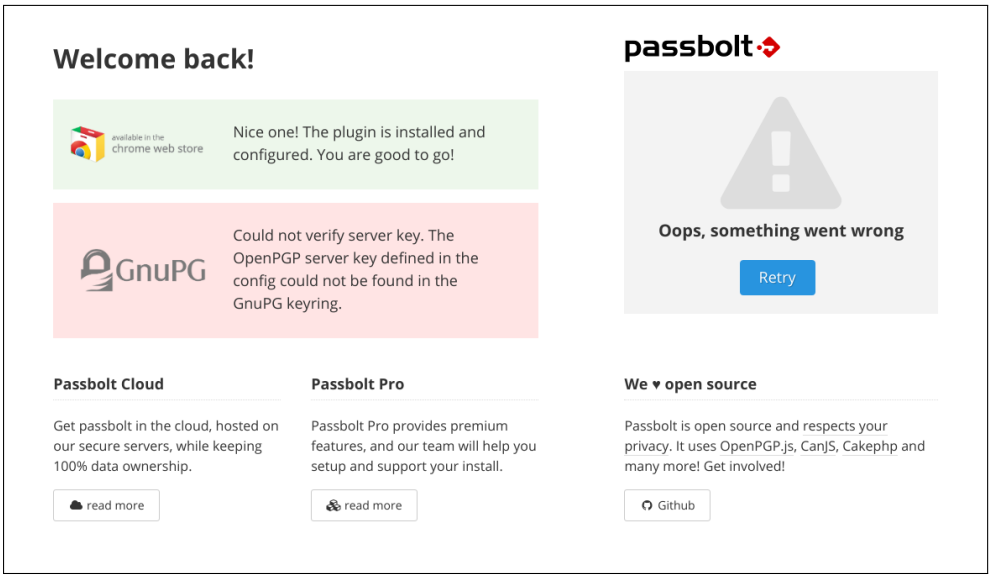
Nepředpokládané chování bylo objeveno při nedostupnosti sítě. V takové situaci aplikace každé zadané heslo označí jako vyhovující a zobrazí uživateli informaci, že dané heslo s úspěchem prošlo slovníkovou kontrolou. Takové chování může v uživateli vyvolat falešný pocit bezpečnosti jeho hesla.

Vytváření hlavního hesla a soukromého klíče hraje klíčovou roli v zabezpečení správců hesel, proto je místu přidělen kód `Z1` pro důkladnější analýzu ve zdrojovém kódu.

Alternativně může uživatel importovat vlastní soukromý klíč. Aplikace provede kontrolu, zda klíč není někým používán. Pokud ano, nedovolí ho použít. Touto cestou importování lze předat aplikaci i delší klíč než 2048 bitů. Ale také i kratší, který může být pro bezpečné šifrování nedostačující. Dokonce lze importovat nešifrovaný klíč a hlavním heslem se poté stává prázdný řetězec. Prověření validace importovaného klíče ve zdrojovém kódu probíhá pod označením `Z2`.

Poslední část inicializace tvoří volba libovolné barvy a třímístného kódu. Smysl ochrany vysvětluje kapitola „Představení správce hesel Passbolt“ v posledním odstavci sekce 2.1. Nakonec aplikace seznámí uživatele s důležitostí soukromého klíče a vyzve ho k vytvoření zálohy klíče.

V případě přeinstalování prohlížeče, ztráty či poruchy počítače lze využít možnost obnovy. Obnovovací proces se shoduje s inicializačním až na část generování klíče. Tam se vyžaduje import původního soukromého klíče. Bez zálohy klíče nelze účet obnovit, ani dešifrovat uložená data.



Obrázek 4.2: Kontrola identity serveru

4.2 Autentizace

Autentizační proces ověřuje identitu serveru a v případě neshody s nastavením vzniklým při inicializaci doplňku zobrazí uživateli výraznou hlášku a nezobrazí přihlašovací formulář (viz obrázek 4.2). Ověření serveru, první část *GPGAuth* protokolu, není odolná proti útokům typu man in the middle. Výrobce na stránkách uvádí, že o problému ví [36] a tuto ochranu zajišťuje pouze SSL/TLS certifikát stejně jako u ostatních cloudových správců hesel. Autentizace je zajímavá především ze síťového pohledu bezpečnosti, a proto se jí věnuje větší pozornost v další kapitole.

4.3 Vytvoření nového hesla k uložení

Vytvoření nového záznamu provází jednoduchý formulář, ve kterém se také zobrazuje bezpečnostní barva a kód. Význam této ochrany zde není proti odposlechu hlavního hesla, ale aby útočník nemohl podvrhnout tuto část stránky a získat tak nově vytvořené heslo uživatele.

Na první pohled ve formuláři chybí pole pro zopakování hesla, nicméně uživatel má možnost odmaskovat heslo z vyplněných teček do čitelné podoby, aby si mohl ověřit jeho správnost. Aplikace také zobrazuje přibližnou sílu hesla stejným způsobem jako u vytváření hlavního hesla. Protože se jedná o správce hesel, předpokládá se, že uživatel často využije možnost generovat náhodné silné heslo. Tlačítko pro generování se nachází vedle možnosti odmaskování, ale není zde k dispozici žádné nastavení generátoru hesel, především

délka generovaného hesla. Toto nastavení nebylo nalezeno ani v jiných částech aplikace. Generování hesel představuje důležitou funkci správců hesel, a proto je místu přidělen kód `Z3` pro prověření v kódu.

Samotné šifrování není z uživatelského pohledu příliš zajímavé. Funguje tak, jak by každý mohl očekávat, a nedává žádný prostor pro uživatelskou interakci. Mohou nastat situace, kdy uživatel potřebuje bezpečně uložit informaci, která nesplňuje požadavky na bezpečné heslo. Z toho důvodu formulář nevyžaduje minimální délku ani jiné požadavky. Jakou šifrovací knihovnu program používá ověřuje začátek analýzy kódu.

4.4 Dešifrování a změna hesla

Dešifrování vyvolává dialogové okno pro zadání hlavního hesla. Opět se zobrazuje bezpečnostní barva a kód jako proti phishingové opatření. Okno nabízí také zapamatování hesla na určitou dobu, nejdéle však do odhlášení. Místu se přiděluje označení `Z4` pro identifikaci způsobu a místa, kam se dočasně hlavní heslo ukládá. Nebo jestli se ukládá dešifrovaný soukromý klíč uživatele.

Dešifrovací proces spouští také úprava hesla. Ostatní nešifrované údaje záznamu lze měnit i bez zadání hesla. Přepsání resp. změně uloženého hesla předchází jeho dešifrování popř. zobrazení. To vyžaduje zadání hlavního hesla. I když by bylo technologicky možné provést změnu i bez znalosti hesla, tento způsob poskytuje určitou ochranu před neoprávněnou změnou šifrovaných údajů v případě přístupu k účtu a neznalosti hlavního hesla.

Po úspěšném dešifrování se heslo zkopíruje do schránky. Během používání aplikace bylo zjištěno, že heslo ve schránce přetrvá i značnou dobu po jeho použití a není aplikací odstraňováno. Jelikož se schránka sdílí mezi všemi uživatelskými procesy a eventuelně k ní mají přístup i další weby, je místu přidělěn kód `V1` pro zhodnocení rizika.

4.5 Sdílení hesel

Příjemným prvkem podporujícím týmovou spolupráci při sdílení hesla je napovídání uživatelů aktivních v systému. Po potvrzení a zadání hlavního hesla do dialogového okna s bezpečnostní barvou a kódem se následně heslo zašifruje veřejnými klíči všech vybraných uživatelů. Aplikace nabízí 3 různé varianty práv sdílení.

can read – S tímto oprávněním lze pouze dešifrovat a používat heslo.

can update – Toto právo umožňuje číst, měnit nebo i smazat daný záznam. Smazaní sdíleného hesla se projeví u všech uživatelů včetně vlastníka.

is owner – Toto právo značí vlastníka hesla, kterých může být i více. Oproti právu *can update* dovoluje navíc přidávat další uživatele do okruhu sdílení a měnit jim práva.

Každé heslo musí mít alespoň jednoho vlastníka. Aplikace nedovolí změnit práva hesla pokud by zmizel poslední vlastník. Také kontroluje, zda uživatel není jediným vlastníkem sdíleného hesla v případě odstraňování uživatele. V takové situaci vynutí převod vlastnictví na vybraného uživatele, se kterým se heslo rovněž sdílí.

