VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

FORENZNÍ ANALÝZA TĚŽEBNÍCH SERVERŮ KRYPTOMĚN

FORENSIC ANALYSIS OF CRYPTOCURRENCY MINING SERVERS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAKUB KELEČÉNI**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR VESELÝ, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/21288/2017/xkelec00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Kelečéni Jakub, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Forenzní analýza těžebních serverů kryptoměn**
Forensic Analysis of Cryptocurrency Mining Servers
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s problematikou těžby kryptoměn se zaměřením na metadata komunikace mezi těžařem a serverem.
2. Rozšiřte existující katalog o nové kryptoměny, pooly a servery.
3. Navrhněte úpravu stávajícího nástroje, která by přidala k informacím v katalogu i časovou rovinu a zvýšila spolehlivost uložených metadat.
4. Dle doporučení vedoucího implementujte rozšíření nástroje. Spolu s tím zajistěte veřejnou dostupnost nástroje i v rámci overlay sítí.
5. Testujte a demonstrujte činnost vylepšeného nástroje na anotovaném příkladu. Zhodnoťte dosažené výsledky.

Literatura:

- N. T. Courtois, M. Grajek, R. Naik, The unreasonable fundamental incertitudes behind bitcoin mining, arXiv preprint arXiv:1310.7935.
- J. D'Herdt, Detecting Crypto Currency Mining in Corporate, Tech. rep., SANS Institute (January 2015).

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 včetně.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veselý Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá problematikou ťažby kryptomien, so zameraním na analýzu komunikácie medzi baníkom a serverom. Teoreticky spracúva základné princípy kryptomien, ich ťažbu a komunikačné protokoly pri nej využívané. Popisuje tiež návrh a implementáciu modifikácie existujúceho systému (katalógu), ktorej cieľom je doplnenie časovej roviny do informácií uložených v katalógu. Výsledkom implementovaného rozšírenia bude vyššia spoľahlivosť uložených metadát. Súčasťou práce je aj popis existujúceho systému, funkcionality a jeho účelu.

Abstract

This thesis focuses on the mining of cryptocurrency with emphasis on analysis of communication between miner and server. It describes basic principles of cryptocurrencies, mining and employed communication protocols. The next part of thesis is about design and implement modification of existing system (catalogue). This modification will add temporality to the catalog, what increase reliability of stored metadata. Description, functionality and purpose of existing system is included in the next text.

Kľúčové slová

Kryptomena, bitcoin, forenzná analýza, blockchain, PHP, Laravel, katalóg kryptomien, ťaženie, ťažobný pool, getwork, getblocktemplate, stratum.

Keywords

Cryptocurrency, bitcoin, forensic analysis, blockchain, PHP, Laravel, cryptocurrency catalogue, mining, mining pool, getwork, getblocktemplate, stratum.

Citácia

KELEČÉNI, Jakub. *Forenzní analýza těžebních serverů kryptoměn.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Veselý, Ph.D.

Forenzní analýza těžebních serverů kryptoměn

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Vladimíra Veselého, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Jakub Kelečéni
19. mája 2018

Poďakovanie

Ďakujem vedúcemu práce doktorovi Vladimírovi Veselému, za jeho odbornú pomoc, podporu a motiváciu, ktorú mi poskytoval. Veľké poďakovanie patrí taktiež mojim rodičom za to, že mi umožnili štúdium a podporovali ma vo všetkých smeroch. Rád by som poďakoval aj svojim súrodencom a kamarátom, ktorý ma morálne podporovali a boli vždy na blízku keď som potreboval rozptýlenie a oddych od školských a pracovných povinností.

Každému kto je na počiatku čítania môjho diela odporúčam, aby sa pred tým poriadne najedol. Takéto čítanie môže byť celkom vyčerpávajúce a preto je dôležité mať plný žalúdok, nech túto činnosť nie je potrebné prerušovať. Riešením môže byť príprava jedla podľa môjho jednoduchého a rýchleho receptu, ktorý zvládne aj ne-kuchár ako ja.

Výsledkom nášho snaženia bude kurací wrap s guacomale. Na prípravu potrebujeme zrelé avokádo, tortilla placky, cesnak, cibuľu, cherry paradajky, kuracie prsia (tofu, robi alebo inú alternatívu pre prípadných vegetariánov), tvrdý syr, zeleninu podľa chuti (odporúčam kukuricu, mix šalátov), soľ, korenie, citrónovú šťavu. Najprv si pripravíme guacomale a to tak, že avokádo rozpolíme, vyberieme kôstku a vydlabeme lyžicou. Dužinu roztlačíme vidličkou, pridáme nadrobno nasekané cherry paradajky a cibuľu, roztlačený cesnak. Zmes dochutíme citrónovou šťavou a soľou. Kuracie mäso nakrájame na kocky, dochutíme sójovou omáčkou, korením a soľou podľa chuti (môže sa namarínovať). Následne ho opečieme a podusíme do mäkka. Môžeme pridať kečup a horčicu. Tortilly orosíme vodou a vložíme na pár sekúnd do mikrovlnky alebo zohrejeme na panvici. Následne tortillu potrieme guacomale a obložíme mäsom a zeleninou. Pred zabalením môžeme pokvapkať nejakým designom (ak je po ruke) a posypať nestrúhaným syrom. Hotový pokrm si môžete vychutnať so svojim obľúbeným pivom.

Obsah

Kapitola 1

Úvod

Táto práca spracúva problematiku krpytomien. Zameriava sa pri tom na analýzu komunikácie medzi baníkom (minerom) a ťažobným serverom. Text práce je logicky rozčlenený do niekoľkých kapitol, ktoré pozostávajú z teoretického pohľadu na riešený problém a praktickej realizácií požiadaviek špecifikovaných zadaním práce.

Predmetom teoretickej časti práce je rozbor komunikácie medzi „baníkom“ (minerom) a serverom so zameraním na vymieňané metadáta. Ďalej sú spracované základné princípy kryptomien. Teoretická časť práce je obsiahnutá v kapitole 2.

Cieľom praktickej časti je návrh (viď kap. 4) rozšírenia funkcionality existujúceho nástroja/služby – katalógu kryptomien. Rozšírenie má do tohto nástroja pridať časovú rovinu informácií. Vďaka tomu budú informácie uložené v katalógu aktuálnejšie a spoľahlivejšie.

Popis katalógu a implementačných detailov je spracovaný v samostatnej kapitole (viď kap. 3). Tá je tvorená z niekoľkých častí popisujúcich konkrétnu oblasť. V kapitole je tiež zahrnutý popis rozšírenia katalógu o nové kryptomeny, pooly a servery.

Nástroj je aktuálne verejne dostupný na internete[1]. Jednou z požiadaviek vypracovania práce bolo sprístupnenie tohto nástroja v rámci overlay sietí. Detaily súvisiace z týmto bodom sú popísané v kapitole 5.

Na základe vypracovaného návrhu bolo implementované rozšírenie nástroja, ktoré pridalo novú funkcionalitu (viď kap 6). V tejto kapitole je špecifikovaný popis realizácie, ktorým bola dosiahnutá požadovaná funkcionalita. Ten pozostáva z vysoko-úrovňového popisu riešenia a implementačných detailov. Implementácia je logicky rozdelená do súvisiacich blokov. Obsiahnutý je tiež popis zmien, ktoré bolo potrebné vykonať v rámci zachovania závislostí a integrácie do existujúceho nástroja.

Posledná časť práce je venovaná testovaniu a zhodnoteniu výsledkov (viď kap. 7). Popisuje metodiku testovania doplnenej funkcionality a hodnotí získané výsledky v porovnaní s existujúcim riešením. Táto časť tiež obsahuje diskusiu k možným budúcim rozšíreniam katalógu.

[1] http://smashed.fit.vutbr.cz

Kapitola 2

Princípy kryptommien

Kapitola stručne popisuje a vysvetľuje základné princípy a pojmy súvisiace s oblasťou kryptomien. Obsiahlejšiu časť tejto kapitoly tvorí popis komunikácie vo vzťahu `miner` $\Longleftrightarrow$ `server` v jednotlivých protokoloch.

2.1 Kryptomena

Pojem kryptomena môže byť definovaný ako digitálny majetok (aktívum, prostriedok), prostredníctvom ktorého je možné vykonávať transakcie. Zvyčajne je založený na nejakom kryptografickom probléme, vďaka čomu poskytuje vysokú mieru bezpečnosti toku transakcií. Na rovnakom kryptografickom probléme je tiež postavené získavanie (ťaženie/generovanie) nových jednotiek daného prostriedku[7].

V súčasnosti existuje relatívne veľké množstvo kryptomien. Medzi najznámejšie možno zaradiť Bitcoin, Ethereum, Monero, atď. Ich trhová hodnota má rastúcu tendenciu a v kontexte globálnej ekonomiky začínajú byť tieto meny považované za plnohodnotné platidlo. Tento fakt so sebou prináša mnoho komplikácií súvisiacich s bezpečnosťou, stabilitou, legislatívou krajín a so spĺňaním regulácií bánk. Na druhej strane kryptomeny poskytujú mnoho výhod, ako napr. rýchlosť platieb, anonymita (možno považovať aj ako negatívnu vlastnosť), decentralizácia, nízke (žiadne) poplatky.

2.1.1 Prehľad vybraných kryptomien

Stručná charakteristika uvedených mien:

- Bitcoin[23][8] - prvá decentralizovaná mena, najrozšírenejšia, má najväčšie zastúpenie na trhu;
- Ethereum[30] - ide o `Turing-complete` jazyk, nad ktorým je postavená táto mena;
- Ethereum Classic[11] - alternatívna verzia meny Ethereum, jej blockchain nezahŕňa DAO Hard-fork[1], taktiež turingovsky úplná;
- Litecoin[19] - prvá kryptomena využívajúca `Scrypt` hašovací algoritmus;
- Dash[10] - mena založená na Bitcoin-e, prináša okamžité vykonanie transakcií, lepšiu bezpečnosť a súkromné transakcie;

[1] popis: https://www.investopedia.com/news/dao-hacker-donates-stolen-funds-ethereum-classic-dev-team/

Názov meny	Značka	Rok	Zakladateľ	Hash	Blockchain
Bitcoin	BTC	2009	Satoshi Nakamoto	SHA-256d	PoW
Ethereum	ETH	2015	Vitalik Buterin	Ethash	PoW
Ethereum Classic	ETC	2015	N/A	Ethash	PoW
Litecoin	LTC	2011	Charles Lee	Scrypt	PoW
Dash	DASH	2014	Evan Duffield	X11	PoW, PoS
Dogecoin	DOGE	2013	J. Palmer, B. Markus	Scrypt	PoW
Monero	XMR	2014	Monero Core Team	CryptoNight	PoW
Zcash	ZEC	2016	Zooko Wilcox	Equihash	PoW
Namecoin	NMC	2011	Vincent Durham	SHA-256d	PoW
Peercoin	PPC	2012	Sunny King	SHA-256d	PoW, PoS
Viacoin	VIA	2014	N/A	Scrypt	AuxPoW
Vertcoin	VTC	2014	Bushido	Lyra2RE	PoW
Bitcoin Cash	BCH	2017	N/A	SHA-256d	PoW
Ripple	XRP	2012	Arthur Britto	N/A	PoW

Tabuľka 2.1: Prehľad kryptomien[18].

- Dogecoin[29] - vytvorená z internetového „meme“ obrázku, uvedená ako vtip avšak získala si veľké množstvo užívateľov a jej hodnota výrazne stúpla;
- Monero[22][32] - zameriava sa na decentralizáciu, anonymitu a škálovatelnosť;
- Zcash[42][33] - decentralizovaná, open-source mena, poskytuje vysokú mieru bezpečnosti a anonymity transakcií;
- Namecoin[24] - prvá mena, ktorá vznikla „forkom“ Bitcoinu, prináša vylepšenia v bezpečnosti a zrýchlenie všetkých súčastí infraštruktúry internetu (napr. DNS);
- Peercoin[16] - prvá mena, ktorá používa PoS (Proof-of-stake) blockchain kryptomeny;
- Viacoin[40] - taktiež odvodená od meny Bitcoin, zrýchllenie transakcí, požíva AuxPoW (Auxiliary proof of work);
- Vertcoin[39] - decentralizovaná mena, ktorej vlasníkmi sú užívatelia, Peer-2-peer, odolná voči ASIC minerom;
- Bitcoin Cash[4] - vznikla „hard-forkom“ Bitcoin-u, zväčšená veľkosť bloku na 8 MB;
- Ripple[34] - mena prinášajúca vysokú rýchlosť transakcií a stabilitu, podpora zo strany svetových bánk;

2.2 Blockchain

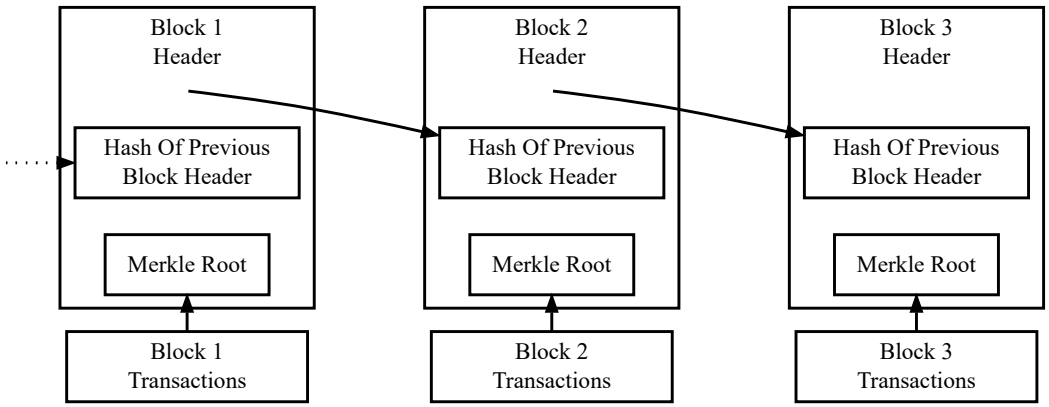
Koncept blockchainu bol predstavený anonymnou skupinou `Satoshi Nakamoto` v roku 2008. Neskôr bol implementovaný ako jadro kryptomeny Bitcoin. Jeho úlohou je udržovať záznamy všetkých transakcií v rámci jednej meny, tzv. „účtovnú knihu“.

Blockchain má uložený každý užívateľ siete pričom musí byť vždy aktuálny (každá zmena sa distribuuje všetkým). Obsahuje transakcie spojené do blokov. Blokom rozumieme skupinu transakcií uzavretú v jednej dátovej štruktúre. Samotné bloky sú zoradené v lineárnom

zozname, kde každý blok obsahuje informáciu o hash hodnote predchádzajúceho bloku. Zreťazením je zabezpečené to, že nebude vymenený blok, ktorý už bol spracovaný v minulosti.

Uchovanie transakcií vo vnútri bloku je realizované v dátovej štruktúre `Merkle Tree`. Ide o binárny strom, v ktorom každý uzol obsahuje hash hodnotu vypočítanú podľa hodnôt potomkov. Transakcia v mene Bitcoin tvorí listy takéhoto stromu. Hash koreňového uzlu je uchovaný v hlavičke bloku. Vďaka tejto architektúre možno overovať integritu transakcií bez toho, aby museli byť kontrolované všetky listy (transakcie) v bloku[28].

Zjednodušená schéma ilustrujúca štruktúru blockchainu je viditeľná na obrázku 2.1.



Obr. 2.1: Schéma štruktúry blockchainu (prevzaté z [3]).

Popis blockchainu uvedený vyššie sa vzťahuje k architektúre meny Bitcoin. Každá mena môže blockchain implementovať iným spôsobom. Vo väčšine prípadov ide o malé modifikácie popísaného princípu (rozdiel v uchovaných informáciach, spôsob určenia platného blockchainu, atď).

2.2.1 Proof-of-work (PoW)

Ide o spôsob, ktorým je definované, ako užívateľ vykazuje vykonanú prácu v rámci siete. Jednoducho povedané, akú výšku odmeny dostane za svoju prácu. Tento mechanizmus tiež zabezpečuje integritu blockchainu, pretože pri ľubovoľnej zmene bloku musí byť prepočítaný proof-of-work pre všetky bloky (vrátane aktuálneho)[28].

Pravdepodobnosť vyťaženia bloku závisí od množstva práce vykonanej minerom. Napríklad počet cyklov procesora, ktoré prebehli pri overovaní hash-u.

2.2.2 Proof-of-stake (PoS)

Pri tomto prístupe je objem zdrojov (výkon) porovnávaný s počtom jednotiek meny, ktoré miner vlastní. Teda ak niekto vlastní 1% podielu meny v obehu, potom môže vyťažiť 1% „Proof of stake“ blokov[31].

V praxi tento koncept pracuje tak, že každý miner uzavrie stávku na to, ktorý nasledujúci blok bude vložený do blockchainu. Víťazný miner potom získava odmenu a ten, ktorý prehral svoj podiel stráca.

2.2.3 Auxiliary Proof-of-work (AuxPoW)

Ako bolo uvedené vyššie, používa ho mena `Viacoin` a nazýva sa tiež `Scrypt Merged Mining`. Vďaka nemu je možné ťažiť viaceré meny využívajúce algoritmus Scrypt súčasne, bez toho, aby sa vzájomne ovplyvňovali. Typ bloku je podobný tomu, ktorý využíva mena Bitcoin s výnimkou dvoch rozdielov. Prvým je, že hlavička neovplyvňuje náročnosť blockchainu. Druhý rozdiel je v pridaní prvkov, ktoré uchovávajú informáciu o tom, kto vytvoril daný blok a ťažil na nadradenom blockchaine[37].

Merged mining je taktiež dostupný pri iných kryptomenách. Dovoľuje ťažbu viacerých kryptomien, založených na rovnakom algoritme súčasne. Príkladom, okrem uvedeného, môže byť mining Litecoin+Dogecoin, Bitcoin+namecoin, Ethereum+Pascal Coin a pod.

Uvažujme ťažbu dvoch kryptomien. Všetky transakcie v sieti budú zoradené a jednotlivé merkletree extrahované. Ich blockchainy budú identifikované ako hlavné (parent) a vedľajšie (auxiliar). Koreň (`Merkle root`) vedľajšieho je vložený do sekcie `extraNonce` hlavného blockchainu. V momente keď sú informácie spropagované do druhého blockchainu, získavame možnosť použiť jeden proof-of-work algoritmus pre obe kryptomeny.

Jedná sa o komplexný proces, ktorý vyžaduje modifikáciu zdrojových kódov pridávajúci podporu AuxPow. Ťaženie hlavnej kryptomeny nie je žiadnym spôsobom ovplyvnené (spomaľovanie, zníženie pravdepodobnosti riešenia,...). Poskytuje výhodný spôsob ako uplatniť a rozšíriť novú kryptomenu s nízkym hashrate.

Pri združenej ťažbe môžu nastať 3 prípady, kde nájdené riešenie:

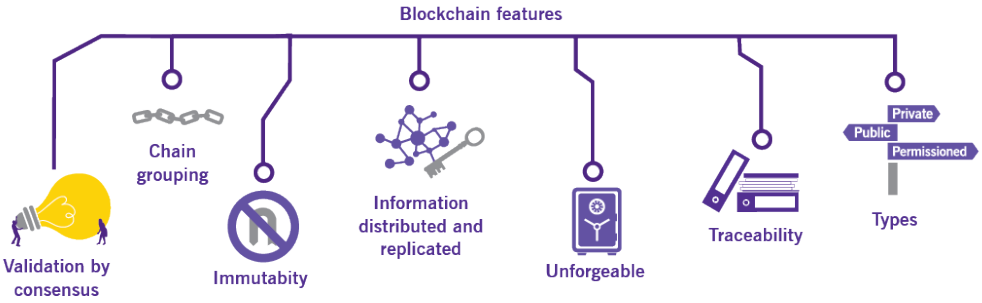
- nevyhovuje obtiažnosti ani jednej mene – voľba inej nonce a pokračovanie v ťažbe
- vyhovuje vedľajšiemu blockchainu – riešenie je propagované v sieti a miner získava odmenu
- vyhovuje obom blockchainom – obe riešenia sú distribuované v príslušnej sieti a miner získava odmenu z oboch

2.2.4 Praktické využitie blockchainu

V súčastnosti sú skúmané rôzne oblasti využitia tejto štruktúry. Ukázalo sa, že blockchain nemusí byť použiteľný iba v súvislosti s kryptomenami. Vlastnosti (viď obrázok 2.2) ktoré poskytuje môžu byť užitočné v rôznych odvetviach. Experimentálne nasadenie prebieha v oblasti zdravotníctva, kde slúži pre vedenie zdravotných záznamov. Uplatnenie nachádza taktiež v oblasti logistiky a dodávok tovaru. Kľúčovou vlastnosťou je formát „zdieľanej knihy“, čo znamená že každý vlastní rovnakú štruktúru a má prístup k rovnakým dátam. Nedochádza ku žiadnej inkonzistencií dát medzi jednotlivými subjektmi. Významná je tiež špecifická vlastnosť pravosti (overených) záznamov a nemožnosť ich modifikácie, zmeny poradia a podobne. V niektorých prípadoch je nad blockchain zavedené členenie prístupu na základe oprávnení. Tým je možné povoliť prístup k transakciám iba autorizovaným subjektom. Riešenia postavené nad konceptom blockchainu majú veľkú odolnosť voči chybám a prinášajú dobrú bezpečnosť.

