Bezpečnost uživatelských hesel a jejich prolamování

David Gála

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav počítačových a komunikačních systémů

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David Gála**
Osobní číslo:	**A20030**
Studijní program:	**B0688A140008 Informační technologie v administrativě**
Forma studia:	**Prezenční**
Téma práce:	**Bezpečnost uživatelských hesel a jejich prolamování**
Téma práce anglicky:	**Security of User Passwords and Their Cracking**

Zásady pro vypracování

1. Popište základní techniky prolamování hesel.
2. Porovnejte možnosti lámání hesel na CPU a GPU.
3. Demonstrujte funkčnosti keyloggeru.
4. Otestujte možnosti prolamování hesel na běžně dostupných počítačích.
5. Na základě zjištěných informací definujte bezpečné heslo a jak s ním pracovat.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PICOLET, Joshua. Hash Crack: Password Cracking Manual. INDEPENDENTLY PUBLISHED, 2019. ISBN 1793458618.
2. MAD76E. Basic Hash Cracking. Lulu.com, 2016. ISBN 1365481891.
3. BURNETT, Mark. Perfect Password: Selection, Protection, Authentication. Elsevier, 2006. ISBN 9780080489513.
4. TIRADO, Emanuel, et al. A new distributed brute-force password cracking technique. In: International conference on future network systems and security. Springer, Cham, 2018. p. 117-127.
5. AGGARWAL, Sudhir; HOUSHMAND, Shiva; WEIR, Matt. New technologies in password cracking techniques. In: Cyber security: power and technology. Springer, Cham, 2018. p. 179-198.
6. MARCHETTI, Kaden; BODILY, Paul. John the Ripper: An Examination and Analysis of the Popular Hash Cracking Algorithm. In: 2022 Intermountain Engineering, Technology and Computing (IETC). IEEE, 2022. p. 1-6.

Vedoucí bakalářské práce: **Ing. Lukáš Králík, Ph.D.**
Ústav bezpečnostního inženýrství

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **24. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Ing. Petr Šilhavý, Ph.D. v.r.
garant oboru

Ve Zlíně dne 8. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis studenta

ABSTRAKT

Bakalářská práce se zabývá problematikou zabezpečení uživatelských hesel a jejich možného prolamování. V teoretické části práce jsou probírány možnosti zabezpečení hesel a způsoby, kterými lze hesla prolamovat. V praktické části práce jsou demonstrovány pokusy o prolomení jednotlivých zabezpečovacích funkcí a popsána tvorba bezpečného hesla, včetně jeho zacházení. Výstupem práce jsou benchmarkové testy a testy na jednotlivých heslech.

Klíčová slova: heslo, hash, Hashcat, GPU

ABSTRACT

The bachelor thesis deals with the issue of user password security and their possible cracking. The theoretical part of the thesis discusses the possibilities of password security and the ways in which passwords can be cracked. The practical part of the thesis demonstrates attempts to crack the various security functions and describes the creation of a secure password, including its handling. The output of the work are benchmark tests and tests on individual passwords.

Keywords: password, hash, Hashcat, GPU

Tímto bych rád poděkoval svému vedoucímu bakalářské práce Ing. Lukáši Králíkovi Ph.D. za konzultace a vedení práce a cenné rady, které mi věnoval při zpracování této práce. Dále bych chtěl poděkovat všem učitelům na FAI UTB, díky kterým jsem nabyl znalostem, které mě dovedly až sem. Poděkování taktéž patří mé rodině a kamarádům, kteří mě podporovali po celou dobu studia a při tvorbě této práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V současném světě, kde je téměř vše vyřizováno a evidováno elektronickou formou, jsou hesla nedílnou součástí života téměř každého člověka. Bez hesel bychom se v dnešní době neobešli, jelikož jsou potřeba pro zabezpečování celé řady nejen počítačových aplikací ale i různých účtů a zařízení. Můžeme hovořit například o heslech k uživatelským účtům na sociálních sítích, heslech k bankovním portálům, nebo třeba PIN kódech k platebním kartám. Bohužel většina uživatelů však své účty nezabezpečuje dostatečně a používá velmi jednoduchá hesla, které není tak složité prolomit. Správné zabezpečení účtu by mělo být v zájmu každého uživatele. V dnešní době stále existuje předpoklad, že pro tvorbu bezpečného hesla je dostačujících osm znaků, alespoň jedno velké písmeno, alespoň jedna číslice a v ojedinělých případech jeden speciální znak. Problematika zabezpečení hesel, možnosti jejich prolamování a ověření tohoto předpokladu jsou faktory, které mě vedly ke zvolení tohoto tématu bakalářské práce.

V teoretické části této práce bude řešena problematika a způsoby zabezpečení uživatelských hesel, jelikož je nutno nejdříve pochopit, jakým způsobem bývají uživatelská hesla zabezpečena. V této části budou popsány dvě skupiny funkcí, které se pro zabezpečení uživatelských hesel používají. Dále budou popsány dva typy nástrojů, které jsou využívány k prolamování hesel, s uvedením jejich příkladů. Budou popsány i jednotlivé typy útoků, které je možno na hesla aplikovat. Zde budou popsány útoky prováděné v již zmíněných nástrojích i s nástrojem, jehož úkolem je zachycovat znaky z klávesnice. Následně bude popsán rozdíl útoků provedených pomocí centrální procesorové jednotky a grafického procesoru a útok pomocí hradlových polí.

V praktické části se nejprve budu věnovat porovnání výkonnosti prolamování centrální procesorové jednotky a grafického čipu. Poté bude demonstrována funkčnost obou forem zařízení zachycující znaky. V následující části bude demonstrována výkonnost na vybraných zařízeních, kde budou jednotlivá zařízení popsána. Během demonstrování budou nejprve provedeny benchmarkové testy a následně testy útoků na vybraných funkcích. Funkcí používaných k zabezpečování hesel je velké množství, takže byly vybrány pouze nejpoužívanější z nich. V závěru bude popsáno, jakým způsobem by měla být tvořena bezpečná hesla a jak taková hesla správně zabezpečit.

I. TEORETICKÁ ČÁST

1 ZPŮSOBY ZABEZPEČENÍ UŽIVATELSKÝCH HESEL

Moderní způsob zabezpečování uživatelských hesel se výrazně liší od způsobů využívaných jen před několika lety. To především v jejich komplexnosti a možné modifikovatelnosti.

Nejprve se hesla ukládala jako tzv. plaintext, tudíž bez jakékoliv úpravy a ochrany. Takové ukládání nevydrželo dlouho, jelikož jakmile byl prolomen přístup k databázím, tak byla veškerá hesla odhalena. Přešlo se tedy na metodu hashování hesel. Předchůdci moderních hashů byly hashe MD5 a SHA1. Při rostoucím výpočetním výkonu jsou ale tyto funkce již nedostačující.

Nyní se využívají funkce, které jsou pro tyto účely mnohem spolehlivější. Těmito funkcemi jsou jmenovitě Bcrypt, argon2 a další key derivation funkce. To hlavně proto, že při použití předešlých funkcí je získán vždy stejný hash. U moderních funkcí tomu tak není. Moderní funkce mohou být modifikovány na mnoha parametrech. Tím se zajistí mnohem vyšší ochrana hesla.

Pro ještě větší ochranu hesla je možno na hash použít sůl a pepř. Některé funkce mají ve svém algoritmu již zabudované použití soli.

1.1 Hashovací funkce

Jedná se o funkce, jejichž úkolem je ze vstupu o libovolné délce vytvořit výstup(hash) o pevné délce. V průběhu algoritmů takových funkcí je vstup nejdříve rozdělen do bloků o určených bitových délkách. Každý algoritmus používá jinou bitovou délku těchto bloků. Na blocích jsou následně prováděny logické a bitové operace. Mezi takové operace patří například logický součin, logický součet a bitové rotace, posuny a masky. Algoritmy těchto funkcí se ale nemění a nemohou být modifikovány, tudíž ze stejného vstupu je vždy získán stejný výstup. Nejznámějšími zástupci těchto funkcí, kteří byli dříve používány pro zabezpečování hesel, jsou MD5, SHA1, SHA-256 a SHA-512. Do jisté míry jsou k tomuto účelu používány i v dnešní době, i když nejsou tolik bezpečné a existují lepší alternativy. Algoritmy těchto funkcí k tomuto účelu již nedostačují z důvodu, že jsou navrženy tak, aby byl jejich průběh rychlý, což je se vzrůstajícím výpočetním výkonem problém.[1]

Každá z funkcí využívá jiný algoritmus, ale všechny mají stejný nedostatek pro zabezpečování hesel. Tímto nedostatkem je právě vždy stejný výstup ze jednoho vstupu. To je zapříčiněno tím, že ani jedena z funkcí nedisponuje možností modifikování hashovací funkce. Jinými slovy heslo projde jednou iterací a je ihned vytvořen hash.

Tento nedostatek byl poměrně rychle zneužit. Ke zneužití došlo vytvořením tzv. „Rainbow tables“. To jsou tabulky, které obsahují již vygenerované hashe pro určitá slova nebo již prolomená hesla. Využívání těchto tabulek se výrazně ulehčuje útok hrubou silou na tyto hashe.[2]

Dalším nedostatkem je nedostatečná délka výstupního hashe, což může zapříčinit kolize. Vznik kolize znamená, že pro dva nebo více různých vstupů jsou vytvořeny totožné výstupní hashe. S touto komplikací se setkala funkce SHA1. Kvůli vzniku kolize byla tato funkce označena za nedostatečně bezpečnou a vznikla rodina funkcí SHA2, kam patří například SHA-256 a SHA-512. Hlavním rozdílem je navýšení délky výstupního hashe, která v bitech odpovídá číselným označením daných funkcí. Dalším rozdílem je odlišný počet kroků v průběhu algoritmu a také jejich komplexnost.[3]

Přes své slabiny jsou tyto funkce však stále využitelné, a to pro ověření integrity dat.

1.2 Key derivation funkce

Tyto funkce mají za úkol, stejně jako funkce hashovací, zabezpečovat hesla. Oproti funkcím hashovacím jsou však trochu komplexnější. Funkce nevyužívají k získání výstupu pouze čistý algoritmus s jedním průchodem, ale algoritmus je na heslo proveden vícekrát a jeho vstupní proměnné lze modifikovat. Obecný průběh většiny těch funkcí se skládá z několika částí: počet iterací, použití soli, hashovací algoritmus, tvorba výstupu. Většina těchto funkcí je modifikovatelná při jejich implementaci s tím, že každá nabízí jinou úroveň modifikovatelnosti. Některé umožňují modifikovat pouze počet iterací, jiné umožňují modifikovat i délku použité soli, délku výstupu nebo počet použitých výpočetních jednotek a vnitřní hashovací algoritmus. Díky těmto možnostem jsou tyto funkce mnohem bezpečnější než funkce hashovací. Mezi nejpoužívanější z těchto funkcí patří Bcrypt, Scrypt, Argon2 a PBKDF2.[4]

1.2.1 Bcrypt

Bcrypt je key derivation funkce, která využívá šifry blowfish. Jedná se o adaptivní funkci. Adaptivní v tomto případě znamená, že lze libovolně nastavovat počet iterací, kterými heslo projde. Tato funkce má 3 vstupní parametry. Těmi jsou: cost, salt, key. Cost neboli cena zastupuje čas potřebný k výpočtu všech iterací. Salt neboli sůl je řetězec náhodných znaků, který je přidán k heslu před iteracemi. Tato operace chrání před útoky pomocí Rainbow tables. V poslední řadě key je heslo zadané uživatelem. [5]

Heslo zabezpečené touto funkcí je tedy chráněno před útoky pomocí Rainbow tables díky použití soli a proti útokům hrubou silou za použití vícera iterací, při kterých je použito vícero solí.

