FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Detekce těžby kryptoměn na základě periodického chování síťové komunikace
Student:	Vojtěch Chvojka
Vedoucí:	Ing. Josef Koumar
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Nastudujte problematiku detekce těžby kryptoměn v síti [1] a metodu detekce periodického chování síťového provozu [2]. Pomocí metody detekce periodického chování vytvořte datovou sadu atributů periodického chování komunikace těžby kryptoměn v síťovém provozu. Analyzujte vytvořený dataset s cílem odhalení významných atributů periodického chování pro detekci těžby kryptoměn a na základě nich navrhněte vhodný detekční algoritmus založený na strojovém učení. Vytvořte softwarový prototyp rozšiřující stávající architekturu [2] v jazyce Python. Otestujte a vyhodnoťte vytvořený prototyp z pohledu nasaditelnosti do reálného síťového provozu.

[1] Richard Plný, Karel Hynek and Tomáš Čejka. DeCrypto: Finding Cryptocurrency Miners on ISP networks. 28th Nordic Conference, NordSec 2022.
[2] Josef Koumar and Tomáš Čejka. Network traffic classification based on periodic behavior detection. International Conference on Network and Service Management (CNSM), 2022.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 24. listopadu 2022 v Praze.

Bakalářská práce

DETEKCE TĚŽBY KRYPTOMĚN NA ZÁKLADĚ PERIODICKÉHO CHOVÁNÍ SÍŤOVÉ KOMUNIKACE

Vojtěch Chvojka

Fakulta informačních technologií
Katedra počítačových systémů
Vedoucí: Ing. Josef Koumar
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat svému vedoucímu Ing. Josefovi Koumarovi za ochotu, rady a vstřícnost. Dále bych rád poděkoval rodině, přátelům a kolegům z Rockwell Automation, s.r.o. za neutuchající podporu. Největší díky však patří mojí partnerce Nikole Kadlecové za to, že je podpůrným pilířem mého života, za nekonečnou trpělivost a za skvělé rady a poznámky k práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně imnožstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2023 .....

Abstrakt

Tato bakalářská práce se zabývá detekcí těžby kryptoměn ze síťového provozu. Takový provoz je zpravidla šifrovaný. Periodické vlastnosti síťové komunikace jsou jako vstup strojového učení vhodné proto, že je lze aplikovat i na šifrovanou komunikaci. Byl vytvořen program, který na základě analýzy a testování statistických klasifikátorů vybral klasifikátor XGBoost a vybral periodické vlastnosti síťových toků, které vyhodnotil jako nejvýznamnější pro detekci těžby kryptoměn. Na testovacích datech se podařilo dosáhnout specificity 99,77 % a senzitivity 98,39 %.

Klíčová slova detekce těžby kryptoměn, periodické chování síťové komunikace, monitoring síťového provozu, strojové učení, sklearn, python

Abstract

This bachelor's thesis deals with the detection of cryptocurrency mining from network traffic. Such traffic is usually encrypted. Periodic properties of network communication are suitable as machine learning input because they can be applied to encrypted communication as well. A program was created that, based on the analysis and testing of statistical classifiers, selected the XGBoost classifier and selected the periodic properties of network flows that it evaluated as the most significant for the detection of cryptocurrency mining. A specificity of 99.77 % and a sensitivity of 98.39 % were achieved on the test data.

Keywords crypto mining detection, network traffic periodicity detection, network traffic monitoring, machine learning, sklearn, python

Seznam zkratek

IP	Internet Protocol
TCP	Transmission Control Protocol
CPU	Central Processing Unit
SHA	Secure Hash Algorithm
BTC	Bitcoin
DNS	Domain name system
TLS	Transport layer security
P2P	Peer to peer
RPC	Remote procedure call
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
IoT	Internet of Things
ML	Machine learning
URL	Uniform Resource Locator
IPFIX	IP Flow Information Export
UDP	User Datagram Protocol
CESNET	Czech Education and Scientific NETwork
NEMEA	Network Measurements Analysis
SVM	Support Vector Machine
CSV	Comma-separated values
IANA	Internet assigned numbers authority
LS periodogram	Lombard-Scargle periodogram
SCDF	Scargle cumulative distribution function
STFS	Single flow time series
QoS	Quality of Service

Kapitola 1
Úvod

Detekce těžby kryptoměn je důležitá z několika důvodů:

- Neoprávněné použití: Těžba kryptoměn může být náročná na zdroje a může významně ovlivnit výkon a spotřebu energie systému, kde probíhá. Neautorizovaná těžba, známá také jako cryptojacking, zahrnuje hackery nebo zlomyslné subjekty využívající výpočetní zdroje jiných lidí bez jejich vědomí nebo souhlasu. Detekce takových těžebních aktivit pomáhá identifikovat a předcházet neoprávněnému používání výpočetního výkonu, chrání jednotlivce a organizace před potenciálním snížením výkonu a nadměrnými náklady na energii.
- Bezpečnostní rizika: Cryptojacking může také představovat bezpečnostní rizika. Hackeři mohou využívat těžební software jako prostředek k získání neoprávněného přístupu k systémům nebo sítím, což může vést k narušení dat, krádeži citlivých informací nebo jinému ohrožení zabezpečení systému. Detekce těžebních aktivit může pomoci identifikovat tyto bezpečnostní hrozby a zabránit dalšímu využívání.
- Optimalizace výkonu sítě: Těžba kryptoměn může spotřebovat značnou šířku pásma sítě, což může ovlivnit celkový výkon a stabilitu sítě. Detekcí těžebních aktivit mohou správci sítě identifikovat jakákoli úzká místa v síti způsobená těžbou kryptoměn a přijmout nezbytná opatření k optimalizaci výkonu sítě.
- Sledování legitimního použití hardwaru: Například v podnikovém prostředí mohou zaměstnanci využívat firemní zdroje k těžbě kryptoměn bez oprávnění. Detekce takových těžebních aktivit pomáhá organizacím prosazovat jejich zásady přijatelného používání a zajistit, aby byly výpočetní zdroje využívány pro legitimní účely.

Úspěšná automatická detekce těžby kryptoměn a její možné blokování by mělo řadu pozitivních přínosů pro dnešní společnost. Mimo jiné to jsou ochrana proti neoprávněné činnosti, zmírnění finančních ztrát způsobených neautorizovanou těžbou, ochrana zdrojů energie nebo zlepšení propustnosti sítě internetu.

Řada relevantních prací v klasifikaci síťového provozu pomocí strojového učení klasifikuje na základě portů a IP adres [1, 2, 3, 4]. Nicméně používání IP adres pro klasifikaci hrozeb není vhodná, protože dopředu musím vědět, jaké zařízení je napadené nebo z jakého zdroje v internetu hrozba přichází, ale pokud vím z jakého zdroje hrozba je, tak nepotřebuji používat techniku založenou na strojovém učení. Dále, pokud budeme používat porty na transportní vrstvě TCP/IP modelu, tak je jisté, že se strojové učení na tuto vstupní vlastnost zaměří, protože veřejně datové sady hrozeb jsou tvořeny v laboratorních podmínkách a port je typicky využit pro anotování datové sady. Pro útočníka je tak snadné porty změnit a model nedosáhne slibované přesnosti. Metoda využívající detekci periodického chování prezentovaná v práci [5] je navržená tak, aby

tuto slabinu velké části publikovaných prací neměla. Tato práce je i díky tomu navržená tak, aby těžař kryptoměn měl utajení svojí identity co nejtěžší.

V teoretické části této práce si představíme kryptoměny a technické detaily, které jsou pro tuto práci relevantní. Následně se podíváme hlouběji na existující metody detekce těžby kryptoměn. Na konci teoretické části bude popsáno strojové učení. Dále bude ukázána struktura dat, jejich převedení na časové řady a získání frekvenčních vlastností časových řad. Na závěr práce bude vybrán vhodný klasifikátor, bude vytrénován a laděn pro predikci těžby kryptoměn. Použitý klasifikátor bude použit pro identifikaci významných vlastností časových řad. Statistická úspěšnost klasifikátoru bude vyhodnocena.

1.1 Cíle bakalářské práce

Hlavním cílem bakalářské práce je vytvořit softwarový prototyp, který je rozšířením stávající architektury detekce periodického chování síťového provozu, pro detekci těžby kryptoměn v síťovém provozu.

Prvním dílčím cílem je popsat problematiku detekce těžby kryptoměn, známé metody detekce těžby kryptoměn a metodu detekce periodického chování síťové komunikace. Dalším cílem je vytvořit datovou sadu atributů periodického chování komunikace těžby kryptoměn v síťovém provozu. Sada bude vytvořena pomocí zmíněné metody detekce periodického chování. Následným cílem je analyzovat vytvořenou sadu za účelem odhalení významných atributů periodického chování pro detekci těžby kryptoměn. Na základě významných atributů navrhnout vhodný detekční algoritmus založený na strojovém učení a vytvořit softwarový prototyp. Závěrečným cílem je otestovat vytvořený prototyp na datech poskytnutých vedoucím práce [6] a vyhodnotit vytvořený prototyp z pohledu nasaditelnosti do reálného síťového provozu.

Kapitola 2

Teoretická část

2.1 Kryptoměny

2.1.1 Co to jsou kryptoměny

Obchod na internetu se začal spoléhat téměř výhradně na finanční instituce, které slouží jako důvěryhodné třetí strany pro zpracování elektronických plateb.[7]

Zatímco systém funguje dostatečně dobře, většina transakcí stále trpí inherentními slabinami modelu založeného na důvěře třetí straně. Zcela nevratné transakce nejsou ve skutečnosti možné, protože finanční instituce se nemohou vyhnout transakčním sporům/konfliktům. Náklady na zprostředkování zvyšují transakční náklady, omezují minimální praktickou velikost transakce a snižují možnosti pro malé příležitostné transakce.[7]

Ztráta schopnosti provádět nevratné platby za nevratné služby s sebou nese širší náklady. S možností zvratu se šíří potřeba důvěry. Obchodníci musí být obezřetní vůči svým zákazníkům a obtěžovat je, aby získali více informací, než by jinak potřebovali. Určité procento podvodů je považováno za nevyhnutelné. Těmto nákladům se lze vyhnout provedením platby osobně za použití fyzické měny, ale neexistuje žádný mechanismus pro provádění plateb přes komunikační kanál bez důvěryhodné strany.[7]

Proto je potřeba elektronický platební systém založený na kryptografickém důkazu provedení platby namísto důvěry, který umožní kterýmkoli dvěma ochotným stranám obchodovat přímo mezi sebou bez potřeby důvěryhodné třetí strany. Transakce, jejichž zrušení je výpočetně nepraktické, by chránily prodejce před podvody. Na ochranu kupujících by bylo možné snadno zavést rutinní mechanismy úschovy u třetí osoby.[7]

Satoshi Nakamoto definuje v roce 2008 první kryptoměnu -Bitcoin - jako řetězec digitálních podpisů. Převod měny se provede digitálním podepsáním předchozí transakce spolu s veřejným klíčem nového vlastníka kryptoměny a přidáním nového bloku na konec blockchainu. Příjemce pak může ověřit podpisy a ověřit i celý blockchain vlastnictví.[7]

Aby se zabránilo vícenásobnému utracení kryptoměny jedním účastníkem, i bez důvěryhodné třetí strany, musí být všechny transakce veřejně oznámeny a je zapotřebí systém, který umožní aby se účastníci dohodli na jednotné historii pořadí, v jakém byli transakce provedeny. Příjemce potřebuje důkaz, že v době provedení každé transakce většina uzlů souhlasila s tím, že byla přijata jako první. [7]

Každý blok obsahuje odkaz na blok, který bezprostředně předchází pomocí jeho hash. Změna dat automaticky změní i hash [8]. Aby hacker změnil blok v blockchainu musel by reprodukovat celý řetězec za ním, protože pokud by tak neučinil, vytvoří řetězec neplatných hash hodnot, které nebudou přijaty ostatními účastníky.[9]

2.1.2 Jak se kryptoměny těží

Aby se dalo jednoznačně dokázat, který blok přišel dříve, je potřeba, aby každý blok obsahoval i časovou značku - timestamp - transakce. Timestamp dokazuje, že data musela existovat už v čase přidání bloku do chainu, aby se timestamp dostal do hashe. Každý timestamp obsahuje předchozí timestamp v hashi předchozího bloku. S každým dalším timestamp se posilují ty před ním.[7]

Centralizovaný timestamp server, s sebou nese veškeré nevýhody závislosti na důvěryhodné třetí straně, a přidává její nadměrné zatížení. K implementaci distribuovaného timestamp serveru pro Bitcoin byl zvolen systém proof-of-work, podobný systému Hashcash Adama Backa[10].