4.6 Odstraňování hesel

Odstraňování probíhá bez zadání hlavního hesla a bez použití soukromého klíče. Vymazáním záznamu z databáze nemůže dojít k prozrazení tajemství uživatele a dodatečné ověření tedy není třeba. Sdílené heslo může smazat kdokoli s právy *can update* nebo *is owner*.

4.7 Emailové zprávy

Při každém vytvoření či editaci hesla uživatel obdrží informační email. Tento email se odesílá také jako upozornění uživatele na nové sdílené heslo a obsahuje údaje záznamu spolu se zašifrovaným heslem daným uživatelským klíčem. Způsob šifrování byl v pořádku ověřen dešifrováním jiným programem s příslušným soukromým klíčem. Emailové zprávy tak mohou sloužit nejen jako upozornění, ale rovněž jako jistá forma zálohy.

Kapitola 5

Analýza síťového provozu

Zachycení a analýza síťového provozu byla provedena pomocí nástoje Wireshark. Kvůli použitému SSL/TLS šifrování celé komunikace bylo pro provedení analýzy nezbytné nastavit prohlížeč tak, aby ukládal použité šifrovací klíče do souboru, který se následně využije pro dešifrování. Tato volba se nastavuje proměnou prostředí `SSLKEYLOGFILE` a následným spuštěním prohlížeče ve stejném terminálu.

Analýza zkoumala především části komunikace týkající se dat uživatele určených k bezpečnému uložení, sdílení těchto dat s jinými uživateli, inicializace doplňku prohlížeče a autentizačního protokolu aplikace. Také bylo ověřeno, zda se pro šifrování používá standard *OpenPGP*, uváděný v dokumentaci. Ověření probíhalo extrahováním šifrovaných dat z komunikace a dešifrováním externím nástrojem *GnuPG* s použitím příslušného soukromého klíče uživatele.

Komunikace s *Passbolt* serverem probíhá ve formátu JSON a využívá tyto HTTP metody:

GET – pro jednoduché čtecí požadavky

POST – pro požadavky na server obsahující data (např. autentizace)

PUT – pro ukládání nových hesel, změny hesel a sdílení hesel

DELETE – pro mazání hesel z databáze

Zachycená komunikace odpovídá výše zdokumentovanému *GPGAuth* protokolu. Pro ověření serveru se využívá veřejný klíč serveru uložený lokálně od inicializace doplňku. Správné nastavení doplňku při inicializaci je nezbytné pro bezpečnost, a proto se uživatel vyzývá k důsledné kontrole otisku klíče serveru. V dešifrované komunikaci rovněž nebyl nalezen žádný soukromý klíč ani hlavní heslo uživatele.

V případě vytváření či změny záznamu se přenáší šifrované heslo spolu s ostatními údaji záznamu jako pojmenování záznamu, přihlašovací jméno, URL adresa, popis... Všechny údaje kromě hesla již šifrované nejsou a do databáze jsou ukládány v čitelné podobě. Pokud útočník dokáže obejít SSL/TLS šifrování, může tyto informace číst. Nicméně větší riziko představuje únik databáze s těmito údaji a následná enumerace uživatelů a jejich používaných služeb. Vyhodnocení rizika je provedeno pod označením `N1`.

Sdílení hesel funguje obdobným principem. Klient nejdříve informuje server o sdílení hesla, následně zaktualizuje veřejné klíče uživatelů, zašifruje heslo klíči vybraných uživatelů a odešle na server. U sdílení hesel se také posílá důležitý záznam o právech uživatelů, se kterými sdílení probíhá. Všechna hesla, šifrovaná pomocí doplňku prohlížeče, bylo možné dešifrovat pomocí příslušného soukromého klíče jiným programem.

Kapitola 6

Analýza zdrojového kódu

Aplikace *Passbolt* využívá instalovaný doplněk prohlížeče kvůli potřebnému přístupu k bezpečným kryptografickým operacím a popř. dalším privilegovaným příkazům. Chování backendu doplňku odděluje návrhový vzor od javascriptového kódu, použitého na frontendu stránky. Frontend stránky zpracovává vstup, zachytává události a zobrazuje výstupy. Určitá forma jejich vzájemné komunikace je pro správné fungování nezbytná. Realizuje se pomocí daného rozhraní vysíláním signálů připomínajících službu.

Tato architektura ukrývá zajímavou bezpečnostní myšlenku. Doplněk běží v odděleném sandboxu a není schopen měnit kód stránky. Na druhou stranu frontend nemůže zasahovat do běhu doplňku, nemůže číst jeho proměnné ani s nimi jakkoli nakládat. [44]

Serverová část aplikace potřebuje šifrování pouze pro účely autentizace uživatelů. K tomu používá *GnuPG Php Extension* [45] a *singpolyma/openpgp-php* [46]. Šifrování komunikace pomocí SSL/TLS zajišťuje webový server (např. apache), na kterém aplikace běží. [34]

Základem šifrování v aplikaci *Passbolt* se stává doplněk prohlížeče. Proto se mu věnuje největší pozornost v průběhu celé analýzy. Doplněk pro všechny kryptografické účely využívá opensource knihovnu *OpenPGP.js* [47]. Knihovna prošla v roce 2014 kompletním bezpečnostním auditem německé společnosti Cure53, který odhalil řadu zranitelností, z nichž dvě byly klasifikovány jako závažné. V současnosti jsou známé pouze malé nedostatky či optimalizační návrhy [48].

6.1 Shrnutí potenciálně rizikových míst

Předchozí části analýzy objevily 2 místa určená přímo pro vyhodnocení rizika a 4 místa, která by mohla představovat bezpečnostní riziko. Nalezená místa jsou blíže zkoumána ve zdrojovém kódu aplikace v následujících sekcích. Přehled nalezených míst k bližší analýze v kódu lze nalézt v tabulce 6.1.

Označení	Popis
Z1	Vytváření hlavního hesla a soukromého klíče
Z2	Import soukromého klíče
Z3	Generování náhodného hesla
Z4	Dočasné uložení hlavního hesla

Tabulka 6.1: Místa určená k analýze zdrojového kódu

6.2 Z1 – Vytváření hlavního hesla a soukromého klíče

Prvním testovaným místem bylo vytváření hlavního hesla a generování dvojice klíčů. Obsluhu vytváření hlavního hesla lze nalézt v souboru `secret.js`, který má na starosti načtení zadaného hesla, kontrolu kritérií a zobrazení stavu jednotlivých kritérií. Po potvrzení předává zadané heslo doplňku prohlížeče k validaci a v dalším kroku i ke generování dvojice klíčů.

O zobrazení entropie hesla, obsažnost skupin znaků či slovníkový test se stará přímo javascript stránky bez využití asistence doplňku. K tomu slouží `secretComplexity.js` pro sílu hesla a `pwnedpasswords.js` pro slovníkovou kontrolu. Entropii hesla aplikace počítá analyzováním použitých skupin znaků, sečtením velikostí těchto skupin a následným výpočtem:

$$Entropie = delka\_hesla \cdot \frac{\ln(soucet\_velikosti\_skupin)}{\ln(2)}$$

Zobrazovaná síla hesla vznikne vyhledáním spočítané entropie v příslušném řádku tabulky 6.2.

Entropie	Síla hesla
0	nedostupné
>= 1	velmi slabé
>= 60	slabé
>= 80	slušné
>= 112	silné
>= 128	velmi silné

Tabulka 6.2: Tabulka síly hesla podle entropie

Slovníková kontrola se vyvolává při každé změně formulářového okna hesla událostí `step.onPasswordInput()` a aplikuje se pouze na hesla s délkou 8 a více znaků. Volaná funkce `step._checkPasswordPwnd()` se dále odkazuje na funkci `secretComplexity.ispwned()` a očekává vrácení hodnoty `true`, `false`, nebo vyhození výjimky v případě jiných potíží. Kód funkce lze vidět na obrázku 6.1.

```
step._checkPasswordPwnd = async function(password) {
  step.criterias['dictionary'] = undefined;
  var isPwnd;
  try {
    isPwnd = await secretComplexity.ispwned(password);
  } catch (error) {
    // something went wrong (like a network issue)
    // leave it undefined
    console.error(error.message);
    return;
  }
  if (typeof step.criterias['dictionary'] !== 'undefined') {
    // password was cleared in meantime, ignore this request
    return;
  }
  step.criterias['dictionary'] = !isPwnd;
  if (isPwnd) {
    step.strength = 1;
  }
  step._onUpdateAll(password);
};
```

Obrázek 6.1: Ukázka funkce step._checkPasswordPwnd()

Funkce `ispwned()` v souboru `secretComplexity.js`, zachycená obrázkem 6.2, volá funkci `pwnedpasswords()` ze souboru `pwnedpasswords.js`, která slovníkový test realizuje. Implementace slovníkového testu využívá `https://api.pwnedpasswords.com/range/` API. Dotaz obsahuje pouze prvních 5 znaků SHA-1 hashe a není tedy možné z něj heslo zpětně odvodit. Po vykonání dotazu se porovnají jednotlivé odpovědi se zbytkem hashe a případně je vrácen pozitivní nález ve formě počtu nalezených výskytů. Podle tohoto čísla funkce `ispwned()` vrátí `true`, `false`, nebo `undefined` v případě zachycení výjimky.