1.2.2 Scrypt

Scrypt je adaptivní funkcí, která umožňuje modifikaci nejen počtu iterací neboli hodnoty cost, ale také výpočetní složitost a schopnost paralelizace. Parametr výpočetní složitosti určuje počet kroků během výpočtu jednoho bloku, což má dopad na paměťovou náročnost celkového výpočtu. Parametr schopnosti paralelizace určuje, kolik výpočetních jednotek bude použito pro celkový výpočet funkce. Nastavení těchto parametrů ovlivňuje náročnost celkového výpočtu na hardware a také určuje dobu potřebnou k výpočtu. Nastavením hodnoty pro výpočetní složitost na nízké číslo by znamenalo nižší úroveň zabezpečení. Hodnota však nemůže být nastavena na příliš vysokou, neboť je nutno dbát na hardwarové prostředky a jejich výkon. Nastavení parametru schopnosti paralelizace na vysokou hodnotu umožňuje využití více výpočetních jednotek, a tudíž by výpočet urychlil, takže je výhodnější tuto hodnotu nastavovat na nižší.[5]

1.2.3 Argon2

Argon2 je adaptivní funkce, která umožňuje větší možnosti modifikování. Funkce dovoluje modifikování vícero prvků nezávisle na sobě. Na rozdíl od Bcrypt, kde se zadáním „ceny“ vypočítává počet iterací, Argon2 dovoluje určení počtu iterací a „ceny“ nezávisle na sobě. Je tedy možno nastavit „cenu“, která určuje, jak dlouho se bude finální hash zpracovávat, a také počet iterací. Dále dovoluje nastavit, kolik výpočetních jednotek bude pro vytvoření finálního hashe využito. Tím zamezuje efektivnímu paralelnímu prolamování hashů. Paralelní prolamování hashů znamená, že útočník využívá k prolomení vícero výpočetních jednotek, aby mohl provádět vícero souběžných útoků. V poslední řadě dovoluje nastavit délku náhodného textu přidaného před hashováním neboli soli a délku výsledného hashe. Těmito možnostmi modifikace zaručuje unikátnost hashů a nemožnost je prolomit metodami jako je útok hrubou silou nebo slovníkový útok.[6]

1.2.4 Password-Based Key Derivation Function 2 (PBKDF2)

PBKDF2 je adaptivní funkce umožňující uživateli modifikování vnitřního algoritmu funkce. Kromě klasických modifikací, jako je počet iterací nebo délka výstupního hashe, umožňuje

tato funkce zvolit jaká hashovací funkce bude použita ve vnitřním algoritmu. Při výběru je možno vybrat z funkcí, které jsou založeny na algoritmech rodiny SHA2 a MD5.[5]

1.3 Salt and Pepper

Salting a Peppering neboli používání soli a pepře dodává heslům další vrstvu ochrany. Ve zjednodušené formě se jedná o přidávání řetězců znaků k heslu, aby se zamezilo nebo byly zpomaleny určité typy útoků.

1.3.1 Salting

Salting neboli solení je akce, při které je k heslu, které zadal uživatel, přidán řetězec náhodných znaků ještě před hashováním. To dodává ochranu proti útokům pomocí Rainbow tables a to tím, že hash nereprezentuje žádný již prolomený nebo předdefinovaný hash v tabulkách. Do databáze se následně ukládá výsledný hash a odděleně jeho sůl, aby bylo možno provést ověření jednodušeji.[7]

1.3.2 Peppering

Peppering neboli pepření je velmi podobné saltingu tím, že touto akcí taktéž přidáme náhodnou hodnotu k heslu, čímž kompletně změníme jeho hash. Na rozdíl od soli však pepř nemůžeme uchovávat v databázi společně s heslem. Pepř má stejný úkol jako sůl, a to je zpomalení útoku hrubou silou. Na rozdíl od soli však není pro každé heslo unikátní a většinou je to stejný řetězec, pro všechna hesla na webové stránce nebo portálu.[7]

2 NÁSTROJE NA PROLAMOVÁNÍ HESEL

K prolamování hesel existuje více nástrojů, ale většina z nich funguje na stejném systému. Tyto nástroje se dají dělit na dva typy. Prvním typem jsou nástroje, se kterými je pracováno v příkazové řádce. Druhý typ jsou nástroje, které mají grafické rozhraní a jsou spustitelné jako okenní aplikace. Hlavním rozdílem mezi těmito typy je možnost nastavitelnosti vstupních parametrů. Nástroje pracující v příkazové řádce dovolují uživatelům nastavení mnohem více parametrů, a tudíž mohou přizpůsobit útok přesně ke svým požadavkům. Mezi nejznámější z těchto nástrojů patří například Hashcat [8], který patří k nejrychlejším a nejrozvinutějším z nástrojů. Aplikační nástroje sice nedovolují uživateli natolik přizpůsobit prováděný útok, ale pro méně zkušené uživatele mohou být přívětivější právě proto, že poskytují grafické rozhraní, ve kterém lze pracovat i bez znalosti příkazové řádky. Obstojným zástupcem těchto nástrojů je nástroj HashSuite [9].

2.1 Hashcat

Hashcat je nejrozšířenějším nástrojem pro obnovu a prolamování hesel, který patří k nejrozvinutějším a nejrychlejším v oblasti. Podporuje velký počet hashovacích a key derivation funkcí, ale i funkce používané u kryptoměn a peněženek. Jedinou podmínkou pro spuštění programu je podpora OpenCL neboli Open Computing Language, což je standard pro multiplatformové paralelní programování. Program lze tedy spustit, jak na CPU a GPU, tak například i na programovatelných hradlových polích. Program je podporován běžnými operačními systémy jako je Windows, Linux nebo OSX. Hashcat není vyvíjen a nespadá pod žádnou vývojářskou firmu nebo organizaci, ale je vyvíjen a spravován komunitou odborníků z oboru.

Hashcat nabízí uživatelům mnoho funkcí, se kterými mohou pracovat. První funkcí je již vestavěný benchmark, který uživateli zobrazí přibližné rychlosti obnovy hesel pro nejběžněji používané funkce. Tyto rychlosti však nemusí být vždy přesné, jelikož záleží na mnoha faktorech, jako nastavení vstupních parametrů, využití paměti zařízení a momentální teplotě. Dále program kontroluje, jakou má teplotu zařízení, na kterém je spuštěn, a pokud by přesáhla určitou hodnotu, tak by byla obnova nebo útok zastaven. Probíhající obnovu nebo útok lze taktéž zastavit manuálně, a to kompletním ukončením nebo pozastavením na bodu obnovy, ze kterého je možno opět navázat.

Nástroj však není dokonalý, jelikož se stále nachází ve vývoji. Tento nedostatek lze poznat na sadách znaků, které využívá, protože nepodporuje znaky, které se nenacházejí v tabulce ASCII, což je velká nevýhoda u speciálních znaků. Další nevýhodou je možná délka prolamovaných hesel. Zde se nevýhoda vyskytuje v okamžik, kdy je nastavena maska pro útok hrubou silou na velký počet znaků za použití sady obsahující všechny znaky, což zahltí zásobník a program nemůže pracovat. V takovém případě je tedy nutno použití hybridních útoků. Hashcat má taktéž zabudované opatření, které mu nedovolí využít plnou kapacitu paměti zařízení. Toto opatření je zabudováno jak z důvodu nezneužití programu, tak proto aby nebyla vyčerpána paměť a systém se nezhroutil.[8]

2.2 HashSuite

HashSuite je okenní aplikace pro operační systém Windows, která má za úkol obnovu hesel a testování jejich bezpečnosti. Aplikace nabízí velké množství slovníků, které lze stáhnout přímo přes aplikaci. Před započetím útoku je možno nastavení pravidel, která budou na útok aplikována. Tato pravidla definují, jakým způsobem bude proveden pokus o prolomení hesla. Lze nastavit jaké sady znaků budou použity, jaký bude použit slovník a jaké budou kombinace složení hybridního útoku. Další funkcí je vestavěný benchmark, který stejně jako proveditelné útoky podporuje pouze 14 funkcí tvořících hash. Dále aplikace umožňuje import hashů z účtů, registrů a souborů a následně export prolomených hesel. Je možno exportovat pouze prolomená hesla, prolomená hesla v podobě slovníku nebo i hashe, které byly prolamovány. Aplikace také obsahuje hashovací kalkulačku, která převede text na hash, ta ale podporuje pouze formáty NTLM a LM.

Nevýhodou stejně jako u Hashcatu je omezení sad znaků pouze na tabulku ASCII. Další velkou nevýhodou je počet podporovaných funkcí vytvářejících hashe. Nastavení pravidel taktéž není nejlépe optimalizované, neboť dovoluje uživateli použít pouze zabudovaná pravidla bez možnosti tvorby pravidel vlastních.[9]

3 TECHNIKY PROLAMOVÁNÍ UŽIVATELSKÝCH HESEL

Je mnoho technik, jak zjistit nebo prolomit uživatelské heslo, některé jsou jednodušší a některé složitější. V případě, že jsou o cíli některé jeho citlivé informace známé, tak mohou být pro tento účel využity. Jiné techniky dále využívají například důvěřivosti člověka nebo jeho neopatrnosti. Mezi takové techniky patří například phishing nebo social engineering. Při použití takových technik je hlavním úkolem získat důvěru cíle a jejího využití. Nejsou to však techniky používané pro prolomení jako spíš pro získání hesla. Techniky, které hesla prolamují, pracují většinou se získaným hashem hesla.

3.1 Slovníkový útok

Tato technika je založena na prolomení hesla za použití textových dokumentů. Tyto textové dokumenty obsahují nezměrný počet slov, čísel a jejich kombinací. Většinově se jedná o slova a kombinace, které již byla prolomena jako hesla nebo by se dala považovat za často používaná. Útok se uskutečňuje za použití dalšího programu, který pracuje s těmito slovníky a s hashi. Do programu je vložen kód, kterým je programu řečeno, jaký hash se má pokusit prolomit a jaký slovník na to použít. Program pak následně projde každý řádek ve slovníku a hodnotu, která je na řádku zastoupena zkusí jako kandidáta pro prolamované heslo. Pokud se hash některého z kandidátů shoduje s hashem originálním, znamená to, že heslo bylo prolomeno. Slovníky mohou být vytvořeny útočníkem na základě dat o cíli nebo získány univerzální slovníky z internetu. Jeden z nejznámějších univerzálních slovníků je rockyou [10]. Další slovníky se mohou specializovat například na výrazy z filmů, seriálů, knih nebo na cizojazyčné slovníky.[11]

3.2 Útok hrubou silou

Tento typ útoku funguje na jednoduchém principu. Během off-line verze tohoto útoku je potřeba mít hash, z něhož chceme bude zjišťována jeho původní hodnota nebo podoba. Dále je potřeba určit, jaké znaky se v hesle mohou vyskytovat. To je potřeba pro zadání setu znaků, které bude program využívat k prolomení hesla. Sety znaků mohou být různě kombinovány a také mohou být vytvářeny sety vlastní. Následně je hash, set a předpokládaná délka hesla zadána do programu, který bude útok provádět. Program pak postupně vytváří kombinace všech znaků setu podle zadané délky, až do prolomení hesla nebo vyčerpání možností. V takovém případě je potřeba změna předpokládané délky hesla. Tento útok je

lepší provádět off-line na známém hashi z důvodu zablokování možnosti pokusů nebo nutné změně hesla při vysokém množství pokusů.[11]

3.3 Útok s maskou

Útok s maskou je specifičtější verzí útoku hrubou silou. To hlavně proto, že hesla prolamuje stejným způsobem, ale dovoluje nastavit parametry, které výrazně zkracují dobu potřebnou k prolomení hesla. Tato schopnost je jeho největší výhodou nad útokem hrubou silou. Možnost nastavení masky je především účinná, pokud je prolamováno heslo, které se drží nějakého jednoduchého vzorce. Jeden z takových vzorců může být například heslo, které obsahuje jméno nebo slovo následované čísly. Internetové stránky většinou požadují, aby heslo pro zabezpečení účtu obsahovalo i velké písmeno. V tomto případě většina uživatelů použije velké písmeno na první pozici hesla. Díky tomuto vzorci je možno vytvořit masku takovou, aby vzorci odpovídala, a tudíž zkrátila dobu prolamování.

