VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

HLEDÁNÍ ANOMÁLIÍ V DNS PROVOZU

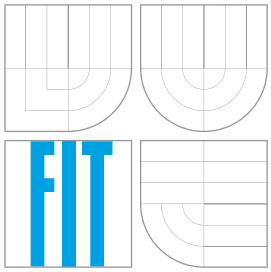
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL VRAŠTIAK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

HLEDÁNÍ ANOMÁLIÍ V DNS PROVOZU

ANOMALY DETECTION IN DNS TRAFFIC

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PAVEL VRAŠTIAK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2012

Abstrakt

Tato diplomová práce je napsána ve spolupráci s firmou NIC.CZ a zabývá se anomáliemi v provozu systému DNS. Obsahuje popis základních principů tohoto systému a vlastností, kterými se jeho provoz vyznačuje. Účelem této práce je pokusit se vytvořit klasifikátor některých z anomálií v této práci uvedených a ověřit jeho schopnosti teoreticky i v praktických podmínkách.

Abstract

This master's project is written in collaboration with NIC.CZ company. It describes basic principles of DNS system and properties of DNS traffic. It's goal is an implementation of a DNS anomaly classifier and its evaluation in practice.

Klíčová slova

Anomálie, DNS, klasifikace, náhodný les

Keywords

Anomaly, DNS, classification, random forest

Citace

Pavel Vraštiak: Hledání anomálií v DNS provozu, diplomová práce, Brno, FIT VUT v Brně, 2012

Hledání anomálií v DNS provozu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D. a mého konzultanta Ing. Karla Slaného. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavel Vraštiak
13. května 2012

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu práce a ochotu, kterou projevoval během tvorby této práce. Dále bych rád poděkoval mému konzultantovi, panu Ing. Karlu Slanému, za jeho podnětné rady a množství článků, kterým mě zahrnoval. Nakonec bych rád poděkoval své rodině za morální i finanční podporu, kterou mi po celou dobu studia poskytovala.

Obsah

Kapitola 1

Úvod

DNS systém je neodmyslitelnou součástí veškeré komunikace na Internetu a většina komunikace na Internetu začíná právě DNS dotazem. Stopy těchto komunikací lze zaznamenávat na jednotlivých úrovních hierarchie DNS systému a využít je k rozpoznání těch typů nebo zdrojů provozu, které vykazují odlišné chování. Prioritou je odlišit a detekovat to chování, které vede ke zvyšování zátěže, obtěžování nebo poškozování běžných uživatelů (špatné nastavení, spamboti, botnety, ditribuované útoky - ne nutně pouze na DNS infrastrukturu).

Tato práce je psána ve spolupráci s firmou CZ.NIC, správcem české domény, a snaží se navázat na již existující práci [12] zabývající se detekcí anomálií v provozu DNS. Účelem práce je vytvoření nástroje pro automatickou klasifikaci těchto anomálií a usnadnit tak jejich identifikaci.

V kapitole 2 jsou nejprve představeny základní principy a vlastnosti DNS systému, je zde popsáno, z jakých komponent se tento systém skládá a jak funguje. V následující kapitole je popsána charakteristika legitimního DNS provozu. V části 5 je uvedeno několik metod vhodných k použití pro klasifikaci a jsou diskutovány jejich vlastnosti, aby byla nakonec v kapitole 6 popsána vybraná klasifikační metoda, Náhodný les (Random Forest), která poskytuje jedny z nejlepších výsledků v porovnání s ostatními klasifikačními metodami. Cílem výsledné práce bude implementace zde navrženého systému a jeho testování na reálných datech.

Kapitola 2

DNS - Domain Name System

Téměř všechny aplikace, které na Internetu zajišťují komunikaci mezi uživateli, používají k identifikaci komunikujících uzlů IP adresu. Délka IPv4 adresy je 32 bitů a je tvořena čtyřmi oktety. Zapisuje se jako čyři čísla v rozsahu $0--255$ oddělená tečkami, např. `147.229.9.23`. V případě novějšího protokolu IPv6 je délka adresy dokonce 128 bitů a zapisuje se jako osm 16-bitových hexadecimálních čísel oddělených dvojtečkami. Ukázkou IPv6 adresy je například adresa `2001:1234:5678:90:ABCD:800:200C:417A`. Jelikož pro člověka nejsou tyto číselné formáty snadno zapamatovatelné, používají se jmenné ekvivalenty těchto síťových rozhraní (počítačů), které se vždy na cílovou IP adresu přeloží. A právě překlad IP adres a jejich srozumitelných ekvivalentů, tzv. doménových jmen, je hlavním úkolem a příčinou vzniku této celosvětově distribuované databáze.

Systém DNS se skládá ze tří hlavních komponent [19]:

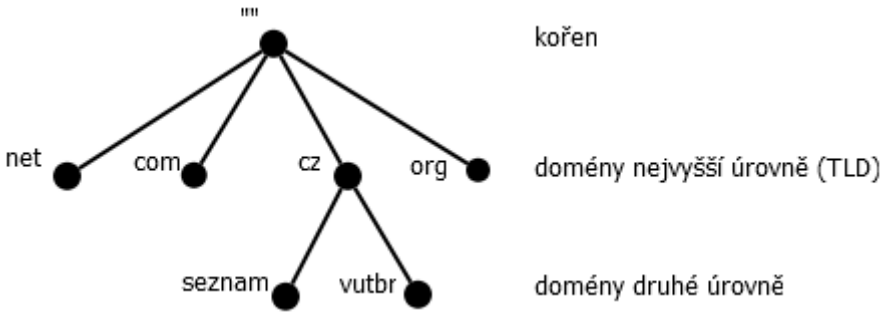
- √ **Prostoru doménových jmen a DNS záznamů**, které popisují strukturu jmenného prostoru a dat s tímto spjatými. Každý uzel tohoto stromu obsahuje jistou podmnožinu DNS záznamů. Na tyto jsou pak pokládány dotazy za účelem zjištění konkrétní informace. Dotaz obsahuje jméno dotazované domény a specifikaci požadovaného typu DNS záznamu.

- √ **Serverů DNS**, mezi které je distribuována informace o struktuře doménového stromu a přidružených informací. Mohou teoreticky mít informaci o libovolné části doménového stromu, ale obecně obhospodařují pouze část z prostoru jmen. Dále obsahují ukazatele na další jmenné servery, aby bylo možné dostat se k libovolné části jmenného prostoru. Pokud má server úplnou znalost o nějaké části doménového stromu, pak je označován jako autoritativní pro daný jmenný prostor.

- √ **Resolveru**, který zprostředkovává přeložení jména na IP adresu. Jedná se o klientský program, který získává informace od serverů DNS, ale může být nakonfigurován i staticky bez nutnosti DNS. Proces vyhledání odpovědi v DNS systému se nazývá rozlišení jména nebo také rezoluce [18]. Pro správnou funkci resolveru na Internetu musí být tento schopen přistupovat alespoň k jednomu serveru DNS. Od něj získá hledanou odpověď přímo, nebo mu server vrátí odkaz na další server, kde je hledaná informace uložena. Resolver tak může přeposílat dotazy dalším serverů.

2.1 Prostor doménových jmen

Systém DNS tvoří databáze uspořádaná jako kořenový strom doménových adres, viz obrázek 2.1. Název kořenového uzlu je tvořen řetězcem nulové délky, ostatní uzly stromu jsou



Obrázek 2.1: Ukázka části prostoru doménových jmen

pojmenovány textovým řetězcem o délce nejvýše 63 oktetů/znaků. Celé jméno domény nesmí překročit 255 znaků [20].

Listy ve stromu označuji konkrétní síťová zařízení patřící do dané domény. Správa domén je decentralizována a je delegována na další organizace.

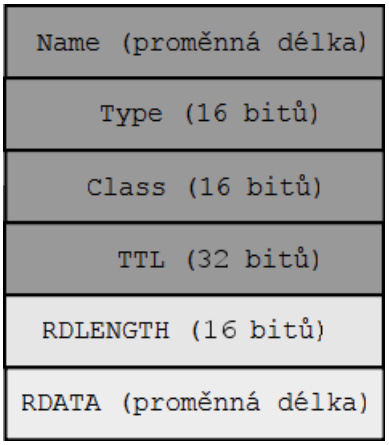
Fyzicky jsou části podstromu celého prostoru doménových adres uloženy na jmenných serverech, které tvoří DNS systém. Fyzické části prostoru DNS, které jsou pod jednou správou, se nazývají zóny. Zóna nemusí být totožná s doménou.

2.2 Záznamy DNS

Pro ukládání informací v datovém prostoru DNS slouží speciální záznamy, které se nazývají zdrojové (resource records, RR) a jsou uloženy v textové podobě v zónových souborech na serverech DNS. Všechny zdrojové záznamy mají stejnou strukturu [20], která je znázorněna na obrázku 2.2.

Význam jednotlivých položek je následující:

- NAME – Doménové jméno vlastníka záznamu.
- TYPE – Typ záznamu (viz dále).
- CLASS – Třída záznamu (IN – pro Internet, a CH – pro Chaos).
- TTL – Doba, po kterou může být záznam uložen v cache paměti.
- RDLENGTH – Délka následujícího pole RDATA.
- RDATA – Vlastní data ve tvaru řetězce, jehož formát je určen typem a třídou záznamu.



Obrázek 2.2: Obecná struktura DNS záznamu

Nejčastěji používané typy zdrojových záznamů jsou:

- **A** – Slouží k mapování doménových jmen na IPv4 adresy. Používají se k uložení IPv4 adresy vázané na dané doménové jméno.
- **AAAA** – Používají se k uložení vazby mezi doménovým jménem a jeho IPv6 adresou.
- **CNAME** – Kanonické jméno určuje, že dotazované doménové jméno je alias a má hodnotu jiného kanonického doménového jména.
- **MX** – Záznamy serveru pro výměnu elektronické pošty.
- **NS** – Záznamy serveru jmen uvádí autoritativní jmenný server pro pro podřazené domény. Používají se v bodu delegace v podřazené doméně dané zóny nebo v nadřazené zóně pro danou zónu.
- **PTR** – Záznamy PTR umožňují vyhledání hostitelského jména dle dané IP adresy.
- **SOA** – Start of autority. Uvozuje každou zónu.

2.3 Server DNS

Na nejvyšší úrovni stojí kořenové jmenné servery, které poskytují informace obsažené v kořenovém zónovém souboru ostatním DNS serverům. Představují zásadní část technické infrastruktury Internetu, na které závisí spolehlivost, správnost a bezpečnost operací na Internetu. Vrací přímo odpovědi na dotazy v kořenové zóně nebo odpovídají seznamem příslušných autoritativních serverů pro dané domény nejvyšší úrovně. Kořenové servery jsou rozmístěny po celém světě[1]a jsou tvořeny distribuovanými svazky počítačů. Díky maximální velikosti UDP datagramu byl jejich celkový počet zvolen 13. Fyzických serverů je samozřejmě mnohem více, neboť jejich vytížení je obrovské a je nutné tuto zátěž rozptýlit mezi mnoho serverů. V základní podobě rozlišujeme [18]:

[1]Viz http://www.root-servers.org/map/

- **primární** (*master, primary*) server – poskytuje autoritativní odpovědi pro domény, které spravuje. Záznamy přidržené k těmto doménám jsou nakofigurovány v lokálním souboru. Pro každou doménu musí existovat právě jeden primární jmenný server.
- **sekundární** (*slave*) server – je automatickou kopií primárního. Průběžně si aktualizuje data a slouží jednak jako záloha pro případ výpadku primárního serveru, jednak pro rozkládání zátěže u frekventovaných domén. Každá doména musí mít alespoň jeden sekundární server. Také poskytuje autoritativní odpovědi.
- **záložní** (*caching-only*) server – slouží jako vyrovnávací paměť pro snížení zátěže celého systému. Uchovává si odpovědi a poskytuje je při opakování dotazů, dokud nevyprší jejich životnost. Neposkytuje autoritativní odpovědi. Klient může požádat o autoritativní odpověď, v běžných případech ale stačí jakákoli.

2.4 Zodpovídání DNS dotazů

Jak bylo již řečeno, proces vyhledávání odpovědi v systému DNS bývá někdy označován jako rezoluce [18]. Díky hierarchickému uspořádání prostoru doménových adres v podobě stromu, stačí každému jmennému serveru pouze jediná informace jak vyhledat libovolný uzel ve stromu, a to sice adresa kořenového serveru DNS.

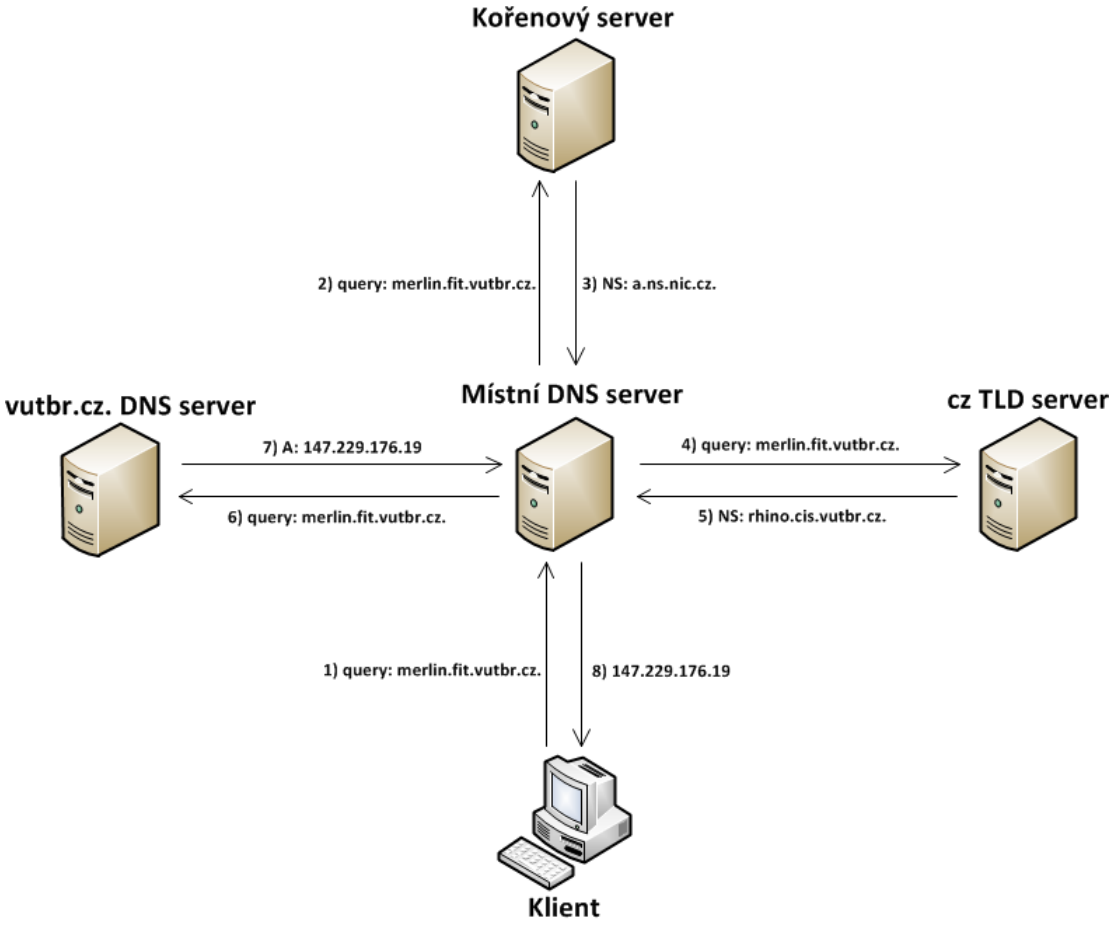
Pokud jmenný server žádnou informaci o dotazovaném jménu v zóně nebo v cache nemá, zeptá se právě kořenového serveru. Kořenový server je autoritativní pro všechny domény nejvyšší úrovně a odpoví buď přímo hledanou odpovědí, nebo vrátí adresu jmenného serveru, který informaci obsahuje. Dotaz je tedy následně směrován na některý z autoritativních serverů domény nejvyšší úrovně, v níž se nachází cílové jméno. Ten je opět schopen poskytnout informace o své doméně a posunout řešení o jedno patro níže v doménovém stromě. Tímto způsobem řešení postupuje po jednotlivých patrech doménové hierarchie směrem k cíli, až se dostane k serveru autoritativnímu pro hledané jméno, který pošle definitivní odpověď. Ukázka vyhodnocení dotazu na překlad doménového jména `merlin.vutbr.cz` je uvedena na obrázku 2.3.