2.3 Ťaženie – mining

Pre kryptomeny, ktoré používajú PoW blockchain chápeme mining ako proces vytvárania proof-of-work. Ten sa vykonáva pri ukladaní bloku B do blockchainu, kedy je v sieti generovaný hash H a miner sa snaží vyhľadať hodnotu nonce N takú, že platí: hash N a



Obr. 2.2: Vlastnosti blockchainu (prevzaté z [14]).

hlavičky bloku $B \leq H$. V momente, keď je takýto nonce nájdený, dôjde k vyťaženiu bloku. Prebehne aktualizácia blockchainu pridaním bloku a miner, ktorý našiel správne riešenie dostane odmenu v podobe jednotiek meny. Použitá hashovacia funkcia H závisí na použitej kryptomene[28].

Samotný mining vyžaduje inštaláciu niektorej z podporovaných aplikácií. Do aplikácie je nutné špecifikovať adresu peňaženky a niektoré aplikácie môžu vyžadovať registráciu. Spomínaný hash N je od servera získavaný prostredníctvom niektorého z komunikačných protokolov, ktorý aplikácia implementuje. Jednotlivé protokoly sú popísané v kapitole 2.4.

Existuje viacero možností, ako pristupovať ku ťažbe kryptomeny. Bližší popis možností ťažby je uvedený nižšie.

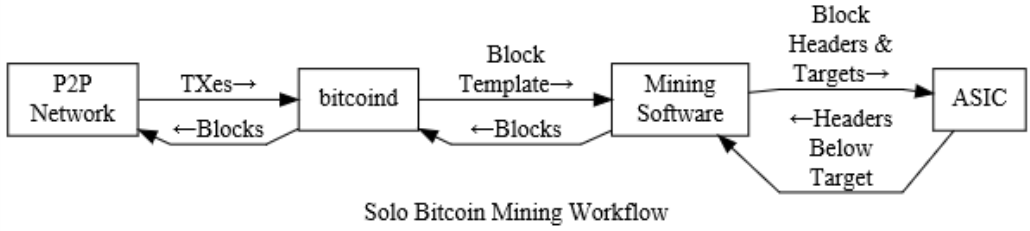
Ťažbu je možné rozdeliť na dve skupiny. V prvej skupine ide o tzv. „iniciálnu“ ťažbu meny, kedy sú jednotky meny v sieti vytvárané. Každá mena môže mať iný dizajn a s tým súvisiaci konečný počet jednotiek, ktoré je možné vyťažiť. Druhá skupina charakterizuje ťažbu ako overovanie transakcií. Za toto overenie sú minerom vyplácané odmeny vo forme jednotiek kryptomeny. V oboch prípadoch je výpočet založený na rovnakom kryptografickom probléme.

2.3.1 Solo mining

Prvou a zároveň najstaršou možnosťou je tzv. `solo mining`. Miner nie je členom žiadnej skupiny (`pool-u`) a odmena za všetky vyťažené bloky náleží iba jemu. Typicky pracuje tak, že miner žiada sieť kryptomeny o nové transakcie. Softvér na strane minera skladá hlavičku bloku, ktorú následne odošle do ťažobného hardvéru (CPU, GPU, ASIC). Hardvér po nájdení vyhovujúceho riešenia navráti hlavičku, softvér vytvorí blok na základe hlavičky a ten následne odošle do siete. Schému ťažby ilustruje obrázok 2.3.

Výhodou takéhoto prístupu je spoľahlivosť, pretože neexistuje žiadna závislosť na ďalšej strane (`pool`). Kladnou stránkou je tiež fakt, že z daného zisku nie sú odvádzané žiadne poplatky, takže celý zisk je pripísaný na účet minera.

Medzi nevýhody patrí zbytočný overhead generovaný pri vykonávaní `pull`, protokolu `Getwork` (2.4.1). Je tiež potrebné si uvedomiť, že dnešná obtiažnosť a celkový HashRate[1] je pri obľúbených menách príliš vysoký. Z toho plynie, že pokiaľ neprevádzkujeme farmu, je zbytočné pokúšať sa ťažiť týmto spôsobom, pretože pravdepodobnosť vyťaženia bloku bude veľmi nízka[15].



Obr. 2.3: Ilustračný priebeh ťažby – `solo mining` (prevzaté z [3]).

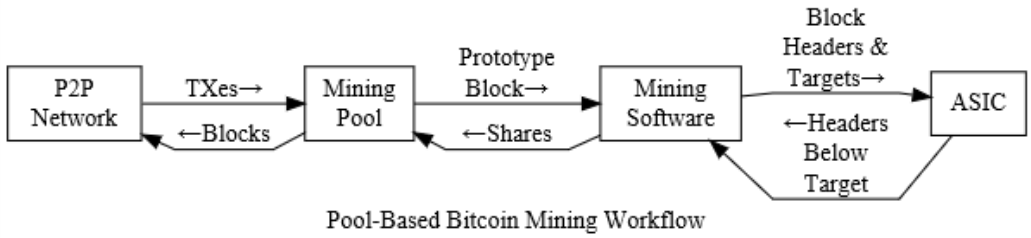
2.3.2 Pooled mining

Princíp tohto prístupu je taký, že zdroje jednotlivých minerov vytvárajú skupinu `pool`, v ktorej sa hľadá riešenie pre daný blok. V momente nájdenia riešenia niektorým užívateľom skupiny, je odmena rozdelená medzi všetkých zúčastnených členov. Tento prístup prináša výhodu v stabilnejšom zisku avšak po menších čiastkach. Nevýhodou je závislosť na prevádzkovateľovi `pool-u`, ktorý v prípade výpadku spôsobí potenciálnu stratu všetkým užívateľom. Výpadok môže byť zapríčinený technickou závadou alebo útokom (napr. DoS) na tento pool[15].

`Pool` nastaví obtiažnosť na hodnotu nižšiu, ako je hodnota stanovená sieťou kryptomeny. Pre minerov je omnoho jednoduchšie nájsť riešenie vyhovujúce takejto hodnote. Je pravdepodobné, že medzi riešeniami od minerov budú aj také, ktoré splnia obtiažnosť siete.

Existuje viacero možností ako realizovať vyplácanie odmien užívateľom. Všetky pracujú s tzv. podielom (share), pre ktorý je omnoho jednoduchšie nájsť riešenie ako pre celý blok. Vďaka nemu je možné vykázať prácu na nájdení bloku. Získaná odmena je potom závislá od počtu potvrdených shares. Proces ťažby je podobný ako pri `solo mining-u` s tým rozdielom, že komunikácia neprebieha priamo zo sieťou kryptomeny, ale s mining pool serverom (viď obrázok 2.4).

Najpoužívanejšie systémy vyplácania odmien sú nasledovné[2]:



Obr. 2.4: Ilustračný priebeh ťažby – pooled mining(prevzaté z [3]).

- Maximum Pay-per-Share (FPPS) – odmena za potvrdenú share je pevne daná, nemenná a vopred známa. Prevádzkovateľ pool-u vypláca odmeny z peňaženky pool-u, na ktorej keď nie je dostatok prostriedkov, dlhy sú zaevidované a vyplatené v momente, keď bude prostriedkov dostatok. Jeho výhodou je to, že poplatky, ktoré platia užívatelia sú zahrnuté k vyplácanej odmene, vďaka čomu sú dosahované vyššie zisky ako v prípade (PPS).

[2]Zdroj: https://en.bitcoin.it/wiki/Mining_pool_reward_FAQ

- Pay-per-Share (PPS) – odmena za share je taktiež vopred známa avšak prináša komplikáciu pre prevádzkovateľa pool-u, pretože vopred nepozná veľkosť odmeny, ktorá sa vyplatí užívateľom. Z toho dôvodu si prevádzkovateľ účtuje relatívne vysoké poplatky za takéto služby. Jej výhodou je to, že užívateľ dostáva odmenu okamžite, bez potreby čakania na potvrdenie nájdeného riešenia. Tento fakt prispieva k minimalizácii rizík v podobe útokov a podvodov.

- Proportional – pri každom nájdení bloku je odmena rozdelená medzi užívateľov, ktorý sa na hľadaní riešenia podieľali (na základe potvrdených shares)

- Score-based (BPM) – každá potvrdená share je ohodnotená na základe veku, odmena je rozdelená medzi užívateľov s ohľadom na toto ohodnotenie. Táto metóda je najmenej stabilná, čo sa týka výšky odmien. Metóda je tiež známa ako `Bitcoin pooled mining` alebo `slush's pool`.

2.3.3 Hardware

V počiatkoch rozvoja kryptomien bolo ťaženie realizované iba na bežných procesoroch. Postupom času sa ukázalo, že grafické karty sú pre túto činnosť omnoho efektívnejšie. Ich efektivita plynie z faktu, že sú schopné vykonať niekoľkonásobne väčšie množstvo inštrukcií za 1 hodinový takt ako procesor. To je zapríčinené rozdielnym počtom jadier – GPU v rádoch tisícov na rozdiel od jednotiek až desiatok v CPU. Prehľad efektivity bežného hardvéru je dostupný na [26].

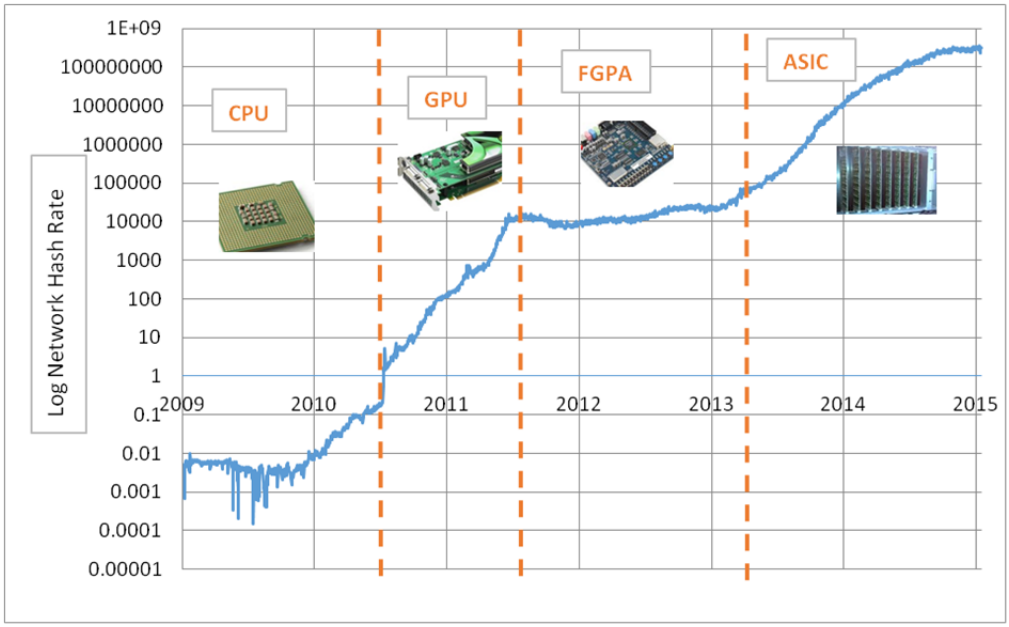
Ďalšou evolúciou bolo použitie programovateľných hradlových polí `FPGA`. Ich nástup priniesol nižšiu spotrebu ako grafické karty pri zachovaní rovnakého, až vyššieho hash rate. Zatiaľ posledným posunom, ktorý je stále aktuálny bol prechod na technológiu `ASIC`. Jedná sa o integrovaný obvod navrhnutý pre špecifickú aplikáciu/úlohu. Takéto riešenie minimalizuje spotrebu (generované teplo) a maximalizuje výkon. Nevýhodou je jedno-účelovosť takýchto zariadení a v prípade nahradenia výkonnejšími zariadeniami alebo inou technológiou budú tieto prakticky nepoužiteľné.

Grafické znázornenie vývoja hardvéru pre ťažbu kryptomien ilustruje obrázok 2.5. Na zvislej osi je uvedený hash rate siete bitcoin zatiaľ čo vodorovná predstavuje čas.

Ťažiť je možné na širokej škále hardvéru, od procesorov až po špeciálne čipy. Pre zvýšenie efektivity je možné zostaviť tzv. `mining rig`, v ktorom je zapojených viacero grafických kariet. Existujú tiež špeciálne zariadenia, `minery` postavené na ASIC, ktoré je možné si zakúpiť a prevádzkovať. Ich obstarávacia cena býva relatívne vysoká. Porovnanie špecializovaného hardvéru je dostupné na [21].

Cloud Mining

Ide o formu ťaženia, kde si miner prenajme určitý objem hardvérových prostriedkov. Miner teda nemá potrebu vlastniť žiaden hardvér, takže mu odpadá množstvo problémov (platba za elektrinu, priestor, hluk, atď). Toto riešenie prináša radu nevýhod ako napr. nižší zisk, viazanosť, dôveryhodnosť poskytovateľa.



Obr. 2.5: Vývoj hardvéru na ťaženie Bitcoin-u (prevzaté z [20]).

2.4 Komunikačné protokoly

Každá kryptomena špecifikuje vlastný protokol/sieť, prostredníctvom ktorej je zabezpečená komunikácia medzi minermi (poolom) a serverom. Definuje typy správ, protokol transportnej vrstvy, ktorý využíva, spôsob pripojenia a podobne.

Vzhľadom k tomu, že v súčastnosti prevláda technika pooled mining, táto práca je zameraná na komunikáciu medzi minermi a poolom. V nasledujúcej časti sú popísané používané komunikačné protokoly, cez ktoré klient žiada pool o „prácu“, ktorú má vykonávať. Po nájdení riešenia zase potrebuje túto informáciu spropagovať do zvyšku siete. To je taktiež zabezpečené prostredníctvom komunikácie s pool serverom.

2.4.1 Getwork

Jedná sa o metódu JSON-RPC[3] (Remote Procedure Call), ktorú používa miner na získanie práce (mining) zo siete kryptomeny. V súčasnosti sa medzi minermi veľmi nepoužíva pretože bol nahradený protokolom `Getblocktemplate`. Môže byť však využívaný interne v rámci komunity (poolu)[13].

Server odovzdá klientovi hlavičku bloku, s tým že mu neposkytuje spôsob, ktorým by ho mohol modifikovať. Výnimkou je možnosť zmeny hodnoty nonce. Tým je logika klienta obmedzená na hľadanie takej hodnoty nonce, pre ktorú bude splnená stanovená obtiažnosť. Po jej nájdení miner zasiela dotaz na server a žiada o ďalšiu prácu. Tento koncept generuje medzi klientom a serverom veľký objem komunikácie.

V protokole je implementovaných niekoľko rozšírení, viď [13], ktoré prinášajú zefektívnenie protokolu apod.

[3]Popis: http://www.jsonrpc.org/specification_v1

Implementované metódy

Protokol využíva HTTP spojenie, prostredníctvom ktorého prenáša metódy JSON-RPC. Rozlišuje jednu metódu – `getwork`. Žiadosť minera o prácu zabezpečuje volanie:

```
{"method": "getwork", "params": [], "id": 1}
```

Odpoveď servera obsahuje parametre `data`, v ktorom je uložená 80 bajtová hlavička bloku v little-endian usporiadaní (formát SHA-2) a `target` udávajúci cieľovú obtiažnosť. Reťazce sú kódované v hexadecimálnom formáte.

Miner sa snaží nájsť vyhovujúce riešenie, ktoré následne odošle serveru ako proof-of-work. Opäť využíva metódu `getwork`, kde uvádza jeden parameter a tým je nájdené riešenie v podobe hashu. Odpoveďou na túto správu je logická hodnota `true`, ak sieť akceptovala riešenie alebo `false` v prípade odmietnutia riešenia. Činnosť protokolu `Getwork` možno popísať Algoritmom 1.

Algoritmus 1 Pseudokód ťaženia kryptomeny – `Getwork`

1: send *getwork* method without parameters
2: receive and decode data: $originalBlock\_header, target$
3: $nonce \leftarrow 0$
4: $block\_header \leftarrow originalBlock\_header + nonce$
5: $result \leftarrow hash(hash(\text{block\_header}))$
6: **while** $result \geq target$ **do**
7: $\quad nonce \leftarrow nonce + 1$
8: $\quad block\_header \leftarrow originalBlock\_header + nonce$
9: $\quad result \leftarrow hash(hash(\text{block\_header}))$
10: **end while**
11: submit share using “method“: “getwork“and “params“: “*result*“

2.4.2 Getblocktemplate (GBT)

Tento protokol vyvinutý komunitou Bitcoin-u. Prináša decentralizáciu a nahrádza pôvodný protokol `Getwork`. Ten už nespĺňal aktuálne požiadavky siete a nadmieru ju zaťažoval. Motivácia vzniku nového protokolu bola tiež v podpore decentralizácie, škálovateľnosti, pool miningu a prechodu na ASIC genreáciu ťažobného HW[5].

Pomocou tohto protokolu je klientovi dodaná šablóna, na základe ktorej je potom schopný generovať vlastnú prácu (bez potreby komunikácie so serverom). Miner dostane celý blok dát, zahŕňajúci dostupné transakcie, z ktorých si môže vyberať. Pre každú inštanciu sa môže rozhodnúť, že ktorá transakcia bude v bloku zahrnutá. Vzhľadom k tomuto princípu dochádza ku generovaniu trochu väčšieho množstva dát medzi klientom a serverom v porovnaní s protokolom `Stratum`.

Proces ťažby a generovania práce je možné charakterizovať algoritmom 2.

Implementované metódy

Komunikácia prebiehajúca medzi klientom a serverom pozostáva z volaní JSON-RPC. V tejto časti sú popísané základné metódy, ktoré sú volané počas ťaženia.

Algoritmus 2 Pseudokód generovania share a hlavičky bloku – `Getblocktemplate`

send *getblocktemplate*
receive and store block template data: *coinbasetxn*, *target*, *transactions*, *version*, *prevBlk_hash*, *time*
coinbase_tx ← *coinbasetxn*[0 : 41]
ctx_lnt ← *coinbasetxn*[42]
tx_data ← *coinbasetxn*[43 : *ctx_lnt*] // read data from coinbasetxn
coinbase ← *coinbase_tx* + (*ctx_lnt*{+length(*extra_tx*)} + *tx_data*{+*extra_tx*+length(*extra_tx*)} // parameters in {} are optional
tx_list ← []
merklehashes ← [] // define new lists
for all *tx* ∈ *transactions* **do**
    tx_list += *coinbase* + *tx*
end for
for all *tx* ∈ *tx_list* **do**
    merklehashes += (hash(hash(*tx*))
end for
while length(*merklehashes*) > 1 **do**
    if length(*merklehashes*) % 2 **then**
        merklehashes += *merklehashes*.prev()
    end if
    for *i* ← 0 to length(*merklehashes*) **do**
        new_hash ← []
        new_hash +=hash(hash(*merklehashes*[*i*] + *merklehashes*[*i* + 1]))
        i ← *i* + 2
    end for
    merklehashes ← *new_hash*
end while
merkleroot ← *merklehashes*.first()
block_hdr ← *version* + *prevBlk_hash* + *merkleroot* + *time* + *target* + *nonce*
// create block header; *nonce* is 0
while *hash*(*block_hdr*) ≥ *target* **do**
    nonce ← *nonce* + 1
    block_hdr ← *version* + *prev_hash* + *merkle_root* + *time* + *target* + *nonce*
end while
share = *block_hdr* + *length*(*tx_list*) + *tx_list*
submit share using “method“: “submitblock“and “params“: “*share*“

- `getblocktemplate` – pred samotným začiatkom ťažby je potrebné vytvoriť spojenie s poolom a získať šablónu bloku. Práve k tomuto účelu slúži uvedené volanie. Správa so žiadosťou má nasledovný formát:

  ```
  {"id": 0, "method": "getblocktemplate", "params": [{"capabilities":
      ["coinbasetxn", "workid", "coinbase/append"]}]}
  ```

  Odpoveď servera obsahuje všetky potrebné informácie k tomu, aby miner mohol začať ťažbu. Nachádza sa tam napr. zoznam transakcií, ktoré môžu tvoriť nový blok, verzia, referencia na predchádzajúci blok, aktuálny čas, obtiažnosť apod. Na strane minera je následne spočítaný koreň – `Merkle root` (pre zvolené transakcie) a zostavená hlavička nového bloku. Teraz môže začať samotná ťažba hľadaním vyhovujúcej hodnoty nonce.

- `submitblock` – metóda, ktorou klient potvrdzuje bloky/share po nájdení vyhovujúceho riešenia. Obsah správy má nasledovnú podobu:

  ```
  {"id": 0, "method": "submitblock", "params": ["020000003c48a2945
      ..."]}
  ```

  Pole params je tvorené konkatenáciou hlavičky bloku, počtom transakcií a identifikátormi transakcií .

Decentralizácia

Pôvodný protokol `Getwork` pracuje tak, že poskytuje minerovy iba hlavičky blokov. Miner samotný v tomto prípade nemá prístup k obsahu bloku a nie je schopný rozhodnutia o akceptovaní transakcie. Ťaženie je riadené na strane pool-u.

Protokol `Getblocktemplate` rieši tento problém a dovoľuje minerovy rozhodovať o tom, ktoré transakcie akceptuje. Týmto spôsobom je zabezpečená decentralizácia a tiež zvýšená bezpečnosť siete pred kompromitovaným poolom.