Proof-of-work také řeší problém určení zastoupení ve většinovém rozhodování. Pokud by jedna IP adresa znamenala jeden hlas, mohl by ji hlasování rozhodnout kdokoli, kdo je schopen přidělit mnoho IP adres. Proof-of-work je v podstatě implikuje jedno CPU - jeden hlas. Většinové rozhodnutí představuje nejdelší řetězec, do kterého je investováno největší důkazní úsilí. Pokud je většina výkonu CPU řízena poctivými uzly, poctivý řetězec poroste nejrychleji a překoná všechny konkurenční řetězce. Aby mohl útočník upravit minulý blok, musel by znovu provést proof-of-work bloku a všech bloků po něm a pak dohnat a překonat práci poctivých uzlů.[7]

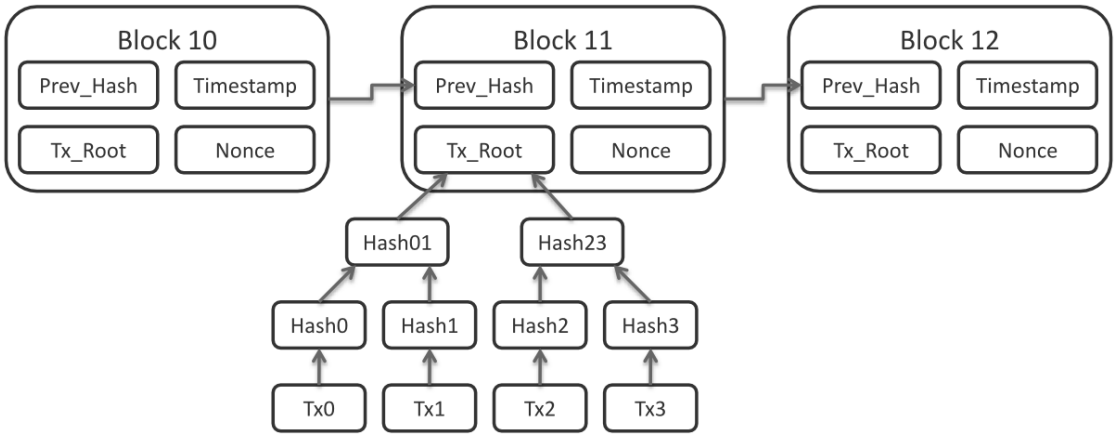
Těžaři jsou jednotlivci, kteří zajišťují síť kryptoměn. Těžba jako taková je proces přidávání záznamů o transakcích do blockchainu kryptoměny[11]. Tento proces zahrnuje řešení hádanky – těžkého matematického problému. Pro Bitcoin je takovým problémem hledání hodnoty nonce (jenom jednou použité číslo), která při zahašování, například pomocí algoritmu SHA-256, vygeneruje hash začínající nějakým počtem nulových bitů. Průměrná požadovaná práce je exponenciální vzhledem k počtu nulových požadovaných bitů, ale lze ji ověřit provedením jediného hash výpočtu [7]. Těžba nového bloku je odměněna získáním coinů dané kryptoměny (buď nových coinů nebo transakčních poplatků). Těžař tedy využívá elektrickou energii výměnou za odměnu. Existují dva typy těžby – sólová a sdružená [12].

Sólová těžba je pokus potvrdit bloky transakcí na Blockchainu sám, jako individuální těžař.[12] Miner získá odměny a transakční poplatky zcela pro sebe – velké platby v delších intervalech. Těžaři, kteří těží sólo, musí komunikovat přímo se sítí kryptoměny. Bohužel existuje velká šance, že sólo těžba nepřinese žádnou odměnu[12].

Dalším typem těžby je sdružená těžba[12]. Těžaři se připojují k těžebnímu poolu a sdílejí zdroje, aby mohli těžit bloky častěji. Odměny za vytěžené bloky jsou rozděleny, což poskytuje menší, ale stabilní platby. Obvykle je rozdělení založeno na práci těžaře, ale přesné nastavení závisí na provozovateli poolu [13]. To umožňuje těžaři získat odměnu, i když to není on, kdo vygeneroval nový blok. Síla poolu se měří v celkovém hash rate – kombinovaném výpočetním výkonu, který je využíván k těžbě a zpracování transakcí [14]. Lze to také interpretovat jako počet hashů, které je pool schopen vygenerovat za sekundu.

Sólový miner postupuje tak, že si stáhne ze sítě nové transakce a vygeneruje hlavička nového bloku. Miner pak začne počítat hashe z hlavičky bloku a z různých hodnot nonce 2.1. Pokud je nalezen hash, který začíná požadovaným počtem nul, blok je dokončen (nebo vytěžen) a odeslán do sítě pro ostatní, aby jej přidali do svého blockchainu[12].

Sdružená těžba popsaná je velmi podobná, ale má jeden hlavní rozdíl. Miner nezíská nové transakce ze sítě, ale připojí se k těžebnímu poolu a požádá o práci[12]. Těžební pool odpoví údaji nezbytnými pro těžbu. Když těžař najde platný hash, oznámí to poolu. Obtížnost v těžebním poolu je obvykle nastavena mírně pod obtížnost sítě. To způsobí, že těžaři posílají výsledky do poolu častěji. Většina těchto hashů není platná, ale používají se k prokázání toho, že horník udělal svůj díl práce. Takové zprávy se nazývají sdílené zprávy [12]. Některé hashe budou platné i pro síť kryptoměny – nový blok je nalezen a vysílán do sítě těžebním poolem. Mechanismus sdílení zpráv lze také použít k rozdělení odměn a poplatků těžařům podle toho, jaké procento akcí udělali.



Obrázek 2.1 Ukázka blockchainu Bitcoinu. Zdroj Wikipedia - Bitcoin blockchain

2.1.3 Protokoly používané při těžbě kryptoměn

Jak při sólo, tak při společném těžení potřebuje těžební software získat informace potřebné k vytvoření hlaviček bloků. [13] K dispozici je spousta těžebního softwaru [15]. Každý těžební software implementuje specifický algoritmus pro kryptoměnu, kterou má těžit. Navíc musí implementovat síťové protokoly pro komunikaci s těžebním poolem, jako je Getwork [16] nebo Stratum [17, 18]. Komunikace staví na standardních síťových protokolech, jako je TCP, DNS nebo TLS [13, 19]. Peer 2 peer síť využívají jak sólo těžaři tak i těžební pooly pro komunikaci s celou kryptoměnovou sítí [13].

Transmission Control Protocol (TCP) je „connection-oriented, end-to-end spolehlivý protokol transportní vrstvy [20]. Jedná se o nejoblíbenější protokol pro spolehlivý přenos dat z transportní vrstvy [21]. Tento protokol se používá pro seřazené a chybově kontrolované doručení tzv. proudu bajtů. Spojení je navázáno třístranným handshakem před odesláním jakýchkoli dat aplikace. [22]

Transport Layer Security (TLS) je protokol z aplikační vrstvy, jehož primárním cílem je poskytnout soukromí a integritu dat mezi dvěma komunikujícími aplikacemi [23]. Je také nástupcem Secure Sockets Layer (SSL). TLS spojení verze 1.3 je soukromé — data jsou šifrována pomocí symetrické kryptografie a pro každé spojení jsou použity jedinečné klíče [23].

Domain Name System (DNS) je protokol který poskytuje mechanismus pro pojmenování zdrojů takovým způsobem, aby byla jména použitelná v různých serverech, sítích, protokolech, internetech a administrativních organizacích [24]. Umožňuje překlad takzvaných doménových jmen na adresy jiných protokolů a naopak.[24]

Nejjednodušší a nejstarší metodou byla, nyní již zastaralá, Bitcoin core Getwork RPC, která vytváří hlavičku přímo pro těžaře. Vzhledem k tomu, že hlavička obsahuje pouze jeden 4bajtový nonce, což znamená více než 4 miliardy možností, mnoho moderních těžařů potřebuje provádět desítky nebo stovky požadavků getwork za sekundu. Sóloví těžaři mohou stále používat getwork na verzi 0.9.5 nebo nižší, ale většina dnešních poolů od jeho použití odrazuje nebo ho přímo zakazuje.[13]

Vylepšenou metodou je Bitcoin core GetBlockTemplate RPC. Ta poskytuje těžebnímu softwaru mnohem více informací potřebných k sestavení hlavičky, včetně veškerých transakcí, které jsou součástí nového bloku, a cílový počet nulových bitů na začátku hashe. Narozdíl od staršího Getwork RPC nepotřebuje nový příkaz pro každý nonce, ty si generuje miner sám.[13]

Oba výše zmíněné Bitcoin core protokoly běží nad HTTP. Aby bylo zajištěno, že dostanou nejnovější práci, většina těžařů používá HTTP longpoll [13].

Široce používanou alternativou k GetBlockTemplate je protokol Stratum mining. Stratum se zaměřuje na to, aby těžařům poskytlo minimální informace, které potřebují k samostatnému

vytvoření hlaviček bloků. Na rozdíl od GetBlockTemplate, těžaři používající Stratum nemohou kontrolovat ani přidávat transakce do bloku, který aktuálně těží. Také na rozdíl od GetBlockTemplate protokol Stratum používá přímo obousměrný TCP soket, takže těžaři nemusí používat HTTP longpoll, aby zajistili, že obdrží okamžité aktualizace z těžebního poolu, když je nový blok vysílán do peer-to-peer sítě.

Stratum V2 je přímým nástupcem Stratum. JSON byl nahrazen binárním formátem, aby se snížila režie. Zprávy proto již nejsou čitelné pro člověka. V2 je navíc autentizovaná a šifrovaná, což zabraňuje hashrate hijacking [25, 18].

2.1.4 Proč detekovat těžbu kryptoměn

Těžba kryptoměn spotřebovává značné množství energie během výpočtu proof-of-work, který je potřeba pro přidání nového bloku do blockcahinu. Na základě dříve vypočítaných vzorců spotřeby energie pro těžbu čtyř prominentních kryptoměn (Bitcoin, Ethereum, Litecoin a Monero) odhadují Goodkind a Jones [26] ekonomické škody způsobené emisemi znečištěného ovzduší na jednu minci a související dopady na lidskou úmrtnost a klima při těžbě těchto kryptoměn v USA a Číně tak, že v roce 2018 byl každý jeden vytvořený dolar bitcoinové hodnoty zodpovědný za škody na zdraví a klimatu ve výši 0,49 dolaru v USA a 0,37 dolaru v Číně. K podobné hodnotě v Číně ve srovnání s USA dochází navzdory extrémně velkému rozdílu ve statistickém odhadu doby dožití v USA a Číně.[26]

Cryptomining je vysoce kompetitivní proces, protože aby jste získali odměnu, musíte být první. Proto čím více hashů, které jste schopni vypočítat za jednotku času, tím větší máte šanci na odměnu. S tím přichází velká motivace ke zneužití cizího hardware. Při nezákonné těžbě kryptoměn využívá útočník zdroje ukradené obětem k těžbě kryptoměn a výsledky pak prezentuje svým jménem [27]. Analýza zisků od autorů Pastrana a Suarez-Tangil [27] odhaluje těžební kampaně s mnohamilionovými výdělky, které spojují více než 4,4 % Monera s nezákonnou těžbou. Jejich analýza infrastruktury také ukazuje, že velká část tohoto ekosystému je podporována pricing modely jako je například Pay-Per-Install [27].

Pastrana a Suarez-Tangil [27] popsali dva typy těžebního malwaru: browser-based a binary-based. Browser-based těžba, jinak také nazývaná cryptojacking, běží ve webovém prohlížeči (na počítači uživatele webové stránky) a používá ke spuštění na webových stránkách skripty (typicky JavaScript). Proces těžby začíná, když uživatel navštíví webovou stránku. Binary-based těžba využívá malware k infikování počítače připojeného k internetu a poté spouští binární program, který provádí proces těžby. Útočníci mohou pomocí stovek infikovaných strojů získat hash rate středně velké těžařské farmy.

Existují důkazy, že malware pro těžbu kryptoměn ovlivňuje technologii internetu věcí (IoT), jako jsou síťová úložiště (NAS), IP kamery, tiskové servery a směrovače. DeJesus věří, že tato hrozba poroste [28].

Těžba kryptoměn představuje pro podnikové sítě řadu potenciálních rizik. Od rizika, že by mohly být napadeny webové stránky podniku za účelem šíření malwaru pro těžbu mincí, až po riziko, že si zaměstnanci mohou do svých prohlížečů nainstalovat pluginy pro těžbu mincí. Většinu hlavních kryptoměn, jako je bitcoin, lze nyní efektivně těžit pouze s využitím výpočetního výkonu na enterprise úrovni, čímž se podnikové sítě staly velkým cílem pro nelegální těžební software[28].