2.5 Zabezpečení DNS

DNSSEC (zkratka pro DNS Security Extensions) je rozšíření systému doménových jmen za účelem zvýšení bezpečnosti a ochraně proti některým typům útoků, jako například otrávení DNS cache paměti [5].

DNSSEC zavádí asymetrickou kryptografii. Používá dvojici klíčů, skládající se ze soukromého a veřejného klíče, která je vygenerována držitelem domény. Svým soukromým klíčem pak elektronicky podepíše všechny záznamy, které o své doméně do DNS vkládá. Pomocí veřejného klíče je pak možné ověřit pravost tohoto podpisu. Držitel klíče vždy publikuje tento klíč u nadřazené autority. Pro všechny .cz domény je touto autoritou registr domén .cz. Veřejná část klíče je tedy podepsána klíčem nadřazené autority. Čili problém důvěryhodnosti se posouvá „o jedno patro výše“. Důvěřuje-li klient zdejšímu klíči, může si postupně ověřit všechny záznamy. V opačném případě může použít stejný postup a postupovat tak v hierarchii ještě výše.

Kořenová doména je podepsána také a za předpokladu, že všechna patra v hierarchii podporují DNSSEC, stačí, aby klient měl důvěryhodný veřejný klíč pouze k této doméně,



Obrázek 2.3: Ukázka dotazu v běžném provozu DNS

a od ní by pak byl schopen rozvinout řetěz důvěry až ke zkoumanému záznamu. Řetěz důvěry je posloupnost střídavě tvořená záznamy veřejných klíčů DNS (DNSKEY) a záznamy Delegation Signer (DS) obsahující otisk klíče [2]. Každý článek řetězu tak ručí za ten následující. DNSKEY záznam je použit k ověření podpisu DS záznamu, který díky němu může být autentizován. DS záznam obsahuje charakteristiku dalšího DNSKEY záznamu, která je použita k ověření jeho pravosti. Tímto způsobem se postupuje až k poslednímu DNSKEY záznamu, k němuž korespondující soukromý klíč byl použit k podpisu požadovaných dat.

Tento mechanismus přinesl změny do samotného DNS protokolu. Především přidal další typy DNS záznamů [4] [16]:

- RRSIG – záznam s podpisem pro každý RRset. RRset je množina záznamů stejného typu k danému doménovému jménu. Podepisují se autoritativní RRsety [3].
- DNSKEY – záznam s veřejnými klíči
- DS – záznam s otiskem klíče pro ověření jeho pravosti
- NSEC3 – záznam sloužící k autentizovanému popření existence záznamu v dotazovaném RRsetu. Nahrazuje záznam NSEC, který se prakticky již moc nepoužívá, neboť umožňuje výčet zóny.

- NSEC3PARAM – obsahuje parametry NSEC3 záznamu (hašovací algoritmus, příznaky, počet iterací a náhodnou hodnotu anglicky označovanou *salt*) nutné pro výpočet hashe jména vlastníka RR záznamu autoritativním serverem.

DNSSEC tedy zajišťuje [2]:

√ autentizaci pravosti DNS dat

√ integritu dat

√ autentizované popření existence dotazovaného

DNSSEC ovšem nezajišťuje bezpečnost dat (jejich šifrování) a také nechrání před útokem typu DDoS, který je popsán dále.

2.6 Shrnutí kapitoly

V této kapitole byla popsána základní struktura systému DNS, jeho principy a ukázka fungování. Dále bylo krátce představeno bezpečnostní rozšíření DNSSEC a jaké změny do komunikačního protokolu DNS přineslo. V následující části se podíváme na vlastní provoz systému DNS, jeho charakteristiku a jak tato může ovlivnit případnou klasifikaci.

Kapitola 3

Charakteristika legitimního provozu DNS

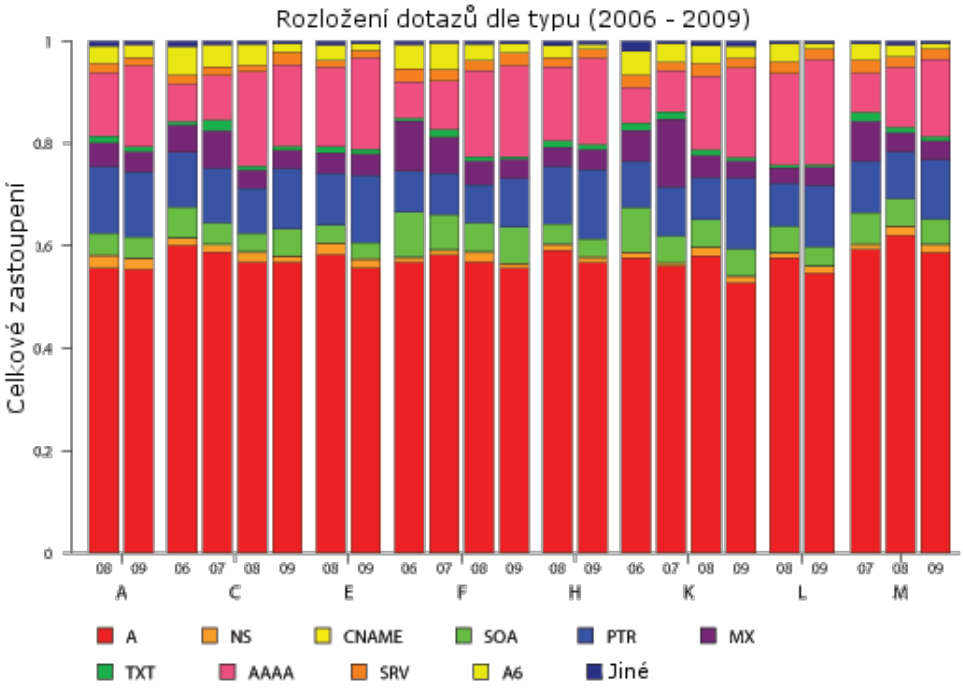
Tato práce je psána s ohledem na možné nasazení na českém ccTLD jmenném serveru, proto se zde zaměřím především na charakteristiku tohoto provozu, jehož vlastnosti se od nižších úrovní DNS hierarchie liší. Jednou z charakteristik tohoto provozu je, že na této úrovni je viditelná pouze část všech dotazů, jelikož značná část je zodpovězena na nižších úrovních. Na vyšších úrovních dochází k jistému odstínění díky tomu, že odpovědi jsou na serverech dočasně ukládány do pamětí (DNS cache), ve kterých mohou záznamy setrvat po nastavenou dobu. Mezilehlé servery tak nemusejí zatěžovat kořenové a TLD servery každým dotazem.

Když se například server dozví, že `a.gtld-servers.net` a další jsou autoritativní pro `.cz` zónu, tak se také dozví dobu, po kterou může být tato informace uložena a je považována za platnou, tzv. TTL (*time-to-live*). Typické hodnoty TTL pro TLD jsou v řádu dní. Teoreticky by se tedy záložní rekurzivní nameserver měl do vypršení platnosti TTL dotazovat kořenových serverů pouze na neznámé domény nejvyšší úrovně. Skutečnost však ukazuje, že kořenové servery dostávají těchto dotazů více, než by měly [26].

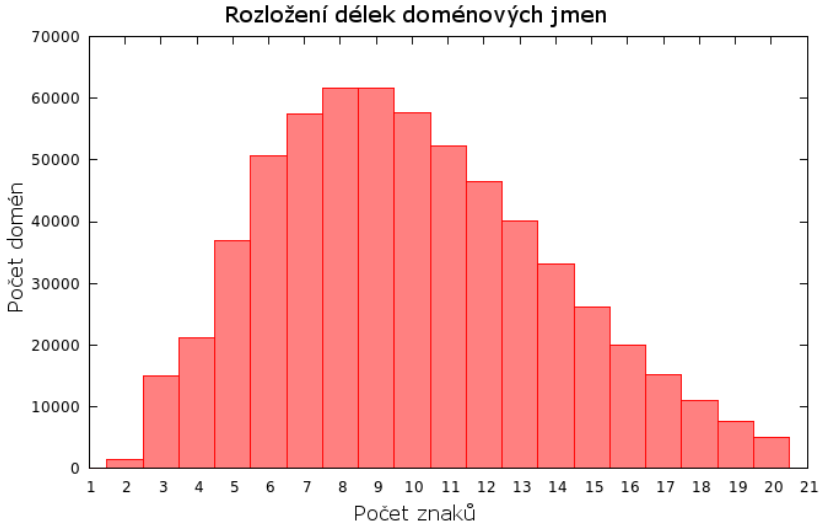
Další vlastností je množství zpráv, které servery přijímají. Dle některých zdrojů [26] správci kořenových serverů DNS uvádí vytížení přesahující 100 miliónů dotazů za den. Architektura i vytížení TLD jmenných serverů jsou podobné jako v případě kořenových. Některé z nich jsou dokonce mnohem vytíženější. S jistotou lze také říci, že došlo v posledních letech k nárůstu [9] objemu dotazů. Ovšem na základě pozorování dotazů směřovaných na dva kořenové a třináct TLD serverů je nutno podotknout, že část celkového množství všech těchto dotazů mohou tvořit zbytečně opakované dotazy (*requery*) [15] způsobené nevyhovujícím resolverem, chybným filtrováním paketů, špatně nastaveným serverem DNS či chybnou implementací aplikace. Opakovaný dotaz sestává ze stejných položek (<QNAME, QTYPE, QCLASS>) a je opakován znovu s neobvykle vysokou frekvencí a to buď z jedné či více IP adres. Takovéto chyby dále zvyšují množství anomálií v DNS provozu.

Převážnou většinu DNS provozu tvoří dotazy na A záznamy, jak můžeme vidět v následujícím grafu na obrázku 3.1. Stejně tak je patrný stálý nárůst dotazů typu AAAA reflektující částečný rozvoj IPv6. Zajímavý je také pokles MX dotazů i přes to, že počet klientů odesílajících MX dotazy se více než ztrojnásobil [9].

Jednoduchá statistika na obrázku 3.2 uvádí rozdělení českých domén v registru z hlediska jejich délky (počtu znaků). Průměrná délka doménového jména se pohybuje okolo devíti znaků.



Obrázek 3.1: Zastoupení jednotlivých typů dotazů v provozu[1]

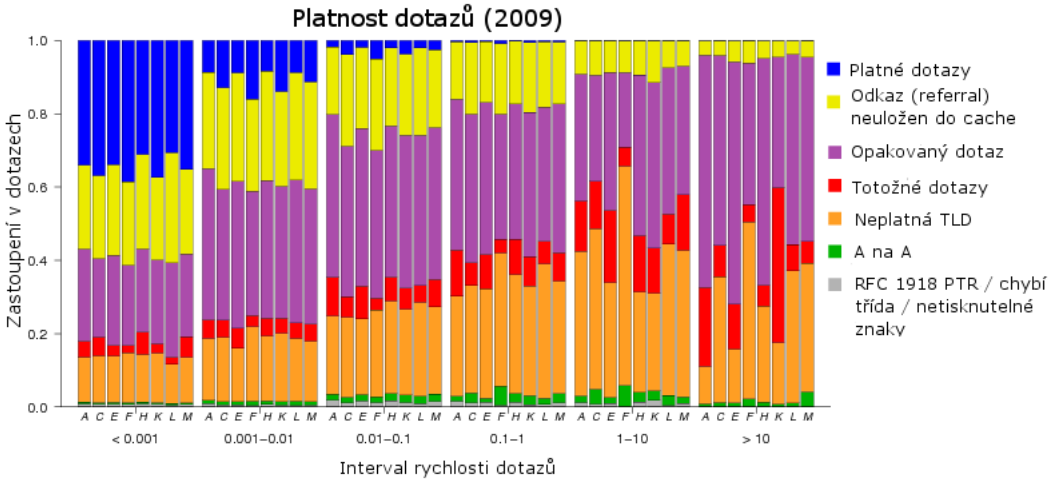


Obrázek 3.2: Rozložení délek českých doménových jmen dle CZ.NIC[2]

Graf na obrázku 3.3 ukazuje procentuální množství platných dotazů v závislosti na intenzitě dotazování na vybraných kořenových serverech. Z grafu je patrné, že množství

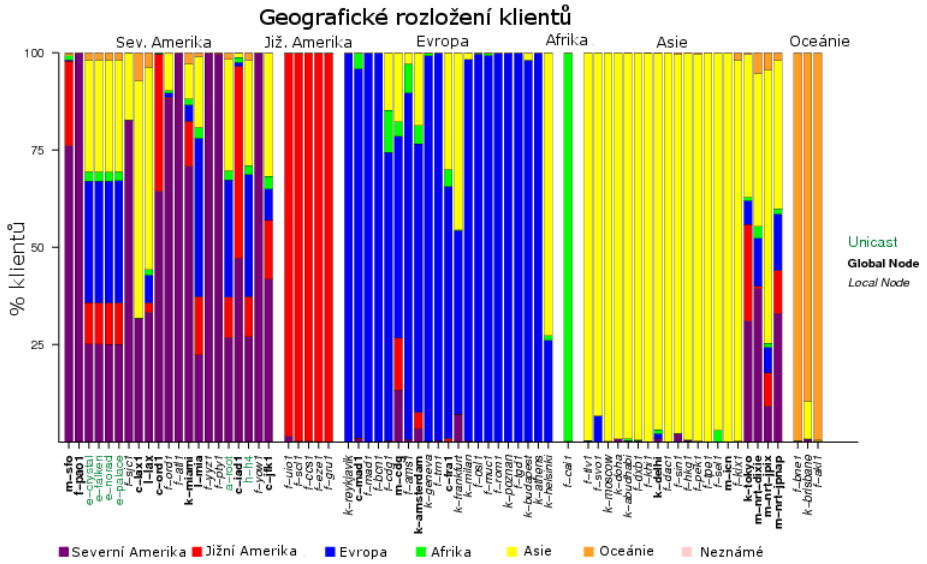
[1] Zdroj: http://www.caida.org/research/dns/roottraffic/evolution/interactive-graphs/
[2] Zdroj dat: http://blog.nic.cz/wp-content/uploads/2010/05/DOMAIN-REPORT-2009-CZ-final.pdf

chybných dotazů je překvapivě vysoké. Dokonce platí, že mezi klienty s vyšší intenzitou dotazů je poměr mezi chybnými a platnými dotazy mnohem horší.



Obrázek 3.3: Poměr mezi platnými a chybnými dotazy v závislosti na různých intenzitách dotazování od náhodně vybraného vzorku 10% klientů[3]

Z geografického hlediska nejspíše není překvapením, že většina serverů je dotazována z míst, kde se fyzicky nacházejí. U kořenových serverů je to především díky technologii *anycast*, která má za cíl právě směrování dotazů na nejbližší instance, viz obrázek 3.4.