Příkladem tomuto útoku bude uživatel, jehož jméno je Alexandr s datem narození 1986. Předpokládané heslo je tudíž Alexandr1986. V jiném případě to může být například jméno partnera nebo domácího mazlíčka. Pokud by takové heslo bylo prolamováno klasickým útokem hrubou silou, tak by muselo projít zhruba 62^{12} pokusy. To z důvodu, že heslo je 12 znaků dlouhé a na každou pozici je nutno využít set znaků obsahující malá i velká písmena a speciální znaky. Vyzkoušet tolik pokusů by při rychlosti například 500M/s (kandidátů za sekundu) trvalo asi 204 608 let. Pokud je předpokládáno využití vzorce, podle kterého lidé vytváří hesla, tudíž jméno a rok, je možno sestrojit masku. Maska bude vytvořena jako velká nebo malá písmena na první pozici, malá písmena na dalších sedmi pozicích a poslední čtyři pozice budou čísla. Při tomto předpokladu se výpočet pokusů redukuje na 52*26*26*26*26*26*26*26*10*10*10*10. To by při stejné rychlosti trvalo asi 96 dnů.[11]

3.4 Hybridní útok

Hybridní útok je ve své podstatě spojením útoku hrubou silou a slovníkového útoku. Během tohoto útoku je použit vybraný slovník a maska použitá pro útok hrubou silou. Je nutno předem určit, zdali bude připojen útok hrubou silou před nebo po slovu, které se nachází ve vybraném slovníku. To je určeno pozicí parametrů v příkazu tohoto útoku. Pokud je v příkazu nejdříve zadán slovník a následně maska, tak je útok hrubou silou použit na konci slova podle pravidel masky. V opačném případě, pokud je nejdřív zadána maska a následně

slovník, tak je nejdřív použit útok hrubou silou podle pravidel masky a za něj jsou vkládána slova z použitého slovníku.[11]

3.5 Útok za použití Rainbow tables

Aby bylo možno tento útok použít, tak je nutné mít k dispozici hash hesla, které je prolamováno. Dále je zapotřebí vlastnit Rainbow table pro odpovídající hashovací funkci a sadu znaků. Rainbow table je tabulka, která obsahuje již prolomená hesla a jejich hashe. Útok pomocí této tabulky využívá kolizí u hesel, která jsou jednoduše zahashována. To znamená, že útočník nepotřebuje znát přímo originální heslo, ale pouze řetězec, který vytvoří stejný hash, jako je hash prolamovaný.

Rainbow table se skládá z řádků, takzvaných chainů, kde jeden chain obsahuje velké množství sloupců. Tyto sloupce reprezentují nové řetězce a hashe vzniklé použitím redukční funkce na prvotní řetězce. Hashe v tabulce jsou porovnávány s prolamovaným hashem a je hledána shoda. Pokud shoda není nalezena, tak je opět použita redukční funkce a je proveden nový pokus o vyhledání shody.[12]

3.6 Keylogger

Jedná se o hardwarové zařízení nebo softwarový podprogram, jehož hlavním účelem je zaznamenávání kláves stisknutých uživatelem. Na neškodné užití bývá propagován a používán například k monitorování aktivit dětí na počítači nebo k monitorování zaměstnanců na firemních počítačích. Pro jiné účely bývá ale používán jako monitorovací nástroj ke sledování aktivit uživatelů a následného odcizení jejich osobních a přihlašovacích údajů. Každý keylogger zaznamenává každou na počítači zmáčknutou klávesu. Mohou se lišit ve svém nastavení záznamu nebo bonusových funkcích. Některé keylogerry odesílají surový výstup v podobě nestrukturovaného textu a jiné mohou znaky od sebe oddělovat například po stihnutí určitých kláves jako je Enter nebo Tab. Dále mohou po stisku takové klávesy započnout nový řádek v záznamu. [13,14]

3.6.1 Softwarový

Softwarové keyloggery se dělí na několik typů v závislosti na jejich funkčnosti. Mezi tyto typy patří aplikační keyloggery, kernel keyloggery a formulářové keyloggery. Každý z těchto keyloggerů plní svou funkci jiným způsobem, ale důvod jejich použití zůstává stejný, a to získání dat o uživateli.

Aplikační keyloggery sbírají o uživateli údaje prostřednictvím dat, které získají ze spuštěných aplikací. Přesněji řečeno odposlouchávají data, které jsou součástí komunikace mezi aplikacemi a operačním systémem. Tyto keyloggery mohou být nainstalovány jako klasické aplikace nebo aplikace běžící pouze na pozadí.

Kernel keyloggery mají k datům uživatele, jako je například stisknutá klávesa, přímý přístup. Tento přístup mají proto, že jsou do systému instalovány tak, že se zdají jako driver k nějakému zařízení. To znamená, že fungují na úrovni jádra operačního systému spolu s dalšími drivery. Takový typ keyloggeru je v systému těžce detekovatelný, jelikož neběží jako aplikace, a tudíž nelze dohledat například přes správce spuštěných úloh.

Formulářové keyloggery fungují pouze na webových stránkách. Takové typy keyloggeru neodposlouchávají, která klávesa byla stlačena, ale extrahují data z formulářů. Tento krok je proveden v moment, kdy je formulář odeslán a data, která se nacházejí v jeho polích, což mohou být například přihlašovací informace, jsou zachycena a uložena do výstupu. Tyto keyloggery se mohou nacházet přímo v počítači uživatele nebo na webové stránce.

Bonusové funkce, kterými mohou softwarové keyloggery disponovat jsou například záznam kopírovaného textu, pořízení a záznam náhodně časovaných snímků obrazovky nebo záznam otevíraných složek, souborů a aplikací.[13,14]

3.6.2 Hardwarový

Hardwarovou formu keyloggeru představuje USB zařízení, které se zapojuje mezi klávesnici a počítač. Jedná se o tu nejjednodušší formu keyloggeru. Hardwarový keylogger se nedá odhalit na počítači přes různé diagnostiky. Ovšem pokud se uživatel počítače nebo správce počítačů v nějaké organizaci podívá na připojené periferie, tak je odhalen prakticky instantně. S tím souvisí i problematika jeho implementace pro záškodné účely. To zejména z toho důvodu, že osoba s takovými úmysly musí mít fyzický přístup k počítači, kam má být keylogger připojen. Připojení zařízení k počítači však není finálním krokem. Hardwarový keylogger je ve většině případů nutno nastavit tak, aby nebyla zjistitelná síť, kterou vytváří pro komunikaci nebo nebyl zjistitelný v rámci sítě, na kterou je nutno jej připojit, aby mohl odesílat záznamy. Tento krok bývá řešen použitím externího vysílače wifi s datovým připojením a skrytou SSID. Hardwarové keyloggery se vyrábí v různých velikostech, které však nemají žádný dopad na funkčnost keyloggeru, ta je stejná, ale s velikostí se mění pořizovací cena. Tato zařízení mohou mít velikost malého USB flash disku nebo být pouze miniaturním prostředníkem mezi klávesnicí a počítačem (Obrázky 1 a 2).[14]

Obrázek 1. Hardwarový keylogger[15]

Obrázek 2. Hardwarový keylogger[16]

4 ZPŮSOBY PROLAMOVÁNÍ HASHŮ

Hashe se dají prolamovat na celé řadě zařízeních. Ať už se jedná o velmi výkonné stolní počítače, výkonné notebooky, kancelářské počítače, minipočítače typu Raspberry Pi nebo programovatelná hradlová pole. Nejběžnějšími a nejlépe dostupnými zařízeními k prolamování hesel by byly nejspíše stolní počítače nebo notebooky, a to jak kancelářské, tak výkonné nebo herní. Tato zařízení využívají ke zpracování dat a výpočtům v zásadě dva komponenty. Těmi jsou procesor (CPU) a grafická karta (GPU). Každý z těchto komponenty slouží k jinému účelu a hodí se na jiné operace. Oba jsou ale v případě prolamování hesel použitelné. V případě užití mikropočítače jako Raspberry Pi je ovšem nutno pracovat pouze s integrovanou grafickou kartou. To znamená, že karta je vestavěná do čipu procesoru a nemá vlastní dedikovanou paměť, protože má s procesorem sdílenou paměť. U programovatelných hradlových polí není žádná grafická karta ani procesor, protože tato hradlová pole jsou pouze výpočetními jednotkami.

4.1 CPU vs GPU

Vzhledem k faktu, že počítače obsahují oba tyto komponenty, tak je jen otázkou, který se pro tento úkol hodí více. Aby bylo možno dojít k závěru, tak je nejdříve potřeba pochopit, jakou funkci každý komponent v počítači plní.

CPU – je hlavní řídící jednotkou v počítači a jak název vypovídá, tak má za úkol řídit ostatní komponenty a řadiče. Procesor je komponent, který není navrhován na paralelismus a to proto, že potřebuje zpracovávat jednotlivé úkony a operace jednu po druhé a v co nejrychlejším čase. To potřebuje kvůli tomu, aby mohl práci, kterou zvládne jiný komponent lépe přidělit jemu. Procesor vykonává základní operace a propočty jako je přidělení správného typu souboru. Procesory v současnosti většinou disponují přibližně šestnácti fyzickými jádry. Tato informace určuje, kolik má daný procesor aritmeticko-logických jednotek (ALU). U každého výrobce to ale stanovuje jiný počet. Příkladem jsou procesory Intel Core i9-12900K[17] a AMD Ryzen 9 5950X[18]. Oba procesory disponují stejným počtem jader, a to je šestnáct. Avšak společnost Intel v této řadě procesoru používá odlišný výrobní proces než společnost AMD. Intel zde používá 10 nanometrový výrobní proces a AMD 7 nanometrový. Tento rozdíl dovoluje AMD vložit více aritmeticko-logických jednotek na stejnou plochu jako Intel. Ve výsledku má AMD procesor 64 ALU jednotek a Intel pouze 48.[19]

GPU – jedná se o speciální typ procesoru, který je určený zejména pro zpracovávání grafických operací. Jeho hlavní funkce je co nejrychlejší zobrazení grafiky a videa na obrazovce. To znamená, že musí vypočítávat barvy a odstíny pro každý pixel. To je velmi repetitivní činnost složená převážně z výpočtů a ukládání výsledků. Právě proto CPU určí, aby takovou činnost vykonávala grafická karta, která se na to perfektně hodí. A to zejména protože grafická karta je navrhována na paralelismus, čehož dosahuje vysokým počtem ALU jednotek v jádrech s nižší frekvencí, než u CPU. Jelikož grafické karty nemusí být tolik versatilní jako CPU, které musí vykonávat řadu operací a rychle mezi nimi přepínat, tak jsou plně uzpůsobeny pro operace, pro které jsou určeny, což nachází využití i v tomto poli působnosti.[20]

Ve výsledku lze říci, že k prolamování hesel, kde mluvíme o útoku hrubou silou, je výhodnější grafická karta. Grafická karta je pro tento účel mnohem vhodnější z jednoho důvodu a tím je mnohem vyšší počet ALU jednotek. Průměrný počet ALU jednotek u grafických karet dnešní doby se pohybuje v řádech tisíců. Například grafická karta společnosti Nvidia s označením RTX 4090 disponuje 16 384 ALU jednotkami [21]. CPU mají mnohem vyšší frekvence, tudíž jsou schopny pracovat rychleji, ale nedokáží zpracovávat tak vysoký počet operací souběžně jako GPU.

4.2 Programovatelná hradlová pole

FPGA neboli programovatelná hradlová pole jsou k prolamování hesel velmi vhodná, protože se jedná o zařízení, které plní přesně takovou úlohu, na jakou jsou navržena a naprogramována. Jedná se o složení programovatelných logických bloků, propojovacích prvků a paměťových prvků. Díky programovatelným logickým blokům a možnosti jejich propojení je možné vytvořit takřka libovolné číslicové zařízení. FPGA se na prolamování hesel hodí hlavně kvůli tomu, že je na něm možno vytvořit přesně taková logická hradla, která jsou zapotřebí pro určité matematické a logické operace použité v hashovacích algoritmech. Samostatné obvody nemají vysokou pracovní frekvenci, tudíž nepracují tak rychle, jako jiné výpočetní jednotky, ale lze je velmi snadno paralelizovat. Pomocí paralelizování těchto obvodů lze výpočty rozdělit mezi velký počet výpočetních jednotek a urychlit výpočty. [22,23]

II. PRAKTICKÁ ČÁST

5 POROVNÁNÍ PROLAMOVÁNÍ NA CPU A GPU

Pokud by se vycházelo z teoretického předpokladu, že grafická karta by měla být lépe uzpůsobena pro výpočty hashovacích a key derivation funkcí kvůli mnohem vyššímu počtu výpočetních jednotek ALU, tak by to znamenalo, že zvládne vypočítat mnohem více hashových kandidátů než centrální procesorová jednotka. Tento předpoklad lze velmi snadno ověřit, a to buď jednotlivými útoky, nebo benchmarkem. Jelikož pro tyto účely není třeba provádět útoky s nastavitelnými parametry, tak stačí využít vestavěného benchmarku v programu HashSuite. Benchmark v tomto programu provede testy na jednotlivých funkcích, které podporuje, a to jak pro grafickou kartu, tak pro procesor. Testy jsou v programu prováděny pro jednotlivé počty hashů souběžně. Nejdříve je testováno, kolik kandidátu je komponent schopen vypočítat pro jeden hash a postupně se počty hashů na jeden útok zvyšují. Finálním počtem hashů pro funkce, které nepoužívají soli a jsou výpočetně jednodušší, je deset milionů. Pro funkce, které soli používají nebo jsou náročnější na výpočet, je finální hodnotou šedesát čtyři hashů najednou. Vyšší počet hashů pro funkce, které používají soli, není možné počítat, jelikož jsou náročnější na využití paměti.