2.2 Existující metody detekce těžby kryptoměn

Jingqiang Liu [29] navrhl metodu detekce takzvané ”silent mining”browser-based těžby kryptomněn založenou na analýze vlastnostech browser-based softwaru. Tato metoda získává snapshoty z haldy a ze zásobníku dynamického kódu webového prohlížeče a provádí detekci založenou na rekurentní neuronové síti.

Zhaoyan Liu [30] navrhl metodu založenou na rozpoznávání vzorců v datech o denní spotřebě elektřiny.

Swedan [31] navrhl řešení "Mining Detection and Prevention System (MDPS)", které využívá využívá blacklisty URL a jiné metody založené na URL.

Kharraz [32] ve své práci zkoumal knihovny kryptojackingu. Jako vstupní parametry pro modely ML byly použity čas kompilace JavaScriptu, doba běhu JavaScriptu, Garbage collection a další statistiky. Nejlepší model (SVM) měl více než 95 % skutečných pozitivních výsledků.

Výše zmíněné metody mají své využití, ale používají pro detekci těžby jiné metriky než tato práce a nejsou proto dále diskutovány.

2.2.1 Detekce pomocí metrik NetFlow/IPFIX

Munoz a kol. [33] prezentovali metodu strojového učení, která je schopna odhalit těžaře kryptoměn pomocí síťových metrik NetFlow/IPFIX. Prezentovaná metoda nepotřebuje provádět hloubkovou inspekci paketů, ale přesto dosahuje podobné přesnosti jako techniky na ní založené. Zachycený provoz byl analyzován a bylo zjištěno, že toky těžařů mají dlouhé trvání a mají malý počet přenesených paketů. Server navíc obvykle odesílá 20krát více dat než klient.

NetFlow/IPFIX je protokol, který provádí síťová měření na úrovni toku agregováním veškerého provozu odpovídající stejné pětici. Tato pětice je zdrojová a cílová IP adresa, zdrojový a cílový port TCP/UDP a protokol zabalený v transportním protokolu [33].

2.3 Richard Plný - detekce z IP flow dat

2.3.1 Vytvoření datové sady

Data v podobě síťových toků byla sbírána ze síťového provozu na síti CESNET. Plný [22] dále popisuje, jak vytvořil datovou sadu pro strojové učení. Provedl důslednou analýzu webových stránek kryptoměn a na jejich základě vytvořil pravidlo pro zachycení provozu. Každý účastník provozu (IP adresa + port), který byl podezříván z provozování poolu byl požádán, aby poskytl práci pro Plného těžaře, čímž mu potvrdil, že je to pool a jeho komunikace byla označena jako miner [22]. Zachycená komunikace byla následně pomocí NEMEA modulu logger převedena do podoby síťových toků.

Plný [22] pak vybral/dopočítal vlastnosti síťových toků zobrazené v tabulce 2.1.

Tabulka 2.1 Vlastnosti síťového toku použité Richardem Plným.

Název vlastnosti	Zdroj	Popis
BYTES	síťový tok	Počet bajtů ze zdroje
BYTES_REV	síťový tok	Počet bajtů z cíle
PACKETS	síťový tok	53 Počet paketů ze zdroje
PACKETS_REV	síťový tok	Počet paketů z cíle
SENT_PERCENTAGE	vypočítáno ze síťového toku	procento paketů ze zdroje
RECV_PERCENTAGE	vypočítáno ze síťového toku	procento paketů z cíle
IS_REQUEST_RESPONSE	vypočítáno ze síťového toku	Je-li komunikace vyvážená
AVG_PKT_LEN	vypočítáno ze síťového toku	průměrná velikost paketu
ABG_SECS_BETWEEN_PKTS	vypočítáno ze síťového toku	průměrná doba mezi pakety
OVERALL_DURATION_IN_SECS	vypočítáno ze síťového toku	celková doba síťového toku
PSH_RATIO	vypočítáno ze síťového toku	procento paketů s TCP-PUSH vlaječkou

Tyto vlastnosti byly použity jako vstup pro strojové učení. Na testovacích datech bylo

dosaženo accuracy 97,77-99.29 % a precision 98,07-99.61 % (metriky úspěšnosti accuracy a precison jsou popsány v kapitole 4.1).

Tato práce množinu vlastností tohoto modelu rozšíří o další statistické a periodické vlastnosti popsané v [5].

2.4 Strojové učení

Strojové učení (ML) je vědecká disciplína studující algoritmy a statistické modely, které počítačové systémy používají k provádění konkrétního úkolu, aniž by byly explicitně naprogramovány[34]. Hlavní výhodou používání strojového učení je, že jakmile se algoritmus naučí, co dělat s daty, může svou práci dělat automaticky[34].

Strojové učení se při řešení problémů s daty spoléhá na různé algoritmy[34]. Datoví vědci rádi poukazují na to, že neexistuje jediný univerzální typ algoritmu, který by byl nejlepší k vyřešení problému. Druh použitého algoritmu závisí na druhu problému, který chcete řešit, počtu proměnných, typu modelu, který by mu nejlépe vyhovoval a tak dále[34].

Supervised learning (Učení pod dohledem) je typ strojového učení spočívající ve vytvoření (naučení se) funkce, která mapuje vstup na výstup na základě příkladných párů vstup-výstup. Jako supervised learning označíme ty algoritmy, které potřebují externí pomoc[34].

V případě unsupervised learning (učení bez dozoru), na rozdíl od výše uvedeného supervised learning neexistují (nebo nejsou známy) žádné správné odpovědi a neexistuje žádný vhodný učitel. Algoritmy jsou ponechány, aby objevily strukturu v datech sami[34].

Rozhodovací strom (Decision tree) je strom, jehož vnitřní uzly lze brát jako testy (na vzorech vstupních dat) a jehož listové uzly lze brát jako kategorie (těchto vzorů). Tyto testy jsou filtrovány přes strom, aby se dostal správný výstup ke vstupnímu vzoru[35].

V závislosti na situaci a požadovaném výsledku existují různé typy rozhodovacích stromů, které můžete použít[35]:

1. Klasifikační strom - Nejlépe předvídatelný výsledek lze získat z různých částí informací, které se vypočítaly pomocí klasifikačního stromu. Tento druh stromu by našel uplatnění v pravděpodobnosti a statistice.

2. Regresní strom - Aby bylo možné určit jeden jediný předem určený výsledek z různých informací Regresní strom. Tento strom se používá při výpočtech pro nemovitosti.

3. Rozhodovací stromové lesy - Je vytvořeno několik různých rozhodovacích stromů a poté seskupeny, aby bylo možné přesněji určit, co se stane s konkrétním výsledkem.

4. Klasifikační a regresní strom - Aby byl výsledek co nejlogičtější, je předpovídán pomocí závislých faktorů.

Naivní Bayes je klasifikační technika založená na Bayesově teorému s předpokladem nezávislosti mezi prediktory[34]. Klasifikátor Naive Bayes předpokládá, že přítomnost určitého prvku ve třídě nesouvisí s přítomností jakéhokoli jiného prvku. Používá se hlavně pro účely shlukování a klasifikace v závislosti na podmíněné pravděpodobnosti výskytu.

Další často používannou technikou strojového učení je Support Vector Machine (SVM)[34]. SVM modely jsou modely s přidruženými algoritmy učení, které analyzují data používaná pro klasifikaci a regresní analýzu[34]. Kromě provádění lineární klasifikace mohou SVM efektivně provádět nelineární klasifikaci pomocí toho, čemu se říká kernel trik, implicitně mapující své vstupy do prostorů vysoce dimenzionálních prvků. Je to v podstatě kreslení okrajů mezi třídami. Okraje jsou nakresleny takovým způsobem, že vzdálenost mezi okrajem a třídami je maximální a tím se minimalizuje chyba klasifikace[34].

Termín Boosting se týká rodiny algoritmů, které převádějí slabé algoritmy na silné[34]. Boosting je technika v souborovém učení, která se používá ke snížení zkreslení a rozptylu[34].

Posilování je založeno na otázce, kterou položili Kearns a Valiant: „Může soubor slabých algorimů vytvořit jeden silný algoritmus?“ Slabý algoritmus je definován jako libovolný klasifikátor a silný algoritmus je klasifikátor, který je dobře korelován se skutečnou klasifikací[34].

Kapitola 3
Data

Za účelem strojového učení byly použity datasety "Datasets of Cryptomining communication"[6]. Data byla sbírána na národní síti CESNET2 v obdobích prosinec 2021 - únor 2022 (`decrypto_dataset_design.csv`) a během března 2022 (`decrypto_dataset_evaluation.csv`) [6]. Oba datasety byly vytvořeny pomocí programu 'ipfixprobe'[1] a jsou ve formátu CSV. Sloupce jsou popsány v tabulce 3.1 (Seřazeno abecedně). Poskytnutá data jsou proti původním obohacena o hash IP adres. To je potřeba k vytvoření seznamu síťových závislostí. Jeden řádek těchto dat je roven jednomu IP flow [36].

Tabulka 3.1 Formát použitých dat

Název sloupce	Popis
BYTES	Počet přenesených bajtů (CLI to SRV)
`BYTES_REV`	Počet přenesených bajtů opačným směrem (SRV to CLI)
`DST_IP_ID`	Hash cílové IP adresy (SRV)
`DST_PORT`	Cílový port (SRV)
LABEL	Štítek pro strojové učení. Nabývá hodnot Miner nebo Other.
PACKETS	Počet přenesených paketů (CLI to SRV)
`PACKETS_REV`	Počet přenesených paketů opačným směrem (SRV to CLI)
`PPI_PKT_DIRECTIONS`	Pole směrů prvních 30 paketů. 1 pro CLI to SRV, -1 pro SRV to CLI
`PPI_PKT_FLAGS`	Pole TCP příznaků (flags) prvních 30 paketů.
`PPI_PKT_LENGTHS`	Pole velikosti prvních 30 paketů.
`PPI_PKT_TIMES`	Pole časových značek zachycení prvních 30 paketů.
PROTOCOL	Protokol použitý na transportní vrstvě ISO/OSI modelu.
`SRC_IP_ID`	Hash zdrojvé IP adresy.
`SRC_PORT`	Zdrojový port (CLI)
`TCP_FLAGS`	TCP příznaky (flags) prvního paketu poslaného klientem.
`TCP_FLAGS_REV`	TCP příznaky (flags) prvního paketu poslaného serverem.
`TIME_FIRST`	Časová známka zachycení prvního paketu.
`TIME_LAST`	Časová známka zachycení posledního paketu.

[1] https://github.com/CESNET/ipfixprobe

3.1 IPFIX - RFC7011

IPFIX neboli Internet Protocol Flow Information Export je síťový protokol, který standardizuje formát a přenos informací o toku v rámci sítě IP. IPFIX je definován v RFC 7011 a je založen na předchozí práci provedené protokolem NetFlow v9 společnosti Cisco Systems. Poskytuje flexibilní rámec pro export informací o toku sítě ze směrovačů, přepínačů a dalších síťových zařízení do externích kolektorů pro účely analýzy a monitorování. [36]

Hlavním cílem IPFIX je umožnit efektivní a konzistentní export tokových dat z různých síťových prvků a zajistit interoperabilitu mezi různými prodejci a zařízeními. Umožňuje správcům sítě a bezpečnostním analytikům získat přehled o vzorcích síťového provozu, identifikovat anomálie a provádět různé úkoly analýzy provozu. [36]

IPFIX zavádí sadu standardních informačních prvků, které popisují různé aspekty síťových toků, jako jsou zdrojové a cílové IP adresy, porty transportní vrstvy, typy protokolů, počty paketů a bajtů, časová razítka a další relevantní atributy. Tyto informační prvky jsou definovány v informačním modelu IPFIX, který poskytuje společný slovník pro popis tokových dat. [36]

Protokol IPFIX funguje na modelu klient-server. Síťová zařízení fungující jako exportéři generují záznamy toku IPFIX na základě nakonfigurovaných pravidel vzorkování nebo filtrování. Tyto záznamy toků obsahují příslušné informační prvky a jsou pravidelně exportovány do jednoho nebo více kolektorů IPFIX. Kolektory přijímají, ukládají a zpracovávají záznamy toků, poskytují přehled o chování sítě a usnadňují analýzu provozu. [36]

IPFIX podporuje transportní protokoly orientované na připojení i bez připojení, jako je TCP a UDP, což umožňuje flexibilitu z hlediska spolehlivosti a efektivity. Definuje mechanismus založený na šablonách pro export záznamů toku, kde si exportéři a sběratelé vyměňují šablony, aby se zajistilo, že struktura a formát exportovaných dat jsou dobře definované a srozumitelné. [36]