Pri pohľade na 3. riadok algoritmu 2 vidíme, že miner disponuje transakciami získanými zo šablóny. Prvých 41 bajtov reprezentuje coinbase transakciu. Na 42. bajte sa nachádza celková dĺžka dát transakcií a na zvyšných pozíciach sú uložené dáta transakcií. V nasledujúcom riadku je ukázané vytvorenie coinbase, ktoré pozostáva z coinbase transakcie, celkovej veľkosti a samotných dát transakcií.

Ak server podporuje mutáciu transakcií, tak odpoveď od neho bude obsahovať parameter `mutable` s hodnotou `coinbase/append`. Potom je možné doplniť coinbase o akékoľvek ďalšie transakcie. Celková veľkosť však nesmie presiahnuť 100 bajtov. Po pridaní vlastných dát je potrebné doplniť celkovú veľkosť o dĺžku týchto dát. Za pôvodné dáta transakcií sa potom pripoja tie pridávané, pričom na poslednom bajte musí byť špecifikovaná ich veľkosť. Výsledný prepočet coinbase potom zahŕňa údaje v zložených zátvorkách.

ASIC

Keďže `Getwork` bol navrhnutý tak, že poskytuje iba hlavičku jedného bloku, maximálny hash rate, pri použití rozširujúcej hlavičky `X-Roll-NTime`, je limitovaný na hodnotu 4 GH/s. Systémy ASIC však v dobe uvedenia dosahovali hodnôt niekoľko násobne vyšších (1000 GH/s, v súčastnosti 10000 GH/s) a tak by ich tento protokol nebol schopný obsluhovať. Prístup nového protokolu odstraňuje toto obmedzenie, pretože generovanie bloku presúva zo serveru, na minera. Vďaka tomu je schopný ťažiť maximálnou možnou rýchlosťou.

Škálovateľnosť a rozšíriteľnosť

Protokol výrazne znižuje záťaž siete pri žiadosti o nový blok. Vďaka tomu umožňuje efektívnejší `solo mining` pomocou `bitcoind`. Poskytovatelia pool-ov majú taktiež menšie požiadavky na zdroje, keďže odpadajú nároky na komunikáciu pri dodávaní ďalšej práce.

Nový protokol je navrhnutý tak, aby poskytoval možnosť implementácie budúcich rozšírení. Príkladom môže byť `BIP 23`[4], ktoré sa zaoberá implementáciou rozšírenia, protokolu `Getwork`, nezávislého nad transportným protokolom. Škálovateľnosť protokolu `getblocktemplate` je teda omnoho lepšia.

2.4.3 Stratum

Protokoly `Getwork` a `GBT` operujú nad HTTP, ktoré nie je ideálne v obsluhovaní častých dotazov na konkrétny obsah. Pri pooled miningu existuje efektívnejšia možnosť komunikácie. Touto je možné odstrániť problémy vznikajúce pri `Ntime Rolling`[5], kde dochádza k tomu, že protokol nedokáže dodať dostatok práce výkonnému hardvéru (napr. pri 42 GHash/s by bolo potrebných 10 požiadaviek protokolu Getwork).

`Getwork` prinášal nastavenie intervalu, v ktorom bol zasielaný dotaz o prácu serveru. Voľba správnej konfigurácie nebola jednoduchá, pretože pri nastavení krátkeho intervalu dochádzalo k neúmernému zaťaženiu siete, avšak s vyššou pravdepodobnosťou úspešného nájdenia riešenia. V prípade nastavenia dlhého intervalu, bolo dôsledkom zníženie nárokov na sieť, avšak za cenu nižšej úspešnosti v nájdení správneho riešenia.

Motivácia za vznikom tohto protokolu bola predovšetkým v príchode ASIC ťažobného HW, ktorý dosahoval lepší hashrate v porovnaní s predchádzajúcimi riešeniami (tera hash za sekundu). Výkonnejšie zariadenia však znamenali vyššie požiadavky na komunikačnú sieť. Snahou tohto protokolu bolo znížiť zaťaženie komunikačnej siete (oproti s protokolu Getwork a čiastočne aj GBT). Toto bolo dosiahnuté poskytovaním čo najmenšieho objemu informácií, z ktorých budú mineri schopný samostatného zostavenia blokov.

Protokol bol pôvodne navrhnutý pre bitcoin klienta `Electrum`. Ukázalo sa, že spĺňa všeobecné požiadavky pre ťažbu meny Bitcoin. Následne bol použitý ako základ nového ťažobného protokolu – `Stratum`. Využíva jednoduchý TCP soket, ktorý dáta (payload) zasiela vo forme JSON-RPC 2.0[6] správy. Klient nadviaže TCP spojenie a vytvorí požiadavku na server v podobe JSON správy zakončenú znakom konca riadku – `\n`. Odpovede od servera sú taktiež platnými JSON-RPC fragmentmi obsahujúce odpoveď. Popísané riešenie prináša radu výhod. Je možné ho jednoducho implementovať a odladiť, pretože obe strany posielajú správy v čitateľnej podobe. Protokol je jednoducho rozšíriteľný, bez toho aby obmedzoval spätnú kompatibilitu. Prínosom je tiež podpora JSON-u na väčšine dostupných platformách. Toto riešenie odstraňuje potrebu protokolu HTTP a súvisiace obmedzenia pri jeho použití v miningu.

Hlavnou výhodou oproti `Getwork` je to, že server dokáže kontrolovať „prácu“ vo vlastnej réžií. Odstraňuje tak potrebu použitia alternatívnych riešení v podobe HTTP long-polling[7], rozdeľovania záťaže a problémy súvisiace s paketovým zahltením.[43].

[4] Popis: https://github.com/bitcoin/bips/blob/master/bip-0023.mediawiki

[5] Popis: https://bitcoin.stackexchange.com/questions/1482/what-is-ntime-rolling-nonce-range-and-what-miners-support-it

[6] Popis: http://www.jsonrpc.org/specification

[7] Popis: https://www.pubnub.com/blog/2014-12-01-http-long-polling/

Tento protokol pracuje podobne ako protokol `GBT` s tým rozdielom, že klientovi poskytne iba tie informácie, ktoré sú nevyhnutné pre vytvorenie bloku: coinbase transakcie[8], častí merkletree potrebných pre vytvorenie nového merkleroot, cieľovú obtiažnosť poolu a ostatných informácií potrebných k vytvoreniu nového bloku. Spotrebúva najmenšiu šírku pásma zo všetkých popisovaných protokolov. Je omnoho jednoduchší a lepšie implementovateľný.

Nevýhoda v porovnaní s `GBT` je v tom, že miner nedokáže voliť konkrétne transakcie. Vo väčšine prípadov to však minera nezaujíma, pretože mu ide o maximálny zisk.

Komunikácia je zabezpečená prostredníctvom niekoľkých metód, ktoré je protokol schopný obslúžiť. Metódy sú rozdelené na základe smeru komunikácie. Teda jedným smerom klient → server a v druhom smere server → klient (viď 2.4.3). Demonštračný obsah správ vymenených pri začiatku ťažby je pripojený v prílohe B.

Generovanie lokálnej práce

Miner získa od servera dva druhy informácií po pripojení do poolu – `ExtraNonce1` a `ExtraNonce2_size`. Hodnota `ExtraNonce2_size` uvádza, že koľko bajtov má byť použitých čítačom `ExtraNonce2`. Ide o hexadacimálny čítač, ktorý by mal byť zarovnaný na počet bajtov uvedený v `ExtraNonce2_size`. Jeho hodnota je náhodne zvolená. Zabezpečuje entropiu na strane minera, ktorá zmenšuje pravdepodobnosť kolízie pri hľadaní riešenia.

V momente keď pool odošle prácu, vygeneruje sa transakcia pozostávajúca z dvoch častí. Miner zostavuje hlavičku bloku, na ktorej prebieha výpočet hashu. Tá vyžaduje koreň stromu, merkleroot, ktorého počiatočná hodnota je rovná dvojitému hashu coinbase. Coinbase určíme ako konkatenáciu hodnôt `Coinb1, ExtraNonce1, ExtraNonce2, Coinb2`. Všetky uvedené hodnoty okrem `ExtraNonce2` sú určené pool serverom, kde `Coinb1` je počiatočná časť cionbase transakcie, `ExtraNonce1` zabezpečuje entropiu zo strany servera a to tak, že každý člen poolu pracujúci na rovnakej hlavičke bloku dostane jedinečné číslo (pool tak rieši možnú duplicitnú prácu), `Coinb2` je záverečná časť coinbase transakcie.

V druhej časti sa vykonáva kombinovanie prvej vetvy merkletree aktuálneho riešenia a počiatočnej hodnoty merkleroot. Dostávame reťazec, ktorý tvorí výsledok. Potom sa pokračuje nasledujúcou vetvou merkletree až dovtedy, kým nebudú skombinované hashe všetkých vetiev stromu s predchádzajúcimi výsledkami.

Týmto postupom je vytvorený unikátny koreň stromu pre novú hlavičku bloku, ktorá môže byť štandardne ťažená (32 bitovým nonce čítačom)[35].

Celkový proces ťažby a generovania share popisuje algoritmus 3:

Metódy client-to-server

- `mining.authorize("username", "password")` – autorizuje minera (napr. pred potvrdením share), výsledkom je hodnota true po úspešnej autorizácií a false v prípade neúspechu, parameter password je voliteľný
- `mining.extranonce.subscribe()` – klient informuje sever, že podporuje ExtraNonce
- `mining.get_transactions("job id")` – dotaz na transakcie bloku špecifikované identifikátorom úlohy, odpoveď obsahuje pole transakcií v hexadecimálnom tvare

[8] Popis: https://www.cryptocompare.com/coins/guides/what-is-a-coinbase-or-generation-transaction/

Algoritmus 3 Pseudokód generovania share a hlavičky bloku – `Stratum`

1: send $mining.subscribe()$
2: receive $mining.notify$, $mining.set\_difficulty$ and store values
3: randomly generate $ExtraNonce2$ unique for job_id :
$ExtraNonce2 < 2^{ExtraNonce2\_size \cdot 8}$
4: $coinbase \leftarrow Coinb1 + ExtraNonce1 + ExtraNonce2 + Coinb2$
5: $coinbase\_hash \leftarrow$ hash(hash(toBinaryHexString($coinbase$)))
6: $merkle\_root \leftarrow coinbase\_hash$
7: **for all** $i \in merkle\_branch$ **do**
8: $merkle\_root \leftarrow$ hash(hash($merkle\_root +$ toBinaryHexString(i)))
9: **end for**
10: $merkle\_root \leftarrow$ reverseByteOrder($merkle\_root$)
11: $block\_header \leftarrow version + prev\_hash + merkle\_root + time + target + nonce$
// $nonce$ is 0
12: **while** $hash(block\_header) \geq target$ **do**
13: $nonce \leftarrow nonce + 1$
14: $block\_header \leftarrow version + prev\_hash + merkle\_root + time + target + nonce$
15: **end while**
16: submit share using "method": "mining.submit" and "params":
("$username$", "job_id", "$ExtraNonce2$", "$time$", "$nonce$")

- `mining.submit("username", "job id", "ExtraNonce2", "nTime", "nOnce")` – potvrdenie share kde parametre predstavujú: meno minera, Job ID, ExtraNonce2, nTime, nOnce. Odpoveď od servera má hodnotu true, ak akceptoval požiadavok, inak false.
- `mining.subscribe("user agent/version", "extranonce1")` – nadviazanie spojenia so serverom prostredníctvom ktorého bude klient získavať prácu (nutné odoslať pred ostatnou komunikáciou). Prvý parameter určuje verziu klienta, druhý je voliteľný a používa sa pri pokračovaní spojenia.
- `mining.suggest_difficulty(value)` – parametrom value je špecifikovaná preferovaná náročnosť pre share pool-u, server nie je povinný tejto žiadosti vyhovieť
- `mining.suggest_target("full hex share target")` – klient oznamuje preferovanú veľkosť share-u, server nie je povinný tejto žiadosti vyhovieť

Metódy server-to-client

- `client.get_version()` – žiadosť o meno a verziu klienta
- `client.reconnect("hostname", port, waittime)` – žiadosť o znovu pripojenie klienta po prerušení spojenia. Parameter hostname (port) určuje adresu (port) servera, ktorá keď je klientovi neznáma, mala by byť ignorovaná. Parameter waittime špecifikuje čas v sekundách, po ktorého uplýnutí sa pokúsi o pripojenie.
- `client.show_message("message")` – zaslanie správy užívateľovi vo formáte čitateľného textu

- `mining.notify()` – Zaslanie notifikácie klientovi použitím nasledovných parametrov:
    1. Job ID – je zahrnuté keď miner potvrdí výsledok, ktorý je možné spárovať s príslušnou transakciou.
    2. Hash predchádzajúceho bloku (používané pri vytváraní hlavičký)
    3. Generovanie prvej časti transakcie – miner vloží ExtraNonce1 a ExtraNonce2
    4. Generovanie druhej časti transakcie – miner pripojí túto časť dát transakcie za dve hodnoty ExtraNonce z predchádzajúceho kroku
    5. Zoznam vetiev merkletree – generácia transakcie je hashovaná voči jednotlivým vetvám stromu s cieľom vytvorenia koreňa (merkleroot)
    6. Verzia bloku bitcoinu (použitá v hlavičke bloku)
    7. nBits – zakódovaná obtiažnosť siete (použitá v hlavičke bloku)
    8. nTime – aktuálny čas (ak je podporovaný nTime rolling, nemôže rásť rýchlejšie ako nTime)
    9. Clean Jobs – ak je true, minery okamžite zrušia aktuálnu ťažbu a prechádzajú na nový job. V prípade, že je false tak môžu dokončiť, ale po vyčerpaní rozsahu nonce sa posunú na ďalší.
- `mining.set_difficulty(difficulty)` – možnosť nastavenia obtiažnosti share, ktoré sa aplikuje od ďalšieho prijatého jobu (záleží od nastavenia, niektoré pooly vyžadujú okamžitý prechod na nový job)
- `mining.set_extranonce("extranonce1", extranonce2_size)` – v prípade zaslania tejto správy je možné prepísať pôvodné parametre jobu

2.4.4 JSON-RPC 2.0

Jedná sa o jednoduchý, bezstatový protokol volania vzdialených procedúr. Špecifikuje niekoľko dátových štruktúr a pravidiel. Je nezávislí od komunikačného protokolu. Používa typový systém JSON-u, teda pracuje s primitívnymi typmi String, Number, Bool, Null a štruktúrovanými typmi Object a Array.

V princípe ide o komunikáciu klient-server kde klient vytvára požiadavky na objekty a spracováva objekty odpovedí (Response object). Server generuje odpovede na požiadavky a obsluhuje objekty žiadostí (Request object). Verzia protokolu 2.0 nie je spätne kompatibilná s verziou predchádzajúcou.

Request object

Správa so žiadosťou od klienta obsahuje nasledujúce parametre:

- jsonrpc – reťazec špecifikujúci verziu JSON-RPC, musí byť 2.0
- method – reťazec popisujúci názov volanej metódy/procedúry
- params – štruktúrovaná hodnota uchovávajúca argumenty volanej metódy, je voliteľný
- id – identifikátor spojenia určený klientom, musí byť typu String, Number alebo Null. V prípade, že nie je špecifikovaný, správa má sémantiku notifikácie, na ktorú server nie je povinný reagovať. Odpoveď servera zahŕňa rovnaké id, čím je zabezpečené priradenie objektu žiadosti ku odpovedi.

Response object

Po prijatí žiadosti je server povinný odpovedať s výnimkou notifikácie. Odpoveď má formát JSON objektu, ktorý obsahuje nasledujúce parametre:

- jsonrpc – reťazec špecifikujúci verziu JSON-RPC musí byť 2.0
- result – hodnota parametru je určená volanou metódou, v prípade že bolo volanie úspešné parameter musí byť špecifikovaný (nemusí byť ak došlo k chybe)
- error – parameter vyžadovaný iba pri výskyte chyby
- id – ide o povinný parameter, ktorý musí mať rovnakú hodnotu aká bola v žiadosti od klienta, pri výskyte chyby môže byť Null

Kapitola 3

Katalóg (sMaSheD)

Táto časť je venovaná popisu existujúceho nástroja `sMaSheD`. Ide o katalóg v ktorom sú uložené informácie súvisiace so službami pre ťažbu kryptomien. Konkrétne obsahuje záznamy o kryptomenách a ťažobných pooloch. Taktiež uchováva doménové mená, porty a adresy patriace do príslušných poolov.

3.1 Motivácia

Tento systém vznikol za účelom zhromaždenia informácií súvisiacich s ťažením kryptomien. Cieľom je poskytnutie verejnej databáze údajov, na základe ktorých môže byť odhalená nežiadúca komunikácia v rámci počítačovej siete, alebo iná aktivita vo vzťahu ku kryptomenám, ktorá nie je v súlade s pravidlami organizície alebo so zákonom. Nástroj môže byť nápomocný pri detekcií ťažby kryptomien na sieťovej vrstve organizácií, viď 3.1.1.

Pôvodný nástroj bol vytvorený v rámci projektu `Tarzan`[36] vedeckou skupinou `NES@FIT`[25]. Projekt sa venuje vývoju integrovanej platformy pre spracovanie digitálnych dát z bezpečnostných incidentov. Nástroj `sMaSheD`, ktorého vylepšenie je predmetom tejto práce, smeruje na oblasť riešenia incidentov súvisiacich s kryptomenami (Bitcoin) a prípadnú forenznú analýzu sieťovej prevádzky v tejto oblasti.

Výsledkom implementácie rozšírenia bude katalóg poskytovať presnejšie informácie, vďaka čomu sa zvýši celková použiteľnosť riešenia a spoľahlivosť uložených metadát.

3.1.1 Detekcia ťažby kryptomien

Ťažba kryptomien je dnes čoraz viac populárna. Užívatelia počítačov sa snažia zúžitkovať dostupný výpočtový výkon s cieľom získania finančnej odmeny. Problémom je, že takáto činnosť býva často vykonávaná na pôde/v sieti organizácií (školy, korporácie, verejné prístupové body, ...), kde môže dochádzať k porušeniu pravidiel organizácie. Vzhľadom k tomu, že ťažba kryptomeny môže spôsobovať rôzne druhy bezpečnostných hrozieb, je potrebné takúto činnosť identifikovať a eliminovať. Hrozby môžu vzniknúť predovšetkým používaním pochybného ťažobného softvéru, ktorý obsahuje škodlivý kód. Okrem prípadnej zraniteľnosti dochádza taktiež k nadmernej spotrebe elektrickej energie a nežiadúcemu zahlteniu sieťovej infraštruktúry.

Predchádzanie a detekcia uvedenej činnosti môže prebiehať na rôznych úrovniach. Tieto je možné rozdeliť na[9]:

- fyzickú
- sieťovú
- hostiteľskú
- personálnu

Zabezpečenie fyzickej vrstvy súvisí predovšetkým so zamedzením fyzického (obmedzenie prístupu k HW, uzamknutie v datacentre) a logického prístupu (heslá, separácia privilégií). Vhodné je tiež výhradné použitie autorizovaných zariadení a ich pravidelná kontrola.

Detekcia nežiadúcej prevádzky na sieťovej vrstve je vykonávaná prostredníctvom jej analýzy. Dôležitú úlohu tu zohráva odlíšenie štandardného stavu (baseline) od abnormálneho, pri ktorom je identifikované podozrivé správanie v rámci siete. Existuje viacero možností, na základe ktorých ho možno klasifikovať. Môže ísť napríklad o pravidelné DNS dotazy na pochybné adresy, ustanovenie TCP spojenia na špecifickom porte, IP adrese apod. Výhodné je tiež použitie prístupu „implicit deny“, pri ktorom je vyžadované explicitné povolenie určitej komunikácie (patrične odôvodnené).

Ošetrenie hostiteľskej vrstvy sa zameriava na operačný systém a aplikácie pod ním spustené. Predpokladá sa použitie antivírového softvéru, kontroly zabezpečenia systému, audity apod. Detekcia infikovaných zariadení môže prebiehať aj monitorovaním využitia zdrojov s ohlásením podozrivého zariadenia. Dôležité je tiež obmedzenie administrátorských privilégií tam, kde nie sú vyžadované.

Personálna úroveň bezpečnosti je tvorená rôznymi ustanoveniami a pravidlami, s ktorými musí užívateľ siete súhlasiť. Pri zistení porušenia pravidiel je povinný niesť zodpovednosť za čin a vyvodené dôsledky. Doporučené je tiež logické rozdelenie povinností medzi viacerých užívateľov, ktoré sú navzájom previazané. [9]

3.2 Uložené informácie

Katalóg poskytuje možnosť vkladať informácie o kryptomenách, pooloch, serveroch a portoch. Na začiatku vypracovania tejto práce bola v nástroji uložená testovacia vzorka dát, ktorá pozostávala z

- 12. kryptomien
- 23. poolov
- 46. serverov
- 92. portov
- 68. adries

Prvý bod zadania požaduje rozšírenie katalógu o nové kryptomeny, pooly a servery. Vkladanie nových záznamov do katalógu pozostávalo z niekoľkých krokov. Najprv bolo potrebné sledovať a analyzovať blockchain uložených kryptomien. To prebiehalo cez tzv. `block explorer` (prehliadač blokov). Jedná sa o nástroj dostupný štandardne pre každú kryptomenu, pomocou ktorého je možné zobraziť informácie o vyťažených blokoch. Na

základe zistení z vykonanej analýzy boli určené pooly, ktoré majú najväčší podiel vyťažených blokov. Nasledovalo prehľadávanie webových stránok jednotlivých poolov, pri ktorom boli vyhľadávané konfiguračné parametre potrebné pre pripojenie minera ku tomuto poolu. Spomínané parametre pozostávajú väčšinou z niekoľkých doménových mien/IP adries (pre určitú geografickú oblasť) a niekoľkých portov, prostredníctvom ktorých sú dostupné ťažobné servery. Príklad formy uvádzania parametrov, poskytovateľom `f2pool`, je viditeľný na obrázku 3.1.

