FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Bezpečnostní analýza nástroje Bitwarden
Student:	Jakub Štrom
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

1) Nastudujte problematiku práce s hesly - vytváření, ukládání, používání, nástroje pro správu hesel.
2) Popište nástroj Bitwarden.
3) Srovnejte dokumentované vlastnosti Bitwardenu s aktuálními doporučeními a obvyklými řešeními.
4) Prozkoumejte aplikaci z uživatelského hlediska, vyznačte bezpečnostně citlivá místa a možné útoky na ně (např. použité šifrování, síťová komunikace apod.).
5) Analyzujte zdrojový kód aplikace se zaměřením na takto určené kritické oblasti. Vyhodnoťte, zda je implementace kvalitní a bezpečná.
6) Diskutujte svá zjištění, naleznete-li zranitelnosti, demonstrujte je vhodným způsobem uživateli.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 21. ledna 2022 v Praze.

Bakalářská práce

BEZPEČNOSTNÍ ANALÝZA NÁSTROJE BITWARDEN

Jakub Štrom

Fakulta informačních technologií
Katedra počítačových systémů
Vedoucí: Ing. Josef Kokeš
9. května 2022

Obsah

Seznam tabulek

Rád bych tímto poděkoval vedoucímu své práce Ing. Josefu Kokešovi za vedení a cenné rady během vypracování této práce.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. května 2022 .....

Abstrakt

Správci hesel jsou nástroje umožňující generovat a ukládat hesla. Tato práce se zaměřuje na konkrétního správce hesel Bitwarden a analyzuje bezpečnost jeho implementace. V rámci práce je nástroj popsán, je rozebráno jeho nakládání s uživatelskými daty a bezpečnost daných procesů. Jsou uvedeny konkrétní nalezené zranitelnosti nástroje a diskutován jejich dopad na uživatele. Možnost prolomení klíče hrubou silou z lokálních dat při použití zamykání PINem a možnost získání klíče při použití SSO byly hlavními nálezy.

Klíčová slova Bitwarden, správce hesel, bezpečnostní analýza, hesla, šifrování, zranitelnosti

Abstract

Password managers are software tools that can generate and store passwords. This thesis focuses on password manager Bitwarden and analyses the security of its implementation. As a part of this thesis, Bitwarden is described and its user data handling and the security of the associated processes are examined. Found vulnerabilities are listed and their impact on users is discussed. The main findings were a possibility of cracking the user's key using brute force from local data when a PIN lock is used and a possibility of stealing the key when logging in using SSO.

Keywords Bitwarden, password manager, security analysis, passwords, encryption, vulnerabilities

Seznam zkratek

2FA	Two-Factor Authentication
AES	Advanced Encryption Standard
API	Application Programming Interface
CDN	Content Delivery Network
CPU	Central Processing Unit
CSPRNG	Cryptographically-secure pseudorandom number generator
E2EE	End-to-end Encryption
GPU	Graphics Processing Unit
HKDF	HMAC-based Extract-and-Expand Key Derivation Function
HMAC	Hash-based Message Authentication Code
IV	Initialization Vector
JSON	JavaScript Object Notation
KDF	Key Derivation Function
MAC	Message Authentication Code
MFA	Multi-Factor Authentication
MitM	Man-in-the-Middle
OAuth 2.0	Open Authorization 2.0
OIDC	OpenID Connect
OWASP	Open Web Application Security Project
PBKDF2	Password-Based Key Derivation Function 2
PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards
RSA	Rivest-Shamir-Adleman
SAML 2.0	Security Assertion Markup Language 2.0
SSO	Single Sign-On
TOTP	Time-based One-time Password
ToS	Terms of Service
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Úvod

Hesla jsou zatím neoddělitelnou součástí digitálního světa, což je dnes zřejmé více než kdy jindy. S existencí mnoha digitálních služeb a aplikací je třeba ověřovat identitu uživatelů, a to nejsnáze řeší právě hesla. Tento nejstarší a nejpoužívanější způsob autentizace je dnes stále tak rozšířený hlavně kvůli vyvážení jeho použitelnosti, bezpečnosti a jednoduchosti. Přestože se dnes často uplatňují i jiné faktory autentizace, jsou zpravidla použité v kombinaci právě s hesly a zatím se nezdá, že by hesla mohla být úplně nahrazena.

Protože jsou nejvyužívanějším faktorem autentizace, je třeba aby uživatel zajistil jejich bezpečné používání a měl by pro každou službu používat heslo jiné. Není výjimkou, že by uživatel byl registrovaný u desítek služeb. Takové množství je na zapamatování velmi náročné a svádí k opakování stejného hesla. Volená hesla by měla navíc být komplexní, nejlépe náhodně generovaná. Bohužel v takovém případě jsou pro člověka i v malém množství hůře zapamatovatelná.

Tento problém řeší správci hesel, kteří umožňují zvolit jedno skutečně silné heslo a všechna zbylá mít bezpečně uložená. Uložená hesla mohou být velice komplexní, protože na uživateli nebude si je všechna pamatovat. Tuto službu nabízí právě i nástroj Bitwarden, který v posledních letech nabírá na popularitě. Jeho přednostmi jsou jednoduchost použití, dostupnost na více platformách se synchronizací dat a zároveň, že většinu základních funkcionalit poskytuje zdarma.

Je v zájmu uživatelů tohoto nástroje, aby s veškerými daty, které pomocí něj ukládají, pracoval co nejbezpečněji. Rozsahem tohoto nástroje a funkcí synchronizace napříč platformami je navíc prostor pro zranitelnosti poměrně široký.

Analýzu nástroje Bitwarden jsem zvolil především pro jeho zmíněnou rozšířenost, dostupnost a také proto, že ho sám využívám. Kromě toho, že je nástroj v základu poskytován zdarma je také open-source, což zjednodušuje a vybízí k prošetření dostupné implementace.

Cíle práce

Cílem práce je prozkoumat bezpečnost nástroje Bitwarden a určit, zda mu uživatelé mohou se svými daty skutečně důvěřovat. Prošetřím, jak jsou data synchronizována a zda během tohoto procesu nemůže Bitwarden, či jiná třetí strana získat neoprávněně informace. Dále se v práci zaměřím na platné doporučení nakládání s hesly a na to, jak je nástroj implementuje a dodržuje.

Dalším cílem je popis fungování nástroje Bitwarden. To znamená, jaké jsou použité algoritmy pro zabezpečení dat, jak probíhá autentizace uživatelů a jak jsou spravovány využívané šifrovací klíče.

Hlavním výstupem práce bude rozbor bezpečnosti implementace a popis nalezených slabin.

Kapitola 1
Práce s hesly

V této kapitole rozebírám, jak správně nakládat s hesly a jaká jsou aktuálně platná doporučení. Také zde popíši termíny související s hesly.

1.1 Faktory autentizace

Autentizace označuje proces ověření předkládané identity. Cílem procesu je ověřit, že ten, kdo chce přistupovat k nějakému zdroji je skutečně ten, kdo tvrdí že je. [1]

Při autentizaci je na výběr několik faktorů, na kterých je založeno samotné ověření uživatele. Nejrozšířenější 3 faktory jsou:

- Faktor znalosti – něco, co uživatel zná, například heslo nebo PIN.
- Faktor vlastnictví – něco, co uživatel má, například čipová karta.
- Faktor inherence – to, čím uživatel je, a to jak fyziologicky, tak chováním. Například otisk prstu nebo rozpoznání obličeje.

Kromě těchto nejpoužívanějších existují i další faktory, jako například faktor času nebo místa. [2][3]

Autentizace může probíhat využitím pouze jednoho faktoru. Často jsou ale pro zvýšení bezpečnosti autentizace použity různé faktory zároveň. Dvou-faktorová autentizace (2FA) označuje použití dvou libovolných faktorů. Více-faktorová autentizace (MFA) pak označuje použití dvou a více faktorů. Při MFA je často použit faktor znalosti v kombinaci s některým z ostatních faktorů, nejběžněji faktorem vlastnictví. [2]

1.2 Síla hesla

Aby bylo možné popsat, jak je zvolené heslo silné, je potřeba jeho sílu založit na nějaké konkrétní metrice.

1.2.1 Entropie

Jednou možnou metrikou je entropie hesla. Entropie je definována jako očekávaná hodnota informace v bitech obsažená v řetězci. Propojení se silou hesla demonstroval Massey [4], který ukázal, že entropie poskytuje dolní mez počtu očekávaných pokusů pro uhodnutí nějakého textu. Entropii poté NIST použil jako sílu hesla ve svém dokumentu NIST SP800-63. [5] Tam byla

použita k měření síly hesel ve vztahu s použitím konkrétních politik vytváření hesel. V článku [6] je ale na základě experimentů ukázáno, že takové použití entropie není vhodnou metrikou k měření síly hesel. [4][7]

Entropie vyjadřuje primárně komplexitu hesla. Komplexitou je myšleno, jaké znaky ho mohou tvořit, případně jaké musí nutně obsahovat. Někdy je v rámci komplexity brána i samotná délka hesla. Požadavky na komplexitu mohou být součástí doporučení pro vhodnou volbu hesla. Dnes už se ale od takových doporučení upouští, protože se ukazují spíše jako škodlivá pro hesla vytvořená uživateli. Uživatele příliš složité požadavky motivují k tomu, aby je splnili co nejjednodušším způsobem. To má pak za následek naopak oslabení hesel, kterému by doporučení právě měla předcházet. [8][9]

1.2.2 Uhodnutelnost

Druhou možností je určit sílu hesla na základě jeho „uhodnutelnosti“. Uhodnutelností je myšlena doba, za jakou je efektivní algoritmus schopný heslo uhádnout. Tato metrika bere v potaz, jakou znalost o možném složení hesla má útočník, a je tak závislá na konkrétní situaci. Vzhledem k zahrnutí pohledu útočníka je taková metrika více vypovídající o skutečné odolnosti hesla v případě útoku. [7][9]

Podobnou metrikou je počet uhádnutí zavedený v [7]. Počet uhádnutí je stanoven funkcí, která přiřadí každému heslu počet pokusů potřebných k uhádnutí. Tento počet je specifický pro konkrétní algoritmus prolamování hesel. Funkci je potřeba znovu implementovat pro každý nový algoritmus prolamování.

Oproti entropii hesla je uhodnutelnost hůře měřitelná a vyžaduje více informací o volených heslech. Většina uživatelů sílu hesla chápe spíše ve významu entropie. Zaměří se pravděpodobněji na to, jak heslo může být dlouhé a jaké jsou povolené znaky, či zda obsahují známá slova. Toto jsou ale charakteristiky spojené právě s komplexitou hesla, tedy s mírou entropie. [9]

V této práci je silou hesla myšlena právě jeho uhodnutelnost.

1.3 Platná doporučení

Různé regulační celky a odborníci vydávají doporučení ohledně použití hesel. Tato doporučení mají pomoci veřejnosti s bezpečným použitím hesel. Doporučení mohou být primárně určena pro konkrétní odvětví či státní organizace. Řídit se jimi ale může být prospěšné i pro ostatní, protože jejich cílem je vždy zvyšovat bezpečnost použití hesel.

Neexistuje jediný standard, který by se dal označit za obecně platný, ale existují takové, které by tohoto označení chtěly docílit. Přestože více organizací vydává nezávisle doporučení, často se v hlavních bodech shodují. Někdy mohou být ale jejich názory i protichůdné.

Pravidlům spojeným s životním cyklem hesla se říká heslová politika. Ta odkazuje na operace, které jsou s nimi spojeny od požadavků při jejich vytvoření po jejich přenos a třeba i životnost. [10]

Jedním z rozšířených doporučení je doporučení od NIST, konkrétně se jedná o NIST Special Publication 800-63B. V něm jsou hlavní doporučení následující.

- Alespoň 8 znaků pro uživatelem volené heslo a povolit alespoň 64 znaků dlouhé; 6 znaků pro náhodně volené.
- Povolovat všechny ASCII znaky a i Unicode znaky.
- Kontrolovat volená hesla proti seznamu předvídatelných, prolomených nebo běžně používaných hesel.
- Povolovat vkládání zkopírovaných hesel.

- Nevynucovat požadavky na komplexitu hesla.
- Nezavádět dobu platnosti hesla.
- Vynucovat MFA.

[11][10]

Tato doporučení nahrazují NIST doporučení z roku 2004. Ta se především zaměřovala na vynucování komplexity hesel a jejich pravidelné změny. Dnes, jak sám autor doporučení tvrdí, jsou tyto požadavky vnímány spíše jako škodlivé. Jak už bylo zmíněno v sekci 1.2.1, jejich složitost má často za následek, že si uživatelé volí více předvídatelná hesla. [8][5]

Dalším příkladem doporučení je CIS Password Policy Guide. [12] Tento dokument si klade za cíl sjednotit doporučení na jednom místě a být standardem pro politiku hesel. Je to soubor existujících „best practices“. Speciální důraz také klade na použití MFA.

Hlavními body doporučení jsou.

- Používat MFA, kdykoliv je to možné.
- Minimum 14 znaků bez MFA a 8 znaků s aktivním MFA. Maximum znaků by omezeno být nemělo.
- Zakazovat hesla, která jsou vedena jako prolomená nebo předvídatelná.
- Neposkytovat možnost nápovědy pro heslo.
- Podporovat správce hesel.
- Upřednostňovat tzv. „passphrase“.

  Passphrase je heslo složené z posloupnosti několika slov. Ta mohou, ale nemusí být oddělena mezerami. Samotná slova mají sice relativně nízkou komplexitu, kombinací několika slov ale vznikne silné heslo.
- Monitorování neúspěšných pokusů.

