VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APROXIMACE DOBY VÝPOČTU DISTRIBUOVANÝCH ÚLOH PRO LÁMÁNÍ HESEL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUCIE JADRNÁ**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

153884

Ústav: Ústav informačních systémů (UIFS)
Studentka: **Jadrná Lucie**
Program: Informační technologie
Název: **Aproximace doby výpočtu distribuovaných úloh pro lámání hesel**
Kategorie: Bezpečnost
Akademický rok: 2023/24

Zadání:

1. Seznamte se s architekturou a implementací systému Fitcrack. Nastudujte implementované metody útoků, techniky měření výkonu (benchmarking) výpočetních uzlů a metodiku aproximace doby výpočtu.
2. Identifikujte kritická místa, která ovlivňují přesnost odhadu doby výpočtu.
3. Po konzultaci s vedoucím navrhněte úpravy systému, které umožní přesněji určit maximální dobu výpočtu při zadané konfiguraci úlohy.
4. Navržené úpravy implementujte.
5. Experimentálně ověřte přínos vašich úprav a zhodnoťte dosažené výsledky.

Literatura:

- HRANICKÝ Radek. Digital Forensics: The Acceleration of Password Cracking. *Disertační práce*. Fakulta informačních technologií VUT v Brně, 2022.
- HRANICKÝ Radek, ZOBAL Lukáš, VEČEŘA Vojtěch, MÚČKA Matúš, HORÁK Adam, BOLVANSKÝ Dávid a ŽENČÁK Tomáš. The architecture of Fitcrack distributed password cracking system, version 2. Technická zpráva, FIT-TR-2020-04, Brno: Fakulta informačních technologií VUT v Brně, 2020.
- HRANICKÝ Radek, ZOBAL Lukáš, RYŠAVÝ Ondřej and KOLÁŘ Dušan. Distributed Password Cracking with BOINC and Hashcat. *Digital Investigation*, roč. 2019, č. 30, s. 161-172. ISSN 1742-2876.

Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Hranický Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 30.10.2023

Abstrakt

Tato práce se zaměřuje na zlepšení přesnosti odhadu doby lámání hesel v distribuovaném systému Fitcrack. Na základě analýzy stávajícího řešení byly identifikovány klíčové faktory ovlivňující přesnost výpočtu a navržena sada algoritmů pro zpřesnění odhadu. Provedené experimenty potvrdily účinnost navržených algoritmů, přičemž nové řešení přineslo významné zlepšení přesnosti odhadu doby lámání v 90.32 % měřených úloh, zahrnujících útok hrubou silou, slovníkový útok, kombinační útok a hybridní útoky.

Abstract

This thesis focuses on improving the accuracy of password cracking time estimation in the distributed system Fitcrack. Based on the analysis of the current solution, key factors influencing the accuracy of computation were identified, and a set of algorithms was proposed to refine the estimation. Conducted experiments confirmed the effectiveness of the proposed algorithms, with the new solution bringing significant improvement in the accuracy of cracking time estimation in 90.32 % of measured tasks, including brute-force attack, dictionary attack, combination attack and hybrid attacks.

Klíčová slova

Fitcrack, lámání hesel, aproximace doby lámání hesla

Keywords

Fitcrack, password cracking, cracking time aproximation

Citace

JADRNÁ, Lucie. *Aproximace doby výpočtu distribuovaných úloh pro lámání hesel*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický, Ph.D.

Aproximace doby výpočtu distribuovaných úloh pro lámání hesel

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Radka Hranického, Ph.D. Uvedla jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpala.

.....
Lucie Jadrná
6. května 2024

Poděkování

Ráda bych poděkovala Ing. Radku Hranickému, Ph.D. za vedení mé práce a cenné rady.

Obsah

Seznam obrázků

Kapitola 1

Úvod

S narůstajícím množstvím dat a závislostí na digitálních technologiích se zvyšuje riziko přístupu k neoprávněným informacím a citlivým osobním údajům. Jak jednotlivci tak organizace chrání citlivá data pomocí hesel a jejich prolomení může vést k vážným následkům, jako jsou například finanční ztráty, ztráty duševního vlastnictví nebo odcizení identity a zneužití osobních údajů.

Lámání hesel má většinou negativní podtext spojený s kriminální činností, ale nemusí tomu tak být vždy. Lámání hesel může napomoct při řešení případů, kdy forenzní kriminalisté potřebují získat přístup k zabezpečeným datům, které mohou být nápomocné pro vyřešení případu. Při řešení kriminálních případů může hrát hlavní roli čas, za který se forenzním kriminalistům podaří prolomit heslo a získat tak zabezpečená data. Příkladem může být situace, kdy kriminalisté zadrží osobu podezřelou z plánování útoku na obytnou zónu, které bylo zabaveno zašifrované elektronické zařízení s možnými informacemi o útoku. V takovém případě je užitečné vědět, jak dlouho by lámání hesla trvalo. Pokud by lámání hesla zabralo delší dobu, než za jakou by měl proběhnout útok, kriminalisté by mohli změnit postup řešení případu.

Systém Fitcrack poskytuje uživateli odhad doby lámání před spuštěním úlohy, avšak tato odhadovaná doba se liší od skutečné. Tento rozdíl je způsoben především časem potřebným pro komunikaci mezi serverem a výpočetními uzly, který není v současném odhadu zahrnut. Nepřesný odhad také způsobuje konfigurace úlohy s méně kandidátními hesly a více výkonnými uzly. Je třeba poznamenat, že ve skutečnosti se na úloze nebudou podílet všechny připojené uzly, ale pouze ty s nejvyšším výkonem. Současný vzorec odhadu předpokládá, že se na úloze podílejí všechny připojené uzly.

Přínosem této práce je vylepšení přesnosti odhadu doby lámání hesel prostřednictvím nově navržených algoritmů pro odhad uzlů, které se skutečně budou podílet na lámání hesel. Kromě toho jsem do výpočtu odhadované doby lámání začlenila čas nutný pro komunikaci mezi serverem a výpočetními uzly, což vede k výpočtu přesnějšího odhadu. Výsledky experimentů potvrdily, že nový přístup významně zlepšuje přesnost odhadované doby lámání hesel v porovnání se současným stavem. Konkrétně jsem dosáhla zlepšení přesnosti odhadu v 90.32 % při použití nového řešení.

Hesla mohou být lámána na jednom zařízení pomocí centrální procesorové jednotky (CPU) nebo grafického procesoru (GPU). Pro zvýšení efektivity a rychlosti lámání je možné distribuovat výpočet mezi více lámacích uzlů. Tento přístup implementuje systém Fitcrack, který využívá platformy BOINC pro připojení jednoho nebo více zařízení, které následně provádí lámání hesla pomocí nástroje Hashcat. Systému Fitcrack a jeho komponentám

se věnuje kapitola 2. V této kapitole je také popsána platforma BOINC a nástroj pro lámání hesel Hashcat.

Hesla lze lámat pomocí různých přístupů, které se liší tím, jak se vytváří kandidátní hesla, tj. možné kombinace znaků, které mohou tvořit hledané heslo. Tyto přístupy jsou blíže popsané v kapitole 3. Tato kapitola se také zabývá distribucí úlohy mezi jednotlivé výpočetní uzly. Kritickým místům, která ovlivňují výpočet odhadované doby lámání se věnuje kapitola 4. Na základě analýzy hodnot naměřených v kapitole 4, je následně v kapitole 5 popsán návrh změn v systému Fitcrak, které vedou k optimalizaci výpočtu odhadovaného času lámání. V kapitole 6 je popsána implementace návrhu a kapitola 7 je věnována experimentálnímu ověření přínosu provedených úprav a zhodnocení dosažených výsledků.

Kapitola 2

Systém Fitcrack

Fitcrack vznikl v rámci výzkumné skupiny NES@FIT na Fakultě Informačních Technologií VUT v Brně[1] a je volně dostupný pod licencí MIT na platformě GitHub[2]. Jedná se o distribuovaný systém, sloužící k obnově (lámání) hesel z hešů nebo podporovaných typů šifrovaných médií, jako jsou dokumenty, archivy nebo diskové svazky.

Architektura Fitcracku je složena ze serveru, který řídí proces lámání a jednoho nebo více klientských uzlů, na kterých lámání probíhá. Architektura systému je popsána v sekci 2.3. Maximální počet připojených uzlů není nijak omezený a každý klientský uzel může používat jedno nebo více OpenCL zařízení, které mohou být jiného typu, výrobce nebo modelu .Tyto uzly jsou k serveru připojeny pomocí platformy BOINC, která je blíže popsána v sekci 2.1. Pro samotné lámání hesel využívá Fitcrack nástroj Hashcat, který je popsán v sekci 2.2.

2.1 BOINC

Berkeley Open Infrastructure for Network Computing (BOINC)[3] je platforma pro řešení distribuovaných úloh pomocí veřejných zdrojů. Jedná se o volně dostupný software vyvinutý v Laboratoři kosmických věd na Univerzitě v Berkeley v Kalifornii. BOINC umožňuje dobrovolníkům podpořit existující vědecké projekty poskytnutím výpočetních kapacit svého zařízení. Mezi projekty patří například Seti@home, který se zabýval zpracováním signálu radio teleskopických dat z observatoře Arecibo, Folding@home, který zkoumal skládání proteinů a s tím spojené nemoci nebo projekt Climateprediction.net, zabývající se zpřesněním dlouhodobých předpovědí klimatu [1].

Pro připojení k projektu musí být na zařízení (klientském uzlu) nainstalován BOINC *Client* a volitelně BOINC *Manager*, který poskytuje grafické uživatelské rozhraní pro BOINC Clienta. BOINC *Client* zajišťuje komunikaci se serverem přes plánovací protokol, využívající HTTP [3] nebo HTTPS [10]. Komunikace zahrnuje inicializaci projektu na klientském uzlu, získání jednotky práce zvané *workunit* a hlášení výsledků práce serveru. Po úspěšném připojení BOINC Clienta k serveru Fitcracku se na klientský uzel stáhnou binární soubory potřebné pro lámání hesel: Hashcat (viz sekce 2.2) a Runner (viz sekce 2.3) [9].

[1] https://www.fit.vut.cz/.cs
[2] https://github.com/nesfit/fitcrack
[3] https://boinc.berkeley.edu/

2.2 Hashcat

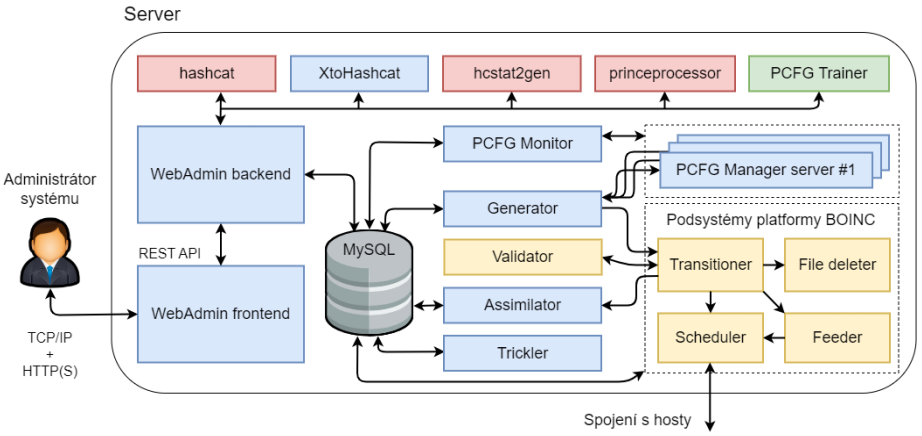
Hashcat[4] samozvaný „Světově nejrychlejší a nejpokročilejší nástroj pro obnovu hesel“ je otevřený software dostupný pod licencí MIT. Podporuje více než 300 typů hešovacích algoritmů a několik typů útoků, např. útok hrubou silou, slovníkový útok, kombinační útok a hybridní útoky. Pomocí Hashcatu lze lámat hesla na CPU podporujícím OpenCL, GPU a dokonce i na FPGA [8]. Tento nástroj podporuje lámání hesel pouze na jednom klientském uzlu, čili nepodporuje distribuované lámání hesel. Systém Fitcrack jej používá pro lámání hesel na jednotlivých uzlech.

2.3 Architektura systému

Architektura systému Fitcrack je postavena na bázi klient-server. V této sekci jsou popsané jednotlivé komponenty serverové a klientské části tohoto systému.

2.3.1 Serverová část

Server je zodpovědný za správu jednotlivých úloh pro lámání zvaných *job* a přidělovaní menších částí těchto úloh (*workunit*) klientským uzlům. *Job* je určena typem útoku (viz sekci 3.2), jedním nebo více heši stejného typu (např. MD5 [11], SHA-1 [5]) a dodatečnými informacemi o útoku (např. definování masky nebo slovníku) [6].



Obrázek 2.1: Architektura serverové části systému Fitcrack [9]

Server se skládá z několika podsystémů, které jsou znázorněny na obrázku 2.1. Části vybarvené modře znázorňují podsystémy Fitcracku, části vybarvené žlutě jsou podsystémy platformy BOINC a červeně vybarvené části patří k nástroji Hashcat.

Hlavními podsystémy, které se podílí na přidělování *workunit* jsou *Generator*, *Validator*, *Assimilator* a *Trickler*. Podsystémy pro komunikaci s klientskými uzly na základě platformy BOINC jsou *Transitioner*, *Scheduler*, *Feeder* a *File deleter*. Pro ukládání dat je použita MySQL[5] databáze a pro správu celého systému je implementováno uživatelské rozhraní *WebAdmin*. Všechny tyto podsystémy jsou popsány níže v této sekci.

[4] https://hashcat.net
[5] https://www.mysql.com/

Generator

Serverový démon zvaný *Generator* je zodpovědný za vytváření nových *workunit* pro jednotlivé připojené uzly, podle jejich zjištěného výkonu. *Workunit* může být typu *benchmark*, *complete benchmark* nebo úloha pro lámání hesla.

Benchmark se provede na začátku každé nové lámací úlohy a změří rychlost lámání pro daný typ heše. Pokud je klientský uzel připojený k serveru poprvé, provede se *complete benchmark*, který vypočítá dosažitelnou rychlost lámání daného uzlu pro všechny podporované typy hešů. Tyto výsledky jsou později použity pro výpočet odhadované doby lámání. *Generátor* komunikuje s ostatními podsystémy serveru pouze skrz databázi [9].

Validator

Pro ověření validity výsledků *workunit* získaných od klientských uzlů používá systém Fitcrack *Validator*[6], který je podsystém platformy BOINC . Kontroluje, jestli výsledek obsahuje všechny potřebné výstupní soubory a jestli mají správný formát. Pokud byla *workunit* replikována, *Validator* také ověřuje, jestli mají repliky stejný výsledek, pokud ano, jsou považovány za validní [9].

Assimilator

Serverový démon zvaný *Assimilator* je zodpovědný za zpracování výsledků *workunit*. Na základě výsledků může *Assimilator* modifikovat stav úlohy v databázi. Mohou nastat tři typy výsledku. Pokud se jedná o úspěšně provedený *benchmark*, *Assimilator* uloží výkon uzlu do databáze. Pokud se jedná o úspěšně dokončenou úlohu lámání, *Assimilator* aktualizuje stav celkové úlohy v databázi, pokud byla zvalidována všechna kandidátní hesla celková úloha je ukončena, pokud bylo prolomeno jedno nebo více hesel, je stav úlohy nastaven na *finished* jinak na *exhausted*. Pokud je výsledek výpočetní chyba, *Assimilator* nastaví spuštění procesu obnovy po selhání [9].

Trickler

Periodické získávání informací o současném stavu výpočtu *workunit* je realizováno podsystémem *Trickler* pomocí *Trickle message API*[7], které je součástí platformy BOINC. Díky tomuto podsystému server získá informace o průběhu lámání dříve než je úloha dokončena.

Transitioner

Zajištění synchronizace dat v databázi obstarává podsystém platformy BOINC zvaný *Transitioner*, na kterém jsou ostatní podsystémy závislé.

Feeder

Další z podsystémů platformy BOINC je *Feeder*. Vytváří segment sdílené paměti, který slouží pro předávání záznamů databáze podsystému *Scheduler*.

[6] https://boinc.berkeley.edu/trac/wiki/ValidationIntro
[7] https://boinc.berkeley.edu/trac/wiki/TrickleApi

File deleter

Podsystém platformy BOINC *File deleter* je zodpovědný za mazání vstupních a výstupních souborů *workunit*, které již byly zpracovány podsystémem *Assimilator*.

Scheduler

Zajištění komunikace s klientskými uzly obstarává *Scheduler*, který je podsystém platformy BOINC.

WebAdmin

Webová aplikace *WebAdmin* slouží pro vzdálenou správu systému Fitcrack. Je rozdělena na dvě části, *WebAdmin frontend* a *WebAdmin backend*, které spolu komunikují skrze REST API.

WebAdmin frontend je webová aplikace implementovaná pomocí javascriptového frameworku Vue.js[8]. Jedná se o uživatelsky přívětivý přístup ke správě většiny částí systému Fitcrack. *WebAdmin backend* je implementován pomocí mikro frameworku Flask[9]. Implementuje všechny potřebné endpointy využívané frontendem. *WebAdmin* využívá sadu externích nástrojů a programů, jako jsou například Hashcat, XtoHashcat, hcstats2gen nebo princeprocessor [9]. Tyto nástroje jsou blíže popsané v sekci 2.3.2.

Uživatel pomocí *WebAdmin* vytváří, spravuje a monitoruje nové lámací úlohy tzv. *jobs*. Při zakládání úlohy uživatel zadává jeden nebo více hešů a jejich typ, popřípadě soubor k prolomení, typ útoku a jeho nastavení, uzly, které budou provádět lámání a volitelně dodatečné nastavení (komentář, čas pro jednotlivé *workunit*). Uživatel si může zobrazit progres úloh, které zadal a následně jejich výsledek a statistiky o průběhu výpočtu.

PCFG Monitor

Serverový démon *PCFG Monitor* periodicky kontroluje, jestli je k novým úlohám, které používají pro vytváření kandidátních hesel PCFG útok, přidělený *PCFG Manager* (viz sekci 2.3.1). Pokud přidělený není, *PCFG Monitor* spustí instanci *PCFG Manager* pro danou úlohu [9].