Problém nastává v případě nedostupnosti sítě. URL dotaz `fetch()` ve funkci `pwnedpasswords()` vyvolá výjimku, kterou okamžitě zachytí volající funkce `ispwned()` a ta vrátí hodnotu `undefined`. Výjimku vyvolávající část kódu funkce `pwnedpasswords()` zachycuje obrázek 6.3. S tím nepočítá původní funkce `step._checkPasswordPwnd()` a její obsluha výjimek (např. pro síťové potíže) se nikdy nevykoná, protože příslušná výjimka již byla zachycena dříve.

Významy hodnot `true` a `false` funkce `ispwned()` a zobrazovaného stavu slovníkové kontroly jsou zcela opačné. Z toho důvodu se pomocí logické negace ve volající funkci `step._checkPasswordPwnd()` z obrázku 6.1 otáčí význam vrácené hodnoty z funkce `ispwned()`. Jednou z vlastností jazyka javascript je

```
var ispwned = async function (password) {
   try {
      var count = await pwnedpasswords(password);
      return (count > 0);
    } catch(error) {
      console.error(error);
      return undefined;
    }
  };
```

Obrázek 6.2: Ukázka funkce ispwned()

```
const response = await fetch(url);
if (response.status === 404) {
  return false;
}
if (response.status !== 200) {
  return Promise.reject(new Error(
  `Failed to load pwnedpasswords API: ${response.status}`));
}
```

Obrázek 6.3: Ukázka části funkce pwnedpasswords() vykonávající URL dotaz

implicitní převoditelnost některých hodnot různých typů proměnných (např. `false` a `undefined`, `0` a `false`). Právě tento převod z `undefined` na `false` se uskuteční těsně před logickou negací hodnoty. Význam síťové chyby s hodnotou `undefined` se tímto implicitně převede na význam „nenalezeno v únicích hesel“ s hodnotou `false` a heslo se z hlediska slovníkové kontroly označí za vyhovující s hodnotou `true`. Stejné chování nastává při vrácení chyby 404 z volaného API. Vyhodnocení `return (count > 0)` ve funkci `ispwned()` s hodnotou `count == false` dopadne pomocí implicitní konverze z `false` na `0` hodnotou `false` značící stav „nenalezeno v únicích hesel“. A opět se prohodí na `true` a označí za vyhovující slovníkové kontrole, avšak bez skutečné kontroly. Místu je přidělen kód `V2` pro zvážení možného rizika a návrhu opravy.

6.3 Z2 – Import soukromého klíče

Proces importování klíče ovládá soubor `importKey.js`. Ve validující funkci `step.validatePrivateKey()`, zobrazené na obrázku 6.4, aplikace komunikuje s doplňkem prohlížeče a podle otisku soukromého klíče kontroluje, jestli zadaný soukromý klíč nepoužívá již někdo jiný. V případě obnovy naopak kontroluje shodu s nějakým existujícím účtem.

```
step.validatePrivateKey = function () {
    return new Promise(function(resolve, reject) {
      passbolt.request('passbolt.setup.checkKeyExistRemotely',
       step.data.privateKeyInfo.fingerprint)
        .then(function () {
          if (step.options.workflow == 'install') {
            reject('This key is already used by another user');
          }
          else if (step.options.workflow == 'recover') {
            resolve();
          }
        })
        .then(null, function () {
          if (step.options.workflow == 'install') {
            resolve();
          }
          else if (step.options.workflow == 'recover') {
            reject('This key doesn\'t match any account.');
          }
        });
    });
  };
```

Obrázek 6.4: Funkce validatePrivateKey() s chybějící validací bezpečnosti klíče

Funkce neobsahuje jinou než tuto kontrolu, proto aplikace dovolí importovat jakýkoli soukromý klíč včetně klíčů s délkou nedostatečnou pro bezpečné šifrování. Místu je přiděleno označení `V3` pro zhodnocení míry rizika. Dalším vektorem útoku může být použití nešifrovaného soukromého klíče a tím pádem prázdné hlavní heslo. V případě kompromitace počítače útočník získává přímo soukromý klíč uživatele. Protože není šifrovaný hlavním heslem, umožní mu přístup do aplikace a ke všem šifrovaným heslům uživatele. Riziko je vyhodnocováno pod označením `V4`.

6.4 Z3 – Generování náhodného hesla

Generování hesla vykonává funkce `generate()` ze `secretComplexity.js` a na vstupu předpokládá délku generovaného hesla a pole názvů skupin znaků, ze kterých se má heslo generovat. Tato funkce se volá z `edit.js` bez jakýchkoli parametrů, proto funkce použije výchozí hodnoty.

Výchozí hodnota délky je 18 znaků. Z jednoho pohledu se může zdát škoda, že aplikace neumožní uživatelům nastavit vetší délku hesla. Na druhou stranu

```
var generate = function (length, masks) {
  var secret = '',
    mask = [],
    masks = masks || ["alpha", "uppercase", "digit", "special"],
    length = length || 18,
    expectedEntropy = null;
  // Build the mask to use to generate a secret.
  for (var i in masks) {
    mask = $.merge(mask, MASKS[masks[i]].data);
  }
  // Generate a password which should fit the expected entropy.
  // Try maximum 10 times.
  var j = 0;
  do {
    secret = '';
    expectedEntropy = calculEntropy(length, mask.length);
    for (var i = 0; i < length; i++) {
      secret += mask[randomRange(0, mask.length - 1)];
    }
  } while (expectedEntropy != entropy(secret) && j++ < 10);
  return secret;
};
```

Obrázek 6.5: Funkce generate() vytvářející náhodná hesla

18 náhodných znaků plně dostačuje a uživatelé nemohou vygenerovat krátké heslo, které by nemuselo být dostatečně bezpečné. Výchozí množina pro generování hesla obsahuje všechny 4 skupiny znaků (malá abeceda, velká abeceda, číslice a speciální znaky) a dohromady čítá 94 znaků.

Před zahájením generování hesla aplikace sloučí vybrané skupiny znaků do jednoho pole, ze kterého následně opakovaně vybírá. Pro zajištění složitosti náhodného hesla se spočítá očekávaná entropie z počtu prvků sjednoceného pole a zvolené délky hesla. Po vygenerování hesla se spočítá jeho entropie stejným způsobem jako při vytváření hlavního hesla v sekci 6.2 a pokud nedosahuje předem spočítané očekávané entropie, proces generování se opakuje, nejvýše však 10 krát. Funkci `generate()` zachycuje obrázek 6.5.

Pro samotné generování náhodných znaků se využívá `Window.crypto` a metoda `getRandomValues()`. Vygenerovaná číselná hodnota se následně namapuje na jeden znak z rozsahu, ze kterého se generuje. Celé heslo postupně vznikne tímto způsobem znak po znaku.

Během analýzy bylo zjištěno, že podmínka, ve kterém se testuje entropie vygenerovaného hesla s očekávanou, není nikdy splněna. Dokonce ani při využití znaků ze všech skupin. Aplikace tedy vezme vždy až 10. vygenerované heslo v pořadí. Očekávaná entropie se počítá ze sjednoceného pole skupin znaků, kdežto entropie generovaného hesla se počítá z původních skupin znaků. Původní skupiny znaků mají svůj počet určený jako člověkem definovanou proměnou, na rozdíl od sjednoceného pole, kde se jeho délka počítá. Použití definovaných velikostí v obdobných situacích není nutné a zavádí riziko lidské chyby. Příčinou celého problému je dvojitý výskyt znaku dvojtečka („:“) ve skupině se speciálními znaky. Definovaná velikost této skupiny má hodnotu 32 znaků, ale její skutečná velikost je o jeden znak vyšší. Očekávaná entropie počítá s 95 možnými znaky, na rozdíl od entropie generovaného hesla, která má maximum 94 možných znaků. Proto při stejné délce hesla a využití všech skupin znaků generované heslo nikdy nemůže dosáhnout očekávané entropie.