IPFIX také obsahuje mechanismy pro správu životnosti šablon, zpracování změn šablon a podporu exportu možností a dat s proměnnou délkou. Tato flexibilita umožňuje reprezentovat různé informace o toku, přizpůsobovat se různým síťovým prostředím a vyvíjejícím se požadavkům. [36]

3.1.1 IPFIX flows - síťové toky

Toky v kontextu IPFIX a síťové analýzy představují sekvenci souvisejících síťových paketů, které sdílejí společné charakteristiky. Tok je definován jako jednosměrný proud paketů, které sdílejí sadu klíčových polí, jako jsou zdrojové a cílové IP adresy, porty transportní vrstvy, typ protokolu a další relevantní atributy. Tato klíčová pole se používají k identifikaci a rozlišení jednoho toku od druhého. [36]

Toky jsou základní jednotky měření a analýzy síťového provozu. Poskytují konsolidovaný pohled na chování sítě seskupováním souvisejících paketů, což umožňuje správcům sítě a bezpečnostním analytikům porozumět vzorcům a charakteristikám síťové komunikace. Některé klíčové aspekty toků zahrnují: [36]

- Klíč toku: Klíč toku je kombinací parametrů, která jednoznačně identifikují tok. Klíč toku obvykle obsahuje zdrojové a cílové IP adresy, zdrojové a cílové porty, typ protokolu a další relevantní pole. Klíč toku se používá ke spárování paketů a určení, ke kterému toku patří.
- Atributy toku: Atributy toku jsou různé vlastnosti a charakteristiky spojené s tokem. Tyto atributy mohou zahrnovat počty paketů a bajtů, časové značky, informace o kvalitě služeb (QoS), informace na úrovni aplikace a další. Atributy toku poskytují další kontext a informace o chování a výkonu síťového provozu.
- Monitorování toku: Monitorování toku zahrnuje shromažďování, analýzu a ukládání dat toku pro účely analýzy sítě. Síťová zařízení, jako jsou směrovače, přepínače a firewally, lze nakonfigurovat tak, aby monitorovala provoz a generovala záznamy o toku na základě předem

definovaných kritérií. Tyto záznamy toků zachycují informace o jednotlivých tocích a jsou exportovány do kolektorů k analýze.

- Export toku: Export toku se týká procesu přenosu záznamů toku ze síťových zařízení do kolektorů pro další analýzu. IPFIX poskytuje standardizovaný formát a protokol pro export záznamů toku, což zajišťuje interoperabilitu mezi různými zařízeními a dodavateli. Záznamy toků obsahují informace o tocích pozorovaných exportérem, což umožňuje kolektorům rekonstruovat a analyzovat síťový provoz.
- Analýza toku: Analýza toku zahrnuje zkoumání záznamů toku, abyste získali přehled o chování, výkonu a zabezpečení sítě. Analýzou dat o toku mohou správci sítě identifikovat top mluvčí (zařízení generující největší provoz), detekovat síťové anomálie, měřit využití šířky pásma, provádět profilování provozu a detekovat potenciální bezpečnostní hrozby, jako jsou DDoS útoky nebo podezřelé komunikační vzorce.

Analýza založená na toku poskytuje škálovatelný a účinný přístup k monitorování sítě a řešení problémů. Tím, že se zaměřuje na agregovaná data toku namísto jednotlivých paketů, snižuje objem dat, která mají být zpracována, a přitom stále zachycuje základní informace o vzorcích síťového provozu.

3.2 Vytvoření časových řad

3.2.1 Časové řady

Časová řada je chronologicky uspořádané pozorování hodnot nějaké náhodné veličiny. To znamená, že časová řada je posloupnost hodnot vygenerovaných nějakým procesem, pro které platí, že mají přiřazen nějakých čas na časové ose [5]. Pokud platí, že časová informace hodnot má tvar t:=1,2,...,n, pak mluvíme o rovnoměrně rozložené časové řadě. Jinak mluvíme o nerovnoměrně rozložené časové řadě [5].

3.2.2 Tvorba časové řady ze síťových toků

Pro potřeby získání periodických atributů komunikace bylo potřeba ze síťových toků vytvořit časové řady. Následující postup popsal Koumar [5] ve svojí práci. Nejprve je potřeba vytvořit seznam síťových závislostí. Síťová závislost je vztah mezi IP adresou klienta a službou na jiné IP adrese a portu, tedy dlouhodobý provoz mezi dvěma IP adresami pod nějakým (většinou registrovaným) portem transportní vrstvy [5]. Příklady závislostí jsou v tabulce 3.2 (zdroj: [5])

Tabulka 3.2 Příklady síťových závislostí.

Zdrojová IP adresa	Cílová IP adresa	Port(y) transportní vrstvy
192.168.0.10	192.168.0.1	53
192.168.0.10	93.25.16.34	443
192.168.0.11	192.168.0.1	53
192.168.0.11	192.168.0.2	21
192.168.0.11	25.162.154.2	80124 - 49012

Seznam registrovaných portů transportní vrstvy byl stažen ze stránek IANA (Internet Assigned Numbers Authority) [37].

Pro Vytvoření seznamu závislostí byl použit modul `add_dependency.py` 3.1. Parametr ”-t flow-csv”vybírá, že síťové toky jsou poskytnuty ve formátu CSV. Následují parametry -f a -d pro

výběr vstupního a výstupního souboru. Parametr –delimiter vybere jaký oddělovač je v souboru CSV použit. Následují řetězce identifikující jak jsou pojmenovány klíčové sloupce v hlavičce.

Výpis kódu 3.1 Ukázka spuštění modulu add-dependency.py

```
./add_dependency.py
    -t flow-csv
    -f design_ready.csv
    -d design_ready.csv.dependencies.csv
    --delimiter=','
    --src_ip="string␣SRC_IP_ID"
    --dst_ip="string␣DST_IP_ID"
    --src_port="uint16␣SRC_PORT"
    --dst_port="uint16␣DST_PORT"
```

Modul přiřadí síťové toky síťovým závislostem. Pokud nemá ani jedna komunikující strana známý port (podle IANA), přiřadí tok k závislosti `{src_ip}{src_port}-{dst_port}{dst_ip}`. Pokud má pouze jedna strana známý port, předpokládá se, že tato strana je server a tok se přiřadí závislosti `{srv_ip}{srv_port}-{cli_ip}`, čímž efektivně agreguje všechny aplikace komunikující z `cli_ip` s aplikací poslouchající na `srv_ip:src_port`. Pokud mají obě komunikující strany známý port, dostane přednost port menší než 1024. Jsou-li oba menší, předpokládá se, že src je klient a tok se přiřadí závislosti `{dst_ip}{dst_port}-{src_ip}`. Identifikátor závislosti je přidán k datům jako nový sloupec `ID_DEPENDENCY`.

Výstup vytváření síťových závislostí je použit jako vstup modulu `create_time_series_from_flow.py` 3.2. Parametry -f a -t slouží pro výběr vstupního a výstupního souboru. Ostatní parametry slouží k identifikaci klíčových sloupců v hlavičce vstupního souboru.

Výpis kódu 3.2 Ukázka spuštění modulu create-time-series-from-flow.py

```
./create_time_series_from_flow.py
    -f design_ready.csv.dependencies.csv
    -t design_ready.csv.flow_timeseries.csv
    --packets="uint32␣PACKETS"
    --packets_rev="uint32␣PACKETS_REV"
    --bytes="uint64␣BYTES"
    --bytes_rev="uint64␣BYTES_REV"
    --start_time="time␣TIME_FIRST"
    --end_time="time␣TIME_LAST"
    --label="string␣LABEL"
    --time_format="%Y-%m-%dT%H:%M:%S.%f"
```

Modul vytváří pro každou závislost jednu časovou řadu. Časovou řadu definuje jako seřazenou posloupnost datapointů, kde jeden datapoint se získá z jednoho síťového toku a je tvořen jako pětice (`PACKETS` + `PACKETS_REV`; `BYTES` + `BYTES_REV`; `TIME_FIRST`; `TIME_LAST`; `LABEL`). Původní použití pythoních slovníků (dict) bylo nahrazeno použitím modulu dataclasses, pro ulehčení typového našeptávání a zlepšení čitelnosti.

Takto vzniklé časové řady mohou obsahovat velký časový interval jedné závislosti. Od okamžiku prvního zachycení této síťové závislosti po poslední zachycení a to i v případě, že budou síťové toky od sebe odděleny dlouhým časovým úsekem neaktivity (pakety zachycené např. v prosinci 2021 budou zařazeny do stejné časové řady jako pakety zachycené v únoru následujícího roku, budou-li mít stejnou síťovou závislost). To nemusí vždy nejlépe vyhovovat strojovému učení pro účely detekce těžby kryptoměn. Nastane-li dlouhá neaktivita nějaké závislosti, je možné, že na té stejné závislosti příště poběží jiný typ provozu.

Z tohoto důvodu byla délka časové řady omezena. Takové omezení má i tu výhodu, že se strojové učení naučí poznávat i dlouho běžící těžaře z relativně krátkého časového úseku. Pokud tedy časová řada přeroste 21600 vteřin, neboli 6 hodin, bude přiřazena nové závislosti (přidáním

sekvenčního čísla na konec původního názvu závislosti) a další síťové toky původní závislosti budou přidávány do nové časové řady. Hodnota 21600 vteřin byla vybrána na základě testování strojového učení. Vedla k dobré úspěšnosti testovaných klasifikátorů.

3.3 Získání periodického chování

Detekce periodické komunikace založená na časových řadách ze síťových toků pracuje s více parametry síťového toku (např. počet paketů, počet bajtů) současně. Tyto parametry dále nazýváme metrikami detekce periodicity [5]. Z časové řady se pomocí tzv. autokorelační funkce získají kandidáti na periodu. Následně se vytvoří Lomb-Scargle (LS) periodogram a pomocí Scarglovy kumulativní distribuční funkce (SCDF) se provede test pro všechny frekvence kandidátů na periodicity a získá se periodické chování. Algoritmus pro detekci periodicity je poskytnut vedoucím práce.

3.3.1 Lobm-Scargle periodogram

Lomb-Scargle periodogram je matematická technika používaná v oblasti zpracování signálu a analýzy dat k odhadu dominantních periodic nebo frekvencí přítomných v nerovnoměrně vzorkovaných datech časových řad. Byl vyvinut Lombem v roce 1976 a nezávisle znovu objeven Scarglem v roce 1982. [38]

Lomb-Scargle periodogram je zvláště užitečný při práci s daty časových řad, které nejsou vzorkovány jednotně, jako jsou astronomická pozorování nebo nepravidelně sbíraná data. Je široce používán v různých vědeckých oborech, včetně astrofyziky, geofyziky a biomedicínského výzkumu. [38]

Periodogram je graf, který ukazuje výkonové spektrum signálu jako funkci frekvence. V Lomb-Scargle periodogramu je výkonové spektrum vypočítáno přizpůsobením sinusového modelu datům na různých frekvencích a odhadem síly každé frekvenční složky. [38]

Zde jsou hlavní kroky při výpočtu Lomb-Scargleho periodogramu [38]:

1. Předzpracování: Pokud v časové řadě chybí nějaké datové body nebo odlehlé hodnoty, mohou být nezbytné příslušné kroky předběžného zpracování, jako je interpolace nebo čištění dat.

2. Střední odečítání: Průměrná hodnota časové řady se odečte od každého datového bodu a získá se časová řada s nulovým průměrem.

3. Výpočet frekvence: Stanoví se frekvence, při kterých bude periodogram vyhodnocován. Tyto frekvence obvykle pokrývají rozsah od nejnižší možné frekvence (vztažené k celkovému trvání časové řady) po nejvyšší frekvenci zájmu.

4. Minimalizace fázového rozptylu: Pro každou frekvenci vypočítává Lomb-Scargleův periodogram statistiku minimalizace fázového rozptylu (PDM). PDM měří správnost přizpůsobení sinusového modelu k datům minimalizací rozptylu datových bodů ve fázových přihrádkách.

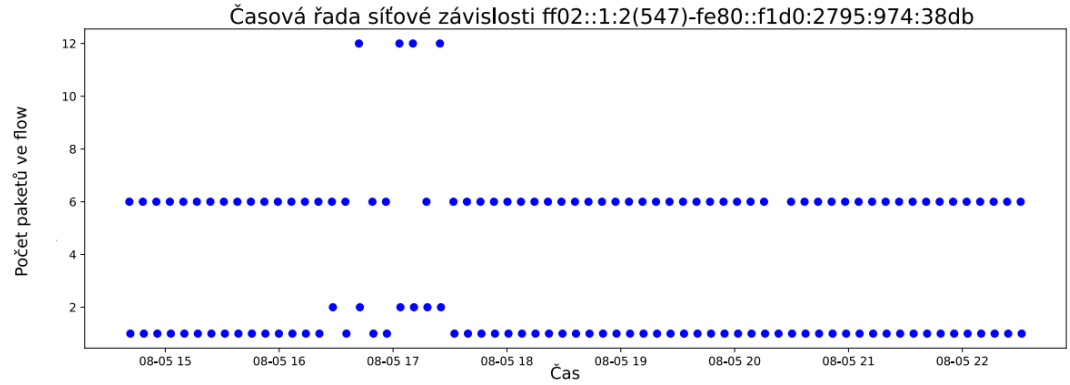
5. Výpočet energetického spektra: Statistiky PDM jsou převedeny na výkonové spektrum převrácením statistik a jejich normalizací.