Complete

Processor Name	Frequency	L1	L2	L3	Ram	Other Information
Intel Core i7-8700K	3.70GHz	384KB	1MB	12MB	16GB DDR4-2400	Windows 10 64-bit
NVIDIA GeForce GTX 1080	1.82GHz	960KB	2MB	0KB	8.0GB GDDR5-5005	Driver 528.49

Format	Threads	1	10	100	1000	10000	100000	1 million	10 millions
LM	12	971M	1.01G	1.06G	1.02G	1.08G	969M	621M	278M
	GPU	0	0	11.8G	11.3G	11.3G	4.27G	1.67G	1.11G
NTLM	12	979M	967M	980M	959M	967M	893M	768M	291M
	GPU	50.8G	26.7G	25.6G	24.1G	23.6G	23.3G	1.71G	1.33G
Raw-MD5	12	807M	801M	807M	793M	798M	726M	649M	296M
	GPU	25.0G	19.3G	19.2G	18.6G	17.8G	17.9G	1.71G	1.33G
Raw-SHA1	12	374M	373M	369M	370M	367M	340M	329M	226M
	GPU	9.09G	8.55G	8.53G	8.41G	8.17G	8.09G	1.73G	1.34G
Raw-SHA256	12	185M	192M	193M	192M	189M	188M	181M	150M
	GPU	3.33G	3.25G	3.24G	3.24G	3.20G	3.21G	1.73G	1.34G
Raw-SHA512	12	77.1M	77.1M	77.0M	75.6M	76.1M	76.4M	75.3M	70.9M
	GPU	1.18G	1.16G	1.15G	1.14G	1.14G	1.13G	1.00G	964M
		1	4	16	64				
DCC	12	656M	283M	83.6M	22.2M				
	GPU	12.9G	5.72G	1.40G	314M				
SSHA	12	317M	84.5M	21.9M	5.49M				
	GPU	8.37G	2.12G	527M	131M				
MD5CRYPT	12	928K	232K	58.1K	14.5K				
	GPU	13.6M	3.40M	850K	213K				
DCC2	12	19.2K	4.81K	1.16K	283				
	GPU	364K	91.1K	23.0K	5.68K				
WPA-PSK	12	24.1K	6.01K	1.48K	357				
	GPU	460K	114K	28.5K	7.00K				
BCRYPT	12	11.2K	2.79K	696	169				
	GPU	14.4K	3.55K	886	222				
SHA256CRYPT	12	21.0K	5.09K	1.30K	322				
	GPU	304K	76.4K	0	0				
SHA512CRYPT	12	13.9K	3.46K	867	215				
	GPU	153K	25.4K	0	0				

View saved benchmark OK

Obrázek 3. HashSuite benchmark

Benchmark byl proveden na procesoru Intel Core i7-8700K a grafické kartě Nvidia GeForce GTX 1080. Benchmark v tomto programu trvá přibližně 12 minut a výstupní data jsou seřazena do tabulky (Obrázek 3). Z těchto výstupních dat je možno vyčíst, že grafická karta opravdu zvládá vypočítat mnohem vyšší počet kandidátů pro určené počty hashů, než procesor. Výsledky z obou komponent lze následně porovnat a určit zhruba kolikrát je grafická karta výkonnější. Při výpočtech kandidátů pouze pro jeden hash je grafická karta mnohem výkonnější než procesor u funkcí, které jsou rychlejší. Například u funkce NTLM je grafická karta výkonnější až 50krát a u funkce MD5 30krát. Se zvyšujícím počtem souběžně počítaných hashů se násobek výkonnosti postupně snižuje. U rychlejších funkcí se násobek výkonnosti opět nepatrně navýší u výpočtu sta tisíc hashů a u funkcí náročnějších se násobek navyšuje u výpočtu šestnácti hashů. U funkcí pomalejších se násobky výkonosti od sebe neliší o tak velké hodnoty jako u funkcí rychlejších. To je způsobeno především tím, že pomalejší funkce jsou náročnější na využití paměti, a tudíž je nelze počítat paralelně, tak efektivně, jako funkce rychlejší.

6 DEMONSTRACE KEYLOGGERU

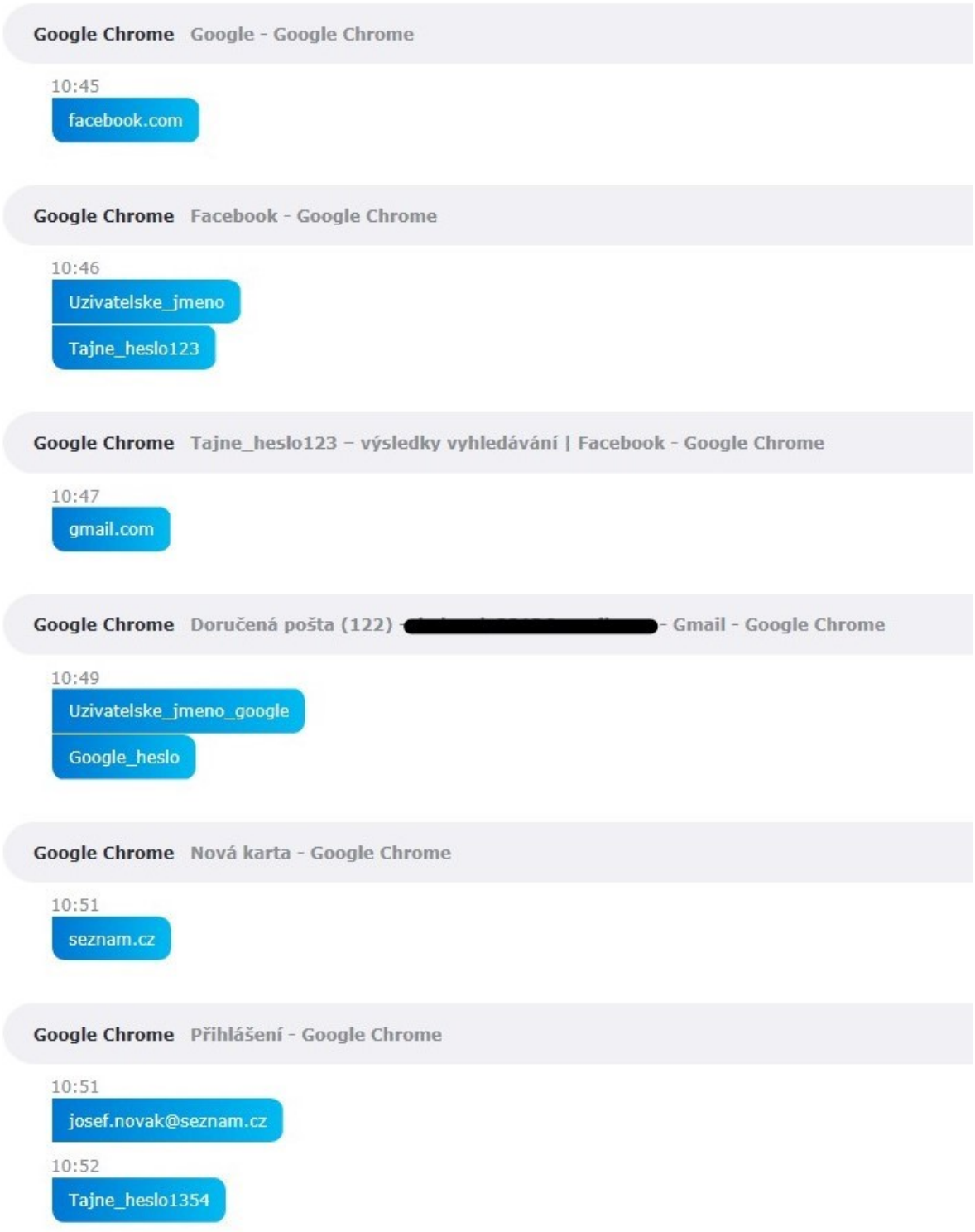
Pro oba keyloggery bylo provedeno testovací navštívení stránek, kde bylo následně psáno na klávesnici. Následně byl u obou keyloggerů kontrolován výstup, který ukazoval, co byly keyloggery schopny zachytit a jak tento výstup reprezentují.

6.1 Softwarový

Jako softwarový keylogger byla zvolena aplikační varianta. Tato varianta byla zvolena z důvodu jednoduché dostupnosti a implementace. Byl zvolen program Revealer Keylogger Free[24]. Byl zvolen pro své přívětivé uživatelské prostředí, možnost fungování bez uživatelského účtu a nabízených schopností. Program nabízí záznam stisknutých kláves a pořizování snímků obrazovky. Program uživateli umožňuje nastavit, zdali bude monitorování uskutečňováno na všech uživatelských účtech nebo jen u konkrétních účtů. Monitorování stisknutých kláves je prováděno za každé situace, kdežto pořizování snímků obrazovky je nastavitelné na spouštěcí akci. Dále je možno nastavit jakou kvalitu bude snímek mít a jaká oblast obrazovky bude zachycena. Dalším nastavením je možnost doručení výstupů z keyloggeru. Zde program nabízí doručení pomocí cloudových služeb, E-mailu, FTP (File Transfer Protocol) a přes lokální síť. Je také možno nastavit filtrování obsahu, kdy program například nezahrne do monitorování určené programy nebo webové stránky. Poslední možností konfigurace je nastavení skrytí programu před nezkušeným uživatelem pomocí skrytí ve vyhledávači souborů a správci úloh. Dalším bezpečnostním prvkem je možnost nastavení otevření programu při stisknutí kombinace specifických kláves nebo odinstalování programu ve stanovený čas.

Pro demonstrování byl tento keylogger testován na webových stránkách, jelikož zde ukazuje nejvíce údajů o aktivitě uživatele. Bylo testováno pouze monitorování stisků kláves a zobrazení přes záznamy, jelikož ostatní funkce, jako pořizování snímků obrazovky, odesílání výstupu a filtrování, jsou u běžně dostupných keyloggerů zpoplatněny lepšími verzemi produktu.

Výstup monitorování kláves je oddělen jako nový záznam v závislosti na otevřené aplikaci a dále u webových prohlížečů na momentálně aktivním okně. Na výstupu je možno vidět označení otevřené aplikace a následně popis otevřeného okna a opět aplikace, ve které je otevřené. Výstup, který lze zobrazit v programu, pak vypadá následovně (Obrázek 4):



Obrázek 4. Záznam softwarového keyloggeru

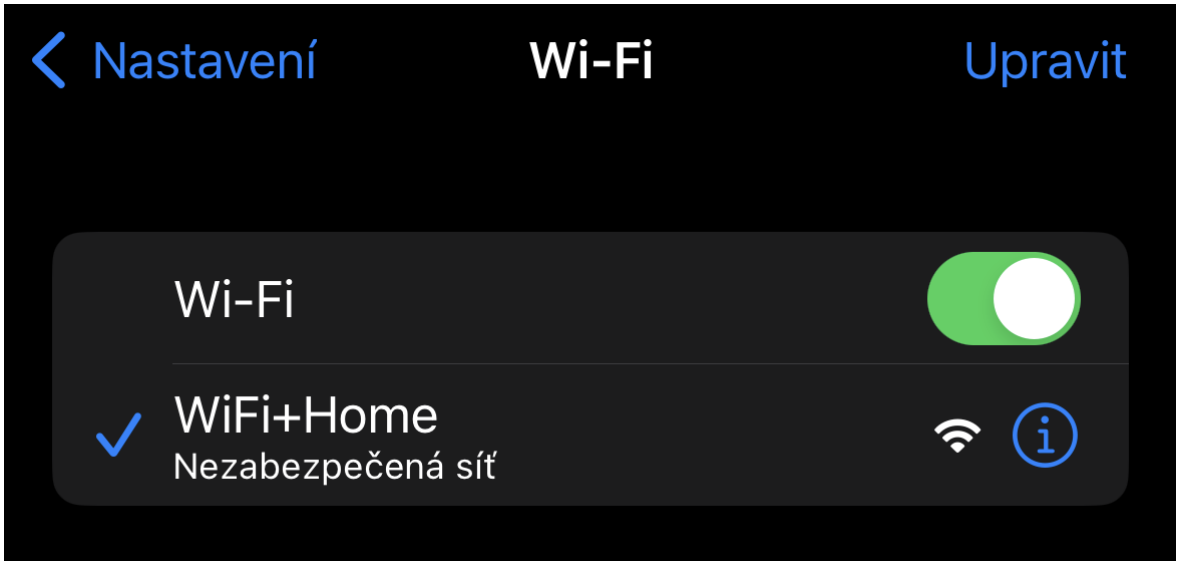
Výstupy tohoto typu lze i uložit do zařízení, a to ve formátech html a rvl. Při uložení ve formátu html je záznam zobrazitelný ve webovém prohlížeči ve stejně stylované podobě jako v programu. Pokud by byl záznam uložen ve formátu rvl, tak jej lze pouze importovat do programu keyloggeru.

6.2 Hardwarový

Hardwarovým keyloggerem, který byl použit pro testování byl Keygrabber Air USB. Tento hardwarový prvek je prodáván na internetových stránkách keydemon.com, které spadají pod společnost Electroware[25]. Jedná se o keylogger velmi malých rozměrů, který je připojen přímo do počítače a do něj je připojena klávesnice počítače. Zařízení je tímto způsobem napájeno a začne vysílat signál wifi. Na tuto síť je poté možno se připojit a pak přejít na IP adresu, na které lze zařízení nastavit. U zařízení je možné nastavit wifi módy, což znamená, jestli bude zařízení pouze přístupovým bodem, na který se lze připojit, nebo se bude moct samo připojit k jiné wifi síti. Pokud by zařízení bylo pouze přístupovým bodem, tak by to znamenalo, že se k němu lze pouze připojit a procházet záznamy stisků klávesnice. Tento mód má i svou variantu se skrytou SSID (identifikátor bezdrátové sítě). Pro obě varianty lze dodatečně nastavit zabezpečení sítě v podobě přístupového hesla. Pokud by zařízení bylo v módu wifi klienta, tak je zařízení připojeno na jinou síť wifi, přes kterou následně posílá záznamy na email nebo pomocí FTP (File Transfer Protocol). V tomto módu lze nastavit, jak často bude zařízení posílat záznamy a jakou maximální velikost bude mít soubor se záznamem.

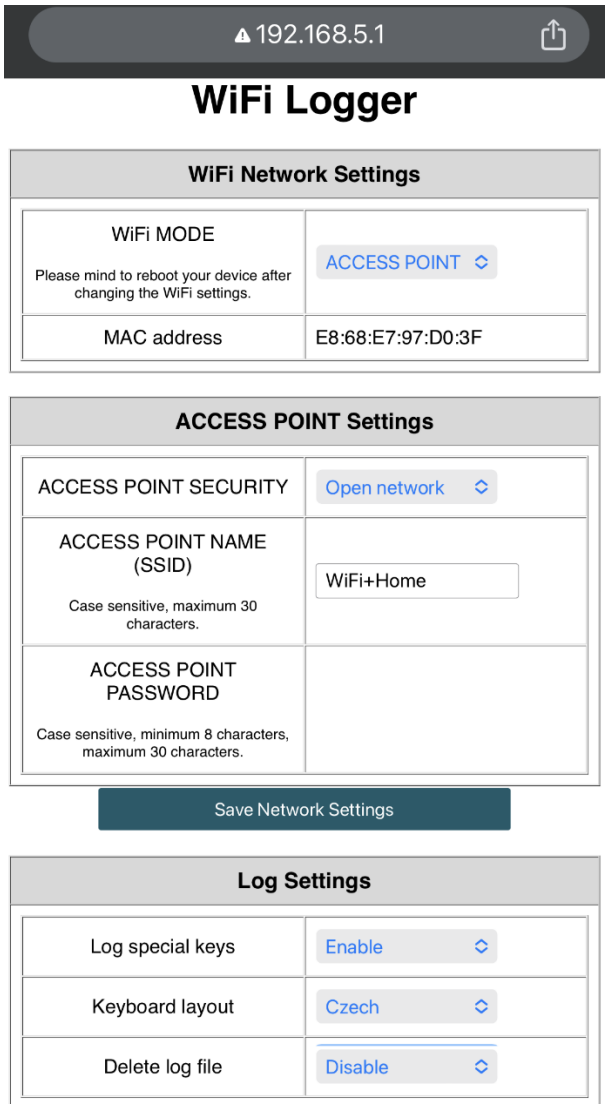
Pokud je zařízení prvně připojeno, tak je v základu nastaveno v módu přístupového bodu s SSID „AP001“ a bez přístupového hesla. Po připojení na síť s tímto SSID se možnosti konfigurace nachází spolu se záznamy na IP adrese „192.168.5.1. Vzhledem k tomu, že síť vysílá signál typu wifi, tak bylo nutno jej konfigurovat z mobilního zařízení.

Zařízení se tedy zdá jako wifi síť (Obrázek 5):



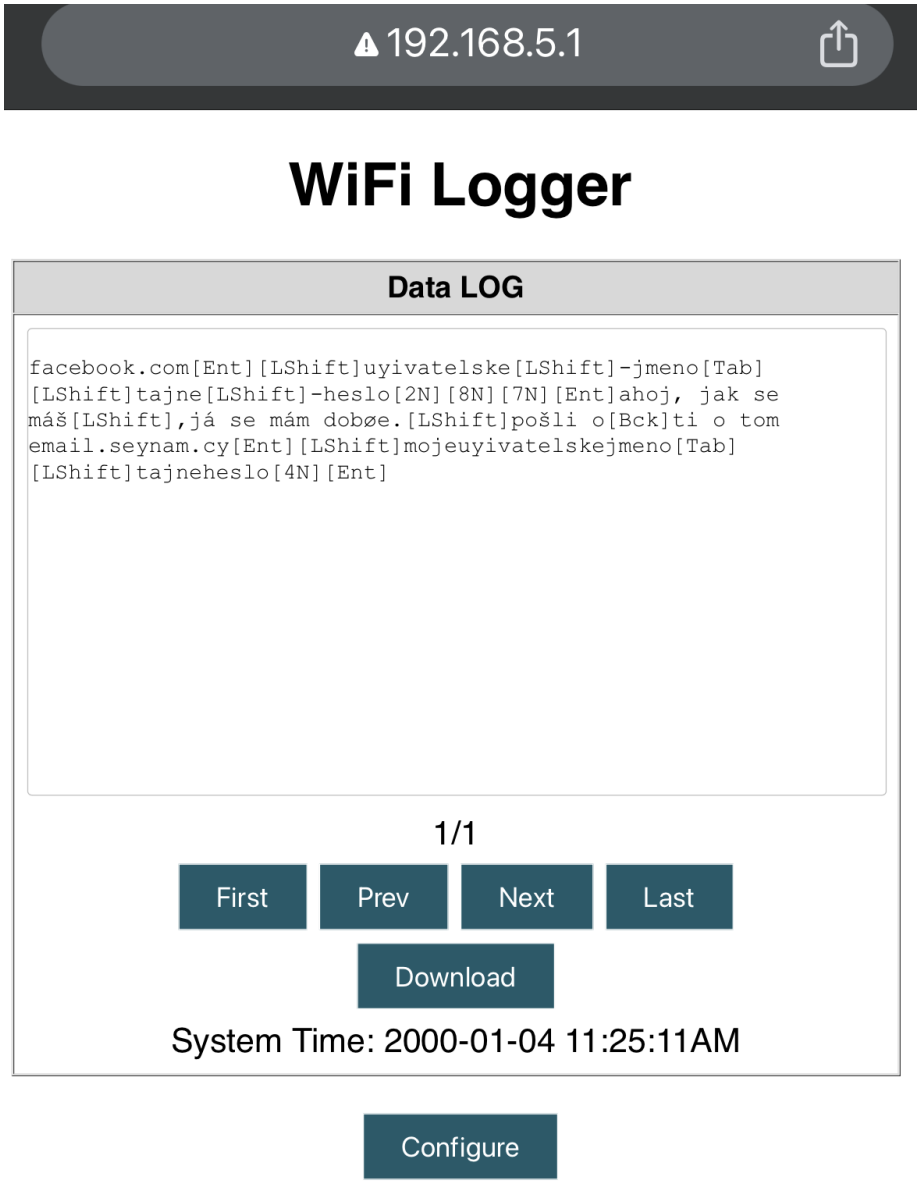
Obrázek 5. Připojení k hardwarovému keyloggeru

Po připojení nabízí a zadání IP adresy do webového prohlížeče je po stisku tlačítka „Configure“ možno nastavit následující parametry (Obrázek 6):



Obrázek 6. Konfigurace hardwarového keyloggeru

Po nastavení požadovaných parametrů lze prohlížet záznamy ve webovém rozhraní na dané IP adrese (Obrázek 7) nebo je stáhnout v textové podobě a formátu txt. Záznamy stisků pak vypadají následovně:



Obrázek 7. Záznam hardwarového keyloggeru

Zařízení detekuje veškeré speciální znaky, ale jejich kombinace rozpoznat neumí. Takže pokud je souběžně stisknuta klávesa shift a písmeno, tak to zařízení vnímá jako dva po sobě jdoucí stisky kláves. I přes možnosti nastavení na české rozložení klávesnice jsou zaměňovány klávesy reprezentující Z a Y.

7 PROLAMOVÁNÍ NA DOSTUPNÝCH ZAŘÍZENÍCH

Na všech dostupných zařízeních byly provedeny benchmarky, které popisují výkonnost zařízení při prolamování uživatelských hesel. Výkonnost zařízení je určena v rychlosti, kterou je zařízení schopno generovat hashe. Tato rychlost je ovlivněna několika faktory a pro každou funkci vyplývá z jiných vstupních parametrů.

Vstupní parametry:

Akcelerace (CPU:GPU) – vyjadřuje hodnotu, kterou je možno nastavit poměr zatížení GPU ku CPU. Tato hodnota v podstatě reprezentuje kolikrát více operací bude GPU zpracovávat oproti CPU. Pokud by se tato hodnota rovnala číslu 1, tak by oba komponenty vykonávaly operace ve stejném poměru. Některé algoritmy, kterými se zabezpečují hesla, využívají jednodušších matematických výpočtů. Například algoritmus MD5 využívá pouze pár matematických výpočtů, které se opakují v cyklech, ale třeba Bcrypt využívá složitějšího algoritmu. Takovou informaci je nutno brát v potaz při nastavování vstupních parametrů, protože každá funkce je jinak náročná a rychlá. Tudíž u rychlejších funkcí je výhodnější nastavit vyšší zátěž na GPU, jelikož jde pouze o opakující se cykly výpočtů, ale u těch pomalejších, které bývají navrženy jako pomalejší, což jsou zpravidla key derivation funkce, je vhodnější nastavit hodnotu nižší.[26]

Počet iterací – tento parametr určuje kolikrát se bude určená funkce opakovat, než dojde k výsledné hodnotě hashe. Ve většině případů není známo, jaká hodnota byla nastavena, takže je potřeba zkoušet vícekrát. Při benchmarku je ale tato hodnota předem určená, aby demonstrovala odolnost funkce. Pro algoritmus MD5 je tato hodnota nastavena na číslo 1024, což hodnota potřebná pro zajištění minimální bezpečnosti. Oproti tomu hodnota pro algoritmus Bcrypt je nastavena na číslo 32, což je hodnota, která je považována za dostatečnou úroveň zabezpečení pro běžné účely, jako je zabezpečení přihlášení do účtů webových stránek.