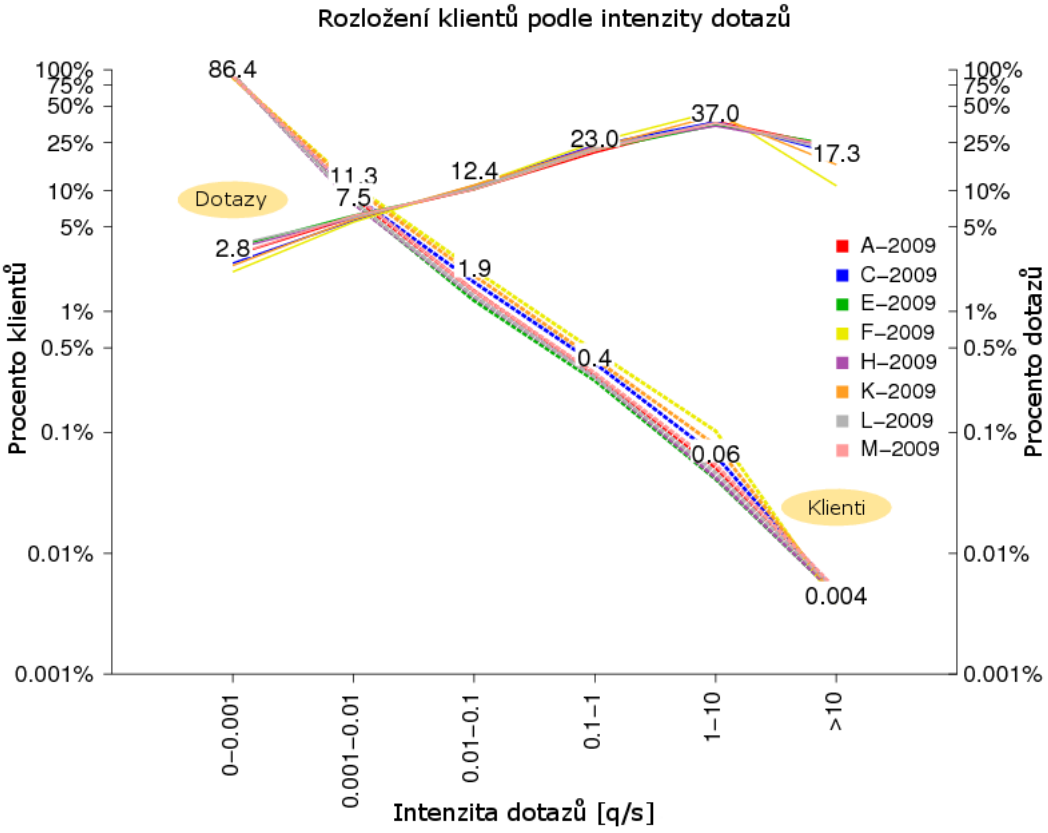


Obrázek 3.4: Geografické rozložení klientů[3]

Mezi přetrvávající charakteristiky provozu patří vysoká proměnlivost v intenzitě dotazování jednotlivých klientů. V grafu na obrázku 3.5 je zachycen vztah mezi počtem tazatelů

[3]Zdroj: http://www.caida.org/research/dns/roottraffic/evolution/interactive-graphs/

a intenzitou jejich dotazů na různých kořenových serverech. Z údajů vyplývá, že za více než polovinu dotazů je zodpovědná pouze hrstka klientů.



Obrázek 3.5: Graf poukazující na existenci silných tazatelů[4]

Servery DNS komunikují s ostatními servery a klienty nad protokoly TCP i UDP, v obou případech na portu 53. Při běžných dotazech se požadavek posílá jedním paketem UDP. Tento protokol byl zvolen pro svou jednoduchost a minimální režii, kdy se nemusí kvůli malým datům složitě navazovat spojení přes protokol TCP. Protokol UDP neřeší ztráty paketů po cestě, ale ve většině případů je postačující.

Graf na obrázku 3.6 ukazuje počet unikátních klientů v každém z desetiminutových intervalů na jednotlivých serverech i celkově. Útlum uprostřed nastal během víkendu. Stejně tak je možné odhadnout pokles během nočních hodin, především u českého ccTLD serveru, jelikož obsluhuje převážně klienty ze své země.

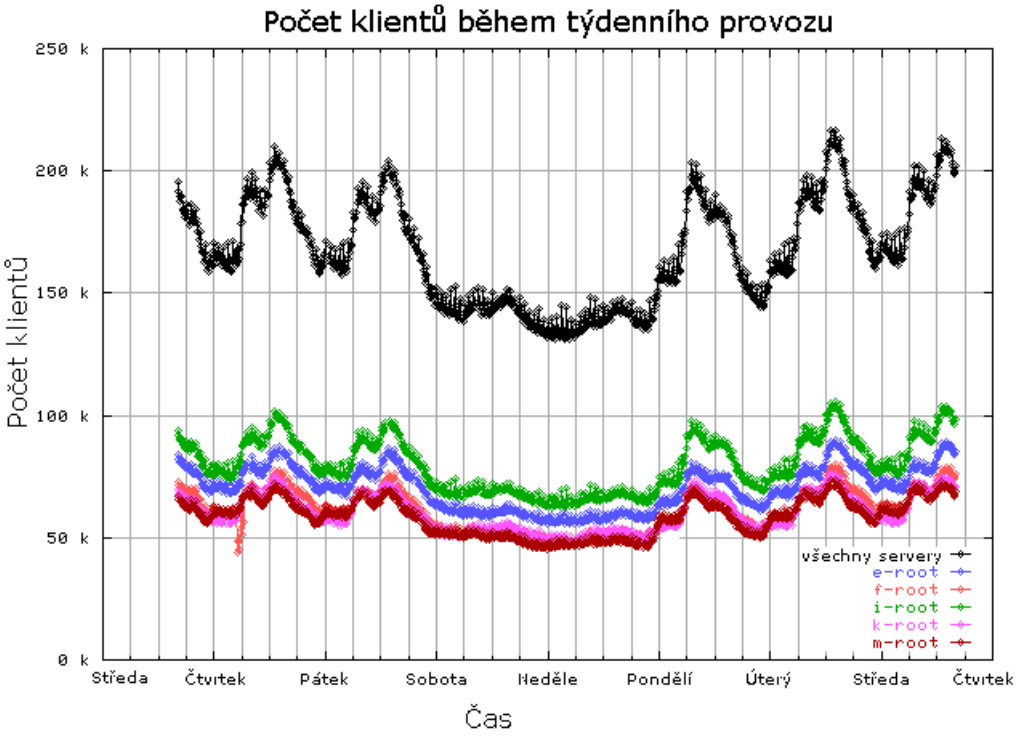
Souhrn charakteristik

Hlavní charakteristiky DNS provozu jsou tedy následující:

√ velké množství dotazů

√ nejčastější dotazy na záznamy typu A

[4]Zdroj: http://www.caida.org/research/dns/roottraffic/evolution/interactive-graphs/



Obrázek 3.6: Počet klientů jednotlivých kořenových serverů během jednoho týdne pro všechny typy dotazů[5]

- √ část dotazů jsou zjevné překlepy, které ovšem také vedou na platné adresy s převážně pochybným obsahem
- √ intenzita dotazů je ovlivněna střídáním dne a noci a také střídáním pracovních dnů s víkendem či svátky
- √ existuje značné množství zbytečného provozu
- √ na vyšších úrovních hierarchie DNS je viditelná pouze část provozu
- √ největší zátěž serveru má na svědomí hrstka klientů
- √ provoz na portu 53
- √ geografický původ klientů serverů je často z okolí (země), kde se daný server nachází

Zde prezentované odchylky souvisejí převážně se špatnou konfigurací a chybně napsanými aplikacemi, což vede ke zbytečnému navyšování objemu dat v provozu. V další kapitole bude uvedeno rozdělení anomálií, které bude zahrnovat kromě zde uvedených problémů i anomálie, jenž jsou důsledkem různých útoků a projevy škodlivých programů.

[5]Zdroj: http://www.caida.org/research/dns/roottraffic/evolution/interactive-graphs/

Kapitola 4

Druhy anomálií v DNS provozu

DNS je otevřený standard a nemá žádné šifrování, a tudíž existuje množství útoků na DNS systémy, které se v provozu projevují nejrůznějšími anomáliemi. Včasná detekce těchto anomálií může vést k odhalení útoků, které je způsobují. Dalšími zdroji jsou také chyby způsobené špatnou konfigurací či chybným návrhem aplikací, jak bylo naznačeno v předchozí kapitole. V této části budou popsány anomálie v provozu DNS, jejich vlastnosti, jak se projevují a možnosti jejich detekce. Níže uvedené rozdělení jsem vytvořil na základě prostudovaných článků, které jsou v jednotlivých sekcích citovány.

4.1 Kontrola správnosti DNS dat na portu 53

Jako první a nejjednodušší metodu lze uvést detekci samotných DNS dat. Jakýkoliv paket neobsahující DNS data je podezřelý, jelikož port 53 je vyhrazen právě pro DNS provoz a žádná jiná aplikace nemá důvod tento port používat také [11]. Jakákoliv jiná data jsou tedy samy o sobě klasifikována jako anomálie.

4.2 Otrávení paměti cache (Cache poisoning)

Tento útok spočívá v tom, že když se uživatel zeptá lokálního serveru, je serveru podstrčena falešná odpověď vytvořená útočníkem předtím, než dorazí skutečná odpověď. Lokální server tak vrátí podvržená data a sám si je uloží do své cache. Tento útok může typicky vést k přesměrování na podvodnou stránku nebo k odmítnutí služby. Částečnou ochranu před tímto útokem nabízí generování náhodných zdrojových portů klientů, ale především technologie DNSSEC zabraňuje tomuto typu útoku.

4.3 Detekce napadených zařízení

Napadené počítače umožňují útočníkovi převzít nad nimi kontrolu, nazývají se potom *boti* nebo *zombie*. K tomuto napadení často dochází jako důsledek spuštění škodlivého programu (tzv. *malware*) na klientském zařízení. Existují obvykle celé sítě infikovaných počítačů, nazývané *botnety*, který obvykle sestávají z napadených počítačů rozptýlených po celém světě.

Tradiční návrh měl centralizovanou strukturu a botnety i další škodlivý software tak využívali pro lokalizaci svého řídícího (Command and control, $C\&C$) serveru a následně

ke komunikaci s útočníkem služeb DNS [1]. Odpovědí na tyto praktiky jsou seznamy (blacklisty) doposud odhalených doménových jmen, které slouží k odhalení kompromitovaných koncových stanic, které se pokoušejí s nimi spojit. Tedy pokud při zpracovávání dotazu DNS server narazí na takovou domény, nebude se překlad provádět. Efektivita toho přístupu je ovšem velice nízká, jelikož ohromné množství nových doménových jmen se objevuje každý den a útočníci je často střídají, tudíž je velice složité udržovat seznamy zakázaných domén aktualizovány. V poslední době je však možné setkat se i s architekturou na bázi peer-to-peer modelu, který ještě více ztěžuje možnost odhalení řídícího centra skrze analýzu provozu [25].

C&C servery často mění doménová jména a tudíž mohou sklouznout k používání náhodně vygenerovaných řetězců. Této vlastnosti bychom mohli využít pro jejich detekci, která by ovšem nemusela být dostatečně přesná především s ohledem na skutečnost, že jedno doménové jméno nemusí obsahovat všechny znaky a jedná se o velmi malé množství dat pro nějakou spolehlivější analýzu. Ke zpřesnění detekce by bylo výhodné uchovávat si historii předchozích překladů, což by vzhledem k množství domén mohlo způsobit značné navýšení paměťových nároků.

Dalším ukazatelem napadení může být nárůst dotazů typu MX z jedné koncové stanice. Za tímto nárůstem může být činnost spambota rozesílajícího nevyžádanou poštu.

4.4 Detekce domén spojených se škodlivými programy

Domény spojené s různými škodlivými programy jsou často dotazovány z velmi široké množiny rozdílných stanic, typicky z velkého počtu sítí různých velkých poskytovatelů připojení k Internetu (ISP). Důvodem k tomu je to, že poskytovatelé typicky neposkytují ochranu koncových stanic proti škodlivým programům a proti jejich šíření. Navíc takové sítě poskytují přístup velkému množství uživatelů a tudíž pravděpodobnost, že nějaká koncová stanice je napadena, je poměrně vysoká.

Naproti tomu legitimní domény jsou běžně dotazované jak ze strany sítí poskytovatelů, tak ze strany menších organizací, jejichž podnikové sítě jsou obvykle lépe zabezpečeny proti takovým hrozbám a z jejich strany nedochází k tak velkému počtu dotazů na domény spojené se škodlivými kódy [1].

4.5 Detekce DNS tunelu

DNS pakety mohou být zneužity k vytvoření skrytého datového kanálu. Typicky je k tomu zapotřebí upravený DNS klient a server. Oba jsou však navrženi tak, aby mohli pracovat s nemodifikovanými rekurzivními DNS servery. Vytvořený kanál může být použit k přenosu tajných informací skrz firewall, který měl právě tomuto zabránit. Jedná se o ideální prostředek, vzhledem k tomu, že DNS data většinou nejsou podrobována hlubší analýze a DNS port je často otevřen.

K vytvoření tunelu mohou být použity všechny typy záznamů (NULL, TXT, SRV, MX, CNAME, A) podle toho, jak velká šířka pásma je dostupná [11], neboť maximální délka doménového jména je omezena.

Pro detekci tohoto skrytého kanálu lze zkoumat vlastnosti paketu nebo inspekci dat samotných. Průměrná délka jednoho milionu nejpoužívanějších doménových jmen je deset znaků. Průměrná délka dotazu v DNS tunelu, který je použit k přenosu náhodného souboru je přes třicet znaků [11]. Dotazy typicky nepřesahují 312 bytů (včetně TCP a IP hlaviček)

a UDP odpovědi většinou nepřesahují 512 bytů. V kombinaci s časovou analýzou mohou být tyto přenosy velkých paketů rozpoznány.

Další možnost detekce skýtá frekvenční analýza. Většina doménových jmen je tvořena lidmi a tedy slovy z přirozeného jazyka, doménová jména tudíž často sledují charakteristiku tohoto jazyka. Tato metoda je například používána v kryptografii pro zjištění charakteristik daného zašifrovaného textu s cílem namapovat tuto charakteristiku na známé rozložení daného přirozeného jazyka. Analýzu lze použít všude tam, kde očekáváme člověku srozumitelné data, která snadno odlišíme od jinak vygenerovaných. Pokud lze správně předpovědět jazyk daného dotazu, pak lze jeho charakteristiku porovnat s očekávaným jazykem a na základě toho rozhodnout, zda doménové jméno bylo vymyšleno člověkem.

Nevýhodou frekvenční analýzy je, že pro každý jazyk bychom museli mít vlastní tabulku rozložení četností jednotlivých znaků. Kdyby byl kanál použit pro přenos zašifrovaných dat, pak by rozložení jednotlivých četností bylo z principu téměř rovnoměrné. Tato odchylka od přirozeného jazyka by byla snáze identifikovatelná.

4.6 Metoda rychlého přepínání adres (metoda fast-flux)

Tato metoda[1] slouží k rychlému přepínání IP adres odpovídajících určitému plně kvalifikovanému doménovému jménu. Těchto adres obecně mohou být stovky nebo i dokonce tisíce. Tyto adresy jsou přiřazovány a odnímány jednotlivým A a/nebo NS záznamům s velmi vysokou frekvencí ve stylu kruhové obměny díky velmi krátkým dobám TTL jednotlivých DNS záznamů. Jméno webové stránky tak může být spojeno s jistou často se měnící množinou IP adres. Prohlížeč, který se během krátké doby připojí k téže stránce opakovaně, se vlastně pokaždé připojí k jinému (napadenému) počítači. Tato technika tedy slouží se k rychlému přesměrování provozu v případě, že původní server selže nebo je třeba zahlcen útokem DDoS. Příkladem legitimního použití najdeme například u velkých poskytovatelů obsahu, jako například Akamai [14].

Naneštěstí tato technika nalézá uplatnění i při nelegitimních činnostech. Phishing (někdy převáděno do češtiny jako rhybaření), pharming (někdy též farmaření) a jiné škodlivé (často ilegální) aktivity představují známou bezpečnostní hrozbu pro uživatele. Infikované stanice v botnetu se pak stávají poskytovateli tohoto obsahu. Skupiny zapletené do těchto aktivit následně využívají techniky rychlého přepínání adres pro maření snahy o jejich lokalizaci a ukončení těchto nekalých operací. Navíc se útočníci snaží o co nejlepší zajištění svých služeb vybíráním stanic s linkami s největší propustností a rovnoměrného rozložení zátěže [21]. Díky rychlým a neustálým změnám topologiím svých sítí, ze kterých poskytují obsah podvodných stránek, se snaží zůstat v utajení před zákonnými složkami.

Ač je nízká hodnota TTL dobrým ukazatelem výskytu, nejedná se o dostatečnou informaci pro označení dané domény využívající tuto techniku v botnetu.