6. Odhad významnosti: Ke stanovení významnosti zjištěných frekvencí lze použít statistické testy. Mezi běžné metody patří výpočet pravděpodobnosti falešných poplachů nebo stanovení úrovní spolehlivosti.

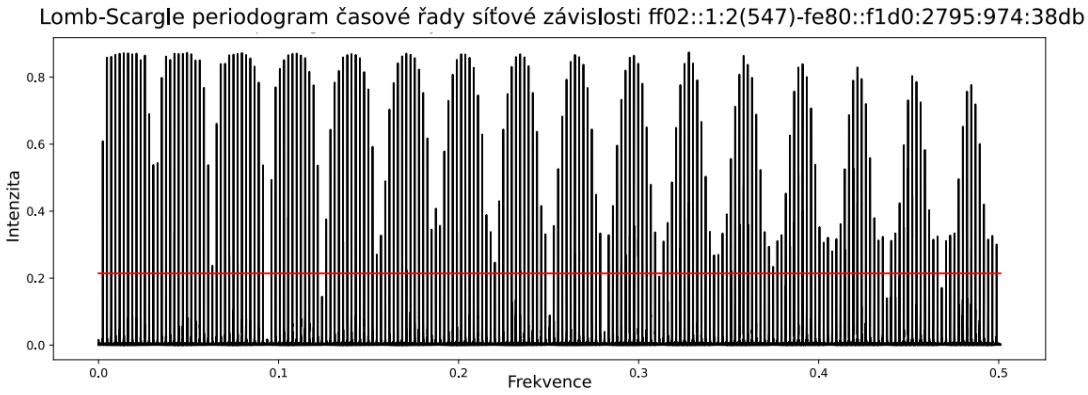
7. Identifikace období: Vrcholy v periodogramu Lomb-Scargle odpovídají potenciální periodicitě v datech. Nejvyšší vrcholy představují dominantní frekvence nebo období přítomné v časové řadě.

Analýzou Lomb-Scargleho periodogramu mohou vědci identifikovat periodické signály nebo skryté vzory v datech časových řad, a to i v přítomnosti šumu nebo nepravidelného vzorkování. Poskytuje mocný nástroj pro odhalování periodických jevů a studium jejich charakteristik v různých vědeckých oblastech. [38]

Na obrázku 3.1 je vidět příklad časové řady. Na obrázku 3.2 je pak LS periodogram z časové řady vygenerovaný. Na obrázku 3.3 je pak vidět přiblížení/detail LS periodogramu.



Obrázek 3.1 Časová řada ze síťové závislosti. Zdroj: Koumar Josef (2022)



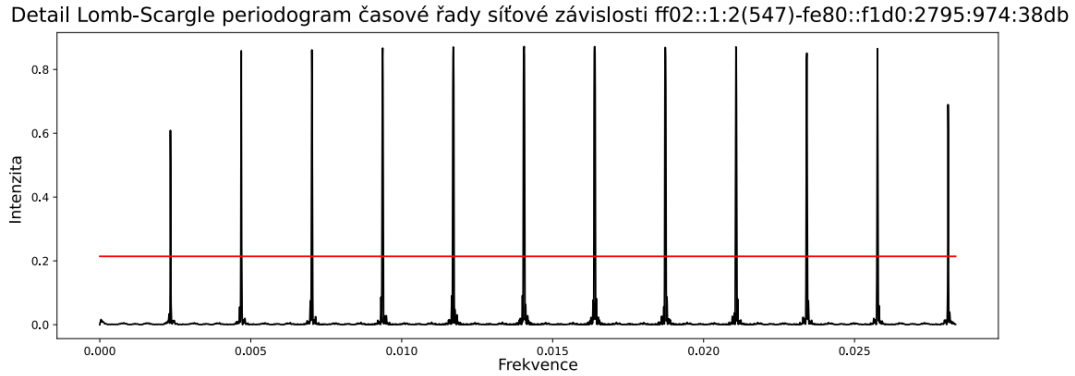
Obrázek 3.2 Lomb-Scargle periodogram. Zdroj: Koumar Josef (2022)

3.3.2 Získání vlastností časových řad

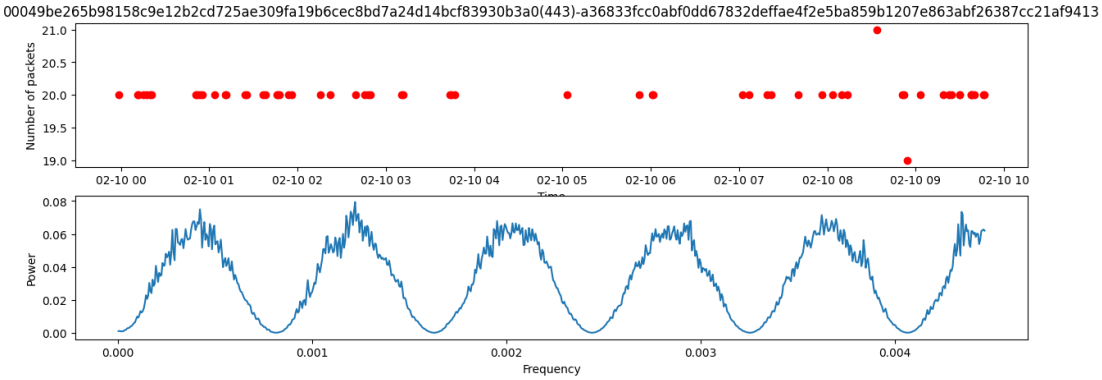
Jak je vidět na obrázcích 3.4 3.5 3.6 časové řady mohou být periodické různými způsoby. Některé menší časové řady mají periodicitu zřetelnou pouhým okem 3.4 3.5, u jiných není periodicita zřejmá 3.6. Velkým rozvolněním požadavků na periodicitu jsem vygeneroval i příklady neperiodických řad 3.7 3.8.

Získání periodických chování časové řady se získá pomocí algoritmu znázorněného na 3.9.

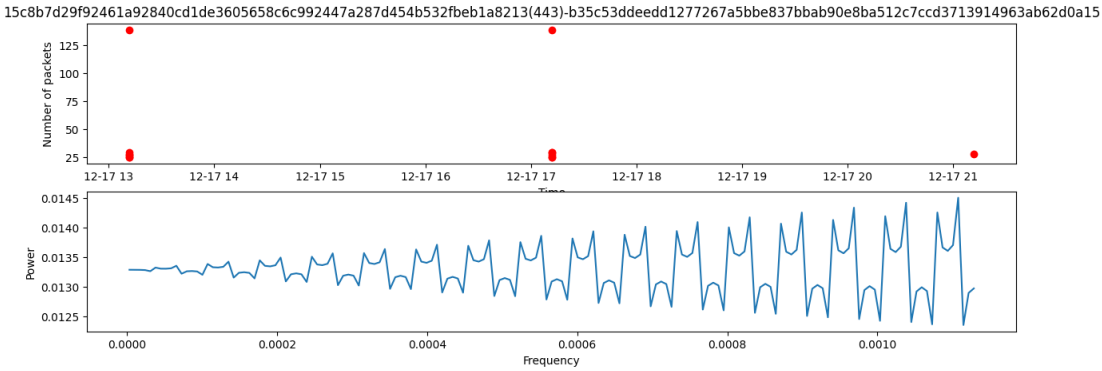
Kandidáty na periodicitu vybíráme tak, že ponecháme ve výsledku autokorelační funkce pouze takové body, které jsou lokální maxima. Posté jsou spočítány rozdíly indexu mezi těmito body a výsledkem jsou tzv. lagy. Pokud se nějaký lag objevuje častěji než je nastavené procento všech lagů, tak je lag předán jako kandidát na periodu [5].



Obrázek 3.3 Defail Lomb-Scargle periodogramu. Zdroj: Koumar Josef (2022)



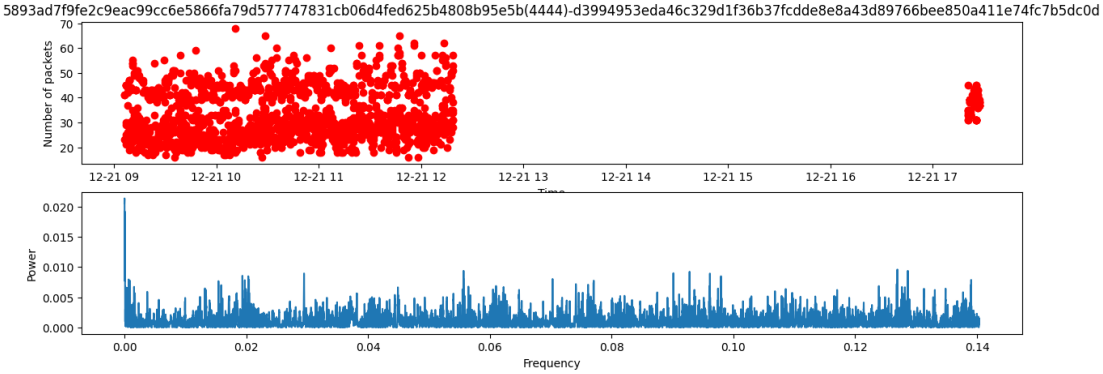
Obrázek 3.4 Příklad periodické časové řady 1



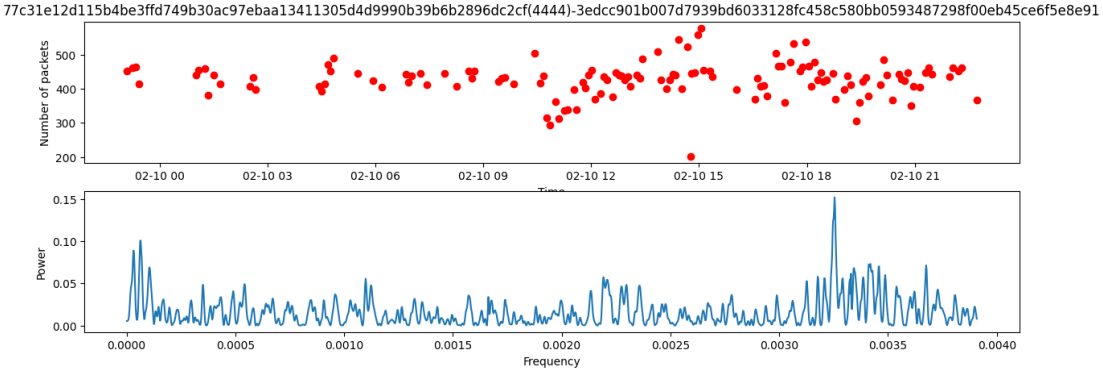
Obrázek 3.5 Příklad periodické časové řady 2

Po získání kandidátů následuje kontrola konstantnosti časové řady a to kvůli drobně nevýhodě Lomb-Scargle (LS) periodogramu, který ne vždy funguje správně na konstantních časových řadách. Test probíhá pomocí histogramu, u kterého hledáme, zda se nějaká hodnota opakuje ze zadaného procenta všech hodnot. Pokud časová řada nějaké metriky toto splňuje, je časová řada označena jako konstantní [5].

