VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

VPLYV SIEŤOVEJ INFRAŠTRUKTÚRY NA DISTRIBUOVANÉ LÁMANIE HESIEL

INFLUENCE OF NETWORK INFRASTRUCTURE ON DISTRIBUTED PASSWORD CRACKING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL EISNER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK HRANICKÝ,**
SUPERVISOR

BRNO 2019

Abstrakt

Lámanie hesiel je proces, ktorý sa používa k nájdeniu správneho kľúča, pomocou ktorého získame prístup k zabezpečenému obsahu. Tento proces zvyčajne funguje na princípe opakovaného skúšania možností a ich overovania pomocou výpočtu kryptografických algoritmov, ktorých náročnosť ovplyvňuje čas strávený výpočtami. Navzdory rôznym metódam akcelerácie je často nutné daný problém distribuovať medzi viacero uzlov, ktoré sú prepojené v lokálnej sieti alebo internetom. Cieľom práce je práve analyzovať vplyv sieťovej infraštruktúry na rýchlosť, škálovateľnosť a vyťaženie siete pri rôznych útokoch na kryptografické heše. Pre tieto účely je vytvorené automatizované experimentálne prostredie pozostávajúce z rôznych topológii, pomocných skriptov a sady testovacích úloh. Na základe analýzy výsledkov získaných pri použití nástrojov Fitcrack a Hashtopolis bolo možné tento vplyv odpozorovať.

Abstract

Password cracking is a process used to obtain the cracking key through which we get access to encrypted data. This process normally works on the principle of the repeated try of attempts and their verification by making calculations of cryptographic algorithms. The difficulty of algorithms affects the time spent on solving of the calculations. In spite of various acceleration methods, it is often necessary to distribute the given problem among several nodes which are interconnected via the local network or the internet. The aim of this thesis is to analyze the influence of network infrastructure on the speed, the scalability, and the utilization during different attacks on cryptographical hashes. For these purposes, there was created an automatized experimental environment, which consists of distinctive topologies, scripts, and sets of testing tasks. Based on the results of the analysis, which were obtained by the usage of tools Fitcrack and Hashtopolis it was possible to observe this influence.

Kľúčové slová

Lámanie hesiel, Hešovanie, Entropia hesla, Útok hrubou silou, Slovníkový útok, Dúhové tabuľky, Kombinačný útok, Hybridný útok, MPI, BOINC, Fitcrack, Hashtopolis, Efektivita, Škálovateľnosť, Lámania heiel, Vytaženie linky

Keywords

Password cracking, Hashing, Password entropy, Brute–force attack, Dictionary attack, Rainbow tables, Combination attack, Hybrid attack MPI, BOINC, Fitcrack, Hashtopolis, Efficiency, Scalability, Password cracking, Network traffic

Citácia

EISNER, Michal. *Vplyv sieťovej infraštruktúry na distribuované lámanie hesiel*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický,

Vplyv sieťovej infraštruktúry na distribuované lámanie hesiel

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Radka Hranického. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Eisner
22. mája 2019

Poďakovanie

Chcel by som sa poďakovať Ing. Radkovi Hranickému za odborné vedenie, veľké množstvo trpezlivosit a rady, ktoré mi pri tvorbe tejto práce pomohli. Veľká vďaka patrí Koroline Pukalíkovej a Dominikovi Drdákovi, ktorí si našli čas a prečítali si moju prácu pred odovzdaním. Nakoniec by som sa rád poďakoval svojej rodine, ktorá ma počas štúdia a tvorby tejto práce podporovala.

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Eisner Michal**
Program: Informační technologie
Název: **Vliv síťové infrastruktury na distribuované lámání hesel**
Influence of Network Infrastructure on Distributed Password Cracking
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s existujícími řešeními pro distribuované lámání hesel (Fitcrack, Hashstopolis, ...). Zaměřte se na implementaci jednotlivých typů útoků a způsob komunikace mezi výpočetními uzly.
2. Pro účely experimentů navrhněte několik modelových výpočetních sítí s různou architekturou a rychlostí linek.
3. Navrhněte sadu experimentů pro zjištění rychlosti a škálovatelnosti distribuovaného lámání hesel v těchto sítích. Sběr měřených dat bude automatizován pomocí vlastních skriptů/aplikací.
4. Experimenty realizujte, zhodnoťte výsledky a identifikujte kritická místa z hlediska síťové komunikace.

Literatura:

- D. P. Anderson, "BOINC: a system for public-resource computing and storage," Fifth IEEE/ACM International Workshop on Grid Computing, 2004, s. 4-10.
- Radek Hranický, Martin Holkovič, Petr Matoušek, and Ondřej Ryšavý. "On Efficiency of Distributed Password Recovery". In: The Journal of Digital Forensics, Security and Law 11.2 (2016), s. 79-96. ISSN: 1558-7215.
- a další dle dohody s vedoucím...

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Hranický Radek, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 30. října 2018

Obsah

Kapitola 1

Úvod

V dnešnej dobe plnej moderných technológií je najčastejšou formou zabezpečenia šifrovanie. Šifrované dáta sú zabezpečené pomocou šifrovacieho kľúča [9], ktorý je odvodený z hesiel. Heslá sa používajú pre prístup do rôznych aplikácií, pamäťových médií, dokumentov, ktoré obsahujú citlivé informácie. V niektorých prípadoch je potrebné sa k týmto informáciám dostať aj bez znalosti tohoto hesla. K tomuto účelu slúži proces lámania hesiel. Tento proces sa využíva najmä v oblasti forenznej analýzy digitálnych dát, ale uplatnenie nájde aj pri lámaní zabudnutého hesla a môže byť ovplyvnený rôznymi faktormi ako sú sila hesla, spôsob jeho zabezpečenia a použitý spôsob lámania.

Sila hesla závisí od dĺžky a kombinácie znakov, ktoré boli použité pri vytvorení hľadaného hesla. Ďalej môže byť heslo zabezpečené pomocou rôznych algoritmov s určitými vlastnosťami a tieto algoritmy môžu proces lámania hesiel skomplikovať a predĺžiť. Zavážiť môže spôsob akým sa hľadané heslo snažíme prelomiť. Nejedná sa len o spôsob, ktorý je použitý k lámaniu hesiel ale aj o to, či je tento proces vykonaný na jednom alebo viacerých výpočtových uzloch. V rámci jedného výpočtového uzla je tento proces ovplyvnený výpočtovými zdrojmi, ktoré nám dokáže poskytnúť. Množstvo týchto zdrojov je obmedzené a preto je dobré, ak proces lámania hesiel distribuujeme medzi viacero výpočtových uzlov. Spolu s distribúciou výpočtov vzniká ďalší faktor ovplyvňujúci lámanie hesiel. Týmto faktorom je prepojenie výpočtových uzlov. Prepojenie môže byť realizované v rámci lokálnej siete alebo internetom. Na druhú stranu môžu vzniknúť ďalšie faktory, ovplyvňujú distribuované lámanie hesiel. Konkrétne rýchlosť a stabilita pripojenia sú faktory, ktoré môžu veľmi výrazne predĺžiť alebo prerušiť proces distribuovaného lámania hesiel.

Cieľom tejto práce je zamerať sa na analýzu vplyvu sieťovej infraštruktúry z pohľadu rýchlosti a škálovatelnosti rôznych útokov na kryptografické heše. K tomuto úkonu bolo vytvorené automatizované experimentálne prostredie využívajúce niekoľko rôznych modelových sieti. V tomto prostredí bude vykonané distribuované lámanie hesiel pomocou dvoch konkrétnych nástrojov, Fitcrack a Hashtopolis. Zásluhou nameraných hodnôt jednotlivých hodnotiacich kritérií z vykonaných experimentov bude možné zistiť, aký vplyv má sieťová infraštruktúra na distribuované lámanie hesiel.

2. kapitola tejto práce sa venuje téme lámania hesiel. Na začiatku tejto kapitoly sa budem venovať sile hesla, princípom hešovania a spôsobom lámania hesiel. 3. kapitola sa venuje technológiám, ktoré umožňujú distribúciu výpočtov a nástrojom pre distribuované lámanie hesiel. 4. kapitola sa venuje implementácií skriptov pre uľahčenie práce s použitými nástrojmi a návrhom sady experimentov spolu so sieťovými topológiami. 5. kapitola sa venuje prezentácii a zhodnoteniu výsledkov získaných v rámci vykonávaných experimentov.

Kapitola 2

Lámanie hesiel

Každá aplikácia, úložisko alebo súbor, s ktorým je potrebné pracovať, môže požadovať istý druh zabezpečenia alebo overenia, že daná osoba má práva používať dané prostriedky, ktoré aplikácia, úložisko alebo súbor poskytuje. Predstavme si situáciu, v ktorej je súbor, aplikácia alebo úložisko, na ktorom sú uložené dáta a sú chránené heslom. V prípade, že toto heslo poznáme, môžeme ho veľmi jednoducho zadať a dáta uložené na daných úložiskách sú nám k dispozícii. Na druhú stranu pokiaľ heslo nepoznáme, je nutné dané heslo obnoviť a extrahovať dané dáta. Obnova alebo lámanie hesiel je významnou a potrebnou časťou digitálnej forenznej analýzy, ktorá skúma digitálny zločin [4]. V tejto kapitole sa budem zaoberať práve lámaním hesiel. Predstavím, ako je lámanie hesiel spojené s digitálnou analýzou. Vysvetlím pojmy, ako sú sila hesla, ktorá je veľmi dôležitá z pohľadu zabezpečenia a obnovy hesla, princíp zabezpečenia hesiel pomocou hešovania a rôzne spôsoby lámaní hesiel.

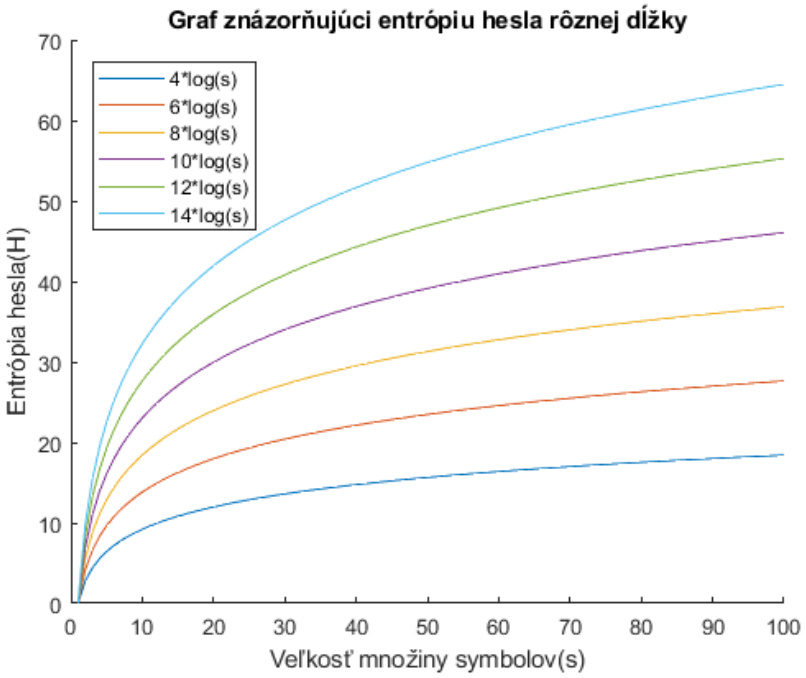
2.1 Sila hesla

Všetci vieme, ako funguje vytváranie hesiel. Každý z nás má určitý spôsob, ako vytvoriť heslá. Niekto si heslo nechá náhodne vygenerovať a naučí sa ho, ďalší si dané heslo vytvorí na základe informácií, vďaka ktorým si toto heslo ľahšie zapamätá. Či už pomocou využitia rôznych známych slov, čísiel, ich kombinácie alebo použitia veľkých a malých písmen. Máme veľa možností, ako si dané heslo vytvoriť. Pri vytváraní našich hesiel sa vždy musíme snažiť o to, aby bolo čo najsilnejšie alebo najťažšie prelomiteľné. V súvislosti so silou hesla úzko súvisí pojem entropia a entropia hesla. Entropia definuje počet stavov, v ktorých sa daný systém môže nachádzať [20]. Entropia hesla (H) je závislá na veľkosti množine symbolov (s) (e.g. počet znakov v danej abecede) a dĺžke hesla (n) [15]. Táto entropia môže byť vyčíslená pomocou vzorca 2.1:

$$H = n \times \log_2 s \tag{2.1}$$

Predstavme si, že by sme si vytvorili heslo, ktorého znaky sú z množiny znakov malej abecedy (26 znakov). Toto heslo by bolo o dĺžke 6 znakov(e.g. abcdef). Entropia tohoto hesla by bola hodnota rovná 19,55. Pre porovnanie predpokladajme, že heslo by mohlo obsahovať aj znaky z veľkej abecedy (26 $\times$ 2 znakov). Entropia tohoto hesla by bola rovná 23.707. V podstate to znamená, že čím väčšia entropia hesla je, tým je heslo silnejšie. V prípade kedy by sme použili aj čísla, počet stavov by sa zvýšil z 52 na 62. Entropia tohoto hesla by bola 24,763. Keďže entropia hesla je definovaná pomocou logaritmickej funkcie, ktorá má

svoj priebeh kde hodnoty logaritmu na začiatku rýchlo narastajú ale po čase sa zväčšujú veľmi pomaly, nemôžeme sa spoliehať na to, že veľkosť množiny symbolov, z ktorých heslo môžeme vytvoriť je dostačujúci parameter pre silu hesla. Podstatná je aj jeho dĺžka. Preto by sme mali voliť a vytvárať dlhšie heslá, ktoré obsahujú rôzne kombinácie malých a veľkých písmen, čísiel a pokiaľ je to možné mali by sme použiť aj rôzne špeciálne znaky. Znázornenie entropie hesiel rôznej dĺžky a rôznej veľkosti množiny symbolov je ukázané na obrázku 2.1. Na tomto obrázku môžeme vidieť priebeh logaritmických funkcií, ktoré popisujú entropiu hesiel (H) s danou dĺžkou (4, 6, 8, 10, 12, 14) a množinou symbolov(s).



Obr. 2.1: Priebehy logaritmických funkcií popisujúce entropiu hesiel.

2.2 Šifrovanie

Šifrovanie [23] je proces, ktorý prevádza dáta do nečitateľnej podoby kvôli zvýšeniu zabezpečenia. Dáta sú zašifrované pomocou šifrovacieho kľúča [9], ktorý je zvyčajne odvodený od hesla, ktoré zadal užívateľ.

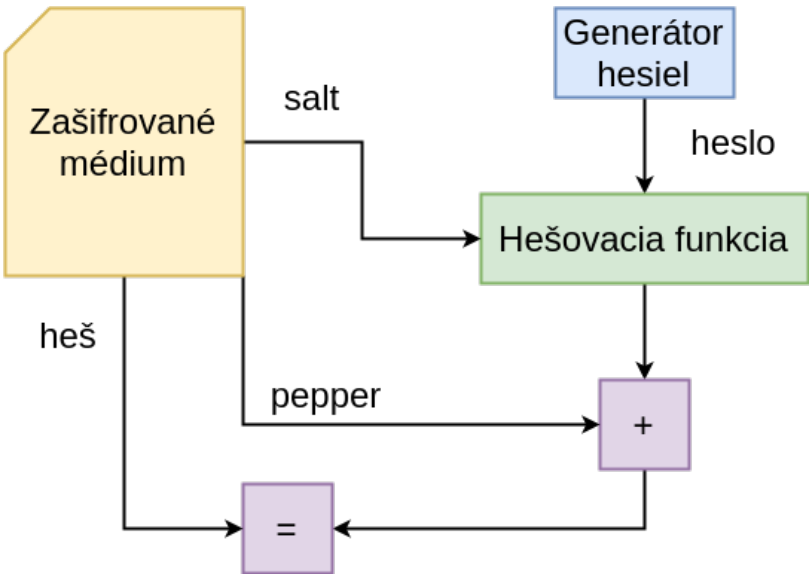
2.3 Hešovania hesiel

Keďže si nevytvárame nové heslo vždy, keď chceme použiť dané úložisko, aplikáciu alebo súbor, je nutné toto heslo niekde uchovávať. Heslá bývajú uchovávané na rôznych miestach. V tabuľkách databáz, súboroch alebo skryté v systémoch. Z hľadiska bezpečnosti by uložené heslo nemalo byť uložené v čitateľnej podobe [16]. Presne k tomuto účelu sa využíva hešovanie.

Hešovanie je proces, pri ktorom sa používa hešovacia funkcia [20]. Príkladom týchto funkcií sú SHA1 [12], MD5 [19], Whirlpool [5], atď. Hešovacia funkcia spracováva neob-

medzene dlhé vstupné dáta, na výstupný kód s predom obmedzenou dĺžkou. Výstupným dátam hešovacej funkcie hovoríme heš.

Proces hešovania je možné zabezpečiť pridaním 2 hodnôt. Tieto hodnoty nazývame salt a pepper. Salt je náhodná hodnota uložená v súbore, ktorá je pridaná k heslu predtým, než je heslo prevedené hešovacou funkciou [11]. Potom, ako je z hesla s pridanou salt vytvorený heš, sa k tomuto hešu pridáva pepper. Pridaním týchto dvoch hodnôt sa zvýšila bezpečnost aj obtiažnosť prelomenia hesiel, dokonca vďaka pridaniu týchto hodnôt, sú niektoré útoky nepoužiteľné. Príkladom takéhoto útoku je útok pomocou dúhových tabuliek [11]. Proces hešovanie je popísaný na obrázku 2.2.



Obr. 2.2: Proces hešovania so salt a pepper [11].

Ako je ale možné, že sa prihlasujem podľa hesla a nie podľa hešu? Je to z toho dôvodu, že pri kontrole správnosti hesla sa zoberie zadané heslo a pomocou hešovacej funkcie sa vygeneruje príslušný heš. V prípadoch, kedy sa používa salt a pepper, sa k danému heslu pridá salt, heslo so salt sa prevedie pomocou hešovacej funkcie na heš, potom sa k tomuto hešu pridá pepper a následne sa to porovná s uloženým hešom [16]. Pokiaľ sú heše zhodné, zadali sme správne heslo. V opačnom prípade sme zadali zlé heslo.

2.4 Lámanie

V rámci lámania hesiel môžeme naraziť na dve rôzne úlohy. Týmito úlohami sú:

- lámanie hešov,
- lámanie zašifrovaných médií.

2.4.1 Lámanie hešov

Heše sa používajú k zabezpečeniu uložených hesiel vo webových aplikáciach, operačných systémoch, atď. [11] Tieto heše sú vytvorené pomocou konkrétnej hešovacej funkcie. Aby sme mohli prelomiť daný heš, musíme poznať použitú hešovaciu funkciu. Lámanie potom

prebieha tak, že si vygenerujeme množinu testovacích hesiel, z ktorých si vypočítame testovací heš a následne tento heš porovnáme s hľadaným hešom.

2.4.2 Lámanie zašifrovaných médií

Za zašifrované média považujeme dokumenty (PDF, Office, atď.), archívy (ZIP, 7zip, RAR, atď.) a ďalšie média, ako napríklad diskové oddiely [11]. Obnova hesiel zo zašifrovaných médií závisí od typu, formátu a algoritmu, ktorý je definovaný výrobcom. K lámaniu potrebujeme získať verifikačnú hodnotu, ktorá predstavuje heš. Väčšina dokumentov a archívov ukladá verifikačnú hodnotu do metadát. Lámanie týchto médií prebieha tak, že sa vygeneruje množina testovacích hesiel. Z tejto množiny sa vyberie heslo, ktoré sa prevedie pomocou jednej alebo viacerých hešovacích algoritmov. Počet týchto algoritmov je dostatočne veľký aby nespomaľoval zobrazovanie obsahu dokumentov a zároveň aby skomplikoval proces lámania.

2.4.3 GPGPU

Proces lámania hesiel je často prevedený pomocou procesora alebo grafických kariet výpočtových uzlov. Grafické karty nám sprostredkovávajú obrovský výkon [14], ktorý môže byť využitý. Kvôli tomu sa pri procese lámaní hesiel používajú princípy založené na *General-purpose computing on graphics processing units* (GPGPU) alebo využívajú iné možnosti hardverovej akcelerácie [9]. GPGPU využíva paralelizáciu procesorov na grafickej karte k prevedeniu výpočtov rôznych algoritmov.

2.5 Spôsoby lámania hesiel

Pri lámaní hesiel nám ide o to, aby sme z hešu zistili pôvodné heslo. To samozrejme nebudeme robiť tak, že budeme náhodne skúšať kombinácie znakov, ale použijeme rôzne spôsoby a prístupy, ktoré nám tento proces umožnia konať systematickejším a niekedy aj rýchlejším spôsobom.

V následujúcej časti predstavím rôzne spôsoby lámania hesiel. Aby sme mohli pomocou týchto spôsobov prelomiť heslo, musíme poznať heš hľadaného hesla a mať množinu testovacích hesiel. Podľa spôsobu lámania, ktorým sme sa rozhodli lámať daný heš, získavame testovacie heslá. Vybrané testovacie heslo, ktoré sa má porovnávať, je zahešované rovnakou hešovacou funkciou, ako hľadaný heš. V prípade zhody sme našli hľadané heslo. V opačnom prípade je heslo, ktoré sme porovnávali, nesprávne. Vtedy vyberieme ďalšie heslo a porovnávame dovtedy, kým nenájdeme to správne heslo alebo neprejdeme celú množinu testovacích hesiel. Spôsoby, ktorými môžeme pristupovať k lámaniu hesiel sú [10]:

- Útok hrubou silou
- Slovníkový útok
- Útok pomocou dúhových tabuliek
- Kombinačný útok
- Hybridný útok (slovník a maska, maska a slovník)

2.5.1 Útok hrubou silou

Predstavme si situáciu, v ktorej vieme o danom hesle jeho dĺžku a množinu znakov použitých, na vytvorenie hesla. Povedzme, že dĺžka daného hesla je 6 znakov a na každom mieste môže byť jeden z 52 znakov abecedy (veľké a malé písmená). Z týchto informácií sme schopní vygenerovať všetky kombinácie 6 znakových slov, ktoré existujú. Počet týchto slov je 52^6, čo predstavuje 19 770 609 664 rôznych slov. Jedno z týchto slov je naše hľadané heslo. Na tomto princípe funguje útok hrubou silou.