Mezi hlavní znaky použití této techniky patří [14]:

- velice malá hodnota TTL
- několik IP adres na jeden NS rozprostřené přes několik autonomních systémů (označených ASN)
- časté změny NS

[1] Též překládáno jako botnet rychlého běhu, viz http://www.root.cz/clanky/bezpecnostni-stripky-jak-se-naplnuji-bezpecnostni-predpovedi-pro-rok-2010/

- in-addr.arpa nebo IP adresa leží uvnitř alokačního bloku pro koncové uživatele se širokopásmovým připojením
- stáří doménového jména (tuto informaci mají pouze registrátoři domén ovšem)
- podvodné údaje v databázi `WHOIS`
- výsledkem překladu doménového jména je vyšší množství A záznamů

Časté přepínání mezi napadenými stanicemi velmi dobře zakryje skutečnost, že se jedná o obyčejné klientské počítače, které nejsou jako poskytovatelé příliš spolehlivé. Přidáním další vrstvy mohou tyto sítě docílit vyšší dynamiky a lepšího skrytí v síti. Tato technika se nazývá *double-flux* a dochází při ní ke změnám NS i A záznamů. Aby tato technika mohla fungovat, musí registrátor domény umožňovat tyto časté změny NS záznamů, která nebývá obvykle příliš častá [21]. Řídící stanice jsou podobné těm v botnetech, ale nabízejí více funkcí. Právě tyto stanice poskytují požadovaný obsah, avšak jsou skryty za vrstvou botnetu. Při útoku tak budou DNS záznamy ukazovat na kompromitované systémy, které fungují jako proxy servery. Tento způsob je také příjemnější pro útočníka v tom, že namísto kopie podvodného obsahu na jednotlivé boty, stačí umístit obsah na jeden stroj a boty využívat pouze pro přesměrování. Drtivá většina (95-99%) všech těchto domén byla odhalena v `.cn`, `.com` a `.net`, ovšem i nás je možné se s několika doménami tohoto typu setkat [14].

4.7 Distribuované odmítnutí služby

Díky významné úloze, kterou DNS servery hrají na Internetu, jsou sami často také cílem různých útoků, jako například distribuovaného odmítnutí služby (*Distributed Denial of Service, DDoS*) [27]. Účelem tohoto útoku je zahlcení daného serveru takovým způsobem, že dojde k vyčerpání všech jeho zdrojů s cílem způsobit omezení služeb legitimním uživatelům nebo přímo pád serveru.

Pokud je odpověď na dotaz uložená již lokálně díky cache, pak tento server nemusí hledat odpověď u autoritativních serverů, než příslušný záznam exspiruje. Zátěž autoritativních serverů je tedy značně snížena právě díky záznamům v DNS cache lokálních DNS serverů.

Za účelem eliminace tohoto (obyčejně žádoucího) účinku DNS cache je možné pokládat mnoho dotazů na neexistující doménová jména, která jsou náhodně vygenerována, a tedy nebyla zatím dotazována nikým jiným. Což tedy znamená, že se určitě nenacházejí v paměti lokálního serveru a dotaz tak poputuje až k cílovému autoritativnímu serveru. Velké množství dotazů se spoustou NXDOMAIN odpovědí tak mohou být známkou útoku, pokusu o slepé procházení zónou nebo poukazovat na nevyžádanou poštu. Konkrétní určení závisí na typu dotazovaného záznamu (NS, A, MX).

4.8 Výčet zóny (Zone walking, Zone enumeration)

DNS je jeden z prvních kroků při jakékoliv síťové komunikaci. Nejinak tomu je i v případě mnohých útoků a činnostech škodlivých programů. Většina útoků začíná nejprve nějakým vyhledáním v síti – ať už vyhledáním dostupných zařízení či software na těchto zařízeních běžící, který obsahuje známé bezpečnostní chyby. V případě, že útočník zná přesný obsah DNS zóny, tak je pro něj toto zjišťování o něco jednodušší. DNS záznamy navíc mohou obsahovat data pro další aplikace. Pokud tedy útočník dokáže vyčíst obsah cílové zóny, může mít usnadněnu práci při následném pokusu o útok, jelikož bude se systémem již

obeznámen. Toto platí zvláště pro IPv6 sítě, kde na rozdíl od IPv4 sítí, je výčet DNS zóny jednodušší, než procházení celého adresového bloku [23].

Výčet může být získán například hrubou silou – postupným dotazováním se na jména v lexikografickém pořadí nebo pomocí slovníku. Alternativní možností je průchod pomocí PTR dotazů. Pro vylepšení svých šancí může útočník využít distribuovanou variantu tohoto útoku, která značně ztěžuje možnost jeho odhalení.

Bezpečnostní rozšíření DNSSEC přidalo nové typy záznamů pro ukládání veřejných klíčů, podpisů a také typ pro autentizovanou odpověď, pokud dotazované jména neexistuje. Poslední zmiňovaný záznam se označuje NSEC a poskytuje podepsanou odpověď při chybových situacích, jako jsou právě dotazy na neexistující jména.

NSEC záznam poskytuje důkaz neexistence tím, že vrátí dvě platné jména z dané zóny, mezi kterými by se hledané jméno nacházelo, kdyby existovalo. Když se tedy klient dotáže například na IP adresu `b.priklad.cz` a je mu vrácen NSEC záznam s `a.priklad.cz` a `c.priklad.cz`, pak ví, že `b.priklad.cz` neexistuje, jelikož spadá do rozsahu jmen mezi `a.priklad.cz` a `c.priklad.cz`. Tato odpověď je podepsána, aby bylo zaručeno, že pochází od autoritativního serveru dané zóny. Vedlejším efektem je, že klient nyní zná dvě jména, která v zóně jsou a o kterých předtím nemusel ani vědět. NSEC záznamy tvoří v zóně řetězec pokrývající celý jmenný prostor provázaný skrze tyto záznamy. Tyto informace může útočník zneužít k průchodu celou zónou zasíláním následných dotazů na jména následující po `c.priklad.cz`, dokud nepokryje celou zónu. Pro získání obsahu zóny s N jmény by bylo zapotřebí N dotazů k získání celého obsahu.

Jedním z prvních navržených řešení spočívalo v zavedení NSEC odpovědí, které pokrývaly pouze nejmenší možný omezující rozsah jmen okolo chybného dotazovaného doménového jména. Toto opatření vedlo ke zvýšení ceny pokusu o výčet zóny tímto způsobem na cenu srovnatelnou jako v případě útoku hrubou silou, jelikož útočníkovi neodhalí žádné z existujících jmen a odpověď pokrývá jen dotazované jméno. Pro server je však nezbytné vytvářet NSEC záznamy a podepisovat je za běhu. V případě, že by útočník byl schopen odeslat velké množství dotazů, by tato vlastnost mohla rychle vést k útoku typu odmítnutí služby.

Vylepšenou variantu původního NSEC záznamu nabízí NSEC3 záznamy, které při negativních odpovědích nepoužívají doménová jména přímo, ale pouze jejich charakteristiky (získané pomocí jednocestné funkce jako je např. SHA-1). Nejedná se však o dokonalé řešení, jelikož je stále možné získat kompletní výčet jmen dané zóny. K jeho zisku je však zapotřebí vynaložení většího úsilí, je totiž zapotřebí nejprve zjistit seznam charakteristik všech jmen a poté zaútočit na tento seznam.

Dalším řešením je zpřístupnění pouze takových DNS záznamů, které by měly být veřejně dostupné. Ostatní záznamy mohou zůstat v odděleném jmenném prostoru přístupném pouze oprávněným tazatelům, například pouze zaměstnancům v dané korporátní síti.

4.9 Shrnutí kapitoly

Tato kapitola přinesla přehled anomálií v provozu systému DNS, jejich vlastnosti a jak se projevují. Útočnící se neustále snaží být o krok napřed a pokoušejí se svou činnost maskovat. Třebaže některé anomálie mohou být rozpoznatelné triviálně ve své čisté teoretické podobě, ve skutečnosti budou mnohem sofistikovanější. Zavedení distribuované podoby jakéhokoliv útoku posunuje šance na rozpoznání na zcela jinou úroveň.

Poznatky uvedené v této kapitole budou využity v kapitole 6. Nejprve však budou popsány nástroje použitelné pro tvorbu klasifikátoru těchto anomálií.

Kapitola 5

Možnosti klasifikace anomálií

V této části je nejprve obecně popsán problém klasifikace a jsou zde nastíněny metody použitelné pro klasifikaci

Anomálií rozumíme chování, které je v rozporu s očekáváním, je něčím výjimečné nebo se jakkoliv odchyluje od obecného pravidla či normálního jevu. Může tak být důsledkem probíhajícího útoku nebo například činnosti škodlivého programu. Včasná a přesná detekce těchto hrozeb je stále aktuálním tématem pro bezpečnost na Internetu. Je velice žádoucí tyto hrozby detekovat již v jejich zárodku, dokud nejsou rozšířeny a jsou schovány za velkým objemem běžného provozu, což může být velice obtížné.

Jednotlivé anomálie se liší svou povahou, dobou trvání a příčinami vzniku. Provoz na Internetu se mění každý den, a proto mohou přibývat nové typy anomálií a statické metody rychle zastarávají. Navíc i běžný provoz sám o sobě vykazuje značkou proměnlivost, což se může projevit na schopnostech a úspěšnosti detekce.

Metody pro detekci anomálií využívají poznatků z oblasti umělé inteligence, statistiky, teorie informací a dalších oborů. Je zřejmé, že ne každá z uvedených metod je vhodná právě pro DNS provoz. Některé zase slouží pouze pro detekci a nikoliv přímo k určení toho, o jakou anomálii se jedná.

5.1 Klasifikace

Formálně lze problém klasifikace (neboli zařazování do tříd) definovat následovně [10]: Nechť y je náhodná proměnná z prostoru $\mathcal{Y} = \{C_1, ..., C_K\}$ reprezentujícího množinu tříd - y tedy může být jednou z K tříd. Dále uvažujme m-rozměrný vektor prediktorů $\mathbf{X} = (X_1, ..., Xm)$ z prostoru $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_m$. Mějme reprezentaci náhodného výběru z prostoru $\mathcal{Y} \times \mathcal{X}$ obsahující n položek, kterou nazveme *trénovací množinou*. Potom klasifikátorem nazveme funkci $h : \mathcal{X} \longrightarrow \mathcal{Y}$, která zobrazuje pozorování $(X_{1i}, ..., X_{mi})$ na nějakou třídu $\hat{y}_i$. Klasifikátor přiřazuje tuto třídu na základě informací z trénovací množiny. Říkáme, že se klasifikátor „učí“ na trénovací množině.

Tento proces se skládá z následujících kroků [24]:

1. učení – dochází ke tvorbě klasifikačního modelu schopného klasifikovat data pomocí trénovacích dat. Metody učení dále rozdělujeme na:

    - učení s učitelem - u trénovacích vzorků dat známe výsledek klasifikace, tj. třídu, do které patří

- učení bez učitele - neboli shlukování, výsledek klasifikace neznáme a potenciálně neznáme ani počet tříd
- posilované učení - učící se agent nemá učitele, učí se z pokusů a omylů na základě příležitostných odměn

2. vlastní klasifikace – použití modelu pro klasifikaci nových dat a jejich zařazení do příslušných tříd.

Před samotnou klasifikací je vhodné provést úpravy dat [24]:

- čištění dat - odstraňuje se šum nebo se snižuje jeho míru a doplňují se chybějící hodnoty
- určení relevance dat - záznamům lze přiřadit váhu na základě jejich významu, případně odstranit málo relevantní a redundantní data
- transformace dat - jejím účelem je přizpůsobení dat použitému klasifikačnímu modelu

Není snadné vybrat vhodnou klasifikační metodu, neboť jich existuje velké množství od rozhodovacích stromů, přes genetické algoritmy až po neuronové sítě. U všech těchto metod ovšem můžeme typicky vyhodnocovat stejné vlastnosti [28]:

- přesnost - pravděpodobně nejdůležitější atribut, jelikož vyjadřuje míru schopnosti modelu klasifikovat neznámá data
- výpočetní složitost - ovlivňuje, zda může metoda pracovat v reálném čase či nikoliv
- robustnost - určuje odolnost vůči chybám a šumu
- škálovatelnost - účinnost při práci s rozsáhlými množinami dat
- snadnost interpretace

Nyní představíme několik vhodných metod a budeme diskutovat možnosti jejich použití.

5.2 Metody využívající vzdálenosti (podobnosti)

Tyto metody klasifikace využívají skutečnosti, že záznamy zařazeny do stejné třídy musí sdílet jisté vlastnosti, tj. musí si být podobné [24]. Vlastní klasifikace pak spočívá ve zvolení vhodné funkce hodnotící podobnost záznamů. Analyzovaný záznam je klasifikován do třídy, která obsahuje mu nejbližší záznamy.

V praxi se používá klasifikace podle k-nejbližších sousedů (k-nearest neighbours), která předpokládá, že existují trénovací záznamy, které jsou správně oklasifikovány. Vlastní určení třídy nového záznamu probíhá na základě toho, do které třídy patří většina z k trénovacích záznamů, které se nacházejí nejblíže zkoumanému záznamu. Kritériem určení blízkosti záznamů je vzdálenost vektorů v Euklidovském prostoru [24].

Pokud trénovací záznamy neexistují, je možné využít metody nejbližších středů (K-means), která je poměrně oblíbená díky své jednoduchosti a rychlosti. Pracuje tak, že vstupní vektor n bodů rozdělí na k shluků. Středy shluků jsou inicializovány náhodně. Každý bod je přiřazen nejbližšímu shluku. Kritériem je minimální vzdálenost od středu shluku. Středy jsou poté přepočítány na základě nově přidaných dat a celý proces se opakuje dokud nejsou všechny body zařazeny do shluku s nejmenší možnou vzdáleností.

5.3 Neuronová síť

Neuronová síť je datová struktura, jejíž cílem je napodobit chování neuronů v biologickém mozku [28]. Je reprezentována neorientovaným grafem, jehož každý uzel je označen (typicky číslem vrstvy a pořadím uzlu ve vrstvě) a hrany spojují jen uzly ze sousedních vrstev. První vrstva neuronové sítě je tzv. *vstupní vrstva*, poslední vrstva je označována jako *výstupní vrstva*. Pokud existují další vrstvy, jsou nazývány *skryté*. Neuronová síť je nazývána n-vrstvou, právě když má n vrstev výstupních jednotek [24].

Jednotlivé hrany v neuronové síti mají přiřazeny hodnoty - váhy. Data vstupují do neuronové sítě přes vstupní vrstvu, kde vstupem každého uzlu je hodnota části zkoumaného záznamu. Výstup dat je realizován přes uzly výstupní vrstvy neuronové sítě. Při klasifikaci záznamu může být příslušnost k m-té třídě reprezentována například hodnotou 1 na m-tém uzlu výstupní vrstvy a hodnotou 0 na ostatních. Vlastní klasifikace a učení neuronové sítě může probíhat například algoritmem zpětné propagace, který zde nebude uveden.

Mezi klady patří obecně poměrně vysoká přesnost predikce a robustnost. Problémem práce s neuronovými sítěmi je porozumění znalosti, kterou reprezentuje konkrétní klasifikační model v podobě neuronově sítě a dlouhý čas potřebný k natrénování sítě [28].

5.4 Klasifikační a regresní stromy (CART)

Jednou z nejznámějších klasifikačních metod jsou klasifikační a regresní stromy (*Classification And Regression Trees* - CART). Tvoří základ dále popsané metody *náhodný les*, a proto je nutné pochopit jejich princip. Klasifikátor má stromovou strukturu a každý uzel stromu představuje jednu testovanou vlastnost vstupních dat, z tohoto uzlu vede konečný počet hran označujících cesty pro výsledky testů. Listové uzly reprezentují výsledné zařazení do tříd.

Prakticky všechny stromy rostou na základě rekurzivního binárního dělení [28]. Na začátku tvorby stromu patří všechna pozorování trénovacího souboru do jednoho uzlu (kořenu). Následně jsou tato pozorování rozdělena do dvou synovských uzlů. Ze všech možných dělení je vybráno to „nejlepší“, podle kterého je množina rozdělena. Procedura dělení pokračuje dokud dokud nejsou splněna kritéria pro zastavení dělení.

Ta jsou dána třemi podmínkami [10]:

1. Pokud je počet pozorování v uzlu menší, než zvolené číslo.
2. Pokud je v uzlu zastoupena pouze jedna třída.
3. Pokud všechny prediktory v uzlu nabývají stejnou hodnotu.

Pro tyto případy se již uzel dále nedělí. Zbývá nám určit, podle jakých kritérií se výsledná dělení volí.

Kritéria pro dělení uzlů

Cílem dělení je, aby vzorky uvnitř uzlů byly homogenní a zároveň co nejvíce rozdílné mezi sebou. Pro vysvětlení zavedeme si pomocný pojem nečistota. Nečistotou množiny M nazveme nezápornou funkci pravděpodobnosti p, že pozorování patřící do této množiny mají stejnou třídu. Od funkce nečistoty očekáváme, že bude minimální, pokud bude množina M zcela homogenní. Naopak maximální hodnotu očekáváme, pokud budou všechny třídy

stejně zastoupeny. Nejlepším dělením je to, které nejvíce sníží nečistotu dělené množiny.

Uvedené podmínky splňují mimo jiné následující tři funkce:

- Gini index $G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$
- Entropie $H = -\sum_{k=1}^{K} p_{mk} \log p_{mk}$
- Chyba klasifikace $E = 1 - p_{mk(m)}$,

kde p_{mk} je podíl pozorování třídy k v uzlu m. Nejčastěji používané měření pro klasifikační stromy je Gini index - jeho hodnota se rovná nule, pokud je v konečném uzlu pouze jediná třída a dosahuje maxima, pokud je v konečném uzlu v každé třídě stejný počet pozorování.

Výhoda CART spočívá v jejich přehlednosti a snadné interpretovatelnosti, která umožňuje uživatelům rychle a lehce vyhodnocovat získané výsledky. Mezi hlavní nevýhody patří nestabilita, i malá změna v datech může vést na zcela odlišné výsledky. Proto se výsledné CART velmi často prořezávají, neboť velké a složité stromy často vykazují velkou nestabilitu vlivem přeučení. Jelikož v algoritmu náhodných lesů se stromy neprořezávají, nebudeme se této činnosti více věnovat.

5.5 Klasifikační les

Jedná se o klasifikační model vytvořený kombinací určitého počtu klasifikačních stromů za účelem zvýšení celkové přesnosti a stability. Každý strom přiřazuje hodnotě vektoru prediktorů nějakou třídu. Výsledná klasifikace je dána hlasováním nebo jako průměr pravděpodobností (zastoupení kategorie v terminálním uzlu).

Metody vytváření lesů:

√ **Bagging** (**b**ootstrap **agg**regating) – Technika pro zvýšení stability a přesnosti u klasifikačních a regresních metod. Nejčastěji používaná pro rozhodovací stromy, ale použitelná i pro ostatní modely. Z trénovacího souboru T o velikosti n se náhodným výběrem s vracením vytvoří k souborů $T_1,\ldots,T_k$, o velikosti $n' \leq n$. Při každém novém náhodném výběru se vychází vždy ze všech dat a vzorky se tedy mohou v jednotlivých testovacích souborech opakovat. Každý z výběrů se použije na tvorbu klasifikačního stromu.

√ **Boosting** – Celkový výsledek je získán jako vážený průměr výsledků klasifikačních modelů, s váhou určenou podle úspěšnosti jejich dřívějších klasifikací.

√ **Náhodné lesy** – Bagging a boosting se snaží o pěstování co nejpřesnějších stromů, zatímco u náhodných lesů na kvalitě jednotlivého stromu příliš nezáleží, cílem není minimalizovat chybu jednoho stromu, ale celého lesa. Jelikož bude právě tato technika použita k výstavbě klasifikátoru, bude blíže popsána v následující části.

5.6 Náhodné lesy (Metoda Random Forest)

Random Forest je poměrně známá metoda a také program pro získávání znalostí a rozpoznávání vzorů. Zvláště v klasifikaci dosahuje jedny z nejlepších výsledků ve srovnání

s ostatními metodami [22]. Formálně byla metoda definována v roce 2001 a jejím autorem je Leo Breiman [8].

Základem algoritmu je vytvoření spousty klasifikačních stromů. Pro klasifikaci je vstupní objekt předán ke zpracování všem stromům v lese. Každý z nich poté vstup zařadí do nějaké třídy a říkáme, že pro ni „hlasuje“. Les zvolí třídu s nejvyšším počtem hlasů.

Mějme tedy M prediktorů (proměnných) $X_1, ..., X_M$ a množinu trénovacích dat T o velikosti l, pak lze algoritmus popsat následujícími kroky:

1. Zvolíme parametry - počet stromů n, který necháme vyrůst a počet prediktorů m, který se bude podílet na každém dělení.

2. Každý strom je vytvořen podle následujících kroků:

   (a) Trénovací množina dat pro jednotlivé stromy, na základě které jsou vystaveny, je vybrána z datového souboru T stejně jako v případě *baggingu* uvedeném dříve.

   (b) Z M vstupních proměnných (prediktorů), které máme k dispozici, jich je m náhodně vybráno při každém dělení uzlu. Dále se již nejlepší větvení hledá klasicky, ale jen mezi těmi větveními, která jsou založena na vybraných m veličinách.

   (c) Stromy rostou do maximální možné velikosti bez prořezávání. V případě souboru klasifikačních stromů, kterým Náhodné lesy jsou, má prořezávání dokonce škodlivý účinek [7]. Vysvětlením může být skutečnost, že klasifikační soubor využívá rozmanitosti jednotlivých modelů, která je prořezáváním právě omezována.

Růst stromu je časově poměrně náročná operace, neboť při každém rozvětvení může být prohledáno velké množství kombinací. Vzhledem k tomu, že jednotlivé stromy mohou růst nezávisle na sobě, lze tuto činnost dobře paralelizovat.

Breiman svým teorémem 1.2 v [8] dokázal, že při dostatečném počtu stromů nedojde díky platnosti silného zákona velkých čísel k přetrénování modelu při přidávání dalších stromů.

5.6.1 Out-of-Bag odhad chyby

Tento algoritmus nemusí používat další testovací sadu k získání odhadu chyby. Každý strom t_i je vytvořen na základě počátečního výběru s opakováním (tzv. bootstrap), přičemž z teorie pravděpodobnosti plyne, že při výběru z množiny velikosti n s vracením zpět je pravděpodobnost, že prvek nebude vybrán rovna e^{-1}. To znamená, že při každém z těchto výběrů zůstane přibližně třetina pozorování nevybrána [13]. Tato pozorování můžeme využít jako testovací množinu dat nazývanou „out-of-bag“ (zkráceně OOB) [8]. Na jejím základě lze vytvořit odhad chyby predikce, který je s dostatečně velkým počtem stromů relativně přesný, Bylander dokonce ukázal, že je vychýlený směrem nahoru [10].

5.6.2 Význam proměnných

Jednou z přidaných hodnot náhodných lesů je měření významnosti proměnných, které nalézá uplatnění především u problémů s velkým množstvím prediktorů mnohdy obsahujích málo informací. Při klasifikaci jediným stromem CART je velmi jednoduché určit, které prediktory jsou pro klasifikaci významné a které ne. Pro náhodný les skládající se z mnoha stromů je situace poněkud komplikovanější. Doposud není zcela jasné, jaký způsob měření významnosti je nejvhodnější pro náhodné lesy. Dva možné způsoby jsou uvedeny v [6]:

1. Princip snížení nečistoty – Při dělení trénovací množiny v uzlu stromu dojde k zaznamenání hodnoty, o jakou se snížila nečistota (např. Gini index). Míru významnosti proměnné lze potom vyjádřit jako součet všech těchto hodnot zprůměrovaný počtem stromů. Nevýhodou je, že získaná hodnota vychází pouze z trénovacích dat použitých při tvorbě stromů.

2. Princip permutace – Při tomto pracujeme s OOB daty. Pokud hodnoty zkoumané proměnné náhodně permutujeme, dojde k narušení vztahu mezi ní a závislou proměnnou (třídou). Rozdíl míry klasifikační přesnosti stromu před a po permutaci lze využít právě k určení významnosti dané proměnné. Čím významnější proměnná je, tím je rozdíl patrnější.

5.6.3 Matice blízkosti

Dalším produktem náhodných lesů je matice blízkosti (*proximity matrix*). Tato dává informaci o tom, jak moc si jsou jednotlivá pozorování blízká (podobná). Je zkonstruována dle následujících kroků:

1. Nechť P je nulová matice o velikosti $l \times l$, kde l je počet všech pozorování v trénovací množině. Pro každý strom k provedeme:
   (a) k-tý strom necháme klasifikovat všechna data (tedy i OOB).
   (b) Pokud se i-té a j-té pozorovaní vyskytlo ve stejné terminální množině tohoto stromu, zvýšíme ij-tý a ji-tý prvek matice P o jedničku.

2. Celá matice je nakonec normalizována dělením každého jejího prvku počtem pozorování l.

Výsledná matice P je tedy maticí blízkosti. Její prvek P_{ij} ukazuje podíl stromů, které zařadily pozorování i a j do stejné terminální množiny. Podobná pozorování by měla končit ve stejných listových uzlech častěji, než ty, které si podobná nejsou. Matice blízkosti nalézá využití při odhalování odlehlých dat, struktur v datech nebo v případě učení bez učitele.

Z praktického hlediska je vhodné si uvědomit, že tato matice může být poměrně velká a práce s ní tedy poměrně obtížná. Je ovšem symetrická a stačí si uchovávat pouze horní nebo spodní trojúhelníkovou matici.

5.6.4 Nastavení parametrů náhodného lesa

Technika náhodných lesů má pouze několik parametrů. Nejdůležitějšími z nich je minimální velikost množiny, počet stromů n, který bývá typicky v řádu stovek, a počet náhodně vybraných prediktorů m. Tyto parametry by měly být optimalizovány tak, aby minimalizovaly celkovou chybu.

Minimální velikost množiny určuje spodní hranici při dělení množin v CART. Množina s méně prvky, než je tento práh je označena za terminální a nebude se již dále dělit. Čím menší je tato velikost, tím menší je také odchylka, ale větší variance. Díky velkému počtu stromů nám však vyšší variance nevadí. Implicitní hodnotu tohoto parametru Breiman zvolil 1 pro klasifikaci [10]. V případě velkého počtu slabých, vzájemně korelovaných, prediktorů by ovšem mohlo být vhodnější vytvářet menší stromy, tedy nastavit tento parametr o neco výše.

Celkové výsledky náhodných lesů nejvíce ovliňuje počet prediktorů m vybraných při každém dělení. Může být překvapující, že i při nízkém počtu vybraných prediktorů se OOB

odhad chyby příliš neliší od případu s vyšším počtem, avšak díky velkému počtu stromů má každý prediktor dostatečnou příležitost zúčastnit se dělení [10]. Tento parametr by měl být zvolen s ohledem na celkový počet prediktorů. V [6] je doporučenou hodnotou druhá odmocnina z celkového počtu prediktorů.

5.6.5 Náhodný les s váženými stromy

Náhodné lesy dosahují vynikajících výsledků, avšak při vytváření lesa může dojít k vytvoření velkého množství slabých stromů. Zvláště v případě, že trénovací data obsahují spoustu příznaků, z nichž je velká část rušivá - velmi malé významnosti. Tyto rušivé stromy mohou ovlivnit celkovou přesnost klasifikace.

Kompenzací této skutečnosti může být zavedení vah jednotlivým stromům. Tyto váhy jsou určeny na základě přesnosti klasifikace jednotlivých stromů. Rušivým (a tedy nepřesným) stromům bude přiřazena menší váha a dojde tak ke snížení jejich vlivu na celkovou klasifikaci. Přesnější stromy mají naopak šanci ovlivnit klasifikaci více. Otázkou zůstává, jak tuto přesnost vyhodnotit. Pro každý strom se ihned nabízí možnost získat ji vyhodnocením úspěšnosti klasifikace na datech, která nebyla použita při jeho vytváření (již dříve popsaná out-of-bag data).

Experimenty porovnávající tuto modifikaci s neváženým Náhodným lesem a C4.5 algoritmem provedené zde [17] ukazují, že ve většině případů zavedení vah skutečně poskytuje lepší výsledky, než zbylé dvě metody. Lze si všimnout, že úspěšnost klasifikace je sice lepší, ale nijak zásadně.

5.7 Shrnutí kapitoly

V této kapitole bylo definováno množství pojmů, se kterými bude pracováno i nadále. Základ tvoří pojem anomálie, jenž by definován hned zkraje kapitoly.

Dále jsme formálně definovali pojem klasifikace a představili si několik metod použitelných pro tuto činnost, především klasifikační stromy, které jsou velmi oblíbené.

Byl zde také vysvětlen princip a vlastnosti algoritmu Náhodný les, jenž je jádrem klasifikátoru popsaného dále a byla zde uvažována i modifikace – Náhodný les s váženými stromy, která se snaží omezit vliv slabých stromů při závěrečném hlasování o výsledku.

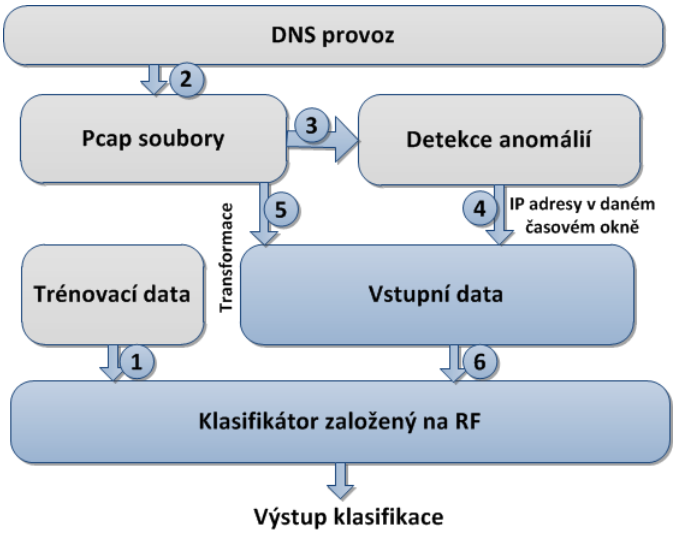
Kapitola 6

Návrh systému pro identifikaci anomálií v DNS provozu

Cílem navrhovaného systému bude schopnost klasifikovat anomálie uvedené v předchozí kapitole, případně i další. Možnosti detekce závisí na trénovací množině dat, ze které se klasifikátor naučí rozpoznávat jednotlivé anomálie.

6.1 Návrh systému

Na obrázku 6.1 je celkový pohled na navrhovaný systém. Světle zakreslené části jsou implementovány nebo dodány ze strany CZ.NIC, zbylé byly implementovány mnou. Následuje



Obrázek 6.1: Návrh systému z pohledu vyšší úrovně. Implementovanou částí je samotný klasifikátor.

detailnější popis jednotlivých částí (čísla korespondují s obrázkem):

1. Základem jsou trénovací data, na základě kterých bude klasifikátor naučen. I tato data je potřeba převést do vhodného formátu.
2. Veškerý provoz na českých serverech DNS je monitorován a archivován.

3. Již existující komponentou v CZ.NIC je detektor anomálií, na který by měl navazovat právě klasifikátor, jenž je podstatou této práce.

4. Výstupem analyzátoru jsou „podezřelé“ IP adresy a časové okno, ve kterém se nachází.

5. Na základě výstupu detektoru je identifikován potřebný časový interval a tedy i relevantní datové soubory se zachyceným provozem. Z těchto surových uložených dat je nejprve potřeba získat potřebné informace (některé metriky uvedené níže) a přizpůsobit je vstupnímu formátu pro implementovaný klasifikátor. Tento formát je popsán v další kapitole stejně jako detaily tvorby některých metrik.

6. Vhodný formát dat pak může již být vstupem klasifikátoru, který provede klasifikaci dle zadaných parametrů. Uživatel bude mít možnost zvolit parametry ovlivňující růst náhodného lesa. Především se jedná o počet stromů n a počet prediktorů m uvažovaných při štěpení uzlu.

