FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Rozšíření reputační databáze o informace z Passive DNS
Student:	Maxmilián Tomáš
Vedoucí:	Ing. Tomáš Čejka
Studijní program:	Informatika
Studijní obor:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Prostudujte technologii DNS a seznamte se s existujícími databázemi s historií překládaných doménových jmen a síťových adres, tzv. Passive DNS. Seznamte se se systémem pro evidenci reputace síťových entit Network Entity Reputation Database (NERD) [1]. Pro testovací účely vytvořte jednoduchý prototyp úložiště překládaných doménových a síťových adres, který bude umět zpracovávat data z open source exportéru síťových toků systému NEMEA [2,3]. Rozšiřte systém NERD o podporu získávání informací k entitám z prototypu Passive DNS úložiště. Otestujte upravenou verzi systému NERD a změřte výkon systému po provedených úpravách.

Seznam odborné literatury

[1] http://nerd.cesnet.cz
[2] T.Cejka, et al.: "NEMEA: A Framework for Network Traffic Analysis," in *12th International Conference on Network and Service Management (CNSM 2016)*, Montreal, Canada, 2016.
[3] https://github.com/CESNET/NEMEA

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 15. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Rozšíření reputační databáze o informace z PassiveDNS

Maxmilián Tomáš

Vedoucí práce: Ing. Tomáš Čejka

15. května 2018

Poděkování

Chtěl bych poděkovat svému vedoucímu, kterým byl Ing. Tomáš Čejka, za jeho rady, pravidelné konzultace, trpělivost, ochotu a motivaci pro dokončení této práce. Zároveň bych chtěl poděkovat celému týmu CESNET za jejich ochotu, rady a nápady, které obohatily PassiveDNS.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. § 46 odst. 6 tohoto zákona 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 i množstevně 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 15. května 2018 .....

Odkaz na tuto práci

Tomáš, Maxmilián. *Rozšíření reputační databáze o informace z PassiveDNS.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Systém DNS (Domain Name System) je systém doménových jmen pro překlad mezi doménovými jmény a IP adresami. Data ze systému DNS je možné také využít v oblasti síťové bezpečnosti. Mohou pomoci blokovat šíření malwaru, odhalit nakažené stroje nebo rozšířit blacklisty o škodlivé domény. Výstupem této práce je systém pro ukládání historie mapování doménových jmen a IP adres. Navržený systém PassiveDNS importuje data ze systému DNS, která jsou zachycená z reálné síťové komunikace. Importovaná data jsou uchovávána v agregované formě, aby nedocházelo k plýtvání s hardwarovými zdroji. Rozhraní systému umožňuje přístup k historii překladu jednotlivých doménových jmen na konkrétní IP adresy. Systém může pomoci detekčním systémům rozšířit jejích vlastní databáze. Vzniklý systém je integrován do souvisejících projektů sdružení CESNET z.s.p.o.

Klíčová slova DNS, Passive DNS, malware, síťová bezpečnost, blacklist

Abstract

DNS (Domain Name System) is a domain name system for translation between domain names and IP addresses. Collection of data from DNS system can be useful for network security. It can help block malware spreading, detect infected hosts, or expand blacklists with malicious domains. The result of this thesis is a system for saving the history of mapping of domain names and IP addresses. The proposed PassiveDNS system imports data from DNS system that are captured from real network communications. Imported data is stored in an aggregate form to avoid the depletion of hardware resources. The system interface allows to access the translation history between individual domain names and specific IP addresses. The system can help detection systems to extend their own databases. The resulting system is integrated into the related projects developed by CESNET a.l.e.

Keywords DNS, Passive DNS, malware, network security, blacklist

Obsah

Seznam obrázků

Seznam tabulek

Úvod

DNS (Domain Name System) je systém doménových jmen pro stroje, služby či jiná zařízení na síti. Je realizovaný pomocí nameserverů, které umožňují lokální správu dat a zároveň poskytující data napříč celou sítí. Slouží obousměrnému překladu mezi doménovými jmény (např. fit.cvut.cz) a IP adresou, což ho činí prakticky nezbytným protokolem pro rychlou a pohodlnou komunikaci. Přesný způsob komunikace je definován v protokolu stejného jména.

Typickým rysem DNS systému je značná proměnlivost záznamů. Záznam tvoří dvojici doménové jméno a IP adresz.. Tedy překlady domén se mohou v čase měnit. To znamená, že pro nové služby vzniknou nové adrey nebo se některé změní v rámci konfigurace služby. PassiveDNS slouží k průběžnému ukládání těchto záznamů do databáze, aby se zachoval historický kontext. PassiveDNS tak umožní například, dohledat na jaké IP adresy se přeložila daná doména v čase. Právě tato možnost umožňuje přínos PassiveDNS pro analýzu aktivity škodlivého kódu. Malware v sobě velmi často obsahuje doménové jméno stroje útočníka. Tento stroj slouží jako kontrolní stanice pro zasílání příkazů nebo jako zdroj odkud se malware stáhne do napadeného stroje. Pro zajištění komunikace provede malware dotaz na doménové jméno. Jednou z možných obran je blokování dané domény, což výrazně sníží nebezpečnost malweru.

Blokování konkrétní domény není vždy možné a je vhodné lokálně filtrovat IP adresy, na které se doména přeložila. Odpověď na otázku, na jakou doménu se malware dotázal, nám může poskytnout forenzní analýza a s pomocí PassiveDNS dokážeme zpětně nalézt všechny IP adresy spojené s touto doménou. Zjištěné IP adresy je pak možné nechat lokálně filtrovat. Bez PassiveDNS by bylo daleko obtížnější IP adresy dohledat.

Systém Network Entity Reputation Database (NERD) je reputační databáze, která si udržuje reputační skóre [1] o podezřelých IP adresách. Reputační skóre vyjadřuje pravděpodobnost, že z dané IP adresy bude proveden útok v blízké budoucnosti. Skóre je spočítané na základě nahlášených

informací. Jedná se pozorované a nahlášené útoky a bezpečnostní incidenty. Informace z PassiveDNS mohou představovat užitečné doplnění. Proto byly informace z PassiveDNS zpřístupněny uživatelům systému NERD přímo přes jeho webové rozhraní.

Cílem práce je navrhnout a realizovat prototyp systému PassiveDNS, který získává informace ze zachycených zpráv protokolu DNS. Součástí práce bude navrhnout efektivní způsob ukládání dat a navrhnout implementaci API sloužící třetím stranám k získání informací uloženým v databázi systému PassiveDNS.

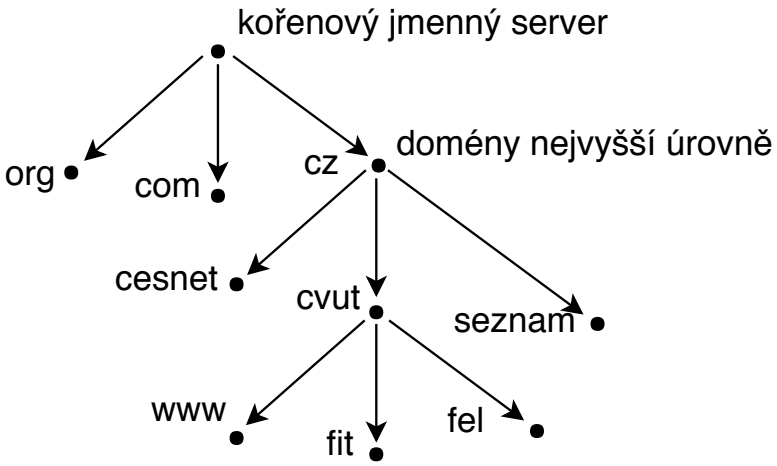
První kapitola této práce představuje systém DNS včetně DNS protokolu a definuje klíčové pojmy, které s DNS souvisejí. V druhé kapitole jsou podrobně analyzovány požadavky na systém PassiveDNS a rovněž je zde diskutován celkový koncept a návrh systému PassiveDNS. Třetí kapitola popisuje implementaci systému PassiveDNS a dalších činností, které souvisejí s pilotním nasazením a testováním navrženého systému. Poslední kapitola se věnuje testování s důrazem na testování výkonu pro import dat.

Kapitola 1

Domain Name System

Tato část se věnuje vysvětlení pojmů spojených s DNS systémem a DNS protokolem a následně zmín využití PassiveDNS. V závěru kapitoly jsou sepsány požadavky práce.

1.1 Domain Name System



Obrázek 1.1: Struktura DNS

DNS má stromovou hierarchii, kde jednotlivé uzly jsou označené jako unikátní doménová jména. Jméno kořenového uzlu je tečka. Konkrétní posloupnost domén oddělených tečkou je pak určitý podstrom v rámci celé struktury. Obrázek 1.1 ilustruje tuto strukturu.

Data, která přísluší určitým doménám jsou uložena v záznamech. Tyto záznamy jsou rozděleny podle tříd. Nejpopulárnější třídou je třído typu internet. V rámci třídy jsou záznamy rozděleny na několik typů. Každý typ

určuje odlišnou informaci. Data v záznamu je možné obdržet prostřednictvím DNS nameserveru. DNS nameserver je v rámci své domény zodpovědný za poskytnutí správné informace.