Pri útoku hrubou silou sa pre generovanie testovacích hesiel najčastejšie používa maska [10]. Táto maska predstavuje zápis dĺžky hesla a množín znakov, z ktorých jednotlivé znaky daného hesla pozostávajú. Dĺžka masky je obmedzená. Podľa nástroja Elcomsoft[1] je maximálna dĺžka masky, ktorú je možné po vygenerovaní hesiel prelomiť v rozumnom čase 14 znakov[2]. Príklady symbolov masky pre nástroj Hashcat sú v tabuľke 2.1.

Symbol masky	Množina znakov	Príklad
?l	malé písmená, a-z	?l?l?l: aaa, ...
?u	veľké písmená, A-Z	?u?u?u?u: AAAA, ...
?d	arabské číslice, 0-9	?d?d?d: 000, ...
?s	špecialne znaky ASCII	?s?s?s: (!?., ...
?a	všetky možnosti	?a?a?a: aA1, V!5, ...
?h	malé hexa znaky, 0-9, a-f	?h?h?h: 0af, ...
?H	veľké hexa znaky, 0-9, A-F	?H?H?H: 0AF, ...
?b	binárne znaky, 0x00-0xFF	?b?b?b: 0x000x000x00, ...

Tabuľka 2.1: Prekladová tabuľka symbolov masky na množinu znakov [10].

Pri tomto útoku sa množina testovacích hesiel vygeneruje pomocou príslušnej masky. Z tejto množiny sa vyberie heslo, ktoré sa prevedie na heš pomocou hešovacej funkcie, ktorou bol vytvorený hľadaný heš. Ten sa porovná s hešom, ktorý vrátila hešovacia funkcia. Pokiaľ dôjde k zhode, našli sme hľadané heslo. V opačnom prípade sa z testovacej množiny vyberie ďalšie heslo a proces sa opakuje dokiaľ nenájdeme zhodu alebo neporovnáme všetky heslá z množiny testovacích hesiel. Príklad testovacej množiny hesiel je znázornený na obrázku 2.3.

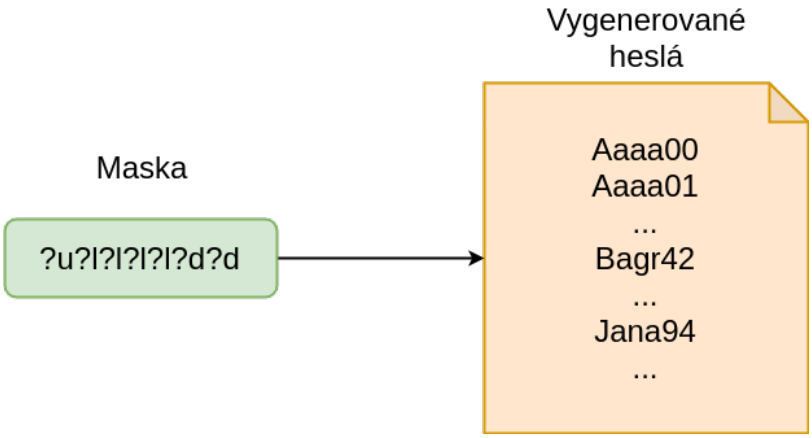
Pokiaľ pomocou masky vygenerujeme množinu testovacích hesiel, do ktorej patrí hľadané heslo, pri útoku hrubou silou sme schopní vždy nájsť riešenie. Problémom tohoto riešenia je čas strávený generovaním a porovnávaním. V prípade dlhších masiek po prípade kombinácií viacerých masiek by čas strávený generovaním a porovnávaním mohol byť nedosiahnuteľný. Z tohoto dôvodu je tento útok nevýhodný a je vhodnejšie použiť iný zo spôsobov lámania hesiel. Príklad veľkosti testovacej množiny hesiel rôznej dĺžky vytvorených z veľkých písmen (26 znakov), malých písmen (26 znakov) a číslic (10 znakov) je uvedený v tabuľke 2.2.

2.5.2 Slovníkový útok

Slovníkový útok, narozdiel od útoku hrubou silou, nepoužíva k vygenerovaniu testovacích hesiel masku, ale používa slovník. Tento slovník reprezentuje zoznam testovacích hesiel. Štruktúra tohoto slovníka je taká, že na každom riadku je maximálne jedno heslo [10]. Veľkosť tohoto slovníku záleží na množstve hesiel, ktoré obsahuje. Slovník môže obsahovať

[1] https://www.elcomsoft.com/

[2] https://www.elcomsoft.com/help/en/ppa/brute-force_attack.html



Obr. 2.3: Príklad množiny testovacích hesiel pri útoku hrubou silou pomocou masky [10].

Dĺžka hesla	Veľkosť množiny testovacích hesiel
2	3844
4	14776336
8	$2.183401056 \times 10^{14}$
16	$4.767240171 \times 10^{28}$
32	$2.272657884 \times 10^{57}$

Tabuľka 2.2: Veľkosť množiny testovacích hesiel podľa dĺžky hesla.

rôzne typy hesiel. Môžu to byť heslá predom vygenerované pomocou konkrétnej masky alebo masiek. Ďalej môže obsahovať heslá z databáz uniknutých hesiel [20] alebo môže obsahovať zoznam najpoužívanejších slov v danom jazyku.

Pri prevádzaní slovníkového útoku sa postupne vyberajú heslá z daného slovníku. Vybraté heslo je prevedené na heš pomocou hešovacej funkcie, ktorou bol vytvorený hľadaný heš a následne ich porovnáme. Slová sa budú prevádzať na heše a porovnávať dovtedy, kým nenájdeme zhodu alebo neskontrolujeme všetky heslá daného slovníka.

Výhodou tohoto útoku je rozdelenie testovaných hesiel do jednotlivých slovníkov. Narozdiel od generovania hesiel pomocou masky, slovníky obsahujú zmysluplnejšie heslá. Samozrejme, pokiaľ sa nejedná o slovník, ktorý obsahuje heslá generované podľa masky. Na druhú stranu veľkou nevýhodou slovníkového útoku je veľkosť slovníka, ktorá rastie s každým heslom, ktoré v ňom je. V tabuľke 2.3 môžeme vidieť počet slov a veľkosť rôznych násobkov slovníka `rockyou.txt`.

Názov slovníka	Počet hesiel	Veľkosť slovníka
2xrockyou.txt	28 519 020	265 MB
4xrockyou.txt	57 038 040	530 MB
8xrockyou.txt	114 076 080	1.1 GB
16xrockyou.txt	228 152 160	2.2 GB
32xrockyou.txt	456 304 320	4.2 GB

Tabuľka 2.3: Veľkosti slovníkov rockyou.

2.5.3 Útok pomocou dúhových tabuliek

Zamyslime sa nad tým, koľko hesiel môže obsahovať slovník a nad operáciami spojenými s lámaním hesiel. V prípade, že by slovník obsahoval 50 000 hesiel a heslo by sa v danom slovníku nachádzalo na konci alebo nenachádzalo vôbec, je nutné vykonať 50 000 prevedení na heše a 50 000 porovnaní. Z pohľadu vykonávania jednotlivých operácií je prevod hesiel na heše časovo najviac náročnou operáciou [20]. Tento čas by sa dal ušetriť, ak by sme si dané heše predpočítali a toto je podstata útoku pomocou dúhových tabuliek. Porovnanie rýchlosti lámania[3] hešov môžeme vidieť v tabuľke 2.4.

Hešovací algoritmus	Počet hešov / Sekundu
MD5	24943.1 MH/s
SHA1	8538.1 MH/s
Whirlpool	253.9 MH/s
Bcrypt	13094 H/s

Tabuľka 2.4: Rýchlosť lámania hešov pomocou grafickej karty NVIDIA GTX 1080.

Dúhová tabuľka je tabuľka obsahujúca heslá a príslušné predpočítané heše [13]. Pri lámaní hesiel týmto spôsobom, sa len porovnávajú jednotlivé heše s hľadaným, vďaka tomu je proces lámania hesiel rýchlejší. Tento útok má výhody z pohľadu šetrenia času pri nahradení prepočítavania hesiel na heše, vyhľadávaním v dúhovej tabuľke, ktoré je rýchlejšie. Na druhú stranu, dúhové tabuľky sú veľmi veľké a rýchlo narastajú s každým ďalším heslom a sú nepoužiteľné, pokiaľ hľadaný heš obsahuje salt alebo pepper [10]. Príklad veľkosti[4] dúhových tabuliek je na obrázku 2.4.

Table ID	Charset	Plaintext Length	Key Space	Success Rate	Table Size
sha1_ascii-32-95#1-7	ascii-32-95	1 to 7	70,576,641,626,495	99.9 %	52 GB 64 GB
sha1_ascii-32-95#1-8	ascii-32-95	1 to 8	6,704,780,954,517,120	96.8 %	460 GB 576 GB
sha1_mixalpha-numeric#1-8	mixalpha-numeric	1 to 8	221,919,451,578,090	99.9 %	127 GB 160 GB
sha1_mixalpha-numeric#1-9	mixalpha-numeric	1 to 9	13,759,005,997,841,642	96.8 %	690 GB 864 GB
sha1_loweralpha-numeric#1-9	loweralpha-numeric	1 to 9	104,461,669,716,084	99.9 %	65 GB 80 GB
sha1_loweralpha-numeric#1-10	loweralpha-numeric	1 to 10	3,760,620,109,779,060	96.8 %	316 GB 396 GB

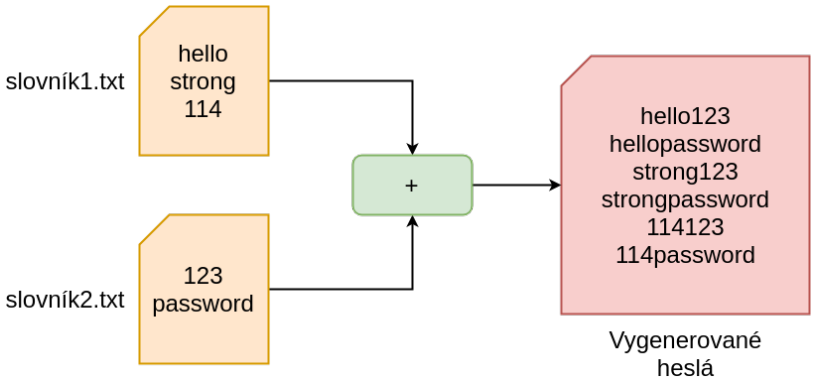
Obr. 2.4: Veľkosti dúhových tabuliek pre SHA1.

2.5.4 Kombinačný útok

Tento spôsob útoku je odvodený od slovníkového útoku. Hlavnou myšlienkou kombinačného útoku je, že máme dva slovníky, jeden pravý a druhý ľavý [10]. Tieto slovníky môžu

[3] https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40
[4] http://project-rainbowcrack.com/table.htm

obsahovať ľubovoľný počet hesiel. Nemusia byť rovnako veľké. Podstatou je, že sa z ľavého slovníku vyberie prvé heslo, toto heslo sa spojí s každým heslom v pravom slovníku a výsledné heslá sa uložia do tretieho slovníka, ktorý obsahuje množinu testovacích hesiel. Príklad vytvárania slovníku obsahujúci množinu testovacích hesiel je uvedený na obrázku 2.5. Následné lámanie hesiel pokračuje ako pri jednoduchom slovníkovom útoku. Vyberieme postupne jedno heslo za druhým, prevedieme na heš a porovnáme.



Obr. 2.5: Generovanie množiny testovacích hesiel pri kombinačnom útoku pomocou dvoch slovníkov [11].

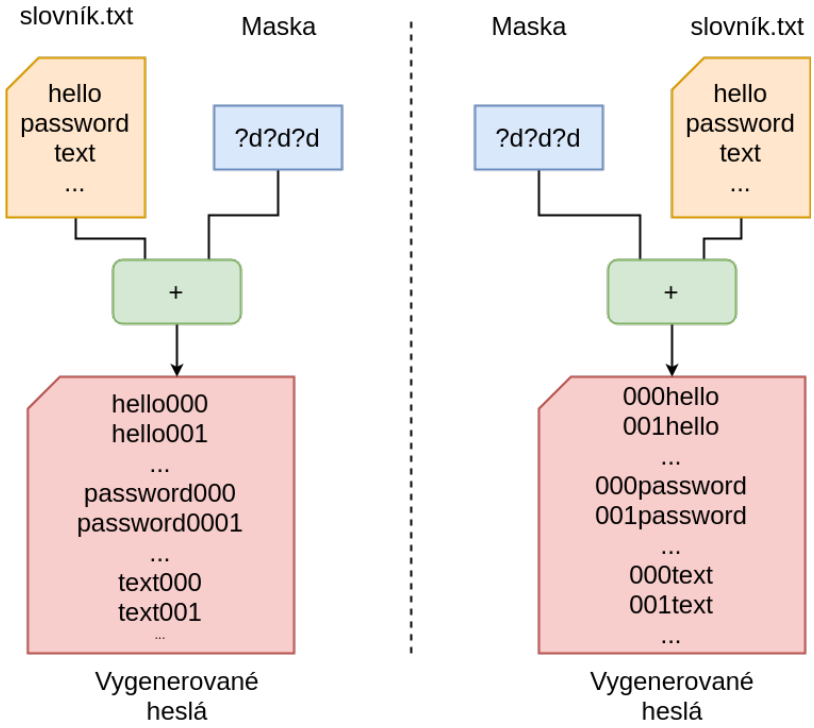
Tento útok je výhodný v tom, že pokiaľ správne kombinujeme, môžeme prísť na veľké množstvo nových hesiel. Predstavme si situáciu, v ktorej je ľavý slovník najpoužívanejších mien a pravý slovník najpoužívanejších čísiel. Kombináciou týchto dvoch slovníkov sme schopní vygenerovať heslá, ktoré predstavujú všetky kombinácie daných mien a čísiel. Pokiaľ by hľadané heslo malo tvar napríklad Peter123 a slovo Peter by bolo v ľavom slovníku a číslo 123 by bolo v pravom slovníku, potom by sme vo výslednom slovníku mali hľadané heslo. Nevýhodou tohoto typu útoku je veľkosť výsledného slovníka, ktorá prudko rastie. V prípade, že by sme mali ľavý slovník o veľkosti 10 000 hesiel a pravý slovník o veľkosti 20 000 hesiel, výsledný slovník by mal veľkosť 10000×20000 hesiel, čo predstavuje 2×10^8 hesiel.

2.5.5 Hybridný útok

Hybridný útok je typ kombinačného útoku, ktorý na vygenerovanie hesiel do slovníka obsahujúceho množinu vygenerovaných hesiel používa buď slovník a masku, alebo masku a slovník. Pokiaľ sa jedná o hybridný útok pomocou slovníka a masky, tak prvá časť hesla je generovaná pomocou slovníka a druhá časť pomocou masky [10]. V prípade hybridného útoku pomocou masky a slovníka je prvá časť generovaná pomocou masky a druhá pomocou slovníka. Nové vygenerované heslo sa uloží do tretieho výsledného slovníka, ktorý predstavuje množinu testovacích hesiel a lámanie môže pokračovať, ako jednoduchý slovníkový útok. Príklad vytvárania množiny testovacích hesiel pomocou hybridného útoku je uvedený na obrázku 2.6.

Hybridný útok má podobné výhody ako kombinačný útok. Vďaka tomuto útoku sme schopní vytvoriť veľké množstvo nových hesiel, ktoré môžeme následne testovať. Na druhú stranu si však musíme dobre zvoliť masku. Predstavme si situáciu, v ktorej použijeme generovanie pomocou slovníku a masky. Slovník obsahuje 50 000 hesiel. Masku si zvolíme o dĺžke 6 a bude tvorená len číselnými hodnotami (10 hodnôt). To znamená, že pomocou

tejto masky sme schopní vygenerovať 10^6 hesiel, čo predstavuje 1 000 000 hesiel. Keďže sa jedná o typ kombinačného útoku, výsledný slovník by obsahoval 5×10^{10} hesiel. V prípade, že by sme si zvolili príliš dlhú masku a väčší slovník, množina testovacích hesiel by bola veľmi veľká a proces generovania, spájania a ukladania do súboru by mohol proces lámania hesiel predlžovať.



Obr. 2.6: Generovanie množiny testovacích hesiel pri hybridnom útoku [11].

Kapitola 3

Distribúcia výpočtov

Na začiatku tejto kapitoly sa budem venovať možnostiam distribúcie výpočtov. Predstavím technológie *Message Passing Interface* (MPI), *Berkeley Open Infrastructure for Network Computing* (BOINC) a protokol nad HTTP. V ďalšej časti tejto kapitoly sa budem venovať dvom nástrojom pre distribuované lámanie hesiel, a to Fitcrack a Hashtopolis. Tieto dva nástroje budú v rámci tejto práce veľmi dôležité, pretože s ich pomocou budeme skúmať vplyv sieťovej infraštruktúry na distribuované lámanie hesiel. V závere tejto kapitoly predstavím hodnotiace kritéria a faktory ovplyvňujúce distribúciu výpočtov.

3.1 Technológie

Na počítačoch sme schopní riešiť danú úlohu pomocou *central processing unit* (CPU). Práca procesora nám mnohokrát ušetrí čas pri vykonávaní jednoduchších úloh. Ale nie je to jediný spôsob, ako riešiť nejakú úlohu a rozhodne nie je najrýchlejší. Ďalšou možnosťou, ako riešiť výpočtovo náročný problém, je využiť výpočtový výkon graphics processing unit (GPU). GPU pozostáva z niekoľkých procesorov, ktoré dokážu paralelne riešiť zadané úlohy. Týmto spôsobom sme schopní pomocou GPU riešiť náročnejšie problémy oveľa rýchlejšie, ako pomocou CPU. Navzdory tomu, že výkon GPU je vyšší ako CPU existujú problémy s tak vysokou výpočtovou náročnosťou, pri ktorých nemá zmysel riešiť tento problém na jednom stroji. Nejedná sa o to, že by to ten stroj nedokázal vyriešiť, ale toto riešenie by mohlo trvať príliš dlho. Z tohoto dôvodu vznikol nápad daný problém distribuovať.

Keď si predstavíme komplikovaný problém, správny postup jeho riešenia by mal byť taký, že daný problém rozdelíme na podproblémy, ktoré sa snažíme riešiť, a v tom spočíva distribúcia výpočtov. Máme niekoľko uzlov, ktorým môžeme zadať tieto podproblémy a tieto uzly sa ho pokúsia vyriešiť, a vrátia nám výsledky. Samozrejme nejde to len tak a pre distribuované výpočty budeme využívať technológie, ktoré nám tento proces uľahčujú. V následujúcej časti kapitoly uvediem tri technológie, ktoré umožňujú distribúciu výpočtov.

3.1.1 Message Passing Interface

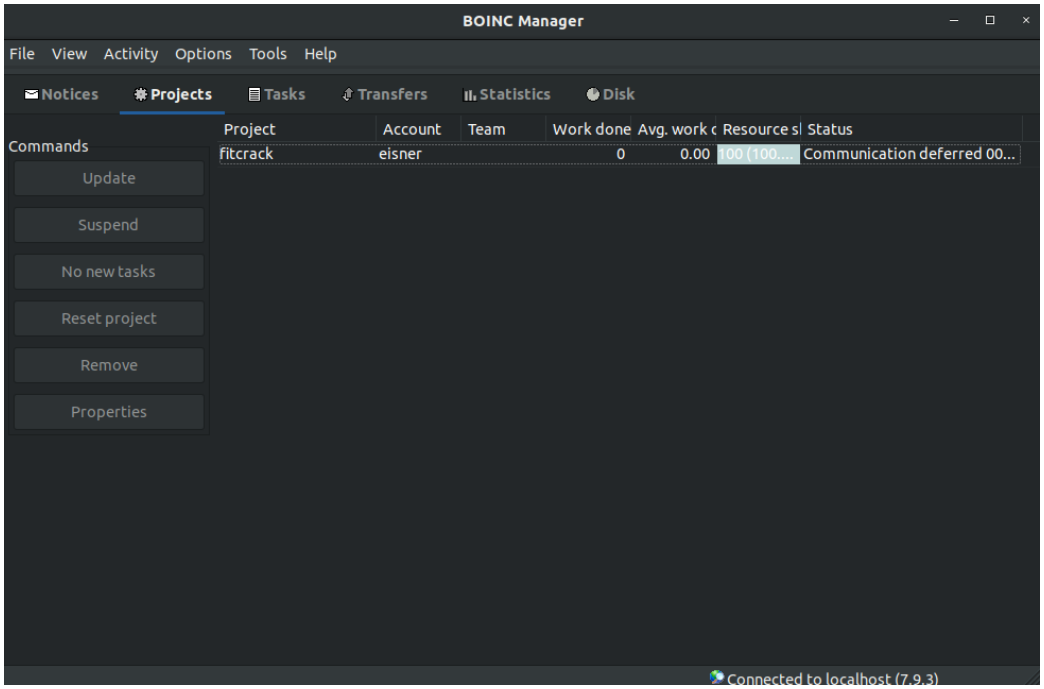
Message Passing Interface (MPI) [9] popisuje špecifikáciu protokolu a knižnice. MPI poskytuje efektívnu možnosť rozdelenia práce na podúlohy medzi viacero uzlov. Tento počet uzlov je statický, to znamená, že nemôžeme dynamicky zvyšovať počet uzlov. Každý z týchto uzlov spracováva podmnožinu vstupných dát. Tento spôsob distribúcie poskytuje výborný výkon pri prevedení výpočtovo náročných operácií nad menším objemom dát [9]. Súčasné implementácie MPI podporujú komunikáciu cez internet, ale neposkytujú žiadne prostriedky

pre vykonávanie výpočtov v nedôveryhodnom prostredí. MPI neposkytuje prostriedky pre šifrovanie a autentizáciu komunikácie. Toto zabezpečenie musí byť riešené pomocou iných aplikácií.

Medzi výhody patrí výkon, ktorý MPI poskytuje pri výpočtovo náročných operáciach nad menším počtom dát. Na druhú stranu má určité nevýhody. Medzi tieto nevýhody patrí hlavne počet uzlov, ktorý je statický, a žiadne prostriedky zabezpečenia a autentizácie.

3.1.2 BOINC

Berkeley Open Infrastructure for Network Computing (BOINC) [2] je platformou pre distribuované výpočty, ktorá podporuje dynamický počet uzlov pripojených v rámci lokálnej siete alebo aj cez internet [9]. Táto platforma funguje na princípe klient–server. Cieľom BOINC je zdieľanie výpočtových zdrojov a výpočtového výkonu v rôznych vedeckých oblastiach, ako sú meteorológia, medicína, matematika, fyzika a mnoho ďalších[1]. Preto, aby sme mohli poskytnúť výpočtové zdroje, niektorému z projekov. Je nutné, aby sme mali nainštalovaného BOINC Client alebo BOING Manager aplikáciu, ktorá nám poskytuje grafické uživateľské rozhranie 3.1.