PCFG Manager server

Instance *PCFG Manager* generuje předterminální strukturu z dané gramatiky a komunikuje se serverovým démonem *Generator* (viz sekci 2.3.1). Po vytvoření nové části úlohy (*workunit*) pro daný uzel je *PCFG Manager* požádán o jednu nebo více předterminálních struktur [9]. Tyto struktury jsou poté společně s ostatními potřebnými soubory poslány klientskému uzlu na kterém běží *PCFG Manager Client*, který tyto struktury použije k vytváření kandidátních hesel.

2.3.2 Serverové nástroje

V této podsekci jsou popsané nástroje, které využívá serverová část systému Fitcrack.

[8] https://vuejs.org/
[9] https://flask.palletsprojects.com/en/3.0.x/

XtoHashcat

Při použití nástroje Hashcat pro dešifrování souborů chráněných heslem, je nutné získat heš ručně. Fitcrack poskytuje abstrakci nad tímto procesem, nástroj zvaný *XtoHashcat*. Tento nástroj umožňuje automaticky detekovat formát vstupního šifrovaného souboru nebo archivu a získat heš potřebný pro lámání. *XtoHashcat* podporuje následující vstupní formáty: soubory MS Office a PDf a archivy RAR, ZIP a 7z [9].

hcstat2gen

Prolomení hesla pomocí útoku hrubou silou s použitím Markovových řetězců (viz sekci 3.2.1) vyžaduje soubor `.hcstat2` obsahující Markovovy statistiky ve formě pravděpodobnostních matic. Fitcrack podporuje automatické vytváření nových souborů `.hcstat2` zpracováním existujících slovníků. K tomuto účelu používá nástroj *hcstats2gen*, který generuje vlastní soubor `.hcstat2` z vybraného slovníku [9].

princeprocessor

Útok PRINCE (viz sekci 3.2.9) využívá nástroj *princeprocessor* na straně serveru pro vypočítání úlohy dané vstupním slovníkem a konfiguračními možnostmi. Při použití možnosti `--keyspace` nástroj vypočítá počet kandidátních hesel tzv. *keyspace* [9].

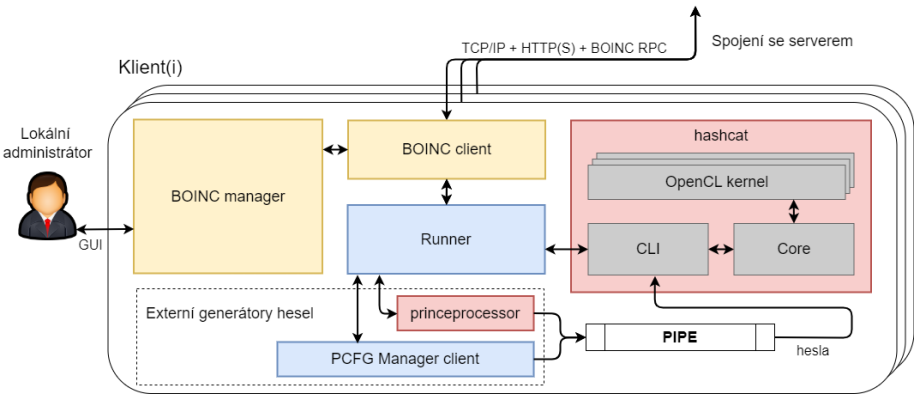
PCFG Trainer

Při útoku PCFG (viz sekci 3.2.11) je využit nástroj *PCFG Trainer*, který slouží k vytvoření pravděpodobnostní bezkontextové gramatiky ze vstupního slovníku.

2.3.3 Klientská část

Klientem je v systému Fitcrack výpočetní uzel, který provádí lámání hesel. Klientem může být jakékoliv zařízení na kterém je nainstalovaný BOINC *Client* a volitelně BOINC *Manager* (viz sekce 2.1). K systému Fitcrak může být připojen jeden a více klientů. Klient provádí samotné lámání a výsledky předává serveru.

Po připojení klienta pomocí BOINC *Client* k serveru projektu, jsou staženy všechny soubory potřebné k lámání hesel. Mezi tyto soubory patří Hashcat (viz sekci 2.2) a podsystém *Runner* (viz sekci 2.3.3). Architektura klientské části systému Fitcrack je znázorněna na obrázku 2.2. Žlutou barvou jsou vyznačené části platformy BOINC, modrou barvou jsou znázorněny části systému Fitcrack a červenou části patřící nástroji Hashcat.



Obrázek 2.2: Architektura klientské části systému Fitcrack [9]

Runner

Runner zapouzdřuje nástroj Hashcat. Je navržený jako samostatný program pro zjednodušení práce s tímto nástrojem a nebo jako middleware program v systému BOINC. Pro kompatibilitu s většinou systémů je *Runner* distribuován jako předkompilovaný binární soubor pro různé architektury.

Poté co je *Runner* spuštěn, na základě souboru `config`, vytvoří argumenty pro nástroj Hashcat [9]. Spustí externí generátor kandidátních hesel, pokud jej spuštěný útok vyžaduje. Následně *Runner* spustí Hashcat s danými argumenty a monitoruje progres lámání. Sbírá výsledky a vytváří výstupní soubor, který je následně předán systému BOINC.

Princeprocessor

Při útoku PRINCE (viz sekci 3.2.9) je použitý externí generátor kandidátních hesel *Princeprocessor* na straně klienta. Generuje kandidátní hesla, která posílá na vstup nástroje Hashcat, který je spuštěn v módu pro slovníkový útok (viz sekci 3.2.3) bez specifikovaného slovníku.

PCFG Manager client

Útok PCFG (viz sekci 3.2.11) využívá k vytváření kandidátních hesel externí generátor *PCFG Manager client*, který vytváří kandidátní hesla na základě předterminálů vytvořených pomocí *PCFG Manager server* (viz sekci 2.3.1). Kandidátní hesla jsou poté poslána na vstup nástroje Hashcat, který je spuštěn v módu pro slovníkový útok (viz sekci 3.2.3) bez specifikovaného slovníku.

Kapitola 3

Lámání hesel v systému Fitcrack

Lámání hesla je proces získání hesla k chráněnému obsahu hrubou silou. Proces lámání spočívá v generování kandidátních hesel a jejich následné verifikaci. Útoky použité pro prolomení hesla lze rozdělit na dva druhy: online a offline [6].

Při online útoku útočník zadává kandidátní hesla přímo do formulářů pro přihlášení na webových stránkách či elektronických zařízeních. Verifikace těchto hesel probíhá ihned při pokusu o přihlášení. Hesla ale nejsou ukládána ve své původní podobě, jsou funkcí transformována na řetězec znaků tzv. heš, z kterého nelze získat opačným výpočtem původní podobu hesla. Při offline útoku má útočník přímý přístup k heši hesla, generuje kandidátní hesla a spočítá jejich heše. Verifikace probíhá porovnáním heše lámaného hesla a heše kandidátního hesla. Pokud jsou si heše rovny, bylo nalezeno heslo. Tento přístup používá systém Fitcrack. V této kapitole jsou popsány jednotlivé typy útoků, které podporuje systém Fitcrack (viz kapitolu 2) a způsob distribuce kandidátních hesel mezi připojené uzly podílející se na lámání.

3.1 Distribuce kandidátních hesel

Lámací úloha zadaná uživatelem do systému Fitcrack je zvaná *job*. Je definovaná jedním nebo více zadanými heši, typem útoku (viz sekci 3.2) a jeho konfigurací. Tato úloha nebývá přidělena celá jednomu výpočetnímu uzlu, vždy je rozdělena na menší části zvané *workunit* a ty jsou postupně přidělovány jednomu nebo více uzlů podle jejich výkonu, který je zjištěn pomocí úlohy *benchmark*, která je popsaná v sekci 3.3. *Workunit* vždy obsahuje část kandidátních hesel celé úlohy. Pokud je k úloze přidělen jeden uzel s velkou výpočetní silou, můžou mu být přiděleny všechny kandidátní hesla v jedné *workunit*.

Systém Fitcrack využívá dynamickou distribuci práce, tzn. nerozdělí všechny kandidátní hesla na začátku lámání, ale přiděluje menší části postupně výpočetním uzlům. Takto je systém méně náchylný ke ztrátě výsledků *workunit*, díky jejich menší velikosti [9].

Připojené uzly disponují různou výpočetní silou, která se v průběhu výpočtu může měnit. Uzly se také mohou dynamicky připojovat a odpojovat od serveru. Pro co nejefektivnější distribuci práce je v systému Fitcrack implementován algoritmus pro **adaptivní plánování**. Velikost *workunit* přidělené výpočetnímu uzlu závisí na jeho současném výkonu, tj. uzlu s větším výpočetním výkonem bude přidělena větší *workunit*, než uzlu s nižším výkonem. Dále velikost *workunit* závisí na uživatelem zadané maximální velikosti *workunit*, uběhlého času lámání a na nastavení proměnných α a β, které jsou vysvětleny dále v této kapitole.

3.1.1 Adaptivní plánování

Algoritmus pro **adaptivní plánování** (viz algorimus 1) slouží k výpočtu doby zpracování *workunit* t_p na základě uběhlého času od spuštění lámání t_J, počtu kandidátních hesel s_R a počtu uzlů podílejících se na lámání k. Řádky algoritmu 2 až 5 zobrazují výpočet celkového výkonu uzlů v_{sum} a zbytek algoritmu určuje hodnotu t_p na základě podmínek. Proměnné t_{pmin} a t_{pmax} určují minimální a maximální velikost *workunit*. Distribuční koeficient α určuje maximální počet zbylých kandidátních hesel, který může být přidělen jedné *workunit*. *Ramp down* koeficient β zaručuje, že počet kandidátních hesel přidělený *workunit* nebude větší než maximální velikost *workunit* zadaná uživatelem krát β.

Algoritmus 1 implementuje techniky *ramp up* a *ramp down*. Technika *ramp up* vytváří na začátku úlohy menší *workunit*. Protože počáteční benchmark nemusí být přesný, je touto technikou zajištěno, že uzel neobdrží větší *workunit*, dokud neprokáže schopnost vyřešit menší. Naopak technika *ramp down* vytváří menší *workunit* ke konci úlohy. Touto technikou je eliminována situace, kdy je rozdělen celý prostor kandidátních hesel mezi připojené uzly a uzly, které skončily s počítáním, musí čekat na ty, které ještě nedopočítaly [9].

Algoritmus 1 Adaptivní výpočet t_p [9]

Vstup: t_J, s_R, k
Výstup: t_p

1: $v_{sum} = 0$
2: **for all** $client_i \in \{0, \ldots, k\}$ **do**
3: **if** $client_i$ *is active* **then**
4: $v_i = \frac{s_{prev}}{t_{prev}}$
5: $v_{sum} = v_{sum} + v_i$
6: **end if**
7: **end for**
8: $t_p = \frac{s_R}{v_{sum}} \cdot \alpha$
9: $t_{prealmin} = \max(t_{pmin}, t_{pmax} \cdot \beta)$
10: **if** $t_p < t_{prealmin}$ **then**
11: $t_p = t_{prealmin}$
12: **else**
13: $t_p = \min(t_p, t_{pmax})$
14: **if** *rampup* **and** $(t_J \leq t_{pmax})$ **then**
15: $t_p = \min(t_p, t_J)$
16: **end if**
17: **end if**
18: **return** t_p

3.2 Typy útoků

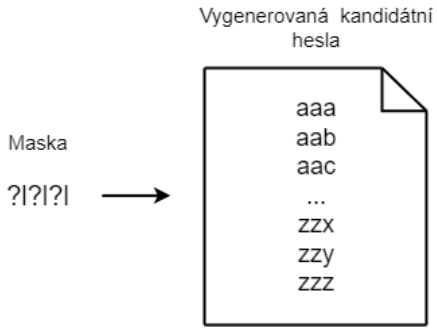
Typ útoku definuje, jakým způsobem budou generována kandidátní hesla. Systém Fitcrack podporuje všechny typy útoků, které implementuje nástroj Hashcat (viz sekci 2.2). V této sekci jsou popsány následující útoky: útok hrubou silou, slovníkový útok, kombinační útok, hybridní útok a útoky PRINCE a PCFG. Tato sekce také popisuje distribuci kandidátních hesel mezi jednotlivé uzly na základě použitého typu útoku.

3.2.1 Útok hrubou silou

Útok hrubou silou systematicky testuje všechny možnosti kandidátního hesla vytvořeného kombinací znaků z konečné sady. Útok hrubou silou je v nástroji Hashcat založen na maskách. Maska je vzor, který specifikuje na jaké pozici se může nacházet daný znak kandidátního hesla. Pro útok může být definována jedna nebo více masek. Proces lámání spočívá ve vytvoření všech možných kombinací znaků, které definuje maska [9]. Maska je složena ze zástupných symbolů, které definují sadu znaků. Zástupné symboly[1], které podporuje nástroj Hashcat jsou:

- **?l** – malá písmena latinské abecedy,
- **?u** – velká písmena latinské abecedy,
- **?d** – čísla,
- **?s** – speciální znaky,
- **?h** – hexadecimální čísla s malými písmeny,
- **?H** – hexadecimální čísla s velkými písmeny,
- **?a** – ASCII znaky,
- **?b** – binární čísla.

Obrázek 3.1 znázorňuje ukázku vytváření kandidátních hesel pomocí útoku hrubou silou. Maska `?l?l?l` vytváří kandidátní hesla o délce 3 znaky obsahující pouze malá písmena latinské abecedy. Kandidátní hesla jsou tvořena lexikograficky.



Obrázek 3.1: Ukázka generování hesel hrubou silou

Počet kandidátních hesel lze vypočítat jako součin mohutností množin zástupných symbolů C_i. Kde n_s je počet zástupných symbolů v masce [9].

$$p = \prod_{i=1}^{n_s} |C_i| \tag{3.1}$$

Maska `?l?l?l` tedy generuje $|C_l| \cdot |C_l| \cdot |C_l| = 26 \cdot 26 \cdot 26 = 17576$ kandidátních hesel.

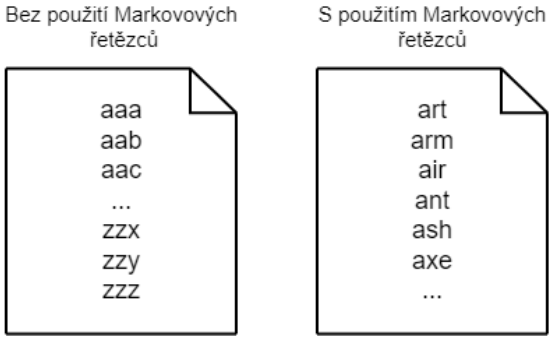
[1] https://hashcat.net/wiki/doku.php?id=mask_attack

Použití Markovových řetězců pro útok hrubou silou

Útok hrubou silou v nástroji *Hashcat* nevytváří kandidátní hesla lexikograficky, ale využívá algoritmu založeném na Markovových řetězcích. Matice pravděpodobnosti Markovova řetězce je ukázaná na obrázku 3.2, určuje pravděpodobnost výskytu znaků následujících za konkrétním znakem [6]. Výše pravděpodobnosti výskytu znaku klesá z leva do prava. Za písmenem `a` je nejpravděpodobnější výskyt písmene `z`, poté písmene `b`, dále písmene `x`, atd. Řádek označený ε značí nejpravděpodobnější znaky, které se budou v řetězci vyskytovat na prvním místě. Při vytváření řetězců z pravděpodobnostní matice na obrázku 3.2 bude jako první znak nejpravděpodobněji vybráno písmeno `a`, jako druhý znak bude nejpravděpodobněji vybráno `z`, jako třetí `b`, atd. Myšlenka použití Markovových řetězců pro vytváření kandidátních hesel spočívá v použití znalostí získaných z učení se na seznamu slov pro generování pravděpodobnějších hesel. Obrázek 3.3 znázorňuje rozdíl mezi inkrementálním útokem hrubou silou a útokem hrubou silou s použitím Markovových řetězců.

ε	a	n	m	h	k	p	v	s	. . .
a	z	b	x	p	l	f	h	d	. . .
b	o	e	d	a	k	p	q	i	. . .
c	r	a	v	n	m	e	i	o	. . .
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
y	a	n	m	h	k	p	v	s	. . .
z	b	j	f	m	d	x	p	v	. . .

Obrázek 3.2: Pravděpodobnostní matice Markovova řetězce



Obrázek 3.3: Generování kandidátních hesel pomocí Markovových řetězců

Nástroj *Hashcat* podporuje dva typy Markovových modelů: 2D Markovův model a 3D Markovův model. 2D Markovův model využívá pro sadu znaků jednu pravděpodobnostní matici a jeho fungování je vysvětleno výše v této kapitole. 3D Markovův model pro danou sadu znaků využívá jinou pravděpodobnostní matici pro každou pozici znaku v kandidátním hesle. Tento model využívá myšlenky, že pravděpodobnost výskytu znaku není ovlivněna pouze předchozím znakem, ale také pozicí, na které se znak nachází [9].

3.2.2 Distribuce útoku hrubou silou

Při distribuci masky či masek mezi uzly podílející se na výpočtu je potřeba posílat uzlu pouze masku a rozmezí indexů, kterými nástroj Hashcat omezí počet hesel k verifikaci. Celkový počet kandidátních hesel vypočtený nástrojem Hashcat neudává reálný počet kandidátních hesel [9]. Systém Fitcrack proto implementuje vlastní algoritmus pro výpočet reálného počtu kandidátních hesel, který je potřebný k výpočtu velikostí *workunit*. Porovnáním počtu kandidátních hesel spočtených nástrojem Hashcat a systémem Fitcrack lze poté určit kolik reálných kandidátních hesel reprezentuje jeden index v nástroji Hashcat.

3.2.3 Slovníkový útok

Slovníkový útok využívá textový soubor zvaný slovník hesel, který obsahuje na každém řádku jedno kandidátní heslo. Nástroj Hashcat postupně čte tyto kandidátní hesla a počítá jejich heše. Jednotlivé heše porovnává s hešem lámaného hesla. Fitcrack podporuje použití jednoho nebo více slovníků. Slovníky hesel mohou obsahovat slova z běžného jazyka nebo hesla, která unikla z různých webových stránek.