Toto místo téměř nemá bezpečnostní dopad. Problém by mohl nastat jedině kdyby 10. heslo neobsahovalo znak od kadé skupiny a mělo nižší entropii. Přesto se jedná o chybu, kterou autoři evidentně neplánovali, a proto je označena kódem `V5` pro navržení její opravy. Protože očekávaná entropie nijak nezávisí na proměnných cyklu, vychází stále stejná hodnota a z optimalizačního hledista lze tento výpočet předřadit před cyklus.

6.5 Z4 – Dočasné uložení hlavního hesla

Způsob uložení hesla může mít velký dopad na jeho bezpečnost. Hesla uložená v nešifrovaném souboru mohou být jednoduše odcizena v případě kompromitace počítače. Zde se jedná o hlavní heslo a jeho šifrování by znamenalo nutnost zapamatování dalšího šifrovacího klíče. To ovšem není cílem u dočasného uložení.

Passbolt ukládá, při volbě dočasného uložení, hlavní heslo pouze do paměti a nejvýše do odhlášení, které je automaticky vynucené po 20 minutách neaktivity. Heslo není šifrované, protože by šifrovací klíč musel být opět někde uložen a znamenalo by to stejný problém. Pokud by útočník měl přístup k počítači uživatele, mohl by výpisem a zdlouhavým průzkumem paměti prolížeče heslo zjistit. Nicméně pokud by taková práva už měl, může heslo získat i mnohem jednodušším způsobem (např. odposlechem stisknutých kláves).

Další problém by mohl nastat v případě nesmazání hesla z paměti. To aplikace důsledně kontroluje a po uplynutí daného časového úseku heslo z paměti odstraní. Heslo mizí také v případě ukončení procesu prohlížeče s výjimkou prohlížeče Google Chrome. U něj zavření okna nezpůsobuje ukončení procesu prohlížeče, ale pouze jeho přesun na pozadí.

Z důvodu důsledné kontroly přetrvání hesla v paměti a nezbytné kompromitace počítače uživatele potřebné pro zneužitelnost není dočasné uložení hesla hodnoceno jako bezpečnostní zranitelnost.

Kapitola 7

Posouzení nalezených zranitelností a jejich řešení

Provedená bezpečnostní analýza aplikace nenašla závažnou zranitelnost přímo ohrožující bezpečnost aplikace nebo uživatelských dat. Nicméně odhalila řadu bezpečnostních nedostatků, které jistým způsobem za spolupráce uživatele mohou snížit zabezpečení nebo vést i k úplné kompromitaci účtu. Také byla nalezena jedna programová chyba neovlivňující bezpečnost. Přehled nalezených chyb a zranitelností shrnuje tabulka 7.1. Pro určení možných rizik jednotlivých zranitelností je používána níže popsaná metodika CVSS v3.

7.1 Common Vulnerability Scoring System

CVSS metodika pro klasifikaci zranitelností standardizuje měření jejich závažností, aby bylo možné porovnávat rizika jednotlivých zranitelností a prioritizovat opravy. Standardizace také umožňuje porování zranitelností napříč společnostmi. Cílem této metodiky je co nejobjektivnější posouzení závažnosti dané zranitelnosti a nepřebírání častých subjektivních pocitů vyhodnocovatele. [49]

Označení	**Popis**
N1	Nešifrované části záznamu
V1	Neodstraňování hesel ze schránky
V2	Zavádějící slovníková kontrola
V3	Chybějící validace délky importovaného klíče
V4	Umožnění prázdného hlavního hesla
V5	Porovnávání očekávané entropie

Tabulka 7.1: Nalezené chyby a zranitelnosti

Metodika se člení na jednu povinnou část, která určuje základ skóre. Další dvě části se využívají pro zpřesnění výsledků pomocí metrik prostředí přímo pro konkrétní společnost nebo pomocí časových metrik.

Základní část CVSS metodiky obsahuje 8 metrik s různými škálami hodnocení:

Attack Vector (AV) – Určuje cestu přístupu k zařízení potřebnou k využití zranitelnosti (síť, blízká síť, lokální přístup, fyzický přístup)

Attack Complexity (AC) – Představuje složitost útoku pro útočníka (nízká, vysoká)

Privileges Required (PR) – Říká, zdali útočník potřebuje nějaká práva v systému (žádná, nízká, vysoká)

User Interaction (UI) – Vymezuje míru nutné spolupráce uživatele (není nutná, nezbytná)

Scope (S) – Zkoumá rozsah a cíl dopadu v případě úspěšného útoku (pouze zranitelná aplikace, dopad na další službu)

Confidentiality (C) – Posuzuje únik neveřejných informací (žádný, nízký, vysoký)

Integrity (I) – Analyzuje možnost manipulace s daty (žádná, nízká, vysoká)

Availability (A) – Hodnotí dopad na dostupnost systému (žádný, nízký, vysoký)

Výsledné skóre je určeno podle vzorce v rozsahu od 0 do 10. Tabulka 7.2 interpretuje CVSS skóre podle závažnosti. [49]

CVSS skóre	**Míra závažnosti**
0.0	Žádná zranitelnost
0.1 - 3.9	Malá zranitelnost
4.0 - 6.9	Střední zranitelnost
7.0 - 8.9	Vysoká zranitelnost
9.0 - 10.0	Kritická zranitelnost

Tabulka 7.2: CVSS skóre

7.2 Posouzení nalezených zranitelností a jejich řešení

Tato sekce posuzuje všechny nalezené zranitelnosti z hlediska dopadu na bezpečnost a diskutuje jejich hodnocení metodikou CVSS. Nedílnou součástí je také následný návrh řešení daných chyb a zranitelností. Pro výpočet skóre byl použit online kalkulátor dostupný na adrese `https://www.first.org/cvss/calculator/3.0`.

Všechny posuzované zranitelnosti nutně vyžadují interakci s uživatelem. CVSS skóre částečně počítá s těmito variantami útoků, proto má metrika *User Interaction* vždy zvolenou hodnotu **nezbytná**. Nicméně metodice chybí širší stupnice vymezení míry potřebné spolupráce. Obdobný případ nastává u složitosti útoku. Proto výsledné hodnocení posuzovaných zranitelností může být v odůvodněných případech oběma směry částečně upraveno.

Nalezené zranitelnosti sdílí další společný rys, metriku dostupnosti služby. Žádná z chyb nijak neovlivňuje dostupnost serveru aplikace, ani při plném zneužití. Metrika *Availability* má proto hodnotu **žádný dopad**.

Situace `V2-V4`, kde se jedná o odcizení soukromého klíče, mají zranitelnou oblast přímo účet uživatele v aplikaci *Passbolt*. Proto je metrika *Scope* limitována pouze na **zranitelnou aplikaci**, i když vzhledem k povaze ukládaných dat lze uvažovat i o možnosti ovlivnění dalších služeb. Tato trojice různými způsoby cílí na soukromý klíč uživatele, a proto má také společnou míru dopadu. Soukromý klíč umožňuje přístup k účtu uživatele. Dešifrování všech jeho chráněných dat odpovídá **vysokému úniku** informací metriky *Confidentiality*. Rovněž možná změna všech těchto dat je hodnocena **vysokým** zásahem do metriky *Integrity*. Ovšem musím podotknout velkou náročnost využití těchto situací a značnou potřebu zmanipulování uživatele.

7.2.1 N1 – Nešifrované části záznamu

Analýza zkoumala rozsah šifrovaných dat a dat uložených v čitelné podobě. Aplikace šifruje pouze ukládaná hesla. Ostatní části záznamů se ukládají do databáze nešifrovaně. Tento koncept je součástí návrhu struktury aplikace. Odcizení nešifrovaných částí záznamů poskytuje informace, jaké účty a služby jednotliví uživatelé vlastní. Tyto informace neposkytují útočníkovi přístup k žádné z využívaných služeb, nicméně mohou být použity pro enumeraci používaných služeb jednotlivých uživatelů.

Při komunikaci se využívá SSL/TLS šifrování, a proto by útočník pro získání těchto dat musel získat přístup k použité databázi. K zamezení získání použitelných dat z databáze lze použít šifrování na úrovni celé databáze.

Z důvodu nemožné využitelnosti dat a obtížnosti jejich získání není místo hodnoceno jako bezpečnostní zranitelnost. I když stránky podpory výrobce na tento fakt upozorňují, nebylo by od věci informovat uživatele o rozsahu

šifrovaných dat, aby se zamezilo ukládání citlivých údajů do nešifrovaných částí záznamu (např. záložní přístupové kódy, bezpečnostní otázky...). Informování by mohlo probíhat např. při nastavování doplňku prohlížeče.

7.2.2 V1 – Neodstraňování hesel ze schránky

Při běžném používání aplikace bylo zjištěno, že po dešifrování hesla a jeho použití heslo zůstává ve sdílené schránce systému po neomezenou dobu. Heslo je tak v čitelné podobě uloženo v paměti až do přepsání jinými daty. Systémovou schránku sdílí všechny procesy a nepotřebné dlouhodobé vystavení citlivých dat zbytečně zvyšuje riziko odcizení.

Současné webové prohlížeče obsahují ochrany před neoprávněným čtením obsahu schránky. Tato práva mohou mít pouze doplňky prohlížečů a explicitně povolené stránky. Nicméně podvodná stránka stále může přimět uživatele k vložení dat ze chránky, k povolení přístupu ke schránce, nebo uživatel vloží obsah schránky omylem.

Určení CVSS skóre

Přístup ke sdílené schránce mají všechny běžící procesy systému včetně malwaru. Obsah schránky mohou získat také načtené webové stránky pomocí javascriptu, ať s využitím udělených práv nebo neopatrností uživatele. Kvůli této možnosti je zvoleno kritérium **síťový přístup**. Složitost vytvoření podvodné stránky manipulující uživatele není velká a zranitelnost nevyžaduje více potřebných kroků. Proto je náročnost hodnocena **nízkou složitostí** útoku.

Ze strany útočníka nejsou potřeba **žádná oprávnění** do aplikace *Passbolt*. Rovněž se jedná o často používanou situaci pro účel správce hesel např. ve strovnání s chybnou slovníkovou kontrolou, která nastává pouze při síťové chybě u vytváření hlavního hesla a vícekrát se neopakuje. Zato se úspěšný útok neobejde bez větší součinosti uživatele.

Cílem útoku není samotná aplikace *Passbolt* resp. účet uživatele v ní, ale přístupové heslo k jiné službě v aplikaci uložené. Tento model popisuje druhá varianta **jiná služba** metriky *Scope*.

Volbu dat útočník žádným způsobem neovlivňuje. Také není předem jasné, ke kterému účtu kompromitované heslo patří. Z tohoto důvodu je zvolena **nízká závažnost** úniku informací. Vzhledem k cíli útoku může být také možná modifikace dat v závislosti na službě, ke které bylo heslo odcizeno. Metrika *Integrity* je proto hodnocena **nízkou závažností**.

Výsledné CVSS skóre vychází na 6,1 a kategorii **střední zranitelnost**. Vzhledem k četnosti použití této základní funkcionality správce hesel není žádoucí finální závažnost zranitelnosti jakkoliv upravovat.

Návrh řešení

Správné vyčištění sdílené schránky je díky omezeným funkcím javascriptu a bezpečnostním ochranám prohlížeče velmi obtížné až nerealizovatelné. Po zkopírování do schránky aplikace nemá kontrolu, kam data uživatel vloží. A protože se události prohlížeče vyvolávají pouze v aktivní záložce, nelze poznat, zda už uživatel data vložil. Další komplikace nastávají při časovaném smazání. Prohlížeče blokují operace se schránkou nevyvolané událostí uživatele a smazání nepovolí. Navíc by bylo nepříjemné, kdyby byl ve schránce už jiný obsah a aplikace by ho smazala ještě před použitím.

V současné době není možné tuto situaci s použitou technologií rozumně řešit. Všechny aplikace, obsahující tuto funkci, nezbytně potřebují nainstalovaný samostatný program mimo prohlížeč. Lokální správci hesel toto kritérium splňují vždy a cloudové služby jako *LastPass* a *1Password* vyžadují nainstalovanou aplikaci přímo na počítači uživatele, jinak funkcionalita není dostupná.

Řešením může být obdobný přístup s využitím instalované aplikace. Mnohem užitečnější varianta by nevyužívala sdílenou schránku, ale implementovala by vyplňování formulářů. Tímto způsobem se lze vyhnout většině kopírování do schránky a navíc by usnadnila uživateli použití aplikace.

7.2.3 V2 – Zavádějící slovníková kontrola

Analýza kódu zjistila chybu ve funkci slovníkové kontroly hlavního hesla při nedostupnosti sítě nebo validačního serveru. Důsledkem chyby bylo přijato každé zadané heslo jako bezpečné. Výsledek kontroly nemá vliv na akceptaci hesla a aplikace zobrazuje přibližnou entropii. Přesto tato chyba může uživatele mylně utvrdit v bezpečnosti jeho hlavního hesla.

V případě odcizení soukromého klíče se slabým hlavním heslem útočník pravděpodobně prolomí ochranu pomocí slovníkového útoku. Pokud útočník získal klíč kompromitací počítače uživatele, má přístup i k nastavení doplňku prohlížeče (především k náhodnému id uživatele v rámci aplikace) a tím pádem se může přihlásit do účtu uživatele se stejnými právy jako on, včetně přístupu k zašifrovaným záznamům. Při odcizení soukromého klíče jiným způsobem lze dešifrovat data šifrovaná přidruženým veřejným klíčem. Na samotné přihlášení do aplikace to ovšem nestačí. Náhodné id uživatele může útočník získat také např. kompromitací emailového účtu uživatele registrovaného v aplikaci a následným vyvoláním obnovy doplňku prohlížeče, nebo jsou tyto informace dostupné všem registrovaným uživatelům.

Určení CVSS skóre

Přímočará metoda k získání dat z počítače uživatele je využitím malwaru. To odpovídá **lokálnímu přístupu** metriky *Attack Vector*, která pokrývá také uživatelem spuštěné viry. Další možností by byl např. fyzický přístup

```
var ispwned = async function (password) {
  try {
    var count = await pwnedpasswords(password);
    return (count > 0);
  }
};
```

Obrázek 7.1: Návrh opravy funkce ispwned()

k počítači. Díky ochranám operačních systémů a antivirů není jednoduché kompromitovat počítač uživatele, proto je zvolena **vysoká složitost** útoku.

Id uživatele, potřebné pro zcizení účtu aplikace, útočník získá s běžnými oprávněními daného uživatele systému nebo s právy jakéhokoliv uživatele aplikace *Passbolt*. Z toho vyplývá volba **nízkých oprávnění** metriky *Privileges Required*.

S použitím společných metrik pro `V2-V4` skóre ukazuje na středně závažnou zranitelnost především z důvodu velkého dopadu. Složitost útoku zahrnuje jak odcizení soukromého klíče, tak i zajištění nefunkčnosti komunikace se serverem slovníkové kontroly s potřebným načasováním v době vytváření klíče. I kdyby tyto okolnosti byly splněny, aplikace správně zobrazuje ukazatel entropie a volba slabého hesla není tedy zajištěna. Z těchto důvodů je situace `V2` vyhodnocena jako **malá zranitelnost**.

Návrh řešení

Chybu s nedostupností sítě a neočekávanou hodnotou `undefined` lze řešit předřazením jednoduché podmínky před zpracování samotné hodnoty. Tento způsob ovšem zavádí do kódu duplicitu zpracování chyby a navíc nebude fungovat v případě 404 chyb.

Správné řešení zahrnuje odstranění předčasného zachytávání výjimky ve funkci `ispwned()` z obrázku 6.2 a také neukončování funkce `pwnedpasswords()` z obrázku 6.3 hodnotou `false`, která při serverové chybě 404 způsobuje úspěch slovníkové kontroly. Navržená oprava částí kódu lze vidět na obrázkách 7.1 a 7.2. Chyby sítě se tím sjednotí pod již existující zpracování volající funkce z obrázku 6.1, které dosud nebylo vykonávané.

7.2.4 V3 – Chybějící validace délky importovaného klíče

Během vytváření účtu *Passbolt* umožňuje použít uživatelem generovaný soukromý klíč. Aplikace sice načte a zobrazí informace o importovaném klíči, ale chybí jakákoliv kontrola délky klíče. Použití slabého soukromého klíče představuje riziko neoprávněného přístupu k šifrovaným heslům. Bezpečná

```
const response = await fetch(url);
if (response.status !== 200) {
  return Promise.reject(new Error(
  `Failed to load pwnedpasswords API: ${response.status}`));
}
```

Obrázek 7.2: Návrh opravy části funkce pwnedpasswords()

délka klíče se s rostoucí výpočetní silou v čase mění. Aplikace by proto měla varovat před použitím krátkého klíče vedoucího ke slabému šifrování.

Se základním přístupem k aplikaci může každý uživatel získat jakýkoli veřejný klíč. V případě použití slabého klíče lze úspěšně faktorizovat RSA modul $n = p \cdot q$, tak získat dvojici prvočísel (p, q) a zrekonstruovat soukromý klíč. Se soukromým klíčem může útočník kompromitovat účet i bez nutnosti napadení počítače uživatele.

Určení CVSS skóre

Pro získání veřejného klíče není potřeba přístup k počítači uživatele. Lze ho získat přímo z aplikace *Passbolt*. Z tohoto důvodu metrika *Attack Vector* odpovídá hodnotě **síťový přístup**. S tím se pojí také metrika *Privileges Required* a potřeba **nízkých oprávnění** pro získání veřejného klíče a id uživatele.

I když se koncept útoku může zdát jednoduchý, potřebná faktorizace představuje výpočetně i časově velmi náročný problém. Její náročnost určuje právě délka použitého klíče. Vzhledem k náročnosti výpočtu faktorizace byla zvolena **vysoká** složitost útoku.

Samotný útok probíhá bez interakce s uživatelem. Nicméně slabý klíč musí uživatel sám vložit při vytváření účtu. Proto je metrika *User Interaction* vyhodnocena jako **nezbytná**. Kompromitace soukromého klíče uživatele může mít velký dopad na přístup k šifrovaným datům. Tento rys sdílí situace `V2-V4`.

CVSS kritéria odpovídají středně závažné zranitelnosti se skórem 6,4 kvůli dopadu a možnosti síťového přístupu k potřebným informacím pro realizaci útoku. S přihlédnutím k vysoké náročnosti faktorizace a jedinému výskytu zdroje problému při vytváření účtu, navíc pouze ve variantě importu vlastního klíče, je finální hodnocení sníženo na hranici malé zranitelnosti. Stále však spadá mezi **středně závažné** zranitelnosti.

Návrh řešení

Aplikace obsahuje validační funkci `validatePrivateKey()`, která kontroluje pouze existenci klíče podle jeho otisku. Navržené řešení zobrazuje obrázek 7.3, kde byla přidána podmínka kontroly délky klíče v části inicializace účtu. Následný pokus vložení krátkého klíče lze vidět na obrázku

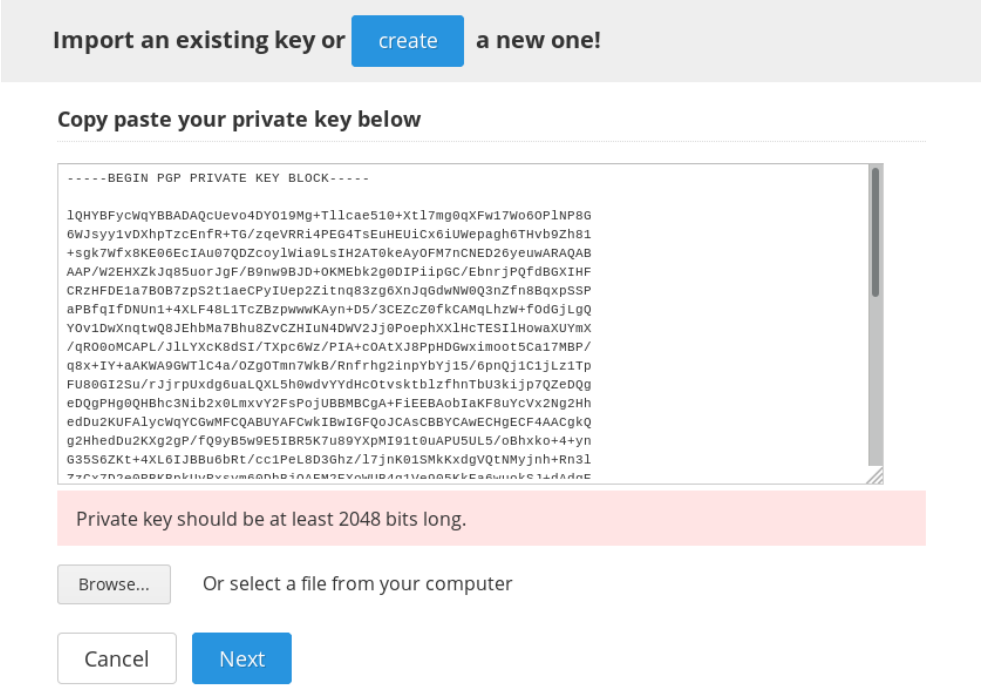
```
step.validatePrivateKey = function () {
  return new Promise(function(resolve, reject) {
    passbolt.request('passbolt.setup.checkKeyExistRemotely',
     step.data.privateKeyInfo.fingerprint)
      .then(function () {
        if (step.options.workflow == 'install') {
          reject('This key is already used by another user');
        }
        else if (step.options.workflow == 'recover') {
          resolve();
        }
      })
      .then(null, function () {
        if (step.options.workflow == 'install') {
        if (step.data.privateKeyInfo.length >= 2048) {
          resolve();
        }
        else {
          reject('Private key should be at least 2048 bits long.');
        }
        }
        else if (step.options.workflow == 'recover') {
          reject('This key doesn\'t match any account.');
        }
      });
  });
};
```

Obrázek 7.3: Návrh přidání podmínky kontrolující délku klíče

7.4. Tento kód se vykonává přímo na stránce a může být za běhu upraven. Pro úplné zamezení by bylo potřeba přidat funkci s touto podmínkou do kódu doplňku a ověřovat pomocí `passbolt.request()` obdobně jako u funkce `checkKeyExistRemotely()`.

7.2.5 V4 – Umožnění prázdného hlavního hesla

Při inicializaci doplňku aplikace umožnila importovat uživatelem vytvořený nešifrovaný soukromý klíč a tím používat prázdné hlavní heslo. Klíč se ukládá v souborech prohlížeče s právy uživatele a v případě kompromitace počítače, může být jednoduše odcizen.



Obrázek 7.4: Ukázka použití kontroly délky klíče

Chybějící ochrana šifrováním klíče způsobí přímý přístup útočníka ke klíči a jeho možné využití nejen k dešifrování všech dat zašifrovaných veřejným klíčem uživatele, ale také získání plného přístupu do účtu aplikace *Passbolt*. Při odcizení klíče způsobem bez přístupu k nastavení doplňku prohlížeče je pro přístup k účtu potřeba také id uživatele, které ovšem mají k dispozici všichni registrovaní uživatelé.

Určení CVSS skóre

Situace vychází ze stejného místa jako zranitelnost `V3`, ale předpoklady zneužití se liší. Podobá se výše popsané situaci `V2` (zavádějící slovníkové kontrole). Dokonce má identické metriky CVSS skóre vycházející na 6,0 a kategorii střední zranitelnosti především díky potenciálně velkému dopadu.

Protože účet v aplikaci nechrání hlavní heslo ale soukromý klíč dostupný pouze uživateli, nejedná se o přímé riziko. I když umožnění prázdného hesla v aplikaci typu správce hesel rozhodně nevyvolává bezpečný dojem. Kvůli velkým předpokladům útoku je i zde, podobně jako u `V2`, finální hodnocení zranitelnosti částečně sníženo. Ve srovnání s `V2` ovšem není potřeba dále dešifrovat odcizený klíč a lze ho přímo použít. Závažnost zranitelnosti je sice snížena od výsledného skóre, ale kvůli jednodušímu útoku stále spadá do kategorie **střední zranitelnost**, na rozdíl od situace `V2`.

```
Keyring.prototype.checkPassphrase = async function (passphrase) {
  const privateKey = this.findPrivate();
  const privKeyObj = (await
   openpgp.key.readArmored(privateKey.key)).keys[0];
  if (!privKeyObj.isDecrypted()) {
    await privKeyObj.decrypt(passphrase);
  }
};
```

Obrázek 7.5: Ukázka kontroly existence šifrování soukromého klíče

Návrh řešení

I když problém vychází ze stejných podmínek jako situace `V3`, jeho spolehlivé a bezpečné řešení není úplně realizovatelné. Aplikace sice může kontrolovat existenci hesla u soukromého klíče. Tato kontrola se dokonce provádí při použití klíče ve funkci doplňku `checkPassphrase()` na obrázku 7.5. Ale kontrola nijak nemůže zaručit sílu použitého hesla. Pro tento účel by uživatel musel současně s klíčem zadat také hlavní heslo. Ovšem vyžadování hesla, když není nezbytně potřeba, není dobrým zvykem a může snížit důvěryhodnost aplikace.

7.2.6 V5 – Porovnávání očekávané entropie

Analýza náhodného generování hesla objevila chybu při porovnávání entropie vygenerovaných hesel s očekávanou entropií. Nejedná se o bezpečnostní chybu a proto není určena její závažnost.

Pro opravení postačí vymazat duplicitní dvojtečku z proměnné `MASKS`. Kvůli zamezení opakování podobných chyb by bylo vhodné neurčovat velikost pole pomocí člověkem definované hodnoty ale využít vlastnost `pole.length`.

Z optimalizačního hlediska lze předřadit výpočet očekávané entropie před generující cyklus, protože na něm nijak nezávisí. Stejný výpočet se tak nemusí opakovaně provádět v každém průchodu cyklem.

7.3 Shrnutí nalezených zranitelností

Během provedené bezpečnostní analýzy bylo nalezeno a zhodnoceno 6 potenciálně rizikových situací. Dvě z nich nebyly vyhodnoceny jako bezpečnostní riziko, ale jako součást návrhu aplikace nebo programové chyby bez dopadu na bezpečnost. Zbylé čtyři situace byly zhodnoceny s využitím metodiky CVSS. Souhrné zhodnocení ukazuje tabulka 7.3.

Označení	Popis	Závažnost
N1	Nešifrované části záznamu	žádná
V1	Neodstraňování hesel ze schránky	střední
V2	Zavádějící slovníková kontrola	malá
V3	Chybějící validace délky importovaného klíče	střední
V4	Umožnění prázdného hlavního hesla	střední
V5	Porovnávání očekávané entropie	žádná

Tabulka 7.3: Přehled závažnosti nalezených zranitelností

7.4 Zpětná vazba autorů aplikace

K závěru praktické části práce proběhlo nahlášení nalezených chyb tvůrcům aplikace *Passbolt*. Komunikace probíhala emailem a byla velmi příjemná a pohotová. Autoři dokonce projevili i zájem o finální verzi práce v češtině se zajištěním vlastního překladu.

Programové chyby v kódu ověřili a naplánovali jejich opravu do nejbližšího sprintu[1]. U problému chybějící validace importovaného soukromého klíče, ať nedostatečné délky nebo nepřítomného šifrování klíče, uvádí, že považují za svobodnou volbu uživatele jak silný klíč se rozhodne používat a nekontrolování klíče tak považují za součást návrhu aplikace. Ale zároveň uznávají, že by měli uživatele upozornit v případě importu nedostatečného klíče a vysvětlit rizika s tím spojená. Také zmiňují svůj záměr předělat inicializační proces, aby se stal více uživatelsky přívětivý a opravu zmiňovaného nedostatku do něj začlení.

K začátku dubna zvěřejnili autoři dosud chybějící funkcionalitu automatického vyplňování webových formulářů a rychlý přístup k uloženým heslům. Tato funkce velmi usnadňuje použití aplikace a snižuje také riziko při kopírování do schránky.

[1]Sprint je opakující se časový úsek agilní metodiky *SCRUM* (obvykle 2–4 týdny), během něhož se realizuje konkrétní naplánovaná práce. Výsledkem sprintu je nová verze aplikace.[50]

Závěr

Cílem bakalářské práce bylo provedení bezpečnostní analýzy aplikace pro správu přístupových údajů. Teoretická část sloužila k zasazení práce do kontextu bezpečného nakládání s hesly a seznámení čtenáře s potřebou správců hesel spolu s porovnáním několika nejznámějších zástupců a jejich rozdílů.

Dílčím cílem praktické části bylo nalezení takových vektorů útoku, které by umožnily ohrozit bezpečnost účtu uživatele nebo ukládaných dat. Nalezená místa sloužila jako podklad pro hlubší analýzu ve zdrojovém kódu. Součástí analýzy bylo také monitorování a následný průzkum síťového provozu. Závěrem analýza vyhodnocuje možná bezpečnostní rizika nalezených zranitelností, hodnotí potenciální dopady a navrhuje případná opatření.

Analýza neodhalila žádnou závažnou zranitelnost přímo vedoucí k narušení bezpečnosti či integrity dat. Avšak bylo odhaleno několik nedostatků, které s nezbytnou interakcí uživatele mohou snížit úroveň zabezpečení nebo dokonce přispět ke kompromitaci účtu. Oprava většiny odhalených chyb není implementačně náročná a lze je jednoduše realizovat pomocí navržených úprav. Tyto návrhy zahrnují také opravu jedné chyby nesouvisející s bezpečností.

V budoucnosti by bylo možné na tuto práci navázat rozsáhlejší analýzou serverové části, analýzou dalších komponent aplikace (např. verze pro docker kontejner, CLI rozhraní), testováním prémiových funkcí nebo bližším prozkoumáním použitých externích knihoven.

Seznam použité literatury

1. TURNER, Dawn M. *Digital Authentication* [online]. Cryptomathic A/S, 2016 [cit. 2019-04-17]. Dostupné z: https://www.cryptomathic.com/news-events/blog/digital-authentication-the-basics.
2. ŠPAČEK, Michal. *Kam zmizel druhý faktor ze SMS zpráv* [online]. 2017 [cit. 2019-04-17]. Dostupné z: https://www.michalspacek.cz/kam-zmizel-druhy-faktor-ze-sms-zprav.
3. DIAS, Renan. *The 5 Factors of Authentication* [online]. A Medium Corporation, 2017 [cit. 2019-04-17]. Dostupné z: https://medium.com/@renansdias/the-5-factors-of-authentication-bcb79d354c13.
4. BRAS, Tom Le. *Online Overload – It's Worse Than You Thought* [online]. Dashlane Inc., 2015 [cit. 2019-04-17]. Dostupné z: https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought/.
5. HAN, Weili; LI, Zhigong; NI, Minyue; GU, Guofei; XU, Wenyuan. Shadow Attacks Based on Password Reuses: A Quantitative Empirical Analysis. *IEEE Transactions on Dependable and Secure Computing* [online]. 2016, roč. 15, č. 2, s. 309–320 [cit. 2019-04-17]. ISSN 1545-5971. Dostupné z DOI: 10.1109/TDSC.2016.2568187.
6. OKYLE, Carly. *Password Statistics: The Bad, the Worse and the Ugly* [online]. Entrepreneur Media, Inc., 2015 [cit. 2019-04-17]. Dostupné z: https://www.entrepreneur.com/article/246902.
7. Have I Been Pwned: Check if your email has been compromised in a data breach. *Have i been pwned* [online] [cit. 2019-04-17]. Dostupné z: https://haveibeenpwned.com/.
8. Have I Been Pwned: Pwned Passwords. *Have i been pwned* [online] [cit. 2019-04-17]. Dostupné z: https://haveibeenpwned.com/Passwords.

9. World Internet Users Statistics and 2019 World Population Stats. *Internet World Stats* [online] [cit. 2019-04-17]. Dostupné z: https://www.internetworldstats.com/stats.htm.

10. THOMAS, Kurt et al. Data Breaches, Phishing, or Malware?: Understanding the Risks of Stolen Credentials. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* [online]. Dallas, Texas, USA: ACM, 2017, s. 1421–1434 [cit. 2019-04-17]. CCS '17. ISBN 978-1-4503-4946-8. Dostupné z DOI: 10.1145/3133956.3134067.

11. 2018 Global Password Security Statistics Report — LastPass. *LastPass* [online] [cit. 2019-04-17]. Dostupné z: https://www.lastpass.com/state-of-the-password.

12. WASH, Rick; RADER, Emilee; BERMAN, Ruthie; WELLMER, Zac. Understanding Password Choices: How Frequently Entered Passwords Are Re-used across Websites. In: *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*. Denver, CO: USENIX Association, 2016, s. 175–188. ISBN 978-1-931971-31-7. Dostupné také z: https://www.usenix.org/conference/soups2016/technical-sessions/presentation/wash.

13. Authentication Methods. *European Union Agency for Network and Information Security* [online] [cit. 2019-04-17]. Dostupné z: https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/authentication-methods.

14. National Institute of Standards and Technology Special Publication 800-63B. *National Institute of Standards and Technology* [online]. 2017 [cit. 2019-04-17]. Dostupné z: https://doi.org/10.6028/NIST.SP.800-63b.

15. Features – KeePass. *KeePass* [online] [cit. 2019-04-17]. Dostupné z: https://keepass.info/features.html.

16. Security – KeePass. *KeePass* [online] [cit. 2019-04-17]. Dostupné z: https://keepass.info/help/base/security.html.

17. Secure Desktop – KeePass. *KeePass* [online] [cit. 2019-04-17]. Dostupné z: https://keepass.info/help/kb/sec_desk.html.

18. Two-Channel Auto-Type Obfuscation – KeePass. *KeePass* [online] [cit. 2019-04-17]. Dostupné z: https://keepass.info/help/v2/autotype_obfuscation.html.

19. Pass: The Standard Unix Password Manager. *passwordstore* [online] [cit. 2019-04-17]. Dostupné z: https://www.passwordstore.org.

20. 1# Password Manager, Vault, & Digital Wallet App — LastPass. *LastPass* [online] [cit. 2019-04-17]. Dostupné z: https://www.lastpass.com.

21. LastPass — Pricing – Premium, Families, Teams & Enterprise Cost. *LastPass* [online] [cit. 2019-04-17]. Dostupné z: `https://www.lastpass.com/pricing`.

22. Security — LastPass. *LastPass* [online] [cit. 2019-04-17]. Dostupné z: `https://www.lastpass.com/enterprise/security`.

23. Keep your secrets & passwords safe and secure — 1Password. *1Password* [online] [cit. 2019-04-17]. Dostupné z: `https://1password.com/security`.

24. Secure password vault encrypted with AES-256 — Sticky Password. *Lamantine Software* [online] [cit. 2019-04-17]. Dostupné z: `https://www.stickypassword.com/security`.

25. Live a simpler, safer life online with our Premium plan — Dashlane. *Dashlane Inc.* [online] [cit. 2019-04-17]. Dostupné z: `https://www.dashlane.com/plans/premium`.

26. GASTI, Paolo; RASMUSSEN, Kasper B. On the security of password manager database formats. In: *European Symposium on Research in Computer Security*. 2012, s. 770–787. ISBN 978-3-642-33166-4. Dostupné z DOI: `10.1007/978-3-642-33167-1_44`.

27. PALANT, Wladimir. *Master password in Firefox or Thunderbird? Do not bother!* [online]. Wladimir Palant, 2018 [cit. 2019-04-17]. Dostupné z: `https://palant.de/2018/03/10/master-password-in-firefox-or-thunderbird-do-not-bother/`.

28. Passbolt — Open source password manager for teams. *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://www.passbolt.com`.

29. Passbolt Help — How is passbolt different from other password managers? *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://help.passbolt.com/faq/discover/how-is-different`.

30. Passbolt Help — Release notes. *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://help.passbolt.com/releases`.

31. Passbolt — Features and roadmap. *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://www.passbolt.com/roadmap`.

32. Passbolt — Passbolt Pro – Self Hosted pricing. *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://www.passbolt.com/pricing/pro`.

33. Passbolt · GitHub. *GitHub, Inc.* [online] [cit. 2019-04-21]. Dostupné z: `https://github.com/passbolt`.

34. Passbolt Help — What kind of encryption does passbolt use? *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://help.passbolt.com/faq/security/encryption-tech`.

35. Passbolt — Passbolt as saas – cloud pricing. *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: `https://www.passbolt.com/pricing/cloud`.

36. Passbolt Help — Authentication in passbolt. *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: https://help.passbolt.com/tech/auth.

37. Passbolt Help — What is the security token? *Passbolt* [online] [cit. 2019-04-21]. Dostupné z: https://help.passbolt.com/faq/security/security-token.

38. HANAMSAGAR, Ameya; WOO, Simon S.; KANICH, Christopher; MIRKOVIC, Jelena. How Users Choose and Reuse Passwords. In: [online]. 2016 [cit. 2019-04-21]. Dostupné z: https://www.semanticscholar.org/paper/How-Users-Choose-and-Reuse-Passwords-Hanamsagar-Woo/c189c902be0cc09d6a50691ffb453fe04995e234#citing-papers.

39. YAN, Jeff; BLACKWELL, Alan; ANDERSON, Ross; GRANT, Alasdair. Password memorability and security: empirical results. *IEEE Security Privacy* [online]. 2004, roč. 2, č. 5, s. 25–31 [cit. 2019-04-21]. ISSN 1540-7993. Dostupné z DOI: 10.1109/MSP.2004.81.

40. PASSBOLT. *Sequence diagram of a form based authentication* [online obrázek] [cit. 2019-04-21]. Dostupné z: https://help.passbolt.com/tech/auth.

41. 2017 phishing and cyber-attacks statistics Cyber awareness training – phishing simulation — CybeReady. *CybeReady* [online] [cit. 2019-04-21]. Dostupné z: https://cybeready.com/2017-phishing-statistics-every-manager-must-know.

42. PASSBOLT. *Sequence diagram of a GPGAuth based authentication* [online obrázek] [cit. 2019-04-21]. Dostupné z: https://help.passbolt.com/tech/auth.

43. Wireshark · Go Deep. *Wireshark* [online] [cit. 2019-04-22]. Dostupné z: https://www.wireshark.org.

44. BERTOT, Remy. Passbolt Open source password manager for teams. In: *Youtube* [online]. 2018 [cit. 2019-04-23]. Dostupné z: https://www.youtube.com/watch?v=VGCHccWNqQQ.

45. PHP: GnuPG Functions – Manual. *The PHP Group* [online] [cit. 2019-04-23]. Dostupné z: https://www.php.net/manual/en/ref.gnupg.php.

46. GitHub – singpolyma/openpgp-php: OpenPGP.php is a pure-PHP implementation of the OpenPGP Message Format (RFC 4880). *GitHub, Inc.* [online] [cit. 2019-04-23]. Dostupné z: https://github.com/singpolyma/openpgp-php.

47. GitHub – openpgpjs/openpgpjs: OpenPGP implementation for JavaScript. *GitHub, Inc.* [online] [cit. 2019-04-23]. Dostupné z: https://github.com/openpgpjs/openpgpjs.

48. Cure53 security audit · openpgpjs/openpgpjs Wiki · GitHub. *GitHub, Inc.* [online] [cit. 2019-04-23]. Dostupné z: `https://github.com/openpgpjs/openpgpjs/wiki/Cure53-security-audit`.

49. CVSS v3.0 Specification Document. *FIRST.org, Inc.* [online] [cit. 2019-04-29]. Dostupné z: `https://www.first.org/cvss/specification-document`.

50. What is Scrum Methodology & Scrum Project Management. *CollabNet, Inc.* [online] [cit. 2019-05-10]. Dostupné z: `https://resources.collab.net/agile-101/what-is-scrum`.

Příloha **A**

Seznam použitých zkratek

AES – Advanced Encryption Standard

API – Application Programming Interface

CLI – Command Line Interface

CSV – Comma-separated values

CVSS – Common Vulnerability Scoring System

ENISA – European Union Agency for Network and Information Security

GnuPG – GNU Privacy Guard

GPL – GNU General Public License

GUI – Graphical User Interface

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

JSON – JavaScript Object Notation

KDF – Key derivation function

KISS – Keep it simple, stupid

NIST – National Institute of Standards and Technology

OpenPGP – Pretty Good Privacy

PBKDF2 – Password-Based Key Derivation Function 2

REST – Representational State Transfer

SHA – Secure Hash Algorithm

SSL/TLS – Secure Sockets Layer / Transport Layer Security

URL – Uniform Resource Locator

VPN – Virtual Private Network

Příloha **B**

Obsah přiložené SD karty

- `readme.txt` ..... stručný popis obsahu CD
- `thesis/`
  - `BP_smejkal_radek_2019.pdf` ..... text práce ve formátu PDF
  - `BP_smejkal_radek_2019.tex` .. zdrojová forma práce ve formátu LaTeX
  - `img/` ..... adresář s obrázky použitými v práci
- `src/` ..... adresář se zdrojovými soubory aplikace
  - `passbolt_api/` .... zdrojové soubory serverové části aplikace Passbolt
  - `passbolt_browser_extension/` .. zdrojové soubory doplňku prohlížeče
- `bin/`
  - `firefox_browser/` ..... spustitelná verze prohlížeče firefox
  - `passbolt_firefox_extension/` .... verze doplňku pro prohlížeč firefox
  - `passbolt_chrome_extension/` .... verze doplňku pro prohlížeč chrome
- `analysis/` ..... adresář se soubory síťové analýzy