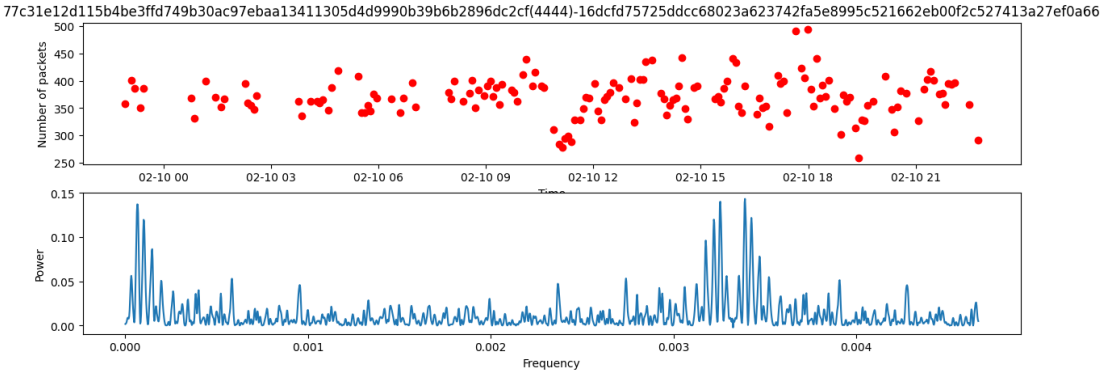
Následně je sestaven LS periodogram. Ten má oproti klasickému periodogramu dva benefity. Za prvé, distribuce šumu na každé jednotlivé frekvenci je chí-kvadrát distribuována



Obrázek 3.6 Příklad periodické časové řady 3



Obrázek 3.7 Příklad neperiodické časové řady 1

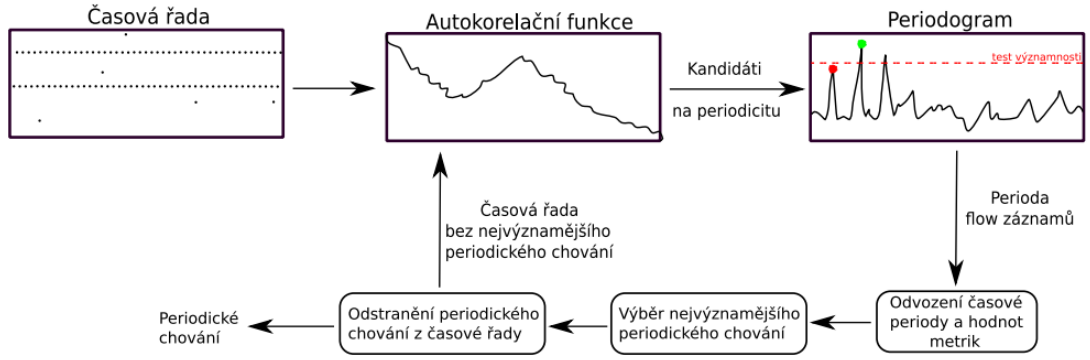


Obrázek 3.8 Příklad neperiodické časové řady 2

podle nulové hypotézy. Za druhé, výsledek je ekvivalentní periodogramu odvozenému z analýzy nejmenších čtverců. Příklad LS periodogramu je včetně časové řady použité na jeho vygenerování na obrázcích 3.1 3.2 3.3 [5].

Na takto vygenerovaný periodogram se aplikuje test pomocí Scarglovy kumulativní distribuční funkce, a spočítá se Spolehlivost statistických testů významnosti [5].

Z LS periodogramu je pak získán o potvrzení kandidáta na periodu p, jehož hodnotu nazýváme flow perioda. Ta říká, kolik flow záznamů se na časové řadě periodicky opakuje. Takže nám zbývá nalézt, jaké přesně hodnoty to jsou. Projdeme tedy časovou řadu každé metriky a hledáme



Obrázek 3.9 Diagram metody detekce periodických chování na časových řadách ze síťového provozu. Zdroj: Koumar Josef (2022)

nejčastěji se vyskytující posloupnosti délky p[5]. Z této posloupnosti se následně získají "power"a "frequency", které slouží k výpočtu spektrálních vlastností periodicity.

Provedení výše popsaného algoritmu je provedeno v jupyter notebooku `new_model.ipynb`, který byl poskytnut vedoucím práce. Tento skript vytvoří nový soubor nazvaný `periodicity_features.csv`, který obsahuje následující sloupečky definované v tabulkách 3.3 3.4.

SCDF test bere parametry `per_level` a `sig_level`, které reprezentují hranici přijetí kandidáta a které významně ovlivní množství nalezených periodických časových řad. Test konstantnosti bere parametr `constant_level`, který reprezentuje, jak hodně musí být časová řada konstantní, aby byla označena jako konstantní. Pro účely této práce jsou parametry ponechány na defaultních hodnotách.

Tímto postupem bylo v datech nalezeno 10853 periodických časových řad. Z toho jich 3633 patřilo botnetu a 7239 jich patřilo ostatnímu provozu.

Hodnoty mohou někdy obsahovat neplatné hodnoty: $+\inf$, $-\inf$, $Null$. To však není vhodné a proto byly nahrazeny defaultními hodnotami. Pro periodická vlastnosti a statistické vlastnosti je to dle instrukcí 0, pro frekvenční vlastnosti je to dle instrukcí -1.

Tabulka 3.3 Popis vlastností periodických časových řad pro detekci těžby kryptoměn.

Název vlastnosti	Typ	Popis
`id_dependency`	identifikátor	kombinace (cli-ip; srv-ip; cli-port; srv-port)
`label`	label	Buď MINER nebo Other.
`packet_value`	periodická	Počet paketů, mají-li periodicky se opakující toky stejný počet paketů.
`packet_value_x`	periodická	Dolní mez počtu paketů, mají-li periodicky se opakující toky různé počty paketů.
`packet_value_y`	periodická	Horní mez počtu paketů, mají-li periodicky se opakující toky různé počty paketů.
`packet_mean`	statistická	Průměr počtu paketů periodicky se opakujících toků.
`packet_std`	statistická	Směrodatná odchylka počtu paketů periodicky se opakujících toků.
`packet_skewness`	statistická	Koeficient šikmosti počtu paketů periodicky se opakujících toků.
`packet_kurtosis`	statistická	Koeficient špičatosti počtu paketů periodicky se opakujících toků.
`bytes_value`	periodická	Počet bajtů, mají-li periodicky se opakující toky stejný počet bajtů.
`bytes_value_x`	periodická	Dolní mez počtu bajtů, mají-li periodicky se opakující toky různé počty bajtů.
`bytes_value_y`	periodická	Horní mez počtu bajtů, mají-li periodicky se opakující toky různé počty bajtů.
`bytes_mean`	statistická	Průměr počtu bajtů periodicky se opakujících toků.
`bytes_std`	statistická	Směrodatná odchylka počtu bajtů periodicky se opakujících toků.
`bytes_skewness`	statistická	Koeficient šikmosti počtu bajtů periodicky se opakujících toků.
`bytes_kurtosis`	statistická	Koeficient špičatosti počtu bajtů periodicky se opakujících toků.
`duration_value`	periodická	Délka toků, mají-li periodicky se opakující toky stejné délky.
`duration_value_x`	periodická	Dolní mez délky toků, mají-li periodicky se opakující toky různé délky.
`duration_value_y`	periodická	Horní mez délky toků, mají-li periodicky se opakující toky různé délky.
`duration_mean`	statistická	Průměr délek toků periodicky se opakujících toků.
`duration_std`	statistická	Směrodatná odchylka délek toků periodicky se opakujících toků.
`duration_skewness`	statistická	Koeficient šikmosti délek toků periodicky se opakujících toků.
`duration_kurtosis`	statistická	Koeficient špičatosti délek toků periodicky se opakujících toků.
`difftimes_value`	periodická	Doba mezi toky, je-li doba mezi periodicky se opakujícími toky stejná.
`difftimes_value_x`	periodická	Dolní mez doby mezi toky, jsou-li doby mezi periodicky se opakujícími toky různé.
`difftimes_value_y`	periodická	Horní mez doby mezi toky, jsou-li doby mezi periodicky se opakujícími toky různé.
`difftimes_mean`	statistická	Průměr doby mezi periodicky se opakujícími toky.
`difftimes_std`	statistická	Směrodatná odchylka doby mezi periodicky se opakujícími toky.
`difftimes_skewness`	statistická	Koeficient šikmosti doby mezi periodicky se opakujícími toky.
`difftimes_kurtosis`	statistická	Koeficient špičatosti doby mezi periodicky se opakujícími toky.

Tabulka 3.4 Popis vlastností periodických časových řad pro detekci těžby kryptoměn - pokračování.

Název vlastnosti	Typ	Popis
`max_power`	frekvenční	Maximální síla LS periodogramu.
`max_frequency`	frekvenční	Frekvence maximální síly LS periodogramu.
`min_power`	frekvenční	Minimální síla LS periodogramu.
`min_frequency`	frekvenční	Frekvence minimální síly LS periodogramu.
`spectral_energy`	frekvenční	Celková energie přítomná na všech frekvencích v LS periodogramu.
`spectral_entropy`	frekvenční	Míra náhodnosti/neuspořádanosti v LS periodogramu.
`spectral_kurtosis`	frekvenční	Může indikovat nestacionární/negaussovské chování ve výkonovém spektru [39].
`spectral_skewness`	frekvenční	Koeficient šikmosti silového spektra LS periodogramu [40].
`spectral_rolloff`	frekvenční	Frekvence, pod kterou je soustředěno 85 % distribuční energie [41].
`spectral_cetroid`	frekvenční	Frekvence, na které je soustředěna energie spektra (těžiště energie) [42].
`spectral_spread`	frekvenční	Rozdíl mezi nejvyšší a nejnižší frekvencí v silovém spektru [40].
`spectral_slope`	frekvenční	Strmost trendu silového spektra v daném frekvenčním rozsahu [43].
`spectral_crest`	frekvenční	(aka flatness) Odhaduje rovnoměrnost distribuce energie signálu ve frekvenční oblasti [44].
`spectral_flux`	frekvenční	Rychlost změny síly LS periodogramu s rostoucí frekvencí [42].
`spectral_bandwidth`	frekvenční	Šířka spektrálního pásma. Popisuje rozdíl mezi horními a dolními frekvencemi, při kterých je spektrální energie poloviční než maximální hodnota [45].

Kapitola 4

Detekce těžby kryptoměn

4.1 Výběr vhodného klasifikátoru

Nejprve si řekneme, jak budeme evaluovat úspěšnost ML klasifikátoru (modelu). Odhady klasifikátoru rozdělíme do 4 kategorií.

1. Skutečně pozitivní (TP) - model správně označil časovou řadu jako MINER
2. Falešně pozitivní (FP) - model označil časovou řadu jako MINER, ale řada je OTHER
3. Skutečně negativní (TN) - model správně označil časovou řadu jako OTHER
4. Falešně negativní (FN) - model označil časovou řadu jako OTHER, ale řada je MINER

Z těchto hodnot bude vytvořena konfuzní matice, tak že na ose X bude zobrazeno predikce modelu, na ose Y bude zobrazen label [46].

Dále definujeme metriky úspěšnosti. Accuracy 4.1 je poměr skutečně pozitivních ke všem predikcím.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

Precision 4.2 je poměr skutečně pozitivních ku všem pozitivním predikcím.

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall 4.3 (v češtině senzitivita) je poměr skutečně pozitivních ku skutečně pozitivním + falešně negativním.

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

F1 skóre 4.4 kombinuje precision a recall:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precison + recall} \tag{4.4}$$

Pro výběr vhodného klasifikátoru byla data rozdělena na trénovací a testovací sadu. Každý klasifikátor byl vytvořen s defaultními parametry, natrénován na trénovacích datech a jeho úspěšnost vyzkoušena na testovacích datech. Každý klasifikátor měl stejnou sadu trénovacích

Tabulka 4.1 Výsledky výběru klasifikátoru.

Název klasifikátoru	accuracy	presision	recall	F1 skóre
RandomForestClassifier	99.23	99.65	98.05	98.84
HistGradientBoostingClassifier	99.30	99.31	99.20	99.25
BaggingClassifier	98.62	98.72	97.13	97.92
ExtraTreesClassifier	99.20	99.53	98.05	98.79
GradientBoostingClassifier	98.89	99.30	97.36	98.32
CatBoost	99.43	99.54	98.73	99.14
LogisticRegression	78.66	77.45	50.86	61.40
LinearDiscriminantAnalysis	75.21	60.02	77.04	67.47
GaussianNB	52.34	41.12	99.20	58.14
SVC	66.67	57.14	0.46	0.91
KNeighborsClassifier	91.26	88.78	84.50	86.59
DecisionTreeClassifier	98.24	97.14	97.59	97.37
AdaBoostClassifier	98.31	98.36	96.56	97.45
XGBClassifier	99.43	99.54	98.74	99.14

a testovacích dat. Byly sledovány výsledky accuracy, precision, recall a F1 skóre. Výsledky jsou v tabulce 4.1

Nejlepší výsledky měly klasifikátory HistGradientBoosting, CatBoost a XGBoost. Přes více běhů si na nalezených periodických časových řadách nejlépe vedl XGBoost a byl proto vybrán pro tuning parametrů, nalezení významných periodických vlastností.

4.2 Nalezení nejlepších parametrů pro vybraný klasifikátor

Pro výběr významných periodických vlastností časových řad, nejdříve získáme parametry pro vytvoření optimálního modelu XGBoost. Pro to použijme python modul hyperopt [47]. Funkce fmin ze zmíněného modulu který umožňuje nalézt minimum/maximum návratové hodnoty funkce pro sadu parametrů z definovaného prostoru. Pro klasifikátor budeme hledat maximum F1 skóre, které bere v potaz jak accuracy, tak recall.