Z matematického hlediska můžeme slovník označit za konečnou uspořádanou množinu D. Pro slovníky obsahující n hesel lze počet kandidátních hesel p (v terminologii systému Fitcrack nazývaný *keyspace*) vypočítat jako sumu mohutností těchto slovníků. Kde D_i je i–tý slovník [6].

$$p = \sum_{i=1}^{n} |D_i| \tag{3.2}$$

Slovníkový útok lze rozšířit použitím pravidel upravujících kandidátní hesla definované slovníkem. Tato pravidla mohou zahrnovat výměnu znaků na definovaných pozicích, převádění malých písmen na velká a naopak, odstranění znaků z kandidátního hesla atd. Nástroj Hashcat podporuje více než 70[2] různých pravidel. Sada takových pravidel je nazývána *ruleset*. Na každém řádku sady se nachází jedno pravidlo [9].

Proces lámání spočívá v modifikaci každého z kandidátních hesel ve slovníku postupně každým pravidlem. Výsledný počet kandidátních hesel p lze spočítat jako sumu počtu kandidátních hesel slovníků vynásobených počtem pravidel. Kde r je počet řádků v sadě pravidel.

$$p = \sum_{i=1}^{n} (r \cdot |D_i|) \tag{3.3}$$

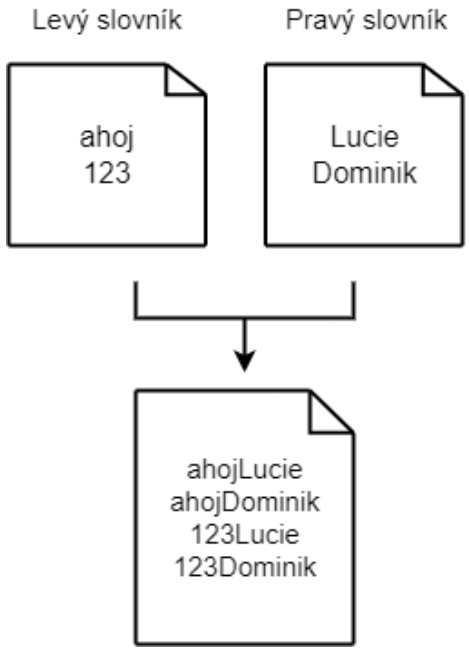
3.2.4 Distribuce slovníkového útoku

Systém Fitcrack distribuuje slovník či slovníky mezi jednotlivé výpočetní uzly po segmentech. Velikost segmentu závisí na aktuální výpočetní síle uzlu, pro který je segment tvořen. Slovník by mohl být poslán celý každému uzlu pouze s jinými indexy, jak tomu je při útoku hrubou silou, ale větší velikost slovníku by mohla zapříčinit značné prodloužení verifikace hesel [9]. Pokud tedy úlohu, která obsahuje 1000 kandidátních hesel počítají dva uzly s výpočetními silami 20 h/j a 80 h/j *(hesla za jednotku času)*. Na začátku výpočtu tedy budou pomalejšímu uzlu přiřazeny hesla 1 až 20 a rychlejšímu uzlu 21 až 100.

[2] https://hashcat.net/wiki/doku.php?id=rule_based_attack

3.2.5 Kombinační útok

Kombinační útok využívá k vytvoření kandidátních hesel dva slovníky, levý a pravý. Kandidátní hesla jsou tvořena jako konkatenace řetězců. Hesla z levého slovníku jsou konkatenována hesly z pravého slovníku. Vytváření kandidátních hesel pomocí kombinačního útoku je znázorněno na obrázku 3.4.



Obrázek 3.4: Ukázka generování hesel kombinačním útokem

Počet kandidátních hesel p lze vypočítat vynásobením mohutnosti obou slovníků, kde D_1 je levý slovník a D_2 je pravý slovník [6].

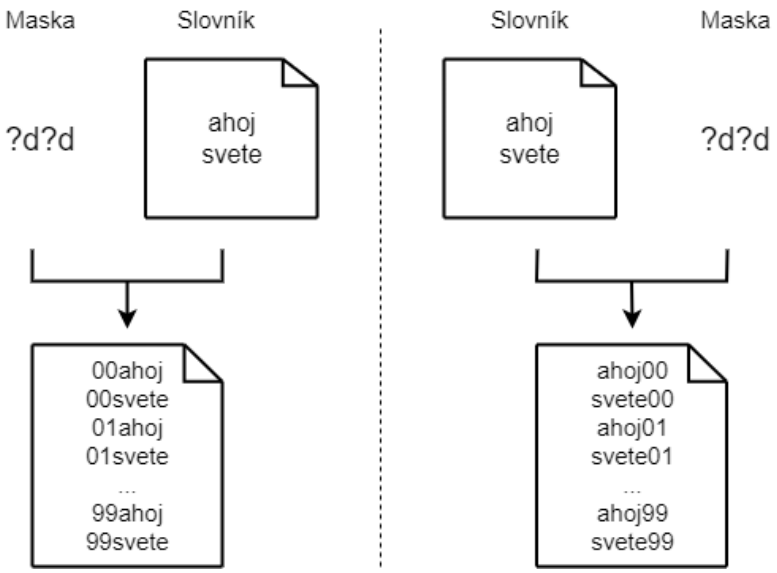
$$p = |D_1| \cdot |D_2| \tag{3.4}$$

3.2.6 Distribuce kombinačního útoku

Systém Fitcrack řeší distribuování kombinačního útoku tak, že posílá celý prostor kandidátních hesel prvního slovníku všem připojeným hostům v první *workunit* a následně jsou uzlům postupně zasílány menší segmenty druhého slovníku. Část prvního slovníku, která je přidělena danému uzlu je omezena indexy, které jsou v nástroji Hashcat definovány parametry `--skip` a `--limit` [9].

3.2.7 Hybridní útoky

Hybridní útoky kombinují útok hrubou silou (viz 3.2.1) se slovníkovým útokem (viz 3.2.3). Nástroj Hashcat podporuje dva typy hybridních útoků: útok hrubou silou na levé straně a slovníkový útok na pravé straně a obráceně. Vytváření kandidátních hesel spočívá v konkatenaci řetězců znaků těchto dvou částí [6]. Obrázek 3.5 znázorňuje ukázku vytváření kandidátních hesel konkatenací řetězců vytvořených pomocí masky a řetězců obsažených ve slovníku.



Obrázek 3.5: Ukázka generování hesel hybridními útoky

Počet kandidátních hesel pro hybridní útok lze vypočítat jako součin rovnice pro útok hrubou silou (3.1) a rovnice pro slovníkový útok (3.3).

$$p = |D| \cdot \prod_{i=1}^{n_s} |C_i| \tag{3.5}$$

3.2.8 Distribuce hybridních útoku

Při hybridním útoku se slovníkem na levé straně a maskou na pravé je maska rozdělena na několik menších masek. Počet kandidátních hesel ze slovníku, který je posílaný uzlům je limitován parametry `--skip` a `--limit` s použitím algoritmu, který určuje, jestli tyto parametry použít nebo nikoli [9]. Při útoku s maskou na levé straně a slovníkem na pravé straně je slovník rozdělen na menší segmenty stejně jako při kombinačním útoku (viz sekci 3.2.6) a kandidátní hesla vzniklé z masky jsou limitovány parametry `--skip` a `--limit` a algoritmem stejně jako u opačného typu tohoto útoku.

3.2.9 Útok PRINCE

PRINCE (PRobability INfinite Chained Elements) je moderní algoritmus pro generování kandidátních hesel, který je možné použít na pokročilé kombinační útoky. PRINCE používá pro generování hesel jeden slovník, z kterého vytváří řetězce (*chains*). Tyto řetězce mohou být vytvořeny z jedno a více slov pocházejících ze vstupního slovníku [4].

Celkový počet kandidátních hesel p lze vypočítat jako sumu počtu kandidátních hesel jednotlivých řetězců, kde n je počet vygenerovaných řetězců [9].

$$p = \sum_{i=1}^{n} keyspace(chain_i) \tag{3.6}$$

3.2.10 Distribuce útoku PRINCE

Pro výpočet kandidátních hesel je na Fitcrack serveru nejdříve spuštěn externí generátor *princeprocessor* (viz sekci 2.3.3) s konfigurací úlohy. Na straně klienta je poté pomocí parametrů `--skip` a `--limit` a algoritmu PRINCE spočítán počet kandidátních hesel, který bude přidělen danému uzlu [9]. Výstup generátoru *princeprocessor* je připojen na vstup nástroje Hashcat a kandidátní hesla jsou jsou verifikována hned po vytvoření.

3.2.11 Útok PCFG

Tento typ útoku je založen na povědomí o heslech daného uživatele. Struktura těchto hesel je reprezentována pravděpodobnostní bezkontextovou gramatikou.

3.2.12 Distribuce útoku PCFG

Při útoku PCFG jsou kandidátní hesla tvořena externím generátorem a je spuštěn *PCFG Manager*, kterého výstup je připojen na vstup nástroje Hashcat. Hashcat je spuštěn v módu slovníkového útoku (viz sekci 3.2.3) bez specifikovaného slovníku a kandidátní hesla jsou čtena přímo ze standardního vstupu [9].

3.3 Techniky měření výkonu výpočetních uzlů

Celková úloha pro prolomení hesla zadaná uživatelem je rozdělena na více částí (*workunit*), které jsou rozeslány jednotlivým výpočetním uzlům. Způsob jakým jsou tvořeny *workunit* je popsán v sekci 3.1. V rámci systému Fitcrack může být výpočetnímu uzlu přidělen jeden ze tří typů *workunit*: úloha pro samotné lámání, *benchmark* nebo *complete benchmark*. V konfiguračním souboru je úloha pro samotné lámání označena písmenem `n`, *benchmark* je značen písmenem `b` a *complete benchmark* je značen písmenem `a` [9]. Úlohy, které se zabývají měřením výkonu jsou *benchmark* a *complete benchmark*.

Úloha ***complete benchmark*** je standardně spuštěna pouze jednou a je provedena po připojení nového výpočetního uzlu k systému Fitcrack. Implicitně je *complete benchmark* vypnutý a lze jej zapnout v nastavení webové aplikace *WebAdmin* pro vzdálenou zprávu systému Fitcrack (viz sekci 2.3.1). Cílem *complete benchmark* je změřit dosažitelnou rychlost lámání výpočetního uzlu pro všechny hešovací algoritmy a typy útoků. Úloha provede benchmark pro všechny podporované hešovací algoritmy a výsledek uloží do databáze do tabulky `fc_benchmark`. Tato data jsou posléze použita pro výpočet odhadovaného času lámání (viz sekci 3.4) před spuštěním úlohy pro lámání.

Úloha ***benchmark*** je provedena na každém výpočetním uzlu před spuštěním úlohy pro samotné lámání. *Benchmark* určí aktuální rychlost lámání pro daný uzel, hešovací algoritmus a typ útoku. Úloha, kterou uzel obdrží pro změření jeho výkonu je ta, kterou by obdržel při samotném lámání. Tato úloha je spuštěna pouze na krátký časový úsek a je omezen počet kandidátních hesel. Pro *benchmark* spuštěný při útoku hrubou silou jsou měřenému uzlu zaslány pouze kandidátní hesla první zadané masky [12]. Po ukončení úlohy *benchmark* je výsledek uložen do databáze do tabulky `fc_benchmark`, pokud v databázi již existuje záznam tohoto typu, je nahrazen aktuálnějším výsledkem úlohy [9].

Tabulka `fc_benchmark` obsahuje výsledky úloh typu benchmark jednotlivých připojených výpočetních uzlů. Každý záznam v této tabulce reprezentuje rychlost lámání pro daný uzel a hešovací algoritmus. Atributy jednotlivých záznamů v této databázové tabulce jsou definovány následujícím způsobem:

- **id** – primární klíč,
- **boinc_host_id** – identifikační číslo hosta v tabulce hostů platformy BOINC,
- **hash_type** – číslo[3] hešovacího algoritmu v nástroji Hashcat,
- **power** – naměřená hodnota rychlosti lámání v heších za sekundu,
- **last_update** – čas poslední úpravy záznamu.

3.4 Metodika aproximace doby výpočtu

Současné řešení výpočtu maximální odhadované doby lámání je řešeno v souboru `functions.py`, který se nachází v cestě `fitcrack/webadmin/fitcrackAPI/src/src/api/fitcrack/endpoints/job/functions.py`[4]. Tento soubor obsahuje funkci `computeCrackingTime()`, která je klíčová pro poskytnutí uživatelům systému Fitcrack představu o maximální možné době, kterou zabere prolomení hesla s daným nastavením a na základě dostupných výpočetních zdrojů. Funkce nejprve zkontroluje jestli je validní číslo hešovacího algoritmu a jestli je jeden nebo více aktivních výpočetních uzlů pro danou úlohu. Poté jsou z tabulky `fc_benchmark` vybrány záznamy uložené z předchozích úloh na základě kombinace identifikačního čísla aktivních hostů, typu útoku a hešovacího algoritmu. Pokud pro daný typ útoku neexistuje žádný záznam, jsou vybrány záznamy pouze na základě hešovacího algoritmu.

Celkový výkon všech aktivních hostů *total_power* je vypočítán jako suma hodnot atributu *power* všech vybraných záznamů. Počet kandidátních hesel, tzv. *keyspace* je vypočítán na základě zadaného typu útoku a jeho konfiguraci. Popis výpočtu *keyspace* se nachází v sekci 3.2. Všechna potřebná data jsou získána z databáze. Maximální odhadovaná doba lámání je vypočítána jako poměr *keyspace* a celkového výkonu aktivních uzlů. Ukázka výpočtu je znázorněna ve výpisu 3.1.

```
display_time = None
if (total_power > 0):
    display_time = float(keyspace / total_power)
    try:
        time_delta = datetime.timedelta(seconds=math.floor(display_time))
        if time_delta.total_seconds() < 60:
            display_time = 'About a minute'
        else:
            display_time = str(time_delta)
    except OverflowError:
        display_time = 'really long'
```

Výpis 3.1: Ukázka části funkce `computeCrackingTime()`

Z ukázky části kódu funkce `computeCrackingTime()` lze extrahovat samotný vzorec použitý pro výpočet maximální odhadované doby lámání 3.7.

[3] https://hashcat.net/wiki/doku.php?id=example_hashes

[4] https://github.com/nesfit/fitcrack/blob/master/webadmin/fitcrackAPI/src/src/api/fitcrack/endpoints/job/functions.py

$$display\_time = \frac{keyspace}{total\_power} \tag{3.7}$$

Kde *keyspace* je celkový počet kandidátních hesel a *total_power* je suma výpočetního výkonu všech uzlů podílejících se na řešení úlohy.

Popsané řešení je použito pokud úloha byla nakonfigurována uživatelem, ale nebyla ještě spuštěna. Po spuštění je odhadovaný čas stále přepočítáván. Pro tento účel je použita obdobná funkce funkci `computeCrackingTime()`. Implementace této funkce se nachází v souboru `models.py` na cestě `fitcrackAPI/src/src/database`.

3.5 Efektivita distribuovaného výpočtu

Při distribuci výpočtu mezi více výpočetních uzlu je doba, za kterou obdržíme výsledek, složena ze samotné verifikace kandidátních hesel a komunikace mezi uzlem a serverem. Efektivita lámání reprezentuje procento výsledného času, který uzel strávil samotným lámáním hesla bez komunikace se serverem. Efektivita je vypočítána vzorcem 3.8, kde N je počet uzlů, které se podílely na výpočtu, t_x je čas jednotky práce tzv. *workunit* a T_{fin} je celkový čas práce [7].

$$E_{ff} = \frac{\sum_{x=1}^{N} t_x}{N \cdot T_{fin}} \tag{3.8}$$

Hodnota efektivity není standardně zobrazena při dokončení zadané práce. Uživatel může povolit zobrazení této hodnoty v pokročilém nastavení webové aplikace *WebAdmin* pro vzdálenou zprávu systému Fitcrack (viz sekci 2.3.1) zapnutím režimu pro vývojáře.

Kapitola 4

Kritická místa aproximace doby lámání

Tato kapitola se věnuje faktorům, které ovlivňují odhad maximální doby lámání, která je uživateli zobrazena po zadání konfigurace dané úlohy a před jejím spuštěním. Měření probíhala pro hešovací algoritmus MD5 a byla primárně zaměřena na útok hrubou silou. Zjištěná kritická místa v této kapitole jsou společná pro všechny typy útoků implementované v systému Fitcrack.

4.1 Měření hodnot

Pro měření hodnot byly využity čtyři počítače s operačním systémem Microsoft Windows, které sloužily jako výpočetní uzly připojené k serveru na doméně `live2.fitcrack.cz`. Specifikace jednotlivých počítačů jsou uvedeny v tabulce 4.1.

Č.	**Operační systém**	**Procesor**	**Grafická karta**
1	Windows 10	Intel Core i3-7100U	Integrovaná
2	Windows 10	Intel Core i5-4460	NVIDIA GeForce GTX 750
3	Windows 11	13th Gen Core i5-13400F	NVIDIA GeForce RTX 2060
4	Windows 11	13th Gen Core i7-13700H	NVIDIA GeForce RTX 4050

Tabulka 4.1: Specifikace klientů

Všechny úlohy byly spuštěny s hešovacím algoritmem MD5 a útokem hrubou silou. Vzorec pro výpočet odhadované doby lámání popsaný v sekci 3.4 určuje maximální možnou dobu lámání, tj. dobu za kterou se porovnají všechny vytvořená kandidátní hesla s hledaným heslem nebo hesly. Hledané heslo může být nalezeno dříve než je prohledán celý prostor kandidátních hesel. Pro co nejpřesnější naměřené hodnoty byly masky pro útok hrubou silou tvořené tak, aby se neshodovaly s hledanými hesly a reálná doba trvání celkové úlohy byla rovna době potřebné pro prohledání celého prostoru kandidátních hesel.

Pro každou úlohu byly do tabulky 4.2 zaznamenány tyto informace:

- **Odh. čas (odhadovaný čas)** – výsledek vzorce pro aproximaci doby výpočtu (viz sekci 3.4),
- **Výsl. čas (výsledný čas)** – reálná doba, za kterou byla úloha dokončena,

- **Abs. rozdíl (absolutní rozdíl)** – rozdíl výsledného a odhadovaného času
- **Rel (relativní rozdíl)** – procentuální vyjádření, o kolik procent se odhadovaný čas liší od výsledného,
- **Ef. (efektivita)** – efektivita lámání popsaná v sekci 3.5,
- **Suma časů wu (suma časů *workunit*)**,
- **Čas lámání** – doba, kterou strávil výpočetní uzel pouze samotným lámáním,
- **Čas bench. (čas benchmarku)** – doba, kterou trvalo uzlu udělat benchmark, pokud do úlohy bylo zapojeno více uzlů, doba benchmarku je průměr doby trvání benchmarků všech uzlů,
- **Čas režie** – doba, kterou uzel nestrávil lámáním, ale komunikací se serverem,
- **Keyspace** – počet kandidátních hesel vypočítaný ze všech zadaných masek,
- **Průměrný keyspace** – průměrný počet kandidátních hesel na jednu masku,
- **Keyspace 1. masky** – počet kandidátních hesel vytvořených z první zadané masky,
- **P. uzlů** – počet výpočetních uzlů, které se podíleli na zadané úloze,
- **Čas wu (čas *workunit*)** – maximální doba trvání *workunit* zadaná uživatelem při vytváření úlohy,
- **P. m.** - počet masek zadaných uživatelem při vytváření úlohy.

#	Odh. čas	Výsl. čas	Abs. rozdíl	Rel. [%]	Ef. [%]	Suma časů wu	Čas lámání	Čas bench.	Čas režie	Keyspace	Průměrný keyspace	Keyspace 1. masky	P. uzlů	Čas wu [s]	P. m.
1	0:01:00	0:03:06	0:02:06	-67,74	21	0:01:24	0:00:39	0:00:43	0:02:27	1,76E+04	1,76E+04	1,76E+04	1	3600	1
2	0:01:00	0:03:41	0:02:41	-72,85	33	0:01:52	0:01:13	0:00:37	0:02:28	1,86E+04	9,29E+03	1,76E+04	1	3600	2
3	0:01:00	0:04:05	0:03:05	-75,51	43	0:02:30	0:01:45	0:00:43	0:02:20	3,92E+07	1,31E+07	1,76E+04	1	3600	3
4	0:01:00	0:04:25	0:03:25	-77,36	45	0:02:47	0:01:59	0:00:46	0:02:26	3,92E+07	1,31E+07	3,91E+07	1	3600	3
5	0:03:49	0:07:14	0:03:25	-47,24	70	0:05:32	0:05:04	0:00:28	0:02:10	5,43E+12	2,71E+12	5,43E+12	1	3600	2
6	0:05:16	0:09:10	0:03:54	-42,55	68	0:06:54	0:06:14	0:00:36	0:02:56	5,43E+12	2,71E+12	1,76E+04	1	3600	2
7	0:01:11	0:03:23	0:02:12	-65,02	15	0:02:07	0:00:30	0:00:32	0:02:53	2,09E+11	6,96E+10	2,09E+11	2	3600	3
8	0:37:43	0:50:40	0:12:57	-25,56	91	1:34:03	0:46:06	0:00:33	0:04:34	6,12E+13	1,53E+13	3,04E+11	2	3600	4
9	0:13:24	0:47:13	0:33:49	-71,62	61	1:28:07	0:28:48	0:00:32	0:18:25	2,16E+13	7,20E+12	2,09E+13	3	3600	3
10	0:01:55	0:21:03	0:19:08	-90,89	88	0:20:08	0:18:31	0:00:29	0:02:32	3,09E+12	3,09E+12	3,09E+12	3	3600	1
11	0:01:00	0:03:08	0:02:08	-68,09	11	0:02:08	0:00:21	0:00:15	0:02:47	4,71E+08	7,86E+07	3,09E+08	4	300	6
12	0:07:32	0:12:15	0:04:43	-38,5	73	0:37:21	0:08:57	0:00:20	0:03:18	2,09E+13	2,09E+13	2,09E+13	4	300	1
13	0:02:14	0:09:14	0:07:00	-75,81	50	0:05:29	0:04:37	0:00:50	0:04:37	4,73E+11	7,89E+10	1,63E+11	1	3600	6
14	0:01:35	0:05:20	0:03:45	-70,31	43	0:03:11	0:02:18	0:00:52	0:03:02	4,73E+11	7,89E+10	1,63E+11	1	3600	6
15	0:01:00	0:04:09	0:03:09	-75,9	30	0:03:18	0:01:15	0:00:25	0:02:54	4,73E+11	7,89E+10	1,63E+11	2	3600	6
16	0:29:33	0:35:05	0:05:32	-15,77	91	0:33:09	0:31:56	0:00:15	0:03:09	4,27E+13	4,27E+13	4,27E+13	1	3600	1
17	0:01:00	0:09:02	0:08:02	-88,93	58	0:05:53	0:05:14	0:00:34	0:03:48	2,45E+04	2,45E+03	2,60E+01	1	240	10

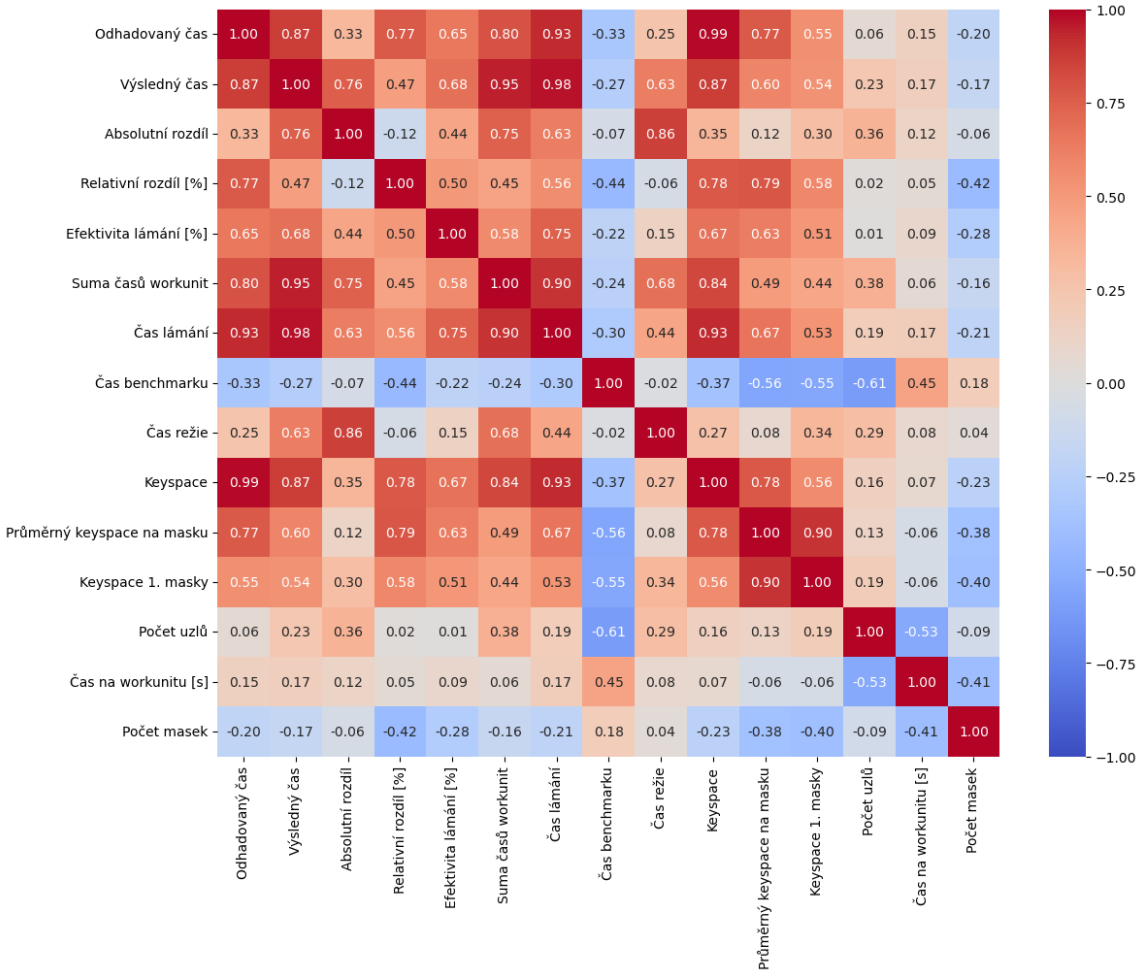
Tabulka 4.2: Tabulka naměřených hodnot pro úlohy typu útok hrubou silou spuštěné v systému Fitcrack

4.2 Korelace naměřených hodnot

Analýzu naměřených dat (viz tabulku 4.2) jsem provedla pomocí korelační matice. Korelace je statistická metoda, která se využívá k vyjádření míry závislosti mezi dvěma proměnnými. Výsledkem této metody je korelační koeficient, který může nabývat hodnot mezi -1 a 1. Pokud je hodnota korelačního koeficientu blízká 0, proměnné spolu nekorelují, tzn. není mezi nimi žádná závislost. Hodnota korelačního koeficientu blízká 1 indikuje pozitivní korelaci (tj. když jedna proměnná roste, druhá také roste), naopak hodnota blízká -1 indikuje negativní korelaci (tj. když jedna proměnná roste, druhá klesá).

Výhodný způsob reprezentace velkého počtu korelačních koeficientů je jejich vložení do tabulky, zvané korelační matice. Korelace jakékoli proměnné sama se sebou je jedna, tudíž úhlopříčka matice obsahuje pouze hodnoty 1 [2].

Pomocí skriptu napsaného v jazyce Python jsem vytvořila korelační matici (viz obrázek 4.1) z hodnot získaných měřením, které jsou ukázány v tabulce 4.2. Z hodnot korelačních koeficientů zobrazených v této matici lze vyčíst, které proměnné mohou mít vliv na odhad času lámání.



Obrázek 4.1: Korelační matice

Z korelační matice lze vyčíst, že **absolutní rozdíl** mezi odhadovaným časem a reálným časem má silnou pozitivní korelaci s **časem režie**, což značí, že čas strávený komunikací

mezi uzly a serverem může být významným faktorem ovlivňujícím odhadovaný čas lámání. Tento faktor byl hlavní inspirací návrhu algoritmů pro zlepšení odhadu času lámání. Detailnější rozbor této problematiky je uveden v sekci 4.3.2.

4.3 Zjištěná kritická místa

V této sekci jsou popsány místa v systému Fitcrack, která ovlivňují odhad doby lámání. Tato kritická místa byla zjištěna analýzou naměřených hodnot, které jsou zaznamenány v tabulce 4.2 a analýzou současného vzorce pro výpočet odhadované doby výpočtu.

4.3.1 Benchmark

Vzorec pro výpočet odhadované doby lámání, popsaný v sekci 3.4, počítá odhadovanou dobu jako poměr počtu kandidátních hesel a celkového výkonu všech aktivních hostů. V systému Fitcrack je vždy před samotným lámáním na každém připojeném uzlu spuštěn benchmark (viz sekci 3.3). Současný vzorec zanedbává dobu, která je potřebná pro provedení benchmarku na uzlech podílejících se na úloze. V tabulce 4.2 lze vidět že benchmark trval nejméně 15 sekund a nejvíce 52 sekund.

Dalším problémem spojeným s benchmarkem je fakt, že úloha benchmark je pro útok hrubou silou vytvořena pouze s první zadanou maskou. Pokud je první maska malá, např. `?l?l?l` rozdíl mezi odhadovaným časem a reálným časem je větší.

Č.	**Odhadovaný čas**	**Reálný čas**	**Absolutní rozdíl**	**První maska**	**Rychlost lámání[kH/s]**
1	0:03:49	0:07:14	0:03:25	?l?l?l?l?l?l?l?l?l	17 161 838
2	0:05:16	0:09:10	0:03:54	?l?l?l	3 197

Tabulka 4.3: Naměřené hodnoty pro dvě stejné úlohy s různým pořadím zadaných masek

Spustila jsem na jednom výpočetním uzlu dvě skoro totožné úlohy, které se lišily pouze v pořadí zadaných masek. Pro první úlohu jsem zvolila první masku `?l?l?l?l?l?l?l?l?l` a druhou masku `?l?l?l`, pro druhou úlohu byly masky zvoleny obráceně. Naměřené hodnoty jsou uvedeny v tabulce 4.3. Hodnoty času jsou uvedeny ve formátu `hodiny:minuty:sekundy`.

Problém, který je také viditelný v tabulce 4.3 je hodnota rychlosti lámání měřená v kiloheších za sekundu. Pro malé masky je benchmark velmi nepřesný a vypočítaná rychlost lámání se pro výpočetní uzel neshoduje s reálnou rychlostí lámání. Tato nepřesná hodnota rychlosti lámání poté ovlivní odhadovaný čas další spuštěné úlohy.

Po úloze s první maskou `?l?l?l` byla spuštěna úloha na stejném výpočetním uzlu s maskou `?l?l?l?l?l?l?l?l?l`. Podle dat z tabulky 4.3, by úloha měla trvat zhruba 7 minut. Odhadovaný čas pro tuto úlohu byl vypočítán na 19 dnů, 15 hodin, 42 minut a 25 sekund. Úloha byla dokončena za 5 minut a 59 sekund. Takto špatný odhad času je zapříčiněn uložením nepřesné rychlosti lámání do databázové tabulky `fc_benchmark` a následným použitím těchto nepřesných dat k výpočtu odhadovaného času pro další úlohu. Otázkou použití malých masek pro benchmark se zabývá v letošní bakalářské práci na téma *Optimalizace vysoce náročných úloh v systému Fitcrack* Ondřej Dacík a tak tuto problematiku nebudu v této práci dále rozvádět.

4.3.2 Režie distribuovaného výpočtu

V sekci 4.2, která se zabývá korelacemi mezi různými proměnnými, byl identifikován vliv času režie na přesnost odhadovaného času potřebného k prolomení hesla. Stávající vzorec pro odhad času se soustředí výhradně na dobu, po kterou probíhá samotné lámání hesla a nezahrnuje dobu strávenou komunikací uzlů se serverem. Tato část procesu je ve výpočtu opomenuta, což vede k odchylkám mezi odhadovaným a skutečným časem nutným pro prolomení hesla.

Hodnoty sloupce **Efektivita** v tabulce 4.2 udávají procentuální podíl času, který byl efektivně využit připojenými uzly na proces lámání hesel vůči celkovému strávenému času. Z tabulky lze vyčíst, že čím menší je efektivita, tím větší je relativní rozdíl. Vysvětlením tohoto jevu je fakt, že při distribuovaném rozdělení práce mezi více uzlů vzniká větší režie způsobená komunikací mezi uzly a serverem.

Zadaná úloha do systému Fitcrack je vždy rozdělena na menší části, tzv. *workunit*, které jsou postupně posílány připojeným výpočetním uzlům. Maximální dobu trvání *workunit* lze nastavit při vytváření úlohy. Čím menší je maximální čas *workunit*, tím větší je režie, protože je potřeba posílat uzlům více *workunit* a vzniká větší nárok na režii. Návrh řešení zahrnutí času potřebného pro komunikaci do vzorce pro odhad doby výpočtu je popsán v sekci 5.2.

4.3.3 Skutečný počet uzlů podílejících se na lámání

Během analýzy výsledků úloh spuštěných v systému Fitcrack jsem zaznamenala, že při připojení více výkonných uzlů k jednoduché úloze nedochází k rozdělení prostoru kandidátních hesel mezi všechny připojené uzly. Namísto toho jsou kandidátní hesla přidělena pouze nejrychlejším uzlům. Přestože odhadovaná doba lámání je vypočtena tak, jako by se na ní podílely všechny připojené uzly, skutečná doba lámání je ovlivněna pouze nejvýkonnějšími uzly.

Kapitola 5

Návrh úprav

Tato kapitola je věnována návrhu úprav vzorce pro odhad doby lámání hesel v systému Fitcrack. Tento návrh vychází z analýzy provedené v předchozí kapitole, kde byla identifikována kritická místa současného vzorce a poskytnuty podklady pro jeho zdokonalení.

5.1 Benchmark

V této sekci se zabývám úpravou vzorce pro výpočet maximální odhadované doby lámání v systému Fitcrack související s měřením výkonu výpočetních uzlů. Kritická místa jsou popsána v sekci 4.3.1 a problematika je popsána v sekci 3.3.

Z hodnot času benchmarku zaznamenaných v tabulce 4.2 jsem vypočítala minimální čas trvání benchmarku, maximální čas trvání benchmarku, medián, nejčastější dobu trvání benchmarku (modus), průměr a směrodatnou odchylku. Tyto hodnoty v sekundách jsou vyznačeny v tabulce 5.1.

Minimum	**Maximum**	**Průměr**	**Modus**	**Medián**	**Směrodatná odchylka**
15	52	34	43; 32; 15	33	11,08

Tabulka 5.1: Statistické hodnoty spočtené z času benchmarku v tabulce 4.2 (hodnoty jsou zadané v sekundách).

V rámci optimalizace vzorce pro odhad času navrhuji zahrnout do výpočtu konstantu, která reprezentuje průměrný čas strávený výpočtem benchmarků. Přidáním této konstanty do vzorce dostáváme upravený výraz 5.1, kde *keyspace* je počet kandidátních hesel, *total_power* je součet výkonů všech uzlů podílejících se na dané úloze a α je nově přidaná konstanta, která se rovná průměru naměřených hodnot doby trvání benchmarku, tj. 34 sekund.

$$estimated = \frac{keyspace}{total\_power} + \alpha \tag{5.1}$$

Tato modifikace vzorce umožňuje přesnější odhad doby lámání tím, že reflektuje nejen čas strávený přímo lámáním hesel, ale také čas, který je nutný pro provedení benchmarku.

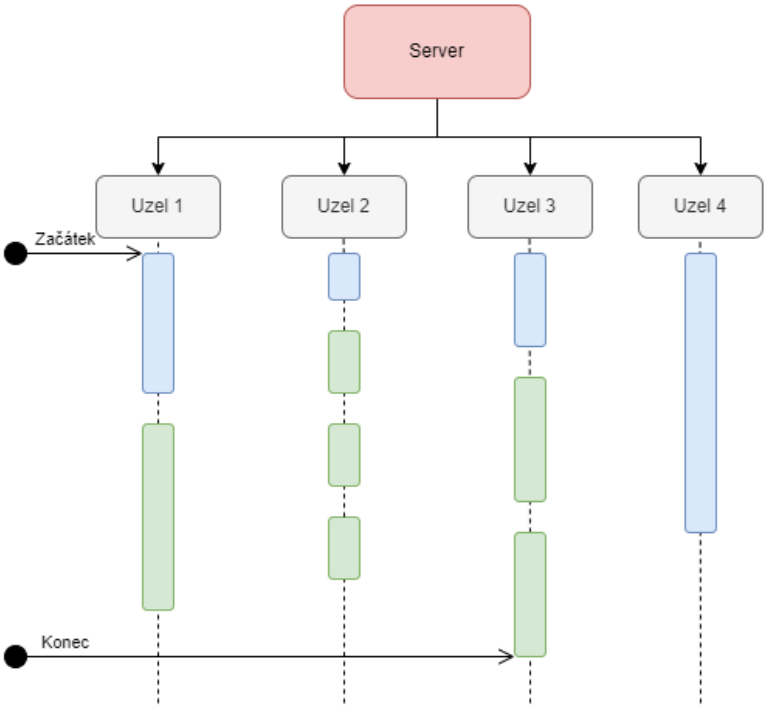
5.2 Zahrnutí doby režie do vzorce pro odhad

Kritickým faktorem, který zásadně ovlivňuje maximální odhadovanou dobu lámání, je čas potřebný pro komunikaci mezi připojenými uzly a serverem. Tento kritický faktor byl zjištěn z analýzy současného vzorce pro výpočet odhadované doby lámání (viz sekci 3.4), z měření provedeném na jednotlivých úlohách spuštěných v systému Fitcrack a ze studie metod tvoření *workunit*. Hodnoty z měření jsou zaznamenány v tabulce 4.2.

5.2.1 Rozdělení *workunit* připojeným uzlům

Při lámání hesel na jednom připojeném uzlu není třeba odhadnout rozdělení *workunit*, jelikož tento uzel obdrží všechny. Pouze je nutné odhadnout počet *workunit*, které uzel obdrží. Myšlenka úpravy vzorce spočívá ve stanovení režie potřebné na poslání jedné *workunit*, odhadnutí počtu *workunit*, které budou vygenerovány a následně přičíst k původnímu vzorci (viz sekci 3.4) čas režie vynásobený odhadovaným počtem *workunit*.

Pokud je k úloze připojeno více uzlů, které se podílejí na lámání, stává se určení odhadovaného času komplikovanějším. Pokud by byl použit stávající vzorec pro odhad času popsaný v sekci 3.4 a k němu připočten čas režie pro odhadovaný počet vygenerovaných *workunit*, tak bude dosaženo chybného výsledku. Nelze jednoduše určit, kolik *workunit* bude vygenerováno pro každý uzel, a tak nelze distribuovat čas režie na *workunit* mezi připojené uzly.



Obrázek 5.1: Časová osa úlohy a znázornění možného rozdělení *workunit* mezi připojené uzly.

Rozdělení *workunit* mezi připojené uzly může být ovlivněno různými faktory, kvůli kterým nelze před spuštěním úlohy přesně vypočítat počet a velikost *workunit* daného uzlu. Může dojít k výpadku uzlu nebo snížení jeho výkonu při zatížení jiným procesem. Velikost

workunit pro jednotlivé uzly je v průběhu úlohy počítána algoritmem pro adaptivní plánování (viz sekci 3.1.1), který určuje velikosti *workunit* na základě aktuálního výkonu uzlu, na nastavené maximální velikosti *workunit* a na koeficientech α a β popsaných v sekci 3.1.1.

Možný scénář průběhu a rozdělení *workunit* je znázorněn na obrázku 5.1. Obrázek znázorňuje čas, který daný uzel stráví lámáním, nikoli velikost *workunit*. V tomto scénáři jsou k serveru připojeny čtyři uzly s rozdílnými výkony (výkon uzlu 2 > výkon uzlu 3 > výkon uzlu 1 > výkon uzlu 4). Modrou barvou jsou zobrazeny *workunit* typu *benchmark*, které jsou blíže popsané v sekci 3.3 a zelenou barvou jsou vyznačené úlohy, ve kterých je prováděno samotné lámání. Na začátku lámání je každému uzlu poslán *benchmark* a na základě výkonu uzlu je mu poté pomocí adaptivního plánování přidělena *workunit*. Princip adaptivního plánování je popsán v sekci 3.1.1.

Z obrázku 5.1 lze vyčíst, že lámání skončí až dolámou všechny uzly, kterým byla přidělena práce. Může se stát, že prostor kandidátních hesel bude rozdělen mezi výkonnější uzly a méně výkonným uzlům nebude přidělena žádná *workunit*. Mnou navržená úprava vzorce pro výpočet odhadované doby lámání spočívá v použití části algoritmu pro výpočet velikostí *workunit*, který je popsán v sekci 3.1.1. Na základě výsledku tohoto algoritmu odhadnout, kolik kandidátních hesel zvládne každý uzel verifikovat v jedné *workunit*. Dále odhadnout počáteční rozdělení prostoru kandidátních hesel mezi připojené uzly, určit, které uzly se budou na výpočtu skutečně podílet a kolik *workunit* jim bude přiděleno. Poté pro každý uzel zvlášť vypočítat odhad času a režii. Výsledný odhadovaný čas bude určen jako maximální hodnota ze všech odhadovaných časů všech uzlů.

5.2.2 Stanovení času režie *workunit*

Následujícím krokem je určit konstantu, která bude určovat dobu režie, která je potřebná na poslání jedné *workunit* ze serveru uzlu a odeslání výsledku z uzlu na server. Hodnotu konstanty lze nejlépe určit analýzou nasbíraných dat obdržených z úloh spuštěných v systému Fitcrack. Pro automatizaci vytváření a spouštění úloh jsem modifikovala existující soubor `superbench.py` nacházející se v adresáři `fitcrack/tools`. Modifikovaný soubor `attacksLauncher.py` zakládá a spouští nové úlohy a po jejich dokončení (úloha ve stavu *exhausted*) ukládá získaná data do souborů `xlsx` a `csv`. Tento soubor se nachází na datovém nosiči v adresáři `experimenty` pod názvem `attacksLauncher.py` (viz přílohu B).

Z nasbíraných dat lze vypočítat čas režie pro každý útok, na základě známé hodnoty efektivity útoku (viz sekci 3.5). Hodnotu celkové režie poté podělit vygenerovaným počtem *workunit* a získat tak čas režie na jednu *workunit*. Z naměřených dat jsem vypočítala průměrnou dobu režie na *workunit*, tj. 37 sekund. Tento postup je možné aplikovat pouze pokud je připojený jeden uzel. Při větším počtu uzlu by tento výpočet nedával smysl, jelikož nejsou při více připojených uzlech části úlohy zasílány sériově, ale paralelně. Možný přístup určení času režie na *workunit* spočívá v zavedení různých hodnot režie do nového navrženého vzorce a následně sledovat rozdíl mezi odhadovaným časem a reálným časem výpočtu. Pro hodnotu režie na *workunit* by tedy byla vybrána ta hodnota, při jejímž použití by byl rozdíl mezi odhadovaným a reálným časem nejmenší. Tabulky s daty k odhadu režie se nachází v příloze A.

Režie útoku hrubou silou

Tabulky s odhadovanou režií pro útok hrubou silou se nachází v příloze A.1. Z nasbíraných dat pomocí skriptu `attacksLauncher.py` jsem pro útok hrubou silou získala výsledky zobrazené v tabulce A.1. Různé časy režie jsem vybírala na základě předem vypočtené hodnoty

průměrné režie na *workunit*. Byly tedy vybrány hodnoty v rozmezí 20 sekund od průměrného času režie, přesněji hodnoty 15, 20, 30, 35, 40, 45, 50, 55 a 60. Tyto hodnoty byly použité i při odhadu režie na jednu *workunit* pro zbylé měřené typy útoků.

Řádek v tabulkách A pojmenovaný *Rozdíl* určuje průměrné procentuální rozdíly mezi reálným a odhadovaným časem ve srovnání s reálným na základě použité hodnoty času režie. První sloupec v tabulce znázorňuje procentuální rozdíl mezi reálným a odhadovaným časem původního vzorce. Měření probíhalo na jednom, dvou, čtyřech a osmi uzlech. Nejlepší výsledky pro jednotlivé počty uzlů a počet sekund na režii:

- 1 uzel: pro 35 sekund na režii se odhadovaný čas od reálného liší o 11.82 % z původních 36.12 % (viz tabulku A.1),
- 2 uzly: pro 30 sekund na režii se liší o 8.54 % z původních 52.98 % (viz tabulku A.2),
- 4 uzly: pro 30 sekund na režii se liší o 9.51 % z původních 69.41 % (viz tabulku A.3),
- 8 uzlů: pro 20 sekund na režii se liší o 14.64 % z původních 72.08 % (viz tabulku A.4).

Z výsledků lze vypočítat průměrnou hodnotu režie, pro kterou upravený vzorec nejlépe aproximuje reálný čas a následně tuto hodnotu zaokrouhlit na nejbližší měřený čas režie. Pro útok hrubou silou tedy použiji 30 sekund na režii.

Režie slovníkového útoku

Tabulky s daty pro výpočet režie slovníkového útoku se nachází v příloze A.2. Použití slovníkového útoku způsobuje vyšší hodnotu režie jedné *workunit* oproti ostatním měřeným útokům. Nejlepší výsledky pro jednotlivé počty uzlů a počet sekund na režii:

- 1 uzel: pro 50 sekund na režii se odhadovaný čas od reálného liší o 18.88 % z původních 72.77 % (viz tabulku A.5),
- 2 uzly: pro 60 sekund na režii se liší o 12.84 % z původních 66.70 % (viz tabulku A.6),
- 4 uzly: pro 55 sekund na režii se liší o 11.10 % z původních 65.42 % (viz tabulku A.7),
- 8 uzlů: pro 55 sekund na režii se liší o 10.77 % z původních 65.29 % (viz tabulku A.8).

Z nasbíraných dat vychází, že nový vzorec nejpřesněji aproximuje reálný čas při použití 55 sekund na režii.

Režie kombinačního útoku

Výsledky výpočtů z nasbíraných dat pro kombinační útok se nachází v příloze A.3. Nejlepší průměrné procentuální rozdíly pro jednotlivé počty uzlů a počet sekund na režii:

- 1 uzel: pro 15 sekund na režii se odhadovaný čas od reálného liší o 13.13 % z původních 46.00 % (viz tabulku A.9),
- 2 uzly: pro 30 sekund na režii se liší o 13.47 % z původních 39.90 % (viz tabulku A.10),
- 4 uzly: pro 30 sekund na režii se liší o 14.46 % z původních 33.98 % (viz tabulku A.11),
- 8 uzlů: pro 30 sekund na režii se liší o 8.39 % z původních 33.09 % (viz tabulku A.12).

Z naměřených dat pro aproximaci reálné doby lámání vychází nejlépe použití 30 sekund pro režii kombinačního útoku.

Režie hybridních útoků

Tabulky, které náleží měření režie hybridních útoků se nachází v přílohách A.4 a A.5.

Nejlepší průměrné procentuální rozdíly pro jednotlivé počty uzlů a počet sekund na režii při použití hybridního útoku se slovníkem vlevo a maskou vpravo:

- 1 uzel: pro 15 sekund na režii se rozdíl mezi odhadovaným časem a reálným zvýšil z původních 70.59 % na 92.39 % (viz tabulku A.13),
- 2 uzly: pro 15 sekund na režii se odhadovaný čas od reálného liší o 56.85 % z původních 59.09 % (viz tabulku A.14),
- 4 uzly: pro 20 sekund na režii se liší o 10.61 % z původních 29.70 % (viz tabulku A.15),
- 8 uzlů: pro 20 sekund na režii se liší o 7.24 % z původních 23.52 % (viz tabulku A.16).

Z prezentovaných výsledků lze spočítat, že použití 20 sekund pro režii v upraveném vzorci aproximuje reálný čas hybridního útoku se slovníkem vlevo a maskou vpravo nejlépe. Výsledky měření druhého typu hybridního útoku se nachází v příloze A.5. Nejlepší průměrné procentuální rozdíly pro jednotlivé počty uzlů a počet sekund na režii při použití hybridního útoku s maskou vlevo a slovníkem vpravo:

- 1 uzel: pro 15 sekund na režii se odhadovaný čas od reálného liší o 13.13 % z původních 46.00 % (viz tabulku A.9),
- 2 uzly: pro 30 sekund na režii se liší o 13.47 % z původních 39.90 % (viz tabulku A.10),
- 4 uzly: pro 30 sekund na režii se liší o 14.46 % z původních 33.98 % (viz tabulku A.11),
- 8 uzlů: pro 30 sekund na režii se liší o 8.39 % z původních 33.09 % (viz tabulku A.12).

Z nasbíraných dat vychází nejlepší výsledky aproximace reálného času při hodnotě režie 30 sekund.

Výsledky analýzy dat získaných prostřednictvím měření režie pro jednu *workunit* ukázaly, že optimální hodnota režie pro útok hrubou silou, kombinační útok a hybridní útok s maskou vlevo a slovníkem vpravo je 30 sekund. Pro hybridní útok se slovníkem vlevo a maskou vpravo byl zvolen čas 20 sekund a pro slovníkový útok byl zvolen čas 55 sekund. Hodnoty režie pro jednotlivé typy útoků jsou použity v algoritmu 4 (viz sekci 5.3) pro výpočet odhadovaného času lámání pro jednotlivé uzly. Algoritmus počítá odhadovaný čas lámání na základě času samotného lámání, času režie, odhadovaného počtu *workunit* a průměrného času potřebného k výpočtu výkonu uzlu.

5.3 Algoritmy pro výpočet odhadované doby lámání

Na základě zjištěných poznatků jsem sestavila algoritmus 2, algoritmus 3, algoritmus 5, 6 a algoritmus 4.

Algoritmus 2 je inspirován algoritmem 1 pro adaptivní plánování, který je blíže popsaný v sekci 3.1.1. Algoritmus na základě vstupních parametrů vypočte odhadovanou dobu trvání *workunit*. Na rozdíl od algoritmu 3.1.1 neobsahuje vstupní parametr t_J, který udává uběhlý čas od spuštění úlohy. V tomto případě by hodnota t_J byla 0 a mohlo by dojít k vynulování odhadované doby trvání *workunit*, proto jsem tento parametr nezahrnula do výpočtu.

Algoritmus 2 Výpočet t_p pro odhad maximální doby trvání výpočtu

Vstup: t_{min}, t_{max}, s_R, $powers$, α, β
Výstup: t_p

1: $v_{sum} = sum(powers)$
2: $t_p = (s_R / v_{sum}) \cdot \alpha$
3: $t_{prealmin} = max(t_{min}, t_{max} \cdot \beta)$
4: **if** $t_p < t_{prealmin}$ **then**
5: $\quad t_p = t_{prealmin}$
6: **else**
7: $\quad t_p = min(t_p, t_{max})$
8: **end if**
9: **return** t_p

Vstupními parametry algoritmu 2 jsou t_{min}, t_{max}, s_R, $powers$, α a β. Hodnota t_{min} udává nejmenší možnou velikost *workunit*. Parametr α je tzv. distribuční koeficient, který určuje jaká maximální část prostoru kandidátních hesel bude přidělena uzlu. Pokud je například koeficient nastaven na hodnotu 0.1, uzlu je přiděleno 10 % z celkového prostorou kandidátních hesel. Parametr β je tzv, *ramp up* koeficient. Zaručuje, že maximální přidělený počet kandidátních hesel uzlu nebude větší než maximální velikost *workunit* násobená parametrem β. Hodnoty všech těchto parametrů lze nastavit v pokročilém nastavení webové aplikace *WebAdmin* pro vzdálenou správu systému Fitcrack. Implicitně jsou hodnoty těchto parametrů nastaveny následovně: $t_{min} = 20$, $\alpha = 0.1$ a $\beta = 0.25$. Parametr t_{max} určuje maximální velikost *workunit*. Lze jej nastavit při konfiguraci úlohy v poslední části v poli *Desired time per workunit*. Implicitně je hodnota t_{max} nastavena na jednu hodinu. Parametr s_R je celkový počet kandidátních hesel spočtený pro danou úlohu a parametr *powers* je pole výpočetních výkonů všech uzlů podílejících se na lámání. Výstupní parametr t_p udává vypočtenou velikost *workunit*. Jeho hodnota je omezena parametry $t_{prealmin}$ a t_{max}.

Řádek 1 v algoritmu 2 se věnuje výpočtu celkové výpočetní síly všech uzlů podílejících se na lámání. Na řádku 2 se nachází výpočet počáteční hodnoty t_p a na řádku 3 je výpočet $t_{prealmin}$, což je spodní ohraničení t_p. Řádky 4 až 9 jsou věnovány určení t_p na základě hodnot $t_{prealmin}$, t_{max} a počátečního výpočtu t_p.

Algoritmus 3 určuje počet kandidátních hesel, který je přiřazený uzlu na základě hodnoty t_p vypočtené v algoritmu 2 a výkonu daného uzlu. Vstupními parametry algoritmu 3 jsou t_p, $powers$ a $totalKeyspace$. Parametr t_p je odhadovaná doba trvání *workunit*, $powers$ je pole výpočetních výkonů všech uzlů podílejících se na úloze a $totalKeyspace$ je celkový počet kandidátních hesel určených k verifikaci. Výstupní parametr $hostKeyspaces$ je pole polí ve formátu `[[keyspace, power, numWu]]`, kde `keyspace` je počet kandidátních hesel přiřazených uzlu, `power` je výpočetní síla tohoto uzlu a `numWu` je počet *workunit*, které byly uzlu přiřazeny. Algoritmus 3 přiřazuje každému uzlu pouze jednu *workunit*.

Algoritmus 3 pro každý uzel, který se podílí na lámání a je známa jeho výpočetní síla určí na základě hodnoty t_p počet kandidátních hesel, které bude tento uzel lámat. Pokud by tato hodnota přesáhla celkový počet kandidátních hesel, tak je uzlu celý tento prostor hesel přiřazen. Počet kandidátních hesel, výpočetní výkon a číslo jedna jsou uloženy do pole, které je poté přidáno do pole $hostKeyspaces$.

Algoritmus 4 slouží k výpočtu maximální odhadované doby lámání. Vstupními parametry jsou $compNodes$ a $wuOverhead$. Parametr $compNodes$ je pole polí získané algoritmem 5, 6 ve formátu `[[keyspace, power, numWu]]`, kde `keyspace` je počet kandidátních

Algoritmus 3 Stanovení počtu kandidátních hesel pro jednotlivé uzly

Vstup: t_p, $powers$, $totalKeyspace$
Výstup: $hostKeyspaces$

```
1: hostKeyspaces = [ ]
2: for power in powers do
3:     keyspace = min(tp · power, totalKeyspace)
4:     hostKeyspaces.append([keyspace, power, 1])
5: end for
6: return hostKeyspaces
```

hesel, které jsou uzlu přiděleny pomocí algoritmu 5 a 6, `power` je výpočetní síla tohoto uzlu a `numWu` je počet *workunit* přiřazených danému uzlu. Toto pole obsahuje pouze uzly, které se budou reálně podílet na výpočtu. Pokud by k úloze, ve které je potřeba verifikovat malé množství kandidátních hesel, byl například přiřazen jeden uzel s velkým výpočetním výkonem a 4 uzly s malým výkonem, je velmi pravděpodobné, že nejvýkonnější uzel obdrží celý prostor kandidátních hesel. Tudíž se na řešení úlohy nebudou podílet všechny uzly, které k ní byly přiřazeny. Parametr *wuOverhead* znázorňuje čas režie, který vznikne jednou *workunit*, hodnota režie se liší pro různé typy útoků.

Algoritmus 4 Určení maximální odhadované doby lámání

Vstup: $compNodes$, $wuOverhead$
Výstup: $maxEstimated$

```
1: times = [ ]
2: for node in compNodes do
3:     compTime = node.keyspace/node.power
4:     numWu = node.numWu + 1
5:     times.append(compTime + numWu · wuOverhead + 34)
6: end for
7: maxEstimated = max(times)
8: return maxEstimated
```

Výstupem algoritmu 4 je maximální hodnota z odhadovaných dob lámání všech uzlů. Tato hodnota tedy představuje výslednou odhadovanou dobu lámání zadané úlohy. Řádky 2 až 6 znázorňují výpočet odhadovaného času pro každý uzel. Hodnota *compTime* znázorňuje čistý čas lámání a hodnota *numWu* počet *workunit*, které uzel obdrží. Odhadovaný čas je poté spočítán jako suma čistého času lámání, počtu *workunit* násobeného režií na jednu *workunit* a konstanty 34, která znázorňuje průměrný čas trvání úlohy *benchmark*.

Poslední implementovaný algoritmus je rozdělený na dvě části. Pro první část viz algoritmus 5 a pro druhou část algoritmus 6. Tento algoritmus určuje uzly, které se budou reálně podílet na lámání, počet kandidátních hesel, který jim bude přidělen a počet přidělených *workunit*. Vstupními parametry jsou pole polí *hostKeys* ve formátu `[[keyspace, power, numWu]]`, který je již popsán u algoritmu 3 a celkový počet kandidátních hesel *hostKeys*.

První část algoritmu (viz algoritmus 5) určuje uzly, které se budou podílet na výpočtu, pokud je suma hesel, které jsou uzly schopny prolomit v jedné *workunit* větší nebo roven celkovému počtu kandidátních hesel v úloze. Tento scénář znamená, že se na úloze nebudou podílet všechny připojené uzly nebo se budou podílet maximálně jednou. Řádky 4 až 11 podchycují případ, kdy počet hesel uzlu, které zvládne prolomit v jedné *workunit* je větší

než celkový počet kandidátních hesel. V tomto případě bude úlohu počítat pouze tento uzel. Řádky 12 až 30 postupně přidávají do pole *CompNodes* nejvýkonnější uzly, tj. uzly s největším počtem kandidátních hesel na jednu *workunit*. Po přidání každého uzlu je od celkového počtu kandidátních hesel odečten počet hesel, které zvládne validovat daný uzel. Tento uzel je poté odstraněn ze vstupního pole *hostKeys*. Proces přidávání uzlů končí v případě, že počet hesel uzlu je větší než počet zbylých hesel z celkového počtu. V tomto případě je uzlu přidělen zbytek kandidátních hesel, uzel je přidán do pole *CompNodes* a algoritmus je ukončen.

Algoritmus 5 Odhad uzlů, které se budou podílet na výpočtu, část 1.

Vstup: $hostKeys$, $totalKey$
Výstup: $CompNodes$

```
1:  CompNodes = [ ]
2:  sumNodeKey = sum(hostKeys.keyspace)
3:  if sumNodeKey > totalKey then
4:    if max(hostKeys.keyspace) ≥ totalKey then
5:      for host in hostKeys do
6:        if host.keyspace == max(hostKeys.keyspace) then
7:          CompNodes.append(host)
8:          return CompNodes
9:        end if
10:     end for
11:   end if
12:   while hostKeys and totalKey − max(hostKeys.keyspace) > 0 do
13:     x = max(hostKeys.keyspace)
14:     for host in hostKeys do
15:       if host.keyspace == x then
16:         CompNodes.append(host)
17:         hostKeys.remove(host)
18:         totalKey = totalKey − x
19:         y = max(hostKeys.keyspace)
20:         if totalKey − y ≤ 0 then
21:           for host in hostKeys do
22:             if host.keyspace == y then
23:               CompNodes.append([totalKey, host.power, 1])
24:               return CompNodes
25:             end if
26:           end for
27:         end if
28:       end if
29:     end for
30:   end while
31: end if
32: # Pro druhou polovinu algoritmu viz algoritmus 6.
```

Algoritmus 6 Odhad uzlů, které se budou podílet na výpočtu, část 2.

```
1: if sumNodeKey > totalKey then
2:   # Pro první polovinu algoritmu viz algoritmus 5.
3: else
4:   CompNodes = hostKeys
5:   sumKeys = sum(hostKeys.keyspace)
6:   cutTotalKeyspace = totalKeyspace − sumKeys
7:   while cutTotalKeyspace − sumKeys ≥ 0 do
8:     for host, i in CompNodes do
9:       CompNodes.append([host.keyspace + hostKeys[i].keyspace, host.power,
         host.numWu + 1])
10:    end for
11:    cutTotalKeyspace = cutTotalKeyspace − sumKeys
12:  end while
13:  if cutTotalKeyspace − sumKeys < 0 then
14:    while hostKeysandcutTotalKeyspace − max(hostKeys.keyspace) do
15:      x = max(hostKeys.keyspace)
16:      for host, i in hostKeys) do
17:        if host.keyspace == x then
18:          if cutTotalKeyspace − x ≤ 0 then
19:            CompNodes[i] = [CompNodes[i].keyspace + cutTotalKeyspace,
               CompNodes[i].power, CompNodes[i].numWu + 1]
20:            return CompNodes
21:          else
22:            hostKeys.remove(host)
23:            CompNodes[i] = [CompNodes[i].keyspace + x, CompNodes[i].power,
               CompNodes[i].numWu + 1]
24:            cutTotalKeyspace = cutTotalKeyspace − x
25:          end if
26:        end if
27:      end for
28:    end while
29:    if cutTotalKeyspace − max(hostKeys) then
30:      x = max(hostKeys)
31:      for host, i in hostKeys do
32:        if host.keyspace == x then
33:          CompNodes[i] = [CompNodes[i].keyspace + cutTotalKeyspace,
             CompNodes[i].power, CompNodes[i].numWu + 1]
34:          return CompNodes
35:        end if
36:      end for
37:    end if
38:  end if
39: end if
40: return CompNodes
```

Cyklus na řádcích 7 až 12 v druhé části algoritmu 6 je vyhodnocen jako pravdivý pokud suma hesel, které jsou uzly schopny prolomit v jedné *workunit* je menší než celkový počet

kandidátních hesel. V těle tohoto cyklu je ke každému počtu kandidátních hesel uzlu v poli *CompNodes* připočten počet kandidátních hesel z pole *hostKeys* a počet *workunit* daného uzlu je zvýšen o jednu. Pokud je zbylý celkový počet kandidátních hesel menší než suma hesel všech uzlů, tak je v cyklu na řádcích 14 až 28 vybíráno z nejvýkonnějších uzlů a těm jsou přičítána v poli *CompNodes* kandidátní hesla, které budou verifikovat a počet *workunit* je vždy navýšen o jednu. Na řádcích 29 až 40 jsou uzlu s nejvyšším výkonem připočtena hesla k verifikaci, která zbyla z celkového počtu a algoritmus je ukončen.

Kapitola 6

Implementace navržených úprav v systému Fitcrack

Tato kapitola se zabývá implementací úprav v systému Fitcrack, které mají za účel aproximovat odhadovanou dobu lámání. Návrh těchto úprav je popsán v kapitole 5. Navržené úpravy v systému Fitcrack zasahují do částí `webadmin/fitcrackFE` (viz sekci 6.1) a `webadmin/fitcrackAPI` (viz sekci 6.2).

6.1 Úprava frontendu

Implementace algoritmů pro aproximaci doby lámání vyžadovala úpravu na straně frontendu, a to v souboru `webadmin/fitcrackFE/src/components/job/addJobView.vue`. Při vytváření úlohy se zobrazuje odhadovaná doba lámání. Pro výpočet odhadované doby lámání je zaslán `POST` požadavek na server na endpoint `/job/crackingTime` s parametry: `hash_list_id`, `boinc_host_ids` a `attack_settings`. Pro realizaci výpočtu odhadovaného času pomocí navržených algoritmů byl přidán parametr `workunit_time`, který znázorňuje maximální dobu trvání jedné `workunit`. Tento `POST` požadavek se nachází v bloku `watch`, který sleduje změny provedené v konfiguraci vytvářené úlohy a při každé změně odešle požadavek. Odpověď na tento požadavek je hodnota odhadované doby lámání a počet kandidátních hesel. Tyto hodnoty se uživateli zobrazují při vytváření nové úlohy.

6.2 Úprava backendu

Výpočet odhadovaného času lámání se nachází v systému Fitcrack na dvou místech. První místo výskytu je v adresáři `webadmin/fitcrackAPI/src/src/database` a druhý výskyt v adresáři `webadmin/fitcrackAPI/src/src/fitcrack/api/endpoints/job`. První výskyt v adresáři `/database` slouží k přepočtu odhadované doby lámání v průběhu lámání. Druhý výskyt v adresáři `/job` slouží k prvotnímu výpočtu odhadu před zahájením lámání.

6.2.1 Úprava výpočtu před zahájením lámání

V adresáři `/job` byla potřeba úprav v souborech `argumentsParser.py` a `functions.py`. Oba tyto soubory jsou napsány v jazyce `Python` s využitím rozšíření Flask-RESTX[1] knihovny

[1] https://flask-restx.readthedocs.io/en/latest/

Flask[2] pro vývoj a správu API a knihovny SQLAlchemy[3] pro práci s databází. V souboru `argumentsParser.py` byla nutná definice nového parametru `workunit_time`, který očekává API endoint v POST požadavku pro výpočet odhadované doby lámání. Parametr `workunit_time` znázorňuje maximální velikost `workunit` zadanou uživatelem při konfiguraci úlohy. Tento parametr byl přidán na frontendu do těla požadavku pro výpočet odhadované doby lámání (viz sekci 6.1) a tak je nutné jej přidat i na straně backendu do parametrů, které jsou v tomto požadavku očekávány.

V souboru `functions.py` jsem provedla změny ve funkci `computeCrackingTime`. Tato funkce obsluhuje endpoint `/job/crackingTime` a slouží pro výpočet počtu kandidátních hesel na základě vybraného typu útoku a pro výpočet odhadovaného času lámání. Mnou provedené změny se týkají části výpočtu odhadované doby lámání, kde jsem implementovala upravenou logiku výpočtu a využila algoritmy popsané v sekci 5.3.

V adresáři `job/` jsem vytvořila nový soubor `crackingTimeFunctions.py`, ve kterém se nachází implementace navržených algoritmů pro výpočet odhadované doby lámání (viz sekci 5.3). Funkce `adaptive_scheduling` implementuje algoritmus 2, funkce `compute_hostkeyspace_from_tp` implementuje algoritmus 3, funkce `which_node_will_compute` implementuje algoritmus 5 a 6 a funkce `compute_estimate_forall` implementuje algoritmus 4.

6.2.2 Úprava výpočtu v průběhu lámání

V adresáři `/database` jsem upravila soubor `models.py`. Tento soubor definuje třídy, které reprezentují jednotlivé tabulky v databázi. Třída `FcJob` reprezentuje tabulku `fc_job` v databázi. Tato třída obsahuje vlastnost `estimated_cracking_time_str`, která je v průběhu lámání přepočítávána. Pro výpočet této vlastnosti jsem importovala funkce ze souboru `crackingTimeFunctions.py`. Tento soubor obsahuje implementaci navržených algoritmů, které slouží k výpočtu odhadované doby lámání. Pomocí těchto funkcí jsem implementovala logiku pro výpočet odhadované doby v průběhu lámání.

[2] https://flask.palletsprojects.com/en/3.0.x/
[3] https://www.sqlalchemy.org/

Kapitola 7

Experimentální ověření přínosu úprav

Tato kapitola se zabývá vyhodnocením experimentů provedených po úpravách v systému Fitcrack. Cílem experimentů je zhodnotit účinnost mnou navrženého vzorce pro odhad času lámání hesel v porovnání se starým vzorcem. Primární zájem je tedy posoudit, zda nový vzorec poskytuje přesnější aproximaci reálného času lámání hesel než původní vzorec.

Experimenty byly provedené na počítačích v laboratoři Fakulty informačních technologií VUT v Brně. Počítače disponují operačním systémem `Microsoft Windows 11`, procesorem `AMD Ryzen 5 PRO 5650G` a grafickou kartou `GeForce GTX 1050 Ti`. Na těchto počítačích byly postupně spuštěny úlohy pro útok hrubou silou, slovníkový útok, kombinační útok a hybridní útoky.

Pro automatizaci spouštění úloh jsem vytvořila skript `attacksLauncher.py` vytvořený úpravou skriptu `superbench.py`. Soubor `attacksLauncher.py` je přiložen na datovém médiu ve složce `experimenty` (viz příloha B). Všechny sady úloh byly měřeny postupně na jednom, dvou, čtyřech a osmi připojených uzlech. Data z jednotlivých měření jsou dostupné na připojeném médiu (viz příloha B).

7.1 Experimenty pro útok hrubou silou

V této sekci jsou popsané experimenty provedené pro útok hrubou silou. Všechny experimenty byly spuštěné s hešovacím algoritmem MD5.

7.1.1 Sada úloh

Pro útok hrubou silou jsem vytvořila sadu úloh, která pokrývá různé případy možné konfigurace tohoto útoku. Sada devíti úloh, která pokrývá proměnlivou délku masek, různý počet masek a různé maximální velikosti *workunit*:

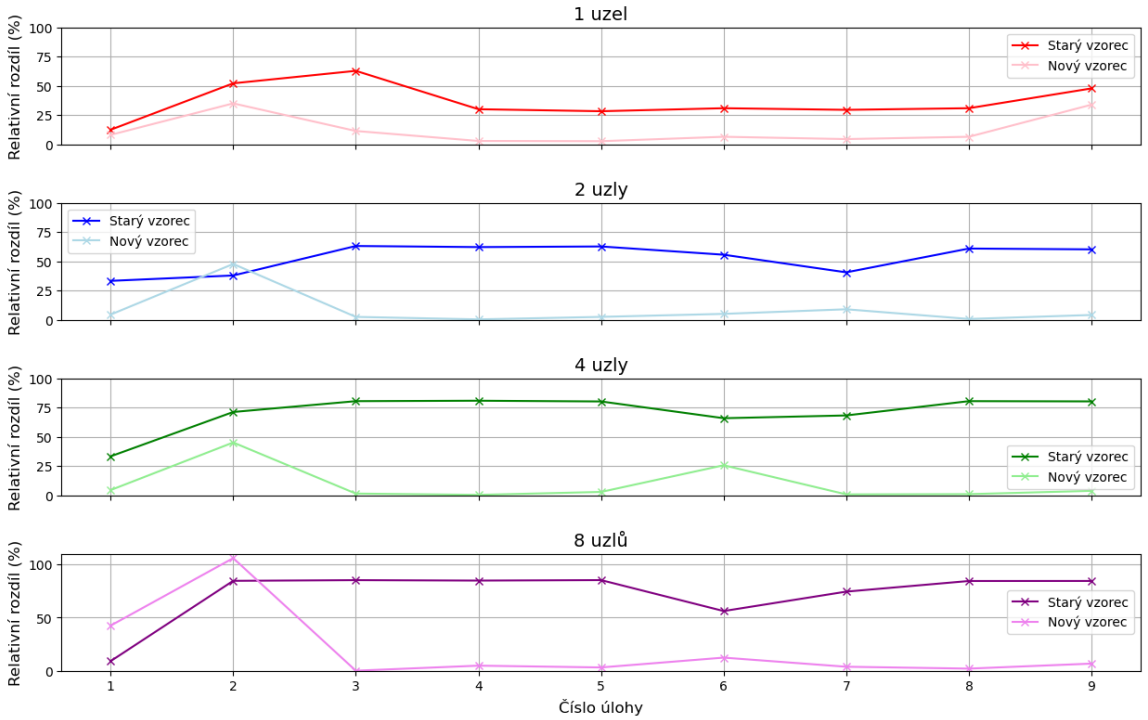
1. maska `?l?l?l?l?l?l` a maximální velikost *workunit* 3600 sekund,
2. maska `?l?l?l?l?l?l?l?d` a maximální velikost *workunit* 3600 sekund,
3. 1x maska `?l?l?l?l?l?l?l?l?d` a 1x maska `?l?l?l?l?l?l?l` a maximální velikost *workunit* 3600 sekund,
4. 1x maska `?l?l?l?l?l?l?l?l?d` a 3x maska `?l?l?l?l?l?l?l` a maximální velikost *workunit* 3600 sekund,

5. 1x maska `?l?l?l?l?l?l?l?l?d` a 7x maska `?l?l?l?l?l?l?l` a maximální velikost *workunit* 3600 sekund,
6. maska `?l?l?l?l?l?l?l?l?d` a maximální velikost *workunit* 60 sekund,
7. maska `?l?l?l?l?l?l?l?l?d` a maximální velikost *workunit* 600 sekund,
8. maska `?l?l?l?l?l?l?l?l?d` a maximální velikost *workunit* 1200 sekund,
9. maska `?l?l?l?l?l?l?l?l?d` a maximální velikost *workunit* 1800 sekund.

Tato sada úloh má za účel otestovat funkčnost mnou navrženého vzorce při různém nastavení a proměnlivém počtu připojených uzlů.

7.1.2 Výsledky experimentů

Výsledky jednotlivých experimentů provedených na úlohách pro útok hrubou silou jsou vynesené v grafech na obrázku 7.1. Každý z grafů znázorňuje relativní rozdíl mezi odhadovaným časem spočteným původním vzorcem a reálným časem. Tato data jsou znázorněna sytějšími barvami a označena v legendě ke grafům popisem *Starý vzorec*. Popisem *Nový vzorec* jsou označeny hodnoty relativního rozdílu mezi odhadovaným a reálným časem při použití mnou navrženého vzorce.



Obrázek 7.1: Grafy relativních rozdílů mezi odhadovaným a reálným časem pro útok hrubou silou

Čím blíže je hodnota relativního rozdílu 0 %, tím přesněji byla aproximována reálná doba lámání. Výsledky vynesené v grafech prokazují zpřesnění odhadu v 91.67 % z 36 spuštěných úloh. Z analýzy dat z experimentů lze vypočítat, že průměrně došlo ke zpřesnění odhadu

spočteného novým vzorcem o 45.44 procentních bodů od odhadu vypočteného původním vzorcem. Zpřesnění odhadu bylo při úlohách s více maskami (úlohy 3 až 5) a různými velikostmi *workunit* (úlohy 6 až 9) stoprocentní. Neúspěch se projevil dvakrát ve druhé úloze a jednou v první. Možná příčina nepřesného odhadu může vycházet z chybně naměřeného výkonu uzlů při první úloze a následném použití těchto hodnot pro odhad v druhé úloze. První úloha obsahovala menší počet kandidátních hesel a nástroj Hashcat má tendenci při malém počtu hesel ohodnotit uzel menším výkonem, než je jeho reálný.

7.2 Experimenty pro slovníkový útok

V této sekci je popsaná sada úloh pro slovníkový útok, která má za úkol otestovat funkcionalitu navrženého vzorce pro odhad a určit, zda při jeho použití dojde ke zlepšení aproximace reálné doby lámání. Dále sekce obsahuje vyhodnocení experimentů. Úlohy byly spuštěny s hešovacím algoritmem MD5.

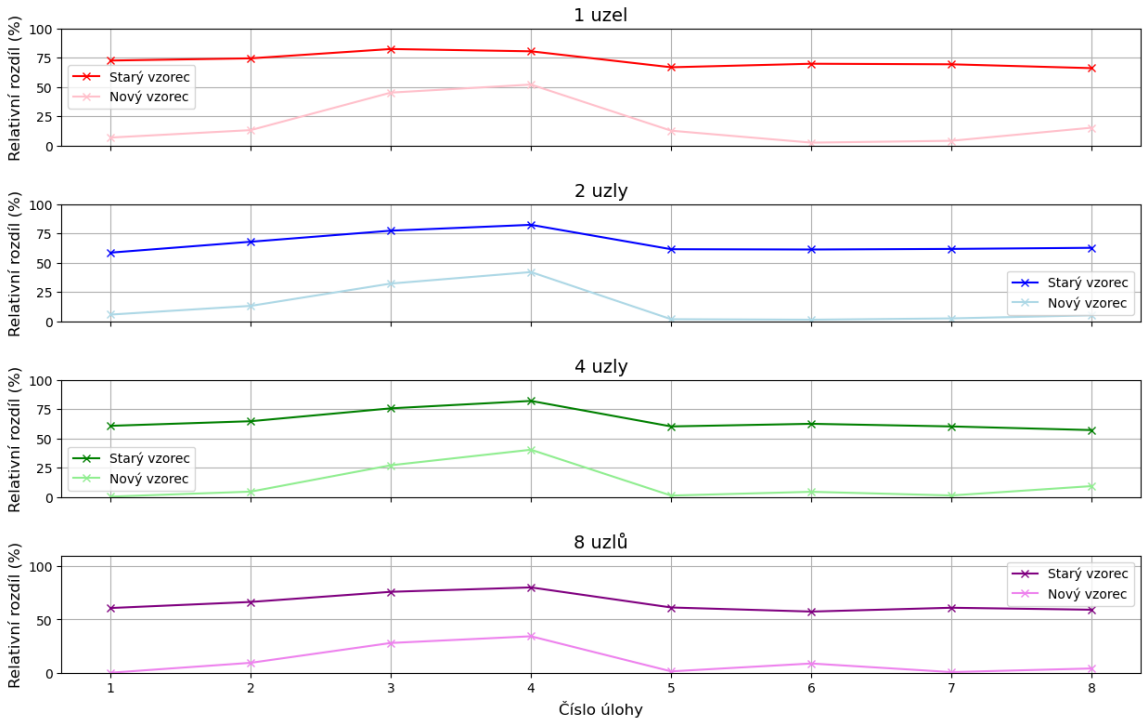
7.2.1 Sada úloh

Sada úloh pro slovníkový útok obsahuje úlohy zaměřující se na různý počet slovníků připojených k úloze a odlišné velikosti *workunit*. Sada osmi úloh vytvořených pro slovníkový útok:

1. 1x slovník a maximální velikost *workunit* 3600 sekund,
2. 2x slovník a maximální velikost *workunit* 3600 sekund,
3. 4x slovník a maximální velikost *workunit* 3600 sekund,
4. 6x slovník a maximální velikost *workunit* 3600 sekund,
5. 1x slovník a maximální velikost *workunit* 60 sekund,
6. 1x slovník a maximální velikost *workunit* 600 sekund,
7. 1x slovník a maximální velikost *workunit* 1200 sekund,
8. 1x slovník a maximální velikost *workunit* 1800 sekund.

7.2.2 Výsledky experimentů

Na obrázku 7.2 jsou v grafech znázorněné výsledky jednotlivých experimentů založených na sadě úloh. Grafy zobrazují relativní rozdíly mezi reálným a odhadovaným časem na základě reálného pro původní vzorec a mnou navržený vzorec. Z grafů lze vyčíst, že zlepšení aproximace reálné doby lámání je pro danou sadu úloh 100%. Nově navržený vzorec odhaduje dobu lámání velmi efektivně. U většiny úloh je relativní rozdíl pod 10 % a blíží se nule. Výjimkou jsou úlohy 3 a 4, kde se relativní rozdíl pohybuje mezi 25 % a 50 %. Úloha 3 obsahuje čtyři slovníky a úloha 4 obsahuje šest slovníků, je tedy možné předpokládat, že mnou navržený vzorec hůře aproximuje úlohy s více slovníky. Stále je u těchto úloh znatelné zlepšení odhadu oproti původnímu vzorci. Ze získaných dat lze vypočítat, že v průměru došlo při použití nového vzorce ke zlepšení o 54.09 procentních bodů od odhadu starým vzorcem.



Obrázek 7.2: Grafy relativních rozdílů mezi odhadovaným a reálným časem pro slovníkový útok

7.3 Experimenty pro kombinační útok

Tato sekce obsahuje popis sady úloh pro kombinační útok a vyhodnocení experimentů. Pro úlohy byl použit hešovací algoritmus MD5.

7.3.1 Sada úloh

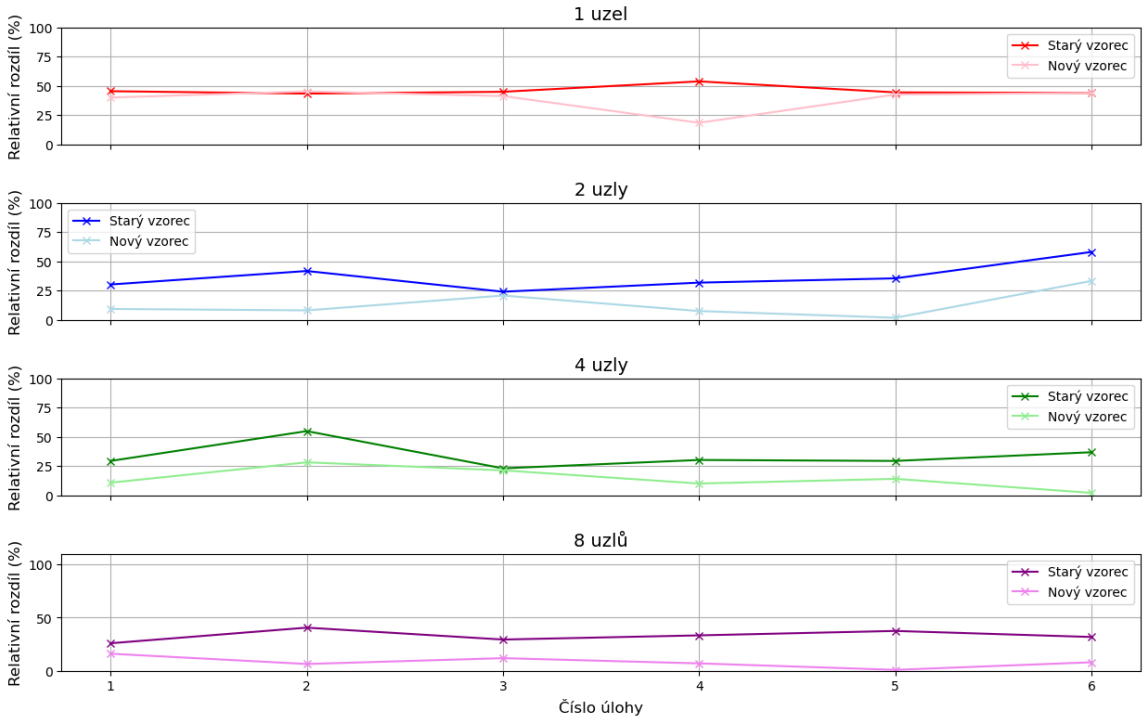
Sada úloh vytvořená pro kombinační útok sleduje chování mnou navrženého vzorce a původního vzorce při různé velikosti *workunit* a při odlišném počtu levých a pravých slovníků.

1. 1x pravý slovník, 1x levý slovník a maximální velikost *workunit* 60 sekund,
2. 1x pravý slovník, 1x levý slovník a maximální velikost *workunit* 600 sekund,
3. 1x pravý slovník, 1x levý slovník a maximální velikost *workunit* 1200 sekund,
4. 1x pravý slovník, 1x levý slovník a maximální velikost *workunit* 1800 sekund,
5. 2x pravé slovníky, 1x levý slovník a maximální velikost *workunit* 60 sekund,
6. 1x pravý slovník, 2x levé slovníky a maximální velikost *workunit* 60 sekund.

7.3.2 Výsledky experimentů

Výsledky experimentů pro kombinační útok jsou zaznačeny v grafech 7.3 pro různé počty připojených uzlů. Z grafů je viditelné, že pro kombinační útok nedošlo k výraznému zlepšení

odhadu oproti původnímu. V průměru došlo ke zlepšení o 18.76 procentních bodů oproti starému vzorci. Odhad pomocí nového vzorce byl v 91.67 % blíže k reálnému času než odhad spočten původním vzorcem a relativní rozdíl od reálného času se nachází u poloviny úkolů pod hranicí 15 %. Z grafů lze vyčíst, že nejhůře nový vzorec odhaduje dobu lámání pro jeden připojený uzel. Průměrně došlo při odhadu pro jeden připojený uzel ke zlepšení pouze o 7.41 procentních bodů od odhadu původním vzorcem. Výraznější zlepšení od původního odhadu vykazuje pouze úloha 4. Společný parametr všech úloh, které nevykázaly zásadní zlepšení je velikost *workunit* nastavená na 1200 sekund a méně. Toto nastavení nevykazuje snížení úspěšnosti odhadu pomocí nového vzorce při připojení více uzlů.



Obrázek 7.3: Grafy relativních rozdílů mezi odhadovaným a reálným časem pro kombinační útok

7.4 Experimenty pro hybridní útoky

V této sekci je popsána sada úloh pro dva typy hybridních útoků, a to pro útok se slovníkem vlevo a maskou vpravo a pro útok s maskou vlevo a slovníkem vpravo. Princip hybridních útoků je vysvětlen v sekci 3.2.7. Tato sekce také popisuje výsledky experimentů provedených s cílem zjistit, zda nově navržený vzorec pro odhad času lámání hesel aproximuje reálný čas lépe než původní vzorec. Experimenty byly zaměřeny na porovnání relativních rozdílů mezi odhadovaným a reálným časem lámání hesel pomocí obou vzorců. Distribuce mezi připojené uzly probíhá pro každý typ útoku jiným způsobem. Cílem analýzy výsledků experimentů je také určení, zda je nový vzorec vhodný pro použití při hybridních útocích a zda vykazuje rozdílné chování při různém typu hybridních útoků.

7.4.1 Sady úloh

Sady úloh pro oba typy hybridních útoků jsou zaměřené na odhad času při zadání různých velikostí *workunit*, tudíž byly použity stejné slovníky a masky. Všechny úlohy v těchto sadách byly spuštěny s hešovacím algoritmem MD5. Sada úloh pro hybridní útok se **slovníkem vlevo** a **maskou vpravo**:

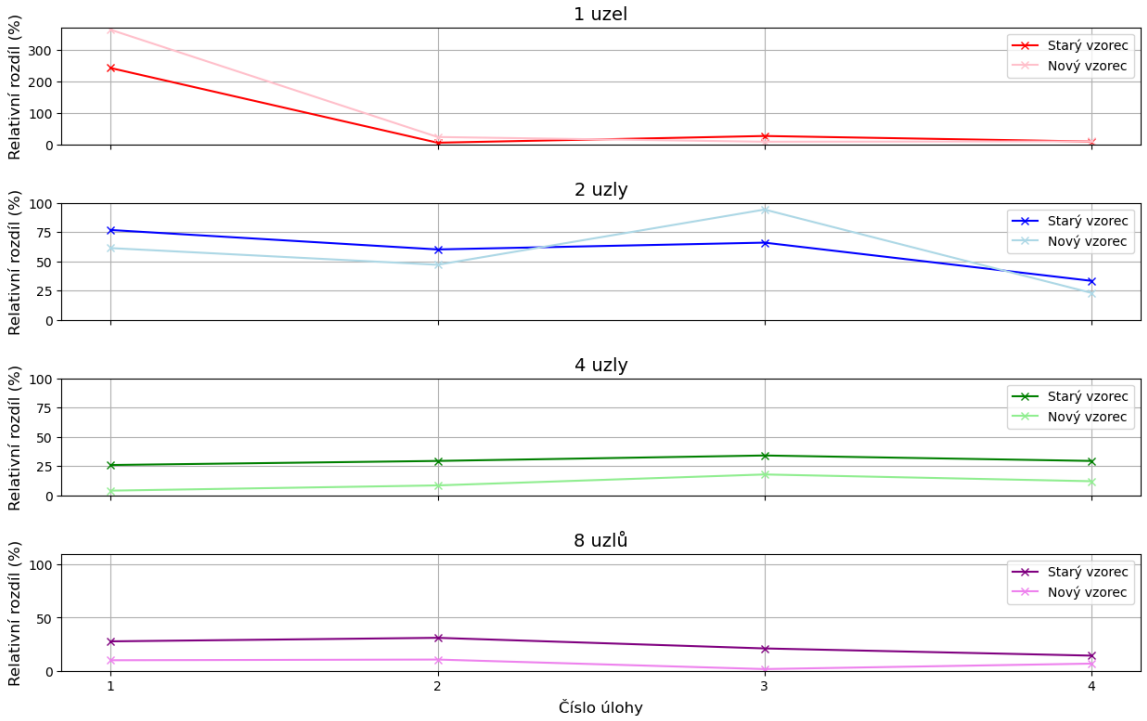
1. slovník vlevo, maska `?l?l?l?l` a maximální velikost *workunit* 60 sekund,
2. slovník vlevo, maska `?l?l?l?l` a maximální velikost *workunit* 600 sekund,
3. slovník vlevo, maska `?l?l?l?l` a maximální velikost *workunit* 1200 sekund,
4. slovník vlevo, maska `?l?l?l?l` a maximální velikost *workunit* 1800 sekund.

Sada úloh pro hybridní útok s **maskou vlevo** a **slovníkem vpravo**:

1. maska `?l?l?l?l`, slovník vpravo a maximální velikost *workunit* 60 sekund,
2. maska `?l?l?l?l`, slovník vpravo a maximální velikost *workunit* 600 sekund,
3. maska `?l?l?l?l`, slovník vpravo a maximální velikost *workunit* 1200 sekund,
4. maska `?l?l?l?l`, slovník vpravo a maximální velikost *workunit* 1800 sekund.

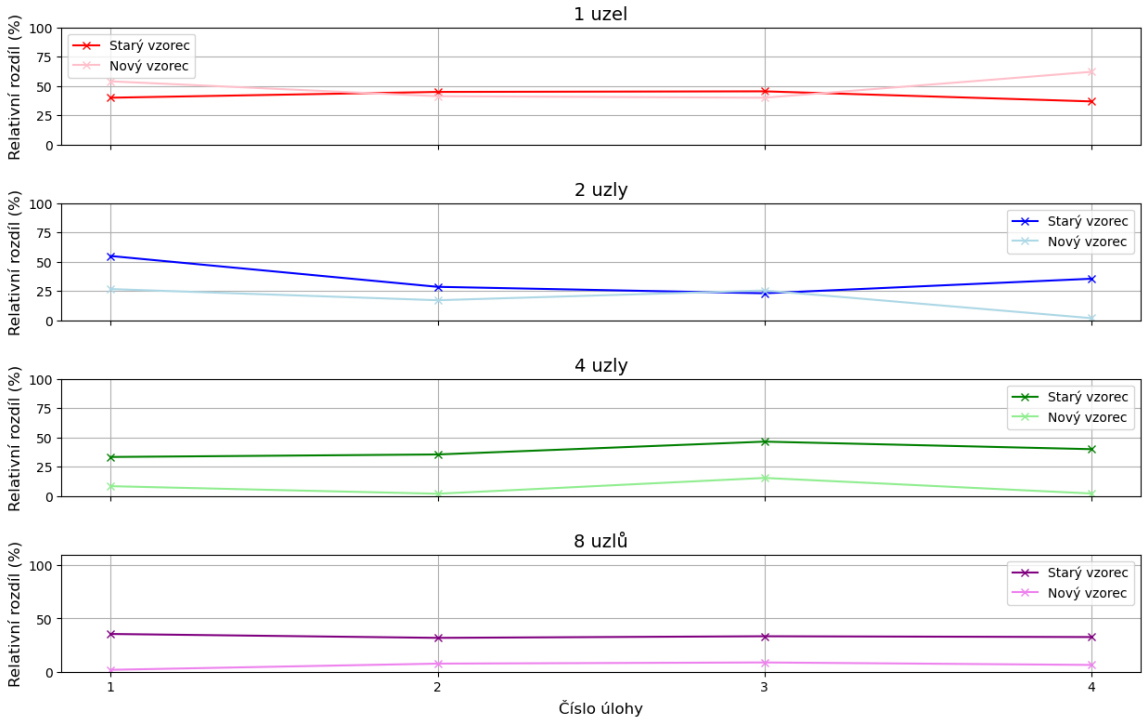
7.4.2 Výsledky experimentů

Výsledky experimentů pro hybridní útoky jsou vyneseny v grafech na obrázcích 7.4 a 7.5. K typu hybridního útoku se slovníkem vlevo a maskou vpravo se vztahuje obrázek 7.4 a k typu s maskou vlevo a slovníkem vpravo se vztahuje obrázek 7.5.



Obrázek 7.4: Grafy relativních rozdílů mezi odhadovaným a reálným časem pro hybridní útok se slovníkem vlevo a maskou vpravo

Z výsledků experimentů pro hybridní útok se slovníkem vlevo a maskou vpravo (viz obrázek 7.4) vyplývá, že došlo ke zlepšení odhadu v 75 % měřených úloh. Průměrné zlepšení odhadu oproti původnímu vzorci činí pouze 1.92 procentních bodů. Hlavní příčinou této situace je výpočet odhadované doby lámání pro jeden připojený uzel, kde došlo k výraznému zhoršení výsledků o 30.37 procentních bodů oproti původnímu vzorci. Nicméně v ostatních případech dochází k pozitivnímu zlepšení. Pozoruhodným je také trend, který naznačuje zvýšení relativního rozdílu při vytváření menších *workunit*. S narůstající velikostí *workunit* se relativní rozdíl postupně snižuje a přibližuje se k 0 %, což naznačuje, že odhad je přesnější a blíže odpovídá reálnému času lámání hesla. Je pravděpodobné, že mnou navržený vzorec pro odhad má potenciál k dalšímu zdokonalení, zejména v oblasti odhadů pro jeden připojený uzel a při vytváření menších *workunit*.



Obrázek 7.5: Grafy relativních rozdílů mezi odhadovaným a reálným časem pro hybridní útok s maskou vlevo a slovníkem vpravo

Při analýze výsledků experimentů pro hybridní útok s maskou vlevo a slovníkem vpravo (viz obrázek 7.5) bylo zjištěno zlepšení odhadu pomocí mnou navrženého vzorce v 81.25 % případů. Průměrně došlo k zpřesnění odhadu o 17.30 procentních bodů ve srovnání s původním vzorcem. Pro tento typ hybridního útoku je tedy výpočet odhadu pomocí nového vzorce přesnější. Stejně jako u opačného typu hybridního útoku, i zde došlo k zhoršení přesnosti odhadu při použití jediného připojeného uzlu, a to o 7.54 procentních bodů ve srovnání s původním vzorcem. Nicméně zde není pozorován trend zpřesňování odhadu s rostoucí velikostí *workunit*.

7.5 Zhodnocení výsledků experimentů

Výsledky experimentů prokazují, že nově navržený vzorec pro odhad doby lámání hesel má potenciál vylepšit přesnost odhadu při různých typech útoků a jejich konfiguraci. V případě útoku hrubou silou došlo ke zpřesnění odhadu v 91.67 % z 36 zadaných úloh. Při měření slovníkového útoku bylo dosaženo výrazného zlepšení odhadu ve 100 % z 32 měřených úloh, přičemž nový vzorec dosáhl zlepšení o 54.09 procentních bodů od původního vzorce. Prostor pro zlepšení nového vzorce je u úloh obsahujících více slovníků. Pro kombinační útoky bylo zlepšení mírně nižší, ale stále významné. Odhad spočten novým vzorcem byl v 91.67 % přesnější, než odhad spočten původním vzorcem. Při měření hybridních útoků s maskou vlevo a slovníkem vpravo byla zaznamenána významná zlepšení odhadu, zejména při použití více připojených uzlů, konkrétně došlo ke zpřesnění odhadů v 81.25 % z 16 měřených úloh. Nicméně u hybridního útoku se slovníkem vlevo a maskou vpravo bylo zlepšení odhadu mírně nižší, ačkoli stále představovalo zpřesnění oproti původnímu vzorci.

V nejlepším případě došlo při použití nového vzorce k zpřesnění odhadované doby o 85.15 procentních bodů, a to u útoku hrubou silou s osmi připojenými uzly. V tomto případě nastal také nejpřesnější odhad s relativním rozdílem mezi odhadovaným a reálným časem pouhých 0.11 %. Celkově dosáhl nový vzorec relativního rozdílu menšího než 1 % v 7.25 % z měřených úloh. V 35 % případů dosáhl nový vzorec relativního rozdílu pod 5 % a v 66.13 % dosáhl relativního rozdílu pod 15 %. Při použití starého vzorce se pod relativní rozdíl 5 % nedostala žádná z měřených úloh a pod hranici 15 % se dostaly pouze 4 %.

Je důležité zdůraznit, že při použití nového vzorce se v některých případech objevily horší výsledky v porovnání s původním vzorcem, zejména při odhadu doby lámání pro jeden připojený uzel. To naznačuje potřebu dalšího vylepšení a ladění nového vzorce, aby se minimalizovaly případy zhoršení výsledků. Na druhou stranu nový vzorec velmi dobře odhaduje dobu lámání při více připojených uzlech. Tento fakt je způsoben hlavně tím, že nový vzorec odhaduje velikost vytvořených *workunit* a jejich rozdělení mezi uzly, dále také odhaduje uzly, které se budou podílet na výpočtu. Původní vzorec nezohledňoval možnost, že se na výpočtu nebudou podílet všechny k němu připojené uzly. Celkově lze tedy říci, že mnou navržený vzorec pro odhad času lámání hesel přináší významné zlepšení přesnosti odhadu oproti původnímu vzorci v mnoha případech a je vhodný pro použití v různých typech útoků, které implementuje systém Fitcrack.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo navrhnout změny v distribuovaném systému Fitcrack za účelem zlepšení přesnosti odhadu maximální doby lámání hesel. V rámci práce byla prostudována architektura systému Fitcrack spolu s nástroji Hashcat a BOINC, které využívá. Dále byly prostudovány implementované metody útoků a jejich distribuce výpočetním uzlům, techniky měření výkonu výpočetních uzlů a současný vzorec pro výpočet odhadované doby lámání.

Dále bylo provedeno několik měření úloh zadaných do systému Fitcrack s hlavním zaměřením na útok hrubou silou. Naměřené hodnoty byly zaznamenány do tabulky, na základě které byla následně provedena analýza s využitím korelační matice. Analýza umožnila identifikovat klíčové faktory ovlivňující přesnost odhadované doby lámání. Zjistila jsem, že stávající vzorec nezahrnuje čas režie spojený s komunikací mezi výpočetními uzly a serverem a také dobu potřebnou pro měření výkonu uzlů. Tyto faktory ovlivňují všechny typy útoků, které implementuje systém Fitcrack.

Následně jsem sestavila návrh upraveného vzorce pro výpočet odhadované doby lámání, který zohledňuje dobu režie a čas potřebný pro zjištění výkonu uzlu. Návrh zahrnuje vytvoření algoritmů, které mají za cíl odhadnout velikost *workunit* pro každý připojený uzel, poté odhadnout, které uzly se budou na výpočtu reálně podílet a jak dlouho jim zabere prolomit jim přidělený počet hesel.

Na základě návrhu jsem změny implementovala do systému Fitcrack a experimentálně jsem ověřila, zda tyto změny přinesly zpřesnění odhadu. Experimenty byly provedené pro útok hrubou silou, slovníkový útok, kombinační útok a hybridní útoky. Z výsledků experimentů lze stanovit, že mnou navržený vzorec pro odhad přinesl zlepšení v 90.32 % ze 124 měřených experimentů na jednom až osmi uzlech pro útok hrubou silou, slovníkový útok, kombinační útok a hybridní útoky. Největší zlepšení odhadované doby činilo 85.15 procentních bodů. Při použití nového vzorce se podařilo dosáhnout relativního rozdílu mezi odhadovanou a reálnou dobou lámání pod 15 % u 66.13 % z měřených úloh, zatímco při použití starého vzorce se pod hranici 15 % dostaly pouze 4 % z měřených úloh. Zároveň je třeba zdůraznit, že i přes dosažené zlepšení stále existuje prostor pro další inovace a optimalizace, zejména v oblasti útoků PRINCE a PCFG, které používají externí generátory, a kterých se úpravy v této práci nedotkly. V budoucnu je možné rozšířit návrh tak, aby zahrnoval i tyto typy útoků a maximalizoval tak úroveň přesnosti odhadu.

Literatura

[1] Anderson, D. P. Boinc: A system for public-resource computing and storage. In: IEEE. *Fifth IEEE/ACM international workshop on grid computing*. 2004, s. 4–10.

[2] Asuero, A. G., Sayago, A. a González, A. The correlation coefficient: An overview. *Critical reviews in analytical chemistry*. Taylor & Francis. 2006, sv. 36, č. 1, s. 41–59.

[3] Belshe, M., Peon, R. a Thomson, M. *Hypertext Transfer Protocol Version 2 (HTTP/2)* [RFC 7540 (Proposed Standard)]. Request for Comments (RFC) 7540. Internet Engineering Task Force (IETF), květen 2015. Dostupné z: http://www.ietf.org/rfc/rfc7540.txt.

[4] Bolvanský, D. *Lámání hesel pomocí algoritmu PRINCE v systému Fitcrack*. 2020. Diplomová práce. Fakulta informačních technologií VUT v Brně. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/129865.

[5] D. Eastlake, r. a Jones, P. *US Secure Hash Algorithm 1 (SHA1)*. Request for Comments (RFC) 3174. Network Working Group, September 2001. Dostupné z: https://www.ietf.org/rfc/rfc3174.txt.

[6] Hranický, R. *Digital Forensics: The Acceleration of Password Cracking*. 2021. Disertační práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[7] Hranický, R., Holkovič, M., Matoušek, P. a Ryšavý, O. On Efficiency of Distributed Password Recovery. *The Journal of Digital Forensics, Security and Law*. 2016, sv. 11, č. 2, s. 79–96. ISSN 1558-7215. Dostupné z: http://www.fit.vutbr.cz/research/view_pub.php?id=11276.

[8] Hranický, R., Zobal, L., Ryšavý, O. a Kolář, D. Distributed password cracking with BOINC and hashcat. *Digital Investigation*. 2019, sv. 2019, č. 30, s. 161–172. DOI: 10.1016/j.diin.2019.08.001. ISSN 1742-2876. Dostupné z: https://www.fit.vut.cz/research/publication/11961.

[9] Hranický, R., Zobal, L., Večeřa, V., Múčka, M., Horák, A. et al. *The architecture of Fitcrack distributed password cracking system, version 2*. Faculty of Information Technology BUT, 2020. 85 s. Dostupné z: https://www.fit.vut.cz/research/publication/12300.

[10] Rescorla, E. a Schiffman, A. *The Secure HyperText Transfer Protocol*. Request for Comments (RFC) 2660. Network Working Group, August 1999. Dostupné z: https://www.ietf.org/rfc/rfc2660.txt.

[11] Rivest, R. *The MD5 Message-Digest Algorithm* [RFC 1321 (Informational)]. Request for Comments (RFC) 1321. Internet Engineering Task Force (IETF), duben 1992. Updated by RFC 6151. Dostupné z: http://www.ietf.org/rfc/rfc1321.txt.

[12] Ženčák, T. *Optimalizace distribuce úloh v systému Fitcrack*. 2020. Diplomová práce. Fakulta informačních technologií VUT v Brně. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/129860.

Příloha A

Tabulky pro návrh času režie jedné *workunit*

Tato příloha obsahuje výčet tabulek s výsledky z naměřených hodnot pro určení doby režie na jednu *workunit* pro útok hrubou silou, slovníkový útok, kombinační útok a hybridní útoky. První řádek tabulek určuje počet sekund režie použitých pro výpočet odhadu a druhý řádek určuje relativní rozdíl v procentech mezi odhadovaným a reálným časem lámání. První údaj v tabulce určuje relativní rozdíl mezi odhadovanou dobou lámání a reálnou při použití původního vzorce.

A.1 Útok hrubou silou

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	36.12	16.18	14.71	12.41	11.82	12.92	15.84	19.32

Tabulka A.1: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro útok hrubou silou a jeden uzel.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	52.98	15.35	11.77	8.54	10.22	14.27	18.40	22.53

Tabulka A.2: Tabulka procentuálních rozdílů mezi odhadovaným časem a reálným časem ve srovnání s reálným pro útok hrubou silou a dva uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	69.41	15.70	12.50	9.51	12.73	16.17	20.33	24.49

Tabulka A.3: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro útok hrubou silou a čtyři uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	72.08	15.66	14.64	20.19	24.81	29.45	34.09	38.74

Tabulka A.4: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro útok hrubou silou a osm uzlů.

A.2 Slovníkový útok

Čas režie [s]	–	30	35	40	45	50	55	60
Rozdíl [%]	72.77	32.02	27.69	23.35	20.42	18.88	18.99	19.98

Tabulka A.5: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro slovníkový útok a jeden uzel.

Čas režie [s]	–	30	35	40	45	50	55	60
Rozdíl [%]	66.70	39.31	33.76	28.21	22.66	17.11	12.99	12.84

Tabulka A.6: Tabulka procentuálních rozdílů mezi odhadovaným časem a reálným časem ve srovnání s reálným pro slovníkový útok a dva uzly.

Čas režie [s]	–	30	35	40	45	50	55	60
Rozdíl [%]	65.42	36.85	31.10	25.33	19.57	14.34	11.10	11.83

Tabulka A.7: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro slovníkový útok a čtyři uzly.

Čas režie [s]	–	30	35	40	45	50	55	60
Rozdíl [%]	65.29	36.51	30.73	24.94	19.16	13.77	10.77	12.85

Tabulka A.8: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro slovníkový útok a osm uzlů.

A.3 Kombinační útok

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	46.00	13.13	20.59	38.59	47.59	56.59	65.59	74.59

Tabulka A.9: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro kombinační útok a jeden uzel.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	36.90	31.88	21.36	13.47	18.95	27.14	35.32	43.51

Tabulka A.10: Tabulka procentuálních rozdílů mezi odhadovaným časem a reálným časem ve srovnání s reálným pro kombinační útok a dva uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	33.98	27.96	16.96	14.46	22.95	31.45	39.95	49.06

Tabulka A.11: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro kombinační útok a čtyři uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	33.09	27.51	16.36	8.39	17.10	28.24	39.40	50.55

Tabulka A.12: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro kombinační útok a osm uzlů.

A.4 Hybridní útok slovník a maska

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	70.59	92.39	100.97	118.13	126.70	136.02	145.83	155.63

Tabulka A.13: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok slovník a maska a jeden uzel.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	59.09	56.85	56.44	55.61	55.19	54.77	54.36	53.94

Tabulka A.14: Tabulka procentuálních rozdílů mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok slovník a maska a dva uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	29.70	22.32	10.61	12.83	24.54	36.26	47.98	56.69

Tabulka A.15: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok slovník a maska a čtyři uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	23.52	16.59	7.24	21.51	34.39	47.14	59.89	72.63

Tabulka A.16: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok slovník a maska a osm uzlů.

A.5 Hybridní útok maska a slovník

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	41.81	20.25	29.95	49.35	59.05	68.74	78.44	88.18

Tabulka A.17: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok maska a slovník a jeden uzel.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	35.50	27.81	17.06	17.77	24.75	31.75	38.74	47.43

Tabulka A.18: Tabulka procentuálních rozdílů mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok maska a slovník a dva uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	38.81	32.41	22.21	6.99	11.64	18.58	28.78	38.98

Tabulka A.19: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok maska a slovník a čtyři uzly.

Čas režie [s]	–	15	20	30	35	40	45	50
Rozdíl [%]	33.30	27.20	16.09	6.14	17.26	28.38	39.49	50.61

Tabulka A.20: Tabulka procentuálního rozdílu mezi odhadovaným časem a reálným časem ve srovnání s reálným pro hybridní útok maska a slovník a osm uzlů.

Příloha B

Obsah média

Přiložené médium obsahuje kompletní zdrojové kódy systému Fitcrack, které lze najít v adresáři *fitcrack*. Pomocné python notebooky jsou uloženy ve složce *python_skripty*. Data z experimentů jsou uloženy ve složce *experimenty/data*. Text této bakalářské práce se nachází v adresáři *doc*, spolu se zdrojovými kódy napsanými v jazyce LaTeX. Soubor *README.md* v kořenovém adresáři obsahuje popis adresářové struktury a souborů nacházejících se na médiu. Soubor *README.md* ve složce *fitcrack* obsahuje výčet změn v systému Fitcrack a popis instalace.

```
SD
├── README.md
├── fitcrack/
│   ├── README.md
│   └── zdrojove_kody/
├── python_skripty/
├── experimenty/
│   ├── attacksLauncher.py
│   └── data/
└── doc/
    ├── BP.pdf
    └── zdrojove_kody/
```