Primary URL	stratum+tcp://stratum.f2pool.com:3333
Secondary URL	stratum+tcp://stratum.f2pool.com:25
U.S. Servers	stratum+tcp://stratum-us.f2pool.com:3333
Payout	**PPS+ at 2.5%**
Estimated Coins Per Day	0.00005991 BTC / Thash/s
Time of Settlement	Everyday at midnight UTC, we settle your revenue given the balance greater than or equal to the payout threshold, the payo ut address not empty, and no payout address change was performed in the past 72 hours.
Time of Payment	Auto payout daily within 8 hours (before 8:00 UTC) after successful settlement.
Payout Threshold	0.005 BTC

Obr. 3.1: Parametre pre ťažbu kryptomeny Bitcoin v poole `f2pool` (prevzaté z [12]).

Niektoré meny/prehliadače blokov poskytujú priamo štatistiku o najúspešnejších pooloch/mineroch, iné nemusia zobrazovať ani vlastníka adresy. Z toho dôvodu prebiehalo vyhľadávanie najväčších, najlepších a najspoľahlivejších poolov na internete, kde boli využité informácie uvádzané na rôznych fórach komunity, článkoch, webových stránkach apod. V prípade, že bol do katalógu vkladaný významný pool (veľké zastúpenie vyťažených blokov), pre určitú kryptomenu, a poskytoval možnosť ťažby iných kryptomien, potom boli do katalógu vložené aj údaje týkajúce sa všetkých ťažiteľných mien v tomto poole.

Snahou bolo pokryť čo najväčšiu časť ťažobných poolov. Ide však o časovo náročnú úlohu, nehovoriac o komplikovanej orientácií na webových stránkach poolov, ktorých obsah je vzhľadom k lokalite, často napísaný iným ako latinským písmom. Tieto stránky bývajú naprogramované takým spôsobom, pri ktorom nepomôže ani prekladač.

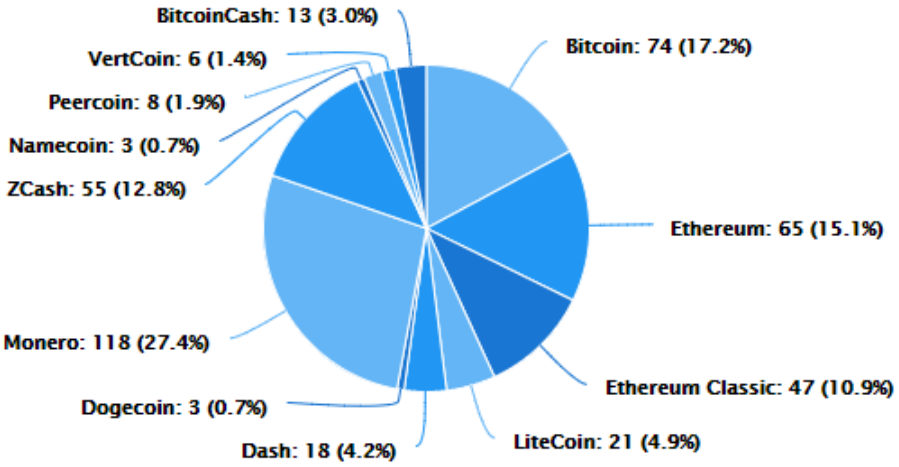
Výsledkom naplnenia katalógu sa podarilo dosiahnuť aktuálny stav, ktorý je viditeľný na obrázku 3.2.

Name	Count
pool count	56
crypto count	13
server count	184
port count	18
address count	459

Obr. 3.2: Aktuálny počet informácií uložených v katalógu.

K tomu, aby katalóg uchovával aktuálne informácie, by bolo vhodné vykonávať pravidelnú aktualizáciu informácií o uložených pooloch. Nakoľko ide o mechanický proces, v budúcnosti je možné rozšíriť nástroj o takúto funkcionalitu.

Zastúpenie celkového počtu záznamov vo vzťahu ku konkrétnym kryptomenám je viditeľné na obrázku 3.3. Obrázky 3.3 a 3.2 sú získané z panelu `Dashboard` implementovaného v rámci katalógu.



Obr. 3.3: Zastúpenie kryptomien v rámci katalógu.

3.3 Popis použitých technológií

Nástroj je dostupný ako webová aplikácia[1] napísaná v programovacom jazyku PHP 7 s využitím frameworku `Laravel`[17] vo verzí 5.4. Dizajn stránky je založený na knižnici `Bootstrap 3`[6]. Aktuálne riešenie beží nad webovým serverom `Apache 2`[2] na operačnom systéme `Ubuntu 16.04 LTS`[38].

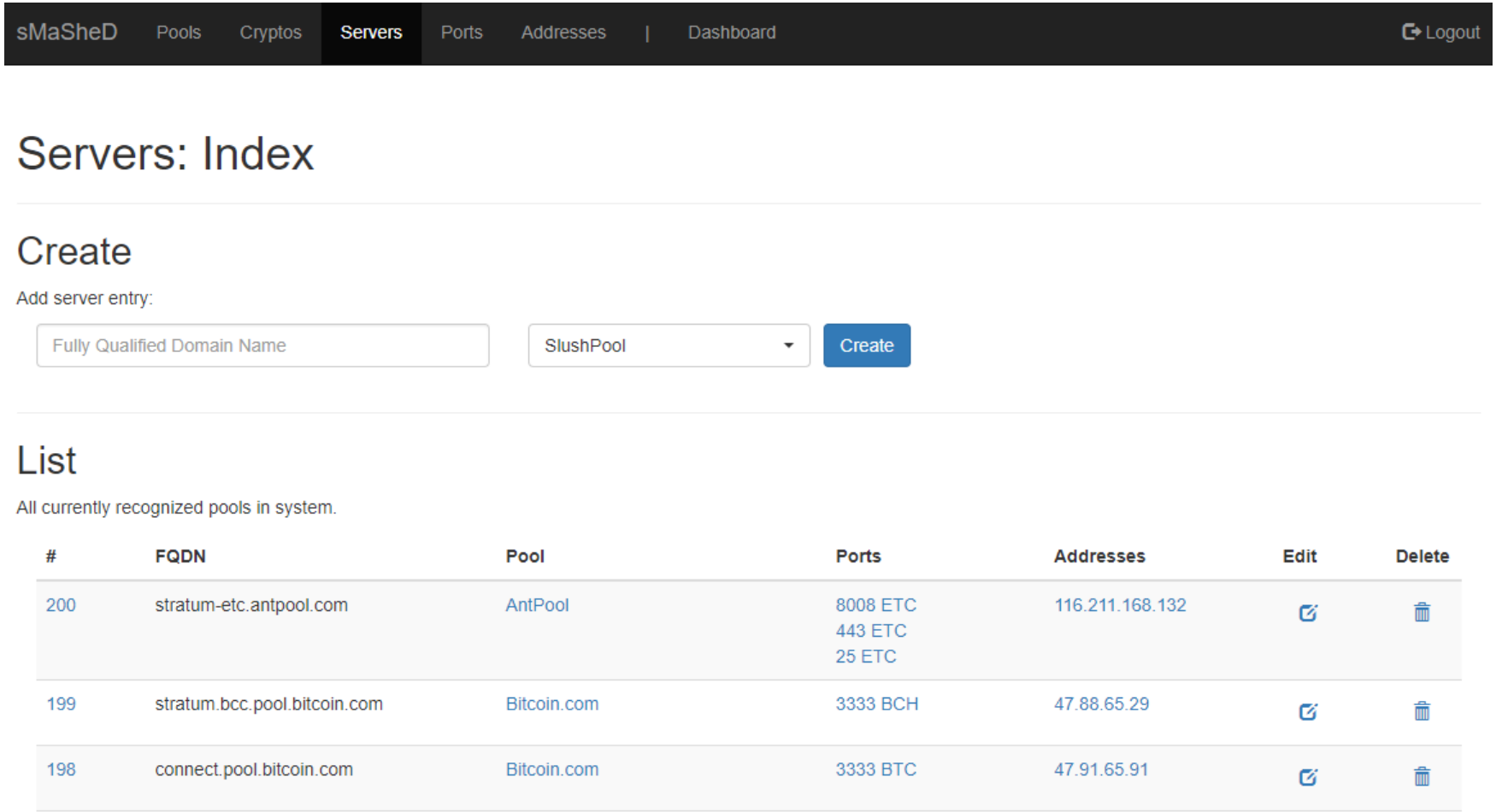
3.4 Aktuálny stav a implementácia

Ako už bolo popísané vyššie, jedná sa o funkčné riešenie, ktoré je naplnené relatívne veľkým objemom dát. Systém podporuje administrátorské rozhranie, prostredníctvom ktorého je možné modifikovať dáta uložené v katalógu tzn. pridávať/odstraňovať/modifikovať kryptomeny, pooly, adresy serverov a porty. Implementovaný je tiež panel (`Dashboard`), ktorý zobrazuje súhrnné dáta a interpretuje ich aj v grafickej podobe. Grafické rozhranie panelu zobrazujúceho informácie o serveroch ilustruje obrázok 3.4.

Adresy serverov, špecifikované vo formáte presného názvu domény (FQDN), sú prekladané na IP adresy pomocou funkcie `gethostbynamel` dostupnej v jazyku PHP. Jej volanie je vykonávané v metóde `updateAddresses` implementované v kontrolére `ServerController`.

Katalóg v tejto implementácií neposkytuje žiadne informácie o dostupnosti ťažobných serverov na daných adresách a portoch. Potreba doplniť rozšírenie požadované zadaním tejto práce súvisí práve s týmto nedostatkom implementácie katalógu. Popis návrhu, implementácie a testovania rozšírenia je zachytený v samostatných kapitolách.

[1] http://http://smashed.fit.vutbr.cz/



Obr. 3.4: Užívateľské rozhranie nástroja `sMaSheD` zobrazujúce panel `Servers` – prihlásený užívateľ.

Kapitola 4

Návrh rozšírenia

Súčasťou tejto práce je rozšírenie existujúceho nástroja. Obsahom tejto kapitoly je popis návrhu modifikácií, ktoré vedú k doplneniu požadovanej funkcionality.

Jednou z požiadaviek rozšírenia je pridanie informácií o časovej rovine, ktoré by viedlo k vyššej spoľahlivosti uložených metadát. Do aktuálneho návrhu nástroja je potrebné pridať funkcionalitu, vďaka ktorej bude katalóg poskytovať informácie o skutočnom stave ťažobných serverov. Samotný návrh tohto rozšírenia je popísaný v bode 4.1.

Ďalšia časť tejto kapitoly je venovaná popisu ostatných vylepšení (viď 4.2), ktoré budú do katalógu doplnené. V bode 4.3 je zhrnutý pohľad na problematiku overlay sietí a kontext, v ktorom je tento pojem chápaný vo vypracovaní tejto práce.

4.1 Časová rovina

K tomu, aby bol nástroj schopný poskytovať informácie o časovej rovine je potrebné vykonať niekoľko modifikácií.

Aktuálna implementácia katalógu získava zoznam IP adries z uloženého doménového mena. Našou úlohou je zistiť dostupnosť/stav servera na danej adrese a porte. V momente kedy bude nástroj schopný získavať takéto informácie, časovú rovinu vytvoríme tak, že stav servera bude overovaný pravidelne a získané výsledky budú určitým spôsobom zaznamenávané. Návrh spôsobu overovania stavu servera je bližšie popísaný bode 4.1.1.

Z uvedeného vypláva potreba vykonávania pravidelných dotazov na dostupnosť serverov patriacich do uložených ťažobných poolov. Výsledky týchto dotazov je potrebné ukladať. S tým súvisí potreba modifikácie databázy popísaná v bode 4.1.2.

4.1.1 Stav servera

Vykonanie testu dostupnosti vyžaduje znalosť dvoch základných údajov – IP adresa a port. To môžeme označiť ako ťažobný parameter (záznam v tabuľke). Pre všetky takéto záznamy bude potrebné zhotoviť a odoslať správu v špecifickom formáte. Na základe získanej odpovede na túto správu prebehne klasifikácia dostupnosti služby. Službu chápeme ako spustenú inštanciu ťažobného protokolu (Stratum, Getblocktemplate, Getwork) na parametroch testovaného záznamu (IP, port).

Obsah zasielaných správ sa môže líšiť v závislosti od protokolu, nad ktorým prebieha test. Protokol môže taktiež podporovať rozdielne JSON-RPC metódy, prostredníctvom ktorých komunikuje. Správa má v zjednodušenej podobe sémantiku žiadosti o prácu (ťaženie kryptomeny). V prípade, že od servera získame validnú odpoveď, služba je považovaná za

aktívnu. Odpoveď na každú metódu je však potrebné analyzovať individuálne. V niektorých prípadoch totiž navrátenie chyby nemusí znamenať, že na serveri nie je aktívna implementácia ťažobného servera (napr. metóda `login` s fiktívnymi prihlasovacími údajmi vráti chybu). Ku výsledkom testov bude doplnená informácia o protokole, časové razítko udalosti a stručný popis dosiahnutého stavu. V prípade negatívnej odpovede bude zaslaná ďalšia správa s využitím inej metódy. Použitie rozdielnych metód vyplýva z toho, že každá kryptomena môže podporovať rôzne metódy a implementovať tak rozdielne JSON-RPC API. Použitý protokol a jeho charakteristika však zostáva rovnaká alebo rozšírená o novú funkcionalitu.

Navrhnuté rozšírenie bude rozlišovať okrem základných stavov `UP` (server je dostupný) a `DOWN` (server je nedostupný), doplňujúci stav `LISTEN`. Tento stav vyjadruje, že server na daných parametroch počúva, avšak nebolo možné klasifikovať protokol, ktorý používa. Typicky budú takto označované prázdne odpovede od servera, prípadne odpovede, s chybovou hláškou nepodporovanej RPC metódy apod.

V rámci implementácie nástroja bude potrebné zabezpečiť pravidelné vykonávanie popísaného testovania, tak aby bola zachytená dostupnosť serverov niekoľkokrát, v priebehu jedného dňa. Keďže sa jedná o webovú aplikáciu, je dôležité navrhnúť spracovávanie dotazov a odpovedí na server bez toho, aby boli negatívne obmedzený prípadný používatelia nástroja. Z toho dôvodu bude testovanie vykonávané na pozadí, tak aby dovoľovalo paralelné používanie aplikácie.

Protokoly, použité metódy a detaily implementácie sú bližšie popísané v kapitole Implementácia (viď 6).

4.1.2 Databáza

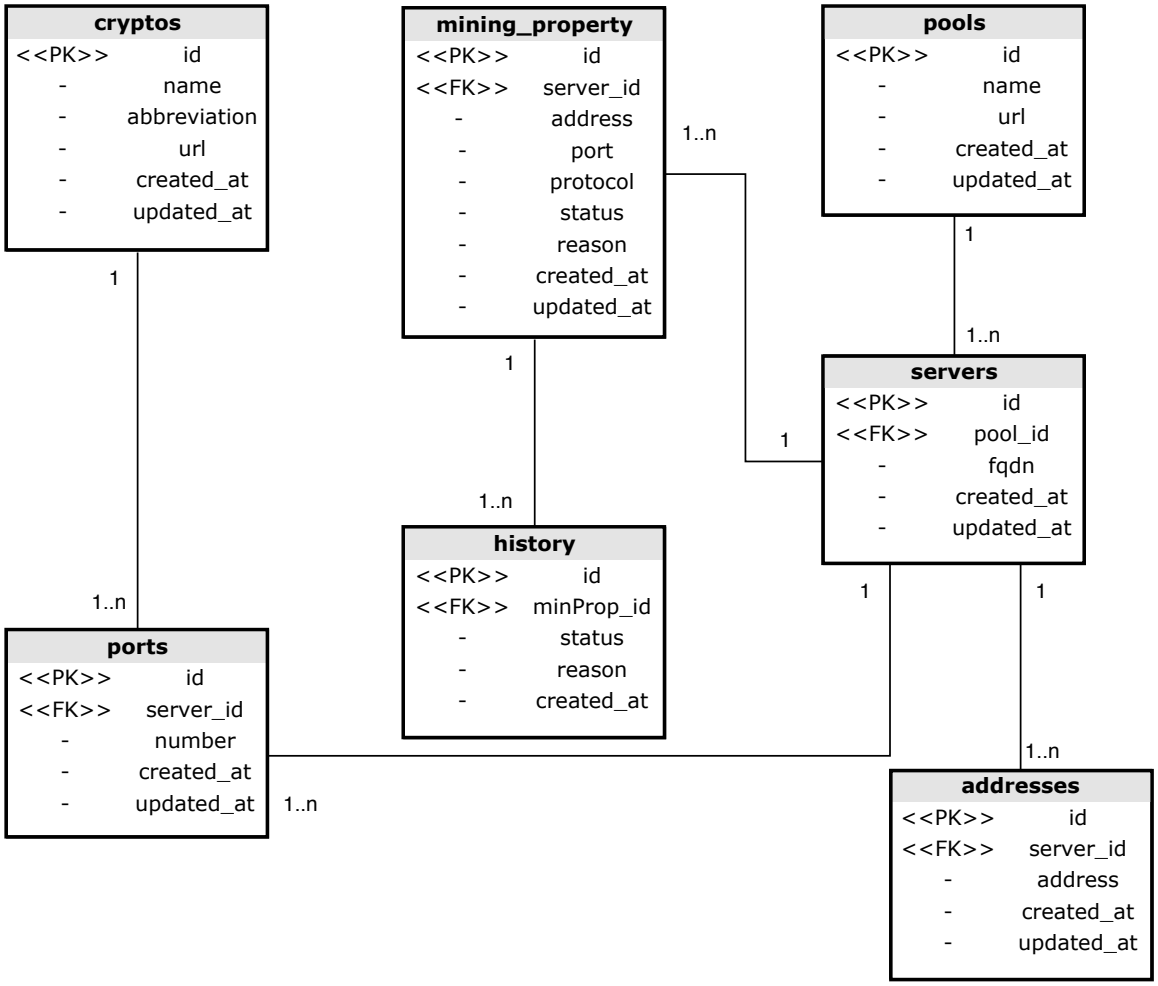
Doplnenie časovej roviny do uložených informácií vyžaduje modifikáciu databáze existujúceho nástroja. Prvá nevyhnutná zmena pozostáva z pridania novej tabuľky `miningProperties`, ktorá bude uchovávať záznamy o parametroch ťažby v relácií k spoločnému serveru. Obsahom tejto tabuľky je jednoznačný identifikátor záznamu, identifikátor servera (cudzí kľúč), IP adresa, port, protokol, stav, dôvod výsledného stavu a časové razítka.

Druhou potrebnou zmenou je doplnenie tabuľky `history`, do ktorej budú ukladané stavy záznamov po ich zmene/aktualizácií. Obsahom tabuľky je jednoznačný identifikátor, identifikátor `miningProperty` (cudzí kľúč), stav, dôvod a čas vytvorenia. Vďaka tejto tabuľke systém dokáže spravovať informácie o časovej rovine a poskytovať tak informácie o dostupnosti služby v určitých časových okamihoch. Návrh modifikovanej databáze vo forme ER diagramu je viditeľný na obrázku 4.1.

4.2 Ostatné rozšírenia

Nástroj sMaSheD bude doplnený o množstvo jednoduchých rozšírení, ktoré zvýšia užívateľský komfort. Údaje uložené v databáze sú zobrazené vo forme tabuliek. Nad tieto tabuľky bude doplnená možnosť filtrovania a zoraďovania dát. Taktiež bude pre lepšiu orientáciu pridané stránkovanie (pagination).

Do ponuky webovej stránky bude doplnená položka `Mining Properties`, ktorej úlohou je zobrazovať údaje z tabuliek pridaných v rámci implementovaného rozšírenia. Po prihlásení užívateľa sa v rámci tejto ponuky zobrazí tiež možnosť pre manuálne spustenie testovania dostupnosti uložených záznamov. Sprístupní sa taktiež zobrazenie detailu konkrétneho záznamu, kde podobne ako v ostatných položkách stránky, budú viditeľné informácie uložené



Obr. 4.1: ER diagram modifikovanej databáze.

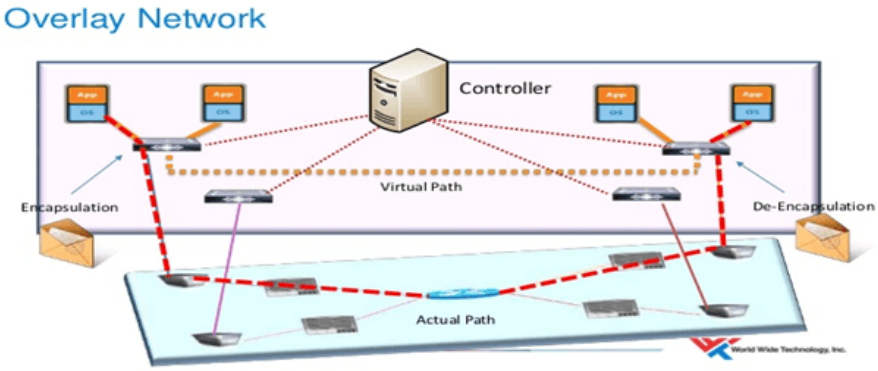
v databáze. Tento detail bude zahŕňať aj zobrazenie dostupnosti servera v čase (históriu stavov).

Aktuálna implementácia systému neposkytuje možnosť exportu uložených dát. Z toho dôvodu bude doplnená funkcionalita, ktorá vráti uložené informácie vo forme JSON objektu. Táto funkcia bude dostupná dostupná na adrese `/json` pre ponuky `Pools, Cryptos, Servers, Ports, Addresses, Mining Properties`.

4.3 Overlay sieť

Ide o druh počítačovej siete postavený nad inou sieťou. Pre lepšiu predstavu môžno uviesť príklad kedy v období začiatkoch internetu bol práve internet postavený ako overlay sieť nad telekomunikačnou sieťou. Dnes je internet postavený prevažne nad sieťou digitálnou a situácia je opačná. Telefónne služby sú zabezpečované prostredníctvom protokolu IP telefonie (VoIP), ktory tvorí overlay sieť nad internetom. V podstate ide o virtuálne/logické prepojenie sieťových uzlov prostredníctvom špecifického protokolu. Výmena dát medzi vzdialenými uzlami ale prebieha cez sieťovú infraštruktúru nad ktorou je overlay sieť postavená. Takáto

situácia je znázornená na obrázku 4.2. Použitie overlay sietí je tiež rozšírené v oblasti virtualizácie.



Obr. 4.2: Grafické znázornenie overlay siete (prevzaté z [27]).

V rámci sieťového modelu TCP/IP existuje mnoho ďalších overlay protokolov. Ako príklad možno uviesť niektoré z nich:

- HTTP/HTTPS
- XMPP (Jaber)
- Peer-to-peer siete (napr. Gnutella)
- overlay VPN (IPsec, GRE, DOVE)
- anonymizačné siete (I2P, Tor)

Zhodnotenie výhod takéhoto riešenia je závislé od technológie, ktorá ho využíva. Často však súvisí so znížením nákladov, s izoláciou od internetovej siete apod.

Za všeobecné nevýhody overlay sietí možno považovať vysokú réžiu pri spracovávaní sieťovými prvkami (horšia latencia), problematický load balancing a prípadnú konfiguráciu firewallov.

V kontexte tejto práce sa budeme zameriavať na anonymizačnú overlay sieť s názvom `Tor`. Bližší popis a spôsob konfigurácie tejto siete je uvedený v kapitole 5. Cieľom zabezpečenia dostupnosti nástroja na tejto sieti je zhromažďovanie informácií o tom, či ho užívatelia anonymizačnej používajú.

Kapitola 5

Tor – overlay (anonymity) network for TCP

Kapitola teoreticky popisuje anonimizačnú sieť Tor. Zachytáva základnú myšlienku a koncept tejto architektúry. Taktiež diskutuje prínos použitia tejto siete a poskytuje jednoduché „know-how“, pomocou ktorého je možné pripojenie a sprístupnenie servera v sieti Tor.

5.1 Motivácia

Sieť Tor je poskytovaná skupinou dobrovoľníkov, ktorý zastávajú názory komunity. Táto sieť zabezpečuje anonymitu a bezpečnosť prístupu na internet. Pripojenie je realizované cez niekoľko virtuálnych spojení na rozdiel od priameho pripojenia, pri ktorom je odhalená identita používateľa. Vďaka službám tejto siete je taktiež možný prístup k obsahu, ktorý je z nejakého dôvodu zablokovaný v rámci internej siete, alebo poskytovateľom internetu.

Vďaka službe Tor je možné užívateľa úplne odtieniť od rôznych analýz sieťovej prevádzky. Zabraňuje teda zisteniu toho, že kto a s kým komunikoval prostredníctvom verejnej siete, aký obsah prehliadal apod. Z časti je možné tento problém riešiť použitím šifrovania dát. V takomto prípade informácie v hlavičkách správ zostávajú v otvorenej podobe. Iným riešením môže byť použitie VPN. Všeobecne však ide o platenú službu a stále je potrebné dôverovať tretej strane, poskytovateľovi VPN.

Cieľom služby Tor (The onion routing) je poskytnúť dostupné riešenie odstraňujúce vyššie popísané problémy.

5.2 Architektúra a princíp

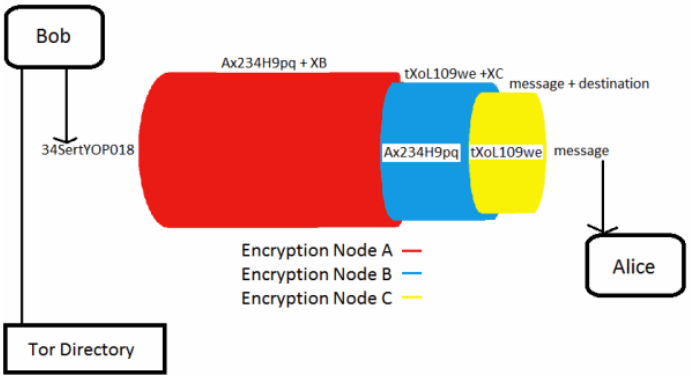
Základom sú uzly, ktoré prevádzkujú Tor, tzv. `Relay`. Pri vytvorení spojenia prostredníctvom tejto siete, je prevádzka vždy preposielaná minimálne cez tri `Relay` uzly. Na základe požiadaviek na tieto uzly a funcinalitu, ktorú pokrývajú, je možné ich rozdeliť do troch skupín:

- stredové/guard – uzly, cez ktoré prevádzka neopúšťa sieť Tor
- východiskové/exit – ide o posledný uzol v rámci spojenia; tento uzol preposiela komunikáciu na adresu destinácie, pričom na cieľovej destinácií je považovaná IP adresa tohto uzlu za adresu zdroja (iniciátora komunikácie)

- bridge – ide o uzly, ktorých IP adresy nie sú verejne dostupné v zozname serverov; použiteľné v prípade, že je Tor zakázaný v krajine, organizácií, ISP

Každá kategória má špecifické požiadavky na priepustnosť siete, veľkosť pamäte a výkon CPU. Detailné informácie o konfigurácií takýchto uzlov sú uvedené v dokumentácií[1].

Vytvorenie spojenia s cieľovou destináciou prebieha tak, že sa v prvom kroku vykoná dotaz na adresárový server Tor-u, kde sú uložené IP adresy relay serverov. Následne sú náhodne zvolené tri (najmenej) z nich. To vytvára náhodnú cestu v tejto sieti. Každý z týchto serverov využíva asymetrickú kryptografiu a vlastní verejný a súkromný kľuč. Pred odoslaním dát sú ku nim „prilepené“ vrstvy, ktoré obsahujú IP adresy nasledujúceho servera, cez ktoré majú byť dáta zasielané. Podstatné je, že každá z týchto vrstiev je zašifrovaná verejným kľúčom toho servera, z ktorého sa bude správa posielať ďalej. Vďaka tomuto prístupu vznikne správa pozostávajúca z niekoľkých vrstiev, ktoré pripomínajú cibuľu. Názov Onoin network/routing má pôvod práve v tejto vlastnosti. Celý proces šifrovania a komunikácie ilustruje obrázok 5.1.



Obr. 5.1: „Cibuľová“ architektúra siete Tor (prevzaté z [41]).

Správa je následne zaslaná na adresu vstupného uzlu (červená), s tým že obsahuje niekoľko vrstiev šifrovania. Po prijatí správy vstupným uzlom siete Tor, tento dešifruje správu svojím súkromným kľúčom a získa IP adresu, na ktorú má byť správa ďalej zaslaná. Ostatný obsah správy je pre neho nečitateľný, pretože je zašifrovaný verejným kľúčom nasledujúceho servera. Takýmto spôsobom je správa preposlaná až ku výstupnému uzlu (žltá), ktorý po dešifrovaní získa skutočnú adresu cieľa. Na túto adresu následne odošle pôvodnú správu klienta. Dôležité je zmieniť, že ak sú dáta posielané v otvorenej podobe (bez použitia šifrovania), výstupný server ich má k dispozícií a je schopný ku nim pristupovať.

5.3 Webový server

V tejto sekcií je popísaná inštalácia Tor klienta a sprístupnenie webového servera pre operačný systém `Ubuntu`.

Pred konfiguráciou webového servera je potrebné mať správne nainštalované a nakonfigurované služby Tor. Pre bežného užívateľa je najvhodnejšia inštalácia balíka Tor Browser[2], ktorý obsahuje korektnú konfiguráciu pre zabezpečenie anonymity a prístup do siete Tor.

[1] https://trac.torproject.org/projects/tor/wiki/TorRelayGuide
[2] https://www.torproject.org/download/download-easy.html.en

V našom prípade je vhodné použiť nasledovný postup:

1. Inštalácia Tor klienta
2. Konfigurácia webového servera
3. Konfigurácia Tor klienta

Inštalácia

Inštalácia je závislá na knižnici `libevent`, ktorú je potrebné mať k dispozícií. Tor klient je možné skompilovať a nainštalovať so zdrojových súborov alebo vykonať inštaláciu prostredníctvom balíčkového systému `APT - Advanced Packaging Tool` zadaním príkazu `apt-get install tor` s oprávnením root. Pre operačný sytém `Ubuntu` sa nedoporučuje použitie balíčkov, pretože nemusia obsahovať najaktuálnejšiu verziu programu. V rámci riešenia tejto práce bol preto použitý postup, ktorý pozostával s pridania nového repozitára do súboru `/etc/apt/sources`. Pre našu verziu operačného systému (Ubuntu 16.04) mali pridané záznamy nasledujúci formát:

```
deb https://deb.torproject.org/torproject.org xenial main
deb-src https://deb.torproject.org/torproject.org xenial main
```

Pridanie kľúča podpisujúceho pridané balíky možno vykonať zadaníkm nasledujúcich príkazov do terminálu:

```
gpg --keyserver keys.gnupg.net --recv
    A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89
gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | sudo apt-key
    add -
```

Po vykonaní predchádzajúcich krokov je potrebné vykonať aktualizáciu balíčkov a spustiť inštaláciu príkazmi

```
apt-get update
apt-get install tor deb.torproject.org-keyring
```

Konfigurácia servera

Vzhľadom k tomu, že webový server (Apache) bol v minulosti spustený a nakonfigurovaný, nie je vyžadovaný žiaden ďalší zásah do jeho konfigurácie.

Z hľadiska zachovania anonymity je však potrebné nastaviť server tak, aby bol dostupný iba na adrese `localhost` (127.0.0.1). Taktiež treba prispôsobiť celkové nastavenie servera nech neposkytuje žiadne informácie o pôvode, lokácií, doméne, atď. V rámci tejto práce nebolo požadované zachovanie anonymity. Požiadavkou bolo zabezpečenie dostupnosti nástroja v rámci overlay siete Tor.

5.3.1 Konfigurácia Tor

Konfiguračný súbor služby Tor sa v predvolenom nastavení nachádza v umiestnení `/etc/tor/torrc`. Do tohto súboru je potrebné definovať cestu, do ktorej budú po spustení služby vytvorené nové súbory. Ďalším krokom je definícia parametrov servera pozostávajúceho z virtuálneho portu, pod ktorým bude server vystupovať v sieti Tor, IP adresy (localhost) a portu, na ktorom je webový server nakonfigurovaný.

Parametre definované do konfiguračného súboru nášho nástroja sú nasledovné:

```
HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 147.229.9.85:80
```

Pre propagáciu zmien konfigurácie je potrebné spustiť/reštartovať službu Tor (napr. `service tor restart`). Následne by mali byť vytvorené súbory `hostname` a `private_key` do priečinku `HiddenServiceDir`. V súbore `hostname` je uvedená adresa servera, pod ktorou vystupuje náš nástroj v overlay sieti. Nástroj je dostupný na adrese `mbmqpamqch27cvjy.onion`, ktorú je možné zadať do prehliadača Tor Browser. V druhom vytvorenom súbore sa nachádza vygenerovaný súkromný kľúč, špecifický pre náš server.

Kapitola 6

Implementácia

Vzhľadom k použitým technológiám v existujúcej implementácií katalógu bola zachovaná rovnaká konvencia. Používal sa programovací jazyk PHP a framework Laravel. Implementáciu rozšírenia do katalógu môžeme rozdeliť do niekoľkých samostatných celkov, na základe blokov súvisiacich s určitou časťou funkcionality.

6.1 MVC

V rámci návrhu boli popísané potrebné modifikácie databáze. Framework Laravel využíva koncept MVC[1] a objektové relačné mapovanie (ORM) `Eloquent`[2]. Do nástroja boli pridané nové modely `MiningProp` a `History`. Na základe schémy databáze je v týchto modeloch definované členstvo a názov tabuľky, ktorú model popisuje.

Modifikácia databáze bola vykonaná pomocou migrácií[3], v ktorých boli definované tabuľky podľa databázovej schémy.

Do frontendu boli definované súbory, uchovávajúce zdrojové HTML kódy obsluhujúce pridaný prvok stránky – Mining Properties. Cesta k týmto súborom, z koreňového adresára aplikácie je nasledovná `./resources/views/miningProperties.php`.

Sprostredkovanie dát medzi vrstvami Model a View zabezpečuje nový kontrolér `MiningPropController`. V triede tohto kontroléra sú definované nasledujúce funkcie:

- `_construct` – konštruktor objetku
- `index` – zobrazenie stránky „Index“
- `show` – detailný náhľad na konkrétny záznam
- `refresh` – obsluha tlačidla pre obnovenie stavu serverov
- `clear` – obsluha tlačidla/URL adresy, ktorá zmaže históriu a aktuálny stav všetkých záznamov
- `historyClear` – obsluha URL adresy pre vyčistenie histórie zobrazeného záznamu
- `json` – obsluha požiadavku na JSON objekt záznamov tabuľky miningProperties
- `jsonHistory` – obsluha požiadavku na JSON objekt histórie zobrazeného záznamu

[1] Popis: https://doc.nette.org/cs/2.4/presenters
[2] Popis: https://laravel.com/docs/5.6/eloquent
[3] Popis: https://laravel.com/docs/5.6/migrations

- `jsonHistoryAll` – obsluha požiadavku na JSON objekt záznamov tabuľky history

Definícia konštruktoru obsahuje objekt `service`, ktorý je inštanciou triedy `RefreshService`. Väčšina logiky súvisiacej s implementovaným rozšírením nástroja sa nachádza práve v tejto triede. Detailný popis metód je uvedený v príslušnej časti (viď 6.2.2). Implementácia triedy sa nachádza v súbore `./app/Services/RefreshService.php`.

Obsluhu obnovenia stavu záznamov tabuľky Mining Properties realizuje samostatná úloha[4]. Úloha je iniciovaná akciou užívateľa (kliknutie na tlačidlo) alebo plánovačom[5].

Pre zabezpečenie korektnej komunikácie medzi vrstvamy kontrolér $\Longleftrightarrow$ view boli pridané definície smerovania[6] do súboru `web.php` nachádzajúceho sa v priečinku `routes`.

Pridávanie nových entít do zdrojových kódov frameworku bolo vykonávané cez konzolový nástroj `Artisan`[7]. Jeho rozhranie poskytuje množstvo príkazov, ktoré zabezpečia pridanie požadovaného prvku. Vďaka tomu sú ošetrené všetky závislosti vyžadované pre korektnú integráciu pridávaného prvku (kontrolér, úoha, migrácia, atď) do aplikácie.

6.2 Testovanie dostupnosti ťažobných serverov

Vstupom pre spustenie testu je IP adresa a port servera. Tieto údaje sú dostupné v rámci záznamov tabuľky `miningProperites`. Celý proces je popísaný v nasledujúcom odstavci.

Požiadavka na obnovenie údajov o stavoch serverov (spustenie testovania) môže byť iniciovaná z dvoch zdrojov. A to od užívateľa alebo plánovača. V oboch prípadoch dôjde ku volaniu úlohy `RefreshMiningProperties`. Detaily implementácie tejto triedy a konfigurácia plánovača je popísaná v bode 6.3. Z tejto úlohy je následne pre každý záznam tabuľky `miningProperties` spustený test. Test pozostáva z vykonania dotazu na ťažobný server, klasifikácie odpovede a navrátenia výsledku. Formát dotazu a použité metódy sú závislé na použitom protokole. Ich bližší popis je uvedený v samostatnej časti 6.2.1. Po navrátení výsledku do úlohy je daný záznam aktualizovaný. Výsledok sa uloží aj do tabuľky `history`. Vrátený výsledok má formu asociatívneho poľa, indexovaného kľúčmi `reason` a `code`. Položka `reason` obsahuje stručný popis, dôvod, výsledného stavu. Ten pozostáva z textovej správy (väčšinou chybovej hlášky) a prípadne použitej metódy. Chybový kód je reprezentovaný celým číslom. Hodnoty, ktoré môže nadobúdať sú priradené stavom špecifikovaným v návrhu:

- 0 - DOWN
- 1 - UP
- 2 - LISTEN

Informácie uchované v databáze sú užívateľovi dostupné v grafickom rozhraní webovej aplikácie – ponuka `Mining Properties`. Na tejto stránke sú zobrazené ovládacie prvky – tlačidlá a možnosť zobraziť detail konkrétneho záznamu.

6.2.1 Zasielanie správ a použité metódy

Implementované rozšírenie zahŕňa dva protokoly – Getblocktemplate a Stratum. Z kapitoly 2 je zrejmé, že protokol Stratum pracuje nad vrstvou transportnou, zatiaľ čo Getblocktemplate

[4]Popis: https://laravel.com/docs/5.6/queues
[5]Popis: https://laravel.com/docs/5.6/scheduling
[6]Popis: https://laravel.com/docs/5.6/routing
[7]Popis: https://laravel.com/docs/5.6/artisan

využíva protokol HTTP aplikačnej vrstvy modelu TCP/IP. Keďže neoperujú na rovnakej sieťovej vrstve, implementácia zasielania správ bola rozdelená do dvoch metód.

Oba protokoly však vykonávajú volania prostredníctvom JSON-RPC. Formát správ má preto podobu JSON objektu. Do zdrojového kódu programu je správa definovaná ako reťazcový literál doplnený o znak nového riadku na konci:

```
'{"id": "id", "method": "metoda", "params": ["parametre"]}' . "\n";
```

Pre lepšiu prehľadnosť a rozšíriteľnosť je implementované rozšírenie rozdelené do menších celkov, funkcie triedy `RefreshService`. Jej popis sa nachádza v nasledujúcej podsekcií.

6.2.2 Trieda RefreshService

Táto trieda obsahuje implementáciu kľúčových funkcií, súvisiacich s doplnením rozšírenia pridávajúceho časovú rovinu do katalógu. Takto oddelená implementácia bola zvolená z toho dôvodu, aby sem bolo možné v budúcnosti jednoducho doplniť ďalšiu funkcionalitu dostupnú z viacerých prvkov frameworku. Najdôležitejšie metódy, ktoré riadia testovanie dostupnosti sú `miningProbeStratum` a `miningProbeGetblocktemplate`. Popis významných členských funkcií je obsiahnutý v podkapitolách s ich názvom.

miningProbeGetblocktemplate

Účelom tejto funkcie je vytvorenie JSON-RPC správy a jej následné odoslanie prostredníctvom protokolu HTTP. Metóda použitá v tejto správe má rovnaký názov ako ťažobný protokol, ktorého overenie je implementované v tejto funkcií. Odosielaná JSON-RPC správa má potom nasledovný tvar:

```
'{"id": 1, "method": "getblocktemplate", "params": [{"capabilities": ["
    coinbasetxn", "workid", "coinbase/append"]}]}'
```

Komunikáciu na úrovni HTTP zabezpečuje knižnica `cURL`[8]. Parametre odosielanej správy boli zvolené tak, aby formát zodpovedal komunikácií generovanej ťažobným softvérom počas aktívnej ťažby kryptomeny. Konfigurácia spojenia je zabezpečená funkciou `curl_setopt`. Po odoslaní správy je prijatá odpoveď konvertovaná do poľa, nad ktorým je vykonaná jej klasifikácia.

Klasifikácia pozostáva z kontroly obsahu poľa. V prípade, že je v ňom definovaný kľúč `error`, extrahuje sa chybová hláška a nastaví sa príslušný chybový kód. Ak sa v správe nachádza kľúč nenulovej hodnoty, test je považovaný za úspešný. Výsledkom je stav `UP`. Pri spracovaní správy knižnicou `cURL` sa môže vyskytnúť chyba. Takýto prípad je v riešení taktiež ošetrený. V rámci tohto bloku je možné odchytiť stav, kedy prijatá správa neobsahuje žiadne dáta. Prázdna odpoveď je klasifikovaná ako stav `LISTEN`.