Nejprve podle dokumentace XGBoost [48] zadefinujeme prostor validních hodnot parametrů 4.1.

Výpis kódu 4.1 Prostor parametrů XGBoost

```
space = {
        'max_depth': hp.randint("max_depth", 100),
    'gamma': hp.randint('gamma', 9),
    'reg_alpha' : hp.randint('reg_alpha', 120),
    'reg_lambda' : hp.randint('reg_lambda', 120),
    'colsample_bytree' : hp.uniform('colsample_bytree', 0, 1),
    'colsample_bylevel' : hp.uniform('colsample_bylevel', 0, 1),
    'colsample_bynode' : hp.uniform('colsample_bynode', 0, 1),
    'min_child_weight' : hp.randint('min_child_weight', 20),
    'n_estimators': hp.randint('n_estimators', 50),
    'subsample': hp.uniform('subsample', 0, 1),
    'eta': hp.uniform('eta', 0.005, 0.5)
}
```

Vytvoříme funkci `"tune_xgb_params"`, která danou podmnožinu vlastností časové řady najde optimální parametry pro vytvoření modelu XGBoost. Protože prostor validních hodnot je docela

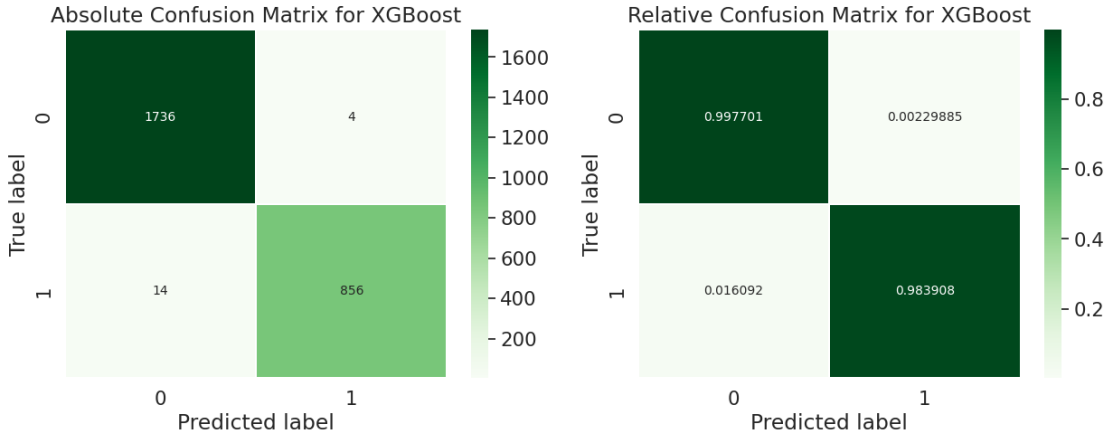
veliký, mohlo by nalezení optimálních hodnot vyžadovat veliké množství opakování. Proto nejprve jedním zavoláním funkce fmin získáme hodnoty parametrů, které mají nejlepší výsledek při zachování relativně malého, ale dostatečně velkého, množství opakování (parametr `"max_evals"` funkce fmin).

Následně budeme volat funkci fmin pro každý parametr zvlášť, kdy ostatní parametry budou zafixované na předchozí nejlepší nalezené hodnotě. To povede k jednorozměrnému prostoru parametrů a k větší šanci nalezení optimálního parametru při nižším počtu opakování. To sice neumožní nalézt novou nejlepší kombinaci parametrů, ale za předpokladu, že po prvním zavolání funkce fmin budou nalezené hodnoty již blízko optimálním hodnotám, umožní nalézt zlepšení jednotlivých parametrů.

Laděním parametrů pro všechny vlastnosti časové řady bylo dosaženo úspěšností zobrazených v tabulce 4.2. Konfuzní matice je zobrazena na obrázku —4.1. Sada vyladěných parametrů je v ukázce kódu 4.2

Tabulka 4.2 Úspěšnost modelu XGBoost při použití všech vlastností.

accuracy	presision	recall	F1 skóre
99.31	99.53	98.39	98.96



Obrázek 4.1 Konfuzní matice XGBoost modelu při použití všech vlastností.

Pro ověření vhodnosti hodnot parametrů použijeme Cross-validaci - StratifiedKFold (kfold) - z python modulu sklearn. Tato metoda rozdělí trénovací data na požadovaný počet podmnožin, čímž ověříme hodnoty pro různá trénovací data. Úspěšnost predikcí XGBoost s nalezenými hodnotami parametrů pro všechny vlastnosti časových řad je zobrazeno v tabulkách 4.3, 4.4. Z tabulek je patrné, že natrénovaný modul má velice dobrou úspěšnost.

Výpis kódu 4.2 Vybrané nejlepší parametry XGBoost

```
{
    'max_depth': 5,
    'gamma': 0,
    'reg_alpha': 0,
    'reg_lambda': 14,
    'colsample_bytree': 0.5936185327074481,
    'colsample_bylevel': 0.6634136251807683,
    'colsample_bynode': 0.89882360124486,
    'min_child_weight': 1,
```

Tabulka 4.3 Výsledky Cross-validace modelu XGBoost při použití všech vlastností.

accuracy	presision	recall	F1 skóre
99.38725490196079	100.0	98.16849816849816	99.07578558225507
99.38725490196079	98.90510948905109	99.26739926739927	99.08592321755026
99.1421568627451	99.6268656716418	97.8021978021978	98.70609981515712
99.26470588235294	98.54545454545455	99.26739926739927	98.9051094890511
99.50920245398774	99.63099630996311	98.9010989010989	99.26470588235294
99.87730061349693	100.0	99.63235294117648	99.8158379373849
98.77300613496932	98.88059701492537	97.42647058823529	98.14814814814815
99.87730061349693	100.0	99.63235294117648	99.8158379373849
99.38650306748467	99.62825278810409	98.52941176470588	99.07578558225507
99.14110429447854	99.625468164794	97.79411764705883	98.7012987012987

Tabulka 4.4 Statistiky Cross-validace modelu XGBoost při použití všech vlastností.

Cross-Validation metric	Cross-Validation score
accuracy	99.375 +/- 0.317
precision	99.484 +/- 0.496
recall	98.642 +/- 0.771
F1-score	99.059 +/- 0.479

```
    'n_estimators': 145,
    'subsample': 0.9623012469341181,
    'eta': 0.438210899643526,
}
```

4.3 Výběr významných periodických vlastností

Natrénovaný model nám umožňuje nahlédnout do toho, podle jakých vlastností se rozhoduje. V pythonu toho za použití modulů sklearn a pandas docílíme například tak, jak je ukázáno v ukázce kódu 4.3.

Výpis kódu 4.3 Získání významných periodických vlastností

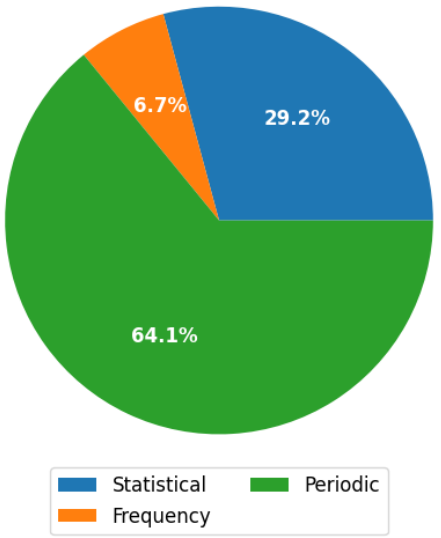
```
feat_importances = pandas.Series(best_model.feature_importances_,
                                 index=dataframe.columns)
feat_importances = feat_importances.sort_values(ascending=True)
```

Když zobrazíme významnost vlastností podle skupiny vlastností 4.2 vidíme, že nejdůležitější jsou periodická vlastností s 64.1 % rozhodování, následují statistické vlastnosti s 29.2 % rozhodování, a následují frekvenční vlastnosti s 6.7 % rozhodování.

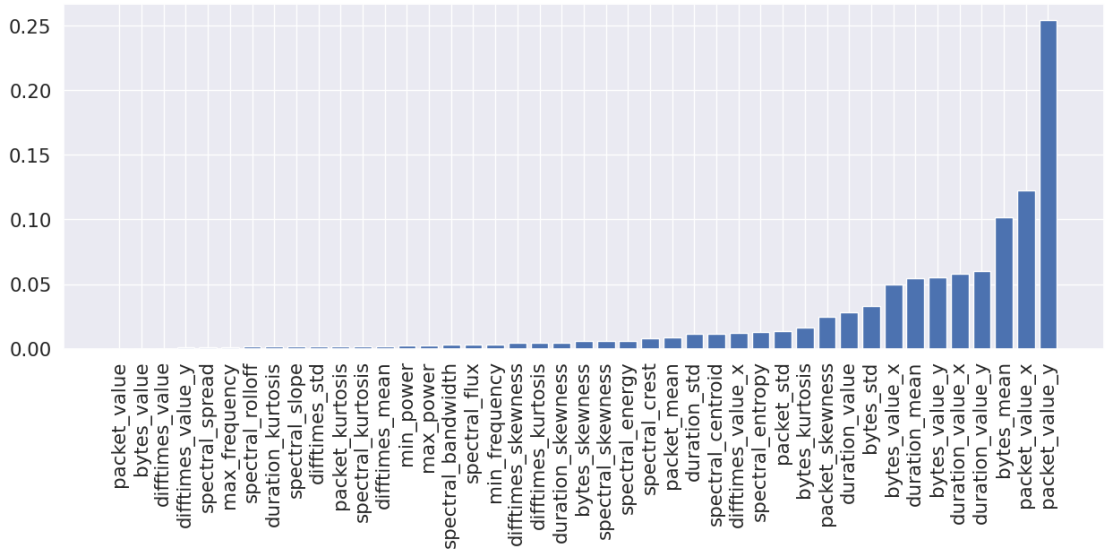
Když zobrazíme významnost jednotlivých vlastností 4.3 vidíme, že 10 nejdůležitějších vlastnosti jsou `packet_value_y`, `packet_value_x`, `bytes_mean`, `duration_value_y`, `duration_value_x`, `bytes_value_y`, `duration_mean`, `bytes_value_x`, `bytes_std`, `duration_value`.

4.4 Vyhodnocení nasaditelnosti prototypu do reálného provozu

Metoda detekce těžby kryptoměn s použitím periodických, statistických a frekvenčních vlastností časových řad síťových toků jako vstup pro strojové učení má dobrou úspěšnost, jak bylo ukázáno



Obrázek 4.2 Významnost skupin vlastností.



Obrázek 4.3 Významnost jednotlivých vlastností.

v minulé kapitole.

Použití zde popsaného a vytvořeného modelu bez dalších způsobů detekce ve vysoko objemových sítích by však navzdory tomu vedlo k relativně vysokému počtu falešně pozitivních výsledků, tento model sám o sobě tedy stále není vhodný pro automatické blokování provozu a při nasazení by stále vyžadoval lidský zásah.

Metoda je ale vhodná pro rozšíření DeCrypto systému od Plného [22]. Přesnost celého systému, která je už tak dobrá, by se přidáním statistických a periodických vlastností zlepšila. Toto rozšíření DeCrypto systému je plánováno jako navazující práce.

Kapitola 5
Závěr

Hlavním cílem této práce bylo spojit systém pro detekci těžby kryptoměn, kterou popsal Plný, s vlastnostmi síťových toků, které mohou být získány převodem na časové řady tak, jak popsal Koumar, a vyhodnotit životaschopnost a proveditelnost tohoto spojení.

Za tímto účelem byly moduly, které toto implementují, spojeny a rozšířeny. Tento postup byl popsán od vytvoření časových řad a následné získání periodických vlastností, přes výběr, trénink a ladění klasifikátoru strojového učení, po vyhodnocení výsledků a zvážení nasaditelnosti.

V budoucnu je plánováno zapojení vytvořeného modulu do již existujícího modulu DeCrypto, při němž bude definováno, jak tyto moduly zkombinovat dohromady pro získání nejlepších výsledků. V rámci této návazné práce bude výsledný modul nasazen do sítě CESNET2 s více než půl milionem uživatelů a vyhodnocena úspěšnost.

Spolu s prací jsou publikovány všechny data, informace a zdrojové kódy. Čtenář si tedy může pokus replikovat a ověřit.

Výsledky této práce budou ověřeny, sepsány a odeslány jako konferenční článek do 19th International Conference on Network and Service Management (CNSM) 2023.