6.2 Sledované metriky provozu za účelem klasifikace

Pro implementaci jsou uvažovány následující sledované vlastnosti DNS provozu:

- Velikost dotazu – průměrná délka doménového jména je 9 - 10 znaků, což také ovlivňuje délku samotného dotazu.
- Hodnota TTL – typická hodnota nebývá menší než 5 minut.
- Země, odkud dotaz přichází – některé země jsou jsou známy jako zdroje útoků.
- Čas dotazu – Intenzita provozu ve večerních hodinách a víkendech typicky nedosahuje takových intenzit jako jindy.
- Různorodost tazatelů – mapování IP adresy tazatele na kód země, číslo autonomního systému a BGP. Důvodem k zachycení různorodosti tazatelů je snaha o odhalení domén spojených se škodlivým software, který bývá dotazován právě z velice širokého spektra tazatelů a to mnohem častěji, než legitimní domény. Příčinou této rozdílnosti je fakt, že dotazy na tyto domény pocházejí z velkého množství různých sítí poskytovatelů připojení, jejichž koncoví uživatelé typicky nepoužívají příliš velké zabezpečení nebo jim chybí patřičné znalosti.
- Absolutní počet různých IP adres, zemí, autonomních systémů a BGP prefixů.
- Počet dotazů během daného časového okna na dotazovanou IP adresu – především k detekci (D)DoS útoku.
- Relativní nárůst počtu dotazů oproti minulému období – především k detekci (D)DoS útoku.
- Procentuální zastoupení typů dotazů z dané adresy.
- Pokles relativní četnosti dotazů na záznamy typu A.
- Rozložení nejvýznamějších chybových kódů (RCODE)[20] pro odpovědi danému tazateli, především tyto:

  - 0 (No error) - v případě bezchybného zpracování.
  - 1 (Format error) - chybně formulovaný dotaz.
  - 3 (Name error) - oznamuje neexistenci dotazovaného jména, lze využít například k detekci (D)DoS útoku.
  - 5 (Refused) - odmítnutí operace díky nastaveným politikám, například někdo se snaží o přenos zónového souboru.
  - Ostatní - další chybové kódy.

Mezi další možné metriky by mohly patřit následující:

- Podobnost QNAME lidskému jazyku – pro detekci datového tunelu skrz DNS provoz.
- Reputace přeložené IP adresy – pro detekci adres/domén spojených se škodlivými programy. Lze použít lokální databázi nebo dotazování některého externího systému za běhu.
- Příznak, zda se dotazovaná jména tvoří monotónní posloupnost (v daném časovém okně) – za účelem jednoduché detekce pokusu o slovníkový výčet zóny.
- Entropie
- ...

6.3 Trénovací množina

Důležitou částí jsou také trénovací data, na základě kterých bude klasifikátor „naučen“. Tato množina je vytvářena ruční klasifikací, což je poměrně pomalý a zdlouhavý proces, a to je také jeden z důvodů vzniku této práce. Fomát dat je popsán v následující kapitole v části 7.3.1. Ke klasifikaci budeme uvažovat následující třídy, neboť prozatím pouze tyto je detektor schopen v provozu odhalit:

- 0 - Neznámé/neklasifikované dotazy
- 1 - Podezřelé dotazy
- 2 - Resolver
- 3 - Špatně nakonfigurovaný resolver - typicky se vyznačuje opakovanými dotazy na stejný záznam ve velice krátkém čase
- 4 - Vyhledávací robot (Web crawler) - snaží se navštívit všechny WWW stránky
- 5 - Slepý výčet zóny - například pomocí slovníku
- 6 - Informovaný výčet - tazatel je obeznámen s obsahem zóny, pouze malé procento odpovědí je typu NXDOMAIN
- 7 - Záznamy s nízkým TTL
- 8 - Možný spam - dotazy na záznamy typu MX

Detektor je schopen odhalit provoz vykazující náhlou odlišnost či regulárnost [12]. Nemusí se přitom jednat o velké množství dat (z celkového provozu). Vidíme, že ne všechny třídy nutně splývají s pojmem anomálie, některé můžeme označit spíše za ukazatele anomálií.

6.4 Shrnutí kapitoly

V této části byl nastíněn základní návrh systému a jeho návaznost na již existující komponenty. Vstupem klasifikátoru jsou data, která detektor anomálií vybral jako odchylky. Třídy, které klasifikátor rozlišuje jsou dány trénovacími daty, která jsou dodána ze strany CZ.NIC.

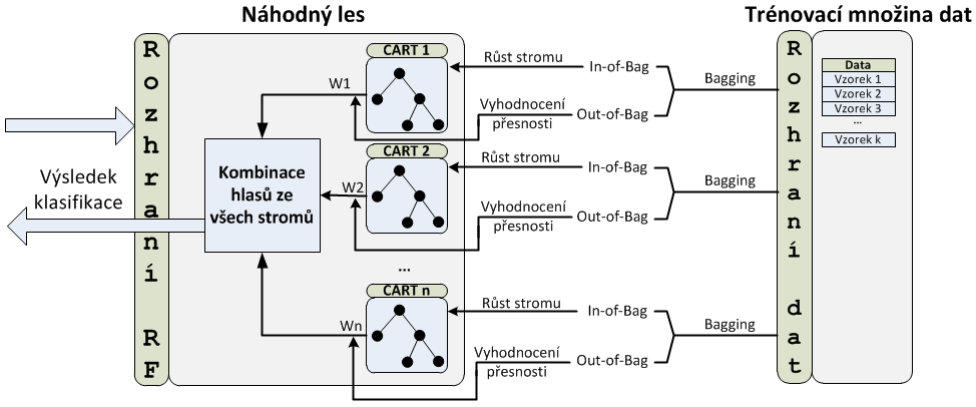
Především však byl navrhnut seznam sledovaných vlastností, které by bylo možné využít při klasifikaci jednotlivých anomálií. Některé z uvedených metrik nebudou použity při implementaci klasifikátoru, například reputace překládaných adres. Následující část bude věnována popisu implementace, formátu vstupních i výstupních dat a blíže popíšeme způsob získávání metrik.

Kapitola 7

Popis implementace

Celá práce je napsána v jazyce C++ a vychází z původní implementace Random Forests, která je napsána v jazyce Fortran[8]. Základní struktura je zobrazena na obrázku 7.1. Jedná se o konzolovou aplikaci. Ta je rozdělena do několika částí. Tyto komponenty mezi sebou úzce spolupracují, jmenovitě se jedná o následující:

- Třída `RandomForest` tvořící Náhodný les
- Třída `Tree` tvořící jeden klasifikační strom
- Třída `DataSet` poskytující rozhraní k trénovacím/testovacím datům



Obrázek 7.1: Schéma implementovaných součástí a komunikace mezi nimi

7.1 Parametry programu

Aplikace přijímá následující parametry:

- `-t, --training-file <filename>` – Specifikuje cestu ke trénovacímu souboru.
- `-f, --load-forest <filename>` – Specifikuje cestu k souboru, ze kterého bude klasifikátor načten. Neobsahuje trénovací data, z čehož vyplývají jistá omezení.

- `-i, --input <filename>` – Specifikuje soubor obsahující vzorky určené ke klasifikaci.
- `-o, --output <filename>` – Volitelný parametr. Specifikuje výstup. Není-li specifikován, je použit standardní výstup.
- `-n, --number-of-trees <number>` – Specifikuje počet stromů použit při vytváření klasifikátoru.
- `-m, --mtry <number>` – Specifikuje počet argumentů uvažovaných při štěpení uzlu.
- `-x, --save-forest <filename>` – Volitelný parametr. Specifikuje cestu k souboru, do kterého bude klasifikátor uložen.
- `-u, --unweighted` – Volitelný parametr. Příznak pro použití nevážených stromů.
- `-w, --weighted` – Volitelný parametr. Příznak pro použití vážených stromů.
- `-e, --decision < soft | hard >` – Volitelný parametr. Specifikuje jeden ze dvou druhů rozhodování - měkce nebo tvrdě.
- `-r, --header < true | false >` – Volitelný parametr. Specifikuje zda datové souboru obsahují hlavičku se jmény proměnných.
- `-d, --delimiter <string>` – Volitelný parametr. Specifikuje oddělovací znak pro datové souboru.
- `-l, --logfile <filename>` – Volitelný parametr. Specifikuje cestu k logovacímu souboru.
- `-L, --loglevel < 0 | 1 | 2 >` – Volitelný parametr. Specifikuje úroveň logování (0 = pouze kritické, 1 = více informací, 2 = všechno).
- `-c, --importance` – Příznak pro tisk významnosti jednotlivých proměnných.
- `-p, --proximity` – Příznak pro tisk matice blízkosti.
- `-s, --print-stats` – Příznak pro tisk dodatečných informací o vytvořeném klasifikátoru.
- `-h, --help` – Vytiskne popis jednotlivých parametrů.

Konkrétní příklad použití je možno nalézt v Makefile, který se nachází na přiloženém médiu.

7.2 Získání metrik

Součástí této práce je také nalezení způsobu získání potřebných atributů, podle kterých bude klasifikace prováděna. Seznam sledovaných metrik byl uveden v předchozí kapitole. Za tímto účelem byl vytvořen program v jazyce C++, který zpracovává uložené *pcap* soubory. Tyto obsahují zachycené DNS pakety - dotazy a odpovědi. Příklad zpracovávaných dat po zobrazení programem Wireshark:

Č.	Čas	Zdroj	Cíl	Protocol	Informace
1	0.000597	192.168.0.1	194.0.12.1	DNS	Standard query A merlin.fit.vutbr.cz
2	0.000694	194.0.12.1	192.168.0.1	DNS	Standard query response
⋮	⋮	⋮	⋮	⋮	⋮

Jedná se pouze o ilustrativní ukázku. Detailní popis obsahu DNS paketů byl popsán v části 2.2. Kromě DNS informací máme tedy také informace ze síťové vrstvy (zdrojová a cílová IP adresa) a transportní vrstvy (TCP/UDP, zdrojový a cílový port).

Na standardním vstupu je očekáván seznam sledovaných IP adres. Na každém řádku smí být pouze jedna adresa. Parametry programu tvoří pouze jména souborů. Vstupní soubory rozdělí na polovinu, přičemž první polovinu chápe jako minulost, vůči které budou počítány rozdílové hodnoty (např. pokles či nárůst počtu dotazů). Druhá polovina představuje aktuální (sledovaný) interval.

Tyto soubory jsou poté zpracovány. Pro parsování DNS paketů jsem využil knihovnu *ldns*. Některé metriky, jako je dotazované jméno, IP adresa či zdrojový port, jsou získány přímo. Jiné musí být odvozeny. Získání mapování IP adresy tazatele na kód země, BGP prefix a ASN jsem získal pomocí DNS dotazu na mapovací službu[1] a zpracováním získané odpovědi. Příklad pro zjištění těchto údajů pro IP adresu 216.90.108.31 je nutné provést DNS dotaz typu TXT na jméno `31.108.90.216.origin.asn.cymru.com`:

```
$ dig +short 31.108.90.216.origin.asn.cymru.com TXT
```

A dostane se nám následující odpovědi:

```
"23028 | 216.90.108.0/24 | US | arin | 1998-09-25"
```

Upravená verze programu je použita pro transformaci trénovací množiny do vhodného formátu. Vstupem tohoto programu je soubor, jenž musí nést jméno serveru, ze kterého data pochází. Na základě tohoto vstupu jsou dohledány příslušné soubory se zachyceným provozem dle uvedených časů. Obsahem souboru tvoří údaj o čase, kdy byla odchylka zjištěna, IP adresa, která toto způsobila a typ odchylky, o kterou se jedná. Příklad několika řádků takovéhoto souboru může vypadat následovně:

```
<From: Sun Apr 01 02:00:00 2012> <To: Sun Apr 01 02:09:59 2012> <IP:
208.115.111.73> <Anomaly_type: 2>
<From: Sun Apr 01 02:10:00 2012> <To: Sun Apr 01 02:19:59 2012> <IP:
213.151.82.226> <Anomaly_type: 2>
<From: Sun Apr 01 02:10:00 2012> <To: Sun Apr 01 02:19:59 2012> <IP:
217.31.207.1> <Anomaly_type: 6>
```

7.3 Detaily implementace třídy `DataSet`

Tato třída tvoří rozhraní k trénovacím datům, která jsou načtena ze souboru, jehož formát je specifikován dále. Po načtení dat ze souboru jsou tato data seřazena bez přesunu a to pro všechny atributy. Poskytované operace jsou především tyto:

- zpřístupnění vektoru indexů seřazených vzhledem k hodnotě specifikovaného atributu

[1] Blíže popsáno zde http://www.team-cymru.org/Services/ip-to-asn.html

- získání třídy, do které specifikovaný záznam patří
- zpřístupnění hodnoty požadovaného atributu pro daný záznam
- zjištění počtu záznamů
- podpůrné funkce pro výpočet významnosti atributů

7.3.1 Formát dat

Vstup tvoří vždy textový soubor. Položky na řádku jsou odděleny specifikovaným znakem, implicitně se předpokládá středník jakožto oddělovací znak. První řádek obsahuje metadata pro určení datových typů jednotlivých sloupců. Podporované typy jsou:

- a – IP adresa (verze 4 nebo 6)
- u – celočíselný datový typ bez znaménka
- i – celočíselný datový typ se znaménkem
- f – číslo v plovoucí řádové čárce
- s – řetězec identifikující jméno třídy

Dále může volitelně obsahovat hlavičku, která pro přehlednost přiřadí atributům jména. Struktura hlavičky je tedy:

$$\texttt{[TŘÍDA,]JMÉNO}_1\texttt{, JMÉNO}_2\texttt{, ..., JMÉNO}_M$$

Třída je přítomna pouze v trénovacích datech. Na pojmenování sloupce s třídou nezáleží.

Trénovací data

Každý řádek odpovídá jednomu záznamu. První položkou je identifikátor třídy. Dále již následují hodnoty atributů. Příklad se čtyřmi proměnnými i s hlavičkou:

s,	a,	i,	i,	i
LABEL,	SRCIP,	SRCPORT,	TTL,	TIME
0,	192.168.10.1,	54133,	300,	1331063441
1,	192.168.10.1,	54133,	1200,	1331063444
⋮	⋮	⋮	⋮	⋮

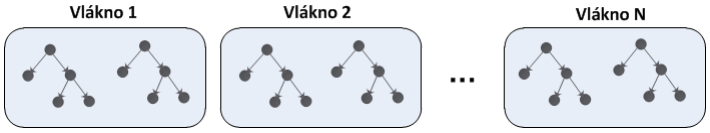
Opět se jedná pouze o ilustrativní ukázku. Skutečná ukázka trénovacích dat obsahuje mnohem více položek a je možné ji najít na přiloženém médiu.

Vstupní data ke klasifikaci

Formát vstupu klasifikovaných dat odpovídá formátu trénovacích dat uvedenému výše. Vstup se liší pouze v tom, že neobsahuje identifikaci třídy, do které patří, neboť typicky není známa.

7.4 Detaily implementace třídy `Tree`

Růst stromu může být časove velice náročná operace. Jednotlivé stromy jsou navíc nezávislé, a proto je možné jejich růst paralelizovat. Každé vlákno je zodpovědné za růst určité skupiny stromů, viz obrázek 7.2. Počet vytvořených vláken je dán systémovým voláním `sysconfig(_SC_NPROCESSORS_ONLN)`.



Obrázek 7.2: Stromy mohou růst paralelně, neb na sobě nejsou závislé

Při vytváření si každý strom vytvoří lokální kopii matice indexů. Každý řádek obsahuje seřazené indexy podle hodnot jednoho z atributů.

Díky této matici je jednodušší hledání hodnoty, která rozdělí trénovací množinu dat v uzlu, který je právě štěpen. Není tak nutné prohledávat všechny možné kombinace rozdělení s exponenciální složitostí, ale pouze lineárně projít tuto seřazenou posloupnost danou jedním řádkem v matici.

Během vytváření stromu je však nutné v této matici provádět změny (proto každý strom potřebuje vlastní). Jakmile strom svůj růst ukončí, není tato matice již potřebná a je uvolněna.

Většina metod, které nabízí třída `Tree` nabízí také třída `RandomForest`, neboť agreguje výsledky získané z jednotlivých stromů.

7.5 Detaily implementace třídy `RandomForest`

Vzhledem k tomu, že konkrétní podoba stromů je částečně dílem náhody, je žádoucí mít možnost zapamatovat si les vykazující dobré výsledky. Z tohoto důvodu je možné les uložit do textového souboru pomoci metody `save` a později jej opět znovu načíst metodou `load`. Načtený les navíc není nutné znovu trénovat, což šetří čas.