Počet ALU jednotek – tato hodnota určuje maximální možný počet ALU jednotek, které je GPU schopno využít k paralelním výpočtům. Každá grafická karta je tvořena z několika streaming procesorů, které obsahují určitý počet ALU jednotek. Například karta GTX 1080, která je využita v této práci obsahuje 20 streaming procesorů a v každém z nich se nachází 128 ALU jednotek. Díky této informaci je možno odvodit, že karta obsahuje celkový počet 2560 ALU jednotek neboli cuda jader. Nastavení této hodnoty určuje, jaký maximální počet ALU jednotek je možno použít na paralelní výpočty, nejedná se však o maximální počet

jednotek na kartě. Příklad může být nastavení hodnoty na číslo 256 na kartě GTX 1080, která obsahuje 2560 ALU jednotek. Nastavení na takovou hodnotu by znamenalo, že karta by byla schopna využít 256 ALU jednotek pro potřeby paralelních výpočtů. Tato hodnota by poté mohla být pro tuto kartu zvolena jako optimální při prolamování například algoritmu MD5 v rámci benchmarku. Nastavením tohoto parametru na vyšší hodnotu je možno docílit vyššího výkonu při prolamování některých algoritmů, ale u algoritmů, které jsou pomalejší, by mohlo naopak dojít ke snížení výkonu.

Počet vektorových jednotek – poslední parametr, který je pro benchmark nastaven určuje, kolik vektorových jednotek by se podílelo na výpočtech nebo také s jak velkým slovem by jedna vektorová operace pracovala. Tento parametr může nabývat hodnot 1,2,4,8 a 16, přičemž hodnota 1 znamená, že jedna operace bude pracovat s jedním 32-bitovým slovem a jednou vektorovou jednotkou. Pokud by tento parametr byl nastaven na vyšší hodnotu jako je 8, tak by vektorová operace pracovala s 8 vektorovými jednotkami a zpracovávala by osm 32-bitových slov paralelně. Nastavení hodnoty tohoto parametru záleží na tom, jak jsou jednotlivé algoritmy navrženy. Například algoritmus MD5 je navržen tak, že využívá větší množství dat a je jednoduše paralelizovatelný, tudíž je možné nastavení vyšších hodnot. Na druhou stranu algoritmy typu SHA jsou navrženy tak, aby se zpracovávalo jedno vlákno dat po druhém, takže by pro tyto algoritmy byla vhodnější volba nižší hodnoty.

7.1 Použitá zařízení

K demonstrování možností prolamování uživatelských hesel byla vybrána tři zařízení. Každé z použitých zařízení se liší ve velikosti, výkonu a využití. Byl vybrán stolní počítač, herní notebook a mini počítač Raspberry Pi. Byla vybrána zařízení o velmi odlišných výkonech proto, aby bylo možno demonstrovat, že některé funkce nezaručují dostatečné zabezpečení a k jejich prolomení není potřeba nejvyššího výkonu.

7.1.1 Skládaný stolní počítač

Stolní počítač, na kterém byla demonstrována schopnost prolamování uživatelských hesel, obsahuje komponenty, které nepatří mezi nejnovější dnešní doby, ale jsou stále použitelné a pro účely práce dostačující.

Komponenty:

Procesor - v této sestavě je procesor od společnosti Intel řady i7 a označením 8700K. Tento procesor má zabudováno šest fyzických jader s čtrnácti nanometrovou architekturou Coffee

Lake, kde Intel osazuje jedno jádro šesti ALU jednotkami, tudíž jich procesor má třicet šest. Pracovní frekvence je uváděna 3,7 GHz a turbo frekvence neboli frekvence při automatickém taktování může dosahovat až 4,7 GHZ.[27]

Grafická karta – grafickou kartou v této sestavě je karta GTX 1080 od společnosti Nvidia. Tato karta má osazeno dvacet stream procesorů, kde každý obsahuje 128 ALU jednotek a celkově jich tedy karta má 2560. Pracovní frekvence čipu této grafické karty je 1,8 GHz. Pracovní paměť této grafické karty je o velikosti 8 192 MB.[28]

Operační paměť – sestava je osazena šestnácti gigabajty operační paměti typu DDR4 a pracovní rychlostí 2400Mhz.

7.1.2 Herní notebook

Notebook, použitý k testování schopnosti prolamování hesel na přenosných zařízeních, je herní notebook od výrobce Lenovo z řady Legion a označením Y530-15ICH. Vzhledem k tomu, že se jedná o herní notebook, tak oproti kancelářským notebookům poskytuje lepší výkon a hodí se k těmto testům.

Komponenty:

Procesor – notebook je osazen procesorem od společnosti Intel z řady i5 s označením 8300H. Procesor je osazen čtyřmi fyzickými jádry a má osm vláken. Jedná se o architekturu Coffee Lake, kde je každé jádro osazeno šesti ALU jednotkami a procesor jich tedy má dvacet čtyři. Pracovní frekvence procesoru je 2,3 GHz a turbo frekvence může dosahovat až 4GHz.[29]

Grafická karta – v tomto notebooku se nachází grafická karta od společnosti Nvidia s označením GTX 1050 Ti. Na kartě se nachází 6 stream procesorů, kde každý obsahuje 128 ALU jednotek. Celkový počet ALU jednotek na kartě je tedy 768. Pracovní frekvence čipu karty je 1,29 GHz. Pracovní paměť grafické karty je o velikosti 4 096 MB.[30]

Operační paměť – v notebooku se nachází osm gigabajtů operační paměti typu DDR4 s pracovní rychlostí 2666 MHz.

7.1.3 Raspberry Pi 4 B

Raspberry Pi je minipočítač, který je navržen tak, aby byl přenositelný a v určité míře využitelný pro každou situaci. Veškeré komponenty tohoto počítače se nachází na jedné desce. Původně byl minipočítač vyvinut pro výuku s účely řízení zařízení jako je mikrovlnná trouba

či pračka. V dnešní době však existuje několik modelů tohoto minipočítače s tím, že některé jsou přímo uzpůsobeny k tomu, aby byly kontrolery. Model, který je použit pro účely testování, je kompletním kompaktním počítačem, který obsahuje v jednom chipu procesor, grafickou kartu i paměti.

Komponenty:

Procesor – minipočítač obsahuje procesor Broadcom BCM2711, který je architektury ARM, která je používána u mobilních telefonů. Procesor je osazen čtyřmi jádry a architektura procesoru podporuje pouze jedno vlákno na jedno jádro, tudíž obsahuje čtyři vlákna. Procesor je osazen dvěma ALU jednotkami na jedno jádro a celkový počet těchto jednotek je roven osmi. Pracovní frekvence procesoru je v základním režimu 1,5 GHz, ale může být vyšší nebo nižší v závislosti na zátěži a teplotě procesoru.[31]

Grafická karta – minipočítač nemá samostatný čip pro grafickou kartu. Grafická karta je součástí čipu, ve kterém se nachází i procesor, operační paměť a řadiče. Grafický čip má k dispozici šedesát čtyři ALU jednotek, které jsou určeny ke grafickým výpočtům. Pracovní frekvence grafického čipu je 500 MHz.[32]

Operační paměť – operační paměť je na tomto minipočítači sdílená pro procesor, grafický čip i řadiče. Velikost operační paměti závisí na daném modelu Raspberry Pi. U modelu 4 B lze zvolit mezi dvěma, čtyřmi nebo osmi gigabajty sdílené operační paměti. Model, na kterém jsou provedeny testy má 4 gigabajty.[31]

7.2 Benchmarkové testy

Pro účely demonstrování byl vybrán program Hashcat a to pro jeho výkon v tomto využití a taktéž různé možnosti útoků a hashovacích algoritmů. V první řadě byly provedeny benchmarkové testy, které určují, jakých maximálních rychlostí hashování je dané zařízení schopno dosáhnout. Tyto hodnoty však nemusí být hodnotami, kterých bude program dosahovat při reálném útoku. Skutečné hodnoty při útoku se odvíjí od vstupních parametrů a počtu kombinací znaků. Vstupní parametry pro benchmarkové testy jsou zvoleny pro vysoký výkon a optimální dobu jednoho testu.

Skládaný stolní počítač:

Tabulka 1. Benchmark stolního počítače

Hashovací funkce	**Hashovací rychlost**	**CPU:GPU**	**Iterace**	**ALU**	**Vektorové jednotky**
MD5	25 673 MH/s	256	1024	256	8
SHA1	8 618 MH/s	128	1024	256	1
SHA-256	3 137 MH/s	32	1024	256	1
SHA-512	950 MH/s	256	64	256	1
NTLM	43 606 MH/s	256	1024	256	8
Bcrypt	20 967 H/s	8	32	11	1

Herní notebook:

Tabulka 2. Benchmark herního notebooku

Hashovací funkce	**Hashovací rychlost**	**CPU:GPU**	**Iterace**	**ALU**	**Vektorové jednotky**
MD5	7 221 MH/s	256	512	256	8
SHA1	2 447 MH/s	128	1024	256	1
SHA-256	868 MH/s	32	1024	256	1
SHA-512	267 MH/s	64	256	256	1
NTLM	12 284 MH/s	256	1024	256	8
Bcrypt	5 686 H/s	8	32	11	1

Raspberry Pi 4 B:

Tabulka 3. Benchmark Raspberry Pi

Hashovací funkce	**Hashovací rychlost**	**CPU:GPU**	**Iterace**	**ALU**	**Vektorové jednotky**
MD5	26 022,7 kH/s	256	1024	1	4
SHA1	11 521 kH/s	256	1024	1	4
SHA-256	6 973,3 kH/s	128	1024	1	4
SHA-512	1 562,8 kH/s	64	512	1	2
NTLM	38 001,8 kH/s	256	1024	1	4
Bcrypt	13 H/s	4	32	1	1

Hlavním výstupem benchmarku je hodnota hashovací rychlosti, která určuje kolik hashů dokáže použité zařízení vygenerovat za jednu sekundu při stanovených parametrech. Tato hodnota se udává v jednotkách Hash za sekundu. Ze získaného výstupu lze dle této hodnoty určit, která funkce je nejméně náročná na výpočty, a tudíž i nejméně bezpečná, protože byla zařízení schopna vygenerovat mnohem více hashů než u funkcí ostatních. V tomto případě lze tedy určit, že nejméně bezpečná z testovaných funkcí je funkce NTLM, což souvisí s tím, jak je funkce navržena. Funkce NTLM je navržena tak, že používá rychlých matematických a logických operací, které nepracují s velkými bloky dat a díky tomu je velmi vhodná pro paralelní výpočty. Další její slabinou je, že její výstupní hash je složen pouze z hexadecimálních znaků.

Jako nejvíce bezpečnou funkci z testovaných lze určit funkci Bcrypt. Na rozdíl od ostatních funkcí, kde hashovací rychlost dosahovala vysokých hodnot, tak pro funkci Bcrypt byla zařízení schopna dosáhnout znatelně nižších hodnot. Vstupní parametry, které byly určeny pro funkci Bcrypt se velmi liší od parametrů pro funkce ostatní, protože Bcrypt je odlišný druh funkce a je záměrně navržen tak, aby byl jeho výpočet pomalejší. To ale není jediný důvod hodnot zvolených pro Bcrypt. Počet iterací byl zvolen takový, aby vygenerování jednoho hashe nebylo příliš zdlouhavé, ale zároveň aby netrvalo příliš krátkou dobu. Nižší počet vláken neboli ALU jednotek byl programem zvolen z toho důvodu, že Bcrypt během svého výpočtu pracuje s více než jednou solí, které ukládá do paměti. Algoritmus je tedy více

náročný na využití paměti a volba vyšší hodnoty by mohla paměť přehltit. Z důvodu zvolení hodnot pro paralelní výpočty na nižší čísla byla hodnota pro počet vektorových jednotek zvolena také na nižší, protože v případu malého paralelismu se tento parametr nevyplatí nastavovat na vyšší hodnotu.

7.3 Testy na konkrétních hashích

Pro demonstrování výkonnosti prolamování hesel je potřeba provést testy s hesly a nastavením pracovních profilů. Testy jsou provedeny na hashích MD5, SHA1, Bcrypt a NTLM. Na hashe je prováděn útok hrubou silou s použitím masky. Pro každý hash jsou provedeny dva testy. První heslo je o délce osmi znaků, což je v dnešní době považováno za bezpečné. Druhé heslo je o délce deseti znaků. Při určení hesel a nastavení masky je brán ohled na styl, kterým lidé tvoří v současné době svá hesla. To znamená, že na prvním místě je velké písmeno, následující znaky jsou náhodné a poslední čtyři znaky jsou čísla. Pro veškeré testy je využit pracovní režim, který využívá výkon použitého zařízení maximálním způsobem pro danou funkci. Prvním testovaným heslem je „Qwer1986“. Písmena „qwer“ jsou vybrána proto, že se nachází v kombinaci s čísly v seznamu nejpoužívanějších hesel roku 2022. Druhým heslem je „P@$WrD2012“. Heslo je vytvořeno jako složitější varianta k nejpoužívanějšímu heslu na světě, které je „password“. Během prvního testu je vždy testováno první heslo na hashi MD5 a následují hashe SHA1, NTLM a Bcrypt. Stejné pořadí hashů je použito i pro heslo druhé.

Pro vytvoření všech hashů pro dané funkce byly použity webové generátory. Pro každý útok bylo potřeba zadat, jaký typ útoku bude použit, jakého formátu je daný hash, jaký bude použit pracovní profil a maska, podle které budou dosazovány znaky. Program hashcat nezobrazuje pouze výsledný výstup, ale na vyžádání uživatelem i momentální postup prolamování. Výstup nebo momentální postup jsou zobrazeny v příkazové řádce a vypadají následovně (Obrázek 8):

```
ac5935173b88af5cc6d4c2ddbdb326f8:Qwer1986

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: ac5935173b88af5cc6d4c2ddbdb326f8
Time.Started.....: Thu May 04 16:49:40 2023 (6 secs)
Time.Estimated...: Thu May 04 16:49:46 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?u?a?a?a?d?d?d?d [8]
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 10398.0 MH/s (22.94ms) @ Accel:256 Loops:1024 Thr:256 Vec:1
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 55972575000/222917500000 (25.11%)
Rejected.........: 0/55972575000 (0.00%)
Restore.Point....: 237500/950000 (25.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1024 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: MARe1921 -> TQU 9733
Hardware.Mon.#1..: Temp: 70c Fan: 47% Util: 99% Core:1683MHz Mem:5005MHz Bus:16

Started: Thu May 04 16:49:34 2023
Stopped: Thu May 04 16:49:46 2023
```

Obrázek 8. Výstup programu hashcat

Na zvýrazněných řádcích (Obrázek 8) lze vyčíst data, která určují, jaké byly nastaveny parametry pro útok na danou funkci a jaká je odhadovaná doba prolomení. Data z každého provedeného útoku byla zanesena do tabulky (Příloha 1), kde je lze porovnat. S ohledem na tato data lze určit, že hesla, která mají pouze osm znaků a jsou tvořena předvídatelným způsobem jsou u všech funkcí, kromě funkce Bcrypt, prolomena velmi rychle. Na stolním počítači i na notebooku se jedná o řády sekund. Na minipočítači Raspberry Pi je tento čas prodloužen na pár hodin, což je ovšem taktéž přijatelné, pokud není heslo vyžadováno ihned. Hesla, která mají znaků deset, ale jsou stále tvořena předvídatelným způsobem a chráněna rychlejšími funkcemi, by byla na stolním počítači a notebooku prolomena během pár dnů, což znamená, že tyto funkce neposkytují dostatečnou ochranu. Na minipočítači Raspberry Pi už se doba prolomení takových hesel zvedá na pár let, což ale souvisí s nízkým výpočetním výkonem. Z dat lze určit, že funkce Bcrypt je z testovaných funkcí jediná, která poskytuje dostatečné zabezpečení. To je způsobeno tím, že se nejedná o funkci hashovací, ale o key derivation funkci, které jsou záměrně zpomalené a nedovolují útočníkovi využívat plnou kapacitu výkonu hardwaru. To je možno vidět na nastavení hodnot GPU akcelerace a počtu ALU jednotek. Pro funkci Bcrypt byly tyto hodnoty nastaveny na nižší než u funkcí ostatních, jelikož tato funkce využívá pro výpočty více paměti. Využití více paměti nedovoluje nastavit hodnoty počtu ALU jednotek na příliš vysoké, jelikož každá ALU jednotka musí pracovat s určitým objemem paměti.

Na základě těchto dat lze tedy určit, že hashovací funkce nejsou pro zabezpečení hesel tolik vhodné, jako key derivation funkce. Dále lze určit, že hesla o délce osmi znaků již nejsou dostatečně dlouhá, a tudíž by se měla volit hesla delší.

8 BEZPEČNÉ HESLO

Pojem bezpečné heslo lze chápat taktéž jako heslo, které je dostatečně zabezpečeno. Z tohoto důvodu je možno tuto kapitolu rozdělit do dvou částí. V první části bude určováno, jak by mělo být tvořeno heslo, které bude dostatečně bezpečné z pohledů délky, komplexnosti a nenáchylnosti k předpokladům tvorby hesel. Druhá část bude popisovat, jak by měla být hesla zabezpečena ze strany správce. V této části budou probrány možnosti použitelných funkcí a ukládání hesel.

8.1 Jak vytvořit bezpečné heslo

Pro tvorbu bezpečného hesla by se měl uživatel řídit několika pravidly. Heslo by mělo být dostatečně dlouhé. V dnešní době je předpokladem, že hesla, která jsou dlouhá osm znaků jsou dostatečně bezpečná. Tento předpoklad vzniká z faktu, že webové stránky po uživatelích požadují délku hesla minimálně o osmi znacích. Většina uživatelů nezadává delší hesla, než je požadované minimum. Ve většině případů využívá stále stejná hesla a pokud je po nich vyžadováno více znaků, tak heslo pouze doplní o pár číslic. V dnešní době by hesla, která zaručují alespoň minimální dostatečnou ochranu, měla být o dvanácti znacích. Dále by hesla neměla být předvídatelná. Dalším požadavkem, který webové stránky většinou vyžadují, je použití alespoň jednoho velkého písmena a jednoho čísla. V tomto případě použije většina uživatelů velké písmeno na první pozici hesla, což je pro potencionální útočníky předvídatelné. Obdobně většina uživatelů umístí číslici na pozici poslední. Taková předvídatelnost útočníkům velmi pomáhá s nastavením masky pro útoky hrubou silou. Dalším pravidlem je náhodnost hesla. Uživatelé by do svých hesel neměli vkládat žádné lehce zjistitelné osobní údaje, jako jsou jména rodinných příslušníků, domácích mazlíčků nebo časové údaje určující dny, měsíce a roky. Hesla by taktéž neměla obsahovat žádná slova, která se nachází ve slovnících a jsou v běžném životě používána. Pokud by takové údaje byly vloženy do hesla, je velmi snadné je zakomponovat do různých slovníků a použít je při útoku. To stejné platí pro existující slova, jelikož existuje velký počet slovníků, které tato slova obsahují, a to v mnohých jazycích.

Pokud by se tedy uživatelé řídili těmito pravidly, tak by byly schopni vytvořit hesla, která nejsou náchylná na slovníkové útoky. Pokud webové stránky požadují užití určitých znaků, tak by je uživatelé neměli vkládat na předvídatelné pozice. Je mnohem vhodnější umístit velké písmeno doprostřed hesla než na jeho začátek. To stejné platí pro číslice, které je lepší použít doprostřed hesla než na jeho konec. Bezpečná hesla by tedy měly vypadat spíše jako

náhodné řetězce znaků. Taková hesla sice nejsou jednoduchá na zapamatování, ale v takovém případě je vhodné použít password managery.

8.2 Jak zabezpečit hesla

Pro vhodné zabezpečení hesla je nutno vybrat dostatečně bezpečnou funkci. Takové funkce by měly být náročné na výpočty, aby jejich průběh nebyl příliš rychlý a jednoduchý. Výstup z těchto funkcí by také neměl být příliš krátký, aby nebyl možný vznik kolizí. Dále je vhodné volit funkce, které je možné modifikovat na základě vstupních parametrů. Možnost modifikování dodává těmto funkcím další vrstvu ochrany. Čím více modifikací funkce nabízí, tím více je zabezpečená. Hodnoty těchto modifikací by měly být voleny s ohledem na výpočetní možnosti zařízení, na kterém budou výpočty uskutečňovány, a také s ohledem na odezvu, za kterou je vytvořen výsledný hash. Funkce, které poskytují takovou úroveň ochrany, jsou zpravidla key derivation. Funkcí, která poskytuje nejlepší možnou ochranu, je Argon2. Tato funkce poskytuje několik modifikací, které zaručují ochranu před několika typy útoků. Tvorba výstupu z této funkce však trvá podstatně déle než u funkcí ostatních. Z tohoto důvodu ji někteří nevolí k zabezpečování, jelikož vyžadují nižší dobu odezvy. Velká většina key derivation funkcí taktéž v základu používá soli, což je ochrana proti útokům rainbow tables a zpomalení útoků hrubou silou.

Již vytvořené hashe funkcí by měly být ukládány v databázích, do kterých bude šifrovaný přístup, aby nebylo jednoduché se k hashům dostat. Výstupem z key derivation funkcí je i použitá sůl. U některých funkcí je neoddělitelnou součástí hashe, jelikož se využívá i k ověřování. Avšak u funkcí, kdy sůl nemusí být součástí hashe, je vhodnější ji ukládat odděleně. Například u funkce Bcrypt je sůl součástí výsledného hashe, kde se ale nenachází v podobě, která byla použita pro výpočty. Další možností ochrany hesel je použití takzvaného pepře, což však není tak vhodné, jelikož pepř je zakomponován přímo na webových stránkách a používá se pro každé heslo stejný. Takže pokud je hodnota pepře odhalena, tak přichází o svůj význam. Vyšší ochrany účtů lze kromě hesel docílit i použitím dvoufázového ověření pomocí sms kódu nebo kódu zaslaného na email.

ZÁVĚR

Cílem práce bylo popsat problematiku zabezpečování uživatelských hesel, jejich následné prolamování a popis tvorby a zabezpečení bezpečného hesla.

Teoretická část se nejprve věnuje způsobům zabezpečení uživatelských hesel, kde je popsáno, jaké kryptografické funkce jsou k zabezpečování hesel v současné době využívány. Zde jsou popsány dvě skupiny funkcí s jednotlivými zástupci. Dále jsou popsány základní principy, pomocí kterých tyto funkce zpracovávají vstupní data a převádějí je na zabezpečený výstup. U těchto funkcí jsou popsány jejich výhody i nevýhody v rámci zabezpečení a implementace. Také jsou popsány dvě další možnosti, kterými je možno dosáhnout vyšší úrovně zabezpečení hesel.

V druhé části teoretického přehledu jsou charakterizovány nástroje, kterými je možné na zabezpečené výstupy útočit a pokusit se o odhalení originálního hesla. Jsou popsány dva druhy nástrojů včetně jejich vzájemných rozdílů. Tyto nástroje plní stejné úkoly, ale s každým je pracováno jiným způsobem. V další části jsou popsány útoky, které je možné s pomocí těchto nástrojů provádět. U jednotlivých útoků je popsán princip, jakým na hesla útočí. Je popsán i nástroj, který má za úkol zachytit znaky z klávesnice, a tudíž pomocí něj lze získat originální podobu hesla. U tohoto nástroje je popsána jeho softwarová i hardwarová forma a také jak se liší ve funkčnosti. Nakonec se teoretická část zaobírá způsoby prolamování hesel v rámci použitých komponent a zařízení, u kterých jsou popsány rozdíly a výhody využití jednotlivých zařízení.

Praktická část práce se věnuje porovnání výkonnosti prolamování hesel pomocí centrální procesorové jednotky a grafického čipu. Toto porovnání je uskutečněno formou benchmarkového testu a následného srovnání dosažených hashovacích rychlostí. Dále jsou demonstrovány obě formy zařízení, které zachycují znaky z klávesnice, také je popsáno, jakým způsobem fungují, jak je možno je přizpůsobit a jaké vytvářejí výstupy. Následně je demonstrována výkonost vybraných zařízení pro prolamování hesel. To je uskutečněno nejprve formou benchmarkových testů, pomocí kterých jsou zjištěny přibližné hashovací rychlosti a optimální hodnoty vstupních parametrů. Následně jsou uskutečněny reálné testy na dvou předem určených heslech o různých délkách. Tyto testy jsou uskutečněny pro nejpoužívanější zabezpečovací funkce. Výsledky těchto testů jsou vloženy do tabulky, kde je možné porovnat, jakých hashovacích rychlostí bylo dosaženo a jaké byly nastaveny optimální parametry při maximální zátěži hardwaru.

Z výsledků jednotlivých testů je definováno, že jedna skupina zabezpečovacích funkcí již není pro zabezpečování hesel příliš vhodná. Z výsledků je také možno určit, že předpoklad délky bezpečného hesla není při použití těchto funkcí aktuální. Po zvážení získaných výsledků je uvedeno, jakým způsobem by měla být hesla tvořena, která hesla je možné považovat za silná, a jakým způsobem tato hesla správně zabezpečit. Je popsáno, jakým stereotypům při tvorbě hesel se vyhnout a jaké funkce rozhodně nepoužívat.

Závěrečná práce zájemce zasvětí do problematiky zabezpečení a zacházení s uživatelskými hesly. Práce by mohla sloužit jako ukázka možností odcizení uživatelských účtů pomocí napadení hesel při zvolení nedostatečně silného hesla. Práce by mohla sloužit taktéž jako návod, jakým způsobem tvořit hesla odolná na dané typy útoků a čeho se určitě vyvarovat. Z výsledků práce také vyplývá, jak daná hesla správně zabezpečit ze strany správce a jakým zabezpečovacím funkcím se při implementaci vyhnout.

SEZNAM POUŽITÉ LITERATURY

[1] Cryptographic Hash Functions: A Review [online]. 9. IJCSI International Journal of Computer Science Issues, 2012 [cit. 2023-05-17]. ISSN 1694-0814. Dostupné z: www.IJCSI.org

[2] KUMAR, Himanshu, Sudhanshu KUMAR, Remya JOSEPH, Dhananjay KUMAR, Sunil Kumar SHRINARAYAN SINGH, Ajay KUMAR a Praveen KUMAR. Rainbow table to crack password using MD5 hashing algorithm [online]. In: . IEEE, 2013, 2013, s. 433-439 [cit. 2023-02-09]. ISBN 978-1-4673-5759-3. Dostupné z: doi:10.1109/CICT.2013.6558135

[3] Announcing the first SHA1 collision [online]. Mountain View, Kalifornie, USA: Google.com, 2017 [cit. 2023-05-17]. Dostupné z: https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

[4] Implementation and Performance Analysis of PBKDF2, Bcrypt, Scrypt Algorithms. In: Proquest.com [online]. Athens: Athens: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016, 2016 [cit. 2023-05-17]. Dostupné z: https://www.proquest.com/docview/1817061029/fulltext-PDF/EBB4A56654F74D2BPQ/1?accountid=15518

[5] PROVIDING PASSWORD SECURITY BY SALTED PASSWORD HASHING USING BCRYPT ALGORITHM [online]. 2015. ©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved., 2015 [cit. 2023-02-09]. ISSN 1819-6608. Dostupné z: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=9d82620f0866ec773b12c30f90d70b74661f972f

[6] BIRYUKOV, Alex, Daniel DINU a Dmitry KHOVRATOVICH. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications. In: BIRYUKOV, Alex a Dmitry KHOVRATOVICH. Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications [online]. 1. Saarbruecken, Germany: IEEE, 2016, 2016, s. 292-302 [cit. 2023-05-17]. ISBN 978-1-5090-1751-5. Dostupné z: doi:10.1109/EuroSP.2016.31

[7] SCOTT, Benjamin. Learning Password Security Jargon: Password Peppering. In: NordPass [online]. Litva: Nord Security, 2012, 12.8.2020 [cit. 2023-05-20]. Dostupné z: https://nordpass.com/blog/pepper-password/

[8] Hashcat. Hashcat [online]. -: hashcat, 2016 [cit. 2023-02-21]. Dostupné z: https://hashcat.net/hashcat/

[9] HashSuite [online]. -: Copyright (c) 2011-2023 Alain Espinosa, 2023 [cit. 2023-05-17]. Dostupné z: https://hashsuite.openwall.net/

[10] Rockyou. In: Github [online]. Internet: hob0, 2016 [cit. 2023-02-21]. Dostupné z: https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt

[11] Hashcat wiki [online]. -: hashcat.net, 2016 [cit. 2023-05-17]. Dostupné z: https://hashcat.net/wiki/

[12] Rainbow tables. Ionos [online]. -: © 2023 IONOS, 2023 [cit. 2023-02-24]. Dostupné z: https://www.ionos.com/digitalguide/server/security/rainbow-tables/

[13] RAHIM, Robbi, Heri NURDIYANTO, Ansari SALEH A, Dahlan ABDULLAH, Dedy HARTAMA a Darmawan NAPITUPULU. Keylogger Application to Monitoring Users Activity with Exact String Matching Algorithm. 954. In: Journal of Physics: Conference Series [online]. 954. Makassar, Indonesia: KO2PI, 2018, - [cit. 2023-05-19]. ISBN -. ISSN 1742-6588. Dostupné z: doi:10.1088/1742-6596/954/1/012008

[14] FRUHLINGER, Josh. Keyloggers explained: How attackers record computer inputs. CSO security news [online]. -: IDG Communications, 2022 [cit. 2023-05-19]. Dostupné z: https://www.csoonline.com/article/3326304/keyloggers-explained-how-attackers-record-computer-inputs.html

[15] BELLINI, Stefan. A Hardware keylogger for USB-Keyboards. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2016 [cit. 2023-05-20]. Dostupné z: https://en.wikipedia.org/wiki/Hardware_keylogger#/media/File:USB_Hardware_Keylogger.jpg

[16] KeyGrabber - Hardware Keylogger - WiFi USB hardware keyloggers. In: Pinterest [online]. San Francisco (CA): Pinterest, 2010 [cit. 2023-05-20]. Dostupné z: https://cz.pinterest.com/pin/797066834030809822/

[17] Intel® Core™ i9-12900K Processor. Intel [online]. Mountain View, Kalifornie: Intel corporation, 1968 [cit. 2023-05-20]. Dostupné z: https://www.intel.com/content/www/us/en/products/sku/134599/intel-core-i912900k-processor-30m-cache-up-to-5-20-ghz/specifications.html

[18] AMD Ryzen™ 9 5950X Desktop Processors. *AMD* [online]. Santa Clara, Kalifornie: Advanced Micro Devices, Incorporated, 1969 [cit. 2023-05-20]. Dostupné z: https://www.amd.com/en/products/cpu/amd-ryzen-9-5950x

[19] Central Processing Unit (CPU). In: Khan Academy [online]. Mountain View, Kalifornie: Khan Academy, 2008 [cit. 2023-05-20]. Dostupné z: https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:computers/xcae6f4a7ff015e7d:computer-components/a/central-processing-unit-cpu

[20] What Is a GPU?. In: Intel [online]. Mountain View, Kalifornie: Intel corporation, 1968 [cit. 2023-05-20]. Dostupné z: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html

[21] Nvidia RTX 4090. Nvidia [online]. Santa Clara, Kalifornie: NVIDIA corporation, 1993 [cit. 2023-05-20]. Dostupné z: https://www.nvidia.com/cs-cz/geforce/graphics-cards/40-series/rtx-4090/

[22] Is it Possible to Use FPGA Hashcat to Decrypt Passwords. In: Ebics [online]. Exhibition Bay South Squre, Fuhai Bao'an Shenzhen China: Ebics, -, 2022 [cit. 2023-05-20]. Dostupné z: https://ebics.net/fpga-hashcat/

[23] Bcrypt password cracking extremely slow? Not if you are using hundreds of FPGAs!. In: Medium [online]. San Francisco (CA): A Medium Corporation, 2012, 2020 [cit. 2023-05-20]. Dostupné z: https://scatteredsecrets.medium.com/bcrypt-password-cracking-extremely-slow-not-if-you-are-using-hundreds-of-fpgas-7ae42e3272f6

[24] Revealer Keylogger [online]. Francie: Logixoft, 2008 [cit. 2023-05-20]. Dostupné z: https://www.logixoft.com/en-eu/index

[25] Keydemon [online]. Polsko, Tychy: Electroware™, 2021 [cit. 2023-05-20]. Dostupné z: https://www.keydemon.com/en/

[26] Rouse, Margaret. GPU-Accelerated Computing. In: Techopedia [online]. London, UK: Techopedia – Finixio, -, 2022 [cit. 2023-05-20]. Dostupné z: https://www.techopedia.com/definition/32876/gpu-accelerated-computing

[27] Intel® Core™ i7-8700K Processor. Intel [online]. Mountain View, Kalifornie: Intel corporation, 1968 [cit. 2023-05-20]. Dostupné z: https://www.intel.com/content/www/us/en/products/sku/126684/intel-core-i78700k-processor-12m-cache-up-to-4-70-ghz/specifications.html

[28] Nvidia GTX 1080. Nvidia [online]. Santa Clara, Kalifornie: NVIDIA corporation, 1993 [cit. 2023-05-20]. Dostupné z: https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080/specifications/

[29] Intel® Core™ i5-8300H Processor. Intel [online]. Mountain View, Kalifornie: Intel corporation, 1968 [cit. 2023-05-20]. Dostupné z: https://www.intel.com/content/www/us/en/products/sku/134876/intel-core-i58300h-processor-8m-cache-up-to-4-00-ghz/specifications.html

[30] Nvidia GTX 1050Ti. Nvidia [online]. Santa Clara, Kalifornie: NVIDIA corporation, 1993 [cit. 2023-05-20]. Dostupné z: https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1050-ti/specifications/

[31] Raspberry Pi 4 B. Raspberry Pi [online]. Cambridge, UK: Raspberry Pi Foundation, 2012 [cit. 2023-05-20]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[32] Broadcom VideoCore VI. Cpu monkey [online]. Henstedt-Ulzburg, Německo: -, 2015 [cit. 2023-05-20]. Dostupné z: https://www.cpu-monkey.com/en/igpu-broadcom_videocore_vi-221

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PIN Personal identification number.

Hash Pseudonáhodný řetězec vytvořený z hesla.

CPU Centrální procesorová jednotka.

GPU Grafická procesorová jednotka.

ALU Aritmeticko-logická jednotka.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: TABULKA TESTŮ HESEL

Skládaný stolní počítač												
	Hashovací rychlost		GPU akcelerace		Iterace		ALU		Vektorové jednotky		Doba k prolomení	
Hashovací funkce	Heslo 1	Heslo 2	Heslo 1	Heslo 2	Heslo 1	Heslo 2	Heslo 1	Heslo 2	Heslo 1	Heslo 2	Heslo 1	Heslo 2
MD5	10 398 MH/s	10 570 MH/s	256	256	1024	1024	256	256	8	8	5s	2d 5h
SHA1	5 599,4 MH/s	5 994,6 MH/s	256	256	1024	1024	256	256	1	1	10s	3d 22h
NTLM	12 825 MH/s	13 357 MH/s	256	256	1024	1024	256	256	1	1	5s	1d 17h
Bcrypt	674 H/s	677 H/s	2	8	512	128	11	11	1	1	10y 215d	>10y
Herní notebook												
MD5	1 708,2 MH/s	2 624,1 MH/s	256	256	1024	1024	256	256	1	1	11s	8d 6h
SHA1	1 750,1 MH/s	1 475,8 MH/s	256	256	1024	1024	256	256	1	1	22s	14d 21h
NTLM	3 708,5 MH/s	3 282,8 MH/s	256	256	1024	1024	256	256	1	1	20s	6d 14h
Bcrypt	156 H/s	156 H/s	4	2	256	512	11	11	1	1	41y 31d	>10y
Raspberry Pi 4 B												
MD5	10 800,7 kH/s	11 132,4 kH/s	256	256	1024	1024	1	1	4	4	5h 30m	5y 265d
SHA1	8 045,4 kH/s	8 047,1 kH/s	256	256	1024	1024	1	1	4	4	7h 36m	7y 336d
NTLM	13 829,3 kH/s	13 825,8 kH/s	256	256	1024	1024	1	1	4	4	4h 22m	4y 223d
Bcrypt	35 H/s	35 H/s	4	4	1024	512	1	1	1	1	62y 349d	4 294 967 228y 273d