Získaný výsledok je uložený v premennej `$result`, ktorá je odovzdaná ako návratová hodnota funkcie.

miningProbeStratum

Táto funkcia má rovnaký účel ako v predchádzajúcom prípade (viď 6.2.2). Zameriava sa ale na protokol `Stratum`. Nakoľko ide o najrozšírenejší protokol, existujú jeho rôzne implementácie. Z toho dôvodu bolo potrebné tento fakt zohľadniť a implementovať možnosť testovania

[8]Popis: http://php.net/manual/en/book.curl.php

prostredníctvom viacerých JSON-RPC metód. V aktuálnom riešení sú implementované nasledujúce metódy protokolu:

- `mining.subscribe`
- `mining.authorize`
- `login`
- `eth.getwork`
- `eth_submitLogin`

V budúcnosti je do riešenia možné jednoducho doplniť ďalšie metódy podľa potreby.

Funkcia vykonáva zasielanie uvedených správ nad protokolom `TCP`. Komunikácia je implementovaná na úrovni sieťových socketov. Socket je vytvorený volaním PHP funkcie `stram_socket_client`. Odoslanie správy je vykonávané v pomocnej funkcií `sentTcpRequest`, volaním PHP funkcie `stram_socket_sendto`. Prijatá odpoveď je následne spracovaná a klasifikovaná.

Spracovanie odpovede prebieha v samostatnej funkcií `parse_JSON`, ktorej úlohou identifikácia JSON objektov v prijatej správe a ich následná konverzia do asociatívneho poľa, z ktorým sa bude lepšie pracovať pri klasifikácií odpovede. Pri dotazoch na niektoré servery dochádzalo k tomu, že server odpovedal vo formáte HTTP. Vo funkcií `parse_JSON` je pre takýto prípad implementované ošetrenie, ktoré vráti obsah vo formáte, očakávanom pri následnej klasifikácií správy. Tá je vykonaná vo funkcií `checkResponse`.

Výsledok, získaný z volania fincie `sentTcpRequest`, je následne konkatenovaný do reťazca `$resultReason` vo formáte `method => reason (return code);`.

V momente keď sa test skončil úspešne (stav UP) do premennej `$result` sa vloží návratový kód a obsah `$resultReason`. Ak je výsledok negatívny pokračuje sa zaslaním dotazu obsahujúceho ďalšiu metódu. Návratové informácie zo všetkých testov sú zaznamenávané a zahrnuté do dôvodu konečného stavu. Ten je dosiahnutý v momente, kedy test skončí úspešne alebo sa vyskúšajú všetky definované správy. Za konečný stav je určený test:

1. ktorého návratový kód je 1 (OK)
2. z najvyššou hodnotou návratového kódu
3. poslednej metódy (správy) – ak je výsledkom predchádzajúcich testov kód 0

Celkový výsledok testu pre danú adresu a port je uložený v premennej `$result` a odovzdaný ako návratová hodnota funkcie.

checkResponse

Ide o pomocnú funkciu, ktorej úlohou je klasifikovať odpoveď na zaslaný dotaz. Klasifikácia prebieha na základe definovaných podmienok, ktoré boli zostavené podľa analýzy všetkých možných odpovedí na implementované metódy.

Podmienky sú postavené na obsahu určitých vlastností (ukazateľov) v získanej odpovedi. Spracovanie odpovede má význam, ak sa v nej nachádzajú nejaké dáta. Tieto dáta môžu všeobecne pozostávať z niekoľkých JSON správ, ktoré sú uložené ako položky poľa. Každá položka uchováva svoje JSON objekty v podobe asociatívneho poľa. Tento formát je dosiahnutý spracovaním prijatých dát vo formáte reťazca funkciou `parse_JSON`. V bloku

tejto funkcie sú správy dostupné v premennej s názvom `$parsed`, ktorá je odovzdaná ako parameter. Každá správa je potom spracovaná individuálne.

Prvým ukazateľom je kľúč `error`, kde nás zaujíma jeho hodnota. Ak je nastavená na `null`, znamená to, že nedošlo k chybe (komplementárne je vhodné overiť kľúč `result`; popis zahrnutý v nasledujúcom odstavci). V opačnom prípade je obsahom kľúča `error` chybová hláška s možným návratovým kódom. Chybové hlášky všetkých správ sú ukladané a na konci zahrnuté v návratovej hodnote. Niektoré chyby však nemusia znamenať ne-aktívnosť ťažobného servera tohto protokolu. Konkrétny príklad vyskytujúci sa v implementovanom riešení je odpoveď na metódu `login`. Užívateľské parametre definované v tejto správe sú fiktívne, v dôsledku čoho ich server nie je schopný priradiť skutočnému užívateľovi. Preto server typicky reaguje zaslaním takejto chybovej hlášky v odpovedi – `invalid address used for login`. Tento fakt je potrebné zohľadniť a kontrolovať obsah chybových správ tak, aby nedochádzalo k reportovaniu „false positives“. Pri prípadnom rozšírení nástroja o ďalšie kryptomeny, používajúce nové metódy je vhodné analyzovať odpovede na doplnené metódy a na základe toho pravdepodobné chybové stavy ošetriť.

Za druhú vlastnosť možno považovať položku `result`. Podobne ako v predchádzajúcom prípadennás zaujíma jeho hodnota. V prípade, že je rozdielna od `null`, odpoveď servera na zaslanú správu je možné považovať za pozitívnu. Môže pozostávať z niekoľkých správ, ktoré poskytujú napríklad parametre potrebné pre ťažbu meny, pripojenie na server a podobne. S tým súvisí ďalší ukazateľ, podľa ktorého je správa klasifikovaná. V správe je hľadaný kľúč `method`, ktorého hodnota špecifikuje RPC metódu server → klient. Aktuálna implementácia identifikuje tri metódy, ktoré sú považované za validné. Sú to `mining.set_difficulty`, `mining.notify` a `mining.set_target`. Ak server odpovie neznámou metódou, takáto odpoveď je klasifikovaná kódom 2, doplnená o správu/dôvod `Unknown method!`.

Stavy získané klasifikáciou všetkých prijatých objektov sú spočítané. Podľa počtu chybných a validných výsledkov je striktne určený výstupný stav testovanej metódy (kódy 0, 1, 2). Ak dôjde ku inkonzistencii medzi chybovými a validnými stavmi, test metódy je označený kódom 2.

addHistoryRecord/addMiningPropRecord

Ide o pomocné metódy, ktorých účelom je vložiť nový záznam do príslušnej tabuľky. Všetky potrebné informácie sú odovzdané v parametroch funkcie. V prípade že je vkladaný nový záznam, časové razítko udalosti je získané z triedy `Carbon` volaním `Carbon::Now`.

6.3 Úloha a plánovanie

Spustenie testov je realizované cez volanie úlohy, ktorá je spustená na pozadí. Táto úloha sa ďalej stará o priebeh obnovenia údajov. Zdrojový súbor úlohy bol vytvorená za pomoci konzolového príkazu frameforku `php artisan make job RefreshMiningProperties`. Tento príkaz vytvoril zdrojový súbor `./app/jobs/RefreshMiningProperties.php`. Úlohu je možné spustiť napríklad z kontroléru volaním `RefreshMiningProperties::dispatch()`.

Vykonanie úlohy môže spracovávať niekoľko dostupných „driver-ov“ obsluhujúcich fronty úloh. V rámci riešenia bola pôvodná konfigurácia využívajúca `driver sync` nahradená novou – `database`. Tá je uložená v súbore `config/queue.php`. Pôvodný driver (ovládač) je vhodné používať iba na ladiace účely, pretože pri vykonávaní úlohy využíva rovnaké výpočtové vlákno ako celá aplikácia. Nová konfigurácia ovládača deleguje úlohy na nezávislé vlákno, čím nedochádza k obmedzeniu paralelnej použiteľnosti aplikácie. Ovládač `database`

využíva k ukladaniu úloh databázu. Preto bolo potrebné vytvoriť nové tabuľky – `jobs` a `failed_jobs`. Ich definícia do databáze bola realizovaná prostredníctvom migrácií.

Po zavolaní úlohy je do tabuľky `jobs` pridaný nový záznam obsahujúci informácie potrebné ku ich spracovaniu. Spracovanie úloh vyžaduje spustenie tzv. `worker-a`. V rámci serveru, na ktorom beží aplikácia bol worker definovaný ako proces bežiaci na pozadí. Spustenie je realizované príkazom `nohup php artisan queue:work -tries 3 > storage/logs/queue.log &` s parametrom vyžadujúcim počet pokusov, ktoré sa pokúsi vykonať v prípade chyby. Štandardný výstup procesu je presmerovaný do súboru `storage/logs/queue.log`. Ak dôjde reštartu operačného systému, je potrebné zabezpečiť, aby bol tento proces znovu spustený.

Samotná implementácia úlohy vykonáva volania „miningProbe“ metód z triedy `RefreshService` nad záznamami tabuľky `miningProperties`. Získané výsledky následne aktualizuje v databáze. Konkrétne pridaním záznamu do tabuľky `history` a aktualizáciou (pridaním, ak neexistuje) záznamu v tabuľke `miningProperties`.

Plánovanie

Obnovenie stavu záznamov tabuľky `miningProperties` je podľa návrhu potrebné vykonávať v pravidelných časových intervaloch. Plánovanie úloh a automatizáciu určitých činností v operačnom systéme typu Unix zabezpečuje program `Cron`. Framework Laravel dovoľuje spoluprácu s týmto nástrojom. Pre jeho používanie je potrebné definovať nasledujúci záznam do konfiguračnej tabuľky `Cron-u` (crontab), na našom serveri:

```
* * * * * php /home/smashed/Sites/mining/artisan schedule:run >> /dev/
    null 2>&1
```

Vďaka tomu môžeme v súbore `./app/Console/Kernel.php`, do funkcie `Schedule` špecifikovať plánovanie úloh. Úloha vykonávajúca obnovenie stavu serverov je nastavená tak, aby sa spustila každé tri hodiny. Definícia plánovania v zdrojovom kóde je zabezpečená týmto riadkom:

```
$schedule->job(new \App\Jobs\RefreshMiningProperties)->cron('0 0-23/3 *
    * *');
```

6.4 Ostatné rozšírenia

Dáta uložené v databáze sú rozhraním aplikácie zobrazované vo forme tabuliek. Tieto tabuľky v pôvodnej implementácií nepodporovali možnosť filtrovania a vyhľadávania nad ich obsahom. V rámci rozšírenia nástroja bola doplnená táto funkcionalita, vďaka čomu je použitie nástroja pre užívateľa komfortnejšie. Nakoľko implementácia používa knižnicu `Bootstrap`, do šablóny frontendu (súbor `./resources/views/layout.blade.php`) bol doplnený štýl `bootstrap-table.min.css` a javascript súbory `bootstrap-table.min.js, bootstrap-table-filter-control.min.js`. Následne je v HTML kóde tabuľky možné použitie triedy `table table-striped` a rozšírenia `data-filter-control` poskytujúceho pridanú funkcionalitu. Uvedené zmeny zahŕňajú súbory `guestlist.blade.php` a `authlist.blade.php`, v priečinku `./resources/views`.

Vzhľadom k tomu, že nástroj vytváral katalóg informácií o ťažbe kryptomien, bola doplnená možnosť získania uložených dát ako JSON objekty. Tieto sú verejne dostupné na adrese `/json` z každej ponuky stánky. Pre verejnú dostupnosť (bez prihlásenia) bol

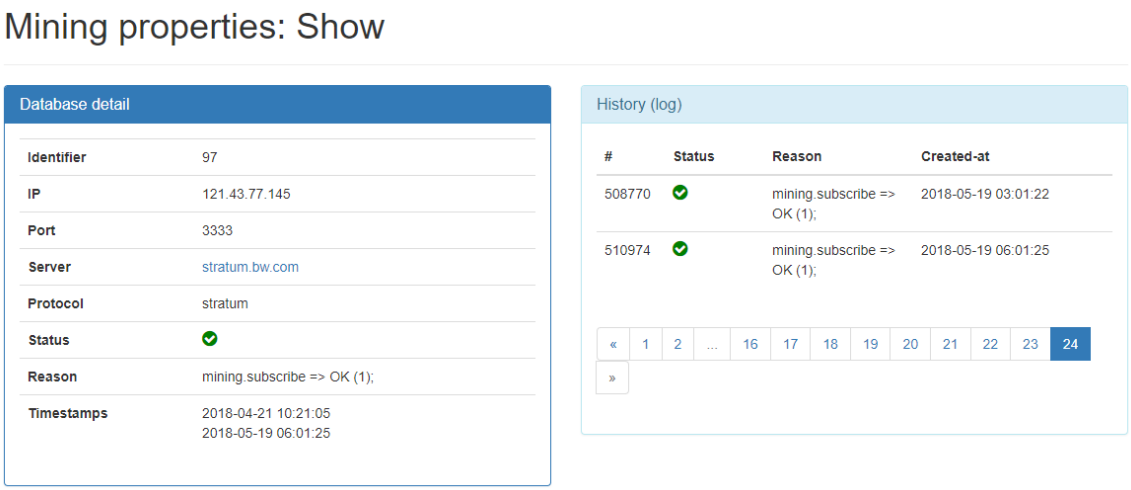
upravený konštruktor kontrolérov a pridaná funkcia `json`, ktorá navracia obsah tabuľky v požadovanom JSON formáte.

Popísané rozšírenia sa vzťahujú na prvky stránky `Pools`, `Cryptos`, `Servers`, `Ports`, `Addresses a Mining Properties`.

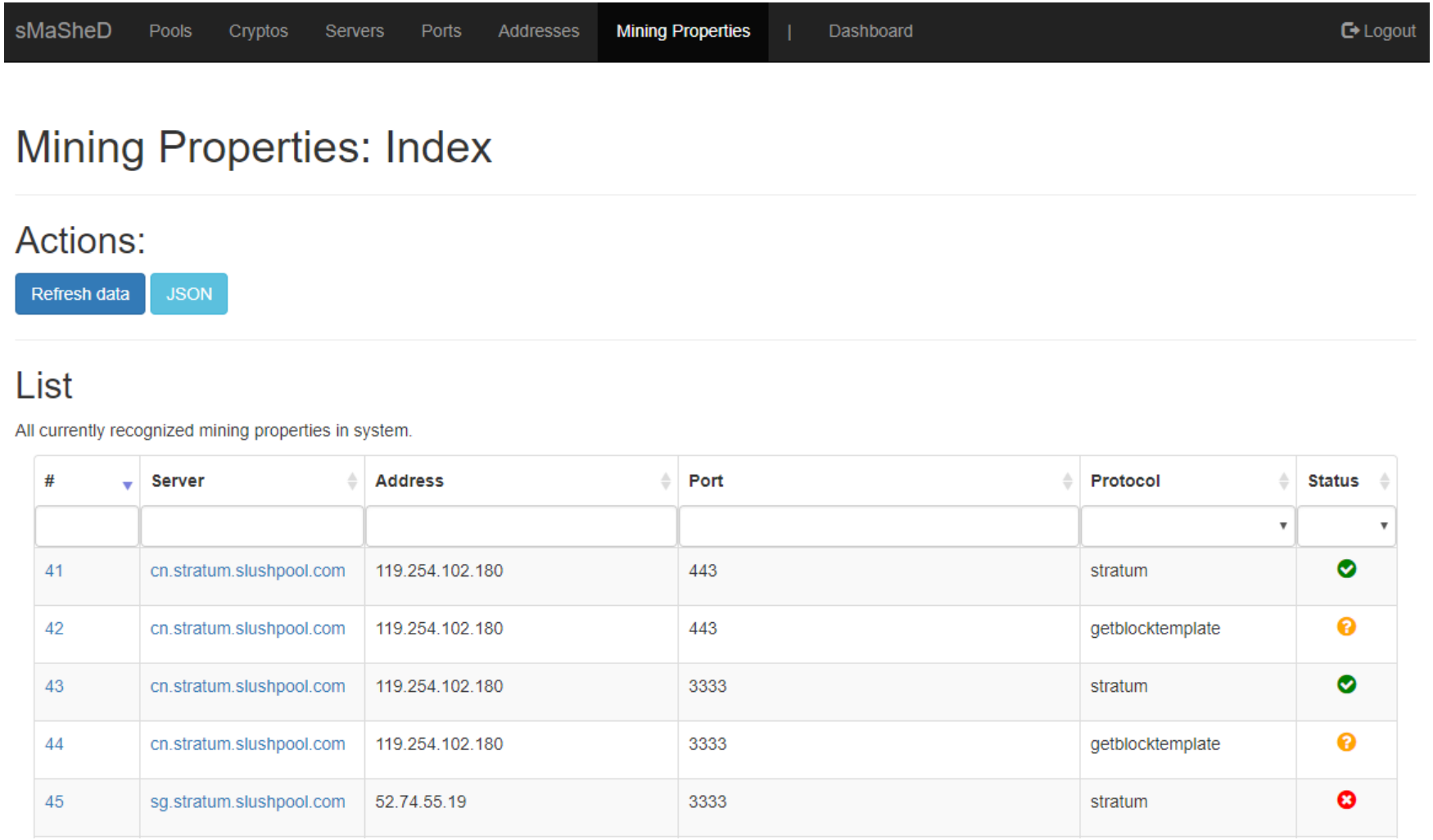
Do zobrazenia stránky `Mining Properties`, v administrátorskom režime, bola vložená sekcia `Actions`. V tejto sekcií sú užívateľovi zobrazené tlačidlá, ktorými môže spustiť obnovenie stavu záznamov a zobraziť stránku http://smashed.fit.vutbr.cz/miningProp/json obsahujúcu JSON objekt. Grafické rozhranie tejto stránky zobrazuje obrázok 6.2.

Účelom je zobrazenie dát z databázovej tabuľky `miningProperties`. To je formátované ako tabuľka s možnosťou zoradenia a filtrovania. Obsahuje základné informácie o záznamoch tejto tabuľky, s grafickým rozlíšením aktuálneho stavu. Červená farba je ekvivalentná so stavom DOWN, zelená s UP a žltá so stavom LISTEN popísaných v kapitole návrhu (viď 4).

Po kliknutí na identifikátor záznamu je zobrazený jeho detail. Ten poskytuje prehľad všetkých informácií s ním súvisiacich. Údaje zahŕňajú aj históriu stavov zvoleného prvku záznamu ako je možné vidieť na obrázku 6.1. Zobrazené dáta histórie je možné taktiež získať ako JSON objekt pridanám suffixu `/json` ku aktuálnej URL (napr. http://smashed.fit.vutbr.cz/miningProp/41/json). Úplný obsah tabuľky `history` je dostupný na adrese http://smashed.fit.vutbr.cz/miningProp/jsonHistoryAll.



Obr. 6.1: Detail jedného záznamu stránky Mining Property – prihlásený užívateľ.



Obr. 6.2: Grafické rozhranie stránky Mining Properties v nástroji sMaSheD – prihlásený užívateľ.

Kapitola 7

Zhrnutie a zhodnotenie výsledkov

Táto časť je zameraná na zhodnotenie výsledku práce. Diskutuje aktuálny obsah informácií uložených v katalógu. Taktiež sa zaoberá distribúciou výsledných stavov jednotlivých záznamov. Kapitola obsahuje aj návrh vylepšení nástroja, o ktoré môže byť v budúcnosti rozšírený.

7.1 Testovanie implementácie

Výsledné stavy ťažobných serverov boli porovnávané voči výsledkom získaných ťažobným softvérom. Pri testovaní implementácie bola použitá aplikácia `cgminer`[1]. Povinným parametrom aplikácie je adresa a port servera doplnená o užívateľské meno a heslo. Nástroj podporuje načítanie konfiguračného súbor, kde je možné špecifikovať vstupné parametre vo formáte JSON. Týmto spôsobom je možné definovať viacero serverov ku ktorým sa aplikácia pripája.

Testovanie rozšírenia prebiehalo nad aktuálnymi dátami tabuľky `miningProperties`. Z nich boli vytvorené konfiguračné súbory pre oba protokoly a každý stav servera. V tejto súvislosti bol vytvorený `wrapper.sh`, pomocou ktorého je vstupný `csv` súbor naformátovaný do tvaru JSON. Vstupný súbor pozostáva z riadkov `adresa,port`, ktoré boli exportované z databáze (vytvorené súbory sú obsahom príloh, viď A). Po spustení nástroja s daným konfiguračným súborom dochádza k pripájaniu sa na špecifikované servery s cieľom ťažiť kryptomenu. Činnosť nástroja je sprevádzaná textovým výstupom, ktorý obsahuje informácie o prebiehajúcej ťažbe (chybové hlášky, výkonnosť, apod.). Z výstupu je možné určiť dostupnosť/stav serverov a porovnať ho s našou aplikáciou. Časť výstupu aplikácie je demonštrovaná na obrázku 7.1. Výstup nerozlišuje dostupnosť servera od chybných užívateľských údajov. Je z neho však možné určiť či je server „online“. V špecifických prípadoch sú vypísané konkrétnejšie chybové hlášky.

Komunikácia generovaná počas behu tohto nástroja bola zaznamenávaná pomocou nástroja `Wireshark`[2]. Získané dáta boli analyzované so zámerom zistiť používané metódy a formát vymieňaných správ. Následne boli porovnané so správami z nástroja `sMaSheD`. Detail správ z oboch nástrojov je demonštrovaný na obrázkoch 7.2 a 7.3. V tomto príklade je vidieť, náš nástroj sa nepokúša autentifikovať voči serveru. Je to spôsobené tým, že v momente validnej odpovede (stav `UP`) nedochádza k testovaniu ďalších implementovaných metód. Naopak nástroj `cgminer` zasiela správu s metódu `mining.authorize`, v ktorej špecifikuje

[1] Popis: https://github.com/ckolivas/cgminer
[2] Popis: https://www.wireshark.org/

užívateľské údaje. Tie sú pri reálnej ťažbe väčšinou vyžadované. Z obrázkov je vidieť, že bez ohľadu na to sú v oboch prípadoch serverom poskytnuté parametre ťažby. Podobný detail komunikácie implementovaných metód protokolov je spolu z PCAP súbormi a konfiguračnými súbormi nástroja `cgminer` priložený v prílohe (viď A a C).

```
[2018-05-11 08:51:58] Pool 282 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Network diff set to 36.8K
[2018-05-11 08:51:58] Switching to pool 345 http://5.196.74.100:9171 - first
alive pool
[2018-05-11 08:51:58] Long-polling activated for http://5.196.74.100:9171/
long-polling
[2018-05-11 08:51:58] Pool 398 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Pool 391 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Pool 227 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Pool 229 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Pool 234 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Pool 170 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] LONGPOLL from pool 345 requested work restart
[2018-05-11 08:51:58] Pool 230 slow/down or URL or credentials invalid
[2018-05-11 08:51:58] Pool 256 slow/down or URL or credentials invalid
[2018-05-11 08:51:59] pool 322 JSON stratum auth failed: [
  201,
  "Cannot Found Minning Coin Type",
  null
]
[2018-05-11 08:52:01] Pool 115 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 112 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 120 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 121 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 128 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 126 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 125 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 124 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 127 slow/down or URL or credentials invalid
[2018-05-11 08:52:01] Pool 118 slow/down or URL or credentials invalid
...
```

Obr. 7.1: Výstup aplikácie `cgminer` – test záznamov Getblocktemplate v stave down.

```
{"id": 195, "method": "mining.subscribe", "params": ["cgminer/3.7.2", "0024b7cc"]}
{"result": [[["mining.notify", "0024b7cc"]], "8d337042", 4], "error": null, "id":
195}
{"id": 202, "method": "mining.authorize", "params": ["X", "Bitcoin"]}
{"params": ["00000000000074de",
"547eacc4e9476c71d05dc2e56745486d630191300017ea670000000000000000",
"0100000001000000000000000000000000000000000000000000000000000000000000ffffffff37
03bbf7070004823df55a0483a8032808",
"284d696e65642062792042572e434f4d290d2f425720506f6f6c2f4e59412fffffffff027208874c0000
00001976a91404b4f2f410aaee6c0aaeb3144f7eba05f315a6d088ac0000000000000000266a24aa21a9e
d3ff0615d2262aa8e245217aa98d49d99294b6eb435a5d6d68c3989219abe9b6f00000000",
["b0291c359f93e35d2c44198c2d7bd5c9b3a0a07436eea019d1b0c14ba5f47485",
"a583a33b8e47037826a2341cfcad1641b5431c14d5646146390139c0d9fbcc6b",
"7687969ac546e363cbb836522b82dd78ef5732c0103fddc1387279daa31be8c5",
"e43efa6ded5c220a884a42d669585bf42937c0c2306de9bd89c40274440e7120",
"4d7749d279f18cb3ce58231a85e3f0714b369cc50e6c70be3e31fa407fab7ae0",
"d86e772d5d72aadcf24b38b08e80c234964ac7df520a27456fd178f4f818b59c",
"a3d735f741b0bf89aae4c9e412a3287f423da6b92d2f547ce07f3a82f660d3d2",
"329a4c6435709844cf32f951e8a58faab82c626ec7ed7b26029787777106f1b6",
"024dabc14e4cc22cb0eaf3b8285c2147e708befb69d3bc7876981c3c69dd9354",
"7a2cb3939da5bff3bb67a303db4155c452b0c36b78536c9ca527ffaf55039448",
"40b73d3934274a6b9288fd14698b6fccb1625d87c512ae758a70fa7b2e24693d"], "20000000",
"1743eca9", "5af53d82", true], "id": null, "method": "mining.notify"}
{"id": null, "method": "mining.set_difficulty", "params": [8192]}
{"id": null, "method": "client.show_message", "params": ["Authorised, welcome X!"]}
{"result": true, "error": null, "id": 202}
```


Obr. 7.2: cgminer – správy protokolu Stratum medzi klientom (červená) a serverom (modrá) (server v stave `UP`).


```
{"jsonrpc": "2.0", "method": "mining.subscribe", "params": ["Miner 1.0"], "id": 1}
{"result": [[["mining.notify", "0024cda8"]], "a5d4f1dc", 4], "id": 1, "error": null}
{"params": ["000000000000751a",
"cdd11ef3fe81084ce97262c42cff4bcb5d8a744700298abf0000000000000000",
"0100000001000000000000000000000000000000000000000000000000000000000000ffffffff37
03c0f70700045948f55a04a58a571508",
"284d696e65642062792042572e434f4d290d2f425720506f6f6c2f4e59412fffffffff0269e7144b0000
00001976a91404b4f2f410aaee6c0aaeb3144f7eba05f315a6d088ac0000000000000000266a24aa21a9e
dedfeaad8dbabbba13c0ac97174cf3e0e23e4da581fa37c7e551841e85ab3055400000000",
["360d5382d35963a9b5991b514fde3e474e7810bc4631d016a94fc604eb4f75a5",
"f07ea5fb0142af785ec316fdb7abe8ca5d1156bcb3819150828f94d20aeb2cf8",
"5147f94c1e4302b1f47765260a034f7c05094562ca694b312e3abb11fda8f3b0",
"05c131e118e591851f66b10e0c1c0e7135ba8692cc52db9b633c743bba221dd2",
"93104db8ccc8b474a7d2a01b9e0645e23e322575c87b68632655ac89f746cb12",
"3776f8e21325d1eba9dd0de855b41b9145a4253f39ecc356217398a76d67ea5d",
"15b37753c27c09a49c1bb86bfaaefc5cddce24bbb0a78c9815f9a0daff1181e1",
"58b96672b1f91218ee3abbc9e6345fe7c01302a51eacfe5e25e938cb7f520ac1",
"c23fd17a7f784530940fa2143acbba6641ac766cce99d50b15ada40d07aa5eb0",
"3694f5ae14411f632eff59633697284a193f5f642e6561faf6acccc944ddb50b",
"24c4bb91b53c71ebbecc9c8976c9a2c059d99855851023caac6fa2ce279fa9ad"], "20000000",
"1743eca9", "5af54858", true], "id": null, "method": "mining.notify"}
{"id": null, "method": "mining.set_difficulty", "params": [5120]}
```


Obr. 7.3: sMaSheD – správy protokolu Stratum medzi klientom (červená) a serverom (modrá) (server v stave `UP`).

7.1.1 Dosiahnuté výsledky

Katalóg je naplnený množstvom dát súvisiacich s ťažobnými servermi kryptomien (viď 3.2). Porovnanie obsahu katalógu pred začiatkom práce a po jeho rozšírení je ukázaný v tabuľke 7.1. Došlo teda k výraznému rozšíreniu pôvodne uchovaných dát katalógu.

Tabuľka	Pred rozšírením	Po rozšírení	Nárast
Cryptos	12	13	8%
Pools	23	56	143%
Servers	46	184	300%
Ports	92	431	368%
Addresses	68	459	575%

Tabuľka 7.1: Objem dát uložený v katalógu – počet záznamov v príslušných tabuľkách.

Distribúciu jednotlivých stavov serverov, získané prostredníctvom implementovaného rozšírenia sú viditeľné v tabuľke 7.2. Z uvedených hodnôt vyplýva, že na väčšine adries uložených v katalógu beží aktívna implementácia ťažobného protokolu Stratum. Nástroj je schopný jednoznačne rozlíšiť stav `UP` a `DOWN`. V niektorých prípadoch ale môže dochádzať ku stavu, kedy server nepozná žiadnu z implementovaných metód, vyžaduje autentifikáciu apod. Tieto stavy sú klasifikované ako `LISTEN` a označujú uzol, ktorý možno prevádzkuje ťažobný server. Ďalej je vidieť, že žiaden záznam protokolu Getblocktemplate nie je klasifikovaný ako stav UP. Nakoľko ide o starší protokol, s horšou podporou ťažobného SW, tak je pravdepodobné že ťažobnými servermi nie je prevádzkovaný. Na základe zistení z testovania však možno konštatovať, že bez správnych užívateľských údajov tento protokol neposkytuje ťažobné informácie. Preto nebol s určitosťou žiaden takýto server označený za dostupný. Vzhľadom k počtu záznamov, nad ktorým je realizované testovanie, nebolo možné vytvoriť užívateľský profil pre všetky pooly.

Protokol	UP	LISTEN	DOWN
Stratum	576	226	300
Getblocktemplate	0	656	446
Getwork	-	-	-

Tabuľka 7.2: Zhrnutie aktuálnych stavov serverov pre jednotlivé protokoly.

Systém aktuálne obsahuje 2204 záznamov tabuľky `miningProperties`. Teda 1102 rozdielnych portov a adries, ktoré sú overované pre dva rozdielne protokoly. Časová náročnosť obnovenia všetkých údajov je v ráde desiatok minút. Pri aktuálnom počte záznamov je doba trvania 20 minút.

7.2 Budúce vylepšenia

Nástroj poskytuje priestor pre množstvo vylepšení, ktoré by do neho mohlo byť v budúcnosti integrované. Vzhľadom k zacieleniu nástroja by bolo zaujímavé jeho rozšírenie o generovanie filtrovacích pravidiel pre sieťové prvky. Ďalším prvkom nástroja by mohlo byť automatické spracovanie PCAP súborov, s cieľom nájsť zhodu s dátami uloženými v katalógu a identifikovať tak zdroj nežiadúcej činnosti v rámci internej siete.

Z hľadiska aktuálneho stavu nástroja by bolo vhodné zapracovať možnosť exkludovať určité záznamy z automatického testovania. Týka sa to predovšetkým adries, ktoré sú v stave

`DOWN` po určitú dobu. Pri nich je možné predpokladať, že ťažobný server na tejto konkrétnej adrese už prevádzkovaný nebude. Vďaka tomu by sa znížila sieťová prevádzka generovaná počas testovania a samotné testovanie by trvalo kratšiu dobu.

Aktuálne nastavenie nástroja vykonáva aktualizáciu stavov každé tri hodiny. Táto je sprevádzaná zápisom nových záznamov do tabuľky `history`. Keďže môžeme predpokladať budúci nárast objemu uchovaných dát, je potrebné zvážiť implementáciu „offloadingu“ týchto dát mimo webovú aplikáciu.

Pri praktickom používaní nástroja sa časom ukážu možnosti, ktoré aktuálne v riešení chýbajú. Uvedené návrhy rozšírení sú subjektívne a vychádzajú z aktuálneho pohľadu na nástroj riešiteľom tejto práce.

7.3 Zhrnutie

Do existujúceho riešenia bola úspešne implementovaná funkcionalita, ktorá pokrýva požiadavky zadania diplomovej práce. Hlavným cieľom bolo pridanie časovej roviny ku uloženým záznamom. Časovú rovinu reprezentuje implementované rozšírenie ako databázovú tabuľku `history`. V nej sú uchované informácie o dostupnosti ťažobného servera. Ťažobný server je v tomto kontexte chápaný ako dvojica vlastností – IP adresy a portu. Tieto vlastnosti náležia rovnakému serveru. Do implementovaného rozšírenia bola pridaná databázová tabuľka `miningProperties`, ktorá uchováva práve tieto vlastnosti. Nad uloženými servermi bolo implementované testovanie ich stavov. Nástroj vykonáva toto testovanie automaticky, v pravidelných troj-hodinových intervaloch. Testy overujú ťažobné protokoly Getblocktemplate a Stratum. Implementáciu a získané výsledky je treba chápať experimentálne, keďže nepokrýva všetky existujúce kryptomeny. Generické riešenie pridania takejto funkcionality nie je v tejto oblasti realizovateľné, pretože vzniká množstvo nových kryptomien a rôznych implementácií. V práci bola preto definovaná množina metód, ktoré nástroj podporuje.

Predmetom analýzy v úvodnej časti práce bol aj protokol Getwork. Jeho integrácia do implementovaného rozšírenia nebola vykonaná, pretože je v súčastnosti prakticky nepoužívaný. V prípade takejto požiadavky je možné ho do nástroja jednoducho doplniť (zakomentované časti zdrojového kódu).

Modifikáciou prešlo taktiež užívateľské rozhranie webovej aplikácie. Do hlavného panela stránky bola pridaná ponuka `Mining Properties`, ktorá zobrazuje údaje uložené v pridaných tabuľkách. Nad zobrazované údaje bola doplnená možnosť vyhľadávania, zoradenia a filtrovania dát. Používanie nástroja sa tým výrazne zjednodušilo a zefektívnilo.

Obsah všetkých dôležitých tabuliek v tomto nástroji je možné získať v podobe JSON objektu. Táto funkcionalita umožňuje použitie nástroja ako zdroja dát pre špecifické účely tretích strán. Uplatnenie katalógu smeruje na oblasť forenznej analýzy sieťovej prevádzky. Uložené informácie môžu byť užitočné pre rôzne organizácie, sieťových administrátorov, bezpečnostné zložky apod. Nástroj taktiež ocenia nadšenci do ťažby kryptomien, ktorým poskytuje prehľad o dostupnosti ťažobných poolov a parametre potrebné pre konfiguráciu ťažobného softvéru.

Kapitola 8

Záver

V rámci riešenia diplomovej práce prebehlo štúdium problematiky, princípov a ťažby kryptomien. Nadobudnuté poznatky boli spracované v príslušných častiach práce. Obsah sa zameriava predovšetkým na techniku miningu v pooloch, kde analyzuje komunikáciu prebiehajúcu medzi minerom a severom. Predmetom skúmania bol tiež formát vymieňaných metadát.

Práca je zameraná na rozšírenie existujúceho nástroja (katalógu), ktorého úlohou je uchovanie rôznych informácií o kryptomenách. V kapitole 3 je obsiahnutý popis tohto nástroja, jeho motivácia a aktuálna implementácia. Súčasťou vypracovania projektu bolo taktiež rozšírenie katalógu o nové dáta. Konkrétne boli pridávané hlavne ťažobné pooly, servery a porty ku vloženým kryptomenám. Vkladané informácie boli volené z ohľadom na distribúciu vyťažených blokov v sieti kryptomeny. Snahou bolo pokryť čo najväčšie množstvo takejto sieťovej prevádzky.

Na základe teoretických poznatkov nadobudnutých v predchádzajúcej časti bol vypracovaný návrh rozšírenia katalógu. Úprava mala do nástroja zaniesť časovú rovinu informácií. Pre zabezpečenie tejto požiadavky bolo potrebné vymyslieť spôsob, ktorým je možné ju realizovať. Navrhnuté riešenie pozostáva zo zasielania dotazov na konkrétnu adresu a port, ktorého obsahom je správa ťažobného protokolu. Odpoveď servera je následne klasifikovaná a definuje výsledný stav servera. Stavy servera sú chápané ako výsledky testu, ktoré môžu nadobúdať stav `UP`, `DOWN` a `LISTEN`. Takéto testovanie je potrebné vykonávať opakovane. Časová rovina je následne vybudovaná pomocou ukladania výsledkov testov, ktoré obsahujú časové razítko. Dáta sú uchovávané v databáze, na ktorú je nástroj napojený. Pôvodný návrh databáze bolo potrebné modifikovať, tak aby do nej bolo možné pridať požadované rozšírenie.

Existujúci nástroj je verejne dostupný na internete ako webová aplikácia[1]. Požiadavkou zadania je zabezpečenie jeho dostupnosti v rámci overlay sietí. Podľa pokynov vedúceho práce bola zvolená sieť Tor. Úspešnému pridaniu nástroja do tejto siete[2] predchádzala teoretická znalosť tejto oblasti. Nadobudnuté poznatky sú spracované v samostatnej kapitole (viď 5). Jej obsahom je tiež postup konfigurácie, potrebnej k sprístupneniu katalógu v tejto sieti.

Rozšírenia, špecifikované v časti návrhu (viď 4), boli do existujúceho riešenia implementované. V tejto časti bolo potrebné zoznámiť sa s architektúrou a implementáciou riešenia. Taktiež bolo potrebné pochopenie návrhového vzoru MVC a prvkov použitého

[1] http://smashed.fit.vutbr.cz
[2] mbmqpamqch27cvjy.onion

frameworku. Po zorientovaní sa v zdrojovom kóde bolo možné implementovať navrhnuté rozšírenie. Implementačné detaily sú popísané v kapitole 6.

Implementáciou rozšírenia sa podarilo do nástroja doplniť informácie o časovej rovine. V rámci toho boli pridané prvky do užívateľského rozhrania stránky, ktoré zabezpečia pohodlnejšie ovládanie aplikácie. Na základe vykonaného testovania je možné usúdiť, že aplikácia je schopná poskytovať informácie o skutočnom stave uložených ťažobných serverov. Vďaka pridanej funkcionalite je užívateľ schopný analyzovať stav konkrétneho servera v určitom časovom okamihu. Uložené informácie sú taktiež dostupné vo formáte JSON, na ktoré sa môžu dotazovať aplikácie tretích strán a spracovávať ich podľa svojich potrieb. Závery vykonaného testovania a stavu aplikácie po implementovanom rozšírení sú diskutované v kapitole 7.

Hlavným prínosom pridanej funkcionality je zlepšenie celkovej použiteľnosti nástroja. Budúce rozšírenia nástroja budú vyplývať až z jeho ďalšieho používania. Návrh možných rozšírení z pohľadu autora tejto práce je diskutovaný v príslušnej časti kapitoly.

Z tejto práce bol vypracovaný článok publikovaný v rámci konferencie Excel@FIT 2018. Vytvorené prezentačné materiály a samotný článok sú zahrnuté v prílohe.

Literatúra

[1] Agrawal, H.: *Explaining Hash Rate Or Hash Power In Cryptocurrencies*. CoinSutra - Community of Bitcoin Users, 2017, [Online; navštívené 27.12.2017].
URL https://coinsutra.com/hash-rate-or-hash-power/

[2] *APACHE HTTP SERVER PROJECT*. The Apache Software Foundation, 2018, [Online; navštívené 03.01.2018].
URL https://httpd.apache.org/

[3] *Bitcoin Developer Guide*. Bitcoin Project, 2017, [Online; navštívené 07.01.2018].
URL https://bitcoin.org/en/developer-guide#block-chain-overview

[4] *BitcoinCash - Peer-to-Peer Electronic Cash*. bitcoincash.org, [Online; navštívené 11.01.2018].
URL https://www.bitcoincash.org/

[5] *Getblocktemplate mining protocol*. bitcoinwiki, 2015, [Online; navštívené 11.11.2017].
URL https://en.bitcoin.it/wiki/Getblocktemplate

[6] *Bootstrap*. Bootstrap, 2018, [Online; navštívené 03.01.2018].
URL https://getbootstrap.com/

[7] Chohan, U. W.: *Cryptocurrencies: A Brief Thematic Review*. University of New South Wales (UNSW), 2017, [Online; navštívené 26.12.2017].
URL https://ssrn.com/abstract=3024330

[8] Courtois, N. T.; Grajek, M.; Naik, R.: The Unreasonable Fundamental Incertitudes Behind Bitcoin Mining. *CoRR*, ročník abs/1310.7935, 2013, 1310.7935.
URL http://arxiv.org/abs/1310.7935

[9] D'Herdt, J.; Carbone, R.: Detecting Crypto Currency Mining in Corporate Environments. *Gold Certification*, 2015.
URL https://www.sans.org/reading-room/whitepapers/threats/detecting-crypto-currency-mining-corporate-environments-35722

[10] Duffield, E.; Diaz, D.: *Dash: A Privacy-Centric Crypto-Currency*. GitHub, Inc., [Online; navštívené 11.01.2018].
URL https://github.com/dashpay/dash/wiki/Whitepaper

[11] *Ethereum Classic Documentation*. Ethereum classic community, [Online; navštívené 11.01.2018].
URL https://ethereum-classic-guide.readthedocs.io/en/latest/

[12] *F2pool – Help Center*. f2pool.com, 2018, [Online; navštívené 07.05.2018].
URL https://www.f2pool.com/help

[13] *Getwork mining protocol*. bitcoinwiki, 2015, [Online; navštívené 11.11.2017].
URL https://en.bitcoin.it/wiki/Getwork

[14] *Blockchain: could it transform your business model*. Grant Thornton International Ltd, 2018, [Online; navštívené 11.05.2018].
URL https://www.grantthornton.global/en/insights/articles/blockchain-could-it-transform-your-business-model/

[15] *Explaining Hash Rate Or Hash Power In Cryptocurrencies*. bitcoinwiki, 2017, [Online; navštívené 27.12.2017].
URL https://en.bitcoin.it/wiki/Pool_vs._solo_mining

[16] King, S.; Nadal, S.: *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake*. peercoin, [Online; navštívené 11.01.2018].
URL https://peercoin.net/assets/paper/peercoin-paper.pdf

[17] *Love beautiful code? We do too.* Laravel, 2018, [Online; navštívené 03.01.2018].
URL https://laravel.com/

[18] *List of cryptocurrencies*. Wikimedia Foundation, Inc., 2017, [Online; navštívené 26.12.2017].
URL https://en.wikipedia.org/wiki/List_of_cryptocurrencies

[19] *THE CRYPTOCURRENCY FOR PAYMENTS*. Litecoin Project, [Online; navštívené 11.01.2018].
URL https://litecoin.org/

[20] Madeira, A.: *How has Bitcoin mining changed?* CryptoCompare, 2017, [Online; navštívené 27.12.2017].
URL https://www.cryptocompare.com/mining/guides/how-has-bitcoin-mining-changed/

[21] *Mining hardware comparison*. bitcoinwiki, 2017, [Online; navštívené 27.12.2017].
URL https://en.bitcoin.it/wiki/Mining_hardware_comparison

[22] *Monero - Private Digital Currency*. The Monero Project, [Online; navštívené 11.01.2018].
URL https://getmonero.org/

[23] Nakamoto, S.: *Bitcoin: A Peer-to-Peer Electronic Cash System*. www.bitcoin.org, [Online; navštívené 11.01.2018].
URL https://bitcoin.org/bitcoin.pdf

[24] *Namecoin - Freedom of information.* namecoin, [Online; navštívené 11.01.2018].
URL https://namecoin.org/

[25] *NES@FIT - Výzkumná skupina počítačové sítě a vestavěné systémy*. FIT VUT v Brně, 2018, [Online; navštívené 20.04.2018].
URL http://www.fit.vutbr.cz/research/groups/nes@fit/.cs

[26] *Non-specialized hardware comparison*. bitcoinwiki, 2017, [Online; navštívené 27.12.2017].
URL https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison

[27] *Overlay Network*. computernetworkingonlinehelp.com, 2018, [Online; navštívené 04.05.2017].
URL http://computernetworkingonlinehelp.com/OverlayNetwork.php

[28] Očenáš, M.: *Analytické zpracování blockchainu kryptoměn*. Diplomová práca, Vysoké učení technické v Brně, Fakulta informačních technologií, 2017.
URL http://www.fit.vutbr.cz/study/DP/DP.php?id=19354

[29] Palmer, J.; Nakamoto, S.: *DOGECOIN*. [Online; navštívené 11.01.2018].
URL http://dogecoin.com/

[30] Pitts, J.: *A Next-Generation Smart Contract and Decentralized Application Platform*. GitHub, Inc., [Online; navštívené 11.01.2018].
URL https://github.com/ethereum/wiki/wiki/White-Paper

[31] *Proof of Stake*. bitcoinwiki, 2017, [Online; navštívené 26.12.2017].
URL https://en.bitcoin.it/wiki/Proof_of_Stake

[32] van Saberhagen, N.: *Monero CryptoNote v 2.0*. CryptoNote, [Online; navštívené 11.01.2018].
URL https://cryptonote.org/whitepaper.pdf

[33] Sasson, E. B.; Chiesa, A.; Garman, C.; aj.: Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, May 2014, ISSN 1081-6011, s. 459–474, doi:10.1109/SP.2014.36.

[34] Schwartz, D.; Youngs, N.; Britto, A.: *The Ripple Protocol Consensus Algorithm*. Ripple, [Online; navštívené 11.01.2018].
URL https://ripple.com/files/ripple_consensus_whitepaper.pdf

[35] *Stratum Mining Protocol*. GitHub, Inc., [Online; navštívené 05.01.2018].
URL https://github.com/ctubio/php-proxy-stratum/wiki/Stratum-Mining-Protocol

[36] *Integrovaná platforma pro zpracování digitálních dat z bezpečnostních incidentů*. FIT VUT v Brně, 2018, [Online; navštívené. 20.04.2018].
URL http://www.fit.vutbr.cz/units/UIFS/grants/index.php.cs?id=1063

[37] Team, V. D.: *Viacoin Whitepaper*. GitHub, Inc., 2017, [Online; navštívené 26.12.2017].
URL https://github.com/viacoin/documents/raw/master/whitepapers/Viacoin_whitepaper.pdf

[38] *Scale out with Ubuntu Server*. Canonical Ltd., 2018, [Online; navštívené 03.01.2018].
URL https://www.ubuntu.com/server

[39] *Vertcoin is a decentralized currency owned by its users*. Vertcoin, [Online; navštívené 11.01.2018].
URL https://vertcoin.org

[40] *VIACOIN THE FUTURE OF CRYPTO-CURRENCY*. Viacoin, [Online; navštívené 11.01.2018].
URL https://viacoin.org/

[41] *Tor Network & Browser Review*. VPN Fan, 2018, [Online; navštívené 06.05.2017].
URL http://www.vpnfan.com/blog/tor-network-review/

[42] *Zcash - Internet money*. ZEROCOIN ELECTRIC COIN COMPANY, [Online; navštívené 11.01.2018].
URL https://z.cash/

[43] Čapek, J.; Moravec, P.; Palatinus, M.: *Stratum těžební protokol*. SLUSHPOOL, [Online; navštívené 03.01.2018].
URL https://slushpool.com/help/manual/stratum-protocol

Príloha A

Obsah CD

Prílohou diplomovej práce je CD s nasledujúcim obsahom:

- `projekt_text/` – zdrojové súbory technickej správy
- `source/` – zdrojové súbory aplikácie
- `test/` – priečinok obsahujúci testovacie súbory (skripty, PCAP, výstupy,)
- `poster.pdf` – plagát
- `projekt.pdf` – technická správa
- `README` – súbor s nápovedou, popisuje štruktúru súborov na CD

Príloha B

Príklad výmeny správ Ťažobných protokolov

Na obrázku B.1 je znázornená výmena správ, ktorá pozostáva z nadviazania TCP spojenia medzi stanicami, výmenou JSON-RPC správ a ukončenia spojenia. Červenou farbou je znázornená komunikácia v smere klient → server. Modrá farba vyjadruje komunikáciu v opačnom smere. Sekvenčný diagram je doplnený o správy zachytené v programe Wireshark, viď obrázok B.2. Obsah správ protokolu v textovej podobe je nasledovný:

1. mining.subscribe:

   ```
   {"id":1,"method":"mining.subscribe","params":[]}\n
   ```

2. možná odpoveď servera:

   ```
   {"error":null,"id":1,"result":[[["mining.set_difficulty","1"],["mining.
   notify","1"]],"016504012d32c6",4]}
   ```

   ```
   {"params":[2048],"id":null,"method":"mining.set_difficulty"}
   ```

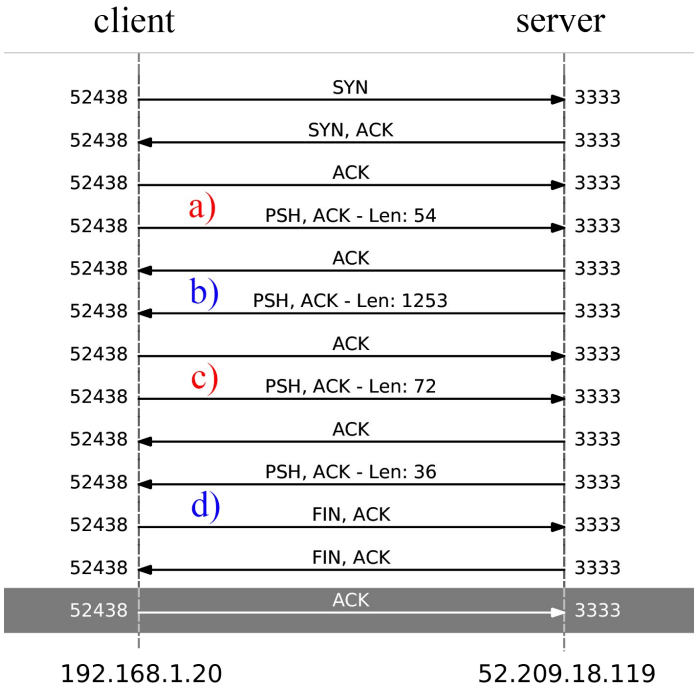
   ```
   {"params":["1517c8","e495a89bae45e84a6631e1f8074f0e06dbe96364005b56910
   000000000000000","0100000001000000000000000000000000000000000000000000000000
   00000000000000000000ffffffff45036db307fabe6d6dfc95dc1beffab6cfc6a74f
   c779ae37150ac8ef5dfb66aec9340e7ca8468b48b90100000000000000","c817152f7
   36c7573682f0000000002621a4268000000001976a9147c154ed1dc59609e3d26abb2d
   f2ea3d587cd8c4188ac0000000000000000266a24aa21a9ed58e41d0c58371158fb072
   6d28414106a9f5b550d99c458c385c1aaabd47beb5100000000",["71bb2d114d21217
   805018024e11264c1e87107449cda00eb977e0a8512be4388","7b333fd75e5a03dc68
   769316043bdd84f0355f2f58d0aa8b3fb51507bdf1d85c","ac40f3e5fe0536feea042
   72b78e1a671f60fa34a13aee5e151c21f22d359fda8","510722eeb44a3d94a22a8c27
   68e47b175ee409be04263c47d82462838ade25cb","f4cec4b8dcbccd93dddcba23c8a
   9a9e2e2a8ed0e1627b440b538c711db8a0229","e61cc7c12cf8c3f7013c1d90e63439
   de3af417cc47456feefd4ba5d12bd16018","fc82a9190ef8926c710e4b8c19ad0f47b
   68da3f88cb28a7c0c000ece8f6ffc2c","fc6250c288248838b23035b84aca619df74a
   8a5061d7900cd12b896b1e084a72","99af555a758e2bb7bcc9257b4c9dcd85bb5d0e3
   006a293a08f6e32f95b7b8f25","1f250cd4f69b281f74ecb8c520ea55928f11c2e2fc
   f0548d1854979fb02a2fd2","c63917cd57b45ba6204cd2597be8a2b381446ecb0b5ca
   29843ff706efe6f65cd","f92c728834bfdb45d1b62a6618eb476e3c3cfa76a1294792
   46c34b95b0bcc9f6"],"20000000","177e578c","5a5f7aae",true],"id":null,"m
   ethod":"mining.notify‘}\n
   ```

3. mining.authorize:

   `{"params":["test","password"],"id":2,"method":"mining.authorize"}\n`

4. možná odpoveď servera:

   `{"error":null,"id":2,"result":true}\n`



Obr. B.1: Výmena JSON-RPC správ protokolu Stratum.

No.	Time	Source	Destination	Protoco	Length	Info
72	13.769557	192.168.1.20	52.209.18.119	TCP	74	52438 → 3333 [SYN] Seq=0 Win=29
73	13.818400	52.209.18.119	192.168.1.20	TCP	74	3333 → 52438 [SYN, ACK] Seq=0 A
74	13.818474	192.168.1.20	52.209.18.119	TCP	66	52438 → 3333 [ACK] Seq=1 Ack=1
75	13.818752	192.168.1.20	52.209.18.119	TCP	120	52438 → 3333 [PSH, ACK] Seq=1 A
76	13.867949	52.209.18.119	192.168.1.20	TCP	66	3333 → 52438 [ACK] Seq=1 Ack=55
77	13.921509	52.209.18.119	192.168.1.20	TCP	1319	3333 → 52438 [PSH, ACK] Seq=1 A
78	13.921916	192.168.1.20	52.209.18.119	TCP	66	52438 → 3333 [ACK] Seq=55 Ack=1
79	13.922388	192.168.1.20	52.209.18.119	TCP	138	52438 → 3333 [PSH, ACK] Seq=55
82	13.968446	52.209.18.119	192.168.1.20	TCP	66	3333 → 52438 [ACK] Seq=1254 Ack
84	14.000425	52.209.18.119	192.168.1.20	TCP	102	3333 → 52438 [PSH, ACK] Seq=125
86	14.004117	192.168.1.20	52.209.18.119	TCP	66	52438 → 3333 [FIN, ACK] Seq=127
89	14.077700	52.209.18.119	192.168.1.20	TCP	66	3333 → 52438 [FIN, ACK] Seq=129
90	14.077752	192.168.1.20	52.209.18.119	TCP	66	52438 → 3333 [ACK] Seq=128 Ack=

Obr. B.2: Odchytená komunikácia v programe Wireshark – Stratum.

Príloha C

Implementované metódy

V tejto kapitole sú uvedený príklady komunikácie klient $\Longleftrightarrow$ server. Odosielané správy implementovaných metód sú označené červenou farbou. Odpovede servera na zaslané dotazy sú označené modrou farbou.

C.1 Stratum

Pre tento protokol bolo špecifikovaných niekoľko metód, nad ktorými je vykonávané testovanie dostupnosti serverov. Výmena správ je zachytená na obrázkoch C.1, C.2, C.3 a C.4.

```
{"jsonrpc": "2.0", "method": "mining.subscribe", "params": ["Miner 1.0"], "id": 1}
{"error":null,"id":1,"result":[[["mining.set_difficulty","1"],
["mining.notify","1"]],"21650600844688",4]}
{"params":[8192],"id":null,"method":"mining.set_difficulty"}
{"params":
["5a346","6602be13cd260aace206979980a7b6b397c8bf30002a29110000000000000000","01000000
0100000000000000000000000000000000000000000000000000000000000000ffffffff4503fcf807f
abe6d6d045a161a62075b44b3f0695511b35ebc7937c0a6ca31380338f013d55b1c841901000000000000
00","46a3052f736c7573682f00000000039bde894a000000001976a9147c154ed1dc59609e3d26abb2df
2ea3d587cd8c4188ac0000000000000002c6a4c2952534b424c4f434b3a021d901f76ec8e9c2565e0348
88070a8c8123db747093897537e917d6851efdf0000000000000000266a24aa21a9edd8d1c0b4170800c2
319d4780f3cb977f99657edda97cea589c011b9df679cd6f00000000",
["6765a95fa49b6787536e7e9afc7166b4152010fe1344894280521611b45f6ac4","c5174d1fe2666afb
55acd0edc94f94ff334c8d0f88b8b29fa840f119ea9b98ea","9d367df7714c49841ba3a0d3792be918fb
5404565c8fddcd18cd2b39d9181a40","8d6fb3a65db74716fc74131c1912b700b10e9d9a58c5e8fbbf05
fe92d0d21960","0920d4ba6755ee423bca110a792a843acc02984f17641573a375ddd672876721","c76
0316582086d7c735d7c9f84a039e55ceafb9865aa3d6458f3f70852c205b5","1319e5181f198c4c31a47
2352e439c7226a684af4578652b46bd1b75c2ba3848"],"20000000","1743eca9","5af804b4",true],
"id":null,"method":"mining.notify"}
```

Obr. C.1: Metóda `mining.subscribe`: ukážka s validnou odpoveďou.

```
{"jsonrpc": "2.0", "method": "mining.authorize", "params": ["Bitcoin", "x"], "id": 1}
{"error":null,"id":1,"result":true}
```

Obr. C.2: Metóda `mining.authorize`: ukážka s validnou odpoveďou.

```
{"jsonrpc": "2.0", "method": "login", "params": {"login": "test.worker1", "pass":
"password", "agent": "xmr/1.0"}, "id": 1}
{"id":1,"jsonrpc":"2.0","error":{"code":-1,"message":"invalid address used for
login"}}
```

Obr. C.3: Metóda `login`: ukážka s negatívnou odpoveďou – „false positive“ z dôvodu chybných užívateľských údajov.

```
{"worker": "eth1.0", "jsonrpc": "2.0", "params": [], "id": 3, "method":
"eth_getWork"}
{"jsonrpc":"2.0","result":
["0x41b7548a77732d39d0ee2e6420f34b82c429ff6322b7c99fa3627043c8c3fb31","0xe6073b5528bd
0132af704e709c5723848c28e74e1d250eff85fc89e916b8515e","0x0000000112e0be826d694b2e62d0
1511f12a6061fbaec8bc02357593e70e52ba"],"id":3}
```

Obr. C.4: Metóda `eth_getWork`: ukážka s validnou odpoveďou.

C.2 Getblocktemplate

Pre tento protokol bola implementovaná jedna metóda – `Getblocktemplate`. Príklad dotazu a odpovede ilustruje obázok C.5.

```
POST / HTTP/1.1
Host: 5.196.74.100:9171
Authorization: Basic Qml0Y29pbjp4
User-Agent: Miner 1.0
Accept: */*
Accept-Encoding: deflate, gzip
Content-Type: application/json
Content-Length: 116
X-Mining-Extensions: longpoll midstate rollntime submitold

{"id": 1, "method": "getblocktemplate", "params": [{"capabilities": ["coinbasetxn",
"workid", "coinbase/append"]}]}
HTTP/1.1 200 OK
Date: Mon, 14 May 2018 14:27:58 GMT
Content-Length: 115
Content-Type: application/json
Server: TwistedWeb/16.0.0

{"error": {"message": "Method not found", "code": -32601, "data": null}, "jsonrpc":
"2.0", "id": 1, "result": null}
```

Obr. C.5: Ukážka s negatívnou odpoveďou – server nepozná metódu.

Príloha D

Plagát

Katalóg kryptomien

Jakub Kelečéni
xkelec00@stud.fit.vutbr.cz
Fakulta informačních technológií, Vysoké Učení technické v Brně

Excel@FIT 2018 ID 21

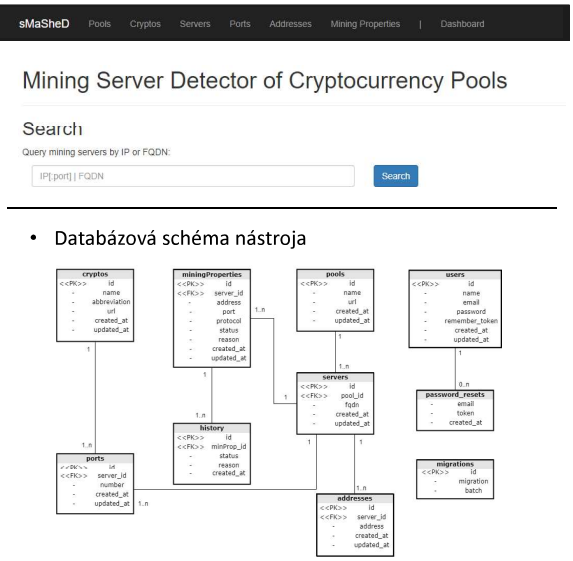
sMaSheD – Mining Server Detector of Cryptocurrency Pools

1. Úvod a motivácia

- uchovanie údajov o ťažobných pooloch
  - databáza
  - kryptomeny, pooly, servery, porty
  - detekcia IP adries serverov
  - prehľad uložených informácií
- ciele
  - zdroj dát sieťových administrátorov, bezpečnostné zložky, ...
  - použitie v odbore forenznej analýzy
  - filtrovanie nežiadúcej komunikácie
- rozšírenie existujúceho nástroja
  - testovanie dostupnosti ťažobných informácií pre adresu/port v pravidelných intervaloch
  - skutočný stav uložených záznamov

Stav	Kód	Popis
DOWN	0	nedostupná cieľová adresa
UP	1	aktívna implementácia ťažobného protokolu
LISTEN	2	neklasifikovaná odpoveď servera

  - ukladanie histórie dostupnosti



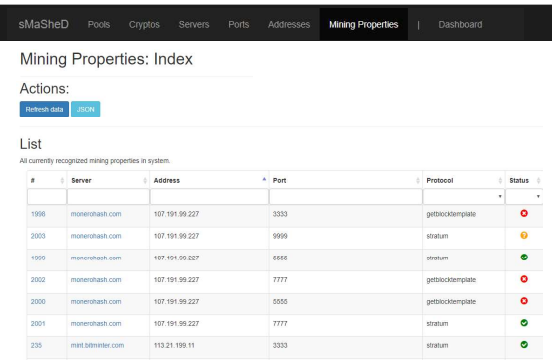
- Databázová schéma nástroja

2. Návrh a implementácia rozšírenia

- analýza „ťažobných“ protokolov
  - Getwork (HTTP), Getblocktemplate (HTTP), Stratum (TCP)
  - používané metódy
  - formát a spôsob výmeny správ

{"jsonrpc": "2.0", "method": "mining.subscribe", "params": ["Miner 1.0"], "id": 1}

- pridanie tabuliek do databáze
  - History
  - MiningProperties
- Pravidelné testovanie uložených záznamov
  - interval 3 hodiny
  - Metódy
    - *mining.subscribe*
    - *mining.authorize*
    - *login*
    - *eth_getwork*
    - *getblocktemplate*
    - *getwork*



- Karta “Mining Properties” – výsledný stav testov

3. Záver

- výsledkom je:
  - doplnenie funkcionality zabezpečujúcej pravidelné testovanie živosti ťažobných poolov
  - dostupnosť uloženého obsahu vo formáte *JSON*
  - lepšia použiteľnosť riešenia ako zdroja dát
    - tvorba pravidiel sieťových filtrov
    - oblasť forenznej analýzy sieťovej prevádzky
  - Stav uložených záznamov
    - 60 % - UP
    - 19 % - LISTEN
    - 21 % - DOWN
- možné rozšírenia:
  - automatické generovanie filtrovacích pravidiel pre sieťové prvky
  - záznam veľkosti hashrate uložených poolov
  - Zahrnutie funkcionality pre správu tzv. multipoolov
- katalóg bol vytvorený v rámci projektu Tarzan
  - Vývoj integrovanej platformy pre spracovanie digitálnych dát z bezpečnostných incidentov