Proces přeložení probíhá tak, že resolver se dotáže lokálního DNS nameserveru (či jiného nameserveru jenž je určen konfigurací) na překlad konkrétní domény. Dotazovaný nameserver zkontroluje zda nemá platnou položku s dotazem v cache. Pokud ne, vyšle dotaz na kořenové nameservery. Kořenové servery si udržují informace o všech doménách nejvyšší úrovně. Dotaz je z kořenového uzlu postupně delegován na sousední nameservery v podstromu. Každý takto dotazovaný nameserver následně zjistí zda má k dotazu správnou odpověď nebo dotaz opět deleguje dál. Dotaz nakonec narazí na nameserver, který je schopen poskytnou odpověď k dotazu odpověď nebo vrátí informaci, že předmět dotazu neexistuje. Následující pojmy jsou definovány ve specifikaci RFC 1034 [2].

1.1.1 Základní pojmy

- **Jmenný prostor:** Doména identifikuje konkrétní vrchol. Tento vrchol disponuje určitým souborem záznamů, který může být prázdný.

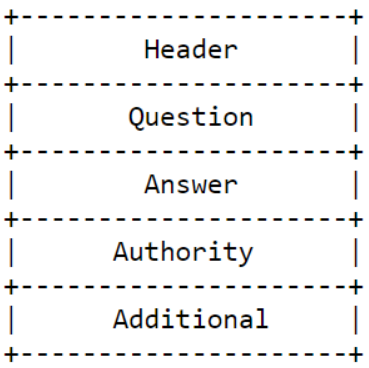
- **Zóna** je strukturovaný textový soubor, který obsahuje informace v rámci jmenného prostoru.

- **Nameserver** je program na straně DNS serveru, který udržuje informace konkrétního jmenného prostoru a jeho zdroje. Nameserver typicky má znalost kde vyhledat všechny informace v rámci svého jmenného prostoru a ukazuje na další nameservery, které mají k dispozici tyto informace z dalších částí podstromu. Pokud má nameserver kontrolu nad daty v rámci své větve stromu, je tento nameserver autoritativní pro tyto informace. Informace se evidují v jednotkách, zvané zóny. Nameserver poskytne odpověď na DNS dotaz v případě, že má kontrolu nad zónou, do které dotaz spadá. V opačném případě se rekurzivně dotáže ostatních nameserverů zda oni nemají odpověď k dispozici.

- **Resolver** je program, který obdrží informace z nameserveru jako odpověď na dotazy klientských programů, kteří potřebují získat informaci z DNS. Může se jedna o webové prohlížeče nebo třeba emailové klienty atd. Klienti typicky komunikují s resolverem přes systémové volání. Resolver musí mít být schopen komunikovat aspoň s jedním nameserverem, aby byl schopen odpovídat. Resolver je typicky systémová knihovna. Pro vyšší výkon, resolver napřed zkontroluje cache, zda již neobsahuje odpověď na nové dotazy. Pokud resolver dokáže komunikovat s nameservery napříč celým DNS systémem je označen jako rekurzivní.

1.1.2 Formát DNS zprávy

DNS dotazy a odpovědi se přenáší ve standardním formátu popsaném v RFC 1035 [3]. Zmíněný článek je také zdrojem pro obrázky v této sekci. Formát je složen z povinné hlavičky, která má fixní počet atributů a dalších 4 sekcí, které obsahují dotaz a zdroje. Formát hlavičky a formát zdrojového záznamu v této sekci též čerpají z RFC 1035 specifikace.



Obrázek 1.2: Formát DNS zprávy

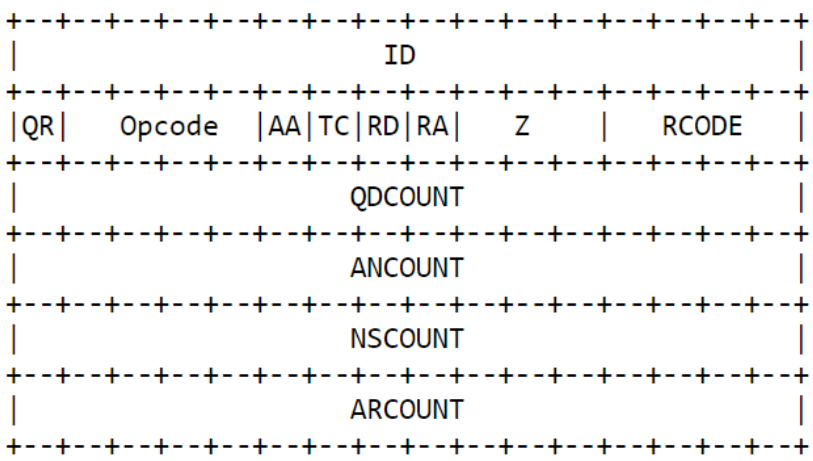
- `Header`: Hlavička obsahuje atributy, které značí jaké sekce jsou ve zprávě přítomny a specifikují zda se jedná o odpověď nebo dotaz.
- `Question`: Atributy, které popisují jaký dotaz byl nameserveru položen.
- `Answer`: Seznam záznamů s odpověďmi na dotaz v sekci Question.
- `Authority`: Seznam záznamů s jménem autoritativního nameserveru.
- `Additional`: Seznam záznamů s doplňujícími informaci k dotazu.

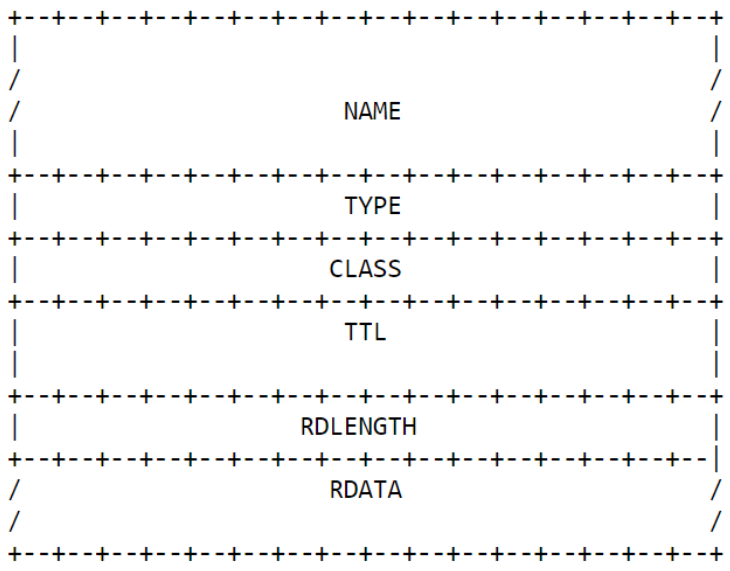
Na obrázku 1.3 je uveden formát DNS hlavičky. Hlavička obsahuje důležitá metadata pro práci se zprávou.

Ta část DNS zprávy, která nese samotná data je označená jako zdrojový záznam. Na Obrázku 1.4 je zobrazen formát a jednotlivé sekce.

- `Name`: Doménové jméno, kterého se záznam týká
- `TYPE`: Typ DNS záznam. Tento atribut určuje obsah sekce RDATA.
- `CLASS`: Specifikuje třída v sekci RDATA
- `TTL`: Číslo, které určuje životnost záznamu v sekundách. Toto číslo říká jak dlouho může být záznam uložen v cache, než by měl být odstraněn.

Obrázek 1.3: Hlavička DNS zprávy





Obrázek 1.4: Formát zdrojového záznamu

- `RDLENGHT`: Označuje délku sekce RDATA v bitech.
- `RDATA`: Sekvence bitů, která popisuje odpověď.

Obsah sekce RDATA je určen kombinací atributů `TYPE` a `CLASS`. Následuje výčet jaký druh dat může zdrojový záznam obsahovat. Uvedené příklady popisují zdrojový záznam s právě daným typem. Příklad má strukturu jako záznam v zóně.

- `CNAME`: Alias pro jiné doménové jméno. Využívá se pro usnadnění konfigurace zóny pokud více doménových jmen má přidělenou stejnou IP adresu. Příklad: `www.cvut.cz. 6400 IN CNAME cvut.cz.`
- `A`: Určuje pro doménové jméno překlad na konkrétní IPv4 adresu. Příklad: `cvut.cz. 6400 IN A 147.32.3.202`
- `AAAA`: Obdoba A záznamu ve formátu IPv6. Příklad: `ns.cvut.cz. 6400 IN AAAA 2001:718:2:2200::100`
- `PTR`: Inverzní obdoba A záznamu. Pro IP adresu uvádí namapované doménové jméno. Příklad: `248.232.32.147.in-adr.arpa. 86390 IN PTR www.fit.cvut.cz.`
- `MX`: Záznam uvádí název emailového serveru, který v dané doméně obsluhuje poštu Příklad: `cvut.cz. 6400 IN MX 10 mailgw10.cvut.cz.`

1.2 Využití PassiveDNS

První zmínka o PassiveDNS pochází z článku [4] **PassiveDNS Replication**, který napsal Florian Weimer v roce 2005. Tato práce si silně inspiruje tímto článkem. V článku diskutoval motivaci pro využití PassiveDNS v boji proti malweru a jeho vztah k DNS systému.

DNS komunikace může výrazně pomoci k detekce malware na sítě a zabránit jeho šíření. Většina malweru spoléhá na DNS službu, aby mohla obdržet IP adresu kontrolní stanice. Analýza DNS komunikace může pomoci odkrýt vazby mezi doménovým jménem kontrolní stanice a IP adresami, na které se daná doména přeložila.

Různých řešení PassiveDNS není příliš mnoho, především kvůli hardwarovým požadavkům, které PassiveDNS vyžaduje. DNS informací je neskutečně mnoho a jejich ukládání (i když v agregovaného podobě) je paměťově náročné. Nicméně je možné nalézt několik využití PassiveDNS ať už v podobě aplikace nebo určitého přístup k pozorování dat DNS komunikace.

Jako konkrétní příklad slouží například monitorovací systém [5] **DNSSM**. Toto řešení tak i řešení této práce navazuje na zmiňovaný článek od Floriana Weimera. Dalším příkladem může být PassiveDNS řešení od Farsight Security [6].

1.3 Požadavky

Požadavkem této bakalářské práce je vytvoření systému PassiveDNS. Od PassiveDNS je vyžadováno, aby bylo schopné dlouhodobě importovat DNS data a bude moci tyto informace poskytnout klientské straně. Dalším požadavkem je rozšíření systému Network Entity Reputation Database.

Rozšíření systému NERD proběhne ve dvou bodech (bude popsáno v sekci 2.4):

- Vytvoření modulu, které pomůže s blokací doménových jmen, které se přeložily na podezřelé IP adresy.
- Rozšíření webového rozhraní systému NERD, aby se uživateli zobrazovaly informace z PassiveDNS.

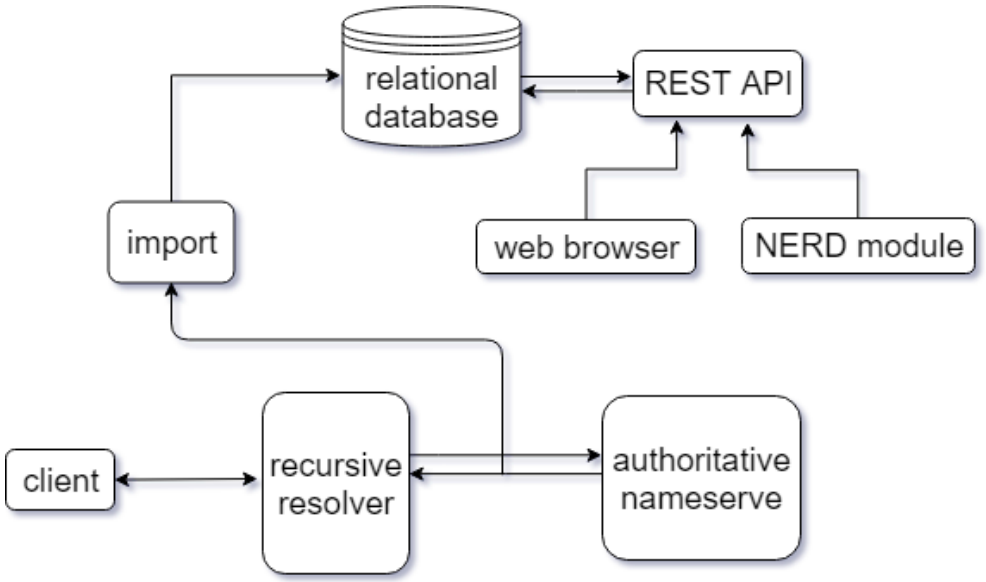
Kapitola 2

Návrh

Tato kapitola se věnuje popisu množných návrhů na vyřešení požadavků pro systém PassiveDNS. Mezi očekávané požadavky patří:

- Efektivní ukládání dat z hlediska paměťově náročnosti.
- Vybrat zdroj odkud budou data importovány do PassiveDNS a zajistit následný import dat
- Navržení struktury API pro dotazování na data ze strany klientů
- Integrace informací z PassiveDNS do systému NERD

Obrázek 2.1: Architektura PassiveDNS



Obrázek 2.1 zobrazuje obecný návrh architektury PassiveDNS a označuje zapojené komponenty. Data přicházejí ze zachycené komunikace při dotazu na DNS. Příchozí data jsou zpracována a importována do relační databáze. Navržené API umožní klientům přistuovat k datům uložené v databázi.

2.1 API

Účelem API bude poskytnou klientské straně prostředek jak položit dotaz na konrétní zdroj, jehož historii chce klient znát. Navržená struktura umožní položit dotazy jak na IP adresy, tak doménová jména. Odpověď bude vracena tak, aby odpovídala formátu navrženém pro uložení dat.

2.2 Návrh databáze

Důležitým požadavkem na PassiveDNS je efektivní ukládání dat z hlediska paměti. Zároveň je nutné zachovat co největší přesnost informací. To znamená uchovat k jakým dotazům patří jaké odpovědi a vědět v jakém časovém okně konkrétní dvojice dotaz-odpověď existovala.

Důvodem je, že PassiveDNS je určeno, aby neptřeržitě běželo a importovalo data. Není možné dopředu předpovědět jaká data budou užitečná a jaká budou zbytečná. Proto je žádoucí uchovat všechna zachycená data. Je tedy možné předpokládat, že objem dat bude velmi vysoký. Napřed je potřeba zvolit vhodnou reprezentaci dat.

Jeden přístup je nechat uložit každou DNS zprávu jako samostatný záznam s časovou značkou. Pro každou dvojici dotaz-odpověď je tak možné dohledat v jakém přesně čase byl dotaz položen. To je vhodné například pro domény, které velmi často mění přiřazenou IP adresu. Nevýhodou je riziko vysoké redundance u dvojici, na které se uživatelé často ptají. Dalším řešením je agregovaný formát uvedený v Tabulce 2.1.

Tabulka 2.1: formát dat v PassiveDNS

název atribut	význam atributu
`count`	počet výskytů (zachycených paketů) dvojice dotaz/odpověď
`request`	data dotazu
`response`	data odpovědi
`time_first`	Časový údaj o prvním zachycení
`time_last`	Časový údaj o posledních zachycení
`type`	Řetězec s názvem typu DNS záznamu, tedy zda se jedná o typ A, AAAA nebo CNAME

Z hlediska PassiveDNS je důležitá informace `request` a `response`. Tato dvojice bude klíčem, který bude od sebe záznamy odlišovat. Určité časové okno uživatel získá dle `time_first` a `time_last` parametrů a atribut `count` udává velmi hrubou frekvenci výskytů konkrétní dvojice dotaz-odpověď. Tatkto se sníží počet opakování záznamů se stejnou dvojicí dotaz-odpověď a zároveň je zachováno určité časové okno.

Nevýhodou tohoto formátu je, že u často měnících se dvojic by mohlo dojít k časovému překrytí. K může dojít například tak, že pro dotaz existuje odpověď. Odpověď se časem změní a po čase se opět změní na původní. Atributy u dotazu a původní odpovědi budou překrývat časové okno u dotazu se změněm odpodvědí. Nicméně tato nevýhoda nepřeváží paměťovou úsporu, kterou nabízí agregovaným formátem. Proto při implementace bude použit agregovaný formát.

Takto vytvořený formát bude použit pro rozložení sloupců v tabulkách. Samotné schéma relační databáze je navrženo tak, že pro každý typ DNS odpovědi je určena samostatná tabulka. Druhou možností je ukládání všech typu do jedné tabulky.

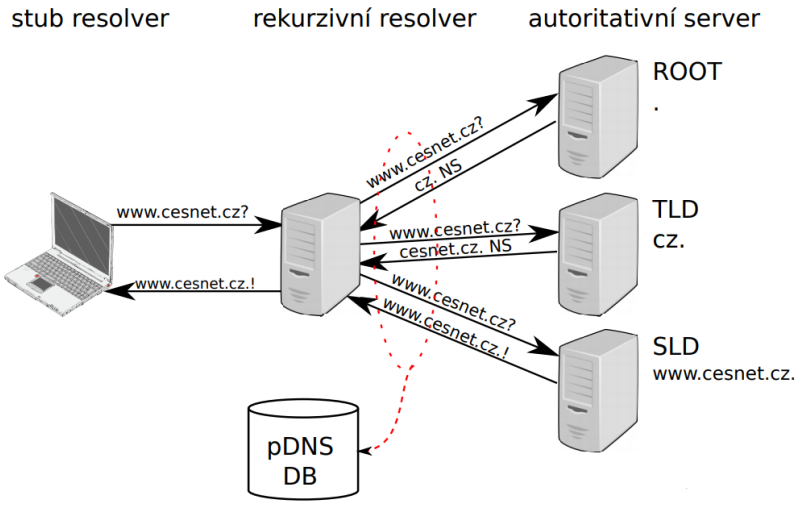
Výhodou prvního přístupu je možnost určit dedikované datové typy dle databázového engine a nabízí lehce jednoduší implementaci. Při psaní dotazů víme přesně, které tabulky se dotázat. Proto se při implementaci uplatnila první možnost.

2.3 Sběr dat

Autor článku [1] diskutuje možné zdroje dat pro PassiveDNS s ohledem na jejich výhody a nevýhody:

- Průběžně stahovat zónové soubory z nameserverů a ukládat jejich obsah. Zde je problém z hlediska přístupu k dané zóně.
- Přímé dotazování lokálních DNS nameserverů a logování jejich záznamů. Nevýhodou je, že je potřeba dopředu vědět na co se zeptat. Není tedy příliš škálovatelné.
- Logování ze strany resolveru (BIND). Nevýhodou je zásah do stávající konfigurace, zvýšení režie na samotné logování a veliký nárůst redundantních dat. Navíc lokální logování by znamenalo prozrazení lokální IP adresy a tedy by se porušilo soukromí.
- Sběr dat na úrovni DNS paketů s autoritativní odpověď. Zde existují dva přístupy:
  - Zachytávání před rekurzivním serverem. Tento přístup umožní zachytit všechny dotazy ze strany klientů.

– Zachytávání za rekurzivním serverem. Tento přístup zachytí méně dat kvůli cache. Nabízí implicitní soukromí klientů, protože nezachytává kdo se ptá, ale na co se lidé ptají.



Zdroj: [7]

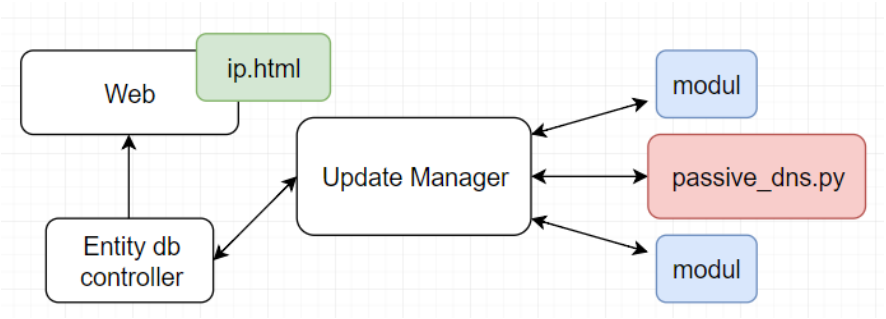
Obrázek 2.2: Sběr dat za resolverem

Pro sbírání dotazů je zvoleno zachytávání zpráv za resolverem. Důvodem je zachování soukromí uživatelů a zároveň takto nedojde k zásahu do konfigurace nameserverů. Navíc díky cache nebude docházet tak často k zachycení stejných zpráv. Obrázek 2.2 ilustruje místo zachycení. Na začátku resolver klienta vyšle dotaz na jemu známý nameserver. Pokud cache nameserveru obsahuje odpověď na dotaz, nepokračuje v dotazování, ale rovnou vrátí odpověď, takže zpráva nebude zachycena. V opačném případě sám nameserver vyšle dotaz na kořenové nameservery. Dotaz se rekurzivně vrací k resolveru dokud není vracen od autoritativního nameserveru. V tento moment dojde k zachycení zprávy.

Pro tento účel budou poskytnuty vybrané DNS nameserverů pod správou sdružení CESNET. V tomto místě budou pakety odpovědí autoritativních nameserverů. Pakety se pak zachytí a exportují do `pcap` formátu. Pakety jsou následně komprimovaný a poslány do vyhrazeného adresáře Na straně PassiveDNS příslušný modul hlídá vyhrazený adresář pro příchozí soubory. Modul obsah `pcap` souboru rozdělí na jednotlivé pakety a extrahuje příslušné atributy. Získané atributy budou předány databázové vrstvě ke zpracování.

2.4 Rozšíření systému NERD

Systém NERD je reputační databáze síťových entit (IP adresy, domény, lokální sítě atd.). Udržuje seznam známých zdrojů škodlivých aktivit na internetu a další dodatečné informace, které souvisejí s danou entitou. Zdroje jsou agregovány podle IP adresy. Dle aktivity mají evidované IP adresy určené reputačního skóre. Reputační skóre vyjadřuje nebezpečnost dané entity. Čím vyšší, tím závažnější. NERD obdrží bezpečností událost ze systému [8] WARDEN. WARDEN je systém pro sdílení a distribuci údajů detekovaných událostí. Detekované události jsou distribuovány v [9] IDEA formátu. IDEA formát je určený pro bezpečností události v podobě JSON dokumentu.



Obrázek 2.3: Schéma rozšíření systému NERD

Obrázek 2.3 ukazuje navržené rozšíření na reprezentaci systému NERD. Vyznačen je samostatný modul hledání doménových jmen v blacklistech a část webového rozhraní, které bude rozšířeno o informace z PassiveDNS.

2.4.1 Web

Informace o entitách jsou přístupné přes webového rozhraní. Jedna ze stránek zobrazuje detailní pohled na danou entitu, například seznam blacklistů. Rozšíření této stránky proběhně přes skript, který bude umístěn na stránce a při načtení stránky získá informace z PassiveDNS, které zobrazí na stránce.

2.4.2 Přidání modulu

NERD si udržuje v databázi záznam o síťových entitách. Záznam obsahuje metadata jako doménové jméno, geolokaci, ASN číslo atd. O změny v databázi se stará modul UpdateManager. UpdateManager dostává žádosti o změnu atributů a sám tyto změny zapisuje do databáze.

Žádost o úpravu atributů je složena ze tří částí: název samotné úpravy, podle jakého klíče vybrat atribut a nakonec seznam konkrétních změn atributu. Jednotlivé moduly mají příslušnou metodu registrovanou na určité události. Při vyvolání události upraví parametry podle svého zadání a po dokončení úkonu, vloží požadavek na změnu do globální fronty požadavků.

Rozšíření bude provedeno přidáním modulu, který bude komunikovat PassiveDNS. Modul se pro danou entitu dotáže na historii doménových jmen, na které se entita namapovala. Úkolem modulu bude pro každé doménové jméno zkontrolovat jeho přítomnost v doménových blacklistech. V případě pozitivního nalezení v blacklistu, modul vytvoří požadavek pro přidání této informace do záznamu entity, který následně vloží do fronty.

Kapitola 3

Realizace

Tato kapitola popisuje implementaci dílčích částí, které tvoří prototyp PassiveDNS. Samotná implementace je napsána v jazyce Python. Jednotlivé části představují moduly s konkrétní činností. Moduly se starají o obsluhu dotazů na REST API, import dat a databázovou vrstvu.

3.1 REST API

REST API je mechanismus pro snadnou distribuci dat mezi různými aplikacemi. Data jsou indentifikována URL řetězcem a je možné pro nět určit několik možných reprezentací. Pro prezetanci dat z Passive DNS pro klientskou stranu je použit formát JSON. REST API nejčastěji používá HTTP protokol pro samotný přenos.

Jednoduchým nástrojem pro tvorbu REST API je Python knihovna Flask. Flask umožňuje svázat konkrétní URL s obslužnou metodou. Při HTTP dotazu na dané URL, metoda vykoná logiku pro obsloužení požadavku. Sada takových metod tvoří dané REST API.

Inicializaci a nastavení REST API má na starost modul `app.py`. Při spuštění modul vytvoří instanci Flask aplikace, která slouží k následnému vytvoření instance třídy Api. Instance třídy Api přes metodu `add_resource` dokáže vytvořit část REST API. Metoda přijímá jako parametry URL řetězec a název definované třídy. Třída dědí ze třídy `Resource` a implementuje virtuální metody, které se zavolají pro vykonání obsluhy dotazu na URL. V rámci třídy je možné specifikovat jaké typy požadavků metoda zpracuje nebo jaké parametry přijme.

Obsluha REST API je definovaná v modulu `resources/dns_record.py`. Modul obsahuje třídy `domain` a `ip`. Pokud není řečeno jinak, dotaz na URL je typu GET. Definová metoda dané třídy příjme HTTP požadavek a odpoví JSON dokumentem, který obsahuje seznam záznamů. V případě chyby systém odpoví dokumentem ve formátu JSON, který obsahuje kód a popis chyby, která nastala.

3.1.1 Domain

Třída domain obsluhuje HTTP dotazy na URL `/pdns/domain/<domainname>`. Tento URL dotaz vrátí seznam IP adres, které se v minulosti přeložily na doménu danou parametrem `<domainname>`. Parametr `<domainname>` je plně kvalifikované doménové jméno s tečknou na konci.

Do URL je možné doplnit volitelné parametry `<to>` a `<from>`. Oba parametry přebírají konkrétní datum. Odpověď je pak omezena na položky, jejichž atributy `time_first` a `time_last` spadají do intervalu.
Příklad dotazu:

```
curl https://passivedns.cesnet.cz/pdns/domain/fit.cvut.cz.
```


```
[{
        "count": 412,
        "response": "195.113.161.8",
        "timefirst": "2018-02-27",
        "timelast": "2018-04-14",
        "type": "A"
 },
 {
        "count": 407,
        "response": "2001:718:1:a100::161:8",
        "timefirst": "2018-02-27",
        "timelast": "2018-04-14",
        "type": "AAAA"
 }]
```


Obsluha dotazu je implementavaná v metodě `get()`. Pro komunikace s databází je použita metoda `find_domain()`, která vrací pole nalezených položek. Každá položka je uložena do JSON dokumentu a ten je následně vrácen klientovy.

3.1.2 Ip

Třída ip obsluhuje HTTP dotazy na URL `/pdns/ip/<ipadress>`, kde ipadress je parametr na IP adresu. Tento URL dotaz vrátí seznam domén, které se v minulosti namapovaly k IP adrese dané parametrem `<ipadddres>`. Parametr je IPv4 adresa v tečkové notaci nebo IPv6 adresa v plném formátu.

Do URL e možné doplnit volitelné parametry `<to>` a `<from>`. Oba parametry přebírají konkrétní datum. Odpověď je pak omezena na položky, jejichž atributy `time_first` a `time_last` spadají do intervalu.
Příklad dotazu:

```
curl https://passivedns.cesnet.cz/pdns/ip/2001:718:2:2901::248
```


```
[{
        "count": 20,
```

```
        "domain": "fit.cvut.cz.",
        "time_first": "2018-03-04",
        "time_last": "2018-04-21",
        "type": "A"
},
{
        "count": 2,
        "domain": "www.fit.cvut.cz.",
        "time_first": "2018-04-15",
        "time_last": "2018-04-15",
        "type": "A"
}]
```

Obsluha dotazu je implementavaná v metodě `get()`. Pro kombinaci s databází je použita metoda `find_ip()`, která vrací pole nalezených položek. Každá položka je uložena do JSON dokumentu a ten je následně vrácen klientovy.

3.2 Import dat

Modul `pdns_importer.py` importuje data z DNS nameserverů do PostgreSQL databáze. Nameservery zajišťují přenos dat. Pro sběr proudících paketů je využit program `dnscap`, který je určen pro extrahování paketů na úrovni DNS protokolu ze síťového provozu. Zachycené pakety jsou uloženy do souboru v `pcap` formátu. `pcap` soubory jsou následně pomocí scp přeneseny na PassiveDNS server do vyhrazeného adresáře.

Vyhrazené podadresáře se nacházejí v adresáři `/var/local/dnscap`. Na sledování příchozích dat je použita knihovna `pyinotify`, která umožní obsluhovat události spojené se souborovým systémem. Obsluha je provedena v registrované metodě, která je asynchronně volána při vzniku události. Je možné tak obsloužit konkrétní soubory při jejich vytvoření, modifikaci nebo smazání.

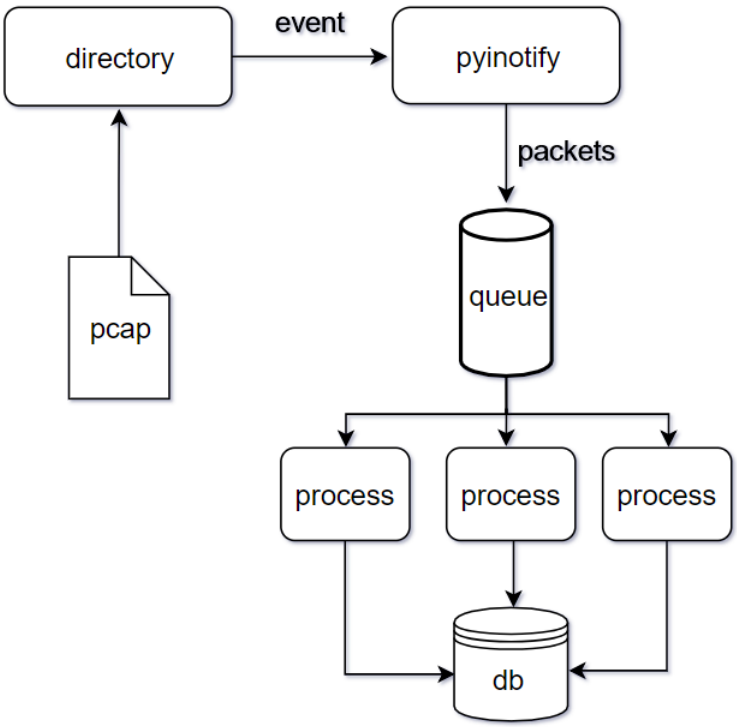
Soubor je komprimovaný podle gzip formátu a obsahuje binární proud dat v `pcap` formátu. Ten je potřeba dekomprimovat a jeho obsah zparsovat na jednotlivé DNS pakety. Paket nese určitý počet zdrojových záznamů. Ze zdrojových záznamů jsou extrahovány atributy rrname a rdata.

Pro rychlý a efektivní import dat ze souboru je použito pararelní řešení, které spočívá v rozdělení práce na více procesů. Podporu práce s procesy nabízí knihovna `multiprocessing`.

3.2.1 pcap

lib`pcap` je knihovna pro zachytávání dat na síti. Formát je složen z globální hlavičky. Ta obsahuje metadata jako magické čislo, verzi formátu, maximální

velikost zachycených paketů atd. Za globální hlavičkou následuje sekvence zachycených paketů. Každý paket začíná hlavičkou, která obsahuje časovou značku zachycení paketů a délku dat. Za hlavičkou následují samotná data.



Obrázek 3.1: Schéma modulu pdns_importer.py

3.2.2 Start

Před zahájením samotného importu je potřeba vytvořit procesy a nastavit sledování adresářů. Vytvoření samostatného procesu je možné pomocí třídy `Process`. Instance třídy přebírá název metody, odkud bude pokračovat vykonávání programu a volitelné parametry pro metodu.

Po převzetí konfiguračních parametrů, hlavní proces vytvoří zadaný počet procesů. Jméno předané metody je `parse_packet()`, která příjmá jako parametr datovou strukturu pro sdílení prostředky mezi procesy. Hlavní proces nechá vytvořit instanci třídy `JoinableQueue`, která zajistí synchronizaci sdílených položek mezi procesy. Tato instance je předaná jako parametr metodě `parse_packet()`. Přes tuto frontu je hlavní proces schopen předávat dílčí problémy synovským procesům.

Import data využívá procesy místo vláken, protože práce s vlákny v rámci jednoho interpretu přináší určité problémy s efektivitou. Pro zajištění synchronizace objektů mezi vlákny používá interpret GIL (Global Interper Lock). GIL

zajistí, že každý objekt v programu je dostupný v daném okamžiku pouze pro jedno vlákno.

Přidání více vláken tak prakticky stále znamená sekvenční zpracování problému. Vlákna je možné nahradit využitím procesů. Každý proces je samostatný interpret, který disponuje soukromým adresním prostorem a vlastním GIL. Tedy každý proces můžu přistoupit k určitému objektu bez vlivu ostatních. Nevýhodou procesů je vyšší nárok na paměť, ale získaná rychlost tuto nevýhodu vynahradí.

Následně je vytvořena instance třídy `WatchManager`. Instance převezme cesty k adresářům, které budou sledovány na událost `IN_CLOSE_WRITE`. Pro obsluhu této události je registrovaná metoda `dns_handler()`. Takto se docílí, že metoda bude asynchronně volána při dokončení přenosu souboru z nameserveru. Metoda přes převzatý parametr získá cestu k souboru, který událost vyvolal.

Po dokončení všech příprav pro import, hlavní proces čeká ve smyčce dokud není nastaven příznak pro ukončení. Vyvolané události se udržují ve vnitřní frontě. Při jejich odebrání se zavolá metoda pro zpracování.

Pro manipulaci s `pcap` formátem je použito rozhraní knihovny dpkt, které je určeno pro tvorbu a modifikaci paketů na různých vrstvách. Metoda `Reader` vrací pole binárních řetězců spolu s časovou značkou. Časová značka je extrahována z hlavičky paketu a značí čas kdy paket byl zachycen. Tento okamžik z hlediska PassiveDNS znamená čas kdy dvojice dotaz-odpověď v paketu existovala.

Každý řetězec, spolu s časovou značkou, ze seznamu je následně předán do `JoinableQueue` fronty, odkud si vloženou položku převezme čekají proces. Po dokončení předání metoda neprodleně končí a je tak možné ihned obsloužit další soubor v pořadí. Tímto je dosaženo, že při zpracování paketů nevzniká úzké hrdlo v místě manipulace se souborem. Zároveň se práce rovnoměrně rozloží na jednotlivá jádra.

Celkový průběh zpracování jednoho souboru je zobrazen na obrázku 3.1.

3.2.3 Parsování DNS paketů

Zpracování paketů na potřebná data probíhá v metodě `parse_packet()`. Synovské procesy čekají ve smyčce na data z fronty. Jedna iterace znamená import data z jednoho paketu. Při odebrání položky z fronty, je paket rozdělen na časovou značku a binární řětězec. Každý proces si zároveň udržuje vlastní objekt pro připojení k databázi, aby se předešlo konfliktům, když se pokusí naráz připojit k databázi více procesů.

Pro manipulaci s pakety je opět použita knihovna `dpkt.dpkt` knihovna nabízí třídy představující síťové vrstvy TCP/IP modelu. Instace třídy při inicializaci převezme binární rětězec dat a přemění binární řetězec na objektovou reprezentaci. Jednotlivé položky paketu jsou tak přístupné přes metody dané třídy. Je tak možné postupně projít vrsty až k samotným DNS datům. Zároveň

lze velmi pohodlně zkontrolovat délku paketu, typ síťové vrstvy nebo zda paket není poškozený. Až metoda vytvoří instanci DNS paketu, pracuje metoda opět s binárním řětězec. Ten ovšem nyní obsahuje zdrojové záznamy.

Pro manipulaci s tímto binárním řetězcem, je použita knihovna `dnslib`, která je určena jako jednoduchý nástroj na kódování a dekódování DNS dat. Rozhraní `dnslib` knihovny umožní z binárního řetězce získat objekt DNS zprávy, která obsahuje pole zdrojových záznamů. Přes toto pole je možné iterovat a obdržet atributy `rname` a `rdata`. Tyto atributy spolu, časová značka a objekt k připojení jsou předány metodě pro uložení záznamu do databáze. Metoda je zvolena podle typu záznamu. Proces počká na dokončení uložení a pokračuje další iterací.

Pokud je obsah DNS zprávy neúplný, poškozený nebo se nejedná o zprávu typu odpovědi, proces přeskočí iteraci. Proces odebírá položky z fronty dokud neobdrží hodnotu None.

3.2.4 Konfigurace

Při startu importu je možné nastavit určité parametry přes konfigurační soubor v `yml` formátu. Cesta ke konfiguračnímu souboru je vstupním parametr při inicializaci. Mezi nastavitelné parametry patří údaje pro spojení s databází, počet spouštěných procesů a cesty k adresářům pro import dat.

Pro jednoduché ovládání importu je vytvořena služba pro `systemd`. Služba je definována v souboru `/lib/systemd/system/passive-dns.service`. Modul je nastaven, aby se ukončil při zachycení signálů SIGTERM, SIGINT, SIGABRT. Při ukončení se zastaví sledování vyhrazeného adresáře a hlavní proces počká na dokončení ostatních procesů.

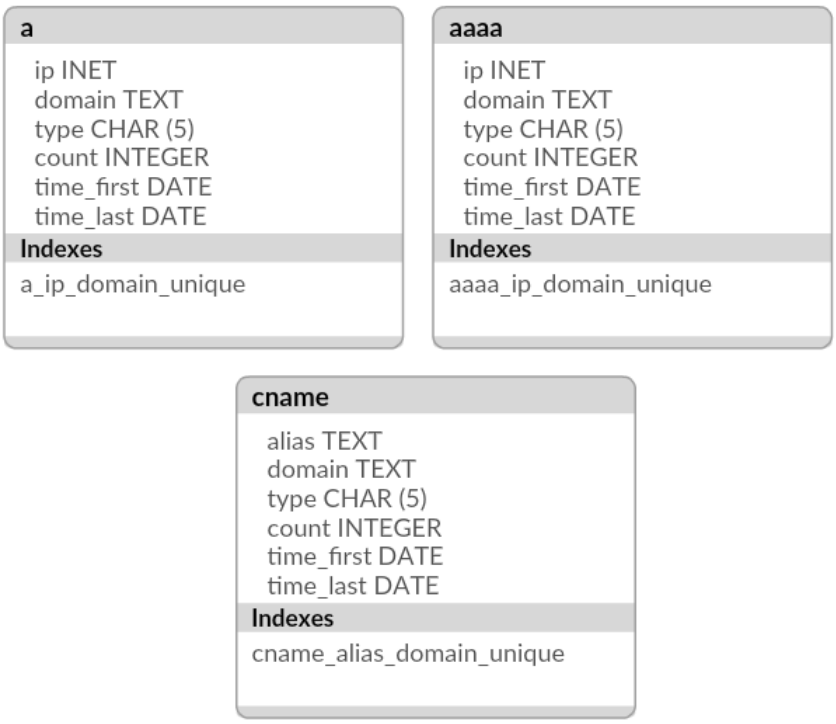
3.3 Databázové schéma

V tento moment existují tabulky pro záznamy typu A, AAAA, CNAME, protože se jedná o nejrozšířenější typy. Rozložení tabulek je zobrazeno na Obrázku 3.2. Například tabulka pro záznam IPv4 obsahuje dvojici IP adresa a namapovaná doména, počet zachycených dotazů na tuto konkrétní dvojici a čas prvního a posledního výskytu.

Sloupce v tabulce aaaa se liší tím, že ve sloupci ip obsahuje IPv6 záznam, CNAME se liší tím, že může obsahovat dvojici doména a její kanonické jméno. Položka ip z tabulek a a aaaa je datového typu `inet` určený pro IPv4, IPv6 a MAC adresy a sítě. Datový typ provádí kontrolu zda má adresa korektní formát.

Databáze je PostgreSQL verze 9.6. PostgreSQL nabízí datový typ pro síťové adresy a z hlediska výkonu se PostgreSQL ukázalo jako postačující pro potřeby PassiveDNS.

Modul `models/dns_record.py` má na starosti databázovou vrstvu aplikace. Pro interakci s databází je použita knihovna `psycopg2`. Modul obsa-



Obrázek 3.2: Struktura databázových tabulek

huje metody `find_ip()` a `find_domain()` pro vyhledání konkrétního záznamu podle jeho typu. Dále obsahuje metody, které přidají nový záznam nebo aktualizují ten starý pokud ho znovu vyhledají. Při vložení nové dvojice dotaz-odpověď modul nechá zkontrolovat databázi zda již neobsahuje existující dvojici. Pokud ano aktualizuje čas posledního výskytu na aktuální čas a přičte k čítači 1. Jinak nechá založit nový záznam s čítačem na 1 a nastaví čas prvního a posledního výskytu na aktuální čas.

Uložení záznamu do databáze umožňují funkce `save_a()`, `save_aaaa()` a `save_cname()` pro dané typy zvlášť. Metody `save_a()` a`save_aaaa()` akceptují ip adresu jako řetězec, doménové jméno a časovou značku. Metoda `save_cname()` akceptuje alisas, doménové jménno a časovou značku. Všechny tři metody také mohou akceptovat objekt pro připojení k databázi.

K navýšení výkonu databázových operací nad tabulkami jsou použity databázové indexy. Bez indexu systém provede sekvenční prohledání celé tabulky dokud nenajde požadované řádky. To může vést ke značné neefektivitě. Mnohem efektivnější je náhodný přístup, který je umožněn udržováním indexu nad tabulkou. Obecně je vhodné použít indexy na řádky, které obsahují data s nízkou mírou opakování. To znamená čím větší podíl rozdílných sloupců vůči celkovému počtu, tím je větší pravděpodobnost, že databázový engine použije

raději index při vykonání dotazu.

Index je datová struktura, která si udržuje informace kde najít často vyhledávané řádky. Index je pravidelně udržován při SQL operacích vložení nebo smazání záznamu. Nejběžněji používaná struktura je B-strom. B-strom zvládá obsluhovat SQL dotazy s operacemi rovnosti nebo porovnání, takže při výskytů operátorů porovnání databázový plánovač zváží použití B-strom pro získání řádků.

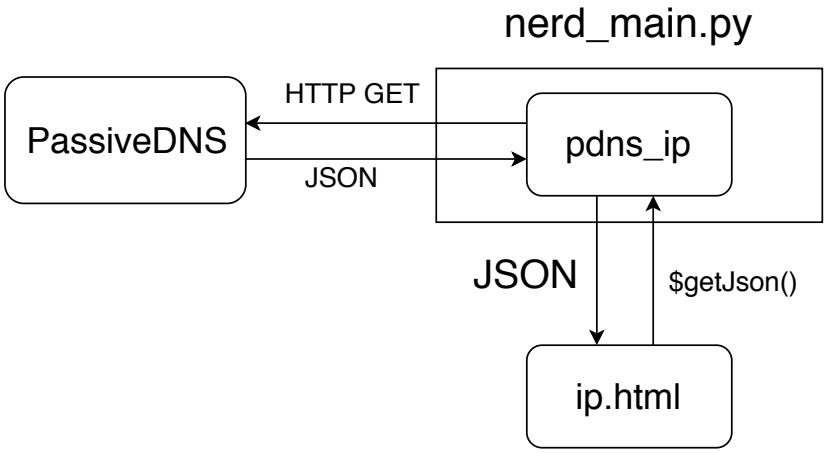
Ve všech tabulkách je vytvořen unikátní index na dvojice dotaz-odpověď. Tím se kromě rychlosti, získá i konzistence řádků v databázi, protože unikátní index neumožní uložení duplicitních hodnot.

pdns_importer.py implementuje uložení dotazu pomocí klauzule **ON CONFLICT ... DO**.

```
'INSERT INTO {tbl} AS ORIG
({ip_alias}, domain, type, count, time_first, time_last)'\
'VALUES (%s, %s, %s, %s, %s, %s) ON CONFLICT'\
'({ip_alias}, domain) DO UPDATE '\
'SET count = ORIG.count + 1, time_last = EXCLUDED.time_last
```

Takto sestavený dotaz uvolní určitou zátěž ze strany databáze při vykonání plánu, protože při vykonání `ON CONFLICT ... DO` se databáze prohledá jenom jednou, na rozdíl při použití výše zmíněných třech dotazů.

3.4 NERD



Obrázek 3.3: Rozšíření ip.html o informace z PassiveDNS

3.4.1 REST API

V systému NERD se samotného rozšíření dosáhlo pomocí javascriptu, konkrétně s knihovnou JQUERY, který souběžné při načtení stránky provede dotaz na `/pdns/IP` URL. Dotaz provede funkcí `$get()`, která výsledek zpracuje v callback funkci. Na `/pdns/IP` URL je v modulu `nerd_main.py` registrovaná metada `pdns_ip` pro obsluhu požadavku. Tento postup je zobrazen na obrázku 3.3.

Detailní sourh informací ohledně IP adresy je možné získat dotazem na URL `/nerd/ip/<ipadress>`. Odpovědí na dotaz je html dokument `html.ip` s informacemi ohledně IP adresy. Javascript program je proto umístěn na stránce `html.ip`. JQUERY převezme hodnotu IP adresy v `/nerd/ip/<ipadress>` URL a s pomocí knihovny requests provede dotaz na PassiveDNS API a čeká na JSON dokument, který vrátí jako odpověď volajícímu javascriptu. Výsledný JSON dokument je po úspěšném načtení k dispozici v callback funkci. Pokud je dokumentem neprázdný, nechá funkce pro každou položku v JSON dokumentu zobrazit jednotlivé atributy v html stránce.

3.4.2 PassiveDNS modul

Účelem modulu `passive_dns.py` je využití informací z PassiveDNS pro doplnění záznamu o entitě. Pro danou entitu modul zkoumá přítomnost domén v blacklistech, na které se namapovala IP adresa v záznamu entity. Konkrétně tak modul ovlivní atribut `dbl`. Jedná se o pole doménových blacklistů. Atributy v rámci jedné položky `dbl` jsou uvedeny v Tabulce 3.1. Při startu modul registruje metodu na události vytvoření nové entity a na událost denní aktualizace.

Tabulka 3.1: Atributy dbl

název atributu	význam
dbl.n	Název blacklistu
dbl.d	Název domény
dbl.v	Příznak přítomnosti domény v blacklistu
dbl.t	Čas posledního dotazu na blacklist
dbl.h	Seznam časových značek kdy dotaz na blacklist vrátil pozitivní výsledek, tedy doména byla v blacklistu nalezena

NERD udržuje doménové blacklisty v lokální Redis databázi. Při inicializaci `passive_dns.py` získá instanci databázového modulu redis a přes tuto instanci vytvoří v paměti seznam objektů představující doménové blacklisty. Každý z těchto objektů si udržuje vlastní seznam registrovaných domén a skrze metodu umožní ověření přítomnosti předané domény. Při obsluze registrované události, modul převezme IP adresu jako klíč záznamu a provede

dotaz na PassiveDNS REST API. Dotaz vrátí JSON dokument se seznamem doménových jmen.

Modul následně provede kontrolu každé domény z převzatého seznamu vůči všem dostupným blacklistům. Pokud metoda pro kontrolu přítomnosti vrátí úspěch, nechá modul vytvořit instanci dbl atributu, kde dvojice doménové jméno a název blacklistu tvoří identifikátor této instanci v seznamu. Tato instance je spojena s příkazem `array_upsert`, která předá dbl položku do záznamu entity nebo tuto položku aktualizuje pokud již v záznamu dané entity existuje.

Kapitola 4

Testování

Tato kapitola se věnuje testování jednolivých komponent PassiveDNS. Úvod se věnuje porovnání různých variant implementace importu dat. Následuje popis jak se prototyp postupně nasadil. V závěru je zmíněno testování funkčnost ukládání importováných dat a testování rozšíření systému NERD.

4.1 Import dat

Tato sekce popíše ladění a testování `pdns_importer.py` modulu. Milníkem pro dosažení uspokojivého výkonu bylo stanovena snížit dobu zpracování příchozích dat pod určitý časový úsek, aby soubory nepřicházely rychleji než se stihnou zpracovat . Samozřejmě určitým řešením může být pozastavení posílání dat z konkrétního zdroje, ale z dlouhodobého hlediska by tato možnost nemohla fungovat.

Konkrétní zdroje dat a jejich objem dat jsou vypsány Tabulce 4.1

DNS nameserver	průměrná velikost komprimovaného souboru
adns1.cesnet.cz	3.5 - 6 MB
adns2.cesnet.cz	1 - 1.5 MB
ns.ces.net	0.5 MB

Tabulka 4.1: velikost příchozích dat

Pro sledování výkonu byl využit nástroj `munin`. Prvotní varianta implementace importu dat spočívala v kombinaci sekvenčního zpracování paketů a následné ukládání pomocí tří SQL dotazů. Veškerý běh programu tedy běžel na jednom jádře. Tato implementace vykazovala výrazně nedostatečné výsledky. Dle výpisů z logů, v nejhorších případech doba zpracování zaslaných souborů s velikosti 5MB průměrně dosahovala přes 130 vteřin.

Následující varianta změnila způsob uložení dat na jeden dotaz s pomocí klazule `ON CONFLICT ... DO`. Došlo ke snížení zátěže na databázi, protože

prováděla už jenom jedno prohledání databáze oproti třem.

Další varianta spočívala v paralelním zpracování vstupních dat. Dle výpisů z monitorovacích nástrojů typu htop, se práce nad pakety rovnoměrně rozložila mezi CPU, ale došlo jenom k mírnému zlepšení.

Po kombinaci paralelního zpracování a optimalizace SQL dotazování, se podařilo dosáhnout okraje milníku, kdy průměrná doba zpracování souborů s velikostí 4-4,5 MB dosahovala kolem 60 vteřin. Úzkým hrdlem nakonec ukázala samotná knihovna scapy, která velmi dlouho parsovala data na pakety než došlo k jejich předání vláknům. Tím pádem došlo k výměně knihovny za knihovny `dpkt` a `dnslib`, které dokázaly dobu parsování výrazně snížit.

Uspokojivé doby importu bylo nakonec dosaženo využitím knihoven `dkpt` a `dnslib` za využití pararelního zpracování na úrovni paketů. Podle nejnovější výpisů z logů největší vrcholy zpracování dosahují okolo 25 vteřin. Soubory se tak na serveru neshromažďují a je možné rozumně rychle data importovat do databáze.

Přechod z vláken na procesy zkrátilo dobu zpracování příchozích dat na 2 vteřiny. Modul `pdns_importer.py` stihne zpracovat zhruba 50 000 záznamů za vteřinu.

4.2 Postupné nasazení

Vývoj začal návrhem a implementací REST API pro dotazování na data. Nato došlo k nasazení PostgreSQL databáze a implementace databázové vrstvy. Následovalo spuštění první varianty modulu pro import dat. Po vyzkoušení na lokálních datech, došlo na testování s opravdovými daty z DNS nameserverů. Při určitém zaplnění databáze došlo ke zpomalení při ukládání dat, což vedlo k zavedení indexů nad tabulkami.

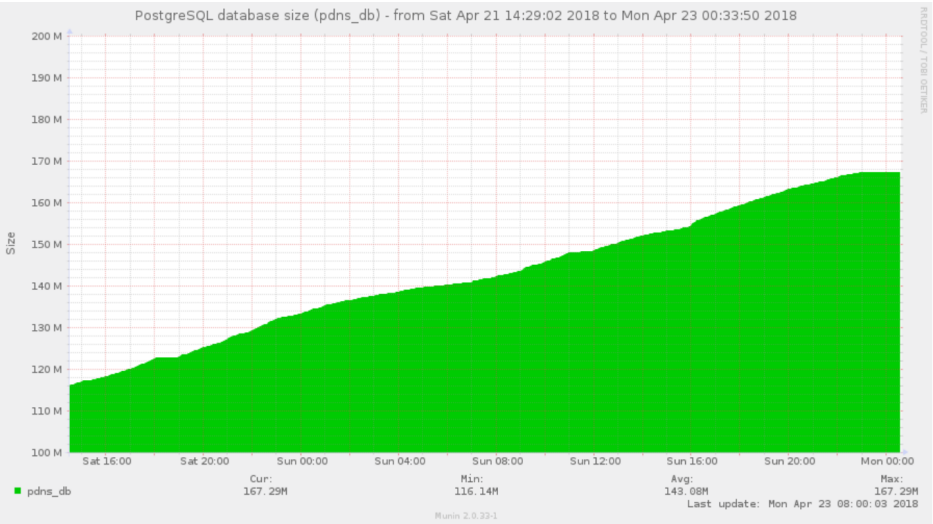
Uspokojivý výsledek vedl k zapojení dalších DNS nameserverů pro zvýšení objemu dat. Tento objem vedl k výraznému zpomalení importu. Postupně se vykoušelo několik variant implementace imporu, než se rychlost importu zastavila na uspokojivé hodnotě.

Souběžně došlo k nasazení certifikátu do konfigurace Apache2 serveru, aby bylo možné komunikovat přes HTTPS protokol. HTTP dotazy se následně přesměrovaly prostřednicvím WSGI protokolu na REST API.

Nakonec přišla část rozšíření systému NERD. Napřed se rozšířila html stránka pro zobrazení detailů ohledně IP adresy, aby se otestovalo spojení. Poslední část zahrnovala vytvoření modulu pro vyhledání doménových jmen v lokálních blacklistech.

4.3 Výkon databáze

PassiveDNS je nasazen na stroji s parametry: CPU Intel(R) Xeon(R) E5-2680 v4 2.40 GHz. Stroj má k dispozici 56 jader a 1000 GB RAM. Jeho největší



Obrázek 4.1: Graf růstu velikost PostgreSQL databáze

oddíl na disku obsahuje kolem 1,5 TiB paměti. Podařilo dosáhnout zpracování zhruba 50 000 záznamů za vteřinu. Současný trend růstu velikosti databáze je zhruba 100 MB za den.

4.4 Unit testy

Implementace testů je obsažena ve skriptu `pdns_test.py` s pomocí knihovny unittest. Jeho účelem je zkontrolovat integrity databázového schématu a ověřit funkčnost rozhraní třídy `dns_record.py` pro ukládání dat. Během testů je vytvořeno dočasné prostředí ze skriptu `create_schema.sql`, které odpovídá tomu produkčnímu.

Test pro integritu schématu nechá ověřit ze systémových tabulek, zda získané tabulky, sloupce a indexy odpovídají předpokládané struktuře. Další test ověřuje metody zodpovědné za uložení dat do databáze. Konkrétně jde o metody `save_a()`, `save_aaaa()` a `save_cname()` třídy `models/dns_record.py` zda korektně uloží vybrané vstupy. Následují testy, zda metody na špatné vstupy odpoví očekávanou chybou a nedojde tak k uložení nesprávných dat.

Při testech se kontroluje zda se čas prvního a posledního zachycení daného záznamu správně rozšiřuje. Následně se kontroluje zda se uložil očekávaný počet výskytů v položce count. Potom se ověří pro danou IP adresu, zda si při postupném ukládání udržuje očekávané domény a následně obdobně pro danou doménu.

4.5 Testování systému NERD

Pro účely testování byla použita kopie systému NERD na virtuálním stroji vagrantu. Kopie běžela na distribuci CentOS. Jako testovací data byl použit vzorek 10 000 zpráv IDEA. Zpracování jednolivých vzorků ukázalo chování modulu passive_dns.py. Výsledkem testu se ukázalo, že velmi malá část dat se nachází jak v systému NERD tak v PassiveDNS. Důvodem je, že většina entit v systému NERD představují nakažené stroje, zatímco v PassiveDNS jsou všeobecné dotazy klientů. A další testy ukázaly, že z nalezeného průniku opět velmi část se nenachází v doménových blacklistech. Funkčnost jako taková se ukázala bezproblémovou. Nicméně v blízké budoucnosti se očekává zapojení více datových zdrojů, což by mělo vést k mnohem lepším výsledkům.

4.6 Statistiky z testování

Nejefektivnější varianta importu dat byla spuštěna 28 dubna. Průměrná doba zpracování příchozích dat se od toho dne stále drží na době 2 vteřin. Jelikož se data posílají pravidelně každou minutu, jedná se o velkou časovou rezervu. V Tabulce 4.2 je uveden několik údajů pro záznamy tabulkách.

Tabulka 4.2: Statistiky z testovacího nasazení

Název tabulky	**a**	**aaaa**	**cname**
# řádků	9 770 000	529 000	1 946 000
Nejvyšší # zachycení	710 000	180 000	130 000
Maximální délka domény	254	86	100
Průměrná délka domény	31	27	26
Průměrný # mapování IP adresy na doménu	4,3	1,57	x
Maximální # mapování IP adresy na doménu	368 000	9 599	x
Průměrný nárůst řádků za hodinu	27 638	1 039	2 630
Velikost tabulky	1 1300 MB	77 MB	220 MB

V Tabulce 4.3 je uvedeno 10 doménových jmen, které jsou nejčastěji překládany.

Tabulka 4.3: 10 nejčastěji dotazovaných domén

doménové jméno	počet zachycení
a-0011.a-msedge.net.	713160
map.fastly.net.	709714
a-0001.a-msedge.net.	619533
a-0001.a-msedge.net.	619531
c-0001.c-msedge.net.	597234
global.fastly.net.	506268
ubl.unsubscore.com.	469551
b-0001.b-msedge.net.	392735
gwrtdp.tclclouds.com.	278420
dns.msftncsi.com.	259963

Závěr

Tématem této bakalářské práce bylo využití informací z reálného provozu DNS protokolu pro evidování historie mapování IP adres a doménových jmen. Výstupem této práce je systém PassiveDNS, který vznikl na základě důkladné analýzy protokolu DNS a požadavků na použití systému PassiveDNS. Data, které se v systému evidují je možné využít k mnoha účelům jako je například analýza bezpečnostních událostí a hledání potenciálně škodlivých adres.

Tato práce obsahuje teoretický úvod, který se zabývá fungováním služby DNS a formátem zpráv, které se přenáší mezi klienty a servery při hledání IP adres k doménovým jménům a naopak. Dále práce popisuje seznam požadavků, které byly identifikovány na základě konzultací s odborníky z praxe ze sdružení CESNET, které je operátorem České národní akademické sítě. Součástí požadavků byla nutnost vytvořit novému systému komunikační rozhraní, které umožňuje ostatním existujícím systémům využívat ukládaná data.

Na základě návrhu byl vytvořen systém PassiveDNS s využitím existující databáze PostgreSQL a jazyka Python, ve kterém byly vytvořeny důležité části systému, jako je načítání a zpracování surových DNS dat a ukládání informací do databáze. Nad databází dále funguje server Flask, který poskytuje uživatelům a systémům RESTAPI, přes které je možné vyhledávat uložená data.

Vytvořený systém byl nasazený pro testování přímo ve sdružení CESNET a začal zpracovávat DNS provoz z legitimních rekurzivních DNS resolverů, které sdružení provozuje. Toto testovací nasazení umožnilo ověřit funkcionalitu vytvořeného systému PassiveDNS a změřit jeho výkonnost. Výsledky testování jsou součástí této práce.

Na závěr se práce zaměřuje na rozšíření systému Network Entity Reputation Database (NERD), který ve sdružení CESNET zpracovává nahlášené bezpečnostní události. NERD udržuje seznam škodlivých adres, které jsou v bezpečnostních událostech označeny jako zdroj problému, a dohledává o nich informace ve volně dostupných zdrojích (OSINT). PassiveDNS je proto vhodným dalším zdrojem informací pro NERD, protože umožňuje dohledat

historii doménových jmen a IP adres.

Rozšíření systému NERD bylo zaměřeno na dva cíle: 1) přidat výpis historie IP adresy ve webovém uživatelském rozhraní systému NERD na požádání uživatelem, 2) vytvořit automatizovaný modul systému NERD, který umí ke každé dové podezřelé IP adrese dohledat historii v PassiveDNS a získané informace zkontrolovat na veřejně dostupných blacklistech. Oba tyto cíle se podařilo splnit a nová rozšířená verze systému NERD je již nasazená v produkci.

Budoucí práce

Na základě zkušeností z testovacího nasazení vytvořeného funkčního systému PassiveDNS jsme identifikovali několik možností, jak systém do budoucna vylepšit. Je možné vyzkoušet alternativu ukládání všech zachycených zpráv v neagregované podobě nebo rozšířit používaný agregovaný formát o další atributy, tedy o další informace obsažené v DNS zprávách. Například hodnutu TTL. Další možností rozšíření je napojení dalších zdrojů dat, aby se import rozšířil o data z dalších sítí.

Literatura

[1] Reputation Score. https://link.springer.com/chapter/10.1007%2F978-3-319-39814-3_13.

[2] Mockapetris, P.: RFC 1034: Domain names: concepts and facilities (November 1987). *Status: Standard*, ročník 6, 2003.

[3] Mockapetris, P.: RFC 1035—Domain names—implementation and specification, November 1987. *URL http://www. ietf. org/rfc/rfc1035. txt*, 2004.

[4] Weimer, F.: Passive DNS replication. In *FIRST conference on computer security incident*, 2005, str. 98.

[5] Marchal, S.; François, J.; Wagner, C.; aj.: DNSSM: A large scale passive DNS security monitoring framework. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, IEEE, 2012, s. 988–993.

[6] Farsight Security. https://www.farsightsecurity.com/.

[7] Caletka, O.: https://www.nic.cz/public_media/IT14/prezentace/Ondrej_Caletka.pdf.

[8] Warden. https://warden.cesnet.cz//cs/index.

[9] IDEA. https://idea.cesnet.cz/en/index.

Příloha A

Seznam použitých zkratek

DNS Domain Name System

SQL Structured Query Language

XML Extensible markup language

NERD Network Entity Reputation Database

WSGI Web Server Gateway Interface

REST Representational state transfer

JSON JavaScript Object Notation

Příloha B

Obsah přiloženého CD

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