Tato doporučení se v některých částech podobají NIST doporučením. Obecně jsou ale více striktní a mají širší rozsah. [12]

Aktuálně jsou doporučována hesla spíše dlouhá a méně komplexní než naopak. Důvodem je hlavně snadná zapamatovatelnost pro uživatele. Dlouhé heslo může být dostatečně silné, i když není komplexní. Cílem je hlavně dosáhnout kompromisu mezi silou hesla a jeho zapamatovatelností, kterou právě požadavky na komplexitu ztěžují. Jednou z možností, jak odstranit problém zapamatovatelnosti, jsou správci hesel. [13]

1.4 Ukládání hesel

Při použití autentizace založené na heslech je velmi důležitým aspektem samotné uložení hesla. Ukládat heslo je potřeba, protože je nutné ověřit, zda heslo předkládané uživatelem je správné. Protože se jedná o tak citlivý údaj, je potřeba zajistit bezpečné uložení, a to nejlépe tak, aby i při případné kompromitaci místa uložení nebylo samotné heslo pro útočníka dosažitelné.

Nejjednodušším způsobem je samozřejmě ukládat heslo jako prostý text. Takto se při zadání uživatelem text snadno porovná, čímž se určí jeho správnost. Tento způsob je ale velice nebezpečný. Uložené heslo je vždy pro každého čitelné, útočníkovi tak stačí přístup k uložišti, aby heslo snadno získal. [14]

Bezpečnější je použít při uložení šifrování. Tento způsob odstraní problém s čitelností uložených dat pro každého. Nyní ale situaci komplikuje použití klíče. Ten je potřeba mít při autentizaci

dostupný a bude typicky uložený na podobném místě jako samotná hesla. V případě kompromitace by tak útočník teoreticky mohl získat i klíč. Tento způsob tak také není dobrou volbou pro ukládání hesel.

Předešlé metody umožňují serveru předložení hesla uživateli, například pokud by heslo zapomněl. To lze ale bezpečněji zajistit způsobem, kdy se uživatel autentizuje jinou metodou a heslo zvolí nové. Možnost zobrazit heslo je zbytečně nebezpečná. Server by hesla neměl ukládat tak, aby bylo možné získat jejich původní podobu. [14]

Lepším způsobem uložení je využití kryptograficky bezpečné hashovací funkce. Tímto způsobem získáme výstup, ze kterého nikdo nemůže snadno získat původní vstup, tedy heslo. Pro ověření hesla vždy znovu aplikujeme funkci, která pro stejný vstup bude mít vždy stejný výstup. Díky vlastnostem bezpečné hashovací funkce nelze získat původní heslo jinak než hrubou silou. [14][15, kap. 6]

Tento způsob má ale stále slabé místo v tom, že pro danou funkci může být heslo již prolomeno a může být součástí nějaké vyhledávací tabulky s prolomenými hesly. Toto je problém obzvláště u slabých hesel. Také lze sestrojovat takzvané Rainbow Tables, které prolamování značně ulehčí.

Odstranění těchto problémů usnadní použití soli (salt). Sůl je hodnota, která se vždy připojí k heslu před tím, než je vytvořen hash. Toto slouží k odlišení výsledných hodnot hashe při různých použití i pokud je vstupní hodnota stejná. Nejúčinnější je použití náhodné hodnoty jako soli. Přestože sůl může být uložena s heslem a není tudíž utajená, stále zamezí předešlým problémům. Útočník však stále může prolomit heslo sám, pokud zná sůl.

Pro ztížení prolamování hrubou silou se používají funkce pro odvození klíče (KDF). Ty jsou navrženy speciálně tak, aby jejich provedení bylo výpočetně a/nebo paměťově náročné. Toto výrazně zpomalí rychlost pokusů při prolamování. Některé tohoto dosahují aplikací vícero iterací jiné hashovací nebo pseudonáhodné funkce, čímž dosáhnou pomalejšího výpočtu. Příkladem takovéto funkce je třeba Password-Based Key Derivation Function 2 (PBKDF2). [14]

Přestože je primárním účelem KDF vytvoření klíče z hesla, lze je aplikovat i na jiné účely. Jedním z nich je právě kontrola hesel, kde nahrazují klasické hash funkce. Toto použití je zmíněno i v RFC2898. [16]

1.4.1 PBKDF2

PBKDF2 je funkce pro odvození klíče definovaná standardem RFC2898. Vnitřně používá zvolenou pseudonáhodnou funkci, kterou v iteracích aplikuje. Jako pseudonáhodná funkce bývá často volená HMAC (Hash-based Message Authentication Code). Konkrétně HMAC-SHA1 je uvedena jako příklad v samotném standardu. [16]

Kromě vstupního hesla jsou hlavními parametry PBKDF2 sůl a počet iterací. Sůl umožní diverzifikovat výstup funkce pro stejné heslo, stejně jako je tomu u hash funkcí. Počet iterací pak slouží k zvýšení náročnosti výpočtu, a tak k zpomalení provedení výpočtu. PBKDF2 funkce nabízí parametr pouze pro zvýšení výpočetní náročnosti, ne však náročnosti paměťové. Standard doporučuje jako minimum 1 000 iterací, které by měly ztížit prolamování hrubou silou a zároveň nemít dopad při legitimním použití na uživatele. Tento počet je však s nárůstem výpočetního výkonu dnes příliš nízký. Dle doporučení NIST z roku 2017 by počet měl být minimálně 10 000, OWASP (Open Web Application Security Project) v roce 2021 doporučil 310 000 iterací pro PBKDF2-HMAC-SHA256. [16][11][17]

1.5 Správci hesel

Hesla jsou ve své podstatě pro člověka nepraktická. Aby nebyla prolomitelná hrubou silou, je potřeba dosáhnout určité komplexity, respektive síly daného hesla. To je však pro člověka značně náročné. Tím je na uživatele směřován velký tlak, se kterým se pak vyrovnávají tím, že přijímají nezdravé bezpečnostní návyky. Aby hesla nezapomněli, volí často nevhodné strategie, jako je

zapisování hesel nebo vytváření snadno zapamatovatelných hesel. Nejsnadnější na zapamatování jsou hesla založená na běžných slovech, jejich variacích či osobních údajích. Taková hesla jsou ale zranitelná vůči slovníkovým útokům nebo asociačním útokům. Zmíněné útoky výrazně sníží množinu možných hesel a jsou útočníkem většinou snadno proveditelné. Při použití osobních údajů je heslo obzvláště uhádnutelné někým blízkým, ale často mohou tyto údaje dostupné i náhodnému útočníkovi, například ze sociálních sítí. [13][18]

Hlavním přínosem správců hesel je právě to, že umožňují uživatelům soustředit svá hesla na jedno místo tak, aby si je nemuseli všechna pamatovat. Ulehčují tak paměti uživatele a eliminují problém špatného nakládání s hesly díky usnadnění jejich zapamatování. Pro bezpečné použití správce hesel je důležité mít silné hlavní heslo, které je jediné, jaké si pak uživatel musí pamatovat. Toto heslo je základem bezpečnosti uložení ostatních hesel spravovaných správcem. Protože ostatní hesla si díky správci pamatovat nemusí, je zde zátěž na paměť uživatele znatelně nižší a dá se očekávat, že tedy bude schopen a ochoten volit své hlavní heslo odpovědněji. Další hesla pak jsou spravována správcem hesel, ve kterém vysoká komplexita uživateli nemusí vadit, protože mu je v případě potřeby správce poskytne.

Je nutné zmínit, že tento přístup má i zřejmou nevýhodu. Vytváří se takto jediné místo selhání. Pokud by se útočník zmocnil uživatelova hlavního hesla, může teoreticky získat přístup ke všem jeho používaným službám. Stejně tak kdyby uživatel heslo zapomněl, ztratí přístup i ke všem ostatním uloženým heslům. To komplikuje fakt, že možnost obnovení hesla je z hlediska bezpečnosti u správců problematická. Pokud by takovou možnost poskytovali, z principu to znamená, že mají přístup i k ukládaným heslům. Aby mohli hlavní heslo nahradit, musí být schopni novým heslem zabezpečit stávající uložená hesla. To pak znamená, že by museli být schopni přistupovat i k jejich textové podobě. Tím by se zavedlo velké bezpečností riziko a zároveň by to snižovalo důvěryhodnost samotného správce pro uživatele, který by neměl mít důvod k takovému přístupu. [13]

1.5.1 Rozdělení

Na trhu existuje mnoho různých správců hesel. Ti se mohou lišit v dostupnosti, nabízených funkcionalitách, uživatelském rozhraní či ceně. Bezpečnost správců hesel v principu spočívá na jednom hlavním hesle. Lze je však dále dělit na základě toho, jak získávají hesla pro služby, a to na hlavní dvě kategorie — generative a retrieval. [19][20][21]

1.5.1.1 Generative správci hesel

Generative (nebo také Vaultless) správci hesel neukládají žádná hesla. Hesla pro konkrétní služby vytvářejí využitím kryptografických funkcí, kde základem je vždy hlavní heslo. K němu jsou přidaná další data, jako např. URL webu nebo přihlašovací jméno uživatele, čímž se dosáhne vygenerování unikátního hesla pro každou službu. [19][20]

Výhodou tohoto přístupu je, že není potřeba ukládat žádná hesla, není tedy prostor k jejich úniku a případnému prolomení. Zároveň je zde uživatel zbaven možnosti vytvářet vlastní hesla, hesla budou vždy generována, a tudíž jsou eliminovány nedostatky, které by do nich mohl zanést uživatel. Také je tak uživateli zamezeno znovupoužívat hesla napříč různými službami, heslo by mělo být unikátní na základě použitých metadat. Na druhou stranu může být tento způsob vytváření hesel pro uživatele nepraktický. Heslo, které může zvolit, je úzce spjaté s jeho hlavním heslem. Pokud již má vytvořený účet se zvoleným heslem, nemůže ve správci hesel použít své stávající heslo a musí ho změnit na vygenerované správcem.

Navíc veškerá bezpečnost leží na hlavním hesle. Na rozdíl od retrieval správců není potřeba, aby útočník získal také databázi zabezpečených hesel. Únikem hlavního hesla se lze teoreticky přihlásit na všechny používané služby. Stačí aby útočník měl přístup ke zbylým údajům vyžadovaných algoritmem použitém při generování. Ty však zpravidla nemají tak utajovanou povahu jako hlavní heslo a bude snazší je získat. Útočník takto může získat nejen současná hesla, ale

také i všechna budoucí hesla, která by mohl uživatel vygenerovat. Změna hlavního hesla by zároveň znamenala změnu všech na něj vázaných hesel pro služby. Stejně tak, pokud by chtěl nebo potřeboval změnit jedno heslo, například kvůli prozrazení, musí opět změnit hesla všechna. [19]

1.5.1.2 Retrieval správci hesel

Pro retrieval správce hesel jsou uživatelská hesla společně ukládána a chráněna jedním hlavním heslem, šifrovaná v databázi. V případě vyžádání jsou pak dešifrována a předána uživateli. [19][20]

Výhodou retrieval správců pro uživatele oproti generative je volnost v tom, jaké heslo pro služby může použít. Konkrétní heslo mu není určováno správcem, heslo si může sám zvolit či vygenerovat. Při přechodu na správce hesel může uložit svá stávající hesla a není nucen je měnit na nová. Zároveň stále platí fakt, že je potřeba aby si pamatoval pouze své hlavní heslo. Ostatní hesla může vygenerovat skutečně komplexní, protože si je nebude muset pamatovat a v případě potřeby mu je poskytne správce hesel. Zde má oproti generative správcům volnost v tom, že si míru komplexity sám může zvolit. Entropie používaných hesel tak není vázaná na hlavní heslo jako je tomu u generativních. Hlavní heslo tu slouží jako tajemství zabezpečující ukládaná hesla služeb, ne jako vstup pro generování těchto hesel. Na rozdíl od generative správců tento způsob také umožňuje větší flexibilitu, kde se stejným způsobem mohou ukládat i data, která nemusí nutně být přihlašovací údaje. Libovolná data jsou pak šifrována stejným způsobem jako je tomu u údajů. Toto by pro generativní nebylo možné bez samostatné implementace, která by ukládání dat umožňovala. Příkladem tohoto ukládání je i nástroj Bitwarden, který takto ukládá různé typy dat, nejen přihlašovací údaje. [19][22, /managing-items]

Nevýhodou je zde, že na rozdíl od generativních správců hesel jsou hesla někde persistentně ukládána. Hrozí zde tedy potenciální únik těchto uložených hesel. Přestože hesla budou zabezpečena hlavním heslem, stále je to příležitost alespoň k útoku hrubou silou v pokusu o prolomení šifrovaných hesel. [19]

Retrieval správce hesel můžeme dále rozlišit dle toho, zda ukládají hesla pouze lokálně nebo na vzdáleném serveru (také označované cloud-based, nebo online).

1.5.2 Použitelnost a rozšíření

S bezpečnostním přínosem správců hesel je úzce spjata jejich použitelnost. Pokud nejsou pro uživatele snadno použitelní, budou se jim raději vyhýbat a spravovat si hesla sami. Má-li tedy být správce přínosný, neměl by zanedbávat důraz právě na snadné použití uživateli.

Přestože jsou správci hesel odborníky doporučováni, jejich rozšířenost stále není příliš velká. Důvodů, proč je lidé nepoužívají, existuje mnoho. Často o nich buď vůbec nevědí, příliš jim nedůvěřují či je odmítají používat z nějakého jiného důvodu. [23][24]

Kapitola 2
Nástroj Bitwarden

V této kapitole popíši části nástroje a jaké funkcionality nabízí. Také popíši jeho vnitřní fungování, jaké využívá algoritmy a jak nakládá s daty uživatelů.