Bibliografie

1. JOSHI, Chirag et al. ANN based Multi-Class classification of P2P Botnet. *International Journal Of Computing and Digital System*. 2021, s. 1319–1325.
2. AHMED, Abdulghani Ali et al. Deep Learning-Based Classification Model for Botnet Attack Detection. *J. Ambient Intell. Humaniz. Comput.* 2022, roč. 13, č. 7, s. 3457–3466. Dostupné z DOI: 10.1007/s12652-020-01848-9.
3. VU, Ly et al. Time Series Analysis for Encrypted Traffic Classification: A Deep Learning Approach. In: *ISCIT 2018*. IEEE, 2018.
4. GANDHI, Rishabh; LI, Yanyan. Comparing Machine Learning and Deep Learning for IoT Botnet Detection. In: *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2021, 234–239. ISSN 2693-8340. Dostupné z DOI: 10.1109/SMARTCOMP52413.2021.00053.
5. KOUMAR, Josef. *Detekce a rozpoznávání periodické komunikace v síťovém provozu*. 2022. Dipl. pr. Czech Technical University in Prague.
6. PLNÝ, Richard; HYNEK, Karel; ČEJKA, Tomáš. Datasets of Cryptomining Communication. In: Zenodo, 2022. Ver. 1.0. Dostupné z DOI: 10.5281/zenodo.7189293. Acknowledgements This research was funded by the Ministry of Interior of the Czech Republic, grant No. VJ02010024: Flow-Based Encrypted Traffic Analysis and also by the Grant Agency of the CTU in Prague, grant No. SGS20/210/OHK3/3T/18 funded by the MEYS of the Czech Republic.
7. NAKAMOTO, Satoshi. *Bitcoin: A Peer-to-Peer Electronic Cash System* [Online]. [B.r.]. Dostupné také z: https://bitcoin.org/bitcoin.pdf. (Accessed on 04/28/2023).
8. STROUKAL, Dominik; SKALICKÝ, Jan. *Bitcoin: peníze budoucnosti: historie a ekonomie kryptoměn, stručná příručka pro úplné začátečníky*. Ludwig von Mises Institut CZ&SK, 2015. ISBN 978-80-87733-26-4.
9. ZOHURI, Bahman; NGUYEN, Hang Thanh; MOGHADDAM, Masoud. *What is the Cryptocurrency? Is it a Threat to Our National Security, Domestically and Globally?* [Online]. 2022. ISSN 2767-3901. Dostupné také z: https://unisciencepub.com/storage/2022/02/What-is-the-Cryptocurrency-Is-it-a-Threat-to-Our-National-Security-Domestically-and-Globally.pdf. (Accessed on 04/28/2023).
10. BACK, Adam. *Hashcash - A Denial of Service Counter-Measure* [Online]. 2002. Dostupné také z: http://www.hashcash.org/papers/hashcash.pdf. (Accessed on 04/28/2023).
11. GHIMIRE, Suman; SELVARAJ, Henry. A Survey on Bitcoin Cryptocurrency and its Mining. In: *2018 26th International Conference on Systems Engineering (ICSEng)*. 2018, s. 1–6. Dostupné z DOI: 10.1109/ICSENG.2018.8638208.

12. TARMAN, Marko. *What is solo mining and how does it work?* [Online]. [B.r.]. Dostupné také z: https://www.nicehash.com/blog/post/what-is-solo-mining-and-how-it-works. (Accessed on 04/28/2023).
13. PROJECT, BITCOIN. *Bitcoin Developer* [Online]. [B.r.]. Dostupné také z: https://developer.bitcoin.org/devguide/mining.html. (Accessed on 04/28/2023).
14. HERTIG, Alyssa; LEECH, Ollie. *What Does Hashrate Mean and Why Does It Matter?* [Online]. 2021. Dostupné také z: https://www.coindesk.com/tech/2021/02/05/what-does-hashrate-mean-and-why-does-it-matter/. (Accessed on 04/28/2023).
15. KURKO, Michael. *Best Bitcoin Mining Software for 2023* [Online]. 2023. Dostupné také z: https://www.investopedia.com/best-bitcoin-mining-software-5095403. (Accessed on 04/28/2023).
16. COMMUNITY, bitcoin.it. *Getwork* [Online]. [B.r.]. Dostupné také z: https://en.bitcoin.it/wiki/Getwork. (Accessed on 04/28/2023).
17. SLUSHPOOL. *Stratum V1* [Online]. [B.r.]. Dostupné také z: https://braiins.com/stratum-v1. (Accessed on n 11/08/2021).
18. SLUSHPOOL. *Stratum V2 mining URLs and guide* [Online]. [B.r.]. Dostupné také z: https://help.slushpool.com/en/support/solutions/articles/77000423566-stratum-v2-mining-urls-and-guide. (Accessed on n 11/08/2021).
19. POOLIN.COM. *TLS protocol applied in Cryptocurrency mining* [Online]. 2021. Dostupné také z: https://medium.com/poolin/tls-protocol-applied-in-cryptocurrency-mining-9de15c08c405. (Accessed on 04/28/2023).
20. CERF, V.; KAHN, R. A Protocol for Packet Network Intercommunication. *IEEE Transactions on Communications*. 1974, roč. 22, č. 5, s. 637–648. Dostupné z DOI: 10.1109/TCOM.1974.1092259.
21. LEUNG, Ka-cheong; LI, Victor O.k.; YANG, Daiqin. An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges. *IEEE Transactions on Parallel and Distributed Systems*. 2007, roč. 18, č. 4, s. 522–535. Dostupné z DOI: 10.1109/TPDS.2007.1011.
22. PLNÝ, Richard. *Crypto-currency miner detection from extended IP flow data*. 2022. Bachelor's Thesis. Czech Technical University in Prague.
23. RESCORLA, Eric. *The Transport Layer Security (TLS) Protocol Version 1.3* [RFC 8446]. RFC Editor, 2018. Request for Comments, č. 8446. Dostupné z DOI: 10.17487/RFC8446.
24. MOCKAPETRIS, Paul. *Domain names - implementation and specification* [RFC 1035]. RFC Editor, 1987. Request for Comments, č. 1035. Dostupné z DOI: 10.17487/RFC1035.
25. WOLF, Karel. *Protokol Stratum V2 a budoucnost těžby bitcoinu - Lupa.cz* [Online]. [B.r.]. Dostupné také z: https://www.lupa.cz/clanky/protokol-stratum-v2-a-budoucnost-tezby-bitcoinu/. (Accessed on 04/28/2023).
26. GOODKIND, Andrew L.; JONES, Benjamin A.; BERRENS, Robert P. Cryptodamages: Monetary value estimates of the air pollution and human health impacts of cryptocurrency mining. *Energy Research & Social Science*. 2020, roč. 59, s. 101281. ISSN 2214-6296. Dostupné z DOI: https://doi.org/10.1016/j.erss.2019.101281.
27. PASTRANA, Sergio; SUAREZ-TANGIL, Guillermo. A First Look at the Crypto-Mining Malware Ecosystem: A Decade of Unrestricted Wealth. In: *Proceedings of the Internet Measurement Conference*. New York, NY, USA: Association for Computing Machinery, 2019, 73–86. IMC '19. ISBN 9781450369480. Dostupné z DOI: 10.1145/3355369.3355576.
28. CALDWELL, Tracey. The miners strike – addressing the crypto-currency threat to enterprise networks. *Computer Fraud & Security*. 2018, roč. 2018, č. 5, s. 8–14. ISSN 1361-3723. Dostupné z DOI: https://doi.org/10.1016/S1361-3723(18)30043-5.

29. LIU, Jingqiang; ZHAO, Zihao; CUI, Xiang; WANG, Zhi; LIU, Qixu. A Novel Approach for Detecting Browser-Based Silent Miner. In: *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*. 2018, s. 490–497. Dostupné z DOI: 10.1109/DSC.2018.00079.

30. LIU, Zhaoyan; CHEN, Binfa; LI, Zhen; LI, Hui; WANG, Dengzheng; LYU, Yang; GAO, Lu; HOU, Bingxu. Bitcoin Mining Recognition Based on Community Detection with Electricity Consumption Data. In: *2021 IEEE 5th Conference on Energy Internet and Energy System Integration (EI2)*. 2021, s. 3091–3096. Dostupné z DOI: 10.1109/EI252483.2021.9713449.

31. SWEDAN, AbedAlqader; KHUFFASH, Ahmad N.; OTHMAN, Othman; AWAD, Ahmed. Detection and Prevention of Malicious Cryptocurrency Mining on Internet-Connected Devices. In: *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*. New York, NY, USA: Association for Computing Machinery, 2018. ICFNDS '18. ISBN 9781450364287. Dostupné z DOI: 10.1145/3231053.3231076.

32. KHARRAZ, Amin; MA, Zane; MURLEY, Paul; LEVER, Charles; MASON, Joshua; MILLER, Andrew; BORISOV, Nikita; ANTONAKAKIS, Manos; BAILEY, Michael. Outguard: Detecting In-Browser Covert Cryptocurrency Mining in the Wild. In: *The World Wide Web Conference*. New York, NY, USA: Association for Computing Machinery, 2019, 840–852. WWW '19. ISBN 9781450366748. Dostupné z DOI: 10.1145/3308558.3313665.

33. MUÑOZ, Jordi Zayuelas i; SUÁREZ-VARELA, José; BARLET-ROS, Pere. Detecting cryptocurrency miners with NetFlow/IPFIX network measurements. In: *2019 IEEE International Symposium on Measurements & Networking (M&N)*. 2019, s. 1–6. Dostupné z DOI: 10.1109/IWMN.2019.8804995.

34. MAHESH, Batta. Machine Learning Algorithms - A Review. In: 2019. Dostupné z DOI: 10.21275/ART20203995.

35. NAVADA, Arundhati; ANSARI, Aamir Nizam; PATIL, Siddharth; SONKAMBLE, Balwant A. Overview of use of decision tree algorithms in machine learning. In: *2011 IEEE Control and System Graduate Research Colloquium*. 2011, s. 37–42. Dostupné z DOI: 10.1109/ICSGRC.2011.5991826.

36. CLAISE, B.; TRAMMELL, B.; AITKEN, P. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [Internet Requests for Comments]. RFC Editor, 2013-09. STD, 77. RFC Editor. ISSN 2070-1721. Dostupné také z: http://www.rfc-editor.org/rfc/rfc7011.txt. http://www.rfc-editor.org/rfc/rfc7011.txt.

37. TOUCH, Joe; LEAR, Eliot; ONO, Kumiko; EDDY, Wes; TRAMMELL, Brian; IYENGAR, Jana; SCHARF, Michael; TUEXEN, Michael; KOHLER, Eddie; NISHIDA, Yoshifumi. *Service Name and Transport Protocol Port Number Registry* [Online]. [B.r.]. Dostupné také z: http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.csv. Accessed: 2023-05-07.

38. SCARGLE, J. D. Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data. 1982, roč. 263, s. 835–853. Dostupné z DOI: 10.1086/160554.

39. VRABIE, Valeriu; GRANJON, Pierre; SERVIERE, Christine. Spectral kurtosis: from definition to application. 2003.

40. PEETERS, Geoffroy. A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *CUIDADO Ist Project Report*. 2004, roč. 54, č. 0, s. 1–25.

41. HUSSAIN, Md Gulzar et al. *Classification of Bangla Alphabets Phoneme based on Audio Features using MLPC & SVM*. 2021. Dostupné z DOI: 10.13140/RG.2.2.21013.04320.

42. SCHEIRER, Eric; SLANEY, Malcolm. Construction and evaluation of a robust multifeature speech/music discriminator. In: *1997 IEEE international conference on acoustics, speech, and signal processing*. IEEE, 1997, sv. 2, s. 1331–1334.

43. LERCH, Alexander. *An introduction to audio content analysis: Applications in signal processing and music informatics*. Wiley-IEEE Press, 2012.

44. BOASHASH, Boualem. *Time-frequency signal analysis and processing: a comprehensive reference*. Academic press, 2015.

45. SZABO, Thomas L. 5 - TRANSDUCERS. In: SZABO, Thomas L. (ed.). *Diagnostic Ultrasound Imaging*. Burlington: Academic Press, 2004, s. 97–135. Biomedical Engineering. ISBN 978-0-12-680145-3. Dostupné z DOI: https://doi.org/10.1016/B978-012680145-3/50006-2.

46. RASCHKA, Sebastian. An overview of general performance metrics of binary classifier systems. *arXiv preprint arXiv:1410.5330*. 2014.

47. BERGSTRA, James; YAMINS, Daniel; COX, David. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: *International conference on machine learning*. PMLR, 2013, s. 115–123.

48. DEVELOPERS, xgboost. *Python API Reference — xgboost 1.7.5 documentation* [Online]. 2022. Dostupné také z: https://xgboost.readthedocs.io/en/stable/python/python_api.html\#xgboost.XGBRFClassifier. (Accessed on 05/09/2023).

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF