VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INTERAKTIVNÍ NÁSTROJ NA TVORBU PRAVIDEL PRO MODIFIKACI HESEL

INTERACTIVE TOOL FOR CREATING PASSWORD-MANGLING RULES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JIŘÍ MLÁDEK**
AUTHOR

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

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

147183

Ústav: Ústav informačních systémů (UIFS)
Student: **Mládek Jiří**
Program: Informační technologie
Specializace: Informační technologie
Název: **Interaktivní nástroj na tvorbu pravidel pro modifikaci hesel**
Kategorie: Bezpečnost
Akademický rok: 2022/23

Zadání:

1. Seznamte se s problematikou lámání hesel, nástroji hashcat, John the Ripper (JtR) a Fitcrack.
2. Nastudujte syntaxi a sémantiku pravidel pro modifikaci hesel v nástrojích hashcat a JtR.
3. Navrhněte uživatelsky přívětivý nástroj umožňující jednoduše tvořit soubor takových pravidel a procházet existující. Součástí musí být náhled výsledných hesel po modifikaci.
4. Nástroj implementujte a integrujte do systému Fitcrack.
5. Prakticky demonstrujte jeho použitelnost vašeho řešení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- DAS, Anupam, et al. The Tangled Web of Password Reuse. In: *NDSS*. 2014. s. 23-26.
- UR, Blase, et al. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In: *USENIX Security Symposium*. 2015. s. 463-481.
- Hranický Radek. Digital Forensics: The Acceleration of Password Cracking. *Disertační práce*. Fakulta informačních technologií VUT v Brně. 2022.

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: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

Abstrakt

Cílem mojí bakalářské práce bylo vytvořit interaktivní nástroj na tvorbu pravidel pro modifikaci hesel a integrovat ho do systému Fitcrack. V práci se zabývám pravidly, která pomocí několika dostupných funkcí modifikují hesla a mohou sloužit pro obohacení slovníkového útoku. Implementovaný nástroj také nabízí živý náhled výsledných hesel po modifikaci pravidly a umožňuje tak vytvářet nové účinnější slovníky. Motivací pro tvorbu nástroje byl dosavadní způsob vytváření pravidel, kdy existující nástroje nenabízejí kombinaci grafického uživatelského rozhraní a velkého výběru funkcí pro tvorbu pravidel. Na experimentech jsou poté předvedeny případy užití a přínosy nástroje. Součástí práce je také uživatelské testování.

Abstract

The aim of this bachelor thesis was to create an interactive tool for creating password-mangling rules and integrate it into the Fitcrack system. In this thesis, I discuss password-mangling rules which use several available rule functions and can be used to enrich a dictionary attack. The implemented tool also provides a live preview of mangled passwords, allowing the creation of new more efficient dictionaries. The motivation for creating the tool was the current way of creating rules, where existing tools do not offer a graphical user interface and a large selection of rule functions at the same time. The experiments are then used to demonstrate use cases and the benefits of the tool. User testing is also part of the thesis.

Klíčová slova

Lámání hesel, Heslo, Pravidla pro modifikaci hesel, Útok založený na pravidlech, Fitcrack

Keywords

Password cracking, Password, Password-mangling rules, Rule-based attack, Fitcrack

Citace

MLÁDEK, Jiří. *Interaktivní nástroj
na tvorbu pravidel pro modifikaci hesel*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický, Ph.D.

Interaktivní nástroj na tvorbu pravidel pro modifikaci hesel

Prohlášení

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

.....
Jiří Mládek
9. května 2023

Poděkování

Tímto bych rád poděkoval vedoucímu mé bakalářské práce, panu Ing. Radkovi Hranickému, Ph.D., za jeho ochotu a vřelý přístup při vedení práce. Zejména za cenné rady, které mi při tvorbě poskytl. Dále děkuji všem, kteří se zúčastnili uživatelského testování mé práce.

Obsah

Kapitola 1

Úvod

Lámání hesel je proces, při kterém se uživatel snaží získat přístup do chráněného systému. Většina lidí si pod tímto pojmem představí nelegální činnost. Je ale důležité si uvědomit, že lámání hesel slouží i k jejich obnově, pokud heslo zapomeneme. Dokonce i pro získání přístupu do digitálních zařízení pachatelů při policejním vyšetřování. Má tedy smysl zkoumat způsoby, jakými lze hesla obnovit.

Pro lámání hesel existuje více způsobů. Ve své práci se zaměřuji zejména na útok, který k prolomení hesla používá pravidla pro modifikaci hesel (*password–mangling rules*). Tato pravidla jsou většinou používána jako obohacení slovníkového útoku. Jednoduchým příkladem může být např. přidání znaku za heslo ze slovníku. Při použití těchto pravidel se zvyšuje šance na prolomení hesla, jelikož mnoho lidí používá ve svých heslech populární vzory.

Hlavní motivací pro vytvoření interaktivního nástroje na tvorbu pravidel pro modifikaci hesel byl fakt, že dosavadní způsob vytváření pravidel je možný pouze jejich ručním vypisováním do souboru. Také je možné pravidla tvořit skrze nástroje zmíněné v podkapitole 3.3, které ale nenabízejí kombinaci grafického uživatelského rozhraní a velkého výběru funkcí pro tvorbu pravidel. Tento problém řeší mnou implementovaný nástroj.

Pro správnou implementaci nástroje bylo nejprve nutné seznámit se se systémem Fitcrack, zejména s dosavadním řešením práce s pravidly. Dále bylo potřeba nastudovat syntaxi a sémantiku pravidel. Poté byl proveden návrh nástroje. To obnášelo stanovení funkčnosti nástroje, návrh jeho grafického rozhraní a architektury pro možnou integraci do systému Fitcrack.

Přínos práce

V této práci se mi podařilo implementovat nástroj, který umožňuje vytvářet a upravovat sady pravidel pro modifikaci hesel. Nástroj je integrován do systému Fitcrack, a je tedy možné přímo využít vytvořené sady pravidel a obohatit tak různé typy útoků. Velkou předností mého nástroje je jeho intuitivní rozhraní, které umožňuje tvorbu pravidel i uživatelům, kteří neznají syntaxi a sémantiku funkcí nástroje Hashcat.

Funkce mohou být do pravidel vkládány jak ručně, tak pomocí interaktivní nabídky. Vkládání funkcí a specifikování hodnot jejich operandů je tak velmi přívětivé. Další užitečnou vlastností nástroje je živá kontrola validity pravidel při tvorbě, čehož je využito, pokud uživatel specifikuje pravidla ručně skrze klávesnici.

Kromě samotné tvorby pravidel slouží nástroj také k živému náhledu specifikovaných hesel po modifikaci. Nástroj tedy přímo umožňuje vytvoření nového, účinnějšího slovníku.

K modifikaci hesel slouží aplikátor pravidel, jehož rychlost v závislosti na počtu vygenerovaných hesel byla sledována v experimentu. Další experimenty předvádí případy užití tohoto nástroje. Nakonec byla díky uživatelskému testování získána zpětná vazba pro vylepšení nástroje do budoucna.

Související řešení

Pravidly pro modifikaci hesel se zabývá několik vědeckých článků. Blase Ur et. al [17] popisují výhody použití pravidel pro modifikaci hesel a zdůrazňují, že síla hesla by neměla být posuzována pouze podle účinnosti konkrétního typu útoku. Anupam Das et. al [1] upozorňují na znovupoužití hesel na internetu a jejich jednoduché změny, které uživatelé napříč stránkami provádějí. Při znalosti jednoho hesla lze pak ostatní prolomit právě s pomocí pravidel pro modifikaci hesel. Matt Weir et. al [19] diskutují zvolení efektivních pravidel pro modifikaci hesel a vytváří bezkontextovou gramatiku, díky které jsou generována pravidla v pořadí podle pravděpodobnosti úspěchu. Shunbin Li et al. [6] se zabývají generováním hesel na základě pravidel a vytváří automatický generátor pravidel. Vyzdvihují také výhody útoku založeného na pravidlech oproti útoku hrubou silou.

Velmi populárními nástroji, které dokáží útoky obohatit použitím pravidel, jsou Hashcat a John the Ripper. Tyto nástroje jsou detailně popsány v sekci 2.3. Dalším nástrojem je L0phtcrack[1], ten sice nabízí grafické uživatelské rozhraní, ale neumožňuje tvorbu vlastních pravidel. Stejně je na tom také komerční nástroj ElcomSoft[2]. Za zmínku ještě stojí nástroj Password Recovery Toolkit, který nabízí společnost AccessData (nyní patří pod společnost Exterro). Ten nabízí širokou nabídku funkcí pro tvorbu pravidel, pravidla je možné upravit a uspořádat podle potřeby[3].

Nástroje, které umožňují tvorbu pravidel kompatibilních s nástrojem Hashcat (případně John the Ripper), jsem detailně představil v sekci 3.3. Nebyl ale nalezen žádný nástroj, který by nabízel jak širokou škálu nabídky funkcí pro modifikaci hesel, tak příjemné grafické uživatelské rozhraní.

Struktura práce

Tato práce sestává ze 7 kapitol. Kapitola 2 pojednává o problematice lámání hesel a typech útoků. Dále se zabývá nástroji pro lámání hesel. V kapitole 3 jsou popsána pravidla pro modifikaci hesel, zejména jejich syntaxe a sémantika. Jsou zde také zmíněny některé existující nástroje pro tvorbu pravidel. Kapitola 4 se zabývá návrhem vytvářeného nástroje. Kapitola 5 je věnována implementačním detailům nástroje. Nejprve je popsána implementace aplikátoru pravidel a generátoru náhodných pravidel. Následně se zabývá vývojem backendu včetně API rozhraní a nakonec vývojem frontendu a finálního grafického uživatelského rozhraní. V kapitole 6 předvádím na několika experimentech případy užití nástroje a rychlost aplikátoru pravidel. Nakonec je v kapitole 7 ukázán postup a vyhodnocení uživatelského testování.

[1] https://gitlab.com/l0phtcrack/l0phtcrack
[2] https://www.elcomsoft.com/
[3] https://d1kpmuwb7gvu1i.cloudfront.net/dna/8.2.1/PRTK_DNA%20User%20Guide.pdf

Kapitola 2

Lámání hesel

Lámání hesla je proces, při kterém je získáno heslo pro přístup do chráněného systému. Proces se skládá ze 2 částí, které se opakují v určitém počtu iterací. První částí je generování kandidátního hesla, jehož způsob probíhá v závislosti na typu zvoleného útoku. Druhou částí je verifikace hesla, pro kterou existuje více způsobů. Jedním ze způsobů je verifikace, která používá známý heš cíleného hesla. Využívá toho, že v dnešní době již hesla většinou nejsou uchovávána jako prostý text, ale jako výsledek kryptografické hešovací funkce [3]. Ověření probíhá tím způsobem, že se postupně generují nová kandidátní hesla a na ty je použita hešovací funkce. Pokud se vytvořený heš shoduje s hledaným hešem, tak bylo lámání úspěšné. Pro zvýšení zabezpečení se využívá přidání soli (anglicky *salt*) k heslu. Až po jejím přidání je na heslo použita hešovací funkce, a díky tomu poté uživatelé se stejným heslem nemají stejný heš [8].

2.1 Znovupoužití hesel

Znovupoužití hesel a používání podobných vzorů při jejich vytváření představuje jedno z největších rizik pro bezpečnost, jelikož zvyšuje pravděpodobnost, že útočníci získají přístup k více účtům při prolomení jednoho hesla uživatele. Důvodem pro znovupoužití hesel a využití stejných vzorů je zejména strach ze zapomenutí hesla, potíž pamatovat si více hesel a nedostatečné povědomí o jejich bezpečnosti. Díky tomu dle studie téměř polovina uživatelů používá stejné heslo na více stránkách a také využívá jednoduché triky pro obměnu těchto hesel [1].

Webové stránky se snaží zamezovat používání slabých hesel pomocí omezení a zásad při vytváření hesel, jako jsou například minimální délka, výskyt velkého písmena apod [12]. Tato kritéria však uživatelé obcházejí velmi předvídatelně, nejčastější je přidání číslic na konec hesla a zvětšení prvního písmena hesla [1]. Této znalosti poté může být využito jak útočníky, tak forenzními vyšetřovateli, a pro prolomení hesla může být využit například slovníkový útok s použitím pravidel pro modifikaci hesel [1].

2.2 Typy útoků

Pro lámání hesel existují 2 základní přístupy:

- **Online útok** – Tento útok probíhá přímo proti online službě. Jedná se např. o útok na webovou stránku, kdy se snažíme získat přístup k účtu. Útočník má sadu kandidátních

hesel a těmi se postupně snaží přihlašovat. Nevýhodou je snadná detekovatelnost útoku. Při mnoha pokusech o přihlášení se může stát, že služba zvýší časové prodlevy mezi pokusy o přihlášení nebo přihlášení znemožní úplně [3].

- **Offline útok** – Tento útok je používán ve většině případů. Útočník musí mít přístup k heši hesla nebo k zašifrovanému souboru. Útok probíhá přímo na útočníkově zařízení, může využít jeho různé prostředky, jako např. GPU. Na rozdíl od online útoku není rychlost lámání omezena službou [3].

Veškeré popsané útoky v této práci využívají offline přístup. Podle způsobu generování hesel rozlišujeme několik typů útoků. Každý typ má své specifické použití a je vhodný pro jiný typ hesla. Časová komplexita útoku vždy závisí na počtu kandidátních hesel [3]. Typy útoků popsané v následujících podkapitolách jsou podporovány systémem **Fitcrack**.

2.2.1 Útok hrubou silou

Klasický útok hrubou silou (anglicky *brute-force attack*) funguje tím způsobem, že postupně tvoří všechny kombinace znaků dané délky nad danou množinou znaků [4]. Klasická verze útoku je nejjednodušší, ale také neefektivní a je používána zejména pro kratší hesla [17].

Pro upřesnění útoku hrubou silou lze použít **masku**, např. v nástroji Hashcat nebo John the Ripper. Ta velmi zefektivňuje lámání tím, že pomocí ní definujeme délku hesla a také konkrétní znaky/skupiny znaků na konkrétních pozicích v hesle. Existují tyto skupiny znaků [4]:

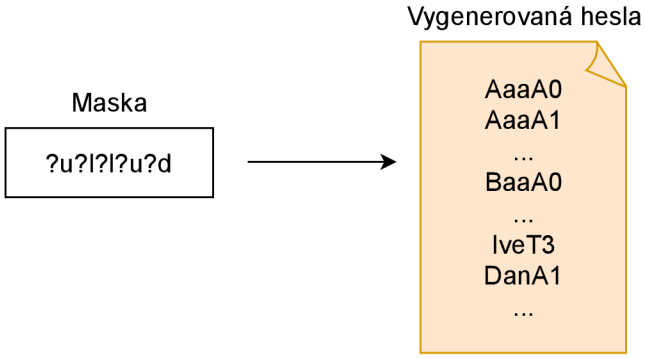
- **?l** – Malá písmena latinské abecedy (a–z), zkratka pro slovo *lower*.
- **?u** – Velká písmena latinské abecedy (A–Z), zkratka pro slovo *upper*.
- **?d** – Číslice 0–9, zkratka pro slovo *digit*.
- **?s** – Speciální ASCII znaky, zkratka pro slovo *special*.
- **?a** – Zahrnuje všechny předchozí možnosti, zkratka pro slovo *all*.
- **?h** – Znaky šestnáctkové soustavy (0–9, a–f), zkratka pro slovo *hexa*.
- **?H** – Velké znaky šestnáctkové soustavy (0–9, A–F), zkratka pro slovo *HEXA*.
- **?b** – Binární znaky (0x00–0xFF), zkratka pro slovo *binary*.

Příklad použití masky je ukázán na obrázku 2.1. Použití masky je vhodné spíše u krátkých hesel, u kterých mnoho uživatelů využívá populárních vzorů, jako např. použití číslice na konci. Podle studie je ale i útok s maskou méně efektivní než ostatní typy útoků [17].

2.2.2 Slovníkový útok

Slovníkový útok (*dictionary attack*), také označován jako přímý útok (*straight attack*), využívá k lámání hesla slovník. To je soubor, ve kterém jsou uložena jednotlivá hesla, každé na samostatném řádku. Tyto slovníky často obsahují databáze uniknutých hesel [1]. Při útoku jsou hesla ze slovníku postupně převedena na heš a vždy jsou porovnána s výsledným hešem. Takto je postupně projitý celý slovník, dokud nenastane shoda.

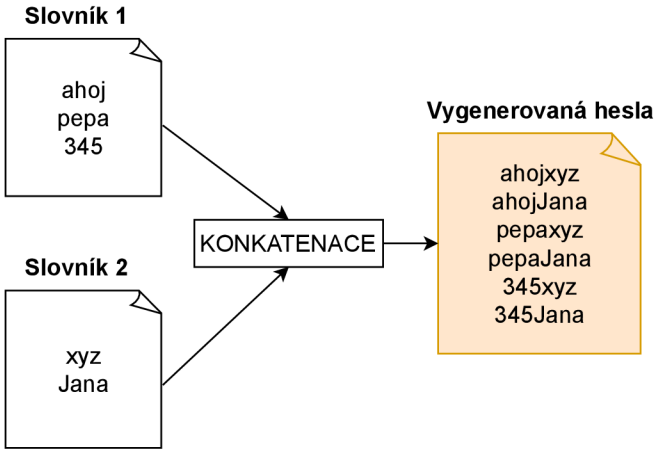
Pro zvýšení úspěchu při lámání je dobré slovníkový útok kombinovat s použitím pravidel. Toto je i s příkladem popsáno v sekci 3.2.



Obrázek 2.1: Generování hesel pomocí masky

2.2.3 Kombinační útok

Kombinační útok (*combination attack, combinator attack*) využívá 2 slovníky hesel. Generování nových hesel probíhá tak, že se postupně konkatenují hesla z prvního slovníku s hesly z druhého slovníku. Celkový počet vygenerovaných hesel je potom dán jako násobek počtu hesel prvního a druhého slovníku. Tento proces lze vidět na obrázku 2.2 [4].



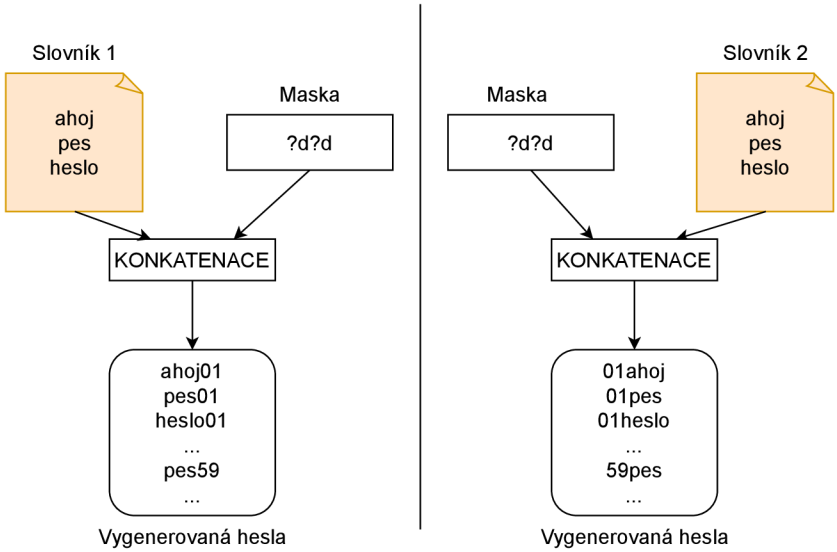
Obrázek 2.2: Generování hesel pomocí kombinačního útoku

2.2.4 Hybridní útok

Hybridní útoky kombinují slovníkový útok (viz podsekci 2.2.2) s útokem hrubou silou s využitím masky (viz podsekci 2.2.1). Existují 2 varianty hybridních útoků:

1. **Hybridní útok s použitím slovníku a masky** – První část hesla je generována ze slovníku a druhá z masky.
2. **Hybridní útok s použitím masky a slovníku** – První část je generována z masky a druhá ze slovníku.

Části jsou následně zkonkatenovány. Oba typy útoků lze vidět na obrázku 2.3, lze si všimnout, že útoky jsou navzájem zrcadlové. Hybridní typ útoku je také možné nahradit použitím pravidel [5, 4].



Obrázek 2.3: Vlevo hybridní útok s použitím slovníku a masky, vpravo hybridní útok s použitím masky a slovníku

2.2.5 Útok PCFG

Útok PCFG (*Probabilistic Context-Free Grammar*) používá pro lámání hesel bezkontextovou gramatiku [19]. Na základě trénovací sady (slovníku) je vytvořena bezkontextová gramatika, která předepisuje povolené tvary kandidátních hesel. Těmto tvarům je poté přiřazena pravděpodobnost, se kterou se v kandidátním hesle budou vyskytovat. Cílem je nejprve vytvořit ta kandidátní hesla, která mají vyšší šanci úspěšnosti [3].

Pravděpodobnosti jsou přiřazovány jak samotným řetězcům, tak i struktuře hesla. Pro příklad, heslu `password523` by byla přiřazena pravděpodobnost pro strukturu {6 písmen}{3 číslice} a pro řetězce `password`, `523` [17, 7].

Tento útok je efektivní zejména pro lámání hesel, která používají předvídatelné vzory. Naopak není vhodný pro hesla, která jsou náhodně generována. Efektivita tohoto útoku také velmi závisí na kvalitě trénovacích dat, ze kterých vzniká gramatika. Konkrétně je vyšší, pokud spolu hesla z trénovací sady souvisejí [17].

2.2.6 Útok PRINCE

Útok PRINCE (*Probability infinite chained elements*) je podobný kombinačnímu útoku. Na rozdíl od něj ale využívá jenom 1 slovník. Ze slov v tomto slovníku vytváří jejich kombinacemi řetězce o různých délkách. PRINCE algoritmus je založen na 3 částech:

1. **Element** – Představuje nezměněné heslo ze slovníku. Podle délky jsou poté elementy (hesla) tříděny do skupin.
2. **Řetězec** – Posloupnost elementů o určité délce. Např. řetězec o délce 3 se může skládat z elementů těchto délek: (3), (1,2), (2,1), (1,1,1).
3. **Celkový počet kandidátních hesel** – Na základě počtu výskytů elementů jednotlivých délek je vypočten celkový počet kandidátních hesel pro konkrétní řetězec. Například pokud elementy délky 4 mají počet výskytů X a elementy délky 2 počet výskytů Y, tak celkový počet kandidátních hesel tohoto řetězce (4,2) je X * Y.

Systém Fitcrack pro použití tohoto útoku využívá referenční implementaci algoritmu *princeprocessor*[1]. Volitelně lze poté specifikovat minimální a maximální délku hesla, minimální a maximální počet prvků v řetězci a maximum vygenerovaných hesel [3, 5].

2.3 Nástroje pro lámání hesel

Existuje velké množství nástrojů pro lámání hesel. Jelikož se ale v této práci zabývám zejména využitelností použití pravidel, tak zde popíšu 3 nástroje, které mají možnost tohoto typu útoku využít. Jedná se o nástroj Hashcat, John the Ripper a distribuovaný nástroj Fitcrack. Konkrétně popíšu zejména vlastnosti těchto nástrojů a typy jimi nabízených útoků.

2.3.1 Hashcat

Hashcat[2] je velmi populární nástroj pro lámání hesel, který byl vytvořen v roce 2009 jako proprietární software. V roce 2015 se stal kód veřejně dostupný pod MIT licencí a jedná se tak nyní o otevřený software [3].

Mezi jeho výhody patří vysoký výkon při lámání hesel, a to zejména také kvůli tomu, že může využívat procesoru na grafické kartě. Je dokonce označován jako "nejrychlejší prolamovač hesel na světě".

Podporuje více než 350 různých typů kryptografických hešů. Nevýhodou tohoto nástroje je nutnost použití externího skriptu pro získání hešů, např. pro dokumenty PDF a Office [15, 3].

Nástroj Hashcat podporuje 6 různých typů útoků, jejich základ byl popsán v sekci 2.2 [15, 4]:

- **Slovníkový útok** – Klasický slovníkový útok, je také podporováno použití pravidel.
- **Kombinační útok** – Klasický kombinační útok. Ke každému ze slovníků lze specifikovat 1 pravidlo.
- **Útok hrubou silou** – Klasický útok hrubou silou s využitím masky. Pro postup generování znaků lze využít Markovovy řetězce.
- **Hybridní útok s použitím slovníku a masky** – Levá část hesla je generována ze slovníku, pravá část z masky.
- **Hybridní útok s použitím masky a slovníku** – Levá část hesla je generována z masky, pravá část ze slovníku.
- **Asociační útok**[3] – Jedná se o nový útok v nástroji Hashcat. Pro lámání využívá známou část hesla nebo pravděpodobnou součást, např. uživatelské jméno. Na tuto část lze poté použít pravidla.

2.3.2 John the Ripper

John the ripper[4] je nástroj na lámání hesel, který byl vytvořen v roce 1996 nejprve pro operační systém MS–DOS. Jedná se o otevřený software. Od roku 2011 také u některých

[1] https://github.com/hashcat/princeprocessor
[2] https://github.com/hashcat/hashcat
[3] https://hashcat.net/wiki/doku.php?id=association_attack
[4] https://www.openwall.com/john/

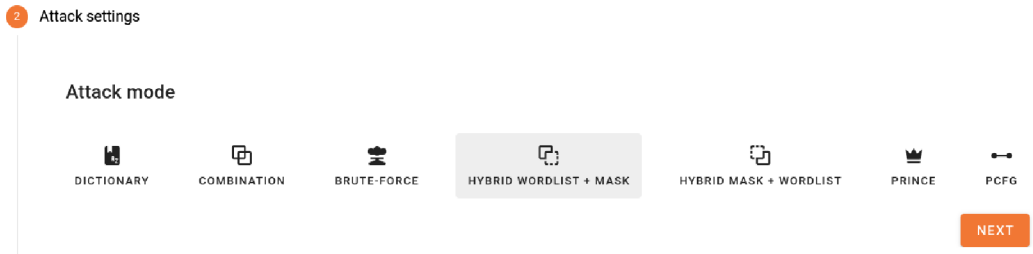
algoritmů nabízí GPU akceleraci pro rychlejší lámání hesel [3]. Podporuje 4 módy útoků [10, 11]:

- **Slovníkový mód** – Jedná se o klasický slovníkový útok. Je také podporováno použití pravidel.
- **Mód „Single crack“** – Podle autora tohoto nástroje se jedná o mód, který by měl uživatel vyzkoušet jako první. Jako kandidátní hesla používá login uživatele, obsah pole GECOS / Full name v UNIXU a jméno domácího adresáře. Na tato hesla poté použije pravidla. Pokud je heslo prolomeno, vyzkouší se také na všechny načtené heše, kdyby měl náhodou některý uživatel stejné heslo.
- **Inkrementální mód** – Jedná se o útok hrubou silou, který je speciální v tom, že využívá frekventovaně využívaných znaků pro tvorbu potenciálních hesel. Pro použití tohoto útoku má uživatel možnost specifikovat limitní délku hesla a soubor znaků, ze kterých bude heslo vytvářeno.
- **Externí mód** – Tento způsob útoku slouží pro použití vlastního externího programu pro tvorbu kandidátních hesel (vytvořený v jazyce C). John the Ripper je poté použit pouze pro ověření úspěšnosti prolomení hesla.

2.3.3 Fitcrack

Fitcrack[5] je distribuovaný systém pro lámání hesel. Je založen na softwaru *Berkeley Open Infrastructure for Network Computing* (BOINC), díky kterému může jednotlivé lámací úlohy distribuovat mezi více uzlů. Díky tomu, že využívá nástroj Hashcat, tak nabízí podporu velkého množství formátů a algoritmů pro lámání hesel. Kombinuje tak příjemné grafické rozhraní s distribucí úloh mezi více uzlů, což jiné nástroje nenabízí. Nástroj byl vyvinut výzkumnou skupinou NES@FIT [5].

Podporuje 7 typů útoků, z toho 5 typů je poskytnuto nástrojem Hashcat. Frontend nástroje poskytuje nad těmito útoky abstrakci, jinak se útokům nastavují stejné parametry jako v nástroji Hashcat. Jedná se o slovníkový útok, kombinační útok, útok hrubou silou a 2 typy hybridních útoků. Dále nabízí útok PCFG a Prince, popis viz sekci 2.2 [5]. Na obrázku 2.4 je vidět grafické uživatelské rozhraní při výběru typu útoku.



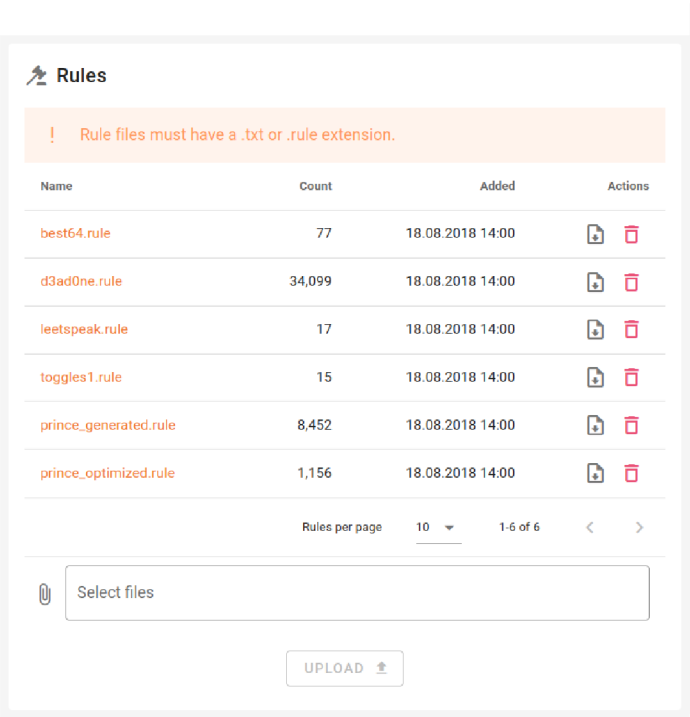
Obrázek 2.4: GUI při vybírání typu útoku v nástroji Fitcrack

Dosavadní práce s pravidly v nástroji Fitcrack

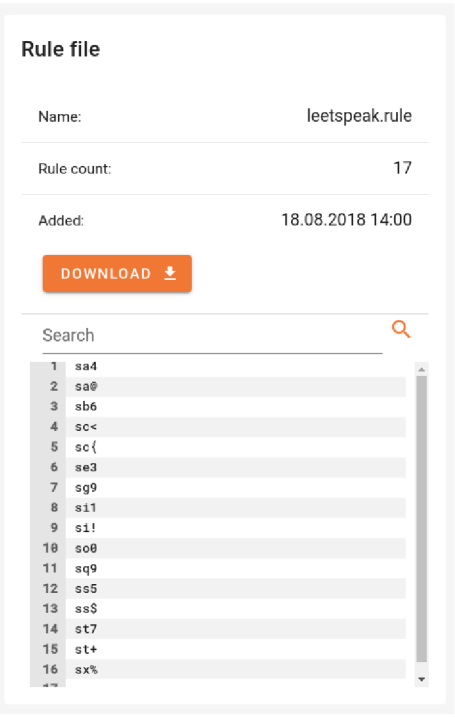
Práce s pravidly probíhala doposud pouze tím způsobem, že bylo možné nechat si vypsat všechny soubory s pravidly a informace o těchto souborech (viz obrázek 2.5). Dále bylo

[5] https://fitcrack.fit.vutbr.cz/

umožněno do systému nahrát nový soubor nebo soubor stáhnout či odstranit. Uživatel si také mohl zobrazit obsah těchto souborů (viz obrázek 2.6). Pravidla však nebylo možné jakkoliv upravovat.



Obrázek 2.5: Zobrazení všech souborů s pravidly



Obrázek 2.6: Zobrazení konkrétního souboru s pravidly

S pravidly se dalo pracovat při vytváření některých typů útoků. Konkrétně u slovníkového útoku, útoku PCFG a Prince lze zvolit soubor s pravidly, který má být použit. U kombinačního útoku a obou typů hybridních útoků lze specifikovat jedno pravidlo, které má být použito na slovník. Typ útoku Prince také podporuje generování náhodných pravidel.

Kapitola 3

Pravidla pro modifikaci hesel

Pravidla pro modifikaci hesel jsou využívána zejména kvůli vysoké efektivitě, přesnosti a flexibilitě při útoku. Nejčastěji jsou používány v kombinaci se slovníkovým útokem (popsaný v podsekci 2.2.2). Tento způsob vylepšení útoku byl nejprve použit v nástroji John the Ripper a následně byl rozšířen v nástroji Hashcat [3]. Pravidlo nebo soubor pravidel téměř připomíná použití programovacího jazyka, jelikož se skládá z různých funkcí (viz tabulky C.1 a C.2). V nástroji Hashcat jich je momentálně dostupných přes 50. Díky těmto funkcím můžeme kandidáta na heslo různě modifikovat, např. přidávat, odebírat, zaměňovat znaky atd., více v sekci 3.1 [16].

Výhodou použití pravidel je zvýšení počtu kandidátů na heslo, a tím pádem i zvýšení pravděpodobnosti prolomení hesla. Jejich použití je vhodné, zejména pokud známe vzor hesel (obecný nebo konkrétního uživatele). Uživatel totiž často používá různá hesla, ale stejné vzory při jejich vytváření [1].

3.1 Syntaxe a sémantika pravidel

Pravidla pro modifikaci hesel musí být specifikována v souboru pravidel. Pro pravidlo je vyhrazen 1 řádek souboru. Tato pravidla se skládají z jedné nebo více funkcí, jednotlivé funkce jsou odděleny mezerou. Většina těchto funkcí je kompatibilní jak s nástrojem Hashcat, tak s nástrojem John the Ripper a PasswordsPro [17]. V této kapitole se ale budu soustředit na funkce podporované nástrojem Hashcat, jelikož je využíván systémem Fitcrack, pro který vytvářím interaktivní nástroj.

V tabulkách C.1 a C.2 jsou uvedené funkce nástroje Hashcat, které je možné používat ve mnou vytvořeném interaktivním nástroji. Nejprve je uvedeno použité jméno funkce. To je uvedeno anglicky, protože v tomto formátu bude použito i v interaktivním nástroji. Dále je v tabulce popsána syntaxe a sémantika jednotlivých funkcí. Každá funkce má určený povinný počet operandů (0 až 3). Nakonec je v tabulce uveden příklad použití funkce, její náležící vstup a výstup po provedení funkce.

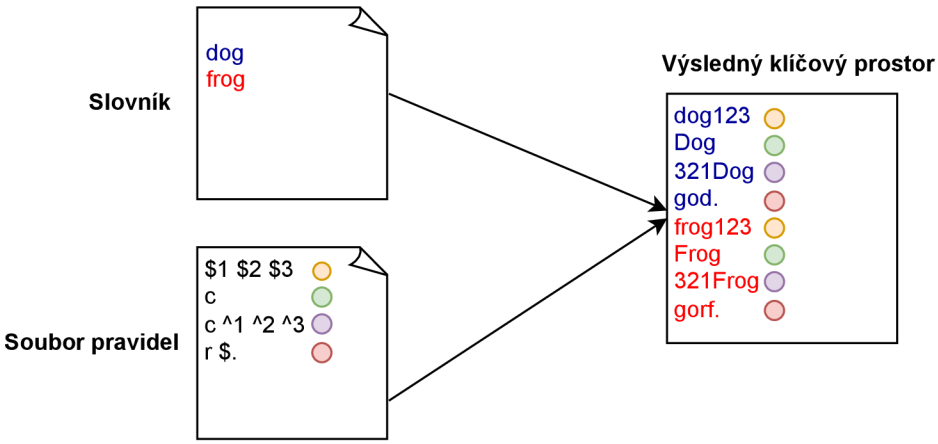
3.2 Použití pravidel u slovníkového útoku

Na obrázku 3.1 je zobrazen postup při použití pravidel na hesla ze slovníku. Nejprve je tedy nutné specifikovat soubor s kandidátními hesly (slovník), následně soubor s pravidly. Soubor s pravidly není nutné vytvářet, je možné využít již existující, např. z repozitáře nástroje Hashcat.

Postup při generování výsledných hesel je takový, že na každé heslo ze slovníku jsou postupně použita jednotlivá pravidla ze souboru pravidel. Nejprve je tedy na první slovo ze slovníku použito první pravidlo. Pro příklad na obrázku 3.1, na slovo dog je použito pravidlo **$1 $2 $3**. První je aplikována funkce `$1` (dog → dog1), poté funkce `$2` (dog1 → dog12) a nakonec funkce `$3` (dog12 → dog123). Následně jsou postupně použita další pravidla a jakmile jsou pravidla vyčerpána, tak se vybere ze slovníku další slovo a postup se opakuje.

Celkový počet vygenerovaných hesel v je počítán jako suma počtu kandidátů na heslo ze slovníků S_i, vynásobena počtem pravidel p [3]:

$$v = \sum_{i=1}^{n} (p * |S_i|)$$



Obrázek 3.1: Použití pravidel na hesla ze slovníku

3.3 Existující nástroje na tvorbu pravidel

Před navrhnutím samotného interaktivního nástroje jsem provedl výzkum již existujících nástrojů, které také umožňují vytvářet pravidla pro modifikaci hesel kompatibilní s nástrojem Hashcat/John the Ripper. Tyto nástroje zde popíšu, včetně jejich využití, výhod a nevýhod.

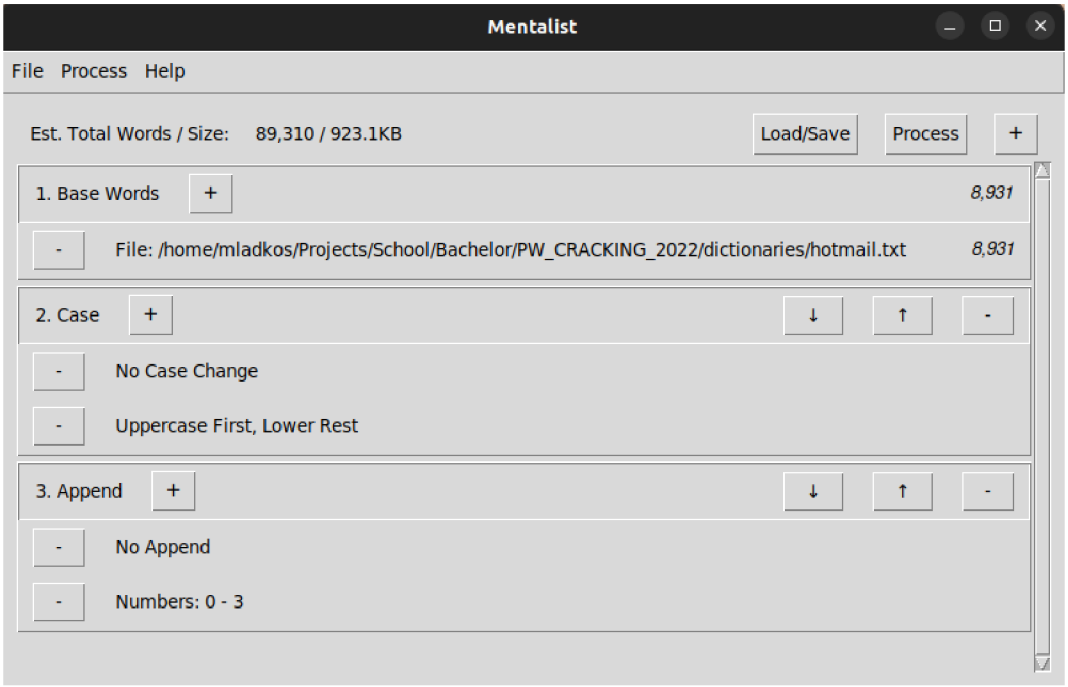
3.3.1 Mentalist

Mentalist[1] je nástroj s grafickým uživatelským rozhraním, který slouží pro generování vlastních slovníků pomocí přidávání pravidel. Dále nabízí možnost zvolená pravidla exportovat, tím vytvoří soubor s pravidly s příponou `.rule`. Tato vygenerovaná pravidla jsou kompatibilní s nástrojem Hashcat a John the Ripper, takže je poté možné je použít např. při slovníkovém útoku[2]. Výhodou tohoto nástroje je grafické ovládání, nevýhodou je ale malá škála funkcí, ze kterých lze pravidla sestavit.

Nástroj se skládá z uzlů, které znázorňují skupiny pravidel, a tyto uzly z atributů, které znázorňují konkrétní pravidla. Na obrázku 3.2 lze vidět grafické rozhraní nástroje.

[1] https://github.com/sc0tfree/mentalist
[2] https://github.com/sc0tfree/mentalist/wiki

Nejprve je nutné vybrat soubor obsahující slovník, na který budou použita pravidla. Poté je možné slova modifikovat pomocí 4 skupin pravidel. První skupina pravidel se stará o změnu velikosti písmen. Lze např. zvětšit první písmeno a zbytek zmenšit, což odpovídá funkci `c`, viz tabulku C.1. Druhá skupina reprezentuje funkce, které se starají o nahrazení znaku v řetězci, je umožněno vybírat mezi nahrazením prvního výskytu znaku, posledního výskytu znaku, nebo nahrazením všech výskytů znaků. Třetí skupina pravidel se stará o připojení znaku/znaků na konec slova a čtvrtá skupina se stará o připojení znaku/znaků na začátek slova. Po přidání pravidel je umožněno exportování do souboru, který může sloužit jako vstup pro útok. Obrázek 3.3 zobrazuje vygenerovaný soubor s pravidly, které odpovídají konfiguraci z obrázku 3.2. V něm lze vidět i zakomentovaný řádek s informací, které uzly byly použity a s jakými atributy.



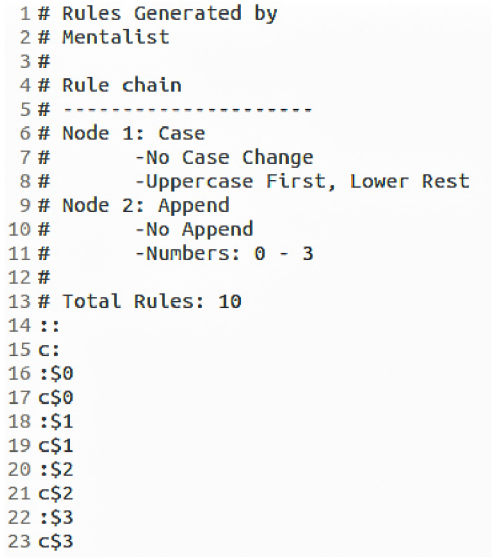
Obrázek 3.2: Použití nástroje Mentalist

3.3.2 PACK – rulegen

Nástroj PACK[3] je sada 4 skriptů, které slouží k analýze hesel za účelem zvýšení úspěšnosti při jejich lámání. Jejich výstup lze využít např. jako vstupní soubor pro některý z útoků nástroje Hashcat.

Jedním ze skriptů je `rulegen.py`, který generuje pravidla na základě analýzy hesla [11]. Autor skriptu vychází z myšlenky, že hesla, která se snažíme prolomit, obsahují překlepy. Díky této myšlence využívá pro zjištění zdrojového slova knihovny pro kontrolu pravopisu (např. `Enchant`). Pro přesnější odhadnutí zdrojového slova je ale ještě předtím použit modul pro předběžnou analýzu hesla. Ten poté poskytne algoritmům pro kontrolu pravopisu přesnější kandidáty na heslo[4].

[3] https://github.com/iphelix/pack
[4] https://iphelix.medium.com/automatic-password-rule-analysis-and-generation-7d2574516e48



Obrázek 3.3: Vygenerovaná pravidla nástrojem Mentalist

Základní použití skriptu zahrnuje analýzu jednoho hesla. Heslo specifikujeme pomocí parametru `--password`, skript poté navrhne potenciální zdrojové slovo a pravidla, která na něj byla použita. Parametrem `--wordlist` lze také specifikovat vlastní slovník, který má být použit pro vybrání zdrojových slov. Toto se může hodit, pokud by heslo obsahovalo např. slangový výraz, který není obsažen ve výchozím slovníku používané knihovny pro kontrolu pravopisu. Pokud v parametru specifikujeme pouze slovník a nespecifikujeme heslo, tak skript zobrazí analýzu hesel ve slovníku. Konkrétně zobrazí 10 nejčastěji použitých pravidel a slov.

Výhodou tohoto nástroje je fakt, že vygenerovaná pravidla vychází ze vzorů, a tudíž je i vyšší pravděpodobnost prolomení hesla.

3.3.3 Hashcat – náhodná pravidla

Nástroj Hashcat nabízí speciální funkci pro generování náhodných pravidel. Ta se může hodit, zejména pokud nestačily pro prolomení hesla dosavadní definovaná pravidla. Podporuje 3 parametry. Pomocí nich lze specifikovat konkrétní počet pravidel, stanovit jejich rozsah, případně lze v parametru specifikovat funkce, ze kterých mohou být pravidla tvořena. Náhodné generování pravidel je spuštěno přepínačem `-g` nebo přepínačem `--generate-rules`. Použití náhodných pravidel je ale neefektivní, jelikož nevyužívá předem známých vzorů [16].

Kapitola 4

Návrh nástroje

Hlavní motivací pro vytvoření interaktivního nástroje na tvorbu pravidel pro modifikaci hesel bylo, že dosavadní způsob vytváření pravidel je možný pouze ručně nebo skrze nástroje zmíněné v kapitole 3.3. Pokud si chce uživatel specifikovat svoje vlastní pravidla a zároveň mít na výběr ze všech funkcí nástroje Hashcat, tak nemá jinou možnost, než si vytvořit textový soubor, a do něj postupně specifikovat pravidla a jejich funkce. Tento způsob může být neintuitivní.

Cílem je tedy vytvořit interaktivní nástroj, který umožňuje pravidla vytvářet skrze grafické rozhraní a zároveň zobrazí živý náhled výsledných hesel po modifikaci. Díky tomu jsou potenciální hesla pro uživatele dostupná při úpravě pravidel, a uživatel tedy může pravidla upravovat lépe podle potřeby. Výsledná modifikovaná hesla je také možné stáhnout jako nový slovník, díky tomu lze nástroj používat i pro vytváření nových slovníků s vyšším potenciálem úspěchu při lámání. Užitečnou vlastností je také živá kontrola validity pravidel a případné upozornění na syntaktickou chybu.

Jelikož je interaktivní nástroj integrován do systému Fitcrack, tak bylo v první řadě nutné nastudovat tento systém, jeho architekturu, ovládání a design. Zejména zjistit, jaký je dosavadní přístup k pravidlům pro modifikaci hesel a kde se s nimi v nástroji pracuje. Dalším krokem bylo vytvořit samotný návrh tak, aby byl nástroj intuitivní, příjemný a byla možná jeho integrace.

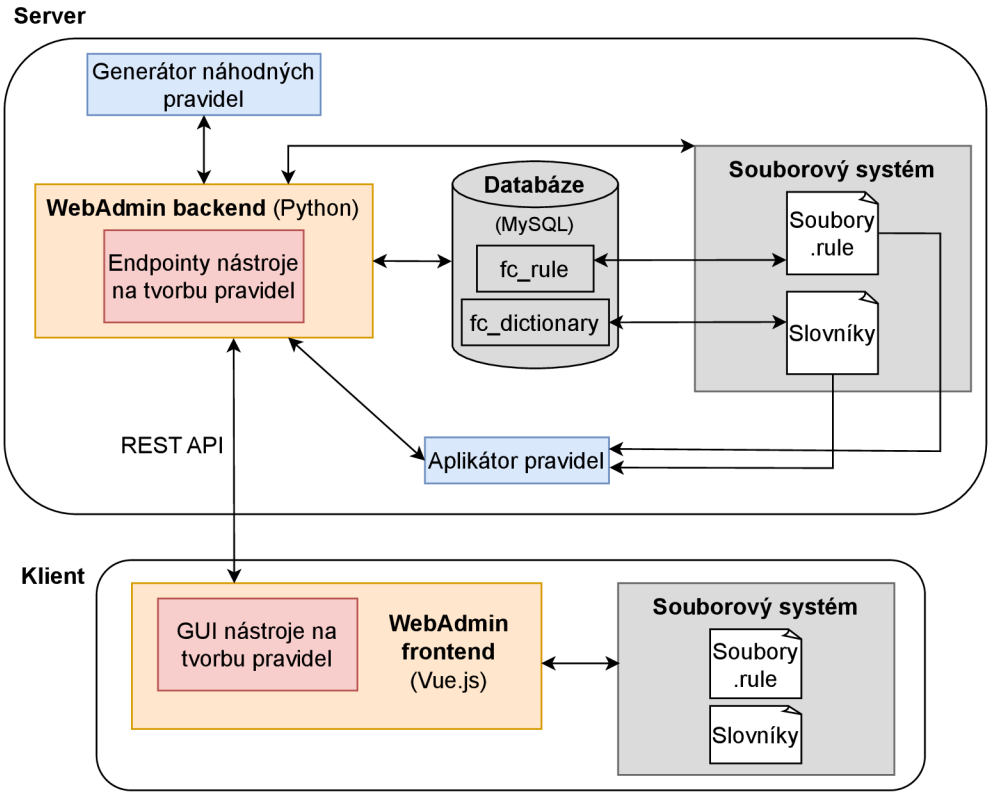
4.1 Architektura nástroje a integrace do systému Fitcrack

Z důvodu integrace nástroje do systému Fitcrack bylo nutné nastudovat jeho architekturu a jednotlivé části. Zejména způsob uchování sad pravidel v databázi a jejich mapování na soubory v souborovém systému serveru. Poté bylo vytvořeno schéma architektury nástroje a způsobu integrace, viz obrázek 4.1. Na tomto schématu lze vidět klientskou a serverovou část, které zde budou nyní blíže popsány.

Klient

Pro zprostředkování webové aplikace je důležitá část systému Fitcrack *WebAdmin frontend*, která je založená na frameworku `Vue.js`[1] a pro komponenty uživatelského rozhraní využívá framework `Vuetify`[2]. Do této části je integrováno GUI mého nástroje. V klientské části je také využíváno souborového systému, konkrétně pokud uživatel připojuje sadu pravidel

[1] https://vuejs.org/
[2] https://v2.vuetifyjs.com/en/



Obrázek 4.1: Schéma architektury nástroje a jeho integrace

nebo slovník za již existující. Komunikace mezi klientem a serverem probíhá skrze REST API.

Server

První částí využívanou na serveru je část nástroje Fitcrack *WebAdmin backend*, která je napsána v jazyce `Python 3` a je založená na frameworku `Flask`[3]. Pro mapování mezi backendem a databází `MySQL`[4] je použita knihovna `SQLAlchemy`[5]. Pro komunikaci mezi serverem a klientem bude využíváno existujících endpointů pro práci s pravidly a slovníky. Bude také potřeba těchto nových, které jsou popsány v sekci 5.2.3:

- endpoint pro aplikování pravidel na sadu hesel,
- endpoint pro změnu sady pravidel,
- endpoint pro vygenerování náhodného pravidla.

Některé endpointy provádějí operace nad daty uloženými v tabulkách v databázi. Jedná se zejména o tabulky `fc_rule`, která obsahuje záznamy sad pravidel, a `fc_dictionary`, která obsahuje záznamy slovníků. K těmto záznamům je přiřazen konkrétní soubor v souborovém systému serveru a těchto souborů tak může být využíváno. Pro aplikátor pravidel

[3] https://flask.palletsprojects.com/en/2.2.x/
[4] https://www.mysql.com/
[5] https://www.sqlalchemy.org/

a generátor náhodných pravidel bude využíváno funkcí ze skriptu `cpu_rules.c`, blíže popsáno v sekci 5.1.

4.2 Funkčnost

Pro lepší představu práce s nástrojem byl vytvořen diagram funkčnosti nástroje 4.2. Hlavním případem užití je vytvoření nové sady pravidel nebo úprava existující sady. To zahrnuje několik operací. Při vytváření sady musí mít přidělen jedinečný název, při úpravě má uživatel možnost sadu přejmenovat. Dále může být přidán řádek s novým pravidlem, které může být libovolně upravováno. Uživatel má dále možnost za specifikovaná pravidla připojit již existující sadu pravidel. Další možností je odebrání pravidla nebo odebrání všech pravidel jedním kliknutím. Funkcí, která může být využita v případě situace, kdy uživatel nemá nové nápady pro vymyšlení pravidla, je **vygenerování náhodného pravidla**. Při hotové editaci je poté možné uložit změny.

Důležitou částí nástroje je **aplikátor pravidel**, který se stará o použití sady pravidel na sadu hesel. Sadu hesel je možné upravovat v nástroji, konkrétně lze přidat heslo, připojit již existující slovník za specifikovaná hesla či hesla odebrat. Aplikátorem jsou tato hesla modifikována a uživateli je zprostředkován jejich živý náhled. Výsledná modifikovaná hesla lze také stáhnout, čímž je vytvořen nový účinnější slovník.

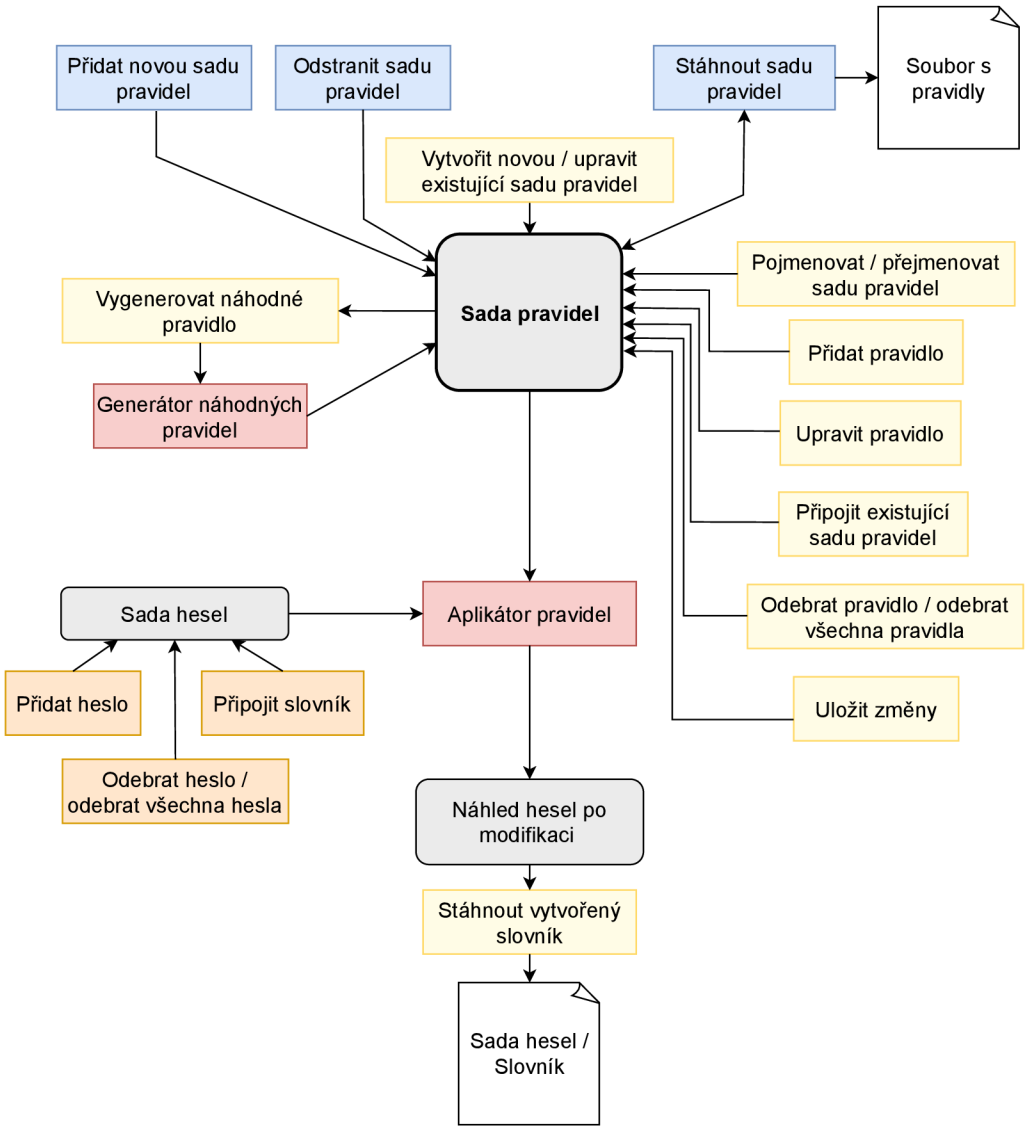
Kromě těchto funkcionalit vytvořeného nástroje lze se sadou pravidel provádět i další, již implementované operace. Na schématu 4.2 jsou zobrazeny modrými rámečky. Uživatel má možnost přidat novou sadu pravidel vybráním souboru ze souborového systému svého počítače, dále odstranit sadu pravidel a stáhnout sadu pravidel na svůj počítač. Umístění tlačítek a oken pro zprostředkování všech vlastností ze schématu 4.2 lze vidět v sekci 4.3.

4.3 Návrh uživatelského rozhraní

Návrh uživatelského rozhraní probíhal s ohledem na existující design systému Fitcrack. Pro vstup do nástroje slouží tlačítka pro vytvoření a modifikaci sady pravidel, viz sekci 4.4. Pro obě možnosti je nástroj identický, pouze je rozlišen nadpisem (vytváření nebo editace). Návrh vzhledu hlavního okna nástroje lze vidět na obrázku 4.3. Skládá se ze 2 částí:

- **Vytváření/úprava sady pravidel** (vlevo) – Tato část slouží k vytváření pravidel a jejich modifikaci. Skládá se z okna pro název souboru, 4 tlačítek pro správu pravidel, hlavního okna pro zobrazení pravidel a tlačítka pro uložení sady pravidel. Také je zde informace o celkovém počtu pravidel v sadě a tlačítko pro zobrazení dostupných funkcí.
- **Živý náhled hesel po modifikaci** (vpravo) – Tato část slouží pro zobrazení náhledu hesel po modifikaci pravidly. Uživatel má na výběr buď specifikovat hesla ručně, nebo může připojit již existující slovník. Také je umožněno hesla naráz všechna odstranit. Uživateli je také zobrazen počet specifikovaných hesel a odhadovaný finální počet hesel po modifikaci. Stisknutím tlačítka `Download mangled passwords` lze stáhnout soubor s modifikovanými hesly.

Při návrhu bylo důležité, aby byly obě části nástroje vidět zároveň a práce s pravidly tím byla příjemnější.

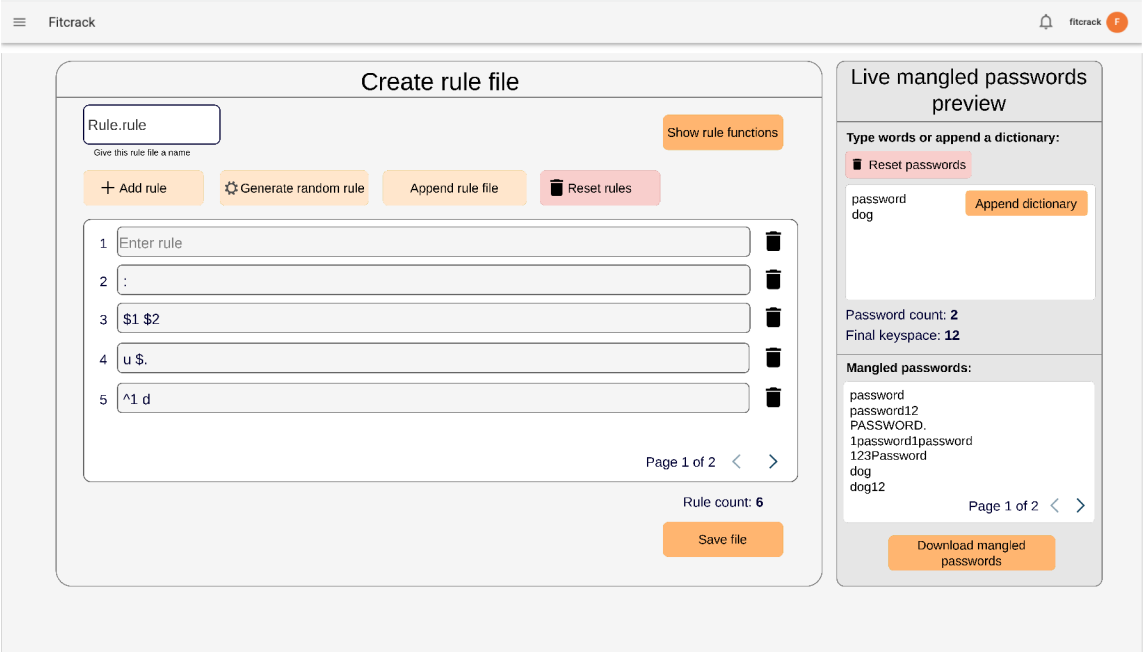


Obrázek 4.2: Diagram funkčnosti nástroje

Vytvoření sady pravidel

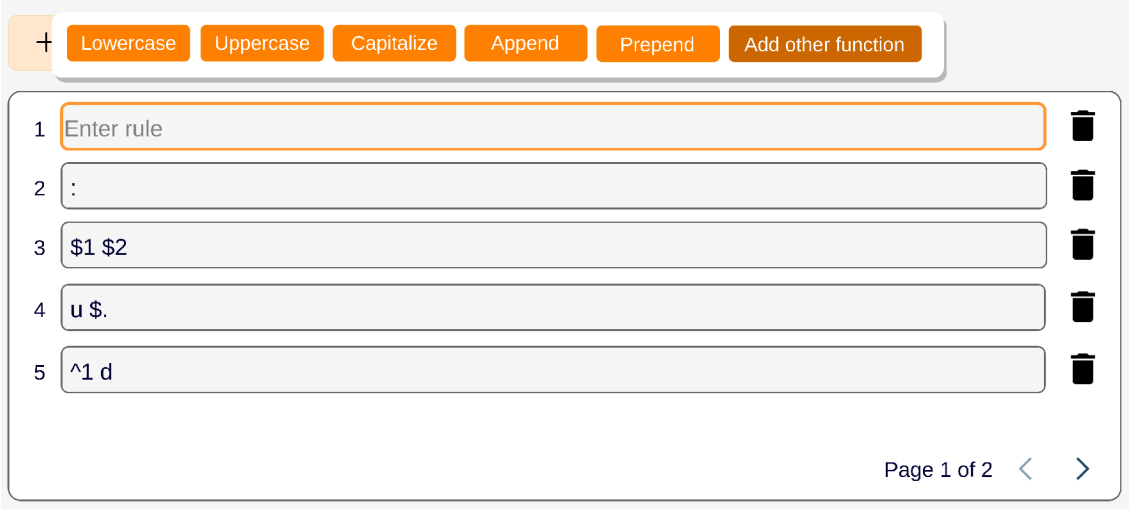
Při vytváření nové sady pravidel je nutné zadat název cílového souboru. Poté má uživatel celkem 3 možnosti, jak může přidat pravidla:

1. Přidání jednoho pravidla tlačítkem `Add rule`. Vytvoří se prázdné pravidlo, do kterého je možné specifikovat funkce.
2. Přidání pravidel z existujícího souboru tlačítkem `Append rule file`. Poté má uživatel možnost vybrat sadu pravidel buď ze souborového systému svého počítače nebo ze serveru.
3. Přidání náhodně vygenerovaného pravidla.



Obrázek 4.3: Návrh hlavního okna nástroje

Zde je popsána první možnost. Kliknutím na tlačítko se přidá nové vstupní pole, vyjadřující prázdné pravidlo. Pro editaci pravidla stačí na toto pole kliknout. Zobrazí se **nabídka hlavních funkcí**, které může uživatel rovnou zvolit. Jedná se o funkci pro zmenšení všech písmen, zvětšení všech písmen, zvětšení prvního písmena, připojení znaku za heslo a přidání znaku před heslo. V této nabídce se také vyskytuje tlačítko pro zobrazení ostatních funkcí (viz obrázek 4.4).



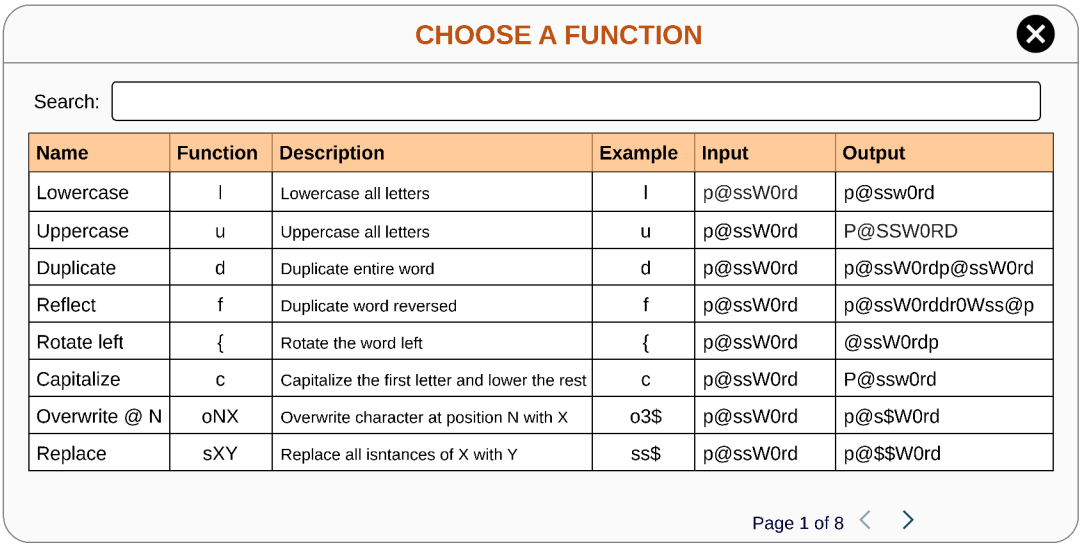
Obrázek 4.4: Nabídka funkcí po kliknutí na řádek (pravidlo)

Při kliknutí na toto tlačítko dojde k přesměrování do **vyskakovacího okna pro zobrazení ostatních funkcí**, viz obrázek 4.5. Sestává z vyhledávacího pole pro funkce a tabulky zobrazující bližší popis funkcí. Jelikož se všechny funkce do okna nevejdou, je jich zobra-

zeno menší množství a jsou na více stránkách, mezi kterými je možné se pohybovat. Při výběru funkce dojde v závislosti na počtu operandů k přesměrování na **vyskakovací okno pro vložení operandů funkce** 4.6. To se skládá z názvu vybrané funkce, jejího popisu a operandů k zadání. Počet operandů se zobrazí v závislosti na výběru funkce. Po vyplnění hodnot operandů může uživatel funkci vložit tlačítkem `Insert`. Poté je vyskakovací okno zavřeno a nacházíme se opět v okně hlavního nástroje 4.3.

Kromě přidání pravidla je zde také možnost pravidlo odebrat (ikona koše) nebo je možné kliknutím na tlačítko `Reset` odebrat všechna pravidla. Pro editaci pravidla opět stačí kliknout na řádek s konkrétním pravidlem a poté je postup stejný, jak již bylo popsáno.

Pro uložení souboru slouží tlačítko `Save file`. Při ukládání probíhá kontrola, zda má název souboru příponu `.txt` nebo `.rule`. Pokud ne, tak se zobrazí chybová hláška a uložení není umožněno. Také probíhá kontrola, zda již neexistuje sada se stejným názvem.

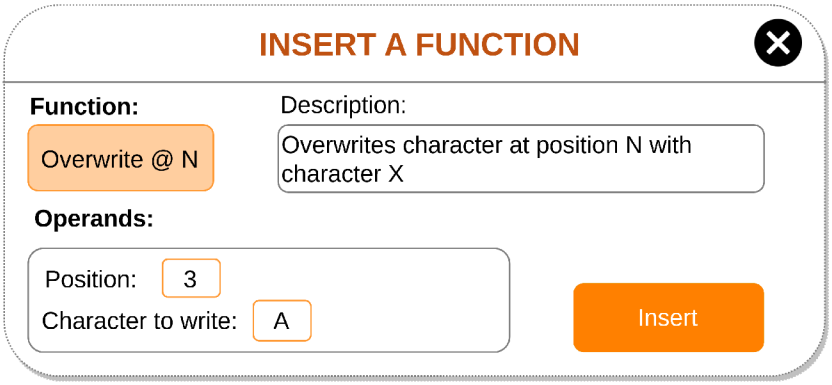


Obrázek 4.5: Vyskakovací okno s nabídkou všech dostupných funkcí

Uchování dat o funkcích pracujících s pravidly

Pro účel zobrazení všech dostupných funkcí a jejich bližších informací je nutné uchovávat o všech funkcích tyto atributy:

- název funkce,
- znak funkce,
- popis funkce,
- operandy funkce, konkrétně jejich počet, datové typy a popis operandu,
- praktický příklad funkce k použití,
- vstupní heslo pro použití funkce v příkladu,
- modifikované heslo po použití funkce v příkladu.



Obrázek 4.6: Zadávání operandů funkce

4.4 Úpravy v nástroji Fitcrack

Kromě přidání samotného nástroje jsou také nutné úpravy již existujících komponent a přidání nových:

- **Tlačítko pro editaci existující sady pravidel** – U zobrazení všech sad pravidel (viz obrázek 2.5) a také u zobrazení konkrétní sady (viz obrázek 2.6) je nutné přidat tlačítko pro jejich editaci. Při kliknutí proběhne přesměrování do nástroje.
- **Tlačítko pro vytvoření nové sady pravidel** – Tlačítko se bude vyskytovat nalevo od komponenty pro nahrání souboru s pravidly ze souborového systému uživatele (viz obrázek 2.5). Při kliknutí na tlačítko proběhne přesměrování do nástroje a umožnění vytvoření nové sady pravidel.

Kapitola 5

Implementace

V této kapitole jsou popsány implementační detaily nástroje a způsob integrace do systému Fitcrack. Nejprve je popsán způsob implementace aplikátoru pravidel a generátoru náhodných pravidel. V dalších sekcích je detailně popsán vývoj frontendu a backendu.

Pro správný návrh a implementaci bylo potřeba mít dobře nastudovanou dosavadní strukturu a implementaci systému Fitcrack. Kvůli integraci nástroje bylo totiž nutné zohlednit jak výběr technologií, tak strukturu zdrojových souborů. V každé sekci jsou zmíněny mnou vytvořené skripty, ale i upravené již existující.

Pro vývoj bylo nutné vytvořit *fork* repozitáře[1] nástroje Fitcrack. Z větve *dev* byla poté vytvořena nová větev *rules*, ve které probíhal vývoj. Při vývoji jsem využíval vývojové prostředí Visual Studio Code[2].

5.1 Aplikátor pravidel a generátor náhodných pravidel

Jak již bylo zmíněno v kapitole 4.2, nástroj podporuje živé aplikování pravidel na specifikovaná hesla a generování náhodných pravidel. Pro implementaci obou částí byla využita sada nástrojů `hashcat-utils`[3], uvolněná pod licencí MIT. Obsahuje několik skriptů, které jsou užitečné pro pokročilé lámání hesel.

Konkrétně byly využity 2 funkce ze skriptu `cpu_rules.c`. Pro použití funkcí jazyka C v jazyce Python bylo nutné vytvořit ze skriptu sdílenou knihovnu. Vytvořená knihovna `cpu_rules.so` byla poté pro účel spuštění na serveru přesunuta do složky *hashcat–utils/src* (viz obrázek 5.1) a z ní je při sestavení obrazu pomocí nástroje docker vložena na server. Knihovna byla vytvořena tímto příkazem:

```
gcc -shared -fPIC -o cpu_rules.so cpu_rules.c
```

Aplikátor pravidel využívá funkci `apply_rule_cpu()` s následující deklarací:

```
int apply_rule_cpu (char *rule,
                    int rule_len,
                    char in[BLOCK_SIZE],
                    int in_len,
                    char out[BLOCK_SIZE])
```

[1] https://github.com/nesfit/fitcrack
[2] https://code.visualstudio.com/
[3] https://github.com/hashcat/hashcat-utils/

Její parametry postupně vyjadřují řetězec s pravidlem k použití, délku pravidla, heslo k modifikaci, délku hesla a emphbuffer pro výsledné modifikované heslo. Návratová hodnota znázorňuje délku modifikovaného hesla či případně chybový návratový kód.

Generátor náhodných pravidel využívá funkci `generate_random_rule()` s následující deklarací:

```
int generate_random_rule (char rule_buf[RP_RULE_BUFSIZ],
                          uint32_t rp_gen_func_min,
                          uint32_t rp_gen_func_max)
```

Její parametry postupně vyjadřují *buffer* pro vygenerované heslo, minimální počet vygenerovaných funkcí a maximální počet vygenerovaných funkcí. Návratová hodnota znázorňuje délku vygenerovaného pravidla.

5.1.1 Změny ve skriptu cpu_rules.c

V tomto skriptu bylo nutné pro účely vytvoření sdílené knihovny provést drobné úpravy. První úpravou byla změna typu proměnné *max_len* z externí na globální. Druhou úpravou byla změna funkce `generate_random_rule()`. Při testování generování náhodných pravidel byla totiž objevena chyba, díky které bylo v určité situaci vytvářeno nevalidní náhodné pravidlo. Důvodem bylo dosazení nesprávné proměnné na pozici v poli.

5.1.2 Knihovna ctypes

Pro volání funkcí z vytvořené sdílené knihovny byla použita knihovna *ctypes*[4] jazyka Python. Tato knihovna slouží k interakci s knihovnami napsanými v jiných jazycích, zejména s knihovnami napsanými v jazyce C. Díky použití knihoven jazyka C pak mohou vývojáři využívat výhod rychlosti a výkonu tohoto kompilovaného jazyka [18].

Nejprve bylo nutné pomocí funkce `CDLL()` načíst sdílenou knihovnu `cpu_rules.so`, která obsahuje požadovanou funkci. Poté bylo třeba specifikovat datový typ návratové hodnoty a typy parametrů, které funkce očekává. Nejedná se o klasické datové typy jazyka Python, knihovna obsahuje speciální datové typy odpovídající datovým typům jazyka C. Pro příklad uvedu deklaraci funkce pro generování náhodných pravidel:

```
genRandomRule = ctypes.CDLL(RULE_APPLICATOR_PATH).generate_random_rule
genRandomRule.restype = ctypes.c_int
genRandomRule.argtypes = [ctypes.c_char_p,
                          ctypes.c_uint32,
                          ctypes.c_uint32]
```

Uvedené datové typy parametrů a návratové hodnoty odpovídají deklaraci funkce v jazyce C, viz sekci 5.1. Postup pro funkci sloužící k aplikování pravidel byl velmi podobný.

5.2 Implementace backendu

V této sekci jsou popsány použité technologie pro vývoj backendu, struktura zdrojových souborů, implementace API rozhraní včetně popisu endpointů a potřebné úpravy v databázi.

[4] https://docs.python.org/3/library/ctypes.html

5.2.1 Použité technologie

Pro vývoj backendu byl použit jazyk `Python 3` a jeho framework `Flask`[5]. Jedná se o webový mikroframework, který se využívá k vytváření webových aplikací a služeb. Jeho výhodou je jednoduchost, flexibilita a také rozsáhlá dokumentace, díky čemuž je jedním z nejpopulárnějších webových frameworků pro Python. Obsahuje rozšíření pro tvorbu REST webových služeb, pro práci s databází, práci s webovými formuláři atd [14].

Při implementaci využívám konkrétně rozšíření `Flask-RESTX`[6]. Toto rozšíření poskytuje nástroje pro definování API. Velkou výhodou je jednoduché zpracování a validace vstupních dat (požadavku) a výstupních dat (odpovědi). Další výhodou je možnost vytváření *Swagger* dokumentace pomocí určených dekorátorů. Díky této vygenerované dokumentaci se zvyšuje přehlednost vytvářeného API.

Pro práci s relační databází je použita knihovna `SQLAlchemy`[7]. Díky ní není nutné vytvářet SQL dotazy, ale pouze vytvořit třídy pro strukturu tabulek a nad nimi poté používat vestavěné funkce této knihovny. Převod mezi objekty v Pythonu do SQL a opačně probíhá pomocí objektově relačního mapování. Díky této knihovně je kód také nezávislý na konkrétním databázovém systému a umožňuje tak snadnější přenos aplikací mezi různými prostředími [9].

5.2.2 Struktura zdrojových souborů

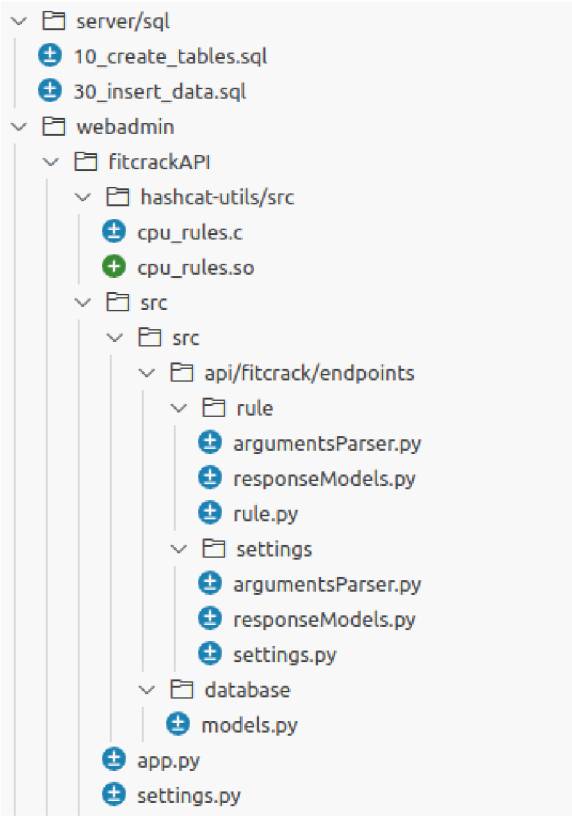
Na obrázku 5.1 lze vidět strukturu zdrojových souborů, které jsem upravoval (modrá barva) nebo vytvářel (zelená barva). Ve složce */server/sql* se nachází SQL skripty, které pracují s databází, více v sekci 5.2.4. Ve složce */webadmin/fitcrackAPI* se nachází veškeré soubory pracující s API systému Fitcrack. Byly upraveny soubory v těchto jejích podsložkách:

- *hashcat–utils/src* – Obsahuje soubor se sdílenou knihovnou a skript v jazyce C, který je pro něj základem, více v sekci 5.1.
- *src/src/api/fitcrack/endpoints/rule* – Obsahuje definici endpointů pro práci s pravidly, více v podsekci 5.2.3.
- *src/src/api/fitcrack/endpoints/settings* – Obsahuje definici endpointů pro práci s nastavením, více v podsekci 5.2.4.
- *src/src/database* – Obsahuje soubor s definicí tříd a sloupců relační databáze, změny jsou popsány v podsekci 5.2.4.

5.2.3 API rozhraní

V této sekci jsou popsané endpointy vytvořené pro interaktivní nástroj, způsob jejich validace a dokumentace. Všechny jsou definovány ve skriptu `rule.py`, modely jejich vstupů ve skriptu `argumentsParser.py` a modely jejich odpovědí ve skriptu `responseModels.py`.

[5] https://flask.palletsprojects.com/en/2.2.x/
[6] https://flask-restx.readthedocs.io/en/latest/index.html
[7] https://www.sqlalchemy.org/



Obrázek 5.1: Struktura zdrojových souborů backendu

Endpoint pro aplikování pravidel na sadu hesel

Tento endpoint reprezentuje HTTP požadavek typu POST s cestou `/rule/preview`. Jsou přijímány 2 vstupní parametry, které jsou validovány modelem `preview_request`:

1. `passwords` – znázorňuje sadu hesel pro modifikaci
2. `rules` – znázorňuje sadu pravidel pro modifikaci.

Endpoint slouží k aplikování pravidel na sadu hesel. Aplikování probíhá tím způsobem, že v cyklu, pro každou kombinaci hesla a pravidla přijatých z parametrů požadavku, je zavolána funkce `apply_rule_cpu()` ze sdílené knihovny (viz sekci 5.1). Tím je získáno modifikované heslo a návratová hodnota funkce, kterou je ještě dodatečně nutné upravit pro účel rozpoznání komentáře v pravidlu. Výsledná návratová hodnota `retCode` poté indikuje několik možných stavů, které jsou zobrazeny v tabulce 5.1. Těchto hodnot je následně využíváno při živé kontrole validity pravidel.

Návratová hodnota	Popis
>= 0	Délka hesla po modifikaci
-1	Syntax error
-2	Prázdný řetězec pravidla nebo hesla
-3	Komentář

Tabulka 5.1: Množina návratových hodnot funkce pro modifikaci hesel a jejich popis

Tyto získané hodnoty z funkce jsou vždy vloženy do objektu. Odpověď formátu JSON (validována modelem `preview_response`) je pole těchto objektů `passwordsPreview`, definované následovně:

```
{
  "passwordsPreview": [
    {
      "mangledPassword": string,
      "retCode": integer
    }
  ]
}
```

Protože je v nástroji nastavena výchozí hodnota maximálního počtu modifikovaných hesel, je při každém průchodu cyklem také kontrolováno, zda nebyla maximální hodnota překročena. Při dosažení limitu nejsou všechna hesla modifikována.

Endpoint pro změnu sady pravidel

Tento endpoint reprezentuje HTTP požadavek typu PUT s cestou `/rule/{id}`. Vstupním parametrem endpointu je soubor obsahující aktualizovanou sadu pravidel. Endpoint slouží k aktualizování existující sady pravidel a jejímu možnému přejmenování. Má na starost následující úkony:

1. Kontrola, zda není upravovaná sada pravidel namapovaná k lámací úloze. Pokud ano, tak není sadu možné aktualizovat a je vrácena odpověď s chybovou návratovou hodnotou 500 a odpovídající chybová hláška. Pokud by nebyla kontrola provedena, tak by byly změněny počáteční podmínky lámací úlohy a tím by se změnil i její zamýšlený průběh. Lámací úloha by také mohla v průběhu skončit chybou.
2. Kontrola, zda nový název sady pravidel obsahuje validní koncovku. Povolenými koncovkami jsou `.rule` a `.txt`. Pokud obsahuje nevalidní koncovku, tak je vrácena odpověď s chybovou návratovou hodnotou 500 a odpovídající chybová hláška.
3. Kontrola, zda již na serveru ve složce */usr/share/collections/rules* neexistuje soubor se stejným názvem. Pokud existuje, tak je vrácena odpověď s chybovou návratovou hodnotou 500 a odpovídající chybová hláška.
4. Výpočet nového počtu pravidel v aktualizované sadě.
5. Pokud je vše v pořádku, tak proběhne přejmenování souboru na serveru, přepis jeho obsahu novou sadou pravidel a změna záznamu této sady v databázi. Zahrnuje to aktualizaci jména, cesty k souboru a nového počtu pravidel.

Validace odpovědi je definována modelem `simpleResponse`. Při úspěšné aktualizaci obsahuje odpověď status a potvrzující hlášku:

```
{
  "status": boolean,
  "message": string
}
```

Endpoint pro vygenerování náhodného pravidla

Tento endpoint reprezentuje HTTP požadavek typu GET s cestou `/rule/randomRule`. Nepřijímá žádné vstupní parametry. Slouží k vygenerování náhodného pravidla, pro což využívá postup zmíněný v sekci 5.1. Toto náhodné pravidlo obsahuje od 3 do 8 funkcí. Validace odpovědi je definována modelem `randomRule_response`. Odpověd ve formátu JSON obsahující vygenerované náhodné pravidlo je definována následujícím způsobem:

```
{
  "randomRule": string
}
```

Dokumentace a validace

Pro dokumentaci a validaci jednotlivých endpointů byly použity dekorátory knihovny flask–RESTX. Konkrétně bylo využito těchto 3 dekorátorů:

- `@api.expect()` – Tato funkce definuje očekávaný formát dat pro určitý endpoint. Její použití umožňuje ověřit, zda jsou data, která jsou odeslána na endpoint, ve správném formátu a zda obsahují všechny potřebné informace.
- `@api.marshal_with()` – Tato funkce definuje formát dat, která jsou vrácena z endpointu. Používá se pro serializaci dat.
- `@api.response()` – Tato funkce umožňuje definovat odpověď, která bude vrácena z endpointu v případě úspěšného nebo neúspěšného požadavku [13].

Poté je automaticky vygenerována dokumentace pomocí nástroje `Swagger UI`, který poskytuje grafické rozhraní pro prohlížení a interakci s dokumentací API. Obsahuje seznam endpointů, jejich parametrů a návratových hodnot. Při spuštění kontejneru dle návodu lze tuto dokumentaci prohlížet na portu 5000.

5.2.4 Úpravy v databázi

Pro dosažení funkcionality, která umožňuje nastavit maximální možný počet modifikovaných hesel pro aplikátor pravidel (viz sekci 5.3.4) byly provedeny změny ve 2 SQL skriptech a několika Python skriptech. Ve skriptu `10_create_tables.sql` bylo nutné upravit strukturu tabulky `fc_settings`, konkrétně byl přidán sloupec `max_mangled_passwords` vyjadřující maximální počet modifikovaných hesel. Ve skriptu `30_insert_data.sql` byla do tohoto sloupce přidána výchozí hodnota '5000'. Pro konzistenci byl upraven skript `database/models.py`, ve kterém byla upravena definice třídy `FcSetting` tak, aby obsahovala nový sloupec. Dále byla nutná úprava endpointu `/settings` HTTP metody POST, aby jako parametr přijímal maximální počet modifikovaných hesel a uložil tuto hodnotu do databáze. Nakonec byl ve skriptu `argumentsParser.py` aktualizován očekávaný vstup tohoto endpointu a také byla aktualizována definice jeho odpovědi ve skriptu `responseModels.py`.

5.3 Implementace Frontendu

V této sekci jsou popsány použité technologie pro vývoj frontendu, struktura zdrojových souborů, routování v nástroji a způsob uchování dat funkcí pracujících s pravidly. Dále je

popsáno grafické uživatelské rozhraní a je vysvětlen význam jednotlivých komponent, jejich implementační detaily a jejich ovládání.

5.3.1 Použité technologie

Pro vývoj frontendu byl použit javascriptový framework `Vue.js 2`[8] a pro komponenty uživatelského rozhraní framework `Vuetify 2`[9]. Pro komunikaci s API a zprostředkování požadavků byla využita knihovna `Axios`.

Vue.js je javascriptový framework pro tvorbu webových aplikací a uživatelských rozhraní. Využívá reaktivního přístupu, rozdělení na komponenty a díky tomu je uživatelské rozhraní rychle renderováno. Každá komponenta se skládá ze 3 částí. První částí je `template`, který obsahuje HTML kód a definuje vzhled komponenty. Druhou částí je `script`, což je kód v jazyce Javascript, který obsahuje definice dat a využívaných metod. Třetí, volitelnou částí, je `style`, která obsahuje CSS kód definující vzhled komponenty. Díky jednoduché syntaxi je tedy tento framework snadno použitelný a oblíbený [2].

Vuetify je open–source framework pro tvorbu uživatelského rozhraní, který poskytuje sadu připravených komponent jako jsou tlačítka, formulářové prvky, ikony, dialogy a mnoho dalších. Je navržen tak, aby zajistil konzistentní vzhled a chování prvků v celé aplikaci.

5.3.2 Struktura zdrojových souborů

Na obrázku 5.2 lze vidět strukturu zdrojových souborů, které jsem vytvářel (zelená barva) nebo upravoval (modrá barva). Nachází se ve složce *webadmin/fitcrackFE/src*. V podsložce *components/rule* se nalezneme komponenty pracující s pravidly. V podsložce *components/-settings* se najdeme komponenty zobrazující nastavení. V podsložce *router* se nachází soubor `index.js` s definicí routování pro aplikaci. Nakonec ve složce *assets* nalezneme statické soubory sloužící pro aplikaci. Konkrétně zde byl přidán soubor `ruleFunctions.json` s definicí struktury pravidlových funkcí. Tyto skripty a soubory budou více popsány v následujících sekcích.

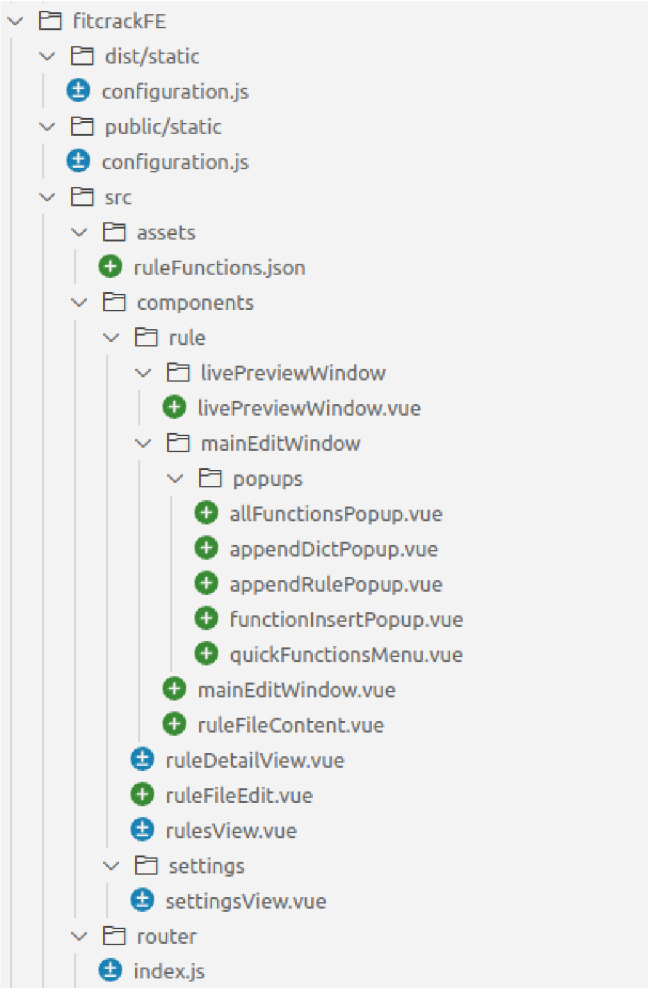
5.3.3 Routování a přesměrování do nástroje

O změnu zobrazení komponentů dle nastavené cesty v prohlížeči se stará nástroj `vue-router`. Routování je definované v souboru `index.js`, ve kterém byly pro účel nástroje vytvořeny 2 nové cesty. Cesta `/rules/edit/new` se stará o přesměrování do nástroje pro vytvoření nové sady pravidel. Cesta `/rules/edit/:id` se stará o přesměrování na úpravu existující sady pravidel, kde `id` je identifikátor sady pravidel v databázi. Zde je úryvek kódu pro definici této cesty:

```
{
   path: '/rules/edit/:id',
   name: 'ruleFileEdit',
   component: ruleFileEdit,
}
```

Tlačítka, při jejichž kliknutí dochází k přesměrování do nástroje, lze vidět na obrázku 5.3. Pokud uživatel zadává URL ručně do adresního řádku prohlížeče a uvede neexistující

[8] https://v2.vuejs.org/
[9] https://v2.vuetifyjs.com/



Obrázek 5.2: Struktura zdrojových souborů frontendu

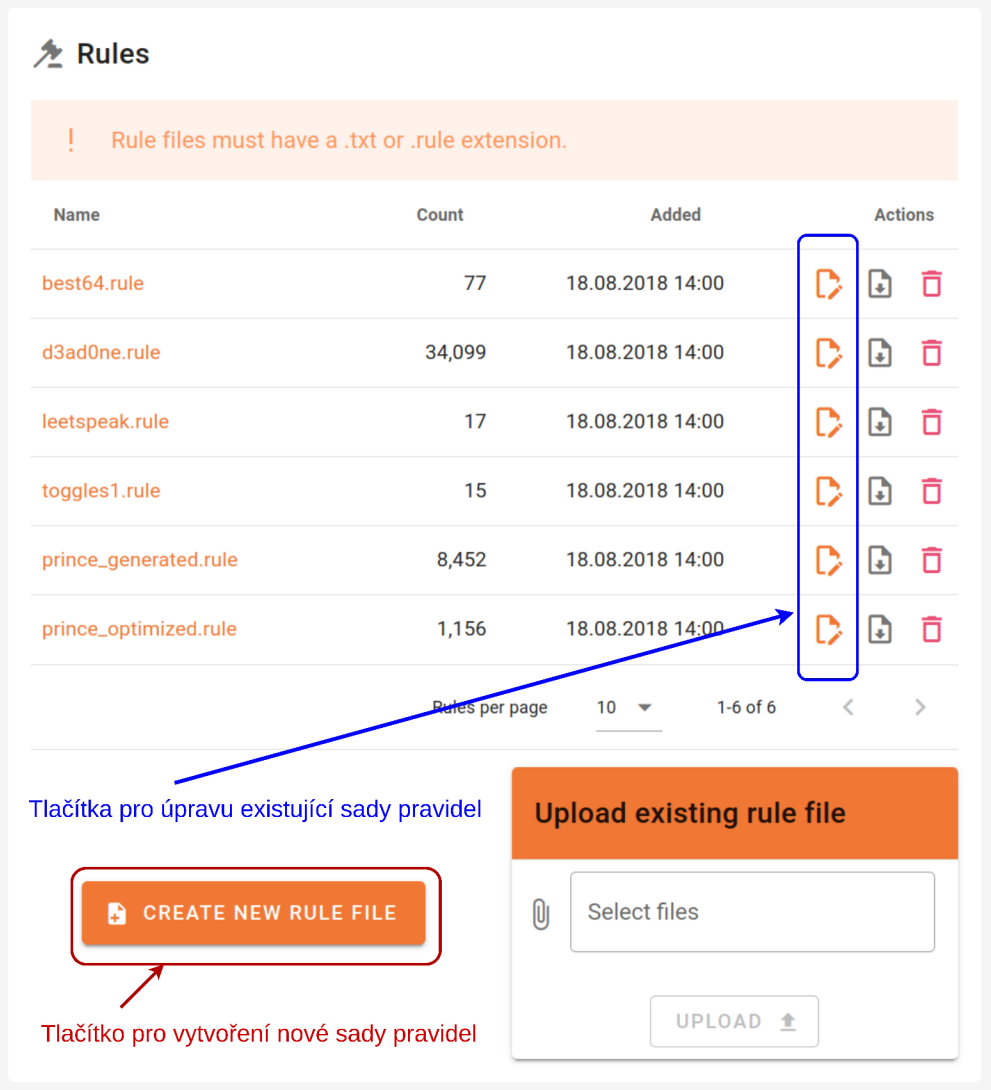
`id`, tak je zobrazena chybová hláška o neexistenci sady pravidel a uživatel je pomocí funkce `router.push()` přesměrován na cestu `/rules/edit/new` a je mu alespoň umožněna tvorba nové sady.

5.3.4 Grafické uživatelské rozhraní

V této sekci se zaměřím na popis grafického uživatelského rozhraní nástroje. Hlavním cílem bylo vytvořit uživatelsky přívětivé rozhraní, které je intuitivní a snadno použitelné pro uživatele.

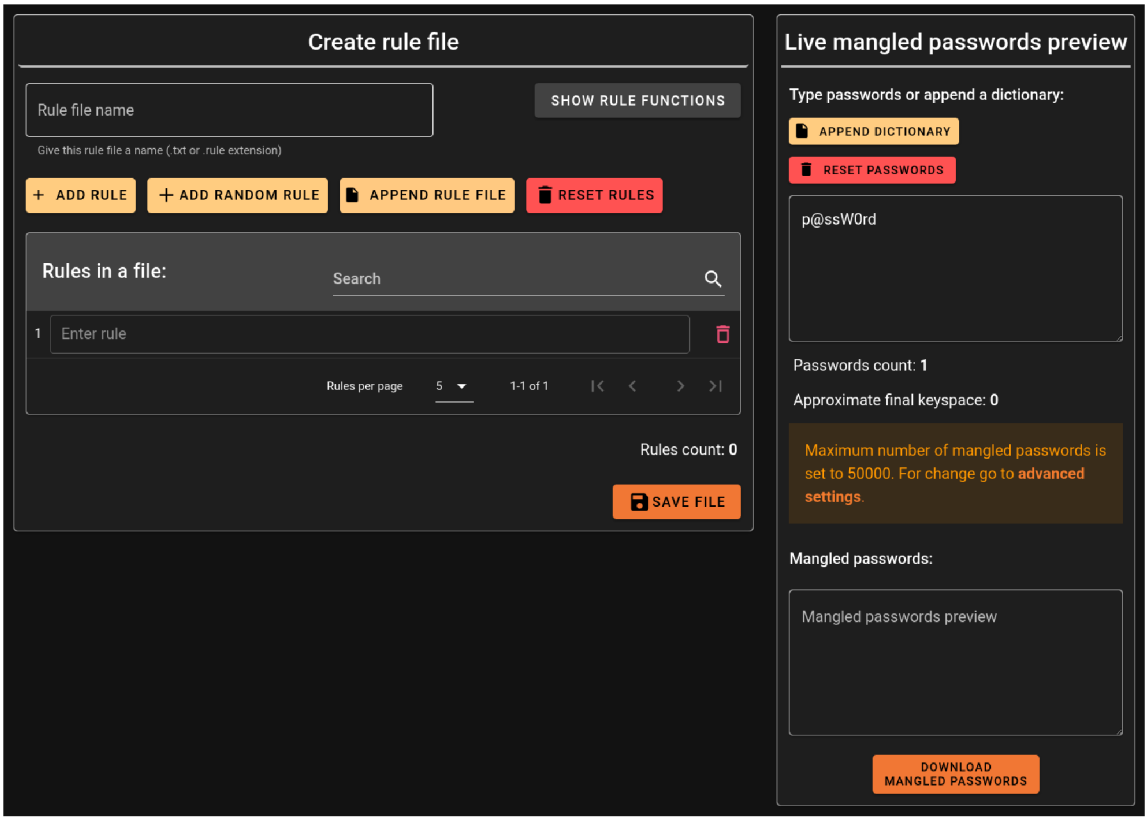
Důležitou vlastností, na kterou jsem také kladl důraz, byla responzivita. Při programování byly komponenty navrženy tak, aby se při změnách velikosti obrazovky měnila jejich velikost a také celkové rozložení nástroje. S nástrojem lze díky tomu příjemně pracovat bez ohledu na to, na jakém zařízení je používán, klidně i na mobilním telefonu.

Při programování komponent a návrhu jejich vzhledu bylo také nutné zohlednit fakt, že systém Fitcrack podporuje 2 režimy zobrazení, konkrétně světlý a tmavý mód. Na obrázku 5.4 lze vidět výsledný nástroj při použití tmavého módu. Veškeré další ukázky komponent budou prezentovány ve světlém módu.



Obrázek 5.3: Zobrazení všech sad pravidel a možnosti přesměrování do nástroje

Nástroj je rozdělen do několika komponent. Jejich funkčnost a rozmístění odpovídá návrhu v kapitole 4.3. Tyto komponenty se nachází ve složkách *components/rule* a *components/settings*. Hlavní komponentou nástroje je `ruleFileEdit.vue`, která se skládá ze 2 oken. Jde o okno pro vytváření/úpravu sady pravidel a okno pro živé zobrazení modifikovaných hesel, viz obrázek 5.4. Tato hlavní komponenta také uchovává globální data sloužící pro celý nástroj, včetně dat sloužících pro zobrazení vyskakovacích oken. Data jsou poté směrem od rodičovských k dětským komponentům předávány pomocí direktivu `v-bind`. Pro aktualizaci dat z dětských do rodičovských komponentů je použit direktiv `v-on` a funkce `emit()`.



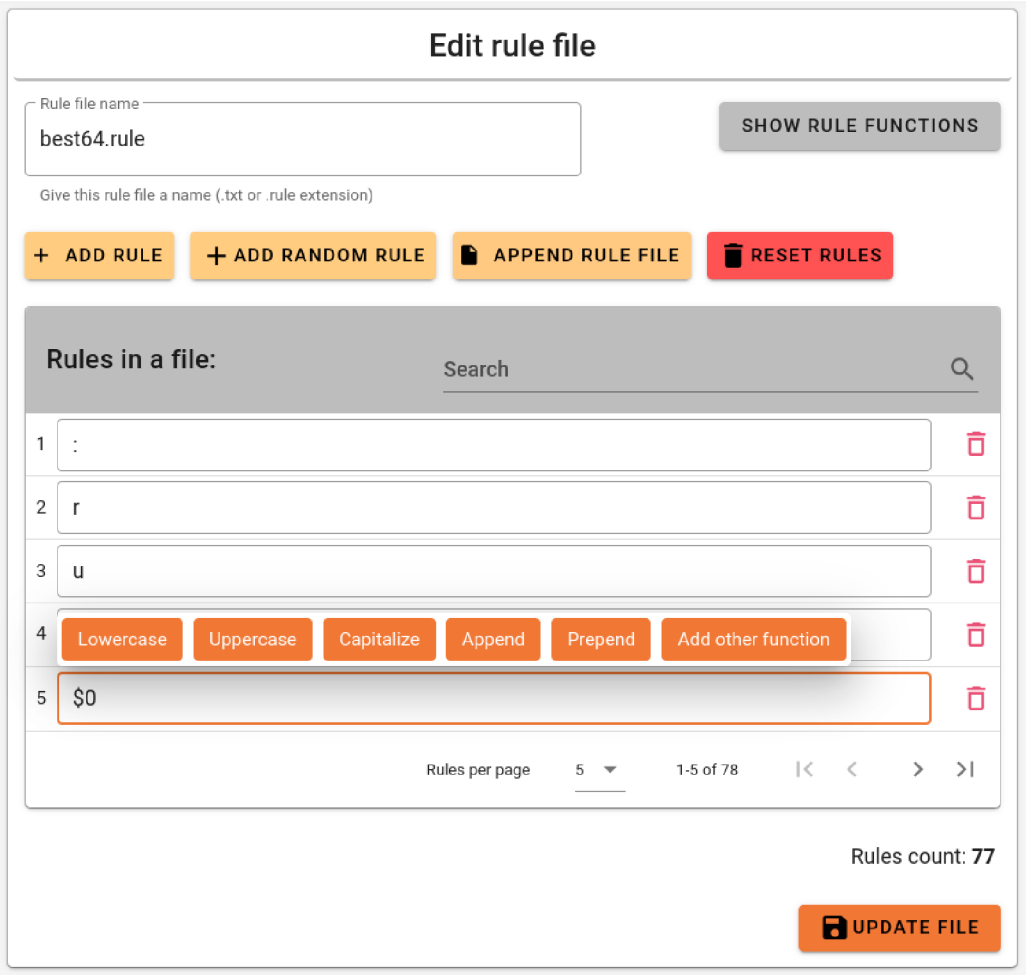
Obrázek 5.4: Zobrazení nástroje v tmavém módu

Okno pro vytváření/úpravu sady pravidel

Toto okno slouží k vytváření a editaci sady pravidel a je obsaženo v komponentě `mainEditWindow.vue`, viz obrázek 5.6. Součástí této komponenty je také okno pro úpravu a přidávání jednotlivých pravidel, obsaženo v komponentě `ruleFileContent.vue`. Při kliknutí na konkrétní pravidlo je umožněno pravidlo vytvářet interaktivně (skrze tlačítka a vyskakovací okna).

Druhou možností je přidávání a úprava pravidel ručně, skrze klávesnici. Pozici v pravidlu lze měnit kurzorem myši nebo příslušnými tlačítky na klávesnici. Z důvodu ruční specifikace pravidel probíhá jejich živá validace. To kvůli tomu, že může jednoduše dojít k překlepu na klávesnici a tím pádem k nevalidině pravidla. Pro tento účel je využito metody `generatePreview()`. Na základě návratového kódu obdrženého z odpovědi požadavku je ke každému pravidlu přidělen příznak `error` vyjadřující informaci o validitě pravidla. Pokud není pravidlo validní, tak se zobrazí oranžová chybová ikona, viz obrázek 5.5.

Obrázek 5.5: Pravidlo a jeho validace



Obrázek 5.6: Hlavní okno nástroje

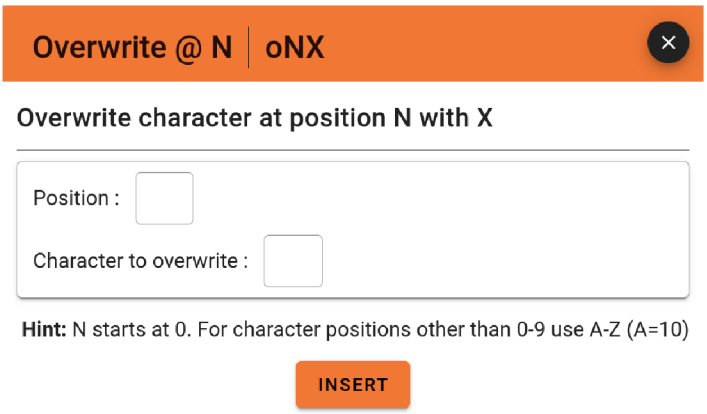
Okno obsahuje několik komponent, které slouží jako vyskakovací okna. Tyto komponenty se nachází ve složce *rule/mainEditWindow/popups*. Jedná se o následující:

1. `quickFunctionsMenu.vue` – Toto okno se zobrazí vždy při kliknutí na vstupní pole konkrétního pravidla. Lze ho vidět na obrázku 5.6 nad zvoleným pravidlem.

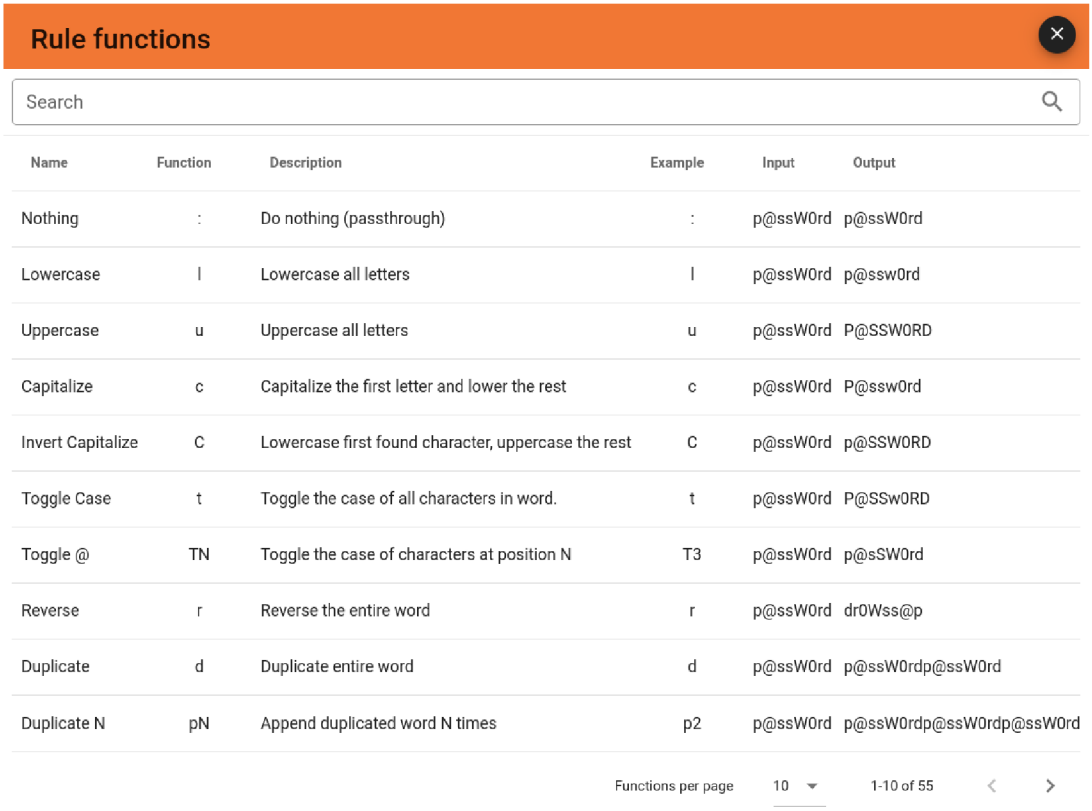
2. `functionsInsertPopup.vue` – Toto vyskakovací okno slouží pro vložení konkrétní funkce včetně vyplnění operandů. Při potvrzení vložení probíhá kontrola správnosti operandů na základě jejich specifikovaných datových typů. K tomu je využíván soubor *ruleFunctions.json*, viz sekci 5.3.5. Vzhled vyskakovacího okna lze vidět na obrázku 5.7.

3. `allFunctionsPopup.vue` – Toto vyskakovací okno slouží k zobrazení všech dostupných funkcí pracujících s pravidly a lze ho vidět na obrázku 5.8.

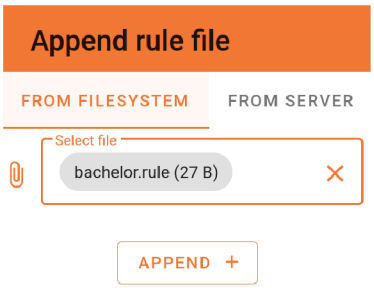
4. `appendRulePopup.vue` – Toto vyskakovací okno slouží pro zvolení sady pravidel, která je připojena za již existující pravidla. Uživatel má 2 možnosti. První možností je výběr sady pravidel ze souborového systému uživatelova počítače, viz obrázek 5.9. Druhou možností je výběr sady pravidel ze serveru, viz obrázek 5.10.



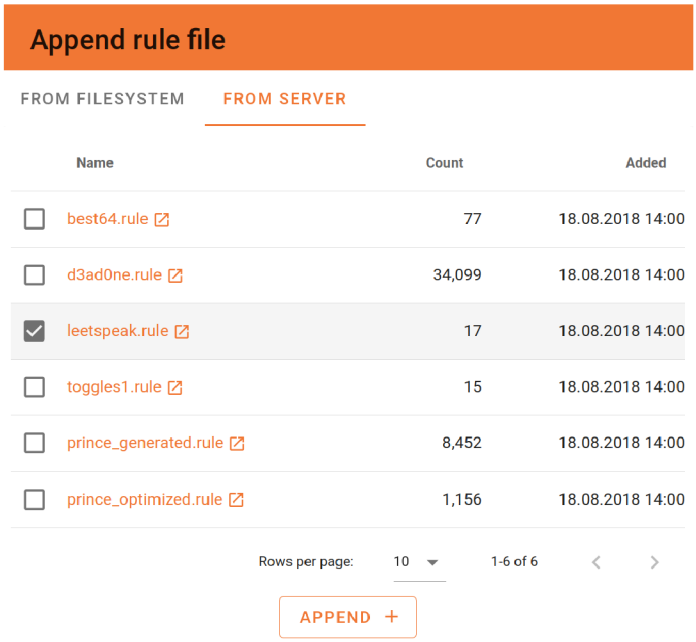
Obrázek 5.7: Vyskakovací okno pro vložení funkce



Obrázek 5.8: Vyskakovací okno se zobrazením všech dostupných funkcí



Obrázek 5.9: Okno pro připojení sady pravidel ze souborového systému uživatele



Obrázek 5.10: Okno pro připojení sady pravidel ze serveru

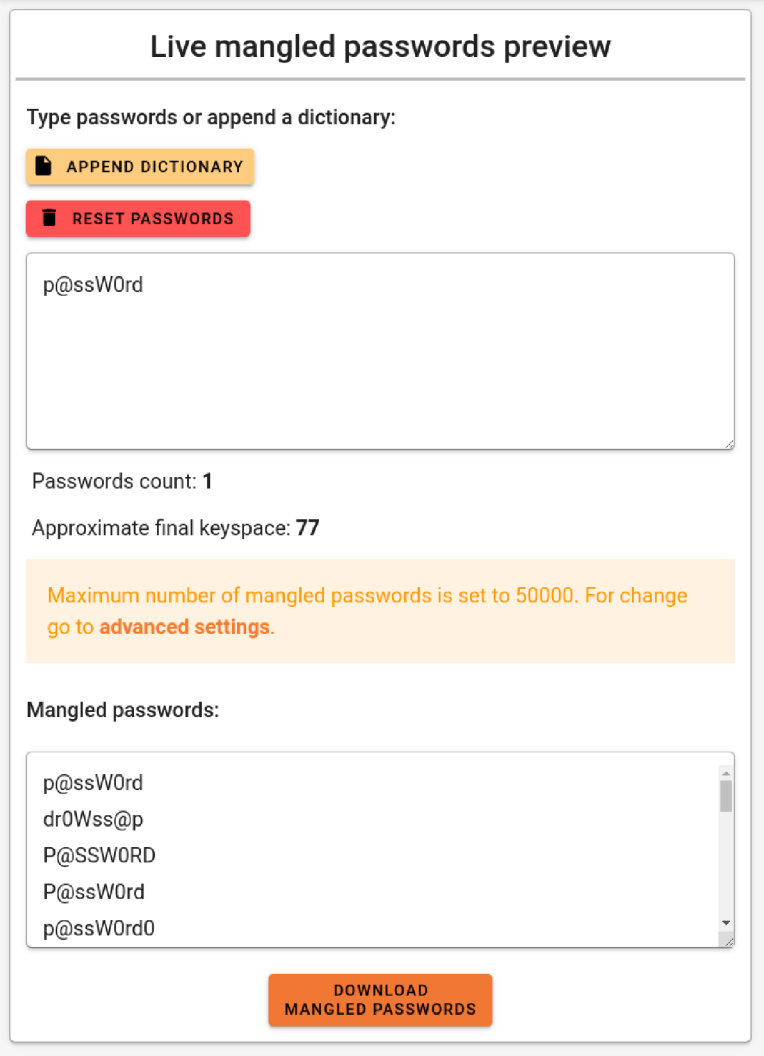
Okno s živým náhledem hesel po modifikaci

Toto okno slouží k živému náhledu hesel po modifikaci a specifikování hesel k modifikaci. Je obsaženo v komponentě `livePreviewWindow.vue`, vzhled viz obrázek 5.11. Při vstupu do nástroje lze tuto komponentu vidět vpravo, při menších rozměrech obrazovky se nachází pod oknem sloužícím pro vytváření a úpravu pravidel.

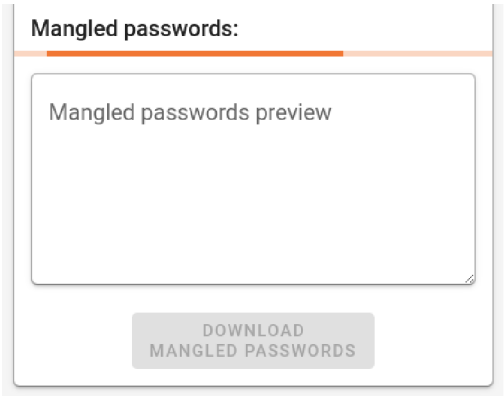
Pro vkládání hesel lze využít jak ruční vkládání (psaní na klávesnici), tak připojení již existujícího slovníku. Pro jeho připojení slouží komponenta `appendDictPopup.vue`, která nabízí 2 možnosti. První možností je výběr slovníku ze souborového systému uživatelova počítače, viz obrázek 5.13. Druhou možností je výběr slovníku ze serveru, viz obrázek 5.14.

Generování modifikovaných hesel probíhá pokaždé, pokud se změní vstupní pravidla nebo vstupní hesla. V těchto případech je zavolána metoda `generatePreview()`, která pomocí knihovny Axios zašle požadavek s aktuálními specifikovanými pravidly a hesly. Po obdržení odpovědi je následně výsledek zpracován, konkrétně jsou vyfiltrována pouze validní modifikovaná hesla. Pro zamezení příliš častých dotazů na server je využita funkce `debounce()` knihovny `loadash`. Díky této funkci jsou modifikovaná hesla generována vždy až určitou dobu po poslední změně pravidel či hesel. Konkrétně byl časovač nastaven na dobu 1,5 sekundy.

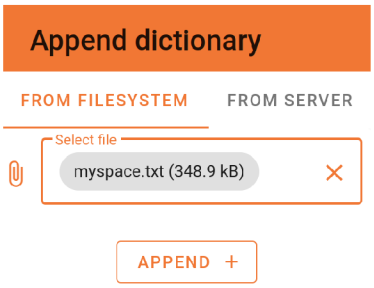
Aby byl uživatel informován o průběhu a dokončení modifikace hesel (konkrétně doba od změny vstupních dat po vygenerování modifikovaných hesel), tak je zobrazena načítací komponenta (viz obrázek 5.12) a po tuto dobu není umožněno stáhnout soubor s modifikovanými hesly.



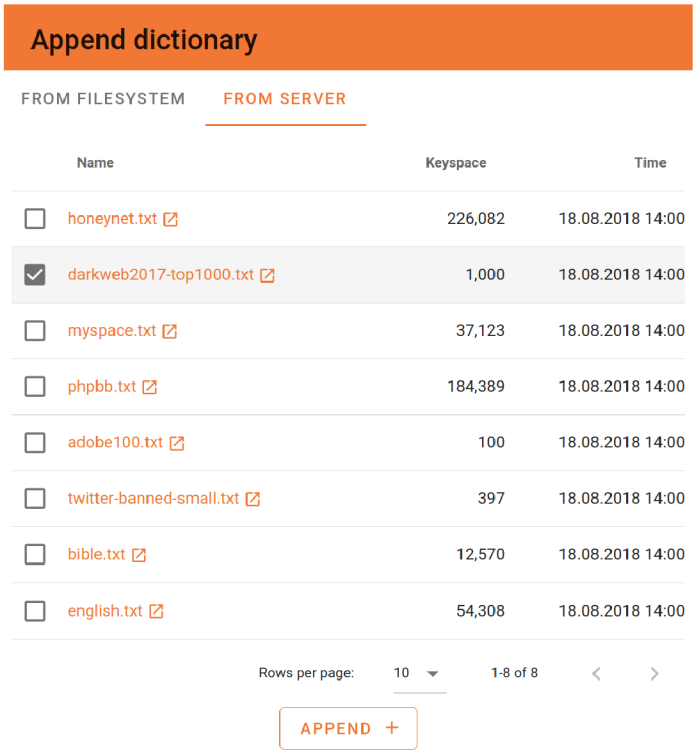
Obrázek 5.11: Okno s živým náhledem hesel po modifikaci



Obrázek 5.12: Ukázka načítání modifikovaných hesel



Obrázek 5.13: Okno pro připojení slovníku ze souborového systému uživatele

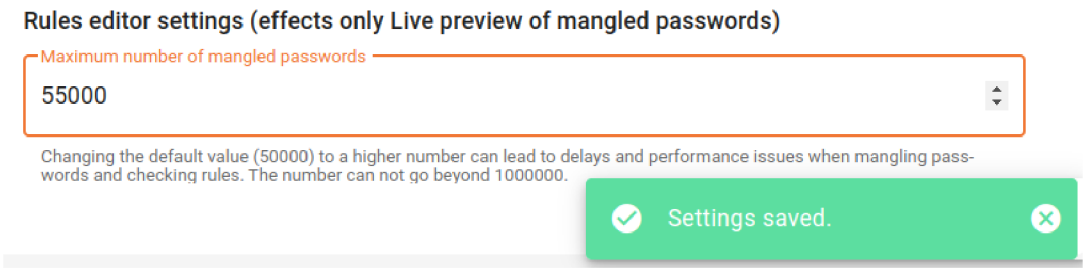


Obrázek 5.14: Okno pro připojení slovníku ze serveru

Nastavení maximálního počtu modifikovaných hesel

Pro optimalizaci nástroje je nastaven maximální počet hesel, která mohou být vygenerována. Důvodem je zamezení dlouhé doby čekání na jejich vygenerování (při vysokém počtu pravidel a hesel). V databázi je tedy nastavena výchozí hodnota maximálního počtu, viz podsekci 5.2.4.

Uživatel je o tomto omezení informován v komponentě `livePreview.vue` prostřednictvím okna typu `v-alert`, které uživatele upozorňuje na nastavený maximální počet. Toto okno obsahuje také odkaz do nastavení. Hodnotu maximálního počtu lze změnit v sekci pokročilého nastavení s názvem `Rules editor settings`, viz obrázek 5.15. Jedná se o část komponenty `settingsView.vue`. Při změně maximálního počtu je hodnota ihned uložena do databáze, ještě předtím však probíhá kontrola, zda hodnota nepřesáhla hranici 1 milionu. Takto velký obsah dat přenášený přes webový prohlížeč by totiž mohl způsobit zvýšení odezvy při jeho používání nebo dokonce jeho zamrznutí.



Obrázek 5.15: Nastavení maximálního počtu modifikovaných hesel a potvrzující hláška

5.3.5 Uchování dat o funkcích pracujících s pravidly

Pro způsob uchování dat o jednotlivých funkcích byl zvolen soubor formátu JSON s názvem `ruleFunctions.json`. Jedná se o pole objektů, kde každý objekt reprezentuje konkrétní funkci. Obsahuje atributy, které pořadím i významem odpovídají návrhu v sekci 4.3. Tato data jsou využívána při zobrazení dostupných funkcí i při samotném vkládání funkcí. Pro validaci operandů slouží pole objektů `operands` a jejich atribut `type`. Zde je uveden příklad uchování dat konkrétní funkce:

```
{
    "name": "Duplicate N",
    "sign": "pN",
    "description": "Append duplicated word N times",
    "operands": [
        {
            "specification": "Number of times",
            "type": "int"
        }
    ],
    "example": "p2",
    "input": "p@ssW0rd",
    "output": "p@ssW0rdp@ssW0rdp@ssW0rd"
}
```

5.3.6 Kontrola změny pravidel při změně URL nebo obnově stránky

Pro předejití ztráty dat při vytváření a editaci sady pravidel je při každé změně URL a znovunačtení stránky prováděna kontrola změny pravidel. K té dochází pomocí metody `rulesChanged()`. Pokud je vytvářena nová sada, tak dochází k porovnání současných pravidel s prázdným řetězcem. Pokud je editována existující sada, tak jsou porovnána současná pravidla s těmi, které byly obdrženy při počátečním načtení existující sady. Pokud je zjištěn rozdíl, tak je uživateli zobrazeno potvrzovací okno, ve kterém je dotázán, zda chce stránku opravdu opustit a ztratit tak provedené změny.

Kapitola 6

Experimenty

Pro předvedení funkčnosti aplikace a jejích hlavních případů užití byly provedeny 4 experimenty. Cílem prvních 3 experimentů je ukázat očekávané chování aplikace, různé způsoby vytváření pravidel a také kontrolu validity některých vstupů. Experiment 6.4 poté analyzuje dobu běhu aplikátoru pravidel v závislosti na počtu generovaných hesel. Pro všechny experimenty platily následující **společné podmínky**:

- instalace nástroje Fitcrack skrze *Docker*:
  - *Docker* verze 23.0.2,
  - *docker–compose* verze 1.29.2,
  - sestavení a spuštění kontejneru podle návodu[1].
- ponechání výchozí konfigurace nástroje Fitcrack včetně souborů s pravidly ve složce *Collections* beze změny,
- vytvořená lámací úloha, která pro útok využívá sadu pravidel `toggles1.rule`,
- parametry zařízení, na kterém byly prováděny experimenty:
  - operační systém Ubuntu 22.04.2 LTS,
  - procesor AMD® Ryzen 5 4600h with Radeon Graphics, 3.00 GHz,
  - grafická karta NVIDIA GeForce GTX 1650 Ti, 4 GB GDDR6.

6.1 Vytvoření nové sady pravidel

Cílem tohoto experimentu je ukázat tvorbu nové sady pravidel a její uložení, včetně možných způsobů tvorby jednotlivých pravidel.

Postup experimentu

1. Přesuneme se na webovou adresu *localhost/rules*. Stisknutím tlačítka `Create new file` je zobrazeno hlavní okno pro tvorbu sady pravidel.
2. Do okna `Rule file name` zadáme název souboru `toggles1.rule`.

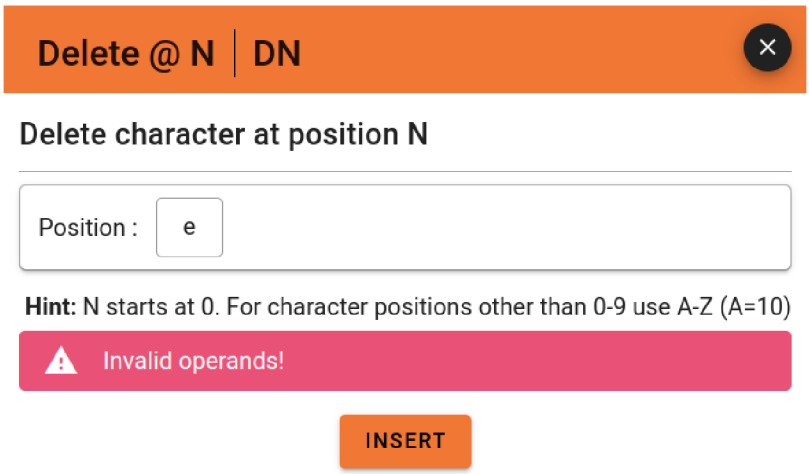
[1] https://github.com/nesfit/fitcrack/blob/master/INSTALL-Docker.md

3. Pro vytvoření prvního pravidla klikneme do okna s popisem `Enter rule`. Při kliknutí na toto pravidlo je zobrazena interaktivní nabídka funkcí, pravidlo ale nejprve zadáme ručně skrze klávesnici, konkrétně `"c $1"`.

4. Stisknutím tlačítka `Add random rule` přidáme náhodně vygenerované pravidlo.

5. Stisknutím tlačítka `Add rule` přidáme další, zatím prázdné pravidlo. Při kliknutí na něj vybereme z rychlé nabídky funkci `Lowercase`. Jelikož je funkce bez operandů, tak je ihned vložena.

6. V této rychlé nabídce funkcí klikneme na tlačítko `Add other function`. Zobrazí se nabídka všech podporovaných funkcí, ze které vybereme funkci `Delete @ N`. Jelikož má funkce 1 operand, je nutné ho specifikovat. Tímto operandem se specifikuje pozice, na které má být odstraněn znak. Pro ukázku kontroly operandu zadáme písmeno '`e`'. Při pokusu o vložení funkce tlačítkem `Insert` se objeví chybová hláška s upozorněním na špatný datový typ operandu (viz obrázek 6.1). Provedeme tedy opravu, konkrétně jako pozici uvedeme číslici '`2`'. Poté je funkce po stisknutí tlačítka `Insert` úspěšně vložena do příslušného pravidla.



Obrázek 6.1: Vyskakovací okno pro vložení funkce s upozorněním na nevalidní operand

7. Stiskneme tlačítko `Append rule file`. Po zobrazení vyskakovacího okna vybereme možnost `From filesystem`, kde je nutné vybrat soubor ze souborového systému. Tento soubor s příponou `.rule` nebo `.txt` je stisknutím tlačítka `Append` připojen za již napsaná pravidla.

8. Uložíme soubor stisknutím tlačítka `Save file`. Dojde k zobrazení chybové hlášky, viz obrázek 6.2, jelikož soubor se zadaným názvem již existuje. Zadáme tedy nový název `new.rule`, který je tentokrát jedinečný. Opětovné uložení je již úspěšné, dojde také k zobrazení hlášky o úspěšném uložení a přesměrování zpět na *localhost/rules*.

Zhodnocení experimentu

Podle předpokladu je při zadání již existujícího názvu sady pravidel uživatel upozorněn, že soubor s konkrétním názvem již existuje, a uložení souboru není umožněno. Po zadání

Obrázek 6.2: Chybová hláška upozorňující na existenci souboru se stejným názvem

jedinečného názvu se aplikace také chová podle očekávání, soubor je vytvořen, může být zobrazen ve výpisu všech sad pravidel a následně může být upravován. Experiment také potvrdil očekávané chování při zadání nesprávného typu operandu při vložení funkce. Výsledkem experimentu je vytvoření validní sady pravidel, která mohou dále sloužit například jako obohacení slovníkového útoku.

6.2 Úprava existující sady pravidel

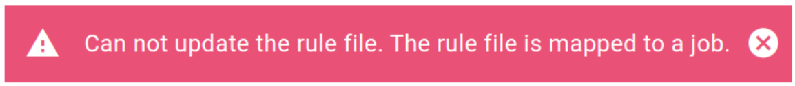
Cílem tohoto experimentu je ukázat editaci již existující sady pravidel a poté její aktualizaci. Dále také ukázat funkčnost kontroly, zda je sada pravidel přidělena k nějaké lámací úloze (*Job*). Experiment také ukazuje živou kontrolu validity pravidel při jejich vytváření.

Postup experimentu

1. Přesuneme se na webovou adresu *localhost/rules*. Ve výpisu najdeme soubor s názvem `toggles1.rule` a v příslušném sloupci `Actions` klikneme na oranžovou ikonu s popisem `Edit file`. Tím dojte k přesunutí do nástroje.

2. Stiskneme tlačítko `Reset rules`. Sada pravidel je následně prázdná.

3. Klikneme na tlačítko `Update file`. Dojde k zobrazení chybové hlášky (viz obrázek 6.3), jelikož tato sada pravidel je již přidělena k lámací úloze, a aktualizování sady pravidel tedy není umožněno. Přesuneme se zpět na stránku s výpisem všech sad pravidel a tentokrát zvolíme soubor k editaci `leetspeak.rule`, který není mapovaný k žádné úloze.



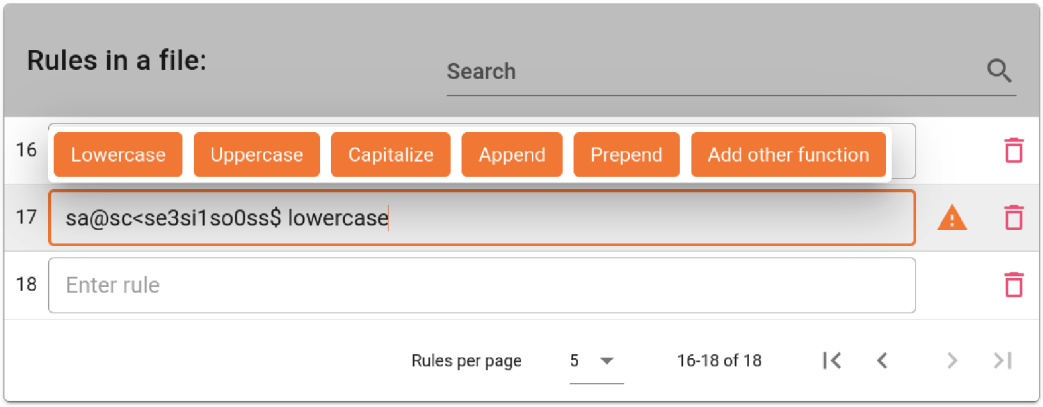
Obrázek 6.3: Chybová hláška upozorňující na přidělení sady pravidel k lámací úloze

4. Do okna `Rule file name` zadáme název souboru `updating.doc`.

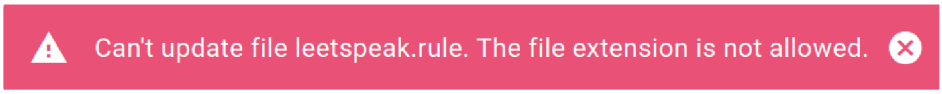
5. Upravíme již existující pravidlo. Pravidlu číslo 17 ručně (skrze klávesnici) dopíšeme řetězec `"lowercase"`. Uživatel je následně skrze vykřičník u pravidla upozorněn, že pravidlo není validní, viz průběh experimentu na obrázku 6.4. Po odstranění tohoto řetězce vykřičník zmizí, pravidlo je opět validní.

6. K současným pravidlům připojíme existující sadu pravidel stisknutím tlačítka `Append rules`. Po zobrazení vyskakovacího okna vybereme možnost `From server`, kde zvolíme sadu pravidel `prince_optimized.rule`. Stisknutím tlačítka `Append` dojde k připojení sady za již specifikovaná pravidla.

7. Uložíme soubor stisknutím tlačítka `Save file`. Dojde k zobrazení chybové hlášky (viz obrázek 6.5), jelikož soubor obsahuje nepovolenou koncovku. Zadáme tedy nový název `updating.rule`, který již kritérium splňuje. Opětovná aktualizace souboru je úspěšná, dojde také k zobrazení hlášky o úspěšné aktualizaci a přesměrování zpět na *localhost/rules*.



Obrázek 6.4: Zobrazení upozornění na nevalidní pravidlo



Obrázek 6.5: Chybová hláška upozorňující na nepovolenou koncovku souboru

Zhodnocení experimentu

Podle předpokladu je uživateli znemožněno aktualizovat soubor, který je již namapován k některé z lámacích úloh. Dále je podle předpokladu uživateli znemožněna aktualizace souboru při zadání názvu se špatnou koncovkou. Po zadání správné koncovky a jedinečného názvu se aplikace chová podle očekávání, soubor je aktualizován včetně jeho názvu a následně může být zobrazen ve výpisu všech sad pravidel, včetně aktualizovaného počtu pravidel. Také byla potvrzena funkčnost živé kontroly validity pravidel.

6.3 Vytvoření nového slovníku pomocí aplikátoru pravidel

Cílem tohoto experimentu je ukázat živé aplikování pravidel na specifikovaná hesla, možné stažení souboru hesly po modifikaci a také ukázat změnu maximálního možného počtu hesel po modifikaci.

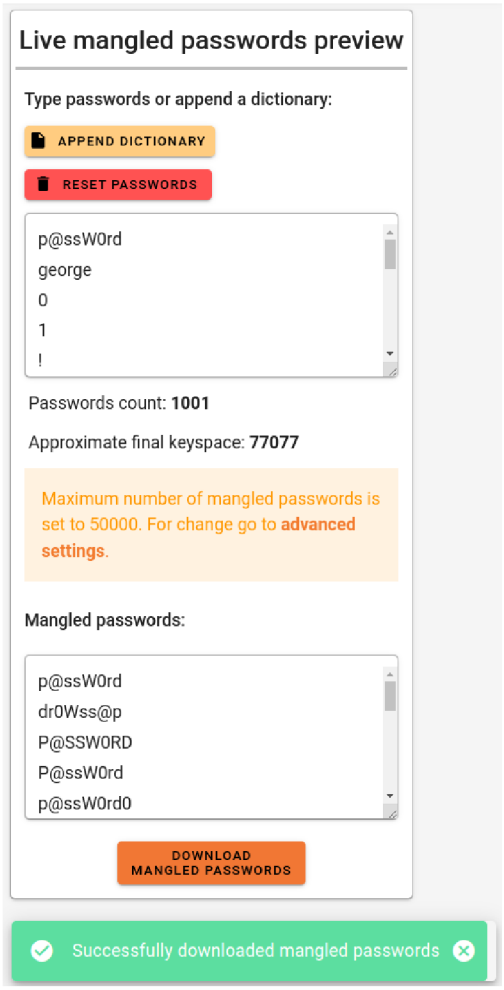
Postup experimentu

1. Přesuneme se na webovou adresu *localhost/rules*. Ve výpisu najdeme soubor s názvem `best64.rule` a v němu příslušnému sloupci `Actions` klineme na oranžovou ikonu s popisem `Edit file`.

2. V okně `Live mangled passwords preview` ručně na nový řádek připíšeme heslo `"george"`.

3. K současným heslům připojíme existující slovník stisknutím tlačítka `Append dictionary`. Po zobrazení vyskakovacího okna vybereme možnost `From server`, kde zvolíme slovník `darkweb2017-top1000.txt`. Stisknutím tlačítka `Append` dojde k připojení slovníku za již specifikovaná hesla.

4. Pro stažení souboru s hesly po modifikaci klikneme na tlačítko `Download mangled passwords`. Objeví se potvrzující hláška, viz průběh experimentu na obrázku 6.6. Je stažen soubor s názvem `mangledPasswords.txt`



Obrázek 6.6: Průběh experimentu 6.3 a úspěšné stažení souboru s hesly po modifikaci

5. Při kontrole tohoto souboru je zjištěno, že nebyla modifikována všechna hesla. Je to díky tomu, že výsledný počet hesel po modifikaci je odhadován na 77077, zatímco maximální počet hesel po modifikaci je stanoven na hranici 50000, viz oranžový rámeček na obrázku 6.6. Pro změnu této hranice se přesuneme do pokročilého nastavení, nebo klikneme na odkaz v oranžovém rámečku s názvem `Advanced settings`.

6. V sekci `Rules editor settings` v okně `Maximum number of mangled passwords` zadáme novou hodnotu `"100000"`. Hodnota je ihned uložena.

7. Přesuneme se do editoru stisknutím tlačítka zpět. Lze vidět, že v oranžovém rámečku byla aktualizována maximální hodnota. Pro kontrolu opět stáhneme soubor s hesly po modifikaci. Nyní již soubor obsahuje všechna modifikovaná hesla.

8. Stiskneme tlačítko `Reset passwords`. Poté je náhled hesel po modifikaci prázdný.

Zhodnocení experimentu

Podle předpokladu došlo k úspěšnému použití aplikátoru pravidel na sadu hesel. Při prvním stažení souboru s hesly po modifikaci byla potvrzena správná funkčnost stanovení hranice jejich maximálního počtu. Při změně této hranice v pokročilém nastavení již soubor dle očekávání obsahoval všechna modifikovaná hesla.

6.4 Analýza doby běhu aplikátoru pravidel

Hlavním cílem tohoto experimentu je ukázat dobu běhu aplikátoru pravidel v závislosti na vygenerovaném počtu hesel. Dalším cílem je zjistit, jaký vliv má počet funkcí v pravidlech na době generování při zachování stejného počtu hesel. Je očekáváno, že čas běhu aplikátoru pravidel roste přímo úměrně s počtem generovaných hesel. Dalším očekáváním je rychlejší běh aplikátoru pravidel při použití pravidel s nižším počtem funkcí.

Postup experimentu

Pro experiment byly vytvořeny 2 sady pravidel. První sada obsahuje 25 pravidel, každé s jednou funkcí. Druhá sada obsahuje také 25 pravidel, ale každé pravidlo v ní má 6 funkcí. Pro měření byly postupně k oběma sadám pravidel přiděleny slovníky. Ty měly pro každé měření jiný počet hesel, aby se měnil celkový počet vygenerovaných hesel. Pro zachování konzistence měla hesla vždy délku 10 znaků.

Každé měření probíhalo tím způsobem, že byla pozorována doba trvání od odeslání požadavku pro vygenerování hesel po obdržení jeho odpovědi v prohlížeči. Naměřené hodnoty lze vidět v tabulce 6.1.

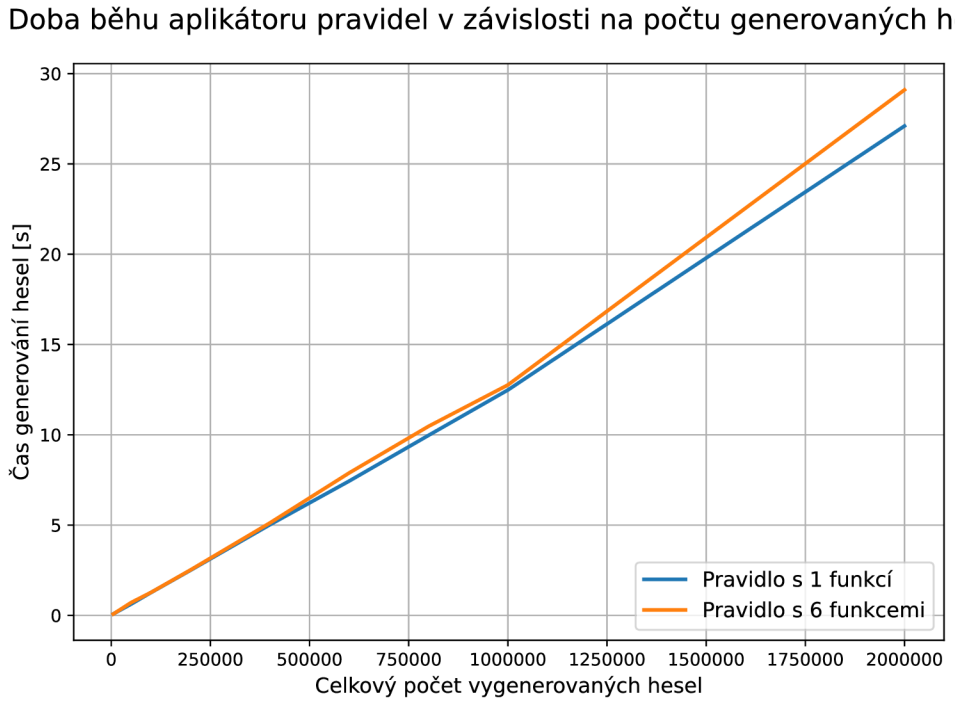
Počet vygenerovaných hesel	Doba generování pro pravidla s 1 funkcí	Doba generování pro pravidla s 6 funkcemi
5000	0,079 s	0,075 s
50000	0,624 s	0,709 s
100000	1,26 s	1,27 s
200000	2,5 s	2,52 s
400000	5,02 s	5,12 s
800000	9,97 s	10,47 s
1000000	12,48 s	12,76 s
2000000	27,1 s	29,1 s

Tabulka 6.1: Doba běhu aplikátoru pravidel v závislosti na počtu generovaných hesel

Zhodnocení experimentu

Při analýze hodnot z tabulky 6.1 a grafu 6.7, který byl pro porovnání z těchto hodnot vytvořen, lze vidět, že doba generování se pro různý počet funkcí v pravidlech téměř neliší. Rozdíl je patrný až při velmi vysokém počtu generovaných hesel. Při bližší analýze zaslaného požadavku však bylo zjištěno, že rozdíl je zapříčiněn větším objemem dat přenášených při přijímání odpovědi. Přijímaná data u sady pravidel s 6 funkcemi měla totiž větší velikost z důvodu vyšší délky vygenerovaných hesel.

Naměřené hodnoty potvrdily, že čas doby běhu aplikátoru pravidel je přímo úměrný počtu generovaných hesel. Naopak nebylo potvrzeno očekávání, že doba generování hesel závisí na počtu funkcí v pravidlech.



Obrázek 6.7: Doba běhu aplikátoru pravidel v závislosti na počtu generovaných hesel

Kapitola 7

Uživatelské testování

Pro ověření uživatelské přívětivosti nástroje, zhodnocení jeho vzhledu a získání námětů pro vylepšení do budoucna bylo uspořádáno uživatelské testování. Celkem se testování zúčastnilo 10 uživatelů. Při výběru uživatelů k testování byl kladen důraz na jejich různorodost. Cílem bylo získat jak uživatele se zkušenostmi se systémem Fitcrack, tak i uživatele, kteří systém nikdy nepoužívali a nemají předchozí zkušenost s tvorbou pravidel.

7.1 Příprava a proces testování

Nejprve bylo potřeba připravit testovací prostředí, tedy zpřístupnit systém Fitcrack uživatelům. Pro jeho zpřístupnění byl využit server `live2.fitcrack.cz`. Na tento server bylo nutné nahrát všechny změny ve zdrojových souborech, následně sestravit Docker obraz a nakonec ho spustit. Díky tomu mohli uživatelé přistoupit do systému Fitcrack a v něm otevřít vytvořený nástroj.

V další fázi byl připraven dotazník v prostředí Google Forms, jeho přepsanou formu lze vidět v příloze B. Dotazník byl anonymní a obsahoval 15 otázek. V úvodu dotazníku byl uživatel nejprve seznámen s tématem mé bakalářské práce, dále byl stručně vysvětlen význam pravidel. Také byl odkázán na manuál k použití a soubor s úkoly, které bylo vhodné pro testování nástroje splnit. Úkoly byly vytvořeny z toho důvodu, aby byl uživatel donucen prozkoumat všechny vlastnosti vytvořeného nástroje. Byly formulovány tak, aby uživatel nedostal přesné instrukce, ale aby si musel některé věci domyslet a tím byla lépe ověřena intuitivnost nástroje. Celkem měl uživatel splnit 3 úkoly.

Úkol č.1 – Vytvoření nové sady pravidel

Cílem úkolu bylo seznámit uživatele s funkcemi pro tvorbu pravidel a ověřit, zda uživatel dokáže vytvořit sadu pravidel. Úkol se skládal se z následujících podčástí:

1. Přesuňte se na zobrazení všech souborů s pravidly (`http://live2.fitcrack.cz/rules`).
2. Vytvořte nový soubor s pravidly.
3. Pojmenujte ho validním názvem.
4. Seznamte se s funkcemi pro práci s pravidly (pomocí příslušného tlačítka).
5. Při vytváření pravidel pozorujte i živé generování modifikovaných hesel.

6. Prázdnému pravidlu přidejte pomocí interaktivní nabídky funkci `Capitalize` a dále funkci `Replace`, která nahradí znak ‘s’ znakem ‘5’.
7. Ručně (pomocí psaní na klávesnici) vytvořte nevalidní pravidlo (např. $12) a tím ověřte funkci živé validity pravidel.
8. K současným pravidlům připojte soubor s pravidly `best64.rule` ze serveru.
9. Přidejte náhodné pravidlo (*Random rule*).
10. Tabulka s pravidly není zobrazena celá, listujte v ní pomocí tlačítek a odstraňte některá pravidla.
11. Soubor uložte a zkontrolujte jeho existenci v zobrazení všech souborů s pravidly.

Úkol č.2 – Úprava existující sady pravidel

Cílem úkolu bylo zhodnocení druhého módu zobrazení (světlý, tmavý) a ověření, zda uživatel dokáže upravit sadu pravidel. Úkol měl následující podčásti:

1. V nastavení (`http://live2.fitcrack.cz/settings`) přepněte režim zobrazení na *Dark mode* (nebo *Light mode*).
2. Přesuňte se k zobrazení všech souborů s pravidly a pomocí tlačítka zahajte úpravu souboru, který jste vytvořili v úkolu 1.
3. Tento soubor libovolně přejmenujte a odeberte všechna jeho pravidla.
4. Dle uvážení vytvořte jakýmkoliv způsobem nějaká pravidla.
5. Soubor uložte a zkontrolujte ho (i s novým názvem) v zobrazení všech souborů s pravidly.

Úkol č.3 – Vytvoření nového slovníku pomocí aplikátoru pravidel

Cílem úkolu bylo seznámit uživatele s živou modifikací hesel a také možností uložení vytvořeného slovníku. Úkol se skládal se z následujících podčástí:

1. Zahajte úpravu souboru, který jste vytvořili v úkolu 2.
2. V okně pro živý náhled hesel po modifikaci přidejte libovolná hesla. Také je možné k heslům připojit již existující slovník.
3. Prohlédněte si hesla po modifikaci v příslušném okně a uložte je na svůj počítač. Můžete zkontrolovat obsah staženého souboru.

7.2 Vyhodnocení

Zde jsou prezentovány výsledky a závěry získané během testování. Cílem dotazníku bylo potvrdit uživatelskou přívětivost nástroje, zhodnotit jeho vzhled a získat náměty pro vylepšení v budoucnu.

Ačkoliv 40 % respondentů před testováním nepoužívalo systém Fitcrack a pouze 20 % z nich mělo zkušenost s tvorbou pravidel, tak nebyla nalezena spojitost těchto zkušeností s celkovým hodnocením nástroje. Přibližně polovina uživatelů si také neprošla manuál a ani v tomto případě nebyla nalezena závislost na celkovém hodnocení. Z toho lze usoudit, že nástroj je jako celek dostatečně intuitivní. Všichni respondenti také konstatovali, že se jedná o užitečné rozšíření do systému Fitcrack.

Otázky v dotazníku, na které bylo odpovídáno škálou 1–10 bodů (kde 1 znamená zápornou odpověď a 10 kladnou), byly zprůměrovány a v kombinaci s otevřenými otázkami z nich bylo zjištěno následující:

- **Intuitivnost tvorby pravidel pomocí interaktivní nabídky** – `8,8b`. To potvrzuje intuitivnost vytváření pravidel. Uživatelé také uváděli tuto vlastnost nástroje jako užitečnou a 90 % uživatelů ji upřednostnilo před ručním zadáváním funkcí pomocí klávesnice. Zejména se jim líbilo, že není nutné znát přesnou syntaxi a sémantiku pravidel.
- **Intuitivnost přidávání hesel a jejich náhled po modifikaci** – `9,1b`. Lze vyvodit, že uživatelé tuto vlastnost nástroje velmi ocenili a práce s ní byla příjemná.
- **Hodnocení rozložení prvků nástroje** – `8,2b`. Ačkoliv se jedná o vysoké hodnocení, uživatelé měli k tomuto aspektu nejvíce připomínek. Bylo navrženo přeuspořádání tlačítek pro tvorbu pravidel do hlavičky tabulky a také bylo upozorněno na neintuitivnost změny velikosti okna při zobrazování všech funkcí.
- **Dojem z vizuální stránky při použití světlého módu** – `8,8b`. Byl potvrzen příjemný návrh grafického uživatelského rozhraní.
- **Dojem z vizuální stránky při použití tmavého módu** – `9b`. Podobně jako u předchozího bodu lze konstatovat, že design GUI byl velmi přívětivý.
- **Responzivita** – `9,2 b`. Výsledek odpovídá dobře navržené responzivitě nástroje, někteří uživatelé nástroj použili i na mobilním telefonu.

Z dalších otevřených otázek, ve kterých mohli uživatelé specifikovat svoje výtky a nápady na rozšíření, byl sestaven **návrh vylepšení nástroje do budoucna**:

1. Přidání popisu k upozornění na nevalidní pravidlo, některým uživatelům nebyl význam oranžového trojúhelníku na první pohled jasný.
2. Automatické přidání koncovky při uložení sady pravidel, pokud není koncovka specifikována.
3. Možnost nastavení preferovaných funkcí v interaktivním menu.
4. Přidat potvrzující popup pro odebrání všech pravidel nebo hesel.
5. Možnost uložení hesel po modifikaci nejen na uživatelovo zařízení, ale také na server.
6. Možnost zobrazení bližšího popisu vytvořených pravidel, jakým způsobem hesla modifikují.

Kapitola 8

Závěr

Cílem práce bylo vytvořit interaktivní nástroj na tvorbu pravidel pro modifikaci hesel a integrovat ho do systému Fitcrack. To se úspěšně podařilo a výsledkem práce je uživatelsky přívětivý nástroj, pomocí kterého lze vytvářet a editovat sady pravidel.

Díky integraci do systému Fitcrack je možné přímo využít vytvořené sady pravidel a obohatit tak různé typy útoků. Velkou předností nástroje je jeho intuitivní rozhraní, které umožňuje tvorbu pravidel i uživatelům, kteří neznají syntaxi a sémantiku funkcí nástroje Hashcat. Užitečným přínosem práce je živá validace pravidel při jejich vytváření, jiné nástroje tuto vlastnost nepodporují. Dalším cílem bylo zprostředkovat živý náhled modifikace hesel pravidly. To se také úspěšně podařilo a nad rámec zadání je uživateli umožněno z hesel po modifikaci vytvořit nový slovník a uložit ho do svého zařízení.

V první řadě bylo nutné se seznámit s problematikou lámání hesel. Zejména s typy útoků, které využívají pravidel, a také se syntaxí a sémantikou těchto pravidel. Dále proběhlo seznámení s nástroji pro lámání hesel, zejména se systémem Fitcrack. Pro účel návrhu nástroje bylo důležité nastudovat současnou práci tohoto systému s pravidly. V další fázi byl proveden samotný návrh a implementace nástroje. Povedlo se navrhnout grafické uživatelské rozhraní, které je intuitivní a responzivní. Další důležitou částí návrhu byl aplikátor pravidel, jehož rychlost je sledována v experimentu 6.4. Bylo zjištěno, že doba jeho běhu závisí přímo úměrně na počtu generovaných hesel a téměř nezávisí na počtu funkcí v jednotlivých pravidlech. V dalších experimentech jsou předvedeny případy užití nástroje. Nakonec bylo provedeno uživatelské testování. Díky němu byla potvrzena intuitivnost nástroje a celkově byl nástroj hodnocen velmi pozitivně. Byla také získána zpětná vazba pro vylepšení nástroje.

V budoucnu plánuji spolupracovat s týmem systému Fitcrack na zakomponování nástroje do oficiálního vydání. Náměty pro vylepšení nástroje byly získány prostřednictvím uživatelského testování. Největší přínos vidím v přidání detailního popisu vytvořených pravidel, aby uživatel nemusel zpětně ručně zjišťovat jejich význam.

Literatura

[1] Das, A., Bonneau, J., Caesar, M. C., Borisov, N. a Wang, X. The Tangled Web of Password Reuse. In: *Network and Distributed System Security Symposium*. 2014. ISBN 1-891562-35-5.

[2] Filipova, O. *Learning Vue. js 2*. Packt Publishing Ltd, 2016. ISBN 9781786469946. Dostupné z: https://books.google.cz/books?&id=nszcDgAAQBAJ&oi.

[3] Hranický, R. *Digital Forensics: The Acceleration of Password Cracking*. Brno, CZ, 2022. Ph.D. thesis. Brno University of Technology, Faculty of Information Technology. Dostupné z: https://www.fit.vut.cz/study/phd-thesis/890/.

[4] Hranický, R., Zobal, L., Večeřa, V. a Múčka, M. *Distribuce výpočtů pro nástroj hashcat*. 2018. 29 s. Dostupné z: https://www.fit.vut.cz/research/publication/11884.

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

[6] Li, S., Wang, Z., Zhang, R., Wu, C. a Luo, H. Mangling Rules Generation with Density-based Clustering for Password Guessing. *IEEE Transactions on Dependable and Secure Computing*. 2022, s. 1–13. DOI: 10.1109/TDSC.2022.3217002.

[7] Ma, J., Yang, W., Luo, M. a Li, N. A Study of Probabilistic Password Models. In: *Proceedings of the 35th IEEE Symposium on Security and Privacy*. May 2014, s. 689–704. DOI: 10.1109/SP.2014.50. ISSN 1081-6011.

[8] Marechal, S. Advances in password cracking. *Journal in computer virology and hacking techniques*. 2008, sv. 4, č. 1, s. 73–81. ISSN 2263-8733. Springer.

[9] Myers, J. a Copeland, R. *Essential SQLAlchemy: Mapping Python to Databases*. O'Reilly Media, 2015. ISBN 9781491916568. Dostupné z: https://books.google.cz/books?id=5p0bCwAAQBAJ.

[10] Peslyak, A. *John the Ripper's cracking modes*. Openwall. [Online; Cit. 2022-12-20]. Dostupné z: https://www.openwall.com/john/doc/MODES.shtml.

[11] Picolet, J. *Hash Crack: Password Cracking Manual*. 2. vyd. Netmux LLC, 2017. ISBN 978-1975924584.

[12] Proctor, R. W., Lien, M.-C., Vu, K.-P. L., Schultz, E. E. a Salvendy, G. Improving computer security for authentication of users: Influence of proactive

password restrictions. *Behavior Research Methods, Instruments, & Computers*. Springer. 2002, sv. 34, č. 2, s. 163–169.

[13] Python restx. *Swagger documentation*. 2020. [Online; Cit. 2023-04-21]. Dostupné z: https://flask-restx.readthedocs.io/en/latest/swagger.html.

[14] Singh, M., Verma, A., Parasher, A., Chauhan, N. a Budhiraja, G. Implementation of database using python flask framework. *International Journal of Engineering and Computer Science*. 2019, sv. 8, č. 12, s. 24890–24893.

[15] Steube, J. *Hashcat Wiki: Description*. Hashcat Wiki. [Online; Cit. 2022-11-21]. Dostupné z: https://hashcat.net/wiki/doku.php?id=hashcat.

[16] Steube, J. *Rule based attack*. Hashcat Wiki. [Online; Cit. 2022-12-08]. Dostupné z: https://hashcat.net/wiki/doku.php?id=rule_based_attack.

[17] Ur, B., Segreti, S. M., Bauer, L., Christin, N., Cranor, L. F. et al. Measuring Real-World Accuracies and Biases in Modeling Password Guessability. In: *Proceedings of the 24th USENIX Conference on Security Symposium*. USA: USENIX Association, 2015, s. 463–481. SEC'15. ISBN 9781931971232.

[18] Van Rossum, G. a Drake, F. L. *Ctypes — A foreign function library for Python*. Python Software Foundation. [Online; Cit. 2023-04-21]. Dostupné z: https://docs.python.org/3/library/ctypes.html.

[19] Weir, M., Aggarwal, S., Medeiros, B. d. a Glodek, B. Password Cracking Using Probabilistic Context-Free Grammars. In: *Proceedings of the 30th IEEE Symposium on Security and Privacy*. Oakland, California, USA: [b.n.], May 2009, s. 391–405. DOI: 10.1109/SP.2009.8. ISBN 978-0-7695-3633-0.

Příloha A

Obsah přiloženého paměťového média

Přiložená SD karta obsahuje:

- tento dokument ve formátu PDF,
- zdrojové soubory této zprávy,
- kompletní zdrojové kódy systému Fitcrack,
- sestavený docker obraz,
- manuál k použití ve formátu PDF.

Součástí manuálu k použití je také návod na instalaci produktu. Výpis souborů, které jsem vytvořil nebo upravoval, se nachází v souboru `fitcrack/upravene_soubory.txt`.

Příloha B

Dotazník pro uživatelské testování

Dotazník k BP - Interaktivní nástroj na tvorbu pravidel pro modifikaci hesel

Dobrý den, jmenuji se Jiří Mládek a rád bych Vám položil několik otázek týkajících se používání nástroje vytvořeného v rámci mojí bakalářské práce. Cílem mé práce bylo vytvořit interaktivní nástroj na tvorbu pravidel pro modifikaci hesel, který je integrovaný do nástroje Fitcrack. Kromě vytváření pravidel slouží nástroj také k živému náhledu hesel po modifikaci a možnosti stažení slovníku s modifikovanými hesly.

* Pravidla pro modifikaci hesel (password-magling rules) slouží nejčastěji jako obohacení slovníkového útoku. **Pravidlo se skládá z jednotlivých funkcí**, např. přidání znaku za heslo, zvětšení prvního písmena hesla atd. Aplikováním pravidel na hesla poté mohou vznikat nové slovníky.

Pro seznámení se s nástrojem je možné si prohlédnout jeho manuál k použití (odkaz). Při testování nástroje Vás prosím o splnění 3 úkolů, díky kterým prozkoumáte všechny vlastnosti nástroje a ověříte, zda je nástroj intuitivní. Soubor s úkoly je dostupný na této adrese (odkaz). Zde jsou informace pro přihlášení do nástroje Fitcrack:

- URL: http://live2.fitcrack.cz/login
- Username: ********
- Password: ********

Děkuji Vám za obětovaný čas a vyplněné odpovědi, díky kterým zpracuji výsledky mojí práce a získám užitečnou zpětnou vazbu pro vylepšení nástroje do budoucna.

Otázky:

Používali jste již v minulosti nástroj Fitcrack? ○ ano ○ ne

Máte již zkušenost s vytvářením pravidel pro modifikaci hesel? ○ ano ○ ne

Byla pro Vás práce s pravidly (vytváření, úprava) pomocí interaktivních oken intuitivní?

neintuitivní ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 velmi intuitivní

Případně co pro vás bylo neintuitivní nebo nelogické? _____

_____

Ocenili jste také možnost psát pravidla ručně (na klávesnici)?

○ Ano, ale raději jsem využíval interaktivního vkládání funkcí ○ Ano, bylo mi to i více příjemné ○ Ne

Bylo pro vás přidávání hesel a jejich náhled po modifikaci intuitivní?

neintuitivní ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 velmi intuitivní

Jak byste celkově hodnotili rozložení prvků (okna, tlačítka, vstupní pole) nástroje?

velmi špatné ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 velmi dobré

Jaký byl Váš celkový dojem z vizuální stránky nástroje při použití light mode?

velmi špatný ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 velmi dobrý

Jaký byl Váš celkový dojem z vizuální stránky nástroje při použití dark mode?

velmi špatný ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 velmi dobrý

Jak byste hodnotili responzivitu nástroje?

velmi špatná ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10 velmi dobrá

Prošli jste si před použitím nástroje manuál? ○ ano ○ ne

Myslíte, že se jedná o užitečné rozšíření pro nástroj Fitcrack? ○ ano ○ ne

Která vlastnost nástroje Vám přišla nejvíce užitečná? _____

_____

Existuje něco, co Vám v nástroji vyloženě chybělo nebo k čemu máte výtky? _____

_____

Máte nějaké nápady pro vylepšení nástroje? Pokud ano, jaké? _____

_____

Příloha C

Funkce pro vytváření pravidel použité v nástroji

Použitý název	Funkce	Popis funkce	Příklad	Vstup	Výstup
Nothing	:	Ponechá heslo stejně	:	H3sl0	H3sl0
Lowercase	l	Převede všechny znaky na malá písmena	l	H3sl0	h3sl0
Uppercase	u	Převede všechny znaky na velká písmena	u	H3sl0	H3SL0
Capitalize	c	První znak převeden na velké písmeno, zbytek na malá písmena	c	H3sl0	H3sl0
Invert Capitalize	C	První znak převeden na malé písmeno, zbytek na velká písmena	C	H3sl0	h3SL0
Toggle Case	t	Invertuje velikosti písmen všech znaků	t	H3sl0	h3SL0
Toggle @	TN	Invertuje velikost písmena na pozici N	T2	H3sl0	H3Sl0
Reverse	r	Obrátí celé slovo	r	H3sl0	0ls3H
Duplicate	d	Duplikuje celé slovo	d	H3sl0	H3sl0H3sl0
Duplicate N	pN	Připojí na konec řetězce duplikované slovo N-krát	p1	H3sl0	H3sl0H3sl0
Reflect	f	Duplikuje slovo, obrátí a připojí na konec řetězce	f	H3sl0	H3sl00ls3H
Rotate Left	{	Orotuje slovo doleva	{	H3sl0	3sl0H
Rotate Right	}	Orotuje slovo doprava	}	H3sl0	0H3sl
Append Character	$X	Připojí znak na konec řetězce	$1	H3sl0	H3sl01
Prepend Character	^X	Připojí znak na začátek řetězce	^X	H3sl0	XH3sl0
Truncate left	[	Odstraní první znak řetězce	[	H3sl0	3sl0
Truncate right	]	Odstraní poslední znak řetězce	]	H3sl0	H3sl
Delete @ N	DN	Odstraní N-tý znak řetězce	D3	H3sl0	H3s0
Extract range	xNM	Získá M znaků, počínaje pozicí N	x02	H3sl0	H3
Omit range	ONM	Odstraní M znaků, počínaje pozicí N	O02	H3sl0	sl0
Insert @ N	iNX	Vloží znak X na pozici N	i3:	H3sl0	H3s:l0
Overwrite @ N	oNX	Přepíše znak na pozici N znakem X	o3:	H3sl0	H3s:0
Truncate @ N	’N	Zkrátí řetězec od pozice N	’2	H3sl0	H3
Replace	sXY	Nahradí výskyt všech znaků X znakem Y	slP	H3sl0	H3sP0
Purge	@X	Odstraní z řetězce všechny výskyty znaku X	@s	H3sl0	H3l0
Duplicate first N	zN	Duplikuje N-krát první znak řetězce	z3	H3sl0	HHHH3sl0
Duplicate last N	ZN	Duplikuje N-krát poslední znak řetězce	Z3	H3sl0	H3sl0000
Duplicate all	q	Duplikuje každý znak řetězce	q	H3sl0	HH33ssll00

Tabulka C.1: Funkce nástroje Hashcat pro vytváření souborů s pravidly

Použitý název	Funkce	Popis funkce	Příklad	Vstup	Výstup
Extract memory	XNMI	Vloží podřetězec délky M startující na pozici N řetězce uloženého v paměti na pozici I současného řetězce	lMX325	H3sl0	h3sl0l0
Append memory	4	Připojí řetězec uložený v paměti za současný	uMl4	H3sl0	h3sl0H3SL0
Prepend memory	6	Připojí řetězec uložený v paměti na začátek současného	rMr6	H3sl0	0ls3HH3sl0
Memorize	M	Uloží do paměti současný řetězec	lMuX034	H3sl0	H3SLh3s0
Reject less	<N	Odmítá řetězce pokud je jejich délka větší než N	<6	H3sl0	H3sl0
Reject greater	>N	Odmítá řetězce pokud je jejich délka menší než N	>6	H3sl0	
Reject contain	!X	Odmítá řetězce, které obsahují znak X	!z	H3sl0	H3sl0
Reject not contain	/X	Odmítá řetězce, které neobsahují znak X	/z	H3sl0	
Reject equal first	(X	Odmítá řetězce, které nezačínají znakem X	(H	H3sl0	H3sl0
Reject equal last	)X	Odmítá řetězce, které nekončí znakem X	)s	H3sl0	
Reject contains	%NX	Odmítá řetězce, které obsahují znak X méně než N-krát	%3a	H3sl0	
Swap front	k	Prohodí první dva znaky řetězce	k	H3sl0	3Hsl0
Swap back	K	Prohodí poslední dva znaky řetězce	K	H3sl0	H3s0l
Swap @ N	*NM	Prohodí znak na pozici N se znakem na pozici M	*23	H3sl0	H3ls0
Bitwise shift left	LN	Bitový posun znaku na pozici N doleva	L1	H3sl0	Hfsl0
Bitwise shift right	RN	Bitový posun znaku na pozici N doprava	R3	H3sl0	H3s60
Ascii increment	+N	Zvýší ascii hodnotu znaku na pozici N o 1	+1	H3sl0	H4sl0
Ascii decrement	-N	Sníží ascii hodnotu znaku na pozici N o 1	-1	H3sl0	H2sl0
Replace N + 1	.N	Nahradí znak na pozici N hodnotou znaku na pozici N+1	.2	H3sl0	H3ll0
Replace N - 1	,N	Nahradí znak na pozici N hodnotou znaku na pozici N-1	,2	H3sl0	H33l0
Duplicate block front	yN	Duplikuje prvních N znaků řetězce	y2	H3sl0	H3H3sl0
Duplicate block back	YN	Duplikuje posledních N znaků řetězce	Y2	H3sl0	H3sl0l0
Title	E	Převede všechny znaky na malá písmena, poté převede na velká písmena první znak a všechny znaky po mezeře	E	H3sl0 he	H3slo He
Title w/separator	eX	Převede všechny znaky na malá písmena, poté převede na velká písmena první znak a všechny znaky po specifikovaném oddělovači	e;	H3sl0;he	H3sl0;He

Tabulka C.2: Funkce nástroje Hashcat pro vytváření souborů s pravidly