2.1 Architektura

Protože se jedná o online správce hesel, používá client/server architekturu. Implementace nástroje se tudíž skládá ze serverové části a klientských aplikací, které se serverem komunikují.

2.1.1 Server

Implementace serverové části je psána v jazyce C# ve frameworku ASP.NET Core. Skládá se z několika dílčích aplikací, kde každá poskytuje různé služby klientským aplikacím. Kromě těchto aplikací server zahrnuje také databázi.

- **Api Server**
- **Identity Server**
- **Icon Server**
- **Notification Server**
- **Web Vault Server**

[25]

Výše zmíněné serverové aplikace lze nasazovat odděleně. Jelikož v klientských aplikacích jsou URL (Uniform Resource Locator) adresy odkazující na dílčí části plně konfigurovatelné, je možné, aby každá měla jiné umístění, třeba i fyzicky. Zmíněná URL adresa lze nastavit buď jednotná pro všechny části serveru, nebo lze dle potřeby zadat URL pro každou část zvlášť. Toto zároveň umožňuje, aby si některé části uživatel nasadil vlastní a jiné používal poskytované z Bitwarden serveru, například icon server. Je ale potřeba brát v úvahu, že nasazení jedné části může být závislé i na vlastním nasazení dalších. To se konkrétně týká vlastně hostovaného api serveru, který je potřeba nasazovat současně s vlastním identity serverem a také databází. Tyto části zahrnují veškeré vzdálené API (Application Programming Interface), jaké mohou klientské aplikace potřebovat. [25]

Krom zmíněných jsou k dispozici další podpůrné aplikace, například Admin portál pro správu serverové části. [22, /admin-portal]

Hlavní rozhraní je zahrnuto v rámci api serveru. Jsou zde operace například pro správu trezoru, organizací nebo Send a dalších funkcionalit. Identity server obstarává autentizaci uživatelů, ne však registraci, ta je zahrnuta v api serveru.

Ukládané údaje mohou být spojeny s nějakými URI (Uniform Resource Identifier), pro která jsou určené. V rámci uživatelského rozhraní se pak mohou zobrazovat i ikony vázané k příslušným doménám webů, což ulehčuje identifikaci uložených přihlašovacích údajů. Smyslem icon serveru je pak nabízet tyto ikony klientským aplikacím. [22][/website-icons]

Pro některé uživatele bude největší výhodou tohoto nástroje fakt, že data jsou synchronizována na serveru. Z tohoto serveru pak klientské aplikace data získávají a zpřístupňují uživateli. Zde je tudíž podstatné zajistit bezpečný přenos a ukládání dat. Data nesmí uživatelovo zařízení opustit v nešifrovaném stavu a už vůbec se tak nesmí ukládat na server. Bitwarden toto zajišťuje použitím „Zero Knowledge Encryption“. [26]

Synchronizovat data na server může být ale pro někoho také odrazující. Přestože jsou data šifrovaná a aplikuje se Zero Knowledge Encryption, mohou být uživatelé, kteří je i tak chtějí mít plně pod kontrolou a nepřejí si je ukládat na cizí server. V tomto směru může být řešením nasadit si vlastní server. Bitwarden toto umožňuje, nabízí i postup popisující nasazení serveru včetně skriptů usnadňující nasazení pro Windows, Linux a MacOS. Protože je Bitwarden open-source, existují i populární alternativní implementace serveru. Mezi nimi asi nejpopulárnější VaultWarden implementovaný v programovacím jazyce Rust. [27][22, /install-on-premise-windows]

Přestože tato možnost existuje, určitě to není dobrým řešením pro každého. Jedná se o citlivou službu, pokud by si ji tedy někdo chtěl svépomocí provozovat doma a vystavovat do internetu, musí pak sám nést rizika. Dobré nasazení vyžaduje jak pečlivou konfiguraci samotné služby, tak bezpečnou infrastrukturu, což vyžaduje velké úsilí. Využití zde ale může být pro firmy, které k tomu mohou mít lepší prostředky. Pro přístup ke všem funkcionalitám je ale potřeba zakoupit licenci. Základní funkce jsou dostupné bez licence. [22, /licensing-on-premise]

2.1.2 Klientské aplikace

Klientskými aplikacemi se rozumí veškeré aplikace používané koncovými uživateli. Ty slouží jako uživatelské rozhraní pro přístup k datům uživatelů a používání zbylých nabízených funkcionalit. V závislosti na využitých funkcionalitách komunikují se serverovým API. Primárně využívány jsou api server a identity server, které poskytují API pro základní funkcionality a správu účtů.

Klientské aplikace jsou dostupné na několika platformách. V prohlížečích je dostupné rozšíření s podporou pro Google Chrome, Mozilla Firefox, Opera, Microsoft Edge, Safari, Vivaldi, Brave a Tor. Pro mobilní zařízení je podporován operační systém Android a iOS. Na desktop je k dispozici multi-platformní aplikace vyvíjená v Electronu. Ta je distribuována pro Windows, Linux a MacOS. V rámci práce pracuji primárně s Android aplikací verze 2.13.0[1] a serverem verze 1.43.0[2], pokud není řečeno jinak. V některých případech pak s Desktop aplikací verze 1.28.3[3]. [22, /getting-started-browserext][22, /getting-started-mobile][22, /getting-started-desktop]

Většina funkcionalit je dostupná ve všech aplikacích, existují však výjimky. Ty se týkají převážně administrativních operací jako například změna hesla, vytvoření organizace, či rotace symetrického klíče. Tyto operace jsou dostupné pouze z web vault aplikace, což je webová aplikace. [22, /getting-started-webvault]

2.2 Ochrana dat

Bitwarden při svém použití nakládá s velmi citlivými daty, které je potřeba důkladně chránit. Toho nástroj dosáhne použitím vhodného šifrování.

[1] https://github.com/bitwarden/mobile/tree/v2.13.0
[2] https://github.com/bitwarden/server/tree/v1.43.0
[3] Dostupné z https://github.com/bitwarden/desktop/releases/tag/v1.28.3

Většina kódu týkajícího se šifrování a šifrovacích klíčů se nachází ve třídě `CryptoService` z `Core` knihovny mobilních aplikací.

2.2.1 Symetrické šifrování

Pro symetrické šifrování je v rámci služby používaný vždy AES (Advanced Encryption Standard) a to jak dle `EncryptionType` tak i dokumentace. `EncryptionType` je nástrojem definovaný enum, jehož hodnoty popisují konkrétní podobu šifrování. Pro symetrické šifrování má definované následující hodnoty:

- AesCbc256_B64
- AesCbc128_HmacSha256_B64
- AesCbc256_HmacSha256_B64

Volba použitého paddingu sice není uvedena, ale ve všech případech je použitý PKCS#7 (Public Key Cryptography Standards #7).

V dokumentaci je k symetrickému šifrování uvedeno, že je používáno AES-256 v módu CBC (Cipher Block Chaining). Mód CBC je dle definovaných hodnot použit vždy. `EncryptionType` ale obsahuje i AES-128, což by neodpovídalo dokumentaci. Z metody `ResolveLegacyKey` však vyplývá, že se jedná o typ určený pouze pro zpětnou kompatibilitu. V aktuálních verzích by tak neměl být používán. [26][22, /what-encryption-is-used]

Typy definované v `EncryptionType` jsou primárně použité při serializaci dat a klíčů a nejsou určující při samotném šifrování. Jaké konkrétní šifrování je použité pak závisí pouze na předaných klíčích při šifrování do metody `AesEncryptAsync`. Dle délky šifrovacího klíče se rozliší mezi použitím 128 či 256 bitové verze AES. Vypočtení MAC (Message Authentication Code) pak závisí čistě na přítomnosti klíče určeného pro MAC.

Dle používaných klíčů v aktuálních verzích, které jsou vždy 64 bytů dlouhé, ale vždy dochází k vypočtení MAC. Protože je klíč 64 bytů dlouhý, tak polovina určená pro šifrování bude 256 bitová. Z toho tak zároveň vyplývá i potvrzení, že je vždy použita 256 bitová verze AES. V takovém případě proces šifrování dat probíhá následovně.

1. Vygeneruje se 16 bytů dlouhý inicializační vektor (IV) pomocí kryptograficky bezpečného generátoru náhodných čísel (CSPRNG).
2. Vstupní data se zašifrují klíčem pro šifrování s použitím vygenerovaného IV.
3. Vypočte se MAC ze zřetězení bytů `IV` a šifrovaných dat.
4. Šifrovaná data, IV i MAC se uloží buď jako syrová data nebo v base64 kódování.

Konkrétní způsob uložení zašifrovaných dat společně se souvisejícími daty závisí na jejich účelu. Dvěma možnostmi jsou, že se buď uloží jako pole bytů nebo do jedné struktury jako řetězce v kódování base64. Tato struktura se pak při přenosu či perzistentním uložení serializuje do jednoho řetězce. Tento řetězec má stejný formát jako pole bytů, který je následující: `typ.IV|DATA[|MAC]`. Typ je číslem vyjádřený `EncryptionType`, který zároveň udává informaci, zda je přítomný MAC. Ten je volitelně přítomný jako poslední.

2.2.1.1 Používané symetrické klíče

Služba pro symetrické šifrování používá více různých klíčů. Klíčem je ve kontextu nástroje myšlena struktura `SymmetricKey`. Ta reálně může obsahovat buď jeden šifrovací klíč, nebo jak šifrovací klíč, tak MAC klíč. Zároveň je součástí struktury i typ klíče vyjádřený hodnotou `EncryptionType`.

Klíče používané v službě se dají rozdělit dle způsobu jejich vzniku na dva druhy. Jedním jsou klíče původem od uživatele a druhým jsou generované.

Klíče původem od uživatele jsou odvozené z hesla dodaného přímo uživatelem. Nejjednodušším příkladem je například hlavní klíč odvozený z hlavního hesla.

- Hlavní Expandovaný Klíč (Hlavní Klíč)

  Toto je základní klíč, na kterém leží bezpečnost celé aplikace. Vzniká původem z uživatelova Hlavního Hesla. Z toho se funkcí PBKDF2 (Password-Based Key Derivation Function 2) odvozuje 256bitový hlavní klíč. Funkci se předává emailová adresa jako sůl a používá se počet iterací, který je pro daný účet nastavený. Pokud si ho uživatel nezmění ve web vault aplikaci, pak je tato hodnota v základu 100 000.

  Pro MAC je ale potřeba také samostatný klíč, a tak se hlavní klíč expanduje na 512 bitů. Expanze se provádí funkcí HKDF (HMAC-based Extract-and-Expand Key Derivation Function), kde vstupem je vždy hlavní klíč a šifrovací klíč je výstupem funkce se solí „enc“ a klíč pro MAC výstupem funkce se solí „mac“. Dohromady takto vznikne hlavní expandovaný klíč dlouhý 512 bitů, kde první polovina se používá při šifrování a druhá pro MAC. Hlavní heslo, hlavní klíč ani hlavní expandovaný klíč nikdy nejsou posílány na server a zůstávají pouze v zařízeních uživatele. [26]

- PIN Klíč

  Tento klíč je použitý v klientské aplikaci při zamykání PINem bez použití hlavního hesla při restartu. Vzniká ze zadaného PINu a je tvořen v metodě `MakePinKeyAysnc`[4]. Odvození probíhá stejným způsobem jako u hlavního hesla, včetně použití emailové adresy jako soli. Pak následuje expanze opět analogicky jako pro hlavní expandovaný klíč.

  Používá se pro zašifrování hlavního klíče, aby bylo možné ho bezpečně uložit při zamčení.

Generovaná hesla vznikají vygenerováním náhodných bytů pomocí CSPRNG. Většina je generována metodou `MakeEncKey` s výjimkou Send klíče. Metoda `MakeEncKey` vygeneruje 64 bytů, které po 32 bytech tvoří šifrovací a MAC klíče. Zároveň ihned provede šifrování vytvořeného klíče na základě klíče předaného při volání. Vrací tudíž jak nezašifrovaný, tak zašifrovaný klíč.

- Generovaný Symetrický Klíč

  Tento klíč vzniká při registraci uživatele a je obtížné ho změnit, ale tato možnost existuje. Je používán jako základní klíč pro šifrování; pokud není pro nějakou šifrovací operaci určen speciální klíč, je použit tento. Především je jím chráněný Vault. Klíč samotný je zabezpečen šifrováním s využitím hlavního expandovaného klíče.

  Pokud by došlo k úniku tohoto klíče, je možné ho přegenerovat, aby se zamezilo jeho zneužití. Tato operace je dostupná pouze přes web vault. Zde je možnost „rotace“ klíče, tím se vytvoří nový, kterým se přešifrují veškerá data. Pro provedení je potřeba se odhlásit ze všech zařízení, aby nedošlo k použití starého klíče při synchronizaci dat. Tím by se Vault data mohla dostat do neplatného stavu a stala by se tak nepoužitelná. Zrotování klíče samozřejmě nepomůže, pokud útočník získá zároveň data šifrovaná starým klíčem, dříve než uživatel provede rotaci. [22, /account-encryption-key]

- Generovaný Symetrický Klíč Organizace

  Generovaný symetrický klíč Organizace je svým účelem a vznikem podobný generovanému symetrickému klíči. Místo využití uživatelem je ale využíván v rámci organizace. Vzniká založením organizace, která lze založit pouze ve web vault aplikaci. Samotný klíč je vytvořený metodou `MakeShareKey`[5]. Tato metoda opět vygeneruje 64 bytů. Největším rozdílem je jeho

[4] Překlep v názvu je přítomný v kódu.

[5] V mobilních aplikacích, které tuto C# knihovnu používají sice založit organizace nelze, knihovní metoda pro vytvoření klíče je ale přítomná. Funkčně je stejná jako skutečně použitá metoda ve web vault.

samotné zabezpečení, aby bylo možné ho sdílet mezi více uživateli. Klíč je šifrován asymetricky pomocí RSA-2048 (Rivest-Shamir-Adleman). Pro každého člena je šifrován jeho veřejným klíčem. Využívá se pro šifrování položek sdílených v rámci organizace, kde se použije namísto generovaného symetrického klíče.

- Send Klíč

  Na rozdíl od ostatních klíčů je vygenerován jako pole 16 bytů. Tyto byty samotné ale nejsou přímo klíčem použitým při šifrování. Před použitím je rozšířen ze 16 na 64 bytů funkcí HKDF pro šifrovací a MAC klíč. Takto rozšířený je pak používán pro zabezpečení Send dat. Samotný Send klíč (nerozšířených 16 bytů), je šifrován generovaným symetrickým klíčem. Šifrovaný je pak připojen k datům a s nimi synchronizován na server.

- Klíč Příloh

  Vygenerovaný speciální klíč pro přílohu dlouhý 64 B. Opět jedna polovina tvoří šifrovací klíč a druhá MAC klíč. Sám je zabezpečený Symetrickým klíčem uživatele nebo organizace. Je jím šifrován pouze samotný soubor tvořící přílohu, ne související metadata uložená v trezoru.

2.2.1.2 Shrnutí symetrického šifrování

Pro symetrické šifrování je použita šifra AES-256 v módu CBC. Šifrování probíhá v kombinaci s vypočtením MAC ve schématu Encrypt-then-MAC pro zajištění integrity dat. Schéma Encrypt-then-MAC znamená, že se data nejprve zašifrují a ze zašifrovaných dat se následně počítá MAC. [15, kap. 8] Pro počítaní MAC je používán algoritmus HMAC-SHA256.

2.2.2 Asymetrické šifrování

V rámci služby je kromě symetrického šifrování na různých místech použito asymetrické šifrování. Hlavní oblastí, kde je využito, je sdílení údajů mezi uživateli, tedy v organizacích. Použitý algoritmus je RSA-2048. Opět je v `EncryptionType` definováno několik možností:

- Rsa2048_OaepSha256_B64
- Rsa2048_OaepSha1_B64
- Rsa2048_OaepSha256_HmacSha256_B64
- Rsa2048_OaepSha1_HmacSha256_B64

Při prozkoumání metody `RsaEncryptAsync` je však vidět, že ve skutečnosti se používá pouze typ Rsa2048_OaepSha1_B64. Při dešifrování v metodě `RsaDecryptAsync` jsou pak typy použity vždy bez HMAC kontroly, liší se tak jen ve volbě SHA-256 nebo SHA-1 pro OAEP.

SHA-1 dnes již není považovaná za bezpečnou, protože je kryptograficky prolomená. V roce 2017 se podařilo nalézt 2 různé PDF dokumenty, které mají stejnou SHA-1 hash, čímž byla prolomena její bezkoliznost. OAEP však má na zvolený algoritmus nízké požadavky, tudíž je pro něj volba SHA-1 stále bezpečná. [28][29]

2.2.2.1 Používané asymetrické klíče

Bitwarden využívá dva různé páry asymetrických klíčů. Jeden pro uživatele a jeden pro organizaci.

- RSA pár klíčů uživatele

  Tento pár je podobně jako generovaný symetrický klíč vytvořen při registraci uživatele. Jedná se o klíče pro používaný pro RSA-2048, délka klíče je tudíž 2048 bitů. Vytváří se metodou

`MakeKeyPairAsync`, samotné vytvoření zajišťuje použitá kryptografická knihovna. Využívaný je pak hlavně v kontextu organizací pro sdílení symetrického klíče organizace. Kromě toho je použitý pro Emergency Access funkcionalitu.

Z veřejného klíče se také generuje takzvaná Fingerprint Phrase. Z hashe daného klíče se vytvoří posloupnost slov, která je takto trvale spjata s účtem uživatele. Fingerprint phrase pak slouží k potvrzení, že používaný klíč je skutečně uživatelův a nebylo s ním manipulováno například MitM (Man-in-the-Middle) útokem. [22, /fingerprint-phrase][30]

- RSA pár klíčů organizace

  Pár klíčů organizace je vytvořen během založení organizace. Opět se jedná o klíče pro RSA-2048. Tento pár je používán pro funkcionalitu Admin Password Reset dostupnou v organizacích pro Enterprise předplatné.

Oba páry klíčů jsou ukládány na serveru. Veřejný klíč je nešifrovaný a soukromý klíč vždy šifrovaný symetrickým klíčem uživatele, respektive organizace.

2.2.3 Zero Knowledge Encryption

Zero Knowledge Encryption, také někdy Zero Knowledge Architecture nebo No Knowledge architektura, znamená, že poskytovatel služby neví nic o datech, která jsou ukládaná na jeho serveru. Reálně to znamená, že data jsou zašifrována před odesláním a poskytovatel služby nikdy nemá přístup k šifrovacímu klíči. Tento přístup je užitečný hlavně u služeb, kde soukromí uživatelů je prioritní, a tak je některými správci hesel používán. Kromě nástroje Bitwarden je to například NordPass nebo Keeper. [31][32][33]

Základem bezpečnosti architektury je šifrování, konkrétně pak E2EE (End-to-End Encryption). E2EE znamená, že kromě samotného zpracování na koncových zařízeních jsou data vždy šifrovaná. Jelikož v nástroji Bitwarden jsou veškerá uživatelská data přístupná nešifrovaně jen při odemčení Vaultu, tak toto splňuje. Aby se pak dala architektura označit za Zero Knowledge, je potřeba zajistit, že používaný klíč je vždy pouze ve vlastnictví uživatele. Bitwarden vždy symetrický klíč uživatele posílá na server pouze šifrovaný. Hlavní klíč, kterým je šifrovaný, ani heslo, ze kterého klíč vzniká, nikdy neopouštějí zařízení uživatele. Tudíž nástroj splňuje požadavky této architektury. [33]

2.2.4 Kryptografické knihovny

Samotné kryptografické operace a kryptografická primitiva zajišťují knihovny třetích stran. Vzhledem k různým platformám, pro které existují aplikace, a jiným programovacím jazykům, kterými jsou psány, je použito i více různých kryptografických knihoven.

Dle dokumentace jsou použité následující knihovny:

- C# (Pro mobilní aplikace)
  - CommonCrypto
  - Javax.Crypto
  - BouncyCastle
- Javascript
  - Web Crypto
  - Node.js Crypto
  - Forge

Všechny tyto knihovny jsou buď součástí používaného frameworku a/nebo se jedná o populární knihovny. Vzhledem k jejich rozšíření je tak lze považovat za bezpečné. [22, /what-encryption-is-used]

2.3 Trezor

Hlavní částí nástroje je takzvaný trezor (Vault). V něm jsou obsažena data ukládaná v nástroji uživatelem. Jednotlivé položky mohou být více typů, v kódu jsou tyto položky označované jako „cipher“. Pro jednoduché organizování položek lze vytvářet složky a položky do nich přiřazovat. Složky lze také dále zanořovat do dalších úrovní. Vault data je souhrnné označení pro všechna data obsažená v trezoru. [22, /managing-items][22, /folders]

Ukládaná data jsou šifrovaná, ale šifrují se pouze jednotlivé hodnoty, ne samotná JSON (JavaScript Object Notation) struktura, ve které jsou posílány a lokálně ukládány. Navíc jsou některá metadata jako například identifikátor položky, která šifrovaná nejsou. Veškerá data ukládaná uživatelem ale šifrovaná jsou vždy pokud s nimi nepracuje uživatel v klientské aplikaci.

Typy jednotlivých položek mohou být následující:

- „Login“ – přihlašovací údaje

  Pod tímto typem lze ukládat přihlašovací údaje. K nim je možné přiřadit URL webových stránek, ke kterým patří. Pokud to příslušná aplikace podporuje, jsou pro přiřazené stránky nabízeny a umožňují automatické vyplnění údajů. V rámci tohoto typu lze kromě klasických hesel ukládat TOTP (Time-based One-time Password). K tomu stačí vyplnit autentizační klíč a nástroj pak zobrazuje vygenerované heslo.

- „Card“ – platební karty
- „Identity“ – osobní údaje
- „Secure note“ – zabezpečené poznámky

  Umožňují ukládat libovolný text. Dále si v nich lze přidávat další vlastní pole. Ta se skládají vždy z názvu a hodnoty, která může být textová, pravdivostní, nebo skrytá. Skrytá hodnota uložený text zobrazuje jako tečky a musí se kliknutím odkrýt. Tento typ je zahrnutý v rámci předešlých typů a vše zmíněné je u nich přístupné také. [22, /managing-items]

2.3.1 Zamykání

Bitwarden nabízí po přihlášení v aplikaci místo odhlašování pouze zamknout trezor. Přihlášení je vždy nutnou podmínkou, protože slouží k získání dat ze serveru, které se po přihlášení uloží šifrovaně v zařízení. Zamknutí slouží jako metoda lokální autentizace uživatele, bez nutnosti komunikace se serverem. Volbou zamykání tudíž uživatel může s trezorem pracovat i offline, v omezeném režimu. Položky trezoru si může zobrazovat ale operace, které vyžadují komunikaci se serverem, už vyžadují připojení; příkladem je třeba synchronizace a úprava položek.

Rozdílem oproti přihlášení je tedy potřeba komunikace se serverem. Zatímco při přihlášení se uživatel autentizuje vůči serveru a od něj teprve získává všechna data, při odemykání už přihlášený je, lokálně má uloženou cache dat trezoru a autentizuje se také pouze lokálně.

Proces odemykání nabízí několik různých metod, jakým odemknutí provádět. Konkrétní metody se ale liší v závislosti na platformě. Základní typy jsou:

- PIN
- Biometrie

- Hlavní heslo

  Stejné jako při přihlášení, ale správnost hesla se kontroluje proti lokální hashi a bez zadání emailové adresy.

[22, /unlock-with-pin][22, /biometrics]

2.3.1.1 PIN

Odemknutí PINem je v principu podobné zadávání hlavního hesla. Uživatel si zvolí PIN, který při odemykání zadá. Na rozdíl od hlavního hesla je pro PIN omezený počet pokusů. Po pátém neplatném pokusu je uživatel odhlášen a musí se opět přihlásit.

Na sílu PINu nejsou z pohledu aplikace kladeny žádné požadavky. Délka může být i 1 znak a komplexitu voleného PINu aplikace také nekontroluje. Abeceda znaků, které lze použít je navíc pro mobilní aplikace omezená pouze na číslice. Pro ostatní aplikace jsou povoleny všechny znaky.

Pro PIN lze volit mezi dvěma variantami. První je, že PIN lze zadávat vždy, i po restartování aplikace. Druhá varianta je, že odemykání PINem je aktivní pouze do restartování aplikace. Po restartu je v tomto případě potřeba zadat nejprve hlavní hesla, další odemčení mohou již probíhat opět PINem. [22, /unlock-with-pin]

2.3.1.2 Biometrie

Biometrickou autentizaci vždy zprostředkovává systém, na kterém aplikace běží. Samotná aplikace pouze volá API a nedostává žádné informace ohledně biometrie, pouze zda se autentizace zdařila.

Konkrétní možnost biometrie je tudíž spjata také s tím, co systém nabízí. Pro mobilní aplikace lze použít otisk prstu či odemknutí obličejem. Pro desktop je možnost na Windows použít Windows Hello a pro MacOS Touch ID. U rozšíření prohlížečů je možné také používat biometrii, ta je ale zprostředkovávaná desktop aplikací. Je tak potřeba mít přihlášený účet i v desktop aplikaci a pak teprve nastavit odemykání v rozšíření prohlížeče. [22, /biometrics]

2.4 Nabízené funkcionality

V této sekci popisuji, jaké funkcionality jsou nástrojem nabízeny a jak se liší dle předplatného. Dále pak možnosti organizací a jak funguje „Send“.

2.4.1 Druhy předplatného

Služba Bitwarden má několik úrovní rozdělených dle předplatného. Základem je předplatné zdarma. To zahrnuje klíčové funkcionality a pro základní použití je dostačující.

Pro rozšiřující funkcionality je ale potřeba zvolit některé z nabízených předplatných. V základu jsou předplatné rozdělené na osobní a firemní. [22, /about-bitwarden-plans]

2.4.1.1 Osobní

Tyto předplatné jsou určeny hlavně pro osobní použití. Spadá mezi ně základní plán zdarma. Jeho rozšířením je premium předplatné, to zpřístupňuje více metod 2FA a také například šifrované přílohy.

Pro osobní předplatné jsou k dispozici také dvě možnosti předplatných pro organizace. Základem je opět předplatné zdarma, které umožní mít v organizaci dva členy. Jedním tak bude vlastník a může přizvat ještě jednoho uživatele. Každý uživatel může být vlastníkem jedné bezplatné organizace.

Druhou možností je rodinná organizace. Ta umožňuje mít v organizaci 6 členů a zároveň každý z nich automaticky získává předplatné. [22, /about-bitwarden-plans]

2.4.1.2 Firemní

Firemní předplatné nabízí dvě možnosti pro organizace. Obě možnosti všem členům opět zpřístupní veškeré premiové funkcionality. Organizace již pro tyto předplatné nejsou omezeny velikostí a cena se stanoví dle počtu členů.

První z nich je Teams předplatné. To zpřístupňuje speciální nástroje jako například záznamy událostí a API pro správu organizace.

Vyšším předplatným je Enterprise, které zahrnuje vše, co nabízí Teams. Navíc ale zpřístupňuje další funkcionality v rámci organizace. Konkrétně SSO (Single Sign-On), nastavování politik nebo Admin Password Reset. [22, /about-bitwarden-plans]

2.4.2 Send

Send je Bitwarden funkcionalita, která umožňuje bezpečně posílat text a soubory. Příjemcem může být každý bez nutnosti Bitwarden účtu. Účet potřebuje pouze ten, kdo Send vytváří.

Send se po vytvoření zašifruje vygenerovaným klíčem a uloží se na server. Pro uživatele je vygenerovaný odkaz, který lze využít pro přístup k datům. Součástí tohoto odkazu je i klíč pro dešifrování. Ten je v nezašifrované podobě přístupný pouze uživateli a dále samozřejmě všem s kým se ho rozhodne sdílet. Server ale k nezašifrovanému klíči přístup nemá. Samotná data se po získání ze serveru dešifrují v zařízení toho, kdo k datům přistupuje využitím klíče přítomného v URL [22, /about-send]

Každý Send splňuje hlavní 3 vlastnosti:

- E2EE

  Data jsou šifrovaná při vytvoření a dešifrovaná klíčem, který je součásti URL.

- Pomíjivost

  Pro každý Send lze nastavit datum smazání, po jehož vypršení je příslušný Send smazán. Maximální nastavitelná doba je 31 dní.

  Kromě datumu smazání lze nastavit i počet maximálních přístupů a datum vypršení platnosti. Jejich překročením se Send nesmaže, ale nebudou přístupné přes URL, ale pouze přes klientské aplikace vlastníkovi.

- Flexibilní nastavení soukromí

  Pro Send lze příjemcům skrýt tvůrcovu emailovou adresu. Dále lze nastavit speciální heslo, kterým je navíc Send chráněn. Pro zobrazení je pak po příjemci požadováno a nestačí mu pouze znalost URL. Heslo je kontrolováno na straně serveru, takže bez hesla nikdo nezíská ani šifrovaná data.

[22, /about-send]

2.4.3 Organizace

Uživatelé se mohou uskupovat do organizací. Ty členům umožňují sdílet položky z trezoru. Členové nemusí sdílet vše a vybírají, které položky budou přístupné v rámci organizace. Organizace jsou vytvořeny uživatelem, který je pak jejím vlastníkem. Vytvoření je jedna z operací přístupných pouze z web vault aplikace. Uživatel může být členem libovolného počtu organizací, kapacita organizace se potom odvíjí od předplatného vlastníka. [22][/about-organizations]

Pro organizaci je vygenerován symetrický klíč obdobně jako je tomu pro uživatele. Pokud chce uživatel sdílet údaje v rámci organizace, potřebuje mít symetrický klíč organizace. Sdílení údajů je možné díky využití asynchronního šifrování algoritmem RSA-2048. Pro bezpečné doručení se symetrický klíč šifruje uživatelovým veřejným klíčem při vstoupení do organizace. Pro uživatele pak server ukládá množinu šifrovaných klíčů, jeden pro každou organizaci jejíž je členem. V klientské aplikaci se pak ze zadaného hesla vytvoří hlavní klíč, kterým se dešifruje generovaný symetrický klíč, tím se následně dešifruje soukromý klíč a tím symetrický klíč organizace. [26]

2.4.3.1 Admin Password Reset

Admin password reset je funkcionalita dostupná pro organizace v Enterprise předplatném. Pokud je aktivní, umožňuje administrátorům v rámci organizace obnovovat hesla uživatelů. K tomu, aby uživateli bylo možné heslo obnovit, je potřeba, aby se k této funkcionalitě zapsal (enrollment).

Uživatelé se mohou buď zapisovat sami, nebo lze pro organizaci nastavit auto-enrollment politiku. Tato politika má za následek, že všichni nově přidaní členové budou automaticky zapsáni pro admin password reset. Uživatelé, kteří již členy byli, stále musí zápis provést ručně. Pokud je tato politika aktivní jsou na to uživatelé před připojením řádně upozorněni.

Aby organizace mohla obnovovat heslo uživatelům, znamená to, že musí mít přístupný jejich symetrický klíč. Ten je při zápisu k této funkcionalitě zpřístupněn tak, že je nejprve šifrován veřejným klíčem organizace a poté poslán serveru. K tomuto dochází v klientské aplikaci při zápisu, a to buď při přijmutí pozvánky do organizace v případě aktivního auto-enrollment nebo při zápisu ručně. Ze strany služby je dodrženo Zero Knowledge Encryption, ke klíči má možnost přistoupit pouze organizace. [22, /admin-reset]

2.5 Životní cyklus uživatelského účtu a synchronizace dat

Práce s daty uživatele je velice důležitou částí služby. V této části popíši, jak se mění během práce s nástrojem a jak se synchronizují se serverem.

2.5.1 Registrace

Registraci zajišťuje api server. Server nejprve nemá žádné údaje uživatele, uživatel zatím nemá vytvořený účet. Registrace začíná v některé z klientských aplikací vyplněním registračního formuláře. Ve formuláři uživatel vyplní potřebné údaje, především zadá email a své hlavní heslo. Volitelně může vyplnit i nápovědu k heslu, ta ale není povinná. Potvrdí ToS (Terms of Service) a odešle formulář.

V klientské aplikaci se provede validace. Ta zahrnuje jednoduchou kontrolu, zda email obsahuje zavináč. U hesla se kontroluje pouze zda má alespoň 8 znaků, další kontrola se v mobilní aplikaci neprovádí. Například pro desktop aplikaci je ale kontrola síly provedena a pokud je heslo slabé, je uživatel varován. Také se v ní kontroluje, zda není nápověda k heslu nastavená na samotné heslo, což opět v mobilní aplikaci chybí.

Následně se nastaví KDF funkce na PBKDF2-HMAC-SHA256 a 100 000 iterací. Vytvoří se hlavní klíč z předešlých údajů. Vygeneruje se a zašifruje symetrický klíč. Následuje vytvoření hashe hesla pro serverovou autentizaci. V poslední řadě se vygeneruje RSA pár klíčů.

Ze všech předchozích informací se vytvoří žádost, která se pošle serveru. Při žádosti může dojít k požadavku o ověření CAPTCHA. V tom případě se zašle žádost znovu, nyní se zahrnutým CAPTCHA tokenem.

2.5.2 Autentizace

Přihlášení uživatelů zajišťuje identity server. Pro přihlášení existují dvě možnosti. Buď přihlášení zadáním hesla a uživatelského jména, které je dostupné v základu, nebo přihlášení pomocí SSO, které je dostupné v enterprise předplatném.

2.5.2.1 Běžné přihlášení

Přihlášení zadáním hesla a emailové adresy je založeno na protokolu OAuth 2.0 (Open Authorization 2.0). Konkrétně je použitý resource owner password credentials grant (password grant). [34]

Samotné heslo se ale serveru neposílá, pro server je předložený pouze hash. Ten vzniká z hlavního klíče a samotné heslo je použito jako sůl. Při vytváření hashe určeného pro serverovou autentizaci je použita 1 iterace. Bezpečnost hashe tak závisí primárně na hlavním klíči, který vzniká z hlavního hesla a emailové adresy, ve výchozím stavu s použitím 100 000 iterací. [35][26]

2.5.2.2 Přihlášení pomocí SSO

Enterprise předplatné zpřístupňuje přihlášení pomocí SSO. Může být využitelné ve firemním prostředí, kde umožňuje použít existující Identity Provider. Podporované protokoly jsou OIDC (OpenID Connect) a SAML 2.0 (Security Assertion Markup Language 2.0).

SSO funguje pouze pro autentizaci serveru, je ale oddělené od samotného zabezpečení trezoru, protože je potřeba splnit Zero Knowledge Encryption. Proto je pro samotné zabezpečení trezoru stále použité hlavní heslo, které je plně oddělené od SSO. Jelikož je potřeba dešifrovat symetrický klíč, je nutné, aby toto heslo uživatel zadal. Po autentizaci přes SSO se aplikace v podstatě dostává do stavu zamčené aplikace, kde zadáním hlavního hesla uživatel aplikaci odemkne. [22, /about-sso]

2.5.3 Synchronizace

Synchronizace dat probíhá hromadně a posílají se během ní ze serveru veškerá synchronizovaná data najednou. Konkrétně se posílají administrativní data, to znamená data spojená s účtem uživatele. Dále se také posílají všechna Vault data.

Synchronizace probíhá vždy při přihlášení a pak v pravidelných intervalech. Pro web vault je synchronizace prováděna v reálném čase, a ne po intervalech. Vždy je v klientských aplikacích možné synchronizaci spustit také manuálně.

Data jsou posílána ve formátu JSON. Samotná struktura šifrovaná není, šifrují se pouze konkrétní citlivé údaje, stejně jako při ukládání dat lokálně v zařízení.

Protože klientské aplikace fungují jako tencí klienti, veškeré operace s položkami trezoru jsou ihned do serveru propisovány. Například při vytvoření libovolné položky v trezoru se automaticky vytvořená data nejprve ukládají na server. Bez možnosti komunikovat se serverem položku pouze lokálně nelze vytvořit. Stejně tak tomu je i při editaci a smazání položek. [22, /vault-sync]

2.5.4 Uložené údaje

Z administrativních dat jsou kromě údajů poslaných při registraci v serverové databázi ukládány údaje vytvořené službou pro potřeby fungování účtu. Například jsou to data spojená s prémiovým předplatným nebo 2FA, konkrétně poskytovatel a recovery code.

Recovery code slouží k odstranění 2FA z účtu v případě, že by uživatel přišel o možnost použití druhého faktoru, například ztratil zařízení na něj vázané. Zobrazit si ho uživatel může kdykoliv ve web vault aplikaci. Uložený je v databázi nezabezpečeně jako prostý text. Teoreticky není důvod uživateli zobrazovat kód vícekrát, bezpečnějším řešením by tak bylo ukládat pouze jeho

hash. Na druhou stranu, pokud by byla kompromitována databáze, je obejití MFA irelevantní, protože útočník už má přístup k datům, které by mělo chránit. [22, /two-step-recovery-code]

Vymazat účet lze skrze web vault aplikaci. Při vymazání jsou veškerá data spojená s účtem uživatele odstraněna z databáze. [22, /delete-your-account]

Kapitola 3

Analýza bezpečnosti nástroje

V této kapitole popíši konkrétní nálezy z provedené analýzy.

3.1 Útok na lokální uložiště

Klientské aplikace z různých důvodů ukládají uživatelská data na disku, mezi těmito daty jsou často i velmi citlivá data. Ukládat se takto mohou jak administrativní data, jako například uložený email, lokální hash z hesla nebo šifrovaný symetrický klíč, tak i data trezoru. Data trezoru jsou ale uložena vždy pouze zašifrovaná.

Konkrétní uložená data jsou pak závislá na stavu aplikace. Například pokud je uživatel přihlášený, ukládají se lokálně všechna data trezoru. Odemknutím účtu uživatel poskytne klíč a není potřeba jakkoliv komunikovat se serverem pro získání dat, stačí je pouze dešifrovat z uložiště. Další rozdíly v tom, co je ukládáno, mohou být dány například zvoleným způsobem uzamykání trezoru.

Všechny klientské aplikace používají k persistenci dat uživatelů souborové databáze nebo JSON soubory. Použitá databáze je závislá na konkrétní aplikaci, ale struktura uložených dat zůstává stejná napříč platformami. Pro desktopové aplikace se data ukládají přímo jako soubory ve formátu JSON. Zbylé aplikace využívají různé databáze, ale samotná struktura dat je pro ně podobná.

Použité soubory se zpravidla ukládají do pro jednotlivé platformy standardních lokací určených pro data aplikací. Pro Windows aplikaci (standardní instalace, ne verze z Windows Store) je to například `%APPDATA%\Bitwarden\data.json`. Data jsou zde uložena v Roaming aplikačních datech, z čehož plyne, že tato citlivá data mohou opustit počítač uživatele, pokud by počítač byl připojen do Active Directory. Toto si běžný uživatel jistě neuvědomí, ani nad tím nebude uvažovat. Je pravda, že data na Domain Serveru by měla být dobře chráněna. Není ale dobré, že je zde možnost kopírování důvěrných dat z uživatelského zařízení bez uživatelova explicitního souhlasu. [22, /data-storage]

3.1.1 Získání souborů a dat z nich

Než přejdu k rozboru samotných dat obsažených v souborech, je potřeba probrat, jak získat samotné soubory a z nich následně konkrétní údaje. Pro klientské aplikace jsou známá místa uložení jejich databázových souborů, a to přímo z dokumentace. Ve většině případů je k přístupu k nim potřeba disponovat přístupovými právy uživatelského účtu pod nímž aplikace běžela, nebo administrátorského účtu. Speciálním případem je Android aplikace. Pro tu jsou data uložena pod `/data/data/com.x8bit.bitwarden/`, tedy interním uložišti zařízení přiděleném pro danou

Android aplikaci. Pro Android zařízení ale může do interního uložiště přistupovat pouze samotná aplikace, ne však uživatel.

Z těchto dat aplikací je možné vytvořit zálohu zprostředkovanou systémem. Tu si uživatel může vyžádat a teoreticky se tak k datům dostat skrz jejich zálohu. To však pro Bitwarden aplikaci nelze využít, protože je tato funkcionalita explicitně vypnuta v rámci manifestu aplikace a je tudíž v tomto ohledu chráněna. Data ze zařízení nelze získat tak jednoduše jako na ostatních platformách. Nejpřímočařejším řešením je získání root oprávnění. To je ovšem na Android zařízeních také obtížné. [22, /data-storage][36]

Po získání samotného souboru je z něj potřeba extrahovat konkrétní data. Protože však každá aplikace používá jiný formát, je tento proces závislý na konkrétních případech. Pro JSON soubory je toto přímočaré, tento formát je rozšířený a také dobře lidsky čitelný. Údaje v něm lze tedy nalézt a vyčíst poměrně snadno. Problémovější jsou aplikace, které používají jiné formáty. Především jsou to rozšíření prohlížečů, které používají uložiště poskytované právě prohlížečem, a také Android aplikace, která používá LiteDB souborovou databázi. Pro Google Chrome (a další prohlížeče založené na Chromium jádře) je pro ukládání používaná LevelDB a pro Firefox IndexedDB. [37][38]

V rámci práce jsem napsal python skript `bitwarden2hashcat.py`, který napomáhá překonat strukturu těchto různých databází a získá z nich přímo potřebná data. Zároveň data převede do formátu očekávaného nástrojem Hashcat.

3.1.2 Útok hrubou silou na hash uživatelova hesla

Pokud je uživatel přihlášený, je jedním z uložených údajů lokální hash z hlavního hesla. Vzhledem k tomu, že je přítomný také uživatelův email, jsou zde veškeré údaje potřebné k pokusu o prolomení hesla hrubou silou. Pro maximální optimalizaci rychlosti prolamování jsem k tomuto zvolil nástroj Hashcat. Ten je určený právě k prolamování hashí a provádí to hardwarově optimalizovaným způsobem, kdy umožňuje využívat výpočetního potenciálu grafických karet. Podpora pro prolamování konkrétního typu hashe, tzv. hash mód, je zajištěna implementací v pluginech. Takový plugin již v nástroji existoval, byl však zastaralý a pro aktuální verze Bitwarden nefunkční.

Samotné implementace Hashcat pluginů se skládají ze dvou částí. První částí je modul, ten obsahuje přípravu rozhraní, nastavení pluginu a parsování dat pro prolamování ze vstupu. Konkrétní kryptografické operace potom hrají roli až v druhé části, v tzv. kernelu. [39]

Pro vytváření hashe z hesla se v nástroji Bitwarden využívá funkce PBKDF2-SHA256, a to hned 2krát. Pro první aplikaci funkce je vždy použito 100 000 iterací s emailovou adresou jako solí, čímž je vytvořen hlavní klíč. Z hlavního klíče se pak mohou tvořit 2 různé hashe, na základě jejich účelu. Jeden slouží pro lokální autentizaci a druhý pro autentizaci na serveru. Pro jejich odlišení se mění staticky daný počet iterací při druhém použití derivační funkce. Ta přijímá hlavní klíč a jako sůl hlavní heslo. Počet iterací má pak pro serverový hash hodnotu 1 a pro lokální 2. Dříve byla tato hodnota 1 pro oba případy, její změnou se právě stal Hashcat plugin nepoužitelným pro aktuální verze nástroje Bitwarden. [26][35]

V pluginu se spoléhalo na případ starších verzí a předpokládala se vždy pouze jedna iterace funkce. Pro opravu tedy bylo zapotřebí přidat druhý parametr umožňující volbu počtu iterací pro druhou funkci, v nynější verzi 2. Oprava zároveň umožňuje i zpětnou kompatibilitu při volbě 1 iterace, a také je flexibilnější do budoucnosti, pokud by se měl počet opět změnit. Při provedení testů nové verze s nastavením 1 iterace nebyl oproti původní verzi zaznamenán žádný dopad na výkonnost pluginu. Oprava již byla přidána do oficiálního repositáře Hashcatu. Konkrétní provedené změny lze vidět přímo v pull requestu. [40]

Hlavní klíč má velikost 256 bitů, což znamená, že provádět útok hrubou silou pouze na druhé použití funkce je nereálné. Samotný hlavní klíč je pak tvořen z hlavního hesla opět funkcí PBKDF2 s dostatečným počtem iterací. Zdrojem bezpečnosti hashe je tak právě heslo, které by mělo být silné a síla je ze strany Bitwarden nástroje vyžadována. Výsledný hash lze tudíž pokládat za relativně bezpečný i přes hardwarovou optimalizaci prolomení a pouze 2 použité iterace pro

druhou aplikaci funkce. V kombinaci se skutečností, že by útočník musel nejprve získat tato lokální data, což znamená mít přístup k uživatelovu účtu na jeho zařízení, není přítomnost hashe v datech závažná. Pokud by útočník k účtu uživatele přístup měl, má k dispozici snadnější prostředky, jak uživatele poškodit.

Testy prolamování pomocí nástroje hashcat jsem prováděl na několika zařízeních. Pro všechny testy jsem použil vlastně zkompilovanou verzi nástroje[1]. Specifikace jednotlivých zařízení jsou uvedeny v tabulce 3.1. Hashcat poskytuje spuštění v tzv. benchmark módu, který pro vybraný plugin měří jeho potenciální rychlost prolamování (hashrate). Rychlostí prolamování je myšlen počet vyzkoušených hash hodnot za sekundu. Benchmark mód se může rychlostí lišit od skutečného prolamování v závislosti na konkrétním vstupu a použitém útoku. Rychlost se však běžně příliš neliší a je přibližně odpovídající většině použití.

Tabulka 3.2 obsahuje naměřené rychlosti pro plugin prolamující Bitwarden lokální hash včetně benchmark módu spuštěného jak na CPU (Central Processing Unit), tak GPU (Graphics Processing Unit). Z tabulky lze vidět několikanásobné zvýšení výkonu při použití GPU oproti CPU. Měření byla prováděna na běžných zařízeních, při využití stroje vytvořeného speciálně na prolamování by rychlost byla ještě znatelně vyšší.

Tabulka 3.1 Specifikace zařízení použitých při testování hashcat pluginů

	Zařízení 1	Zařízení 2	Zařízení 3
GPU	NVIDIA GeForce GTX 960	Intel(R) HD Graphics 520	NVIDIA GeForce GTX 1060 3GB
CPU	Intel(R) Core(TM) i5-4460	Intel(R) Core(TM) i5-6200U	Intel© Xeon© CPU E3-1245 v6
Operační systém	Windows 10 Education	Windows 10 Pro	Ubuntu 20.04 focal (x86-64)
Použitý kompilátor	gcc version 10.2.1	gcc version 10.2.1	gcc version 9.3.0

Tabulka 3.2 Tabulka rychlostí prolamování hashe z hesla pomocí Hashcat pluginu

	Zařízení 1	Zařízení 2	Zařízení 3
Benchmark GPU [H/s]	3539	255	6000
Benchmark CPU [H/s]	316	176	-
Masked útok pro náhodné heslo [H/s]	3548	265	5970

3.1.3 Útok hrubou silou na zamykání pomocí PINu

Potřebné údaje pro tento útok nejsou v rámci uložených dat přítomny vždy, když je uživatel přihlášený, ale závisí na volbě metody odemykání. Konkrétně jsou závislé na zvolení zamykání PINem. Zamykání PINem má navíc dostupné dvě různé možnosti. Buď se při restartu aplikace vyžaduje opět hlavní heslo, nebo lze stále používat nastavený PIN. Tato volba ovlivňuje, jaká data je potřeba perzistentně ukládat za účelem ověření uživatele.

Při zamykání PINem je vždy podstatná hodnota označovaná jako `PinProtectedKey`. Ta, jak název napovídá, obsahuje hlavní klíč šifrovaný klíčem odvozeným z PINu. Kde je tato hodnota ukládána pak závisí na zvolené variantě zamykání PINem. Při volbě vyžadování hlavního hesla po restartu se ukládá `PinProtectedKey` v paměti. U druhé možnosti je `PinProtectedKey` perzistentně ukládán v lokálních datech.

Při šifrování symetrického klíče se používá schéma Encrypt-then-MAC. Odvozený klíč z PINu vzniká stejným způsobem jako hlavní klíč. Proces je stejný, ale místo hodnoty hesla je zde použitý PIN, klíč je pak opět rozšířen na 64 B. Jak bylo rozebráno v sekci 2.2.1.1, odvozený klíč tak obsahuje reálně dva klíče, jeden šifrovací a jeden pro výpočet MAC.

Ověření MAC udává správnost zadaného PINu. Hodnota `PinProtectedKey` obsahuje jak šifrovaný klíč, tak příslušný IV i MAC. Ze šifrovaných dat a IV se vypočte MAC pomocí odvo-

[1] Dostupná z `https://github.com/Greexter/hashcat/tree/986306728772f7425453df7844f27852105c3d39`

zeného klíče ze zadaného PINu. Porovnáním nově vzniklé hodnoty MAC s uloženou se následně určí správnost MAC klíče a tím i PINu a šifrovacího klíče.

Nyní se znalostí hlavního klíče lze dešifrovat symetrický klíč (šifrovaný je uložený jako hodnota `encKey`) a pomocí toho data trezoru. Při přihlášení jsou tato data všechna ukládaná v lokálních datech v uložišti zařízení. Pokud je ale aplikace uzamčena, nutně to znamená, že je uživatel přihlášený, protože bez přihlášení nastavit zamčení trezoru nelze. Zvolením zamykání PINem bez hlavního hesla při restartu jsou tudíž veškeré potřebné hodnoty přítomné v lokálních datech.

Samotná přítomnost těchto dat by nebyla natolik závažná, stále jsou všechny údaje šifrované a jediná možnost útoku je prolomení hrubou silou. Tu ale zjednodušuje fakt, že pro PIN nejsou ze strany aplikace kladeny žádné podmínky na sílu. Není vyžadována žádná délka PINu ani aplikace nedává žádné doporučení pro vytvoření silného PINu. Pro Android aplikaci jsou navíc povoleny pouze číselné znaky. Samotný název „PIN“ pak implikuje použití číslic. Při volbě číselného PINu myslím, že znatelná část uživatelů bude vybírat formát na který jsou zvyklí například z platebních karet, tedy čtyřciferný PIN. Takové hodnoty budou jistě snadno prolomitelné.

K demonstraci možnosti prolomení a jeho rychlosti jsem podobně jako v sekci 3.1.2 vytvořil plugin pro Hashcat. Plugin je dostupný pod číslem módu 90000 a pro získání hodnot z dat aplikací v potřebném formátu lze použít přiložený skript `bitwarden2hashcat.py`. Získání hodnot pro PIN je potřeba při volání skriptu určit přepínačem `--mode pin`.

K prolamování jsem zvolil dvě hodnoty PINu, jeden čtyřciferný a jeden šesticiferný, oba volené náhodně. Myslím, že tyto volby budou u uživatelů zastoupeny nejčastěji. Hodnoty nepředpokládají další informace o volených PINech, jako například volení dle různých vzorů. K těm určitě může docházet a dále zvyšují rychlost prolomení. Při offline útoku hrubou silou na čistě číselné PINy je však očividně prolomení zvládnutelné v zanedbatelné době i zkoušením všech možností, a to i pro méně výkonný hardware, jak lze vidět z tabulky 3.3.

V tabulce lze také vidět znatelný rozdíl mezi rychlostí prolamování čtyřciferných a šesticiferných PINů. Toto je způsobeno principem výpočtu na GPU, který pro vysokou rychlost prolamování silně využívá paralelních výpočtů. Protože ale čtyřciferných PINů není dostatek pro plné využití potenciálu paralelismu, výsledná rychlost prolamování klesá. [41]

Tabulka 3.3 Prolamování zamykání PINem pomocí nástroje Hashcat

	Benchmark		Čtyřciferný		Šesticiferný	
	GPU [H/s]	CPU [H/s]	Rychlost [H/s]	Doba prolomení [s]	Rychlost [H/s]	Doba prolomení [s]
Zařízení 1	3540	314	429	19	3570	175
Zařízení 2	253	156	99	62	246	1449
Zařízení 3	6109	-	630	13	5960	56

Myslím, že tato metoda zamykání je z principu svého navržení nebezpečná. Je velice podobná použití hlavního hesla. Důvodem, proč zvolit PIN místo hlavního hesla, tak zřejmě bude volba jednoduššího nebo zapamatovatelnějšího tajemství. Efektivně tak PIN je pouze oslabeným hlavním heslem.

Samozřejmě je ale důležitý fakt, že toto je použité pouze lokálně. PIN tak může být stále dobrou volbou pro někoho, kdo nemá obavu z toho, že by někdo mohl získat přístup k lokálním datům aplikace. Na nebezpečí použití PINu jsem vývojáře upozornil a navrhl alespoň lépe informovat uživatele. Bitwarden zatím do dokumentace přidal poznámku upozorňující uživatele, aby nevolili předvídatelné hodnoty jako svůj PIN. Upřednostňovat jiné metody ale v současné době neplánuje.

3.2 MitM útok na SSO

Při přihlašování pomocí SSO uživatel musí mít nastavené hlavní heslo, nezávislé na SSO přihlášení. Pokud se tedy přihlašuje pomocí SSO poprvé, je nutné ho nastavit. V rámci nastavení hesla se také pro uživatele vytvoří symetrický klíč. Proces probíhá podobně jako zjednodušená

registrace, bez volby emailové adresy. Během tohoto nastavení si aplikace vyžádá od serveru informace o organizaci, pro kterou se přihlašuje přes SSO, což zahrnuje i platné politiky.

Pokud je aktivní politika admin password reset auto-enrollment, je potřeba, aby se po nastavení hesla uživatel zapsal k příslušné funkcionalitě. Základem tohoto zápisu je, že je nový symetrický klíč zašifrován veřejným klíčem dané organizace a poslán serveru. Potřebný veřejný klíč pro zašifrování získá klientská aplikace ze serveru.

V tomto místě je prostor pro zneužití funkcionality vyměněním veřejného klíče za jiný. Pokud by se útočník zmocnil serveru, či se mu povedlo komunikaci narušit MitM útokem, pak má možnost ukrást uživatelův klíč. Při vyžádání klíče pošle klientské aplikaci svůj veřejný klíč a ta mu zpět pošle uživatelův symetrický klíč zašifrovaný tímto vyměněným klíčem.

Hlavním nedostatkem je, že uživatel nemá žádnou možnost kontroly, že obdržel skutečně správný klíč. Samotný zápis k funkcionalitě je pro něj neprůhledný a o tom, jaký veřejný klíč obdrží, nemá žádnou informaci.

Pokud je aktivní auto-enrollment, je o aktivní politice uživatel při nastavení hesla varován. Myslím si ale, že běžný uživatel ve firemním prostředí toto varování nebude považovat za podezřelé, a tak pro zamezení ukradení klíče nebude stačit.

Pro demonstraci možnosti zneužití zranitelnosti jsem napsal jednoduchý server. Jeho funkcionality jsou zjednodušené jen na zneužití procesu SSO přihlášení. Aplikace komunikuje s ním místo normálního serveru, pro zjednodušení toho lze docílit nastavením přímo v klientské aplikaci. Server pak aplikaci předloží potřebná data a svůj veřejný klíč, po dokončení procesu přihlášení s nastavením hlavního hesla může uživatel pracovat se svým trezorem. Jakákoliv nová položka, kterou uživatel vytvoří v trezoru je ale serverem při uložení dešifrovaná, protože získal uživatelův symetrický klíč. Takto server může narušit Zero Knowledge Encryption a získat přístup k datům.

Podobná zranitelnost s označením BWN-01-008 byla nahlášena v roce 2018 v rámci auditu nástroje. Zde se jednalo o možnost získání symetrického klíče organizace při přijmutí nového člena. Navrženým a použitým řešením zde bylo zavedení fingerprint phrase pro každého uživatele. Jedná se o posloupnost slov, která je vygenerována na základě uživatelova veřejného klíče. Ta poté slouží k umožnění autentizace uživatele jinou cestou než přes nástroj, například osobně nebo telefonicky. [30]

Při nahlášení zranitelnosti vývojářům jsem navrhl použít podobné řešení i zde a zavést fingerprint phrase pro organizace. Ta by pak umožnila ověřit správnost veřejného klíče při zapisování k admin password reset. Vývojáři uznali, že zranitelnost se zdá být zneužitelná podobně jako BWN-01-008 a uvedli, že zváží použití fingerprint phrase pro organizace jako řešení.

3.2.1 Resetování čítače pokusů pro zamykání PINem

Jak už bylo rozebráno, při zamykání PINem se v uložišti zařízení ukládají citlivá data, která umožňují útok hrubou silou. Aby takovému útoku bylo zabráněno z uživatelského rozhraní, je používán čítač chybných pokusů. V případě nástroje Bitwarden znamená 5 chybných pokusů zadání PINu odhlášení z účtu, tudíž i znemožnění dalších pokusů PINu. Použití čítače je dobré omezení, které zamezí manuálním útokům hrubou silou přímo z uživatelského rozhraní, především pak příležitostným útokům, kdy by útočník mohl mít dočasný fyzický přístup k nechráněnému zařízení. Tento útok je navíc snadno proveditelný i bez technických znalostí.

Pro Desktop klientskou aplikaci má však implementace čítače nedostatek. Na rozdíl od Android aplikace je čítač ukládán pouze v paměti a není uložen perzistentně. To znamená, že pokud se paměť přemaže, čítač se obnoví znovu na základní hodnotu, tedy na nulu. Má-li útočník možnost takto paměť obnovit, má pak neomezené množství pokusů, protože čítač efektivně obejde.

Jednoduchou demonstrací, jak tohoto dosáhnout, je vypnutí a zapnutí aplikace. To zajistí novou paměť rovnou celé aplikace, tudíž i včetně tohoto čítače. Podobného výsledku lze docílit i znovunačtením uživatelského rozhraní. Funkce obnovení rozhraní je přístupná přes horní menu *View>Reload*, případně klávesovou zkratku *Ctrl+Shift+R*. Tato metoda lze však použít pouze,

pokud je zámek nastaven na zamykání PINem bez použití hlavního hesla při restartu. Pokud tomu tak není, je z principu tohoto zámku po uživateli vyžadováno heslo a PIN již hádat nelze.

Zavedení perzistence čítače snadno odstraní možnost tohoto útoku z uživatelského rozhraní aplikace. Tak je čítač ošetřen právě ve zmíněné Android aplikaci, kde se při změně ukládá do lokálních dat aplikace.

Obejít lze ale i varianta PINu s použitím hlavního hesla při restartu, ale ne už tak přímočaře. Aplikace je psána v Electronu, je tudíž založena na jádru Chromium. V základu, pokud toto vývojáři nedeaktivují, je možné otevřít vývojářské nástroje, které jsou přítomné v prohlížečích. V rámci těch je pak možné aplikaci za běhu ladit, včetně pozastavování a upravování hodnot v paměti. Protože tyto nástroje v aplikaci jsou přístupné, stačí nalézt místo, kde se nastavuje čítač v paměti, a pomocí ladění ho přenastavit na vyhovující hodnotu.

V novějších verzích (začínaje verzí 1.31.0) je nově možné mít až 5 současně přihlášených či uzamčených účtů. Tato funkcionalita také umožňuje obejít čítač pokusů následovně:

1. Uživatel až 4krát vyzkouší zadat PIN.
2. Přepne na jiný libovolný účet, jediný požadavek je, aby byl odemčený. K tomu může snadno využít svůj nebo vytvořit dočasný.
3. Následně přepne opět na uzamčený, kterému hádá PIN.
4. Může následovat bodem 1, čítač se obnovil na 0.

Stejná zranitelnost byla objevena i v článku [42] ve dvou různých správcích hesel RoboForm a Dashlane. Zde se na rozdíl od nástroje Bitwarden jednalo o Android aplikace, v té má ale Bitwarden implementaci čítače v pořádku. V případě RoboForm a Dashlane byly PINy čistě 4 číselné, zatímco Bitwarden umožňuje i další znaky. Z tohoto pohledu by tudíž prolomení mělo být obtížnější, ale vše závisí na uživateli, jaký PIN zvolí. V tomto ohledu od nástroje žádné rady ani omezení nedostává.

3.3 Ostatní nálezy

V této sekci popíši místa z analýzy, kde se nejedná přímo o zranitelnosti. Jedná se hlavně o očekávané a dokumentované chování, které ale může být nežádoucí nebo vést k oslabení bezpečnosti služby.

3.3.1 KDF iterace

Počet iterací v používané PBKDF2 je v základu nastavený na 100 000. Nastavení ve web vault aplikaci však umožňuje tuto hodnotu snížit. Pevné minimum pro tuto hodnotu je pak 5 000, která je kontrolována i v kódu a nelze použít nižší hodnotu. Toto opatření dobře zabrání případnému snížení počtu například MITM útokem.

Minimum 5 000 iterací je však pro dnešní standardy příliš nízká. NIST doporučení již z roku 2017 je pro PBKDF2 použít alespoň 10 000 iterací. [11] Vývojáři se ale vyjádřili, že hodnota minima je k dispozici hlavně pro zvýšení výkonu, pokud se k tomu uživatel rozhodne. [16]

Zároveň maximum iterací je nastavené na 2 000 000. Tato hodnota je pro mnoho dnešních zařízení zbytečně nízké maximum. NIST doporučení [11] například pro velmi citlivé klíče nebo systémy, které mají vysoký výkon, doporučuje 10 000 000 iterací. Myslím, že hlavní heslo v kontextu správců hesel lze označit za velmi citlivé, a tak určitě mohou být uživatelé, kteří by chtěli počet iterací nastavit na tuto hodnotu. Ta ale výrazně převyšuje pevně stanovené maximum.

Audit z roku 2018 navíc argumentuje, že dávat uživatelům možnost změny počtu iterací je zbytečné bezpečnostní riziko, obzvláště s tak nízkým minimem. Uživatelé si tak mohou snižovat bezpečnost nástroje s vidinou zvýšení výkonu, který je navíc na většině dnešních zařízení zavádějící. [30]

3.3.2 Možný únik informací icon serveru

Při použití ikon klientská aplikace pošle požadavek icon serveru pro každé URI, které určí za odkazující na webovou stránku. Tento požadavek pak obsahuje název příslušné webové stránky. Tato informace je normálně v rámci Vault šifrovaná, v požadavku je ale potřeba ji posílat v textové podobě. Z hlediska soukromí je tento fakt problematický při používání oficiálního icon serveru.

Tyto dotazy takto mají za následek únik jinak zabezpečených informací, a to jak Bitwarden serveru, tak případně i Cloudflare, jejichž CDN (Content Delivery Network) pro Bitwarden poskytuje cache požadavků. Přestože Bitwarden uvádí, že žádné požadavky pro ikony neloguje, je na zvážení uživatele, zda je pro něj a jeho soukromí tento únik stěžejní. Pokud ano, pak Bitwarden nabízí možnost tuto funkcionalitu vypnout. V Bitwarden dokumentaci je na možnost úniku informací upozorněno. [22, /website-icons]

3.3.3 Nedoporučovaný OAuth 2.0 grant

Při běžném přihlášení je použitý password grant, který pro přihlášení přebírá uživatelovo heslo, a to posílá serveru pro autorizaci aplikace. V případě použití tohoto grantu aplikace pro server vystupuje jako uživatel a předloží jeho uživatelské údaje. Toto je v rozporu se smyslem OAuth 2.0 protokolu, který by právě měl odstranit potřebu předávat přihlašovací údaje aplikaci. Od jeho používání je odrazováno a v aktualizované specifikaci OAuth 2.1 bude odstraněn úplně. [43][34]

Dle vyjádření vývojářů je tento grant použit z praktického hlediska. Pro případy použití klientských aplikací je žádoucí, aby bylo možné udržovat dlouhodobé session, což password grant umožňuje narozdíl například od implicitního grantu. Je tudíž potřeba zvolit grant, který umožňuje vydávání Refresh Tokenu pro obnovu Access Tokenu. Vhodným řešením by bylo použití Auth Code Flow + PKCE (PROOF Key for Code Exchange). To je i navrhováno v issue v Bitwarden github repozitáři, kde je na používaný password grant poukazováno. Volba Code Flow + PKCE je v souladu i s doporučeními OWASP pro nativní aplikace: „*On native apps, code grant should be used instead of implicit grant. When using code grant, PKCE (Proof Key for Code Exchange) should be implemented to protect the code grant*“. Zároveň umožňuje vydávání Refresh Tokenu, takže požadavek dlouhodobých session je možné splnit i s tímto grantem. [44][45]

Přestože je tento grant nedoporučovaný, myslím, že v případě nástroje Bitwarden je jeho použití v pořádku. Grant je nedoporučovaný především proto, aby aplikace, kterou má protokol autorizovat, nezpracovávala přihlašovací údaje uživatele. Předpokládá se, že aplikace nemusí být plně důvěryhodná a poskytnutí přihlašovacích údajů znamená zvýšení prostoru k útokům. [43][34]

Pro Bitwarden je však potřeba zajistit důvěryhodnost klientských aplikací v každém případě už kvůli jejich účelu. Protože jejich prostřednictvím uživatel pracuje s ukládanými daty, musí pro něj aplikace data dešifrovat. K tomu jí uživatel musí předat své hlavní heslo. S heslem tak musí aplikace nakládat vždy a musí být k tomu důvěryhodná, nehledě na zvolený grant protokolu. Pokud by byl použitý jiný grant, uživatel by v aplikaci své hlavní heslo musel zadávat stejně.

3.3.4 Chybějící dokumentace a komplexita kódu použití MAC

Přestože by měl být Encrypt-then-MAC v nových verzích vždy, jak bylo rozebráno v sekci 2.2.1, není o použití MAC v dokumentaci žádná zmínka. Navíc v kódu není MAC explicitně vynucován, použití závisí na dodání klíče. Již v auditu z roku 2018 došlo k falešně pozitivnímu nálezu v ohledu kontroly integrity šifrovaných dat. Jako příčinu falešně pozitivního nálezu tým v reportu uvedl právě matoucí logiku kryptografického kódu a navrhoval, že by čitelnost a komplexita daného kódu mohla být zlepšena. Zároveň chybějící informace v dokumentaci mají za následek obavy samotných uživatelů, jak je upozorněno například v [46]. Zde autor cituje dokumentaci ohledně

šifrování, kde je zmíněno pouze, že je použito AES-256-CBC, ne však MAC. Dále uvádí, že s módem CBC by měl být použitý právě i MAC, přestože dle samotné logiky v kódu nástroje použit je. [30]

Kapitola 4
Závěr

Cílem práce bylo zhodnotit bezpečnost nástroje Bitwarden a jeho nakládání s daty uživatelů. V práci jsem popsal procesy spojené s životním cyklem uživatelského účtu a data s nimi související. Z rozboru použitého šifrování a příslušných klíčů vyplývá, že Bitwarden správně zabezpečuje data, a to takovým způsobem, aby splnil předpoklady pro Zero Knowledge Encryption. Právě dodržením tohoto principu je zajištěno, že data uživatele jsou vždy přístupná pouze pro něj. Žádná třetí strana či samotná služba data nemůže dešifrovat, protože klíč, který by toto umožnil nikdy neopustí zařízení, na kterém pracuje s daty uživatel.

V rámci analýzy jsem ale odhalil zranitelnost, která by toto mohla narušit. Jedná se o zranitelnost přítomnou v SSO přihlašování v kombinaci s Master Password Reset funkcionalitou. Princip spočívá v zaměnění veřejného klíče, který v tomto procesu obdrží uživatel. Tato záměna by následně umožnila libovolné třetí straně, která je schopná podstrčit vlastní veřejný klíč, získat potřebný klíč uživatele pro dešifrování dat. Zranitelnost má sice vysoké požadavky a je těžko zneužitelná, pokud by ale k zneužití došlo, mělo by katastrofální následky pro uživatele. O zranitelnosti jsem Bitwarden informoval a aktuálně pracují na nápravě.

Má analýza objevila také další zranitelnosti týkající se mechanismu zamykání PINem. První z nich umožňuje obejít ochranu proti opakovanému zadávání pokusů PINu v Desktop aplikaci. Správně by měl mít uživatel pouze 5 pokusů na zadání PINu a po jejich vyčerpání by měl být odhlášen. Nedostatečná implementace čítače, který toto zajišťuje, však umožňuje jeho vyresetování a tak lze pokračovat v zadávání bez horní hranice.

Druhou zranitelností je ukládání dat využitelných pro prolomení ochrany trezoru hrubou silou do úložiště zařízení. Samotné ukládání těchto dat by nebylo příliš závažné, protože neobsahují žádná nešifrovaná citlivá data. Klíč použitý k šifrování však pochází právě z PINu, pro který aplikace neklade žádné nároky na jeho sílu. Ze samotného principu použití PINu lze předpokládat, že ve většině případů bude snadno prolomitelný hrubou silou. Takové prolomení má opět ničivý dopad na uživatele, jelikož by znamenalo odhalení veškerých jeho ukládaných dat.

Celkově bych přesto bezpečnost nástroje hodnotil jako vysokou. Nalezené zranitelnosti jsou velice specifické a k jejich provedení útočník potřebuje splnit nelehké předpoklady. Zranitelnosti jsou navíc nahlášené vývojářům a ti počítají s jejich opravou.

Bibliografie

1. *Authentication vs. Authorization* [online]. Auth0 [cit. 2022-05-04]. Dostupné z: https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization.
2. TURNER, Dawn M. *Digital Authentication - the basics* [online]. Cryptomathic, 2016-08-01 [cit. 2022-04-26]. Dostupné z: https://www.cryptomathic.com/news-events/blog/digital-authentication-the-basics.
3. DONEGAN, Katie. *5 common authentication factors to know* [online]. TechTarget [cit. 2022-05-08]. Dostupné z: https://www.techtarget.com/searchsecurity/feature/5-common-authentication-factors-to-know.
4. MASSEY, J.L. Guessing and entropy. In: *Proceedings of 1994 IEEE International Symposium on Information Theory*. 1994, s. 204–. Dostupné z DOI: 10.1109/ISIT.1994.394764.
5. BURR, William E; DODSON, Donna F; POLK, Timothy W. *Electronic Authentication Guideline*. 2004. Tech. zpr., NIST Special Publication (SP) 800-63. National Institute of Standards a Technology.
6. WEIR, Matt; AGGARWAL, Sudhir; COLLINS, Michael; STERN, Henry. Testing Metrics for Password Creation Policies by Attacking Large Sets of Revealed Passwords. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. Chicago, Illinois, USA: Association for Computing Machinery, 2010, s. 162–175. CCS '10. ISBN 9781450302456. Dostupné z DOI: 10.1145/1866307.1866327.
7. KELLEY, Patrick Gage; KOMANDURI, Saranga; MAZUREK, Michelle L.; SHAY, Richard; VIDAS, Timothy; BAUER, Lujo; CHRISTIN, Nicolas; CRANOR, Lorrie Faith; LOPEZ, Julio. Guess Again (and Again and Again): Measuring Password Strength by Simulating Password-Cracking Algorithms. In: *2012 IEEE Symposium on Security and Privacy*. 2012, s. 523–537. Dostupné z DOI: 10.1109/SP.2012.38.
8. STATT, Nick. *Best practices for passwords updated after original author regrets his advice* [online]. The Verge, 2017-08-07 [cit. 2022-04-26]. Dostupné z: https://www.theverge.com/2017/8/7/16107966/password-tips-bill-burr-regrets-advice-nits-cybersecurity.
9. WASH, Rick; RADER, Emilee. Prioritizing security over usability: Strategies for how people choose passwords. *Journal of Cybersecurity*. 2021, roč. 7, č. 1. ISSN 2057-2085. Dostupné z DOI: 10.1093/cybsec/tyab012. tyab012.
10. JITHUKRISHNAN. *Top 10 password policy recommendations for system administrators in 2021* [online]. Securden [cit. 2022-04-26]. Dostupné z: https://www.securden.com/blog/top-10-password-policies.html.

11. GRASSI, Paul; NEWTON, Elaine; PERLNER, Ray; REGENSCHEID, Andrew; BURR, William; RICHER, Justin; LEFKOVITZ, Naomi; DANKER, Jamie; CHOONG, Yee-Yin; GREENE, Kristen; THEOFANOS, Mary. *Digital Identity Guidelines: Authentication and Lifecycle Management*. Special Publication (NIST SP), National Institute of Standards a Technology, Gaithersburg, MD, 2017. Dostupné z DOI: https://doi.org/10.6028/NIST.SP.800-63b.
12. CENTER FOR INTERNET SECURITY. Top 10 password policy recommendations for system administrators in 2021. In: [online]. Center for Internet Security, 2021 [cit. 2022-04-26]. Dostupné z: https://www.cisecurity.org/insights/white-papers/cis-password-policy-guide.
13. YILDIRIM, M.; MACKIE, I. Encouraging users to improve password security and memorability. *International Journal of Information Security*. 2019, roč. 18, č. 6, s. 741–759. Dostupné z DOI: 10.1007/s10207-019-00429-y.
14. *How Are Passwords Stored? (5 Methods Used by Developers)* [online]. Patrick Fromaget [cit. 2022-04-24]. Dostupné z: https://infosecscout.com/how-are-passwords-stored/.
15. AUMASSON, Jean-Philippe. *Serious cryptography: A practical introduction to modern encryption*. No Starch Press, 2018. ISBN 978-1593278267.
16. KALISKI, Burt. *PKCS #5: Password-Based Cryptography Specification Version 2.0* [RFC 2898]. RFC Editor, 2000. Request for Comments, č. 2898. Dostupné z DOI: 10.17487/RFC2898.
17. *Password Storage Cheat Sheet* [online]. OWASP [cit. 2022-04-25]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html.
18. KNIERIEM, Brandon; ZHANG, Xiaolu; LEVINE, Philip; BREITINGER, Frank; BAGGILI, Ibrahim. An Overview of the Usage of Default Passwords. In: 2018, s. 195–203. ISBN 978-3-319-73696-9. Dostupné z DOI: 10.1007/978-3-319-73697-6_15.
19. MCCARNEY, Daniel. *Password Managers: Comparative Evaluation, Design, Implementation and Empirical Analysis*. Ottawa, 2013. Dipl. pr. Carleton University.
20. WANG, Luren; LI, Yue; SUN, Kun. Amnesia: A Bilateral Generative Password Manager. In: *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. 2016, s. 313–322. Dostupné z DOI: 10.1109/ICDCS.2016.90.
21. GASTI, Paolo; RASMUSSEN, Kasper B. On the Security of Password Manager Database Formats. In: FORESTI, Sara; YUNG, Moti; MARTINELLI, Fabio (ed.). *Computer Security – ESORICS 2012*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, s. 770–787. ISBN 978-3-642-33167-1.
22. *Bitwarden Help Center* [online]. Bitwarden [cit. 2022-04-08]. Dostupné z: https://bitwarden.com/help/.
23. KENNISON, Shelia M.; CHAN-TIN, D. Eric. Predicting the Adoption of Password Managers: A Tale of Two Samples. In: TMS Proceedings, 2021. Dostupné také z: https://tmb.apaopen.org/pub/5yfwxh2z.
24. PEARMAN, Sarah; ZHANG, Shikun Aerin; BAUER, Lujo; CHRISTIN, Nicolas; CRANOR, Lorrie Faith. Why people (don't) use password managers effectively. In: *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. Santa Clara, CA: USENIX Association, 2019, s. 319–338. ISBN 978-1-939133-05-2. Dostupné také z: https://www.usenix.org/conference/soups2019/presentation/pearman.
25. Server Setup. In: *GitHub* [online]. GitHub, Inc., 2019 [cit. 2022-04-12]. Dostupné z: https://github.com/bitwarden/server/blob/master/SETUP.md.
26. *Bitwarden Security Whitepaper* [online]. Bitwarden, 2020-10 [cit. 2022-03-15]. Dostupné z: https://bitwarden.com/help/bitwarden-security-white-paper/.

27. *Vaultwarden* [online]. Github [cit. 2022-04-08]. Dostupné z: https://github.com/dani-garcia/vaultwarden.
28. STEVENS, Marc; BURSZTEIN, Elie; KARPMAN, Pierre; ALBERTINI, Ange; MARKOV, Yarik. The First Collision for Full SHA-1. In: KATZ, Jonathan; SHACHAM, Hovav (ed.). *Advances in Cryptology – CRYPTO 2017*. Cham: Springer International Publishing, 2017, s. 570–596. ISBN 978-3-319-63688-7.
29. BROWN, Daniel RL. What hashes make RSA-OAEP secure? *Cryptology ePrint Archive*. 2006. Dostupné také z: https://ia.cr/2006/223.
30. 8BIT SOLUTIONS LLC AND CURE53. *Bitwarden Security Assessment Report* [online]. 2018-11-08 [cit. 2022-04-11]. Dostupné z: https://cdn.bitwarden.net/misc/Bitwarden%20Security%20Assessment%20Report.pdf.
31. *Zero-Knowledge Encryption. Why it Matters* [online]. Keeper [cit. 2022-04-08]. Dostupné z: https://www.keepersecurity.com/resources/zero-knowledge-for-ultimate-password-security.html.
32. *Stay safe with zero-knowledge architecture* [online]. NordPass [cit. 2022-04-08]. Dostupné z: https://nordpass.com/features/zero-knowledge-architecture/.
33. ORENSTEIN, Gary. How End-to-End Encryption Paves the Way for Zero Knowledge. In: *Bitwarden Blog* [online]. Bitwarden, 2021 [cit. 2022-04-08]. Dostupné z: https://bitwarden.com/blog/end-to-end-encryption-and-zero-knowledge/.
34. PARECKI, Aaron. *Password Grant* [online]. Okta [cit. 2022-04-24]. Dostupné z: https://www.oauth.com/oauth2-servers/access-tokens/password-grant/.
35. Use 2 iterations for local password hashing. In: *GitHub* [online]. 2021 [cit. 2022-03-15]. Dostupné z: https://github.com/bitwarden/jslib/pull/404.
36. ALTUWAIJRI, Haya; GHOUZALI, Sanaa. Android data storage security: A review. *Journal of King Saud University - Computer and Information Sciences*. 2020, roč. 32, č. 5, s. 543–552. ISSN 1319-1578. Dostupné z DOI: https://doi.org/10.1016/j.jksuci.2018.07.004.
37. NEIMAN, Caitlin. *Mozilla Add-ons Community Blog*. New backend for storage.local API [online]. Mozilla, 2018-08-03 [cit. 2022-04-23]. Dostupné z: https://blog.mozilla.org/addons/2018/08/03/new-backend-for-storage-local-api/.
38. *Chromium Session Storage and Local Storage* [online]. 2021-06-23 [cit. 2022-04-23]. Dostupné z: https://www.cclsolutionsgroup.com/post/chromium-session-storage-and-local-storage.
39. Hashcat plugin development guide. In: *GitHub* [online]. GitHub, Inc., 2021 [cit. 2022-03-15]. Dostupné z: https://github.com/hashcat/hashcat/blob/master/docs/hashcat-plugin-development-guide.md.
40. Added parameter for second PBKDF2 iteration count for -M 23400. In: *GitHub* [online]. GitHub, Inc., 2022 [cit. 2022-03-15]. Dostupné z: https://github.com/hashcat/hashcat/pull/3202.
41. *How to create more work for full speed?* [Online]. Hashcat [cit. 2022-05-05]. Dostupné z: https://hashcat.net/faq/morework.
42. CARR, Michael; SHAHANDASHTI, Siamak F. Revisiting Security Vulnerabilities in Commercial Password Managers. In: HÖLBL, Marko; RANNENBERG, Kai; WELZER, Tatjana (ed.). *ICT Systems Security and Privacy Protection*. Cham: Springer International Publishing, 2020, s. 265–279. ISBN 978-3-030-58201-2.
43. RICHER, Justin; SANSO, Antonio; GLAZER, Ian. *OAUTH 2 in action*. Manning Publications, 2017. ISBN 978-1617293276.

44. Why bitwarden use password flow? In: *GitHub* [online]. GitHub, Inc., 2019 [cit. 2022-04-09]. Dostupné z: https://github.com/bitwarden/server/issues/518.

45. MUELLER, Bernhard; SCHLEIER, Sven; WILLEMSEN, Jeroen; HOLGUERA, Carlos. *OWASP Mobile Security Testing Guide* [online]. 2022-01-21 [cit. 2022-04-09]. Dostupné z: https://github.com/OWASP/owasp-mstg/releases/tag/v1.4.0.

46. Vault Encryption. In: *Bitwarden Forum* [online]. 2022 [cit. 2022-04-11]. Dostupné z: https://community.bitwarden.com/t/vault-encryption/39483.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `data` ..... adresář se soubory analýzy
- `exe` ..... adresář se spustitelnými soubory
- `src` ..... zdrojové kódy
  - `bitwarden` ..... zdrojové kódy Bitwarden aplikací
  - `malicious_server` ..... zdrojové kódy proof of concept serveru
  - `hashcat` ..... zdrojové kódy nástroje Hashcat
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF