VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ APLIKÁCIA NA VZDIALENÚ SPRÁVU SYSTÉMU FITCRACK

WEB APPLICATION FOR REMOTE ADMINISTRATION OF FITCRACK SYSTEM

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE **MATÚŠ MÚČKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. RADEK HRANICKÝ**
SUPERVISOR

BRNO 2018

Abstrakt

Táto práca rieši vzdialenú komunikáciu so systémom na distribuovanú obnovu hesiel - Fitcrack. Zameral som sa na vylepšenie súčasného riešenia. Okrem zvýšenia bezpečnosti a rýchlejšej odozvy, návrh riešenia uvedený v tejto práci ponúka aj podporu autentifikácie užívateľov a ich oprávnení. Systém bude automaticky generovať dokumentáciu pri zmene zdrojových kódov. V riešení bola použitá architektúra REST (viď 3.2). Táto architektúra umožňuje oddeliť aplikačnú logiku od databázového systému, a tým pádom umožniť komunikáciu s aplikáciami pomocou zasielanie správ. Vďaka tomu je možné systém Fitcrack ovládať z rôznych prostredí pomocou jednoduchých dotazov protokolu HTTP. V mojej práci som navrhol systém, ktorý zrýchli komunikáciu so systémom Fitcrack, zlepší zabezpečenie a pridá do systému vylepšenia ako podpora viacerých užívateľov a ich oprávnení.

Abstract

This bachelor's thesis resolves remote communication with distributed password recovery system - Fitcrack. I aimed to improve a current implemented solution. In addition to security improvement and faster responses, the design of solution presented in this work offers support for user authentication and authorization. The system automatically generate documentation when source code changes. REST (see 3.2) architecture was used in this solution. This architecture allows to separate the application logic and the database system which consequently allows application communication by sending messages. Because of that, it is possible to control Fitcrack from different environments using simple HTTP queries. I designed the system that accelerates communication with Fitcrack, refines security and adds improvements such as support for multiple user privileges.

Kľúčové slová

REST, Fitcrack, API, backend

Keywords

REST, Fitcrack, API, backend

Citácia

MÚČKA, Matúš. *Webová aplikácia na vzdialenú správu systému Fitcrack*. Brno, 2018. Semestrální projekt. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický

Webová aplikácia na vzdialenú správu systému Fitcrack

Prehlásenie

Prehlasujem, že tento semestrálny projekt som vypracoval samostatne pod vedením pána inžiniera Radka Hranického. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

.....
Matúš Múčka
16. apríla 2018

Poďakovanie

Ďakujem Ing. Radkovi Hranickému za odbornú pomoc a vedenie pri vypracovávaní tejto práce.

Obsah

Kapitola 1

Úvod

V súčasnosti je používanie aplikačného protokolu HTTP pre sieťový prenos textových dát v rámci služby World Wide Web (WWW) veľmi rozšírené. Spoločne s elektronickou poštou je HTTP najviac používaným protokolom, ktorý sa zaslúžil o obrovský rozmach internetu v posledných rokoch. Jedným z autorov protokolu HTTP je Roy Fielding, ktorý vo svojej dizertačnej práci opisuje Representation State Transfer (REST). REST je architektúra, ktorá poskytuje obecné rozhranie pre vzdialené aplikácie, ktoré komunikujú cez sieť. Napriek tomu, že REST nie je určený priamo pre HTTP, je takmer vždy spojovaný s týmto protokolom.

V tejto práci popisujem všeobecný opis a implementáciu aplikačné rozhranie spĺňajúce zásady architektúry REST (viď 3.2) pre systém Fitcrack. Jedná sa o výkonný systém na obnovu hesiel, ktorý prerozdeľuje prácu medzi viacero pripojených klientov, a tým pádom zvyšuje výpočetnú silu celého systému (detailnejšie je Fitcrack popísaný v kapitole 2). Konkrétne sa v mojej práci zaoberám jeho serverovou časťou, ktorá slúži na správu celého systému. Mojim cieľom je nahradiť súčasné nevyhovujúce riešenie administrácie systému, ktoré bolo navrhnuté len ako prototyp, novým riešením.

Súčasná implementácia je veľmi ťažko rozšíriteľná. Taktiež obsahuje niekoľko bezpečnostných dier. Nové riešenie prináša do systému Fitcrack väčšiu bezpečnosť a rozšíriteľnosť, umožňuje testovanie systému, a vďaka architektúre klient-server znižuje záťaž na serverovú časť systému.

Táto práca pozostáva z piatich kapitol. Popis systému Fitcrack spolu s nástrojmi, ktoré využíva sa nachádza v kapitole 2. Opis využitých technológií, ktoré som pri návrhu nového systému na vzdialenú správu Fitcracku použil, sa nachádza v kapitole 3. Samotným návrhom riešenia, ktorý je rozdelený do modulov, sa zaoberám v kapitole 4. V jednotlivých podkapitolách popisujem každý modul. Nakoniec v závere hodnotím výsledky mojej práce.

Kapitola 2

Systém Fitcrack

Fitcrack je systém, ktorý slúži na obnovu hesiel. Vďaka tomu že ide o distribuovaný systém, je možné rozdelovať prácu na predom neobmedzený počet staníc, ktoré môžu byť rozmiestnené po celom svete. Je tvorený architektúrou klient-server. Serverová časť sa stará o prerozdeľovanie úloh medzi klientov. Klienti pracujú na výpočte kryptografických hešov a svoje výsledky posielajú na server, kde sa spracujú a vyhodnotia. [4]

2.1 Hashcat

Hashcat[1] je výkonný nástroj na obnovu hesiel, ktorý využíva technológiu OpenCL. Podporuje viac ako 200 typov kryptografických hešov. O jeho rýchlostí svedčí aj to, že v rokoch 2010, 2012, 2014 a 2015, tím, pozostávajúci z členov vývojárov Hashcatu, získal prvé miesto v súťaži *Crack Me If you Can*[2]. Žiaľ, tento nástroj sám o sebe nepodporuje výpočet na viacerých staniciach súčasne.

2.2 BOINC

Systém Fitcrack je postavený na voľne šíriteľnom frameworku *Berkeley Open Infrastructure for Network Computing* (BOINC)[3]. Vďaka frameworku BOINC systém Fitcrack distribuuje dielčie úlohy medzi pripojených klientov, na ktorých prebieha pokus o nájdenia hesla pomocou nástroja Hashcat.

Systém BOINC tvorí server a klienti. Pri distribuovaní výpočetných úloh prebieha komunikácia medzi klientom a serverom prostredníctvom XML správ, ktoré sa prenášajú pomocou protokolu HTTP alebo HTTPS. Server je hlavnou súčasťou infraštruktúry BOINC. Stará sa o distribuovanie úloh medzi klientov a spracováva výsledky od užívateľov. Každý klient sa periodicky dotazuje na server a žiada si novú úlohu. Po prijatí výpočetnej úlohy ju spracuje a výsledok odošle na serverom. Následne žiada o pridelenie novej úlohy.[1]

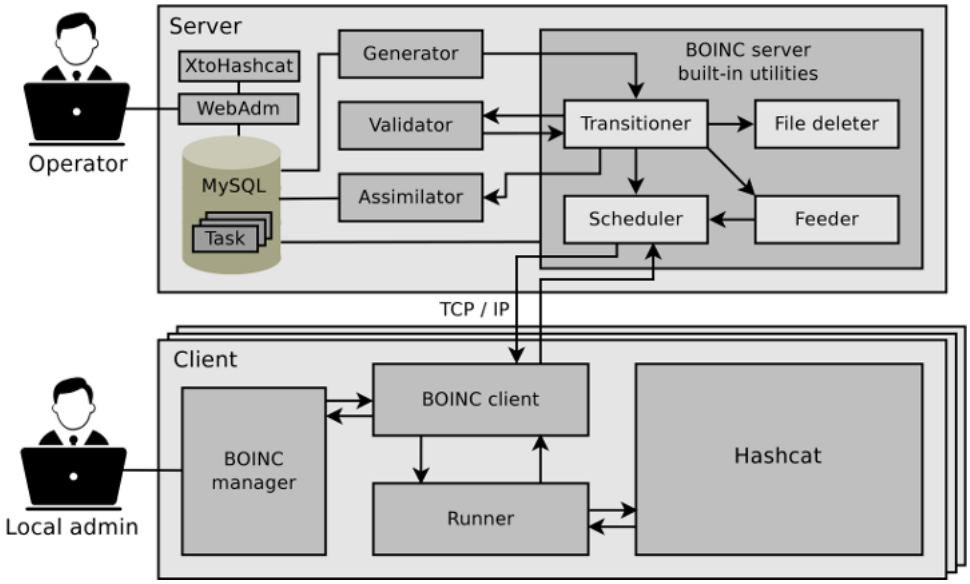
2.3 Architektúra systému Fitcrack

Ako je možné vidieť na obrázku 2.1, architektúra systému Fitcrack je rozdelená do niekoľkých funkčných blokov. Hlavné časti systému sú serverová a klientská.

[1] `https://hashcat.net`
[2] `http://contest-2010.korelogic.com/team_hashcat.html`
[3] `https://boinc.berkeley.edu/`

Serverová časť systému Fitcrack beží na samostatnom stroji. Dohliada na výpočetné uzly (klientov) a má na starosti plánovanie a generovanie úloh, spracovanie výsledkov z výpočetných uzlov a riadenie distribuovaného výpočtu. Server sa na samotnom výpočte kryptografických hešov nepodiela. Serverová časť sa dá ďalej rozdeliť na moduly. Jedným z nich je *Generator*, ktorý sa stará o vytváranie úloh pre výpočetné uzly. Ďalším modulom je *Validator*. Ten slúži na overenie syntaxe odpovedi od výpočetného uzlu. *Assimilator* funguje na princípe nekonečného cyklu, ktorý periodicky kontroluje a spracováva nové výsledky. V prípade nájdenia hesla výsledok priradí do databázy, a všetkým výpočetným uzlom pošle správu aby zanechali prácu. Nástroj *XtoHashcat* slúži na extrakciu šifrovacie kľúča z rôznych typov súborov. Modul *WebAdmin* poskytuje užívateľovi prototyp grafického rozhrania na správu systému. Ostatné moduly serverovej časti sú súčasťou siete BOINC.

Klient (výpočetný uzol), po pripojení žiada od servera novú úlohu, ktorú spracuje, vyrieši[4], a výsledok odošle na server. V súčasnosti sa pridelené úlohy na klientovi riešia pomocou nástroja Hashcat (viď 2), ktorý nie je potrebné inštalovať, pretože sa automaticky po pripojení uzlu odošle klientovi jeho binárna verzia. Po dokončení pridelenej úlohy si klient žiada o ďalšiu.



Obr. 2.1: Architektúra systému Fitcrack [4]

[4]Zvyčajne sa jedná o výpočet kryptografických hešov, ale existujú aj iné typy úloh, ktoré výpočetné uzly systému Fitcrack riešia. Napríklad meranie výkonu.

Kapitola 3

Popis použitých technológií

3.1 Hypertext Transfer Protocol (HTTP)

Pôvodne bol protokol HTTP určený pre výmenu dokumentov vo formáte HTML, ale v súčasnosti sa používa aj pre prenos iných informácií. Vďaka rozšíreniu MIME (Multipurpose Internet Mail Extensions) je tento protokol schopný prenášať akýkoľvek súbor [3].

3.1.1 Princíp protokolu HTTP

Protokol HTTP funguje na princípe dotaz-odpoveď. Užívateľ (klient/user-agent) pošle serveru dotaz. Ten server spracuje a klientovi odpovie. V odpovedi server popisuje výsledok dotazu informáciami, či sa podarilo žiadaný zdroj nájsť, či zdroj existuje, v akom formáte je telo odpovede atď.

3.1.2 Rozšírenie HTTPS

Protokol HTTP neumožňuje zabezpečené spojenie, preto sa často používa protokol TLS (Transport Layer Security) nad vrstvou TCP. Vďaka tomu je možné vytvoriť šifrovaný kanál. Toto spojenie je označované ako HTTPS [5].

3.1.3 Stavové kódy protokolu HTTP

Úspešnosť dotazu vieme zistiť podľa stavových kódov, ktoré sú pribalené v odpovedi servera na dotaz klienta. Vďaka stavovým kódom vieme presne určiť, či behom spracovávania dotazu došlo k chybe. Tým pádom môže klient reagovať na chyby.

Zoznam stavových kódov má na starosti organizácia IANA (Internet Assigned Numbers Authority). Jedná sa o trojciferné číslo v desiatkovej sústave. Prvé číslo určuje kategóriu odpovede a ostatné ju bližšie špecifikujú.

1XX

Kódy, začínajúci číslom 1, sú tzv. informačné. Indikujú že server dotaz spracoval a pochopil. Bližší význam záleží na zvyšných dvoch číslach. V niektorých prípadoch môže klientovi naznačovať, že sa finálna odpoveď ešte spracováva a má na ňu počkať. Tiež môže naznačovať zmenu protokolu.

2XX

Stavové kódy začínajúce číslom 2 indikujú, že dotaz bol serverom obdržaný, pochopený a správne vyhodnotený.

3XX

Tieto kódy naznačujú, že na získanie požadovaného zdroja, je potrebné vykonať ďalšiu akciu. Zvyčajne sa jedná o presmerovanie.

4XX

Stavové kódy, ktoré začínajú číslom 4, indikujú, že nastala chyba na strane užívateľa. Ďalšie 2 čísla presnejšie určujú o akú chybu ide. Najčastejšie sa jedná o chybu `404 Not Found`, ktorá hovorí že žiadaný zdroj nebol na serveri nájdený, ale môže ísť aj o menej časté chyby ako `429 Too Many Requests`, ktorá sa vyskytuje v prípade, že užívateľ žiadal o daný zdroj príliš veľa krát v určitom časovom úseku.

5XX

Stavové kódy začínajúce číslom 5, hovoria o tom, že došlo k chybe na strane servera. Aj keď dotaz mohol byť validný, server ho nedokázal spracovať. Mohlo sa tak stať napríklad kvôli výpadku servera (preťaženie, údržba).

3.1.4 Druhy žiadostí/metódy HTTP

Protokol HTTP využíva niekoľko žiadostí, z ktorých najčastejšie sú:

- **GET** - ide o najbežnejší typ žiadostí. Jej výsledkom je žiadaný zdroj uvedení v dotaze URL. Tento typ dotazu neobsahuje telo správy.
- **POST** - k dotazu je pridané telo správy. Zvyčajne obsahuje hodnoty z HTML formulára.
- **PUT** - využíva sa na nahranie súboru na určitú URI.
- **DELETE** - tento typ žiadosti je len zriedka implementovaní. Zmaže zdroj uvedení v URI.
- **HEAD** - ide o podobný typ žiadosti ako GET, ale na rozdiel od GET, server vracia len hlavičku odpovede.
- **OPTIONS** - v odpovedi vracia metódy, ktoré sú povelené na danej URI.

3.1.5 Príklad komunikácie

Klient začína komunikáciu poslaním dotazu na server. Na výpise 3.1 sa nachádza ukážka dotazu, ktorý obsahuje zvyčajne viac informácií, ale pre zjednodušenie príkladu nie sú uvedené. Server v dotaze špecifikuje metódu žiadosti, svoju totožnosť (Opera verzia 10.60) a podporované kódovanie.

```
GET / HTTP/1.1
Host: www.fit.vutbr.cz
User-Agent: Opera/9.80 (Windows NT 5.1; U; sk) Presto/2.5.29 Version/10.60
Accept-Charset: UTF-8,*
```

Výpis 3.1: príklad HTTP dotazu

Príklad nasledovnej reakcie servera na dotaz sa nachádza na výpise 3.2. Server odpovedá stavovým kódom `200 OK`, čo značí, že dotaz sa podarilo úspešne spracovať. Ďalej hlavička odpovedi okrem iného obsahuje dátum a čas vybavenia žiadosti, a informácie o vrátenom zdroji ako typ (text/HTML), použité kódovanie (UTF-8) a dĺžku odpovede.

```
HTTP/1.1 200 OK
Content-Length: 3059
Server: GWS/2.0
Date: Sat, 11 Jan 2003 02:44:04 GMT
Content-Type: text/html
Cache-control: private
Connection: keep-alive
```

Výpis 3.2: príklad HTTP odpovedi

3.2 Architektúra REST

REST je úzko spojený s protokolom HTTP, keďže ho prvý krát navrhol a popísal Roy Fielding - jeden zo spoluautorov protokolu HTTP, v rámci jeho dizertačnej práce v roku 2000 [2]. Architektúra REST sa používa pre jednotný prístup ku zdrojom. Jedná sa o spôsob, ako klient môže pomocou základných HTTP volaní vytvárať, čítať, editovať alebo mazať informácie zo serveru. Zdrojom môžu byť dáta alebo stav aplikácie, pokiaľ sa dá dátami vyjadriť. K jednotlivým zdrojom sa pristupuje pomocou URI (Uniform Resource Identifier – jednotný identifikátor zdroja). Každý zdroj musí mať vlastný identifikátor URI.

3.2.1 Zásady architektúry REST

Aby služba mohla byť považovaná za RESTful, musí spĺňať šesť formálnych obmedzení. Vďaka nim sú služby vytvorené pomocou REST architektúry výkonné, škálovateľné, jednoduché, ľahko upraviteľné, prenositeľné a spoľahlivé.

Architektúra klient-server

Jednou z najdôležitejších zásad architektúry REST je klient-server architektúra. Vďaka tomu je možné rovnomernejšie rozložiť záťaž. Server negeneruje pre užívateľa grafické rozhranie a môže obslúžiť viac užívateľov. Taktiež rozloženie záťaže umožňuje jednoduchú prenositeľnosť užívateľského rozhrania na viaceré platformy.

Bezstavová architektúra (Statelessness)

Jedná sa o bezstavovú architektúru z pohľadu servera. Medzi dotazmi sa na serveri nemôžu ukladať žiadne informácie o stave klienta. Každá žiadosť od akéhokoľvek klienta musí obsahovať všetky potrebné informácie na vybavenie dotazu. Svoj stav si klient uchováva sám. Trvalý stav klienta môže server uchovávať v databázy.

Možnosť uchovávať zdroje v medzipamäti (Cacheability)

Pre zlepšenie výkonnosti celého systému musí server označiť zdroje ako uložitelné alebo neuložitelné do medzipamäte. Uložitelné zdroje sú také, ktoré sa často nemenia. Keď si klient požiada o takýto zdroj, server mu odpovie okrem dát z daného zdroja aj informáciou, do kedy má uchovať dané dáta v medzipamäti. Keď bude klient v budúcnosti potrebovať prístup k dátam, ktoré má uložené v medzipamäti, a nevypršala exspiračná doba dát, načíta si dáta z medzipamäti. Tým pádom nezaťažuje server a pristúpi k dátam rýchlejšie.

Vrstevnatelnosť (Layered system)

Klient zvyčajne nemôže povedať či je pripojený ku koncovému serveru alebo len k nejakému sprostredkovateľovi. Sprostredkovateľské servery sa používajú na zlepšenie škálovateľnosti systému tým, že umožnia rozložiť záťaž. Môžu tiež uplatňovať bezpečnostné pravidlá (napríklad ochrana proti DDoS útokom).

Zaslanie klientovi spustiteľného kódu (Code on demand)

Jedná sa o voliteľné obmedzenie. Server môže klientovi zaslať spustiteľný kód (zvyčajne v skriptovacom jazyku ako Javascript). Vďaka tomu môže server dočasne rozšíriť alebo prispôsobiť finkčnosť klienta.

Jednotné rozhranie

Základom akejkoľvek služby REST je jednotné rozhranie medzi klientom a serverom. Jedná sa hlavne o typy správ, ktoré môžu byť vo viacerých formátoch (HTML, XML, JSON).

3.2.2 Metódy pre prístup ku zdrojom

Architektúra REST definuje štyri základné metódy pre prístup k jednotlivým zdrojom. Tieto metódy sú implementované pomocou zodpovedajúcich metód HTTP protokolu. Významy metód sa líšia v závislosti od toho, či boli zavolané nad kolekciou, alebo nad určitým prvkom.

- **GET** - ak bol zavolaný na kolekciou, odpoveď obsahuje pole prvkov kolekcie. Tiež môže obsahovať doplňujúce údaje (tzv. metadata), napríklad koľko prvkov je v kolekcii. Pri zavolaní nad konkrétnym záznamom, vráti informácie o zázname.
- **POST** - vytvorí nový záznam v kolekcii. ID záznamu je zvyčajne automaticky vytvorené a vrátené v odpovedi dotazu.
- **PUT** - upraví záznam, alebo ak neexistuje tak záznam vytvorí.
- **DELETE** - zmaže celú kolekciu alebo konkrétny záznam.

Kapitola 4

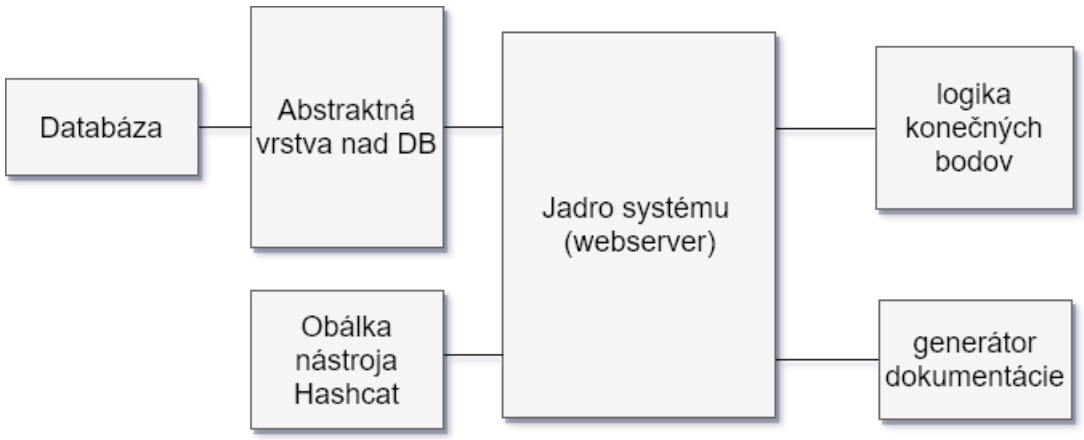
Návrh

4.1 Návrh serverovej časti

V tejto kapitole sa zaoberám návrhom implementácie webového servera pre systém Fitcrack. Ako implementačné prostredie som zvolil Python3. Pri jeho výbere som bral v úvahu už zaužívané technológie v systéme Fitcrack, požadovanú funkčnosť systému a taktiež jeho prenositeľnosť.

4.1.1 Rozdelenie systému na moduly

Pre lepšiu organizáciu som celý systém rozdelil do niekoľko modulov (viď obr. **??**). Jednotlivé moduly sú detailnejšie rozobrané v nasledujúcich podkapitolách.



Obr. 4.1: Rozdelenie systému do modulov

Jadro systému

Najhlavnejším modulom v systéme je jeho jadro, ktoré spĺňa úlohu webového servera a sú v ňom uložené nastavenia celého systému (prístupové údaje do databázy, zložka pre nahrávanie súborov, cesta k nástroju Hashcat atď.). Pri zlyhaní jadra webový server odpovedá HTTP správy s kódom začínajúcim číslom 5 (viď 3.1.3).

Logika konečných bodov

Jedná sa o modul, v ktorom sú obslúžené dotazy na konkrétne URI adresy. Zoznam dostupných URI je uvedený v tabuľke 4.1. Pri chybe v tomto module, webový server odpovedá HTTP správami, ktoré začínajú číslom 4 (viď 3.1.3). Ak užívateľ žiada o neplatnú kombináciu URI adresy a HTTP metódy, je mu vrátená HTTP správa s kódom 404. V prípade že je užívateľ neprihlásený a žiada o zdroj, ktorý vyžaduje autentifikáciu, v odpovedi dostane HTTP správu s kódom 401. Pokiaľ je užívateľ prihlásený, ale na zdroj o ktorý žiada nemá dostačujúce oprávnenia, tento modul vracia HTTP správu s kódom 403. V inakšom prípade je dotaz týmto modulom spracovaný. Ak nedôjde k chybe počas spracovávania dotazu, modul vygeneruje odpoveď vo formáte JSON, ktorú pridá do tela HTTP odpovede s kódom 200 (viď 3.1.3).

Jednotlivé URI adresy		
Názov služby	URI	Metóda
Prihlásenie užívateľa	/user/login	POST
Registrácia užívateľa	/user/register	POST
Odhlásenie užívateľa	/user/logout	GET
Zoznam podporovaných hešov	/hashTypes	GET
Zoznam podporovaných typov útokov	/attackModes	GET
Zobrazenie kolekcie balíkov	/package	GET
Pridať balík	/package	POST
Zmazať balík	/package/{packageID}	DELETE
Upraviť balík	/package{packageID}	UPDATE
Zobrazí podrobné informácie o balíku	/package/{packageID}	GET
Operácia nad balíkom (reštart, štart, stop)	/package/{packageID}/action	POST
Vráti kolekciu úloh, na ktoré je balík rozdelený	/package/{packageID}/job	GET
Zobrazí hosťov, ktorí sú pridelení k balíku	/package/{packageID}/host	GET
Pridá hostí k balíku	/package/{packageID}/host	POST
Odoberie hostí z práce na balíku	/package/{packageID}/host	DELETE
Zobrazí kolekciu hostí	/host	GET
Deaktivuje konkrétneho hosťa	/host/{hostID}	DELETE
Vráti kolekciu slovníkov	/dictionary	GET
Vráti obsah konkrétneho slovníka	/dictionary/{dictionaryID}	GET
Vymaže slovník zo systému	/dictionary/{dictionaryID}	DELETE
Informácie o serveri	/server/info	GET
Operácia so serverom (reštart, štart, stop)	/server/action	POST

Tabuľka 4.1: Zoznam dostupných URI.

Obálka nástroja Hashcat

Aj keď serverová časť systému Fitcrack nepoužíva priamo nástroj Hashcat na generovanie hešov kryptografických funkcií, používa ho na rôzne vedľajšie účely, ako napríklad získavanie podporovaných typov útokov a kryptografických hešov. Z tohto dôvodu je potrebné implementovať obálku na tento nástroj vo forme objektu (triedy) s požadovanými funkciami. To nám uľahčí prístup k nástroju Hashcat a taktiež bezpečnejšie narábanie s ním.

Abstraktná vrstva nad databázou

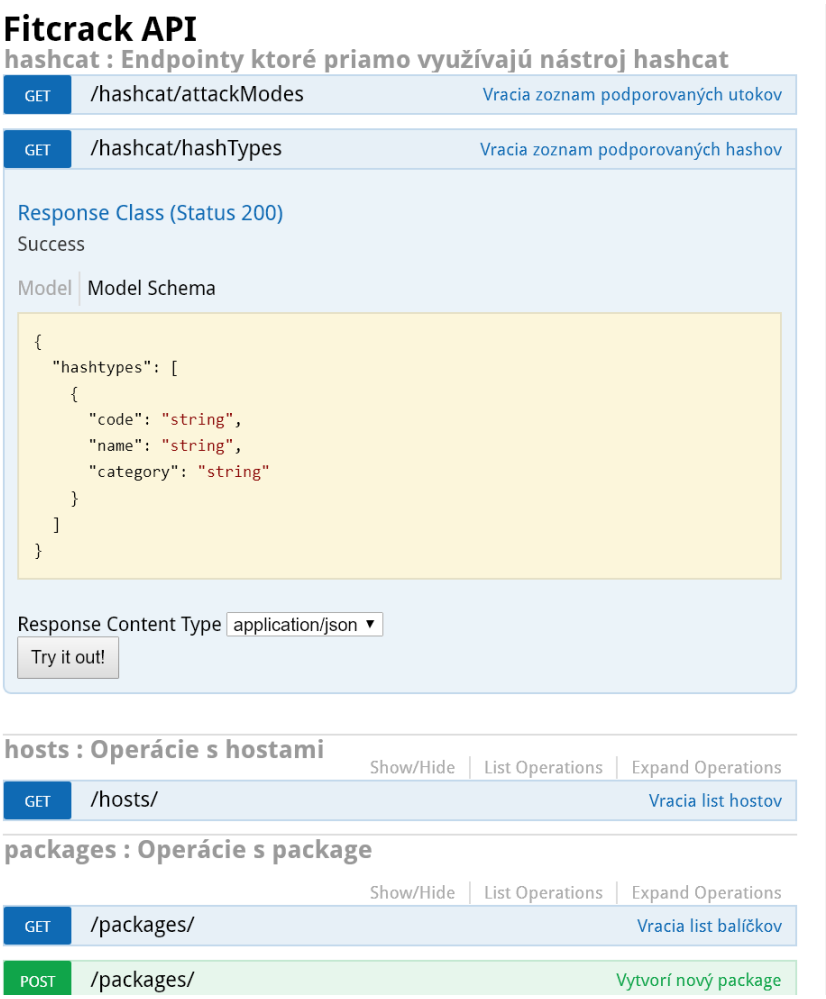
Jedná sa o modul, ktorý výrazne uľahčuje prístup k databáze. Zároveň výrazne neovplyvňuje výkon a flexibilitu systému. Vďaka tomu, že systém s databázou nekomunikuje priamo, ale využíva abstraktnú vrstvu, je databáza lepšie chránená (ochrana pred útokmi typu SQL injection).

Databáza

Vzhľadom nato, že systém Fitcrack využíva databázový server MySQL, rozhodol som sa napojiť na túto databázu. Aby bolo možné implementovať všetky rozšírenia systému, bude potrebné modifikovať štruktúru databázy.

Generátor dokumentácie

Generátor dokumentácie po spustení webového servera spracuje zdrojové kódy aj s komentármi a vytvorí pomocou nich interaktívnu dokumentáciu (viď obr. **??**). Z nej je potom možné zistiť všetky dostupné koncové body spolu s príkladmi odpovede. Generovanie dokumentácie automaticky nastáva aj pri zmene zdrojových súborov.

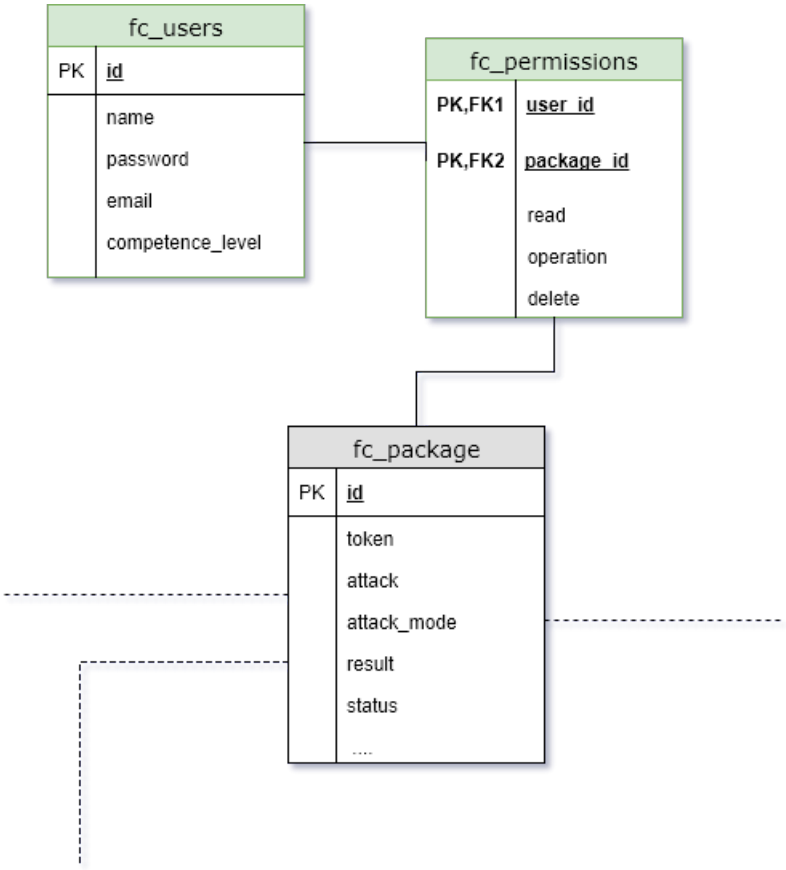


Obr. 4.2: Ukážka vygenerovanej dokumentácie

4.1.2 Autentifikácia a oprávnenia užívateľov

Súčasný systém Fitcrack nepodporuje viacerých užívateľov. Aby bolo možné toto vylepšenie implementovať, je potrebné do databázy pridať tabuľku `fc_users`. Okrem mena užívateľa a zahešovaného hesla táto tabuľka obsahuje užívateľský email a všeobecné právomoci užívateľa na serveri. Jedná sa o číslo, ktoré čím je väčšie, tým väčšie má právomoci užívateľ. Napríklad ak má užívateľ v stĺpci `competence_level` číslo 9, môže pridávať do systému nové užívateľské účty, robiť operácie so serverom (napríklad reštart), a vidieť všetky balíčky ktoré sa v systéme nachádzajú. Oproti nemu, užívateľ so všeobecnými oprávneniami nastavenými na 1, nemôže pridávať nových užívateľov, ani vykonávať operácie so serverom a má prístup len k balíčkom, ktoré sú mu pridelené.

Pre jednoduchšiu implementáciu oprávnení užívateľov k balíkom som pridal ďalšiu tabuľku `fc_permissions` (upravená štruktúra databázy je zobrazená na obrázku **??**). Základné oprávnenia užívateľa k balíku sú tri. Najzákladnejšie oprávnenie dovoľuje užívateľovi vidieť balík medzi kolekciou balíkov, a zároveň zobraziť detailné informácie o balíku. Ďalšie oprávnenie dovoľuje užívateľovi vykonávať operácie s balíkom (štart, stop, reštart a úprava hostov, ktorí sa podieľajú na balíku). Posledné oprávnenie dovoľuje užívateľovi vymazať balík z databázy.



Obr. 4.3: Časť entitno - relačného diagramu databázy systému Fitcrack. Tabuľky označené zelenou farbou sú nové.

4.2 Návrh klientskej časti

Ahoj ako sa máš?

Kapitola 5

Implementácia

Implementácia v prostredí python3

5.1 Štruktúra databázy

Štruktúra databázy je pre implementáciu aplikačného rozhrania kľúčová. Do pôvodnej databázy systému fitcrack som niektoré tabuľky pridal a iné upravil.

/*vymenovat tabulky*/

5.2 Podpora užívateľov

5.3 Autorizácia

5.4 Flask

5.5 Stránkovanie

Kapitola 6

Záver

V tejto práci sa mi podarilo navrhnúť implementáciu serverovej časti systému Fitcrack s automaticky generovanou dokumentáciou. Zameral som sa na vylepšenia súčasnej verzie webového prostredia Fitcracku a navrhol som rozšírenia, ku príkladu systém viacerých užívateľov, ktoré spríjemnia užívateľom používanie. Taktiež som odstránil nedokonalosti súčasného riešenia, ako napríklad dostupnosť len z webovej platformy. Systém, implementovaný podľa môjho návrhu bude dostupný zo všetkých platforiem, ktoré sú schopné sieťovej komunikácie. Súčasne sa zníži záťaž na server, pretože bude prerozdelená medzi klienta a server. Vďaka implementácie pomocou modulov bude celý systém ľahko rozšíriteľný. Návrh opísaný v tejto práci tiež ošetruje niekoľko bezpečnostných dier súčasnej implementácie. Návrh spĺňa všetky zásady architektúry REST (viď 3.2), a taktiež je vhodný na testovacie účely systému.

Keďže som v rámci tejto práce zhromaždil potrebné znalosti, v nasledujúcom semestri budem pracovať na implementácií navrhnutého systému. Tiež sa budem venovať testovaniu spoľahlivosti a experimentom. Systém podrobím výkonnostným testom a výsledky porovnám so súčasným riešením. Mojou snahou je vytvoriť aplikačné rozhranie na vzdialené ovládanie systému Fitcrack takým spôsobom, aby implementácia klientskej časti užívateľského rozhrania bola jednoduchá a vyžadovala čo najmenej informácií o štruktúre serverovej časti.

Literatúra

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

[2] Fielding, R. T.: *Architectural Styles and the Design of Network-based Software Architectures*. 2000, University of California, Irvine Doctoral dissertation, ISBN: 0-599-87118-0.

[3] Fielding, R. T.; Gettys, J.; Mogul, J. C.; aj.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, RFC Editor, June 1999.
URL `http://www.rfc-editor.org/rfc/rfc2616.txt`

[4] Hranický, R.; Zobal, L.; Večeřa, V.: Distribuovaná obnova hesel. Technická Zpráva FIT-TR-2017-04, CZ, 2017.
URL `http://www.fit.vutbr.cz/research/view_pub.php?id=11568`

[5] Rescorla, E.: HTTP Over TLS. RFC 2818, RFC Editor, May 2000.
URL `http://www.rfc-editor.org/rfc/rfc2818.txt`