Obr. 3.1: Uživateľské rozhranie aplikácie BOINC Manager.

[1] https://boinc.berkeley.edu/projects.php

V rámci zapojenia výpočtových zdrojov do distribuovaného výpočtu existujú dve kategórie. Kategórie definujú podmienky, ktoré musia uzly zapojené do distribuovaného výpočtu spĺňať. Tieto kategórie sú [2]:

- volunteer computing,
- grid computing.

Grid computing

Grid computing väčšinou používa výpočtové zdroje poskytnuté rôznymi inštitúciami, ako sú univerzity alebo firmy, ktoré sú ochotné zapojiť do výpočtov svoje stroje. Tieto stroje môžu byť bežné počítače, až superpočítače a mnoho ďalších [2]. Uzly zapojené do distribuovaných výpočtov sú zväčša spravované IT profesionálmi. Sú zapnuté počas celého dňa aj noci a pripojené k širokopásmovým sieťovým linkám.

Volunteer computing

Volunteer computing používa výpočtové zdroje poskytnuté bežnými uživateľmi, ktorí sú ochotní poskytnúť svoje výpočtové zdroje v rámci distribuovaného riešenia projektu, ktorý si vybrali. Pri volunteer computingu zapojené uzly nemusia spĺňať podmienky širokopásmového sieťového pripojenia, ani nemusia byť pripojení počas noci.

BOINC Client

Pokiaľ chceme poskytnúť výpočtový výkon a podporiť distribuované riešenie projektov, musíme mať na danom zariadení nainštalovanú BOINC Client alebo klienta s BOINC Manager aplikáciou (GUI pre klienta). Na danom klientovi sme schopní nastaviť za akých podmienok daný projekt môže využiť výpočtový výkon daného stroja, kedy môže tieto zdroje využiť, a na koľko môže dané zdroje využívať [11]. Klient udržuje komunikáciu so serverom. Stará sa o pripravenie výpočtového uzlu k prijatiu dát potrebných k zahájeniu výpočtov a následne o odoslanie výsledkov jednotlivých pridelených úloh.

BOINC Server

Server je zodpovedný za prideľovanie práce pripojeným klientom, ktorí sú pripojení a požiadali o pridelenie pracovnej úlohy [11]. Klient sa môže pripojiť a odpojiť z daného projektu v priebehu výpočtu a nespôsobí tým škodu ani stratu dát hlavne vďaka možnostiam zotavenia, ktoré technológia BOINC poskytuje. Potom, ako klient dokončí výpočty a pošle odpoveď, server túto odpoveď príjme a spracuje. Technológiu BOINC pre distribuované lámanie hesiel využíva nástroj Fitcrack.

3.1.3 Protokol nad HTTP

V rámci distribúcie výpočtov môže byť použitý aj vlastný alebo existujúci protokol nad protokolom HTTP. Výhodou je možnosť pripojenia výpočtových uzlov pomocou internetu aj zo sietí, ktoré môžu filtrovať komunikáciu podľa typu protokolu.

Príkladom nástroja využívajúci tento spôsob distribúcie je nástroj Hashtopolis[2]. Tento nástroj pri distribúcii výpočtov posiela dáta vo formte JSON pomocou protokolu HTTP.

[2] https://github.com/s3inlc/hashtopolis

3.2 Distribuované lámanie hesiel

Hlavnou motiváciou distribuovať proces lámania hesiel, je snaha čo najviac zvýšiť výpočtový výkon. V tejto kapitole predstavím konkrétne nástroje Fitcrack a Hashtopolis.

Hashcat

Hashcat[3] predstavuje nástroj pre obnovu hesiel, ktorý využíva technológiu OpenCL. OpenCl je štandart popisujúci paralélne programovanie v heterogénnych počítačových systémoch [21]. O rýchlosti nástroja hashcat svedčí to, že v rokoch 2010, 2012, 2014, získal prvé miesto v súťaži `Crack Me If you Can`[4]. Tento nástroj podporuje viac ako 200 typov kryptografických hešov. Medzi podporované útoky patria:

- útok hrubou silou,
- kombinačný útok,
- slovníkový útok,
- hybridný útok,
- a ďalšie[5].

3.3 Nástroj Fitcrack

Jedná sa o open–source nástroj umožňujúci distribuované lámanie hesiel vyvíjaný Fitcrack tímom na VUT FIT v Brne. Tento nástroj využíva technológiu BOINC pre distribúciu výpočtov po sieti a nástroj hashcat pre obnovu hesiel. Vďaka využitiu technológie BOINC umožňuje distribúciu výpočtov aj mimo lokálnej siete.

3.3.1 Klient

Klient nástroja Fitcrack je sprostredkovaný vďaka technolgii BOINC.Klient žiada server o pridelenie práce. Potom, ako dokončí pridelenú prácu pošle na server odpoveď. Vzhľadom k tomu, že nástroj Fitcrack používa technológiu BOINC pre distribúciu výpočtov, klienti komunikujú so serverom pomocou protokolu BOINC scheduling server protocol[6], ktorý je založený na RPC cez HTTP(S) [11]. Server prideľuje a zároveň kontroluje vykonanie pridelenej práce. Klient nie je závislý na sieti, v ktorej bol pridaný do projektov. Vďaka technológii BOINC sa môže zmeniť adresa klienta a stále bude možné vykonávať výpočty.

3.3.2 Server

Server nástroja Fitcrack pozostáva z viacerých častí:

- nástroj Hashcat,
- technológia BOINC,
- nástroj Fitcrack.

[3] https://hashcat.net/hashcat/
[4] http://contest-2010.korelogic.com/
[5] https://hashcat.net/wiki/doku.php?id=hashcat
[6] https://boinc.berkeley.edu/trac/wiki/RpcProtocol

Generátor

Generátor je serverový démon zodpovedný za vytváranie nových workunits pre hosts [11]. V rámci prideľovania workunits existujú dva typy. Tieto typy sú benchmark a normálna lámacia úloha. Benchmark je úloha, pomocou ktorej sú otestovaní klienti. Existuje úplný benchmark, ktorý je spustený len raz, keď je pridaný nový klient. Tento klient je otestovaný na všetky podporované hešovacie algoritmy. Dôvodom tohoto testu je zistenie rýchlosti lámania. Ďalší typ benchmark je spustený pred pridelením lámacej úlohy. Generátor má taktiež na starosti vysporiadať sa s odpojenými klientmi a výpočtovými chybami. V rámci prideľovania práce jednotlivým uzlom je veľkosť workunit prispôsobená na základe zadanej úlohy a náročnosti použitého hešovacieho algoritmu.

Validátor

Validátor je serverový démon, ktorý kontroluje syntax prichádzajúcich výsledkov predtým než sú spracované [11]. Vďaka tomu sú odhalené poškodené uzly, ktoré poskytujú nesprávne výsledky. Nástoj Fitcrack používa BOINC valídátor, ktorý túto syntax kontroluje.

Assimilátor

Assimilátor je serverový démon, ktorý spracováva výsledky, ktoré poskytli výpočtové uzly [11]. V rámci spracovania výsledkov existujú tri typy výsledkov. Jednoduchý benchmark, celkový benchmark a lámacia úloha.

Trickler

Trickler je serverový démon, ktorý má za úlohu udržovať periodickú komunikáciu medzi serverom a klientmi pri čom bude získavať informácie o priebehu výpočtov [11]. Vďaka tomu vieme aktuálny stav daného výpočtu.

Transitioner

Transitioner je serverový démon BOINC, ktorý má za úlohu udržať databázu v synchronizovanej podobe [11]. Všetci ostatní serveroví démoni sú na ňom závislí.

Scheduler

Scheduler je serverový démon zodpovedný za komunikáciu s uzlami [11]. V rámci tejto komunikácie sú zasielané request a response správy v XML formáte. Obsahom týchto správ sú všetky informácie a nová úloha, ktorá bola vytvorená Generátorom.

Feeder

Feeder je serverový démon prevzatý z technológie BOINC, ktorý pracuje so Scheduler a má za úlohu distribuovať časti zdieľanej pamäte [11].

File Deleter

File Deleter je jedným z BOINC serverových démonov, ktorý je zodpovedný za odstránenie súborov, vytvorených počas riešenia úloh [11].

WebAdmin

Nástroj Fitcrack poskytuje prehľadné webové rozhranie pomocou, ktorého môžeme vytvárať úlohy pre distribuovanú obnovu hesiel. Úloha, ktorá je identifikovaná pomocou mena, typu útoku a hešu sa nazýva Job.

Následne je nutné nástroju Fitcrack zadať vstupné dáta, s ktorými bude pracovať. Týmito vstupnými dátami sú typ hešovacieho algoritmu a spôsob načítania vstupu. Ďalej je potrebné špecifikovať spôsob lámania hesiel a dáta spojené s konkrétnym útokom.

Keďže sa jedná o distribuované lámanie hesiel, je potrebné ku konkrétnej vytvorenej úlohe priradiť výpočtové uzly, ktoré sa budú podieľať na distribuovanom riešení. Tieto výpočtové uzly sú v rámci nástroja Fitcrack pomenované ako hosts. Jeden výpočtový uzol môže byť priradený viacerým úloham. Každý výpočtový uzol má v nástroji Fitcrack o sebe uvedené informácie. Medzi tieto informácie patrí názov uzlu, jeho status a ďalšie.

Pri distribuovanom riešení problému je nutné tento problém rozdeliť na podproblémy. Tieto podproblémy sú následne pridelené jednotlivým uzlom. V rámci nástroja fitckrack sa tieto podproblémy nazývajú Workunit. Nástroj Fitcrack umožňuje nastavovať veľkosť riešených podproblémov. Táto veľkosť definuje rozsah hesiel z množiny testovacích hesiel pridelených uzlu.

3.3.3 Spustenie úlohy

Nástroj Fitcrack zahajuje riešenie úlohy keď:

- je vytvorená úloha,
- táto úloha je spustená,
- danej úlohe je priradený aspoň 1 host,
- tento host je autentizovaný voči serveru (má pridaný projekt, ktorý nie je pozastavený),
- host neporušuje nastavenia BOINC Client, ktoré predstavujú možnosti použitia.

3.3.4 Podporované útoky

Nástroj Fitcrack umožňuje distribuované lámanie hesiel pomocou následujúcich útokov:

- distribuovaný slovníkový útok,
- distribuovaný kombinačný útok,
- distribuovaný útok hrubou silou,
- distribuovaný hybridný útok.

Distribuovaný slovníkový útok

V predchádzajúcej kapitole som uviedol, že slovníkový útok využíva slovník hesiel, z ktorého vyberá testovacie heslá, prevádza ich na heše a následne porovnáva. Pri distribuovanom riešení tohoto útoku pomocou Fitcrack sa daný slovník rozdelí na časti a posiela sa iba časť, ktorú má daný uzol spracovať. Toto riešenie šetrí čas, ktorý by sme museli stráviť čakaním na stiahnutie celého slovníka.

Distribuovaný kombinačný útok

Tento spôsob útoku využíva dva slovníky, z ktorých generuje nové heslá spojením hesiel z prvého a druhého slovníku a uložením do výsledného slovníku. V nástroji Fitcrack je tento útok riešený tak, že na každý uzol zapojený do výpočtov sa pošle celý prvý slovník a časť druhého slovníku, ktorý má daný uzol spracovať. Vygenerujú sa všetky kombinácie hesiel z týchto dvoch slovníkov. Vygenerované heslá sa uložia do tretieho slovníku a začne proces lámania na daných uzloch.

Distribuovaný útok hrubou silou

Útok hrubou silou je pomocou Fitcracku realizovaný tak, že sa na výpočtové uzly distribuuje maska a rozsah indexov hesiel, ktoré má daný uzol spracovať. Tento rozsah je určený pomocou celkového počtu hesiel, ktorý je generovaný pomocou masky.

Distribuovaný hybridný útok

Tento útok využíva masku a slovník alebo slovník a masku. V oboch prípadoch je realizácia nástrojom Fitcrack rovnaká. Fitcrack pomocou masky vygeneruje všetky heslá. Tieto heslá uloží do slovníku. V tomto prípade máme kombinačný útok s 2 slovníkmi, ktorý sa rieši ako distribuovaný kombinačný útok. Na každý uzol sa pošle celý prvý slovník a časť druhého slovníku, ktoré má daný uzol riešiť.

3.4 Nástroj Hashtopolis

Hashtopolis[7] je nástroj, ktorý umožňuje zapojenie viacerých výpočtových uzlov do procesu distribuovaného lámania hesiel. Distribúcia výpočtov je zaistená pomocou protokolu nad HTTP. K lámaniu hesiel Hashtopolis používa nástroj hashcat.

3.4.1 Klient

Klient v nástroji Hashtopolis je implementovaný pomocou jazyka C# pre systém Windows a taktiež pomocou jazyka Python pre linuxové systémy. Pri pridaní nového klienta je v aplikácii nastavená url adresa servera a autentizačný token. Vďaka tomuto nezáleží na sieti, v ktorej sa klient nachádza. Podstatné je, aby sa nezmenila adresa servera. Klient žiada o pridelenie úlohy od servera. Pri riešení úloh pridelených serverom si klient ukladá dáta do vytvorených zložiek. Klient tieto zložky nevyprázdňuje automaticky po skončení úlohy, čo môže byť problém pri práci na stroji s obmedzenou kapacitou pamäte. Klient sa nazýva Ägent".

3.4.2 Server

Server prideľuje úlohy jednotlivým klientom. Veľkosť chunk je ovplyvnená na základe konkrétnej úlohy, ktorá je riešená, a náročnosti použitého hešovacieho algoritmu. Server je realizovaný pomocou jazyka PHP a komunikuje s MySQL databázou. Vyhľadávania v databáze je pomocou indexov pre zaručenie rýchleho vyhľadávania.

[7] https://github.com/s3inlc/hashtopolis

Webové rozhranie

Server nástroja Hashtopolis má implementované webové rozhranie. Toto rozhranie umožňuje pridať nový uzol nazývaný Agent, ktorému musí vygenerovať autentizačný token. Ďalej je možné vytvoriť úlohu označený ako Task. Predtým, než je vytvorený Task je potrebné aby heš, ktorý bude tento Task riešiť, bol uložený ako Hashlist. Následne pri vytváraní úlohy len priradíme tento Hashlist, tejto úlohe. Spôsob útoku je nutné definovať v políčku Comand pomocou Hashcat príkazov. Príklady príkazy pre vytvorenie útokov sú uvedené v tabuľke 3.1 .

Útok	Príkaz
Útok hrubou silou	-a3 #HL# ?a?a?a?a?a?a
Slovníkový útok	-a0 #HL# Wordlist.txt
Kombinačný útok	-a1 #HL# Wordlist.Left Wordlist.Right
Hybridný útok S–M	-a6 #HL# Wordlist.txt ?a?a?a?a?a?a
Hybridný útok M–S	-a7 #HL# ?a?a?a?a?a?a Wordlist.txt

Tabuľka 3.1: Príkazy pre vytvorenie konkrétneho útoku.

Pri vytváraní úlohy je možné definovať veľkosť úlohy, ktorá bude pridelená jednotlivým uzlom. Táto úloha sa nazýva "Chunk". Veľkosť úlohy je označená ako "Chunk size". Po vytvorení úlohy je potrebné tejto úlohe prideliť výpočtové uzly. V prípade, keď danej úlohe pridelíme uzol, ktorý je aktívny (má nastavené isActive na 1) tento uzol začne riešiť pridelenú úlohu.

3.4.3 Spustenie úlohy

Nástroj Hashtopolis spustí prácu, ak sú splnené následujúce podmienky:

- bol vytvorený Task s priradeným Hashlist a potrebnou konfiguráciou,
- na danú úlohu je namapovaný aspoň jeden aktívny Agent,
- na Agent je spustený Hashtopolis client,
- Hashtopolis client je autentizovaný tokenom a má pripojenie na server,
- v prípade práce s hashlistom, ktorý obsahuje dôverné dáta musí byť Agent trusted.

3.4.4 Podporované útoky

Nástroj Hashtopolis[8] umožňuje distribuované lámanie hesiel pomocou následujúcich útokov:

- distribuovaný slovníkový útok,
- distribuovaný útok hrubou silou,
- distribuovaný kombinačný útok,
- distribuovaný hybridný útok.

[8] https://github.com/s3inlc/hashtopolis/wiki/Task-Creation-Guidelines

Distribuovaný útok hrubou silou

Pri útoku hrubou silou sa na uzly zapojené do distribuovaného lámania hesiel posiela maska a počet indexov, ktoré má daný uzol spracovať.

Distribuovaný slovníkový útok

Pri distribuovanom slovníkovom útoku je slovník obsahujúci množinu testovacích hesiel posielaný na všetky výpočtové uzly naraz. Uzly si tento slovník stiahnu a potom je im pridelený chunk, ktorý riešia. Po vyriešení prideleného chunk mu server pridelený ďalší dokiaľ nie je úloha vyriešená.

3.5 Hodnotiace kritéria distribúcie výpočtov

V rámci distribúcie výpočtov je nutné definovať hodnotiace kritéria, pomocou ktorých budem môcť porovnávať vykonané experimenty. Tieto hodnoty budem označovať ako metriky. Metriky budú popisovať vlastnosti distribúcie a lámania hesiel. Získavanie týchto dát bude pomocou automatizovaných nástrojov, ktorých návrhy predstavím v poslednej kapitole.

3.5.1 Efektivita a réžia distribúcie výpočtov

Efektivita reprezentuje percento času, ktorý procesor strávil výpočtami mimo komunikácie a čakania [8]. Túto hodnotu je možné vyčísliť pomocou vzorca:

$$E_{ff} = \frac{\sum_{x=1}^{N} t_x}{N \times T_{fin}} \tag{3.1}$$

- E_{ff} je efektivita (desatinné číslo)
- $\sum_{x=1}^{N} t_x$ suma všetkých časov, ktoré uzly strávili výpočtami
- N je počet uzlov, ktoré pracovali na úlohe
- T_{fin} je celkový čas danej úlohy

Pomocou efektivity budem schopný vyjadriť réžiu. Réžia bude označovať čas, v ktorom sa prijímali a odosielali dáta. Toto percento vypočítame pomocou vzorca:

$$R = 1 - E_{ff} \tag{3.2}$$

- E_{ff} je efektivita (desatinné číslo),
- R je réžia

3.5.2 Škálovateľnosť distribúcie výpočtov

Škálovateľnosť je definovaná ako schopnosť systému, siete alebo procesu vysporiadať sa s narastajúcim počtom práce a prispôsobenie sa tomuto nárastu. Táto metrika bude skúmať pomer efektívnosti a réžie vzhľadom k narastajúcemu množstvu práce. To znamená, že pokiaľ pri slovníkovom útoku budeme používať rôzne veľkosti slovníkov, táto metrika nám ukáže, koľko času sme strávili výpočtami a koľko prijímaním a odosielaním dát.

3.5.3 Rýchlosť lámania hesiel

Keďže sa jedná o distribuovaný proces chcem zistiť, ako počet uzlov zapojených do lámania hesiel ovplyvňuje rýchlosť lámania. Je samozrejmé, že vyšší počet uzlov by mal znamenať vyššiu rýchlosť lámania hesiel, avšak chcem pozorovať túto zmenu a zistiť jej charakteristiku, ktorú popíšem pomocou grafov.

3.6 Faktory ovplyvňujúce distribúciu výpočtov

Distribúcia výpočtov funguje vďaka sieti. Táto sieť však môže mať rôzne parametre, ktoré ovplyvňujú kvalitu komunikácie. Medzi tieto parametre patrí:

- rýchlosť linky,
- použité zariadenia,
- topológia.

V súvislosti s tými faktormi budeme počas vykonávania experimentov sledovať metriky, ktoré nám pomôžu ohodnotiť kvalitu jednotlivých testovaných topológii. Medzi tieto metriky budú patriť:

- rýchlosť linky,
- vyťaženie linky.

3.6.1 Rýchlosť linky

Rýchlosť sieťového pripojenia je z pohľadu distribúcie veľmi podstatná. V prípade, že budem prevádzať slovníkový útok a veľkosť slovníku bude príliš veľká, nízka rýchlosť pripojenia by mohla výrazne ovplyvniť efektívnosť lámania hesiel.

3.6.2 Vyťaženie siete

V rámci tejto práce sa jedná o najpodstatnejšie hodnotiace kritérium. Pomocou tejto metriky budem skúmať počet odoslaných a prijatých dát a správanie siete počas distribúcie. Pomocou tejto metriky budem popisovať stav siete. K tomuto účelu použijem protokol *Simple Management Protocol* (SNMP).

Kapitola 4

Návrh automatizovaných experimentov

V prvej časti tejto kapitoly sa budem venovať použitým technológiam a riešeniu automatizácie pri distribuovanom lámani hesiel, pomocou nástrojov Fitcrack a Hashtopolis. Tieto riešenia sú inšpirované skúsenosťami získanými počas prítomnosti u testovania týchto nástrojov a metríkami pomocou, ktorých ich budem porovnávať. V druhej časti tejto kapitoly sa budem venovať návrhom experimentov pomocou, ktorých budem testovať použité nástroje. V rámci týchto návrhov uvediem konkrétne úlohy, testovacie topológie a informácie potrebné k vytvorení týchto topológii.

4.1 Použité technológie

V tejto časti kapily sa budem venovať použitým technológiam, ktoré mi pomohli realizovať experimenty a prezentovať výsledky.

4.1.1 Matlab

Matlab[1] bol použitý k vytvoreniu reprezentácie výsledkov pomocou stĺpcových grafov.

4.1.2 Python

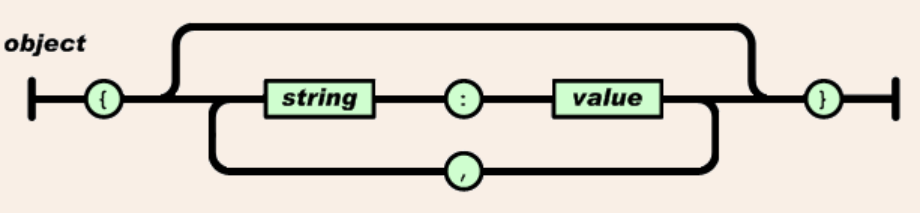
Pre implementáciu automatizovaných skriptov som sa rozhodol využiť jazyk Python [22]. Pre tento jazyk som sa rozhodol z toho dôvodu, že poskytuje mnoho implementovaných knižníc pre spracovávanie vstupných argumentov [2], rôznych formátov dát napr. JSON[3], prácu s databázami[4], REST API[5] rozhraním a protokolom SNMP[6].

4.1.3 JSON

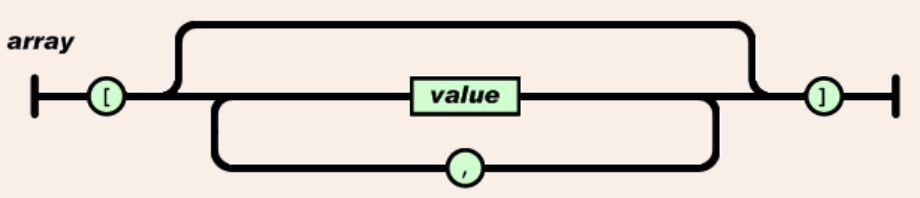
JavaScript Object Notation (JSON) [3] predstavuje štandart popisujúci formát dát, ktorý je nezávislý na platfotme a slúži k prenosu dát. Tieto dáta sú organizované v poliach alebo

[1] https://www.mathworks.com/products/matlab.html
[2] https://docs.python.org/2/library/optparse.html
[3] https://docs.python.org/3/library/json.html
[4] https://www.w3schools.com/python/python_mysql_getstarted.asp
[5] https://realpython.com/python-requests/
[6] http://snmplabs.com/pysnmp/index.html

sú agregované v objektoch. Vstupné dáta môžu obsahovať reťazce, čísla, polia, objekty a boolevské hodnoty. Tieto dáta sú prevedené do výstupného reťazca obsahujúceho názvy hodnôt a hodnoty samotné oddelené špeciálnych znakov. Týmito znakmi sú ohraničujúce objekty, hrtanaté zátvorky ([]) ohraničujúce polia, dvojbodka (:) označujúca priradenie názvu hodnoty ku konkrétnej hodnote a čiarka (,) označujúca oddelovač. Názov hodnoty a konkrétne hodnoty sú uvedené v úvodzovkách. Zápis syntaxe formátu JSON je uvedená na obrázkoch 4.1 a 4.2.



Obr. 4.1: Znázornenie vytvorenia objektov vo formáte JSON.



Obr. 4.2: Znázornenie vytvorenia polí vo formáte JSON.

4.1.4 Protokol SNMP

V rámci monitorovania siete som sa rozhodol používať *Simple Network Management protocol* (SNMP). Tento protokol mi umožnil skúmať počty oktétov prenášaných cez konkrétne rozhranie počas distribuovaných výpočtov. Tieto počty sú uchovávané pomocou dvoch hodnôt predstavujúce 32 bitové nezáporné čísla. Názvy týchto hodnôt sú:

- ifInOctets,
- ifOutOctets.

IfInOctets

Hodnota ifInOctets označuje celkový počet oktetov, ktoré boli prijaté daným rozhraním. Túto hodnotu získavam pomocou OID hodnoty 1.3.6.1.2.1.2.2.1.10.

IfOutOctets

Druhá hodnota ifOutOctets označuje celkový počet oktetov, ktoré boli odoslané daným rozhraním. Túto hodnotu získavam pomocou OID hodnoty 1.3.6.1.2.1.2.2.1.16.

4.1.5 Získanie hodnôt

K tomu, aby som mohol tieto dáta získavať, musel byť nastavený SNMP agent na jednom zo zariadení v testovacích topológiach. Následne som pomocou skriptov v periodických intervaloch požadoval hodnoty IfInOctets a IfOutOctets. V rámci experimntov boli použité 32-bitové hodnoty, ktoré však pri vysoko rýchlostných pripojeniach nestačia k uchovávaniu týchto hodnôt. Z toho dôvodu je zvolená perioda získavania hodnôt dostatočne malá, aby bolo možné pozorovať pretečenie, a bolo možné dopočítať výsledný počet prenesených oktetov jednotlivých hodnôt. Toto dopočítanie prebieha tak, že si spočítam počet prenesených oktetov pred pretečením tak, že od maximálnej hodnoty 32 bitového bezznamienkového čísla odpočítam počet oktétov pred začatím úlohy a k výslednému číslu pripočítam hodnoty prenesených oktétov.

4.2 Automatizácia nástroja Fitcrack

Pri práci s nástrojom Fitcrack bolo potrebné vytvoriť skripty umožňujúce manipuláciu s dátami v databáze. Keďže web server Fitcracku má implementované REST API rozhranie[17] pomocou, ktorého komunikuje s databázov, rozhodol som sa využiť tohoto rozhrania. Vďaka jeho využitiu som bol schopný ľahko manipulovať s dátami v databáze, keďže knižnica *requests*[7] umožňuje posielať request správy obsahujúce dáta písané v JSON formáte pomocou HTTP metód [6]. V jazyku Python[8] a s využitím knižnice *requests* som implementoval následujúce skripty:

- `add_job_fitcrack.py`,
- `delete_job_fitcrack.py`,
- `get_dictionaries.py`,
- `get_all_hosts.py`,
- `get_all_jobs.py`,
- `get_result.py`,
- `start_job _fitcrack.py`.

4.2.1 Konfigurácia skriptov

V rámci automatizácie bolo nutné navrhnúť možnosť pripájať sa na rôzne servery, na ktorých beží nástroj Fitcrack. Z tohoto dôvodu bolo nutné vytvoriť dva moduly, ktoré používa každý z uvedených skriptov. Týmito modulmi sú `structures.py` a `login.py`. Modul `structures.py` obsahuje mapovacie slovníky pre hešovacie algoritmy, URL adresy potrebné k manipulácii s dátami v databáze a mapovanie uzlov. Príklady URL adries použitých v skriptoch sú uvedené v tabuľke 4.1. Modul `login.py` definuje prihlasovacie údaje potrebné pre autentizáciu. URL adresy pre prácu s nástrojom Fitcrack.

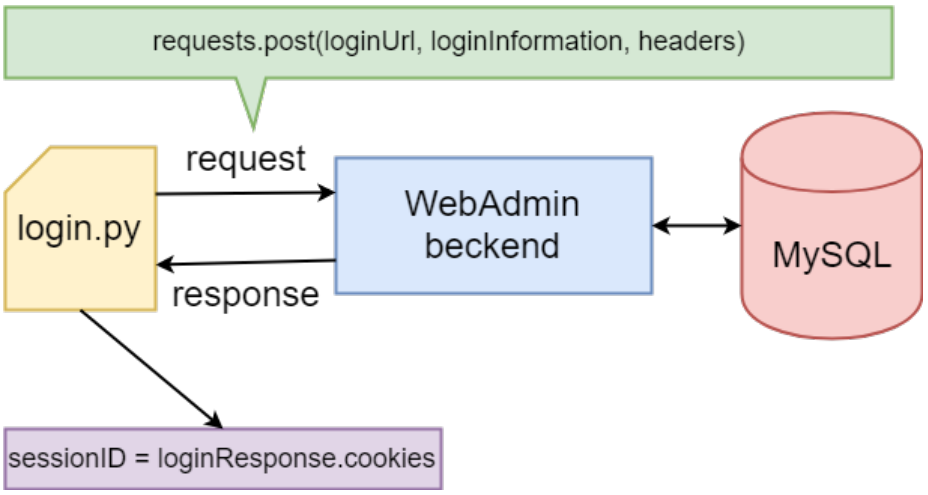
[7] https://realpython.com/python-requests/
[8] https://www.python.org/

URL adresa	Použitie
http://fitcrack.fit.vutbr.cz:15000/login	Prihlasovanie
http://fitcrack.fit.vutbr.cz:15000/jobs	Manipulácia so všetkými úlohami
http://fitcrack.fit.vutbr.cz:15000/jobs/ID	Manipulácia s konkrétnou úlohou
http://fitcrack.fit.vutbr.cz:15000/jobs/ID/action	Nastavenie akcie konkrétnej úlohy
http://fitcrack.fit.vutbr.cz:15000/hosts	Manipulácie so všetkými uzlami
http://fitcrack.fit.vutbr.cz:15000/dictionary	Manipulácia so všetkými slovníkmi

Tabuľka 4.1: URL adresy pre testovací server nástroja Fitcrack.

4.2.2 Autentizácia

K tomu, aby som mohol pristupovať a upravovať riadky v databáze bolo nutné, aby som sa autentizoval. Autentizácia v REST API rozhraní web servera nástroja Fitcrack využíva session [17]. Potom, ako sa úspešne prihlásim pomocou requestu, v ktorom posielam správne prihlasovacie údaje, server vytvorí session a prostredníctvom response správy zasiela session ID v časti cookies. V rámci každej ďalšej request správy je nutné posielať toto session ID. Prihlasovacie údaje sú uvedené v modulu `login.py`. Popis tejto komunikácie je uvedený na obrázku 4.3.



Obr. 4.3: Získanie session ID z response správy nástroj Fitcrack.

4.2.3 Vytvorenie novej úlohy

Prvým krokom k vytvoreniu novej úlohy pomocou skriptu `add_job_fitcrack.py` je správne zadanie vstupných dát pomocou jednotlivých argumentov. Medzi argumenty, ktoré je nutné zadať patria: názov, komentár, typ hešu(podporované sú len SHA1, MD5 a Whirlpool), konkrétny heš, masku alebo názov slovníku (musí byť uložený v databáze pred vytvorením úlohy), počet uzlov (podporované sú párne čísla od 2 do 16), veľkosť úlohy. Tieto argumenty obsahujú všetky potrebné dáta k vytvoreniu novej úlohy slovníkového útoku alebo útoku hrubou silou pomocou masky. Tieto dva útoky sa v rámci argumentov rozlšujú podľa toho, či je medzi vstupnými argumentmi definovaný slovník alebo maska. Po správnom zadaní všetkých potrebných argumentov v rámci konkrétneho útoku je vytvorený objekt dictionary, táto premenná sa volá `parameters` obsahujúci všetky potrebné dáta a flag hodnoty ozna-

čujúce argumenty, ktoré boli zadané. Následne sa pomocou skriptu vykoná autentizácia, z ktorej sa do premennej uloží session ID. V ďalšom kroku sa na základe hodnoty argumentov rozhodne, aká úloha bude vytvorená.

Slovníkový útok

Pokiaľ sa v hodnotách uložených v premennej `parameters` nachádza pri flagu označujúcom zadanie slovníka (`parameters[dictionary_flag]`) nachádza pravdivá hodnota True, vytvára sa slovníkový útok. Pri vytváraní sa použije predom pripravená JSON správa určená pre tento typ útoku. Príklad takejto správy je uvedený na obrázku 4.4.


```
{
  "name": "rest dictionary test1",
  "comment": "rest test",
  "priority": 0,
  "hosts_ids": [
    13,
    3,
    2,
    1
  ],
  "seconds_per_job": 60,
  "time_start": "",
  "time_end": "",
  "attack_settings": {
    "attack_mode": 0,
    "attack_name": "dict",
    "rules": null,
    "left_dictionaries": [
      {
        "id": 14,
        \"name": "bible.txt",
        "keyspace": 12570,
        "time": "2019-04-16T15:28:52"
      }
    ]
  },
  "hash_settings": {
    "hash_type": "100",
    "hash_list": [
      {
        "hash": "fb3931ff2f2c440774ab456622ef58fd4ab4a905",
        "result": "OK",
        "isInCache": false
      }
    ]
  }
}
```


Obr. 4.4: Príklad JSON správy pre slovníkový útok.

V tejto správe budú upravené jednotlivé dáta na základe hodnôt argumentov uložených v `parameters`. Pri slovníkovom útoku je však nutné vo vytvorenej štruktúre pridať informácie o slovníku, ktoré sa nezadávajú pomocou argumentov. Jedná sa o počet riadkov daného slovníka a čas vytvorenia. Tieto hodnoty sú načítané skriptom z databáze pomocou poskytnutého mena slovníka. Z tohoto dôvodu je nutné, aby bol tento slovník uložený v databáze eše pred vytvorením tejto úlohy.

Následne sa odošle request správa obsahujúca JSON dáta, session ID a potrebné hlavičky na URL adresu, ktorá je určená pre manipuláciu s úlohami. Pri úspešnom vytvorení sa v response správe pošlú informácie o vytvorenej úlohe v opačnom pripáde sa vráti chybový kód. Príklad tejto komunikácii je uvedený na obrázku 4.6.

Útok hrubou silou

Pokiaľ sa v hodnotách premennej `parameters` nachádza pri flagu označujúcom zadanie masky (`parameters["mask_flag"]`) pravdivá hodnota True, vytvára sa útok hrubou silou. Podobne ako pri slovníkovom útoku je použitá JSON štruktúra, ktorá je upravená pomocou hodnôt z premennej `parameters`. Príklad tejto štruktúry je na obrázku 4.5.

Potom ako je JSON správa pripravená odošle sa pomocou request správy na server rovnako ako u slovníkového útoku. Komunikácia je znázornená na obrázku 4.6.

4.2.4 Odstránenie úlohy

Skript `delete_job_fitcrack.py` slúži k odstráneniu úlohy podľa zadaného ID. Tento skript poskytuje dva spôsoby použitia.

Zistenie ID

Prvým je vypísanie informácii o úlohách podľa názvu. V prípade, keď poznáme názov úlohy, ktorú chceme zmazať, ale nepoznáme je ID môžeme použiť parameter skriptu n", za ktorý dopíšeme názov úlohy, ktorú chceme zmazať. Skript pošle request správu, ktorá požaduje zoznam všetkých úloh. Táto správa je odoslaná na URL adresu pre manipuláciu s úlohami. Následne v tomto zozname nájde všetky úlohy s daným názvom a vypíše základné informácie vrátane ID.

Odstránenie úlohy

Druhý spôsob použitia je zmazanie úlohy podľa konkrétneho ID, ktoré je uvedené s parametrom "d". Skript pošle request správu pre zmazanie danej úlohy. Táto správa je odoslaná na URL adresu pre konkrétnu úlohu. Úloha odstránená pomocou skriptu, nie je zmazaná úplne, je len nastavená ako skrytá.

4.2.5 Zobrazenie slovníkov

Skript `get_dictionaries.py` načíta základné informácie o všetkých slovníkoch uložených v databáze a vypíše ich na štandardný výstup. Tento skript slúži pre vypísanie slovníkov, ktoré môžu byť použité vo vytváranej úlohe. Potom, ako sa úspešne autentizuje, je poslaná request správa na URL pre manipuláciu so slovníkmi, ktorá žiada informácie o všetkých slovníkoch. Následne response správa obsahuje všetky informácie o uložených slovníkoch alebo chybový kód. Z týchto všetkých informácií sú vypísané len základné ako sú: ID slovníku, veľkosť slovníku, naázov slovníku a čas vytvorenia slovníku.

4.2.6 Zobrazenie uzlov

Skript `get_all_hosts.py` načíta informácie o všetkých uzloch, ktoré sú úložené v databáze a vypíše ich na štandardný výstup. Pomocou týchto informácii môžeme upraviť mapovanie uzlov, uložené v časti skriptu `structures.py`. Potom, ako sa úspešne autentizuje je zaslaná request správa na URL pre manipuláciu s uzlami. Táto správa žiada o informácie o všetkých uzloch uložených v databáze. Response správa obsahuje buď požadované informácie alebo chybový kód. Zo všetkých informácií sú vypísané len základne a to: ID uzla, názov uzla a jeho IP adresa.

4.2.7 Zobrazenie úloh

Skript `get_all_jobs.py` načíta informácie o všetkých úlohách, ktoré sú úložené v databáze a vypíše ich na štandardný výstup. Potom, ako sa úspešne autentizuje je zaslaná request správa na URL pre manipuláciu s úlohami. Táto správa žiada o informácie o všetkých úlohách uložených v databáze. Response správa obsahuje buď požadované informácie, alebo chybový kód.

4.2.8 Zobrazenie výsledkov

Skript `get_result.py` načíta informácie o všetkých úlohách, ktoré boli spustené a vypíše základné informácie o každej z nich na štandardný výstup. Medzi tieto informácie patria: ID, meno, komentár, status a ďalšie.

4.2.9 Spustenie úloh

Skript `start_job _fitcrack.py` umožňuje spustiť úlohy, ktoré sú pripravené (majú status ready), získava informácie od SNMP agenta nastaveného v danej sieťovej topológii a po skončení úlohy informácie o danej úlohe. Tento skript poskytuje 3 možnosti spustenia úloh.

Možnosti spustenia

Prvou možnosťou je spustenie s argumentom -d a hodnoty ID úlohy. Táto možnosť spustí jednu úlohu podľa ID. Ďalšou možnosťou je spustenie pomocou argumentu -s a hodnoty ready, ktorá spustí postupne všetky úlohy, ktorých staus je ready. Poslednou možnosťou, ako spustiť úlohu, je pomocou argumentu -c a vlastnej textovej hodnoty (tag). V tomto prípade sa postupne spustia všetky úlohy, ktoré majú v komentároch uloženú textovú hodnotu zadanú s argumentom -c.

Priebeh spúšťania

Pokiaľ bolo zadané ID úlohy, ktorá sa má spustiť, pošle sa request správa, ktorá zmení stav úlohy so zadaným ID z ready na start. Následne je implementované pasívne čakanie na dokončenie danej úlohy. Počas tohoto čakania sa na štandardný výstup vypisujú informácie o SNMP počítadlách oktetov. Tento výpis je realizovaný každých 20 sekúnd, a to z dôvodu sledovania priebehu vyťaženia rozhraní, z ktorých tieto dáta získavame a zároveň kontrole pretečenia sledovaných SNMP hodnôt. Pokiaľ sa status úlohy zmení na finished, exhausted alebo malformed, pasívne čakanie je ukončené, a sú vypísané základné výsledky danej úlohy. Pokiaľ sa jedná o dávkové spúšťanie úloh pomocou argumentu sälebo c", skript načíta z databáze informácie o všetkých úlohách, následne sa porovnávajú hodnoty komentáre alebo

hodnoty statusov úloh uložených v databáze. Skript prechádza jednú úlohu za druhou a porovnáva ich hodnoty. Pri zhode sa spustí daná úloha, ako pri spúšťaní pomocou ID.

4.3 Automatizácia nástroja Hashtopolis

Podobne, ako automatizácia nástroja Fitcrack je aj automatizácia nástroja Hashtopolis realizovaná pomocou jazyka Python. V rámci nástroja Hashtopolis boli implementované následujúce skripty:

- `startAttack.py`,
- `stopAttack.py`.

Tieto dva skripty budú manipolovať so stĺpcom isActive, ktorý sa nachádza v tabuľke Agent.

4.3.1 Autentizácia nástroja Hashtopolis

V rámci automatizácie nástroja Hashtopolis je použitá knižnica jazyka Python MySQLdb[9], ktorá umožňuje priame pripojenie na databázu. Vďaka tomuto pripojeniu som schopný manipulovať s databázov.

4.3.2 Spústenie úlohy

Spustenie úlohy v nástroji Hashtopolis má na starosť skript `startAttack.py`. Tento skript nastavuje aktivitu uzlov zapojených do distribuovaného lámanie hesiel na hodnotu 1. V rámci nástroja Hashtopolis to znamená, že pokiaľ je daný uzol aktívny a zároveň má pridelenú úlohu začne na nej pracovať. Predtým, ako budú uzly aktivované sa ukladá počiatočný čas úlohy a hodnoty SNMP počítadiel. Následne sa v jednoduchom cykle každých 20 sekúnd vypisujú hodnoty SNMP. Všetky hodnoty sú vypísané na štandardný výstup.

4.3.3 Zastavenie úlohy

Po skončení úlohy je nutné nastaviť aktivitu uzlov nástroja Hashtopolis na 0. To znamená, že môžem uzlom priradiť ďalšiu úlohu, ktorú budú riešiť. Až vtedy, keď im znova nastavím aktivitu na 1, pomocou skriptu `startAttack.py`. Túto úlohu ma na starosti skript `stopAttack.py`. Potom, ako sú deaktivované uzly skript uloží a vypíše koncový čas riešenia danej úlohy.

[9] https://www.tutorialspoint.com/python/python_database_access.htm

4.4 Návrhy experimentov

V následujúcej časti tejto kapitoly sa budem venovať návrhom konkrétnych experimentov , ktoré som vykonal.

4.4.1 Prostredie experimentov

Pre účely tejto práce a vykonanie experimentov mi bol pridelený prístup do špeciálnej učebne, ktorá je vybavený dvadsiatimi počítačmi. Tieto počítače boli použité ako výpočtové uzly. Špecifikácia všetkých počítačov je rovnaký a uvedená v tabuľke 4.2.

Systém operačný systém	CPU	GPU
Windows 7, Centos 7	Intel Core i5-3570K, 3.40GHz	GTX 1050 Ti

Tabuľka 4.2: Špecifikácia počítačov využitých pre distribuované lámanie hesiel.

Ďalším užitočným vybavením sú Cisco, HP zariadenia (prepínače, smerovače) a prepojovacia kabeláž (priame, nepriama RJ-45 UTP káble), ktoré zabezpečujú prepojenie uzlov pomocou eternetu [7]. Pomocou týchto zariadení a kabeláže som bol schopný zostaviť variácie testovacích topológii, na ktorých bolo vykonané distribuované lámanie hesiel. V rámci mojich experimentov a k zostaveniu sieťových topológii boli použité tri cisco smerovače, z ktorých každý jeden disponoval tromi rozhraniami, a taktiež boli použité 4 HP prepínače.

4.4.2 Špecifikácia útokov

Distribuované lámanie hesiel pomocou nástrojov Fitcrack a Hashtopolis som skúmal pomocou 2 typov útokov. Týmito typmi sú útok hrubou silou a slovníkový útok. Pre tieto dva útoky som sa rozhodol z dôvodu, že sa jedná o základné varianty útokov, narozdiel od kombinačného útoku, ktorý je rozšírenou verziou slovníkového útoku a hybridných útokov ktoré sú kombináciou útoku hrubou silou a slovníkového útoku. Z toho dôvodu faktory, ktoré ovplyvnia útok hrubou silou a distribuovaný útok ovplyvnia aj odvodené útoky.

V rámci oboch útokov bol hľadaný heš vytvorený pomocou hešovacieho algoritmu whirlpool. Pre tento typ som sa rozhodol kvôli vyššej výpočtovej náročnosti, ktorá umožní rovnomernejšie rozdelenie práce medzi viacero výpočtových uzlov.

Útok hrubou silou

V rámci experimentov s útokom hrubou silou boli vykonané tri útoky na každej z troch sieťových topológii s rozdielnym počtom pripojených uzlov. V rámci týchto útokov jednotlivé uzly zapojené do distribuovaného lámania hesiel hľadali heslo dĺžky 9 znakov, zahešované pomocou hešovacieho algoritmu whirlpool pomocou jednej masky, ktorej dĺžka je 7. Dĺžku hľadanej masky som zvolil tak, aby proces lámania hesiel netrval príliš dlho. Experimentoval som aj s väčšími maskami, pri ktorých sa ukázalo, že trvanie experimentu zabralo zbytočne veľa času. Čo pre účely tejto práce nebolo nutné.

Pre účely mojich experimentov a pre prepočítanie celej množiny vygenerovaných hesiel, ktoré nám použitá maska poskytuje, som sa rozhodol použiť masku, ktorá nezodpovedá hľadanému heslu. Týmto som zabránil tomu, aby výpočet skončil skôr, než uzly prekontrolujú celú množinu vygenerovaných hesiel. Špecifikácie jednotlivých útokov sú uvedené v tabuľke 4.3.

ID	Názov	Počet uzlov
a	BP_BA_4	4
b	BP_BA_8	8
c	BP_BA_12	12
d	BP_BA_4	4
e	BP_BA_8	8
f	BP_BA_12	12
g	BP_BA_4	4
h	BP_BA_8	8
i	BP_BA_12	12

Tabuľka 4.3: Špecifikácie útokov hrubou silou.

ID je unikátny identifikátor sieťovej topológie a počtu uzlov, na ktorých bol daný útok prevedený. V rámci týchto experimentov bola použitá maska ?l?l?l?l?l?l?l, hešovací algoritmus Whirlpool a veľkosť workunit (Fitcrack), chunk (Hashtopolis) bola nastavená na 60.

Slovníkový útok

V experimentoch so slovníkovými útokmi boli vykonané tri útoky na každej z troch sieťových topológii s rozdielnym počtom pripojených uzlov. V rámci týchto útokov jednotlivé uzly zapojené do distribuovaného lámania hesiel hľadali heslo zahešované pomocou hešovacieho algoritmu whirlpool v troch rôzne veľkých slovníkoch. Informácie o použitých slovníkoch sú uvedené v tabuľke 4.4.

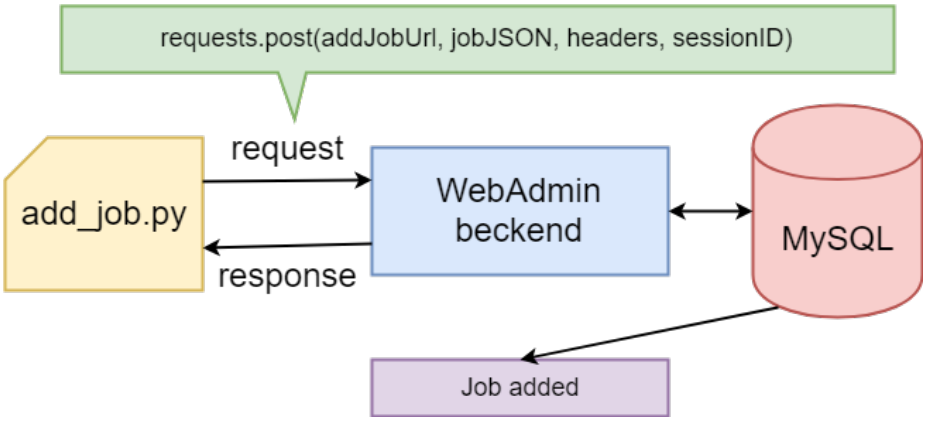
Názov slovníka	Počet hesiel	Veľkosť slovníka
4xrockyou.txt	57 038 040	550 MB
8xrockyou.txt	114 076 080	1.1 GB
16xrockyou.txt	228 152 160	2.2 GB

Tabuľka 4.4: Slovníky použité pri experimentoch.

Pre účely mojich experimentov a pre prepočítanie celej množiny testovacích hesiel, ktoré nám použité slovníky poskytujú, som sa rozhodol vygenerovať pomocou hešovacej funkcie whirlpool heš pre heslo, ktoré sa v daných slovníkoch nevyskytuje. Týmto som zabránil tomu, aby výpočet skončil skôr, než uzly prekontrolujú celú množinu testovacích hesiel. Špecifikácie jednotlivých útokov sú uvedené v tabuľke 4.5.

```
{
  "name": "REST_TEST",
  "comment": "Test pre REST",
  "priority": 0,
  "hosts_ids": [
    13,
    3,
    2,
    1
  ],
  "seconds_per_job": 60,
  "time_start": "",
  "time_end": "",
  "attack_settings": {
    "attack_mode": 3,
    "attack_name": "mask",
    "masks": [
      "?l?l?l?l?l?l"
    ],
    "attack_submode": 0,
    "markov_treshold": null,
    "markov": null,
    "charset": null
  },
  "hash_settings": {
    "hash_type": "100",
    "hash_list": [
      {
        "hash": "40e0ce4a8ec91b2f59373de80e5fd7ffb305961e",
        "result": "OK",
        "isInCache": false
      }
    ]
  }
}
```


Obr. 4.5: Príklad JSON správy pre útok hrubou silou.



Obr. 4.6: Príklad komunikácie.

ID	Názov	Slovník	Počet uzlov
a	BP_DA_8x_4	8xrockyou.txt	4
a	BP_DA_16x_4	16xrockyou.txt	4
a	BP_DA_24x_4	24xrockyou.txt	4
b	BP_DA_8x_8	8xrockyou.txt	8
b	BP_DA_16x_8	16xrockyou.txt	8
b	BP_DA_24x_8	24xrockyou.txt	8
c	BP_DA_8x_12	8xrockyou.txt	12
c	BP_DA_16x_12	16xrockyou.txt	12
c	BP_DA_24x_12	24xrockyou.txt	12
d	BP_DA_8x_4	8xrockyou.txt	4
d	BP_DA_16x_4	16xrockyou.txt	4
d	BP_DA_24x_4	24xrockyou.txt	4
e	BP_DA_8x_8	8xrockyou.txt	8
e	BP_DA_16x_8	16xrockyou.txt	8
e	BP_DA_24x_8	24xrockyou.txt	8
f	BP_DA_8x_12	8xrockyou.txt	12
f	BP_DA_16x_12	16xrockyou.txt	12
f	BP_DA_24x_12	24xrockyou.txt	12
g	BP_DA_8x_4	8xrockyou.txt	4
g	BP_DA_16x_4	16xrockyou.txt	4
g	BP_DA_24x_4	24xrockyou.txt	4
h	BP_DA_8x_8	8xrockyou.txt	8
h	BP_DA_16x_8	16xrockyou.txt	8
h	BP_DA_24x_8	24xrockyou.txt	8
i	BP_DA_8x_12	8xrockyou.txt	12
i	BP_DA_16x_12	16xrockyou.txt	12
i	BP_DA_24x_12	24xrockyou.txt	12

Tabuľka 4.5: Špecifikácie slovníkových útokov.

ID slúži ako unikátny identifikátor sieťovej topológie a počtu uzlov, na ktorých bol daný útok prevedený. V rámci týchto experimentov bol hešovací algoritmus Whirlpool a veľkosť workunit (Fitcrack), chunk (Hashtopolis) bola nastavená na 60.

4.5 Príprava experimentov

V tejto časti kapitoly sa budem venovať úkonom, ktoré súvisia s prípravou experimentov. Týmito úkonmi sú:

- vytvorenie testovacích sietí,
- priradenie IP adries jednotlivým uzlom,
- priradenie IP adries jednotlivým rozhraniam,
- popis testovacích topologii.

4.5.1 Testovacie siete

V rámci experimentov som použil tri sieťové topológie, kde každá z nich má iný počet sieti, ktoré je treba adresovať. Prvá topológia obsahuje tri siete, ktoré je nutné adresovať, druhá topológia obsahuje štyri siete a tretia topológia obsahuje šesť sieti, ktoré je nutné adresovať. Pri vytváraní testovacích sieti bol použitý *Variable-Length Subnet Masking* (VLSM) [18].

Keďže som potreboval adresovať 4 rôzne siete pre výpočtové uzly, a 3 siete pre rozhrania, ktoré prepájajú smerovače, rozhodol som sa použiť sieť 192.168.1.0/24 a vytvoril som z nej šesť podsieti, ktoré budem značiť ako A, B, C, D, E, F. Prvé štyri z týchto podsieti sú rovnako veľké a slúžia k adresovaniu uzlov zapojených do jednotlivých podsieti. Tieto podsiete sú uvedené v tabuľke 4.6.

	A	B	C	D
NET	192.168.1.0	192.168.1.32	192.168.1.64	192.168.1.96
SUB	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.224
FIRST IP	192.168.1.1	192.168.1.33	192.168.1.65	192.168.1.97
LAST IP	192.168.1.31	192.168.1.63	192.168.1.95	192.168.1.127

Tabuľka 4.6: Vytvorené podsiete A, B, C, D pre uzly zapojené do experimentov.

Riadok NET označuje IP adresy sieti, následujúci riadok SUB označuje masky danej siete, riadok FIRST IP označuje prvú použiteľnú adresu siete a riadok LAST IP označuje poslednú použiteľnú adresu v rámci danej siete. Tento popis platí aj pre ďalšie tabuľky 4.7 a 4.8.

	E	F	DHCP
NET	192.168.1.128	192.168.132	——
SUB	255.255.255.252	255.255.255.252	——
R01–GE–0/1	192.168.1.129	——	——
R02–GE–0/0	192.168.1.130	——	——
R01–GE–0/2	——	192.168.1.133	——
R03–GE–0/0	——	192.168.1.134	——
R01–GE–0/0	——	——	10.10.10.206

Tabuľka 4.7: Vytvorené podsiete pre adresovanie rozhraní.

Smerovač R01 má na rozhraní adresu pridelenú pomocou *Dynamic Host Configuration Protocol* (DHCP) [1]. Táto adresa patrí do siete, do ktorej boli pripojené servery nástrojov Fitcrack a Hashtopolis.

	A	B	C	D
NET	192.168.1.0	192.168.1.32	192.168.1.64	192.168.1.96
SUB	255.255.255.224	255.255.255.224	255.255.255.224	255.255.255.224
DG	192.168.1.1	192.168.1.33	192.168.1.65	192.168.1.97
h1	192.168.1.11	192.168.1.41	192.168.1.71	192.168.1.101
h2	192.168.1.12	192.168.1.42	192.168.1.72	192.168.1.102
h3	192.168.1.13	192.168.1.43	192.168.1.73	192.168.1.103
h4	192.168.1.14	192.168.1.44	192.168.1.74	192.168.1.104
h5	192.168.1.15	192.168.1.45	192.168.1.75	192.168.1.105
h6	192.168.1.16	192.168.1.46	192.168.1.76	192.168.1.106
h7	192.168.1.17	192.168.1.47	192.168.1.77	192.168.1.107
h8	192.168.1.18	192.168.1.48	192.168.1.78	192.168.1.108
h9	192.168.1.19	192.168.1.49	192.168.1.79	192.168.1.109
h10	192.168.1.20	192.168.1.50	192.168.1.80	192.168.1.110
h11	192.168.1.21	192.168.1.51	192.168.1.81	192.168.1.111
h12	192.168.1.22	192.168.1.52	192.168.1.82	192.168.1.112
h13	192.168.1.23	192.168.1.53	192.168.1.83	192.168.1.113
h14	192.168.1.24	192.168.1.54	192.168.1.84	192.168.1.114
h15	192.168.1.25	192.168.1.55	192.168.1.85	192.168.1.115
h16	192.168.1.26	192.168.1.56	192.168.1.86	192.168.1.116
h17	192.168.1.27	192.168.1.57	192.168.1.87	192.168.1.117
h18	192.168.1.28	192.168.1.58	192.168.1.88	192.168.1.118

Tabuľka 4.8: Pridelenie IP adries jednotlivým uzlom zapojených do experimentov.

Jednotlivé riadky tabuľky znamenajú:

- NET – adresu siete,
- SUB – masku siete,
- DG – vychodziu bránu,
- h1-h18 – označnenia počítačov.

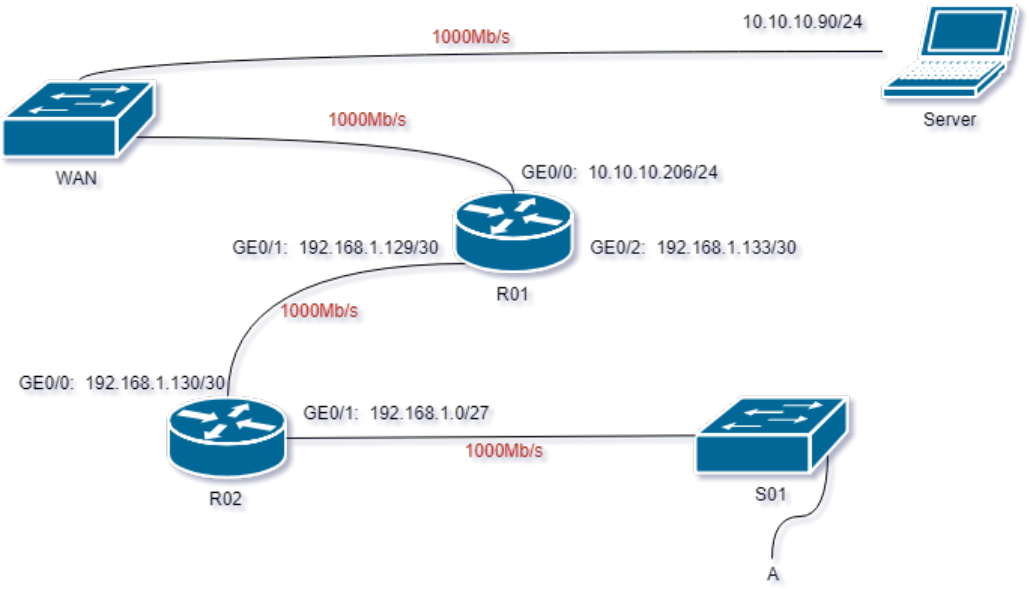
4.5.2 Návrhy testovacích sieťových topológii

Sieťové topológie sú jedným z hlavných ovplyvňujúcich faktorov distribúcie výpočtov. V tejto časti budú uvedené testovacie sieťové topológie, ktoré boli použité v rámci experimentov. Na každej z týchto topológii boli prevádzané všetky slovníkové útoky a–i so slovníkmi, uvedenými v tabuľke 4.5 a všetky útoky hrubou silou a–i uvedené v tabuľke 4.3. Cieľom testovania týchto topológii bolo získať hodnoty metrík potrebných pre zhodnotenie správania distribuovaného lámania hesiel v rámci rôznych sieťových topológii s rovnakými útokmi pri rôznom počte výpočtových uzlov, rôznej rýchlosti liniek, a rôznej veľkosti použitých slovníkov pri slovníkovom útoku.

Topológia pre experimenty a, b, c

Na obrázku 4.7 môžeme vidieť:

- Fitcrack a Hashtopolis server,
- 2x smerovače R01, R02,
- 2x prepínač S01, WAN,
- podsieť označenú ako A,



Obr. 4.7: Návrh topológie pre experimenty a, b, c.

Táto sieťová topológia bola navrhnutá, aby som bol schopný ukázať správanie siete a pripojených uzlov, ktoré majú rovnakú rýchlosť pri distribuovanom lámaní hesiel.

Počty uzlov pripojené v danej podsieti A sú uvedené v tabuľke 4.9. Všetky dáta o jednotlivých útokoch boli získané na strane servera, na ktorom boli spustené skripty, ktoré pomohli tieto dáta získať. Dáta protokolu SNMP boli získané zo SNMP agenta, ktorým bol smerovač R01. Tento agent získaval SNMP hodnoty z rozhrania GE 0/0 na prepínači R02.

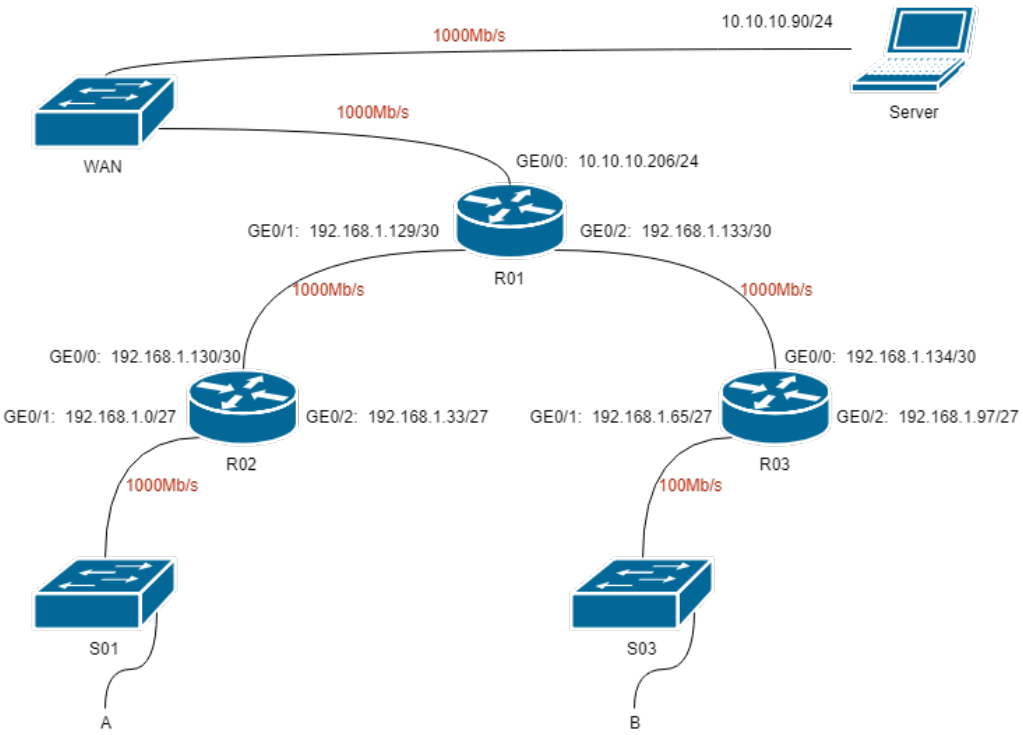
Experiment ID	A
a	4
b	8
c	12
Speed	1000 Mb/s

Tabuľka 4.9: Počet uzlov pripojených v rámci jednotlivých experimentov.

Topológia pre experimenty d, e, f

Na obrázku 4.8 môžeme vidieť:

- 3x smerovač R01, R02, R03,
- 3x prepínač S01, S02, WAN,
- dve podsiete označené ako A, B.



Obr. 4.8: Návrh topológie s viacerými prvkami.

Význam tejto sieťovej topológie je skúmanie vplyvu rozdelenia sieťovej komunikácie do dvoch podsieti, ktoré majú rôzne rýchlosti na distribúciu výpočtov.

Počty uzlov pripojené v danej podsieti A a B a rýchlosti sú uvedené v tabuľke 4.10. Všetky dáta o jednotlivých útokoch boli získané na strane servera, na ktorom boli spustené skripty, ktoré pomohli tieto dáta získať. Dáta protokolu SNMP boli získané z rozhraní GE 0/0 na prepínači R02 a GE 0/0 na prepínači R03. Tieto dáta získaval a poskytoval SNMP agent, ktorým bol prepínač R03.

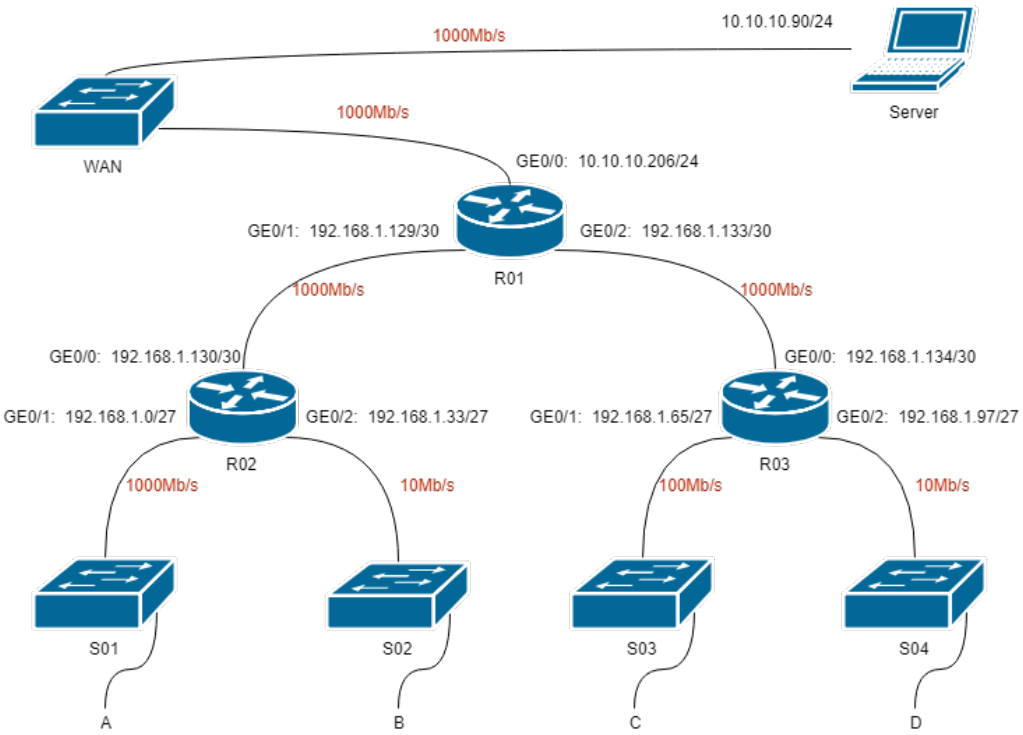
Experiment ID	A	B
d	2	2
e	4	4
f	6	6
Speed	1000 Mb/s	100Mb/s

Tabuľka 4.10: Počet uzlov pripojených v rámci jednotlivých experimentov.

Topológia pre experimenty g, h, i

Na obrázku 4.9 môžeme vidieť:

- 5x prepínač WAN, S01, S02, S03, S04,
- 3x smerovač R01, R02, R03,
- štyri podsiete označené ako A, B, C, D.



Obr. 4.9: Návrh topológie s 2 prepínačmi s rozdielnou rýchlosťou.

V rámci tejto topológie som chcel poukázať na vplyv rôznej a zároveň nízkej rýchlosti linky pri distribuovanom lámaní hesiel.

Počty uzlov pripojené v danej podsieti A a B a rýchlosti sú uvedené v tabuľke 4.11. Všetky dáta o jednotlivých útokoch boli získané na strane servera, na ktorom boli spustené skripty, ktoré pomohli tieto dáta získať. Dáta protokolu SNMP boli získané z rozhraní GE 0/0 na prepínači R02 a GE 0/0 na prepínači R03 prostredníctvom SNMP agenta, ktorým bol smerovač R01.

Experiment ID	A	B	C	D
g	1	1	1	1
h	2	2	2	2
i	3	3	3	3
Speed	1000 Mb/s	100Mb/s	100 Mb/s	10Mb/s

Tabuľka 4.11: Počet uzlov pripojených v rámci jednotlivých experimentov.

Kapitola 5

Experimenty

V rámci odhalenia vplyvu sieťovej infraštruktúry na distribuované lámanie hesiel. Som vykonal experimenty podľa návrhov definovaných v predchádzajúcej kapitole. Vďaka tomu, som bol schopný nameraťa zistiť kľúčové hodnoty potrebné k vyčísleniu jednotlivých metrík. Na základe výsledkov jednotlivých metrík budem schopný vyvodiť závery o tom, aký vplyv má sieťová infraštruktúra na distribuované lámanie hesiel, a taktiež porovnať použité nástroje Fitcrack a Hashtopolis.

5.1 Očakávané výsledky

V tejto časti kapitoly uvediem svoje očakávania výsledkov experimentov.

5.1.1 Slovníkový útok

Fitcrack

Nástroj Fitcrack umožňuje distribúciu slovníka testovacích hesiel po častiach. Z tohoto dôvodu očakávam, že nástroj Fitcrack ukáže vyššiu efektivitu a zároveň nižšiu réžiu počas vykonávania jednotlivých experimentov.

Rýchlosť lámania hesiel bude ovplyvnená rozdelením práce medzi výpočtové uzly. Keďže nástroj Fitcrack nemusí prideliť prácu všetkým uzlom, môže nastať situácia, v ktorej pridelí prácu práve uzlom, ktoré majú pomalšie pripojenie, a tým sa predĺži proces lámania hesiel.

Podobne ako rýchlosť bude ovplyvnené vyťaženie linky závisieť od pridelenia výpočtových uzlov ku konkrétnej úlohe. Vyťaženie liniek by nemuselo byť príliš vysoké, keďže nástroj Fitcrack nemusí prideliť prácu všetkým uzlom.

Hashtopolis

Nástroj Hashtopolis v rámci slovníkového útoku distribuuje celý slovník každému zapojenému klientovi. Pri rozdelení práce zapája každý výpočtový uzol, ktorý bol priradený danej úlohe. Z tohoto dôvodu si myslím, že rýchlosť lámania nástroja Hashtopolis bude veľmi ovplyvnená topológiami s nižšou rýchlosťou pripojení. Podobne, ako rýchlosť lámania hesiel, bude ovplyvnené vyťaženie liniek. Keďže nástroj Hashtopolis zapája do procesu lámania hesiel všetky uzly, pri zvyšujúcom sa počte uzlov, môže dôjsť vysokému vyťaženiu liniek.

Predpoklady výsledkov

V rámci distribúcie slovníkového útoku bude lepším nástrojom práve nástroj Fitcrack, a to v rámci všetkých testovaných metrik. Hlavným dôvodom tohoto predpokladu je spôsob distribúcie slovníka zapojeným uzlom.

5.1.2 Útok hrubou silou

V rámci tohoto útoku predpokladám, že nebude príliš ovplyvnená rýchlosť lámania hesiel, ani vyťaženie linky pri rôznych topológiach, rýchlostiach liniek, ani počte uzlov.

5.2 Výsledky a zhodnotenie experimentov

Významy symbolov v následujúcich tabuľkách sú:

- ID je identifikátor topológie a počtu uzlov použitých pri úlohe,
- uz. označuje počet uzlov pridelených k danej úlohe,
- 1 Gb/s, 100 Mb/s, 10Mb/s reprezentujú počet uzlov, ktoré pracovali na lámaní hesiel pri konkrétnej rýchlosti linky,
- Ge_1^{in}, Ge_1^{out}, Ge_2^{in}, Ge_2^{out} predstavujú vyťaženie linky na konkrétnom rozhraní.

5.2.1 Slovníkový útok

V tabuľkách 5.1, 5.2, 5.3 a 5.4 sú uvedené výsledné hodnoty, ktoré boli získané pri vykonávaní experimentov na nástrojoch Fitcrack a Hashtopolis, počas prevedenia distribuovaného slovníkového útoku.

8x_rockyou.txt

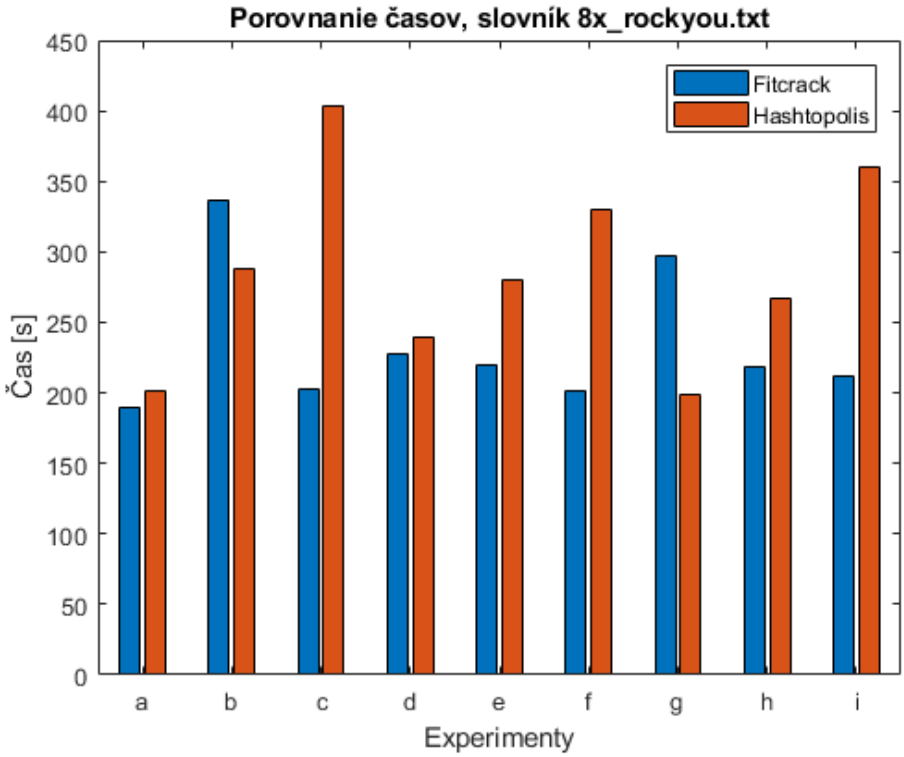
Z výsledných hodnôt efektivity uvedených v tabuľkách, ktoré sú reprezentované pomocou grafov 5.2, 5.6 a 5.10 je vidieť, že nástroj Hashtopolis dosiahol o niečo lepších výsledkov pri distribuovanom slovníkovom útoku, v ktorom figurovali väčšie slovníky (16x_rockyou.txt, 24x_rockyou.txt).

Na druhú stranu z hľadiska dĺžky lámania hesiel, ktorá sú reprezentovaná pomocou grafov 5.1, 5.5 a 5.9 sa ukázalo, že nástroj Fitcrack bol rýchlejším. Z týchto grafov je vidieť, že čím väčší slovník bol použitý, tým bol rozdiel v čase lámania hesiel viditeľnejší.

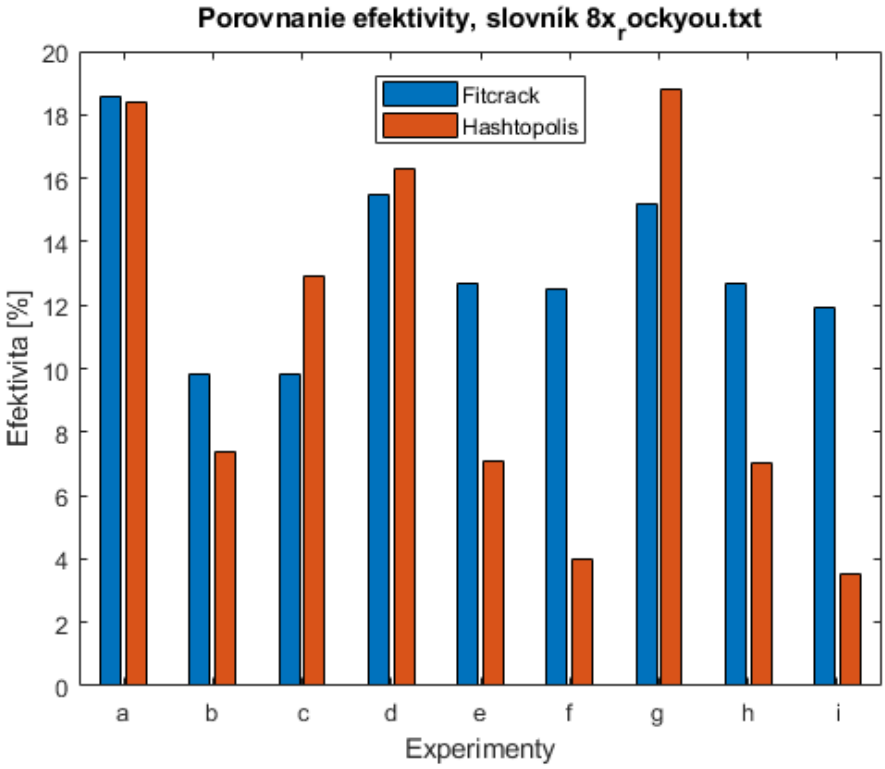
Posledným podstatným meraným kritériom bolo vyťaženie linky. Porovnanie hodnôt tohoto kritéria je uvedené pomocou grafov 5.3, 5.4, 5.7, 5.8, 5.11 a 5.12. Na základe zobrazených výsledkov je vidieť, že nástroj Hashtopolis pri vykonávaní distribuovaného slovníkového útoku spôsobuje oveľa väčšie vyťaženie linky ako nástroj Fitcrack.

Fitcrack: slovník 8x_rockyou.txt (1,1 GB, $114 \cdot 10^6$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	2/4	-	-	0.054	4.847	-	-	02:21	18.6
b	8	2/8	-	-	0.029	2.733	-	-	04:23	9.8
c	12	2/12	-	-	0.058	4.537	-	-	05:14	9.8
d	4	1/2	1/2	-	0.022	1.922	0.045	2.117	02:21	15.5
e	8	2/4	0/4	-	0.023	2.194	0.043	1.992	03:43	12.7
f	12	2/6	0/6	-	0.041	4.558	0.004	0.011	05:04	12.5
g	4	0/1	1/2	0/1	0.022	0.001	0.002	0	03:01	15.2
h	8	0/2	2/4	0/2	0.002	0.001	0.002	0.001	03:42	12.7
i	12	1/3	1/6	0/3	0.025	2.276	0.002	0.001	05:04	11.9
Hashtopolis: slovník 8x_rockyou.txt (1,1 GB, $114 \cdot 10^6$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	4/4	-	-	0.227	15.688	-	-	02:29	18.4
b	8	8/8	-	-	0.407	25.781	-	-	02:50	7.4
c	12	12/12	-	-	0.495	27.567	-	-	02:40	12.9
d	4	2/2	2/2	-	0.118	7.735	0.175	7.735	02:36	16.3
e	8	2/4	4/4	-	0.208	13.26	0.214	8.57	02:39	7.1
f	12	3/6	3/6	-	0.279	16.875	0.229	9.403	02:40	4
g	4	1/1	3/2	0/1	0.072	4.664	0.021	0.881	02:30	18.8
h	8	2/2	4/4	0/2	0.111	6.952	0.023	0.959	02:29	7
i	12	3/3	6/6	0/3	0.133	4.987	0.023	0.915	02:31	3.5

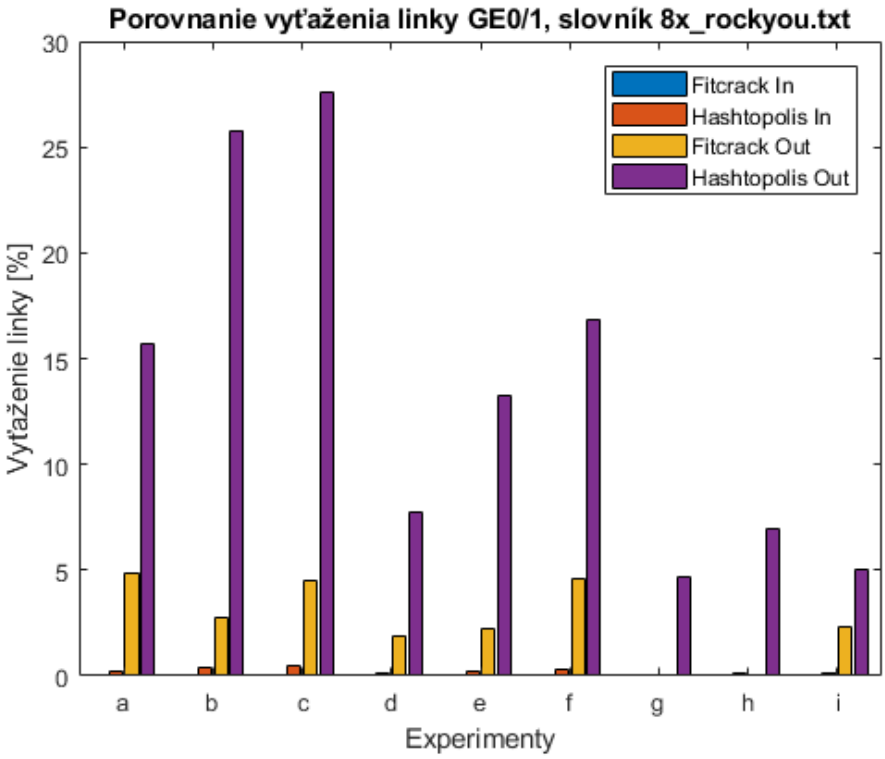
Tabuľka 5.1: Výsledky slovníkového útoku pri použití 8x_rockyou.txt.



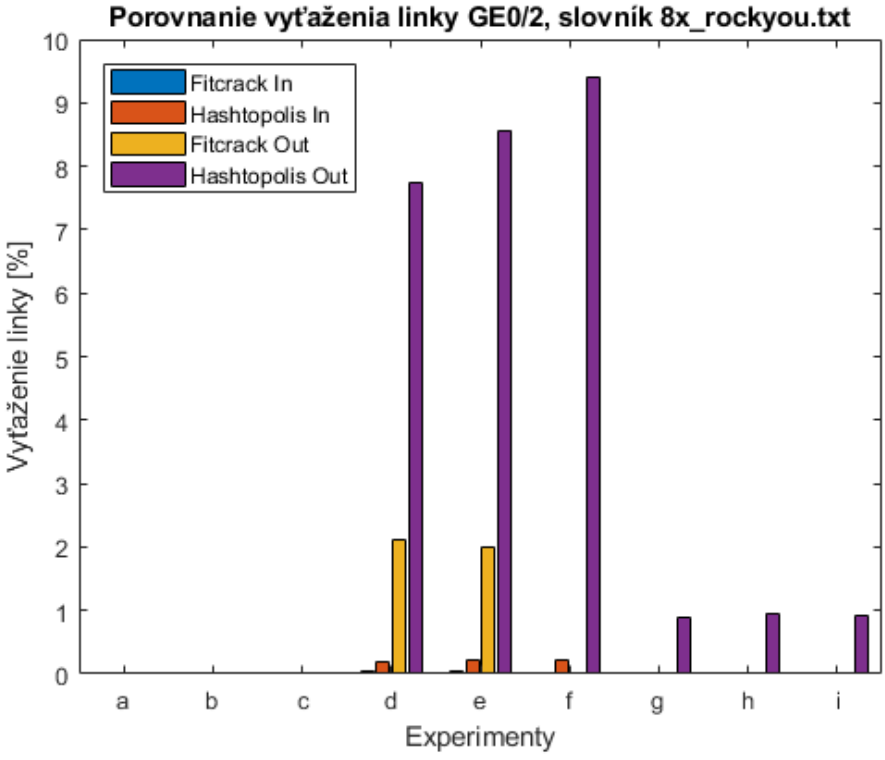
Obr. 5.1: Porovnanie dĺžky lámania hesiel.



Obr. 5.2: Porovnanie efektivity.



Obr. 5.3: Porovnanie vyťaženia linky GE0/1.

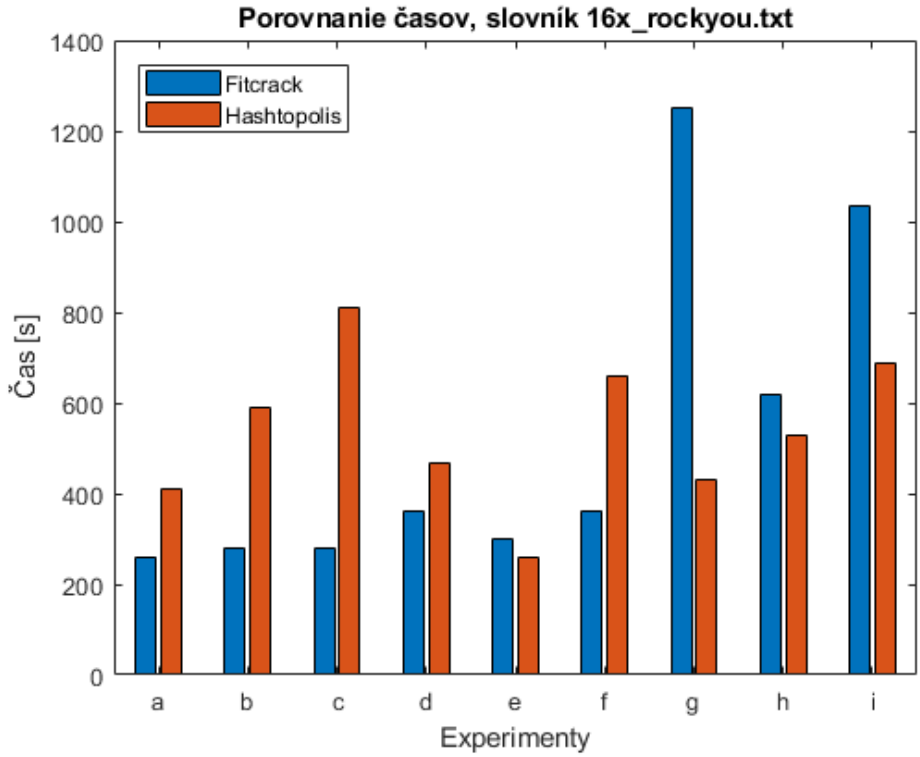


Obr. 5.4: Porovnanie vyťaženia linky GE0/2.

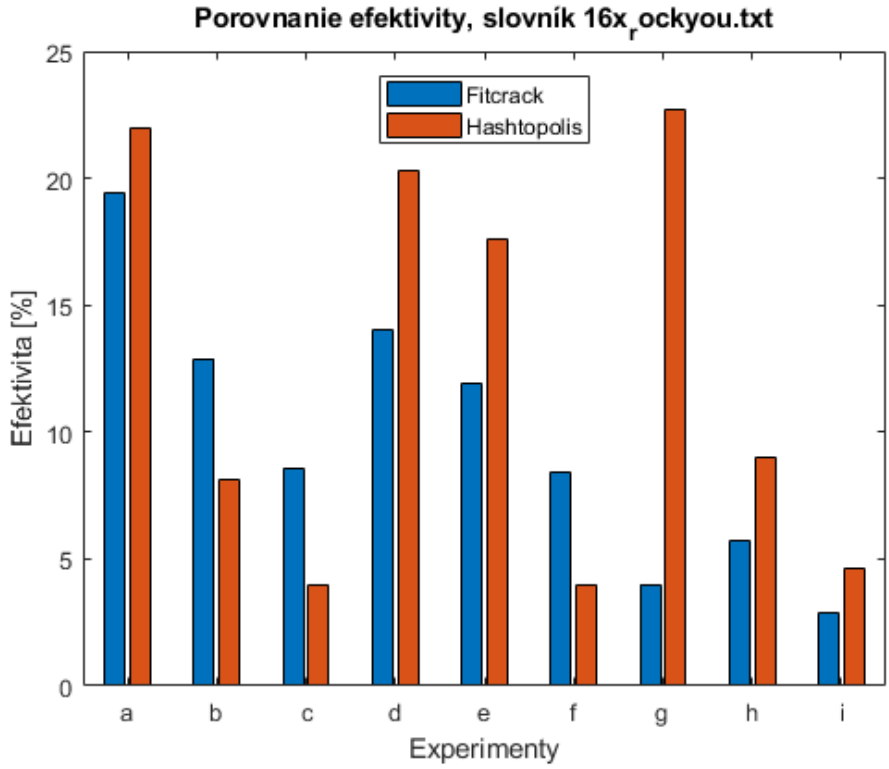
16x_rockyou.txt

Fitcrack: slovník 16x_rockyou.txt (2,2 GB, $228 \cdot 10^6$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	3/4	-	-	0.067	12.214	-	-	03:22	19.4
b	8	4/8	-	-	0.067	6.624	-	-	04:48	12.9
c	12	4/12	-	-	0.063	6.576	-	-	04:48	8.6
d	4	1/2	1/2	-	0.023	2.415	0.056	2.673	03:22	14
e	8	2/4	2/4	-	0.029	2.925	0.069	3.255	04:43	11.9
f	12	1/6	3/6	-	0.017	1.075	0.085	4.012	06:05	8.4
g	4	1/1	1/2	1/1	0.007	0.312	0.029	0.791	03:22	4
h	8	1/2	2/4	1/2	0.01	0.783	0.014	0.628	04:43	5.7
i	12	1/3	1/6	2/3	0.006	0.47	0.020	0.844	06:05	2.9
Hashtopolis: slovník 16x_rockyou.txt (2,2 GB, $228 \cdot 10^6$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	4/4	-	-	0.262	18.110	-	-	06:04	22
b	8	8/8	-	-	0.403	25.255	-	-	06:20	8.1
c	12	12/12	-	-	0.499	23.286	-	-	06:26	4
d	4	2/2	2/2	-	0.119	7.967	0.18	7.967	06:19	20.3
e	8	2/4	2/4	-	0.227	14.279	0.211	8.917	06:06	17.6
f	12	3/6	3/6	-	0.279	16.849	0.232	9.501	05:17	4
g	4	1/1	2/2	0/1	0.065	4.307	0.022	0.94	06:32	22.7
h	8	2/2	4/4	0/2	0.112	7.045	0.023	0.968	06:20	9
i	12	3/3	6/6	0/3	0.139	8.082	0.024	0.966	06:20	4.6

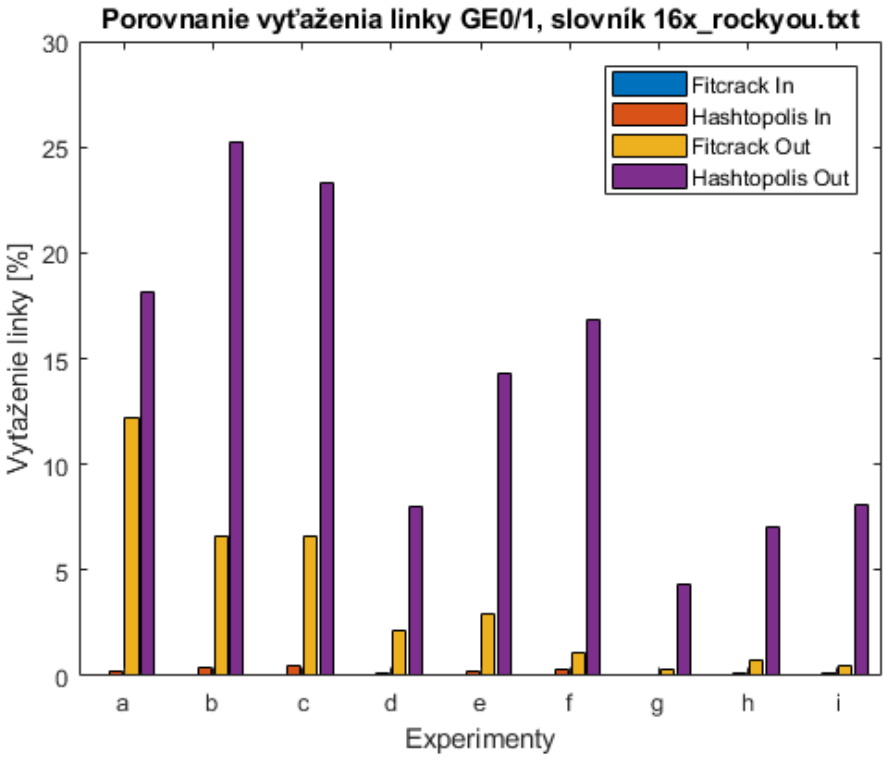
Tabuľka 5.2: Výsledky slovníkového útoku pri použití 16x_rockyou.txt.



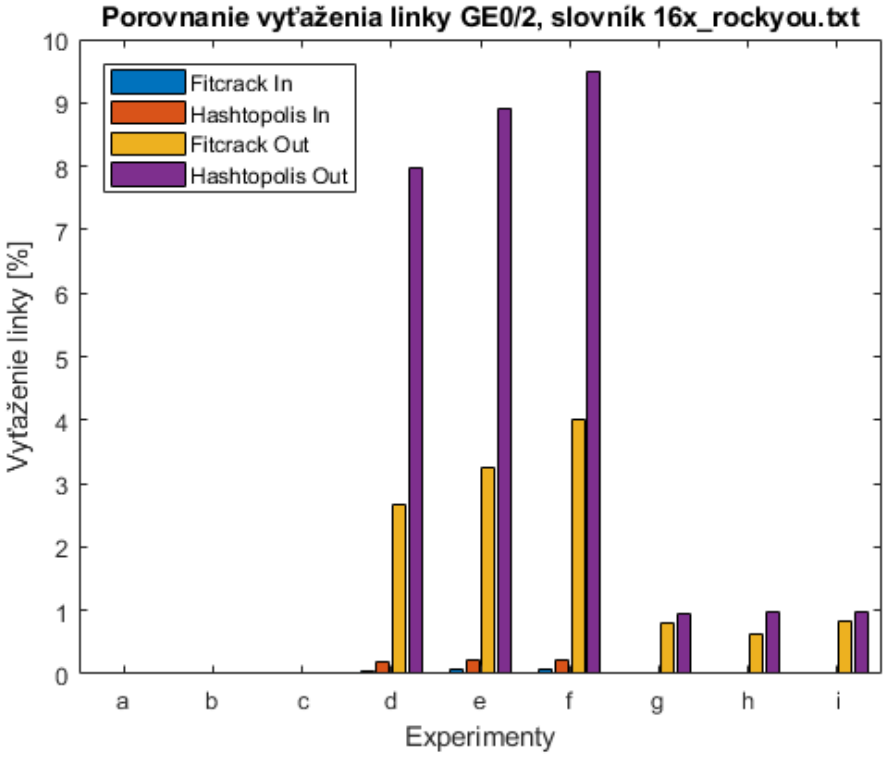
Obr. 5.5: Porovnanie dĺžky lámania hesiel.



Obr. 5.6: Porovnanie efektivity.



Obr. 5.7: Porovnanie vyťaženia linky GE0/1.

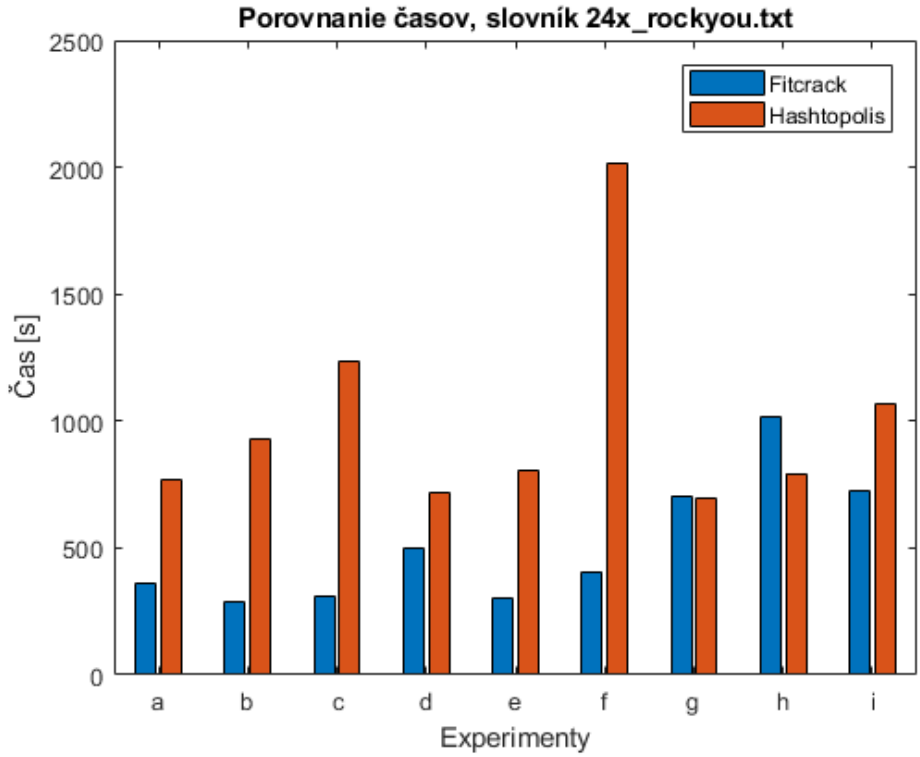


Obr. 5.8: Porovnanie vyťaženia linky GE0/2.

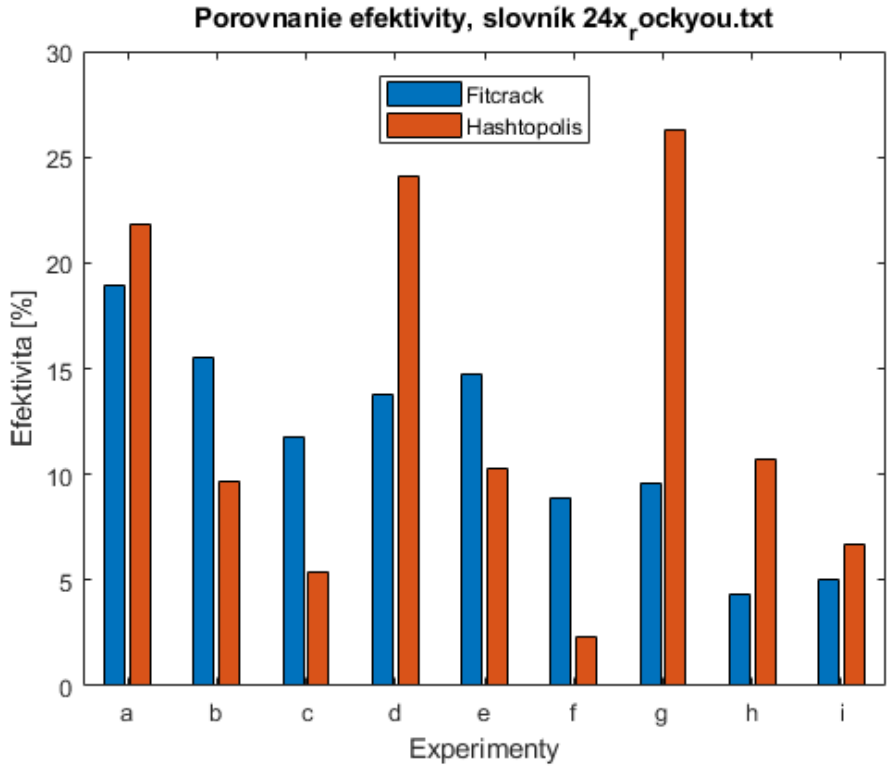
24x_rockyou.txt

Fitcrack: slovník 24x_rockyou.txt (4.4 GB, $456 \cdot 10^6$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	4/4	-	-	0.062	7.63	-	-	04:33	18.9
b	8	6/8	-	-	0.080	9.691	-	-	05:54	15.5
c	12	6/12	-	-	0.067	8.881	-	-	07:22	11.8
d	4	1/2	2/2	-	0.021	1.949	0.076	3.621	04:33	13.8
e	8	4/4	2/4	-	0.062	5.963	0.067	3.212	05:54	14.7
f	12	3/6	3/6	-	0.022	2.378	0.095	4.409	07:15	8.9
g	4	1/1	2/2	1/1	0.016	1.857	0.015	0.688	04:32	9.6
h	8	1/2	3/4	2/2	0.07	0.476	0.018	0.81	05:54	4.3
i	12	2/3	3/6	1/3	0.013	1.138	0.015	0.67	07:15	5
Hashtopolis: slovník 24x_rockyou.txt (4,4 GB, $456 \cdot 10^6$ hesiel)										
		uzly na lince (využ./celk.)			vyťaženie linek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	4/4	-	-	0.141	16.211	-	-	11:11	21.8
b	8	8/8	-	-	0.382	24.055	-	-	12:01	9.7
c	12	12/12	-	-	0.492	27.119	-	-	13:12	5.4
d	4	2/2	2/2	-	0.118	7.767	0.176	7.767	11:31	24.1
e	8	4/4	0/4	-	0.22	13.835	0.225	9.536	11:01	10.3
f	12	3/6	3/6	-	0.138	7.448	0.116	4.757	09:05	2.3
g	4	1/1	2/2	0/1	0.06	4.007	0.023	0.956	12:12	26.3
h	8	2/2	4/4	0/2	0.112	7.014	0.023	0.964	11:16	10.7
i	12	3/3	6/6	0/3	0.135	7.828	0.024	0.979	14:24	6.7

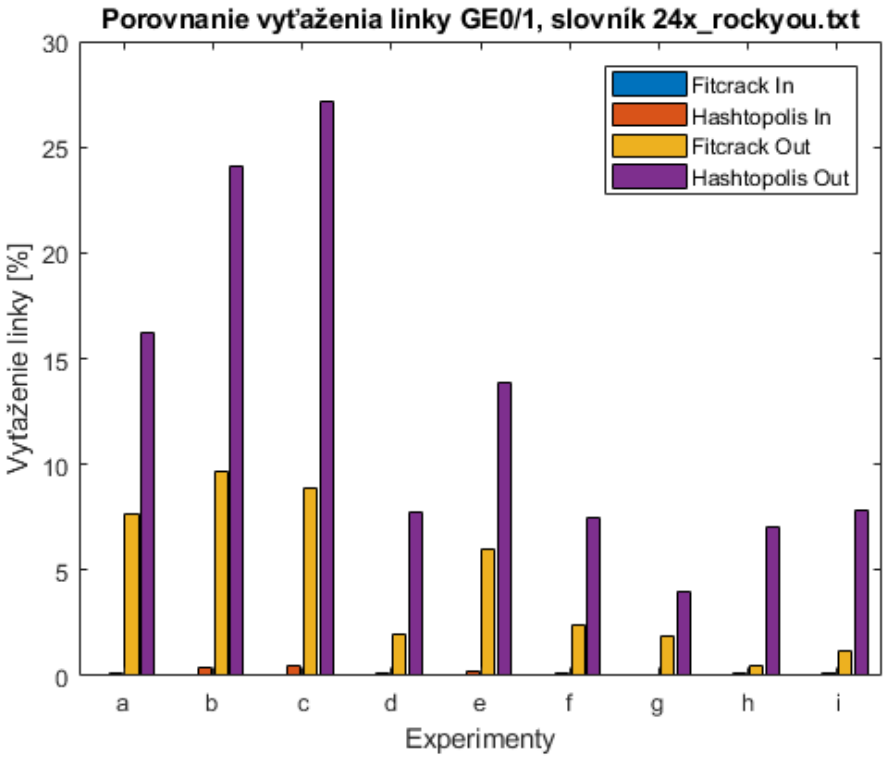
Tabuľka 5.3: Výsledky slovníkového útoku pri použití 24x_rockyou.txt.



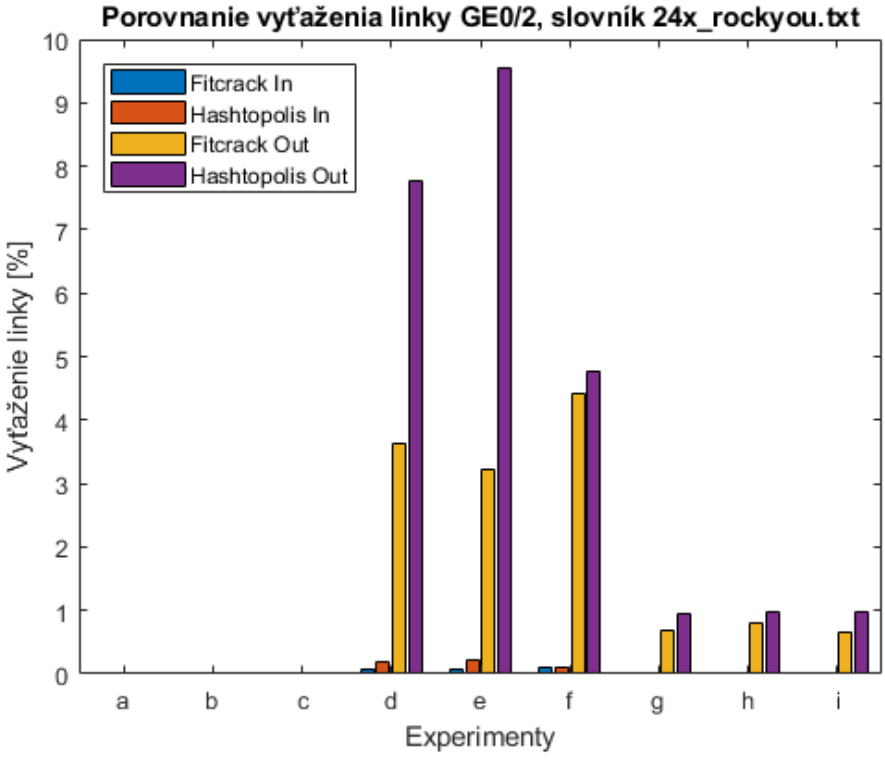
Obr. 5.9: Porovnanie dĺžky lámania hesiel.



Obr. 5.10: Porovnanie efektivity.



Obr. 5.11: Porovnanie vyťaženia linky GE0/1.



Obr. 5.12: Porovnanie vyťaženia linky GE0/2.

5.2.2 Útok hrubou silou

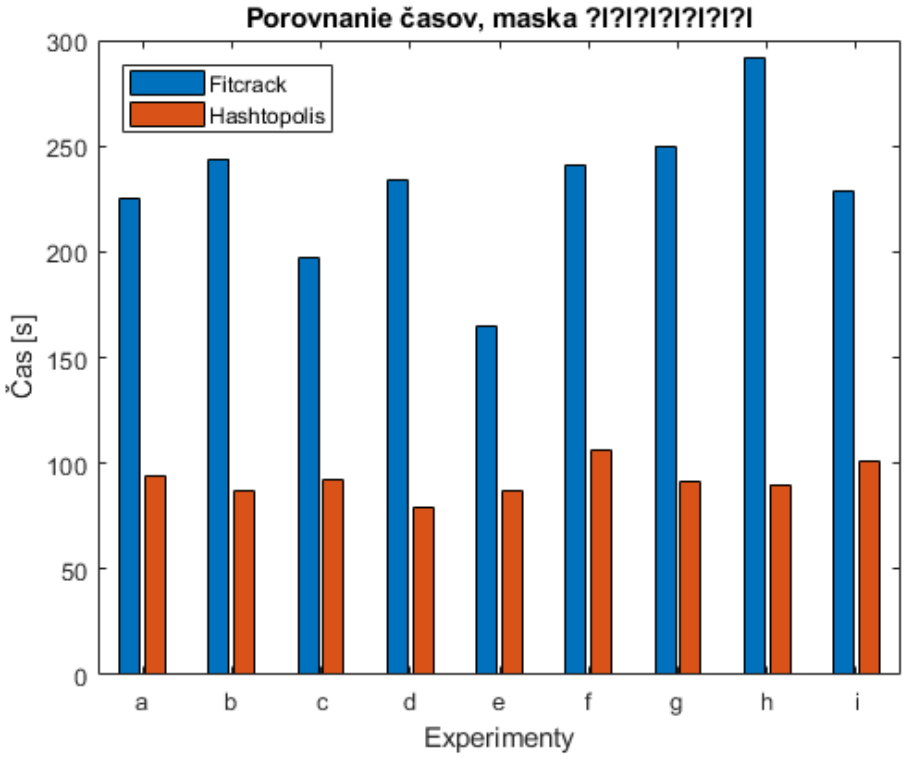
Z výsledkov efektivity uvedených v tabuľke 5.4, ktoré sú reprezentované pomocou grafu 5.14 je vidieť, že nástroj Fitcrack dosiahol lepších výsledkov pri distribuovanom útoku hrubou silou, v ktorom figurovala sedem znaková maska (?l?l?l?l?l?l?l).

Na druhú stranu z výsledkov dĺžky lámania hesiel, ktoré sú reprezentované pomocou grafu 5.13 sa ukázalo, že nástroj Hashtopolis bol rýchlejším.

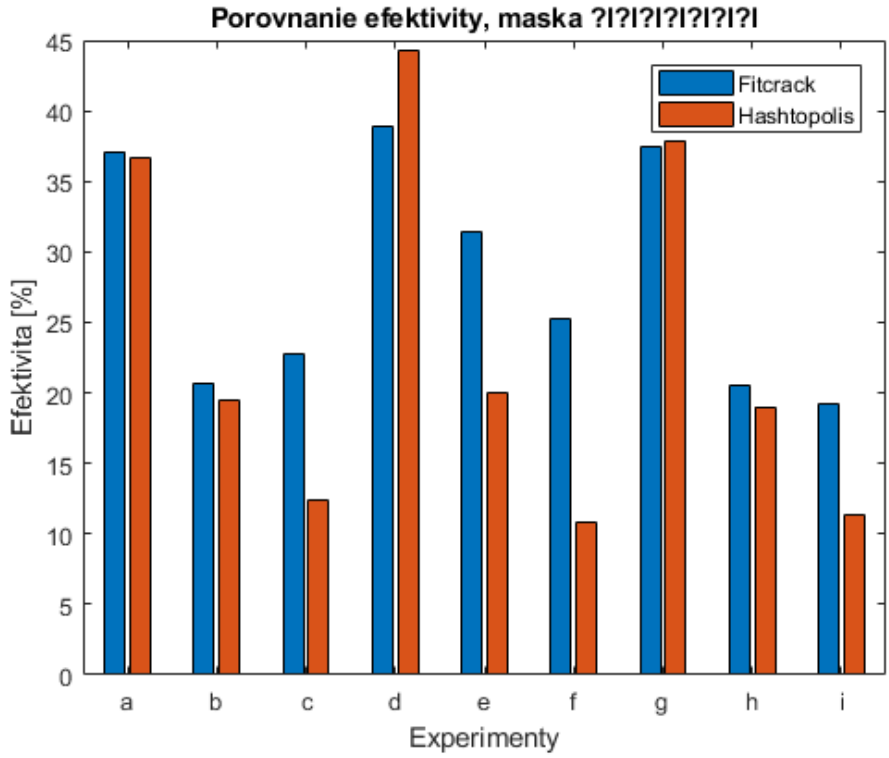
Posledným meraným kritériom bolo vyťaženie linky. Porovnanie hodnôt tohoto kritéria je uvedené pomocou grafov 5.15, 5.16. Na základe zobrazených výsledkov je vidieť, že nástroj Fitcrack pri vykonávaní distribuovaného útoku hrubou silou spôsobuje väčšie vyťaženie linky ako nástroj Hashtopolis.

Fitcrack: maska ?l?l?l?l?l?l?l ($8 \cdot 10^9$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	4/4	-	-	0.008	0.002	-	-	05:34	37.1
b	8	4/8	-	-	0.008	0.002	-	-	06:45	20.7
c	12	6/12	-	-	0.009	0.007	-	-	08:56	22.7
d	4	2/2	2/2	-	0.004	0.001	0.004	0.001	06:04	38.9
e	8	3/4	3/4	-	0.005	0.001	0.004	0.001	06:55	31.4
f	12	3/6	1/6	-	0.004	0.001	0.004	0.001	12:10	25.2
g	4	1/1	2/2	1/1	0.002	0.001	0.002	0.001	06:14	37.4
h	8	0/2	4/4	0/2	0.003	0.003	0.002	0.001	07:58	20.5
i	12	0/3	3/6	1/3	0.003	0.001	0.002	0.001	08:47	19.2
Hashtopolis: maska ?l?l?l?l?l?l?l ($8 \cdot 10^9$ hesiel)										
		uzly na linke (využ./celk.)			vyťaženie liniek [%]				čas	E_{ff}
ID	uz.	1 Gb/s	100 Mb/s	10 Mb/s	Ge_1^{in}	Ge_1^{out}	Ge_2^{in}	Ge_2^{out}	[m:s]	[%]
a	4	1/4	-	-	0.001	0.002	-	-	02:18	36.7
b	8	1/8	-	-	0.001	0.002	-	-	02:16	19.5
c	12	1/12	-	-	0.002	0.002	-	-	02:17	12.4
d	4	1/2	1/2	-	0.001	0.001	0.001	0.001	02:20	44.3
e	8	1/4	1/4	-	0.001	0.001	0.001	0.001	02:19	20
f	12	1/6	1/6	-	0.001	0.001	0.001	0.001	02:17	10.8
g	4	1/1	1/2	0/1	0	0.001	0	0	02:18	37.9
h	8	0/2	2/4	0/2	0	0	0	0	02:17	19
i	12	0/3	2/6	0/3	0	0	0	0	02:17	11.3

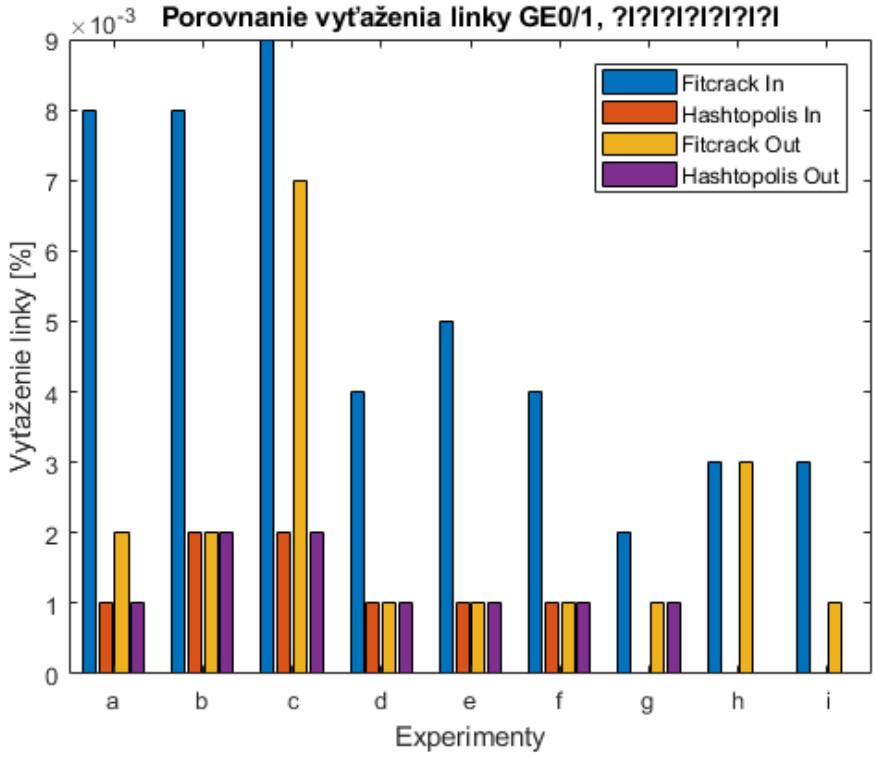
Tabuľka 5.4: Výsledky útoku hrubou silou pri použití masky ?l?l?l?l?l?l?l.



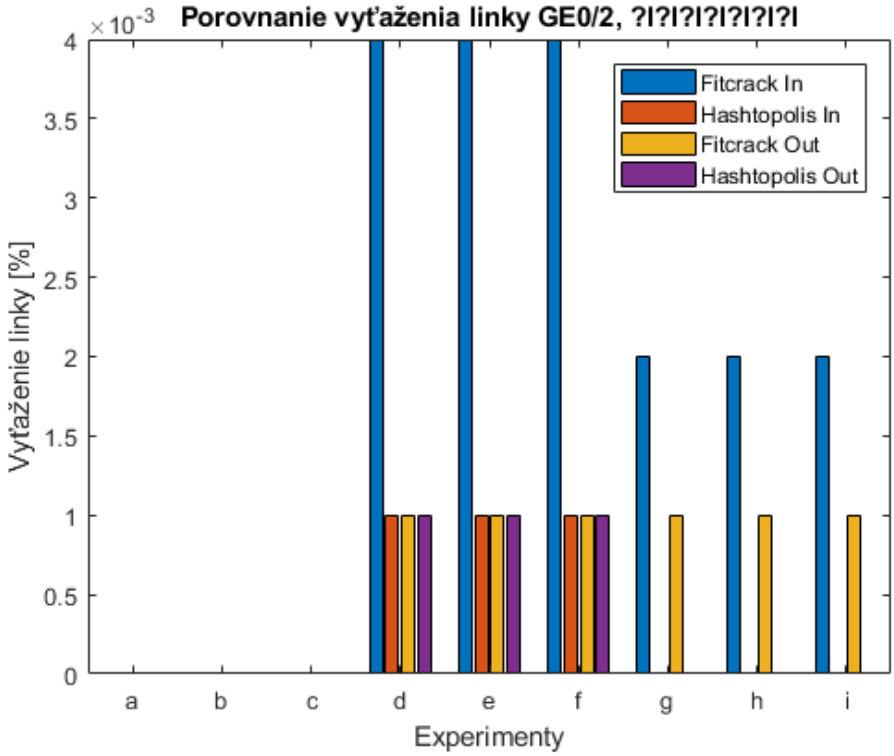
Obr. 5.13: Porovnanie dĺžky lámania hesiel.



Obr. 5.14: Porovnanie efektivity.



Obr. 5.15: Porovnanie vyťaženia linky GE0/1.



Obr. 5.16: Porovnanie vyťaženia linky GE0/2.

5.3 Zhodnotenie experimentov

V tejto časti kapituly zhodnotím namerané výsledky a popíšem vykonávanie experimentov z hľadiska použitých nástrojov.

5.3.1 Nástroj Fitcrack

Na základe výsledkov zo všetkých experimentov s distribuovaným slovníkovým útokom môžem usúdiť, že nástroj Fitcrack preukázal lepší výkon ako nástroj Hashtopolis z hľadiska rýchlosti lámania hesiel, škálovateľnosti a vyťaženia linky. Počas vykonávania experimentov nedochádzalo k žiadnym komplikáciám v rámci distribúcie výpočtov. Tento nástroj síce nezapojil do výpočtov všetky pridelené uzly, ale dokázal distribuovať úlohy rovnomerne medzi všetky siete.

Pri experimentoch na topológii 4.7, v ktorej je rýchlosť všetkých uzlov nastavená na 1 Gb/s (reálna nameraná rýchlosť je 280 Mb/s) bola distribúcia výpočtov prevedená bez zdržania. Pri tejto topológii bol lepším nástrojom z hľadiska rýchlosti lámania hesiel nástroj Fitcrack. Na druhú stranu sa ukázalo, že z pohľadu efektívnosti bol lepším nástrojom Hashtopolis. Ďalej pri topológii 4.8, v ktorej boli dve linky s rozdielnou rýchlosťou 1 Gb/s a 100 Mb/s nástroj Fitcrack dokončil prácu na uzloch s rýchlejším pripojením skôr a čakal na dokončenie výpočtov prevádzaných uzlami s nižšou rýchlosťou pripojenia. Dĺžka čakania záležala na veľkosti použitého slovníka a rýchlosti akou bol stiahnutý na výpočtových uzloch. Zo získaných dát je vidieť, že nástroj Fitcrack dosiahol lepších výsledkov v rýchlosti lámania hesiel. Pri experimentoch vykonaných na poslednej topológii 4.9, v ktorej sa nachádzajú štyri linky s troma rôznymi rýchlosťami 1 Gb/s, 100 Mb/s, 10 Mb/s (reálne rýchlosti 280 Mb/s, 98 Mb/s, 9.5 Mb/s) bola rýchlosť lámania hesiel najviac ovplyvnená najpomalšou linkou. Navzdory tomu sa ukázalo, že aj tu bol lepším práve nástroj Fitcrack.

Pri distribuovanom útoku hrubou silou nástroj Fitcrack preukázal horšie výsledky ako nástroj Hashtopolis z hľadiska rýchlosti lámania hesiel, vyťaženia linky ale lepšie výsledky z pohľadu efektívnosti a distribúcie úlohy. Rôzne topológie pri prevedení útoku hrubou silou mali minimálny vplyv na hodnotiace kritéria.

5.3.2 Nástroj Hashtopolis

V rámci riešenia distribuovaného slovníkového útoku sa ukázalo, že nástroj Hashtopolis preukázal horší výkon ako nástroj Fitcrack z hľadiska rýchlosti lámania hesiel, škálovateľnosti a vyťaženia linky. Na druhú stranu, bol efektívnejší a zapojil do výpočtov všetky priradené uzly čím využil možnosti distribúcie naplno. Počas experimentovania s nástrojom Hashtopolis dochádzalo k častým výpadkom alebo zaseknutiu uzlov, počas sťahovania slovníkov. Keď táto situácia nastala, bolo nutné reštartovať klienta, ktorý následne začal sťahovanie celého slovníku od začiatku.

Pri experimentoch na prvej topológii 4.7, v ktorej je rýchlosť linky pre všetky uzly nastavená na 1 Gb/s (reálna nameraná rýchlosť je 280 Mb/s), bola distribúcia výpočtov podobne ako u nástroja Fitcrack prevedená bez závažnejších problémov. Následne pri topológii 4.8, v ktorej boli dve linky s rozdielnou rýchlosťou 1Gb/s a 100 Mb/s sa ukázalo, že sťahovanie celého slovníka na všetkých uzloch vie proces lámania výrazne predĺžiť. Tento proces bol najprv dokončený na uzloch s rýchlejším pripojením. Po chvíli sa stiahli slovníky aj na uzly pripojené pomocou pomalšej linky. Z hľadiska vyťaženia linky bol nástroj Hashtopolis oveľa viac aktívnejší. U poslednej z topológii 4.9, v ktorej sa nachádzajú štyri linky s troma rôznymi rýchlosťami 1 Gb/s, 100 Mb/s, 10 Mb/s (reálne rýchlosti 280 Mb/s, 98 Mb/s, 9.5

Mb/s). Bol vplyv rôznej rýchlosti ešte výraznejší. Sťahovanie začalo na všetkých uzloch. Najprv sa dokončilo sťahovanie na uzloch s najrýchlejšou linkou. Následne sa dokončilo sťahovanie na linkách s rýchlosťou 100 Mb/s rovnako ako v predchádzajúcej topológii. Nástroj Hashtopolis začal distribuovať úlohy medzi uzly, ktoré dokončili sťahovanie. Celá úloha bola vyriešená skôr než boli stiahnuté všetky slovníky na najpomalších linkách 10 Mb/s. Z tohoto dôvodu je tento nástroj neefektívnym pri distribuovanom slovníkovom útoku. Navyše uzly pripojené pomocou najpomalšej linky pokračujú v sťahovaní aj po dokončení celej úlohy. Pre prerušenie tohoto zbytočného sťahovania bolo nutné ručne reštartovať vybrané uzly.

Pri distribuovanom útoku hrubou silou nástroj Hashtopolis preukázal lepšie výsledky ako nástroj Fitcrack z hľadiska efektívnosti, rýchlosti lámania hesiel, vyťaženia linky. Rôzne topológie pri prevedení útoku hrubou silou mali minimálny vplyv na hodnotiace kritéria.

Kapitola 6

Záver

Cieľom tejto práce bolo analyzovať vplyv sieťovej infraštruktúry na distribuované lámanie hesiel pomocou nástrojov Fitcrack a Hashtopolis. Na začiatku bolo nutné definovať hodnotiace kritéria, na základe ktorých som mohol porovnať použité nástroje. Ďalším krokom bolo implementovať skripty, ktoré mi pomohli s vytváraním, spustením úloh a získavaním hodnôt potrebných k porovnaniu. Následne bolo potrebné vytvoriť sadu testovacích úloh a sieťové topológie, na ktorých boli vykonané všetky experimenty.

Na základe výsledkov získaných z experimentov so slovníkovým útokom sa ukázalo, že nástroj Fitcrack dosiahol lepších výsledkov z pohľadu rýchlosti lámania hesiel, škálovateľnosti a vyťaženia linky. Na druhú stranu pri distribuovanom útoku hrubou silou bol lepší nástroj Hashtopolis podľa všetkých hodnotiacich kritérií. Pri analýze vplyvu sieťovej infraštruktúry na distribuované lámanie hesiel sa ukázalo, že pri distribuovanom slovníkovom útoku je vplyv výpočtových uzlov, ktorých rýchlosť pripojenia je 100Mb/s a 10 Mb/s viditeľný. Zdržanie, ktoré z tohoto hľadiska vzniklo ovplyvnilo oba nástroje. Výraznejší vplyv bol vidieť pri nástroji Hashtopolis. Pri útoku hrubou silou sa rozdiely v rýchlosti dajú považovať za zanedbateľné.

V rámci distribuovania slovníkového útoku nedoporučujem použitie nástroja Hashtopolis ani distribúciu v rámci sietí, v ktorých je rýchlosť pripojenia nižšia ako 100 Mb/s. Tieto siete výrazne ovplyvňujú celkový čas lámania hesiel. V prípade, že by sa zlepšila distribúcia úloh pomocou nástroja Fitcrack, to znamená, že daná úloha by bola rozdelená medzi všetky uzly zapojené do výpočtov, tento nástroj by mohol dosiahnuť oveľa lepších výsledkov.

Pokračovaním tejto práce by mala byť analýza distribuovaného lámania hesiel pomocou liniek s rýchlosťou 10 Gb/s. Doplnenie porovnania ďalších spôsobov lámania hesiel a implementácia monitorovania počtu prenesených dát na nástroji Fitcrack.

Kapitola 7

Prílohy

7.1 Hodnoty SNMP

Slovníkový útok–8x_rockyou.txt

DA	Fitcrack			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	12862704	1151130888	——	——
b	12351305	1151101137	——	——
c	14751125	1151245311	——	——
d	6285761	547852341	12942076	603231775
e	6195032	603252897	11822398	547836268
f	10351533	1150822680	895383	221728
g	823362	201245	602814	154580
h	627303	153541	484060	134588
i	6662626	603144418	464404	124228

Tabuľka 7.1: Porovnanie SNMP hodnôt nástroja Fitcrack.

DA	Hashtopolis			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	57261102	3961187197	——	——
b	146529806	9281120811	——	——
c	249812531	13921366489	——	——
d	35398789	2320448794	52429513	2320339000
e	72797578	4640897588	74859026	2999483133
f	115005668	6960733552	94541578	3878627266
g	17885205	1160212385	5233908	219170917
h	36963677	2320334262	7804774	319926302
i	59972158	2243981732	10122985	411884227

Tabuľka 7.2: Porovnanie SNMP hodnôt nástroja Hashtopolis.

Slovníkový útok–16x_rockyou.txt

DA	Fitcrack			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	21642386	3969394079	——	——
b	23396483	2301744324	——	——
c	21998826	2301740677	——	——
d	10448181	1092597913	25480051	1209403646
e	10778631	1089376594	25803494	1212448551
f	7471805	486408941	38338571	1815512473
g	10372623	487023157	45638222	1235112339
h	7394555	606571862	10905135	486366930
i	7752005	606957817	25951235	1090359963

Tabuľka 7.3: Porovnanie SNMP hodnôt nástroja Fitcrack.

DA	Hashtopolis			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	134405865	9281493291	——	——
b	295967672	18562171268	——	——
c	504859862	23547794640	——	——
d	69539602	4640896835	104974503	4640687762
e	73869732	4640551496	68415805	2898111981
f	230297643	13921390817	191589030	7849927041
g	35005500	2320419795	11907729	506375513
h	74053799	4640651455	15316078	637448518
i	120024575	6960814739	20460997	832136952

Tabuľka 7.4: Porovnanie SNMP hodnôt nástroja Hashtopolis.

Slovníkový útok–24x_rockyou.txt

DA	Fitcrack			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	28165586	3452479278	——	——
b	28595122	3452563903	——	——
c	26091602	3452528930	——	——
d	12902509	1208184772	46887099	2244804393
e	23229188	2243624880	25318872	1208669659
f	11222870	1209627279	48389421	2242965978
g	14455105	1636285505	13429272	606356572
h	8275250	606137210	23106919	1030151809
i	11530087	1030151538	13449213	606356935

Tabuľka 7.5: Porovnanie SNMP hodnôt nástroja Fitcrack.

DA	Hashtopolis			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	135506753	15623145680	——	——
b	442563848	27843230776	——	——
c	756913950	41763890208	——	——
d	105423397	6961301211	157255202	6961048365
e	220934526	13921733888	226721963	9595069213
f	347614309	18787489034	293330266	12000093873
g	52422968	3480702693	19676469	830139467
h	110683838	6960984808	22940730	957150891
i	179549250	10441224870	32097082	1305371836

Tabuľka 7.6: Porovnanie SNMP hodnôt nástroja Hashtopolis.

Útok hrubou silou

BA	Fitcrack			
Rozhranie	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	2335141	586545	——	——
b	2499164	704379	——	——
c	2091471	1678758	——	——
d	1217227	315897	1094346	331587
e	954594	236026	823135	201104
f	1319098	325792	1112559	339090
g	734109	179837	576426	146254
h	896347	1119851	637928	164737
i	701952	186989	551562	171945

Tabuľka 7.7: Porovnanie SNMP hodnôt nástroja Fitcrack.

BA	Hashtopolis			
H_R	GE 0/1		GE 0/2	
ID	IfInOctets	IfOutOctets	IfInOctets	IfOutOctets
a	155913	174568	——	——
b	162292	181241	——	——
c	181509	203891	——	——
d	79747	90025	74466	83391
e	91416	99269	87688	102350
f	120218	135643	119844	134674
g	44139	53554	42573	48993
h	41813	49226	44361	44754
i	44419	51295	42926	50156

Tabuľka 7.8: Porovnanie rýchlosti lámania hešov nástrojov Fitcrack a Hashtopolis.

7.2 Obsah pamäťového média

Priložené CD obsahuje:

- túto prácu vo formáte PDF,
- zdrojové kódy tejto práce,
- zdrojové kódy Matlab skriptov,
- zdrojové kódy Python skriptov vytvorených pre automatizáciu,
- excel súbory obsahujúce získané dáta,
- obrázky grafov použitých v tejto práci.

Literatúra

[1] Aboba, B.; Cheshire, S.: Dynamic Host Configuration Protocol (DHCP) Domain Search Option. Technická správa, 2002.

[2] Anderson, D. P.: BOINC: a system for public-resource computing and storage. In *Fifth IEEE/ACM International Workshop on Grid Computing*, Nov 2004, ISSN 1550-5510, s. 4–10, doi:10.1109/GRID.2004.14.

[3] Bray, T.: The javascript object notation (json) data interchange format. 2014.

[4] Carrier, B.; aj.: Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of digital evidence*, ročník 1, č. 4, 2003: s. 1–12.

[5] Eastlake 3rd, D.: Additional XML Security Uniform Resource Identifiers (URIs).

[6] Fielding, R.; Gettys, J.; Mogul, J.; aj.: RFC 2616: Hypertext transfer protocol–HTTP/1.1. 1999.

[7] Hornig, C.: RFC 894: Standard for the transmission of IP datagrams over Ethernet networks. Technická správa, RFC, IETF, April, 1984.

[8] Hranický, R.; Holkovič, M.; Matoušek, P.: On Efficiency of Distributed Password Recovery. *Journal of Digital Forensics, Security and Law*, ročník 11, č. 2, 2016: str. 5.

[9] Hranický, R.; Zobal, L.; Večeřa, V.: Distribuovaná obnova hesel. Technická správa, 2017.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=11568

[10] Hranický, R.; Zobal, L.; Večeřa, V.; aj.: Distribuce výpočtů pro nástroj hashcat. Technická správa, 2018.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=11884

[11] Hranickỳ, R.; Zobal, L.; Večera, V.; aj.: The architecture of Fitcrack.

[12] Jones, P. E.; aj.: US secure hash algorithm 1 (SHA1). 2001.

[13] Kumar, H.; Kumar, S.; Joseph, R.; aj.: Rainbow table to crack password using MD5 hashing algorithm. In *2013 IEEE Conference on Information Communication Technologies*, April 2013, s. 433–439, doi:10.1109/CICT.2013.6558135.

[14] Luebke, D.; Harris, M.; Govindaraju, N.; aj.: GPGPU: general-purpose computation on graphics hardware. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, ACM, 2006, str. 208.

[15] Ma, W.; Campbell, J.; Tran, D.; aj.: Password Entropy and Password Quality. In *2010 Fourth International Conference on Network and System Security*, Sep. 2010, s. 583–587, doi:10.1109/NSS.2010.18.

[16] Marechal, S.: Advances in password cracking. *Journal in Computer Virology*, ročník 4, č. 1, Feb 2008: s. 73–81, ISSN 1772-9904, doi:10.1007/s11416-007-0064-y. URL https://doi.org/10.1007/s11416-007-0064-y

[17] Múčka, M.: Webová aplikace pro vzdálenou správu systému Fitcrack. 2018. URL http://www.fit.vutbr.cz/study/DP/BP.php?id=21121

[18] Pummill, T. T.: Variable Length Subnet Table For IPv4. 1995.

[19] Rivest, R. L.; aj.: RFC 1321: The md5 message-digest algorithm. 1992.

[20] Samek, J.: Virtuální GPU cluster. 2016.

[21] Stone, J. E.; Gohara, D.; Shi, G.: OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, ročník 12, č. 3, 2010: str. 66.

[22] Van Rossum, G.; aj.: Python Programming Language. In *USENIX annual technical conference*, ročník 41, 2007, str. 36.

[23] Wadi, S. M.; Zainal, N.: High Definition Image Encryption Algorithm Based on AES Modification. *Wireless Personal Communications*, ročník 79, č. 2, Nov 2014: s. 811–829, ISSN 1572-834X, doi:10.1007/s11277-014-1888-7. URL https://doi.org/10.1007/s11277-014-1888-7