Je umožněno provádět klasifikaci za pomoci vážených stromů, jejichž váha je dána přesností klasifikace na OOB datech, ale i neválených s uniformní jednotkovou vahou. Další implementované vlastnosti jsou:

- Matice blízkosti – z důvodu lepší efektivity algoritmus počítá s úspornější trojúhelníkovou maticí.
- Významnost proměnných – měření této vlastnosti je provedeno na základě rozdílu klasifikační přesnosti před a po permutaci hodnot dané proměnné. Podrobnější postup byl uveden v části 5.6.2.
- Měkké a tvrdé rozhodnutí – v případě tvrdého rozhodnutí je přímo vrácena třída jakožto výsledek klasifikace, při měkkém rozhodnutí je vrácen vektor hodnot, přičemž každé třídě přísluší jedna z hodnot. Tyto hodnoty odpovídají tomu, jak moc se klasifikátor přiklání právě k dané třídě.

7.6 Výstup

Výstup je tisknut do specifikovaného souboru. Pokud není soubor zadán, je použito standardního výstupu. Různými přepínači je možné ovlivnit, co dalšího se kromě samotných výsledků klasifikace objeví na výstupu.

Jednotlivé sekce výstupu jsou odděleny záhlavím, které poměrně jasně napovídá, co bude následovat. Mohou to být:

- Matice blízkosti – horní trojúhelníková matice je uvedena v sekci ⟨PROXIMITY⟩.
- Výstup důležitosti proměnných – je v sekci ⟨VARIABLE IMPORTANCE⟩ a obsahuje seznam proměnných seřazených sestupně od nejvýznamnější.
- Výsledky klasifikace – tato sekce je uvedena jako ⟨CLASSIFICATION RESULTS⟩. Příklad výstupu pro měkké rozhodnutí se třemi třídami:

DoS - 100%	SCAN - 0%	SPAM - 0%
DoS - 0%	SCAN - 0%	SPAM - 100%
DoS - 0%	SCAN - 100%	SPAM - 0%
DoS - 0%	SCAN - 0%	SPAM - 100%
⋮	⋮	⋮

- Další statistické údaje – jsou v sekci ⟨STATS⟩.

7.7 Shrnutí kapitoly

Tato kapitola obsahovala základní nástin implementace systému a popis některých implementovaných metod. Dále zde byly popsány parametry, které aplikace přijímá, a jejich význam. Také zde byl popsán formát vstupních a trénovacích dat a také formát výstupu. V neposlední řadě byl také přiblížen proces tvorby vstupních souborů, který se zahrnuje zpracování uložených DNS dat. V následující kapitole se podíváme na chování systému a experimenty s různým nastavením parametrů.

Kapitola 8

Ověření a otestování chování systému

V této kapitole budou popsány výsledky experimentů s programem, především časová a paměťová složitost a vliv volby parametrů na celkovou přesnost klasifikace. U všech grafů byly k jejich tvorbě využity průměrné hodnoty skutečně naměřených časů, případně nejvyšší zaznamenané paměťové požadavky.

8.1 Časová složitost

Vytvoření klasifikátoru a jeho natrénování

Zde se zaměříme na dobu nutnou k vytvoření klasifikátoru a jeho natrénování. Nejprve se pokusíme složitost odvodit teoreticky. Sekvenční část tvoří počáteční inicializace:

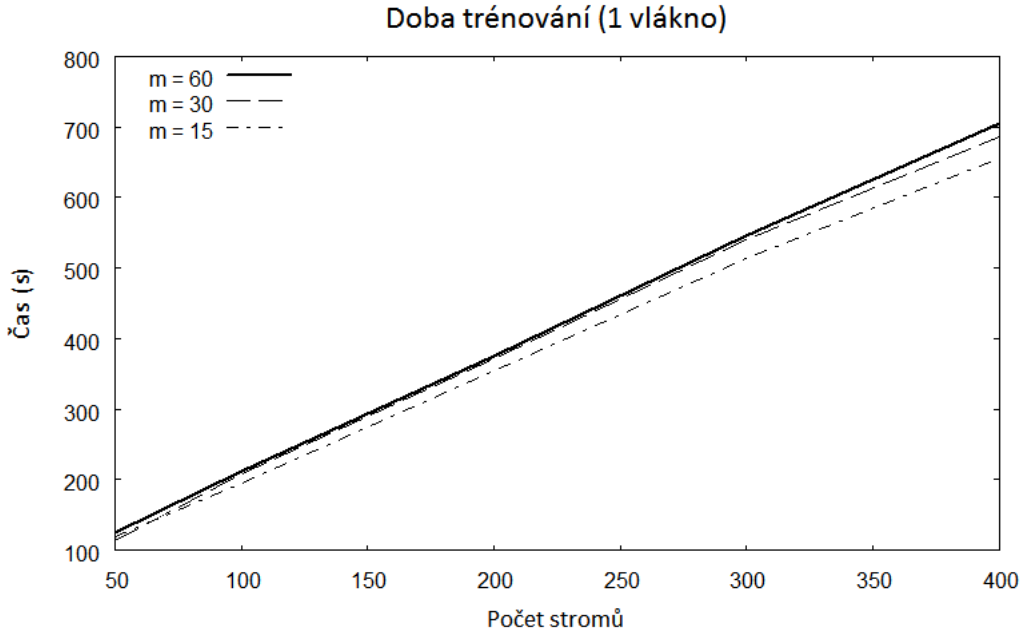
- Vytvoření a zpracování datové sady T záleží na její velikosti t – $O(t)$.
- Vytvoření n stromů a jejich parametrů – $O(n)$.
- Jistou režii lze připsat vytvoření parametrů pro N vláken, avšak typicky platí, že $N \ll n$, proto ji nebudeme ve výsledku uvažovat.

Následný růst již probíhá paralelně, pokud je to možné:

- Každý strom si zkopíruje matici seřazených indexů o velikosti $M \times t$ – $O(Mt)$.
- A dojde k vytvoření jeho uzlů. V nejhorším případě by se mohlo jednat o plný binární strom s t listy a tedy $2t - 1$ uzly. Každý z nich musí také vyzkoušet m zvolených atributů – $O(mt)$.
- Celkově dojde k vytvoření n stromů.

Celkově tedy můžeme složitost vyjádřit jako $O(t + n) + O(\frac{(M+m)t}{N})$.

Pro praktický experiment uvažujme datovou sadu obsahující celkem $t = 13375$ vzorků, přičemž každý z nich má $M = 927$ převážně číselných atributů. Ke klasifikaci budeme uvažovat pouze dvě třídy.



Obrázek 8.1: Závislost trénovací doby na počtu stromů n pro 1 vlákno a různé hodnoty m.

Průměrný výsledek pěti běhů[1] pro tuto datovou sadu je zachycen v grafech na obrázcích 8.1 a 8.2. Každá z křivek je pro jiný parametr m. Doporučená hodnota tohoto parametru se pohybuje okolo odmocniny z celkového počtu atributů, a proto jsou vybrány hodnoty 15, 30 a 60.

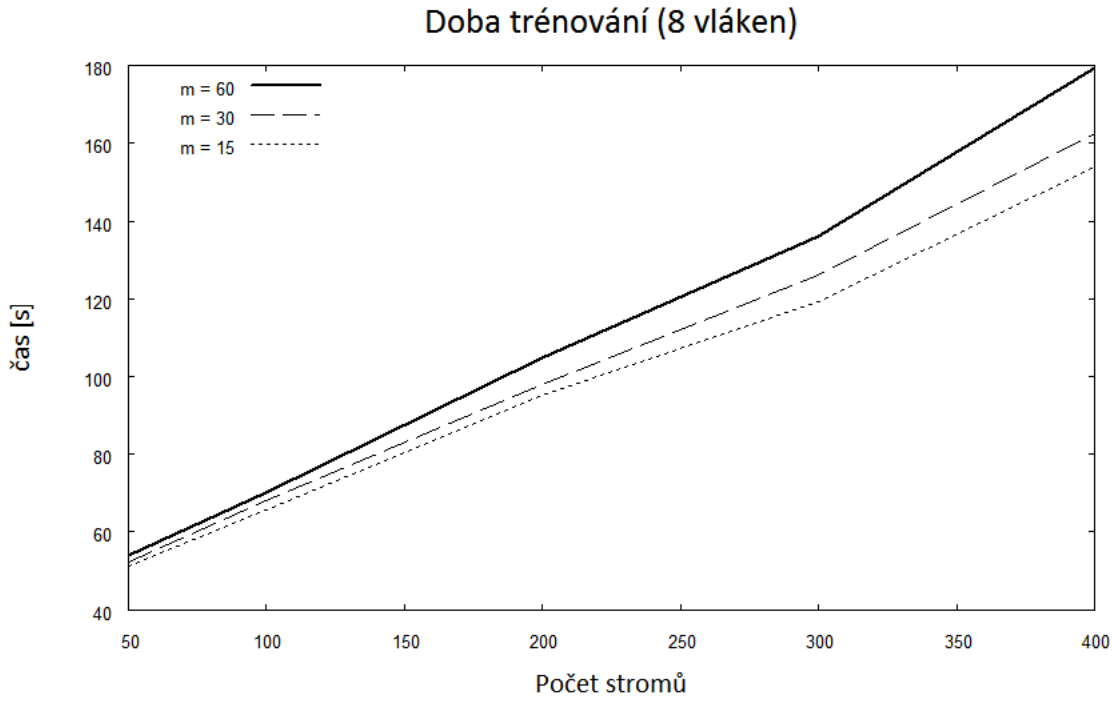
Vidíme, že čas potřebný k vytvoření roste celkem lineárně s počtem stromů. Stejně tak je patrná závislost této doby na parametru m. Pro více vláken je situace zobrazena v grafu na obrázku 8.2.

Ze získaných časů je patrné, že dochází ke zrychlení vlivem paralelismu. Zjištěné hodnoty zrychlení pro konkrétní parametry jsou v tabulce 8.1. Nejvyšší dosažené zrychlení samozřejmě není osminásobné, především díky tomu, že jistá část úlohy je sekvenční a částečně také díky režii nutné k vytvoření vláken.

n	m = 15	m = 30	m = 60
50	2,28	2,29	2,30
100	2,98	3,04	3,09
200	3,73	3,80	3,58
300	4,31	4,29	4,01
400	4,27	4,22	3,93

Tabulka 8.1: Hodnoty zrychlení při použití osmi vláken pro různé parametry.

[1] Měřeno na stroji `merlin.fit.vutbr.cz` pomocí programu `time`.



Obrázek 8.2: Závislost trénovací doby na počtu stromů n pro různé hodnoty m.

Klasifikace

Nyní se zaměříme na časovou náročnost samotné klasifikace, což je také zajímavá oblast vhodná k prozkoumání. Opět zkusme nejprve odvodit časovou složitost teoreticky a poté popíšeme výsledky praktických experimentů.

Pro klasifikaci je nutné projít všech n stromů, přičemž cesta od kořene binárního stromu až k jeho listu má v nejhorším případě složitost $O(\log t)$. Časová složitost pro vstup obsahující t vzorků je tedy $O(nt \log t)$.

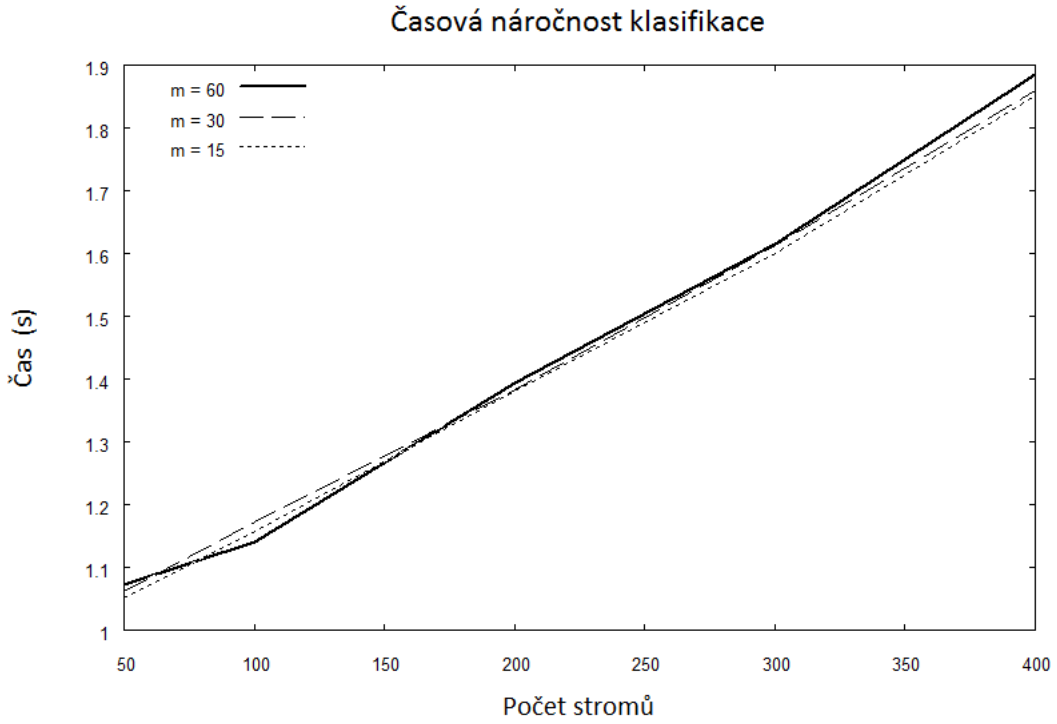
Pro praktický experiment v této části uvažujme testovací sadu s počtem vzorků $t = 500$ a $M = 927$ atributy. Doby klasifikace[2] této testovací sady jsou zaneseny v grafu na obrázku 8.3.

Z grafu vyplývá, že čas potřebný ke klasifikaci roste také lineárně s počtem stromů. Rozdíly mezi hodnotami pro různá m jsou zanedbatelně malé. Naměřený čas se lišil v závislosti na tom, na kterém stroji běžel. Na mém lokálním stroji se hodnoty lišily od uvedených až šestinásobně. Dále můžeme uvést dobu klasifikace vztaženou na jeden záznam:

n	50	100	200	300	400
čas [ms]	2,122	2,344	2,766	3,228	3,722

Tabulka 8.2: Průměrná doba načtení a klasifikace jednoho záznamu.

[2]Měřeno na šestijádrovém AMD Opteron 4180 @ 2600 MHz pomocí programu `time`.



Obrázek 8.3: Závislost doby běhu klasifikace na počtu stromů n pro různá m.

8.2 Prostorová složitost

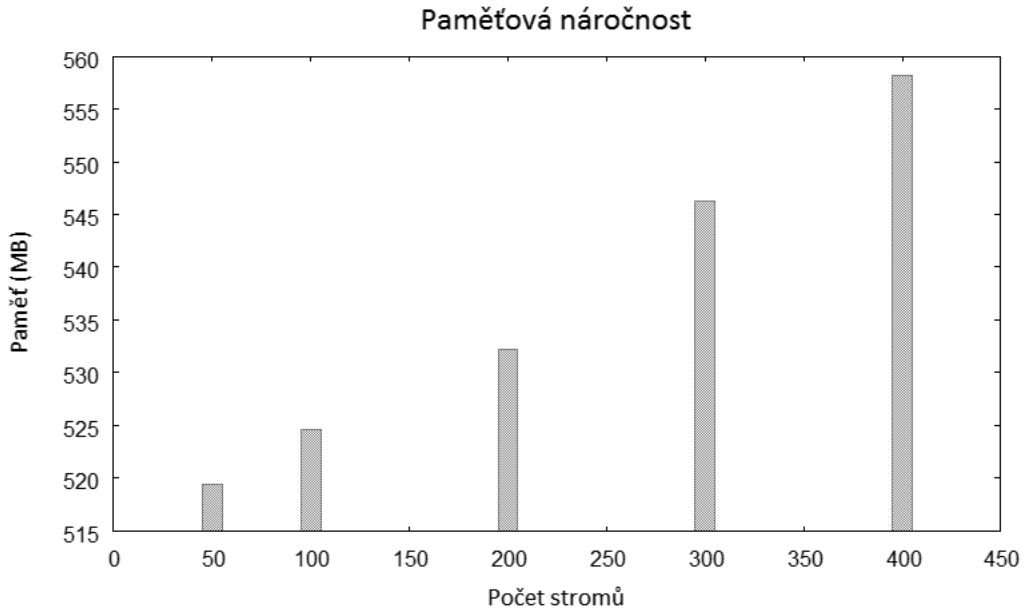
Opět se nejprve zaměříme na paměťové požadavky nutné při trénování klasifikátoru. V této fázi se do celkových paměťových nároků promítnou následující parametry:

- Velikost t trénovací množiny dat, která existuje pouze v jediné kopii. S trénovací množinou souvisí také počet atributů každé datové položky M.
- Počet stromů n, jelikož každý strom si vytvoří vlastní matici indexů pro seřazení dat bez přesunu.
- Vlastní uzly stromů. Pro nejhorší případ lze uvažovat plné binární stromy s $2t + 1$ uzly. Průměrně lze ovšem očekávat méně uzlů.
- Naopak množství paměti není závislé na parametru m.

Z těchto údajů vidíme, že celková paměť je závislá jak na dodaných datech, tak na vytvářeném klasifikátoru. Pro asymptotickou prostorovou složitost tedy platí $O(t) + O(nMt)$.

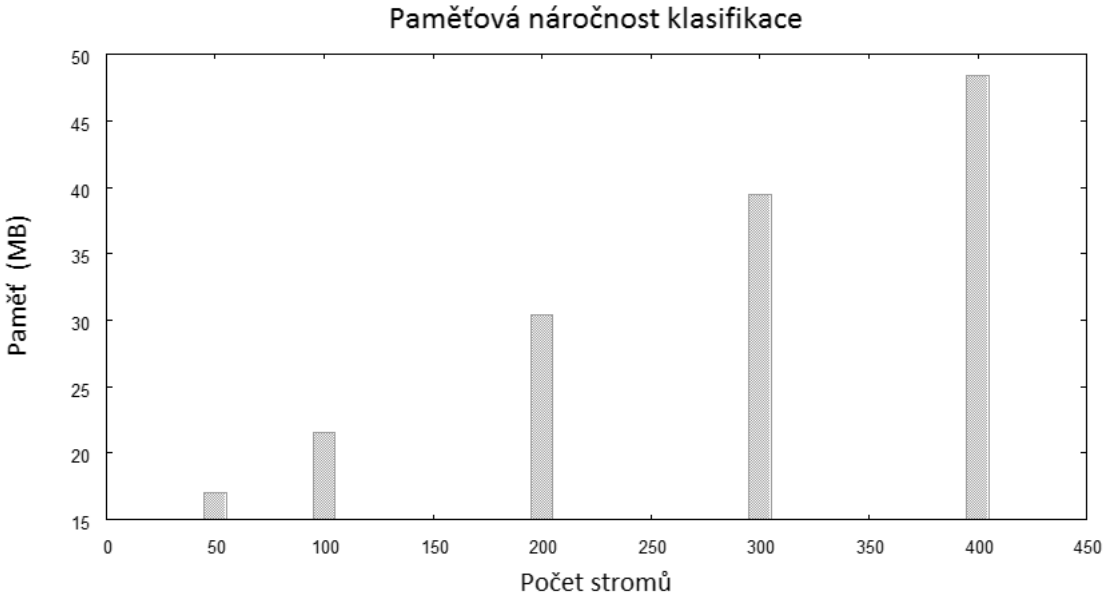
Uvažujme stejnou datovou sadu jako v případě měření paměťových nároků ($t = 13375$ vzorků, $M = 927$ atributů, dvě třídy). Pro tuto datovou sadu jsou paměťové požadavky[3] zaznamenány v grafu na obrázku 8.4. Vyneseny jsou pouze nejvyšší celkové paměťové požadavky naměřeny během trénovací doby.

[3]Měřeno pomocí programu *valgrind*, přesněji pomocí jeho nástroje *massif*.



Obrázek 8.4: Paměťové nároky při trénování v závislosti na počtu stromů n.

Paměťové nároky nutné pro provoz samotného klasifikátoru zmiňované datové sady jsou zachyceny v grafu na obrázku 8.5. Jsou zřejmě menší, než při trénování, jelikož již není potřeba v paměti držet trénovací množinu dat a také pomocnou matici indexů.



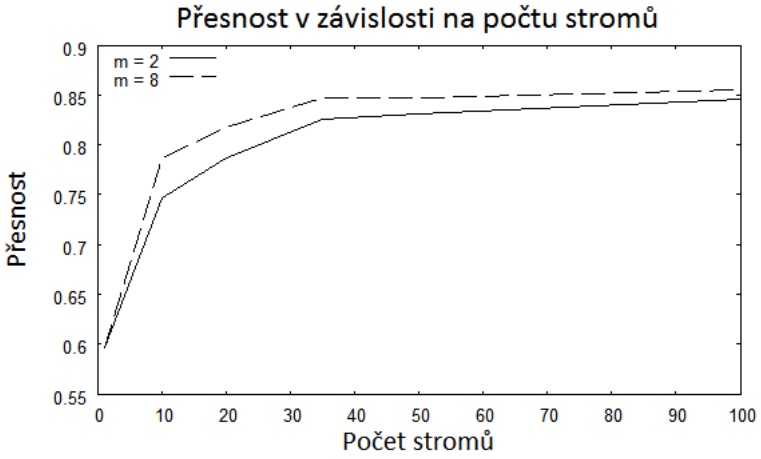
Obrázek 8.5: Paměťové nároky při klasifikaci v závislosti na počtu stromů n.

8.3 Přesnost klasifikace

Nyní přejdeme k patrně nejdůležitějšímu měření celé práce, čímž je vyhodnocení přesnosti systému. Trénovací data mi byla poskytnuta až v pozdní fázi, a proto nebylo mnoho času provést příliš rozsáhlé testy.

K dispozici mi byla dána trénovací data ze sedmi dnů. Následující přesnost jsem měřil po natrénování z šestidenního provozu. Sedmého dne jsem využil pro testování. Důvod pro použití více dnů je ten, že ne všechny anomálie se projevily během jednoho dne, tudíž by klasifikátor o zbývajících anomáliích ani nevěděl.

Počet stromů n nutný pro udržení dobré přesnosti stoupá s počtem prediktorů. Určení vhodného počtu je možné provést porovnáním výsledků lesa a jeho části. Pokud část stromů klasifikuje stejně dobře jako celý les, je n dostatečné. Na obrázku 8.6 je zachycena přesnost klasifikace na zmíněných datech v závislosti na počtu stromů pro různý počet volených atributů. Vidíme, že již od zhruba padesáti stromů se výsledek příliš nemění.



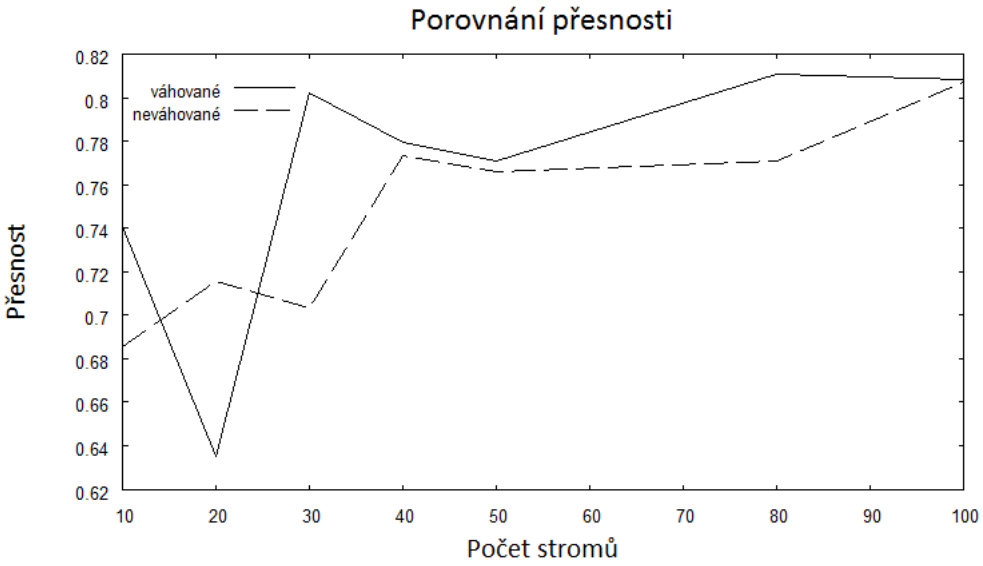
Obrázek 8.6: Přesnost klasifikace v závislosti na počtu stromů n pro různá m.

Můžeme také porovnat přesnost varianty s váhovanými stromy oproti „normální“ verzi, jak je zachyceno v grafu na obrázku 8.7. Z uvedeného grafu vidíme, že váhovaná varianta stromů vykazuje malinko lepší výsledky, především pro více stromů. Pro menší počet stomů typicky platilo opačné tvrzení.

Pro experiment jsem vyzkoušel naučit klasifikátor na extrémně malé trénovací sadě (dvanáct náhodně vybraných záznamů pro každou z anomálií). Následně jsem jej nechal klasifikovat většinu z celého týdenního provozu. Výsledek je zachycen na obrázku 8.7. Přesnost podle počtu paketů se pohybovala kolem 53%, což je způsobeno tím, že výstupem analyzátoru je pouze adresa, nikoliv konkrétní paket. Z pohledu analyzátoru je tedy cílem klasifikovat danou IP adresu, což je také podstatou všech zde uvedených měření přesnosti.

Zjištěné váhy (OOB přesnosti) jednotlivých stromů se pohybovaly zhruba od 50% až po 99%, pouze vyjímečně se objevovala menší přesnost. Jako nejdůležitější proměnné se při klasifikaci pro velkou trénovací sadu jevily v sestupném pořadí tyto:

- IP adresa,
- relativní nárůst v počtu dotazů,



Obrázek 8.7: Srovnání přesnosti obou variant náhodného lesa vytvořených na velmi malé trénovací sadě.

- absolutní počet dotazů,
- relativní nárůst v počtu dotazů typu A,
- procentuální zastoupení chybových kódů v odpovědích.

Pro menší sadu:

- BGP prefix,
- relativní nárůst v počtu dotazů typu A,
- procentuální zastoupení chybových kódů v odpovědích.

V prvním případě lze tvrdit, že díky kladení důrazu na IP adresu byla datová sada příliš rozsáhlá a je možné, že došlo k přílišné specializaci. Ovšem celkově se význam jednotlivých proměnných dosti měnil, proto je velice obtížné vyvozovat z něj nějaké hlubší závěry a je uveden pouze pro úplnost.

8.4 Shrnutí kapitoly

V této kapitole byly popsány provedené experimenty s výslednou aplikací, především byla zkoumána paměťová a časová složitost a také přesnost klasifikace.

Paměťové nároky nejsou na dnešní dobu nikterak přehnané, obecnou použitelnost může komplikovat nutnost držet celou datovou sadu v paměti. Při práci s DNS daty však žádné problémy nenastaly.

I přes to, že samotná klasifikace je poměrně rychlá, celý proces je zpomalen nutností vytvoření vhodného formátu vstupních dat a také faktem, že zkomprimovaná data se v úložišti objeví až po jisté době. Velké množství uložených dotazů procházení těchto dat také dále komplikuje. Výsledná podoba tedy nedovoluje online detekci a klasifikaci.

Přesnost metody se pohybovala v okolí 80% a je závislá na dobře zvolené trénovací množině dat.

Při použití s vhodnými parametry je metoda stabilní. Tímto chápeme především dostatečný počet stromů. Je jasné, že kdybychom měli pouze jeden strom, bude velmi záviset na náhodně vybraných proměnných a výsledná klasifikace stabilní být nemusí.

Byly zde také uvedeny proměnné, které byly pro klasifikaci nejvýznamnější, avšak tento seznam vykazoval jistou variabilitu během testování, a proto není zcela směrodatný.

Kapitola 9

Závěr

Tato práce stojí na počátku cesty poskytnout prostředek k automatické klasifikaci anomálií zachycených na českém serveru DNS. Veškerá klasifikace se zde doposud provádí manuálně, což je náročné na čas i lidské zdroje.

Pro implementaci jsem se rozhodl použít metodu Random Forest, která mi byla navržena konzultantem a která se zdála vykazovat poměrně dobré výsledky při klasifikaci v jiných oborech. Paměťové nároky jsou dány především daty, která jsou držena v paměti. Klasifikace je rychlá a při použití s vhodnými parametry je metoda stabilní. Přesnost klasifikace se pohybovala nad hranicí 80%, což není uplně špatný výsledek. Pro ještě lepší výsledek by nejspíš bylo nutné vytvoření lepší trénovací množiny.

Jádro klasifikátoru je poměrně universální a nijak zvláště není omezen na použití pouze pro systém DNS. Lze tak klasifikovat libovolná data, jejichž vstupní formát bude vyhovovat popsaným požadavkům aplikace.

Formát vstupních dat nebyl nijak stanoven, a proto jsem si vytvořil vlastní. Během deseti minut ovšem projde serverem DNS více než půl milionu paketů, a právě proto je fáze tvorby vstupních dat časově náročná. Vhodnější by bylo, kdyby vstupní soubory byly tvořeny přímo detektorem, odpadla by tak nutnost znovu procházet data a čekat na vytvoření správného formátu.

Dnešní a jistě také budoucí snahy o paralelizaci přinesly své ovoce i zde, především ve fázi učení klasifikátoru, kdy je možné nezávisle na sobě vytvářet několik stromů současně. Tatáž snaha aplikována při samotné klasifikaci ovšem úspěch nezaznamenala. Rychlost klasifikace je relativně dobrá a zavedení více vláken přineslo pouze horší časové výsledky. To bylo způsobeno triviální implementací, kdy docházelo k opětovnému vytváření a rušení vláken. Jisté vylepšení by mohlo nastat v případě, že by k tomuto znovuvytváření nedocházelo. Toto by si vyžádalo další úpravy a experimenty, které z časových důvodů uskutečněny nebyly.

Literatura

[1] Antonakakis, M.; Perdisci, R.; Lee, W.; aj.: Detecting malware domains at the upper DNS hierarchy. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, Berkeley, CA, USA: USENIX Association, 2011, s. 27–27.

[2] Arends, R.; Austein, R.; Larson, M.; aj.: DNS Security Introduction and Requirements. RFC 4033, 2005.

[3] Arends, R.; Austein, R.; Larson, M.; aj.: Protocol Modifications for the DNS Security Extensions. RFC 4035, 2005.

[4] Arends, R.; Austein, R.; Larson, M.; aj.: Resource Records for the DNS Security Extensions. RFC 4034, 2005.

[5] Atkins, D.; Austein, R.: Threat Analysis of the Domain Name System (DNS). RFC 3833, 2004.

[6] Berk, R.: *Statistical learning from a regression perspective*. 2008.

[7] Bostrom, H.: Estimating Class Probabilities in Random Forests. *Machine Learning and Applications, Fourth International Conference on*, ročník 0, 2007: s. 211–216.

[8] Breiman, L.: Random Forests. Technická zpráva, University of California, Berkeley, 2001.

[9] Castro, S.; Zhang, M.; John, W.; aj.: Understanding and preparing for DNS evolution. In *Traffic Monitoring and Analysis Workshop (TMA)*, Zurich, Switzerland: TMA 2010, 2010, s. 1–6.

[10] Gábik, J.: *Kreditný skóring pomocou náhodných lesov*. Diplomová práce, Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky, 2010.

[11] van der Heide, H.; Barendregt, N.: *DNS Anomaly Detection*. Diplomová práce, University of Amsterdam, 2011.

[12] Janoušek, T.; Veselý, J.: Detecting Hidden Anomalies in DNS Communication. Technická zpráva, CZ.NIC, 2009.

[13] Klaschka, J.; Kotrč, E.: Klasifikační a regresní lesy. In *Sborník prací ROBUST*, Jednota českých matematiků a fyziků, 2004, s. 177–184, [Online].
URL `<http://statspol.cz/robust/2004_robust2004.pdf>`

[14] Konings, M.: Final Report on Fast Flux Hosting. Technická zpráva, ICANN, 2009, [Online].
URL <http://gnso.icann.org/issues/fast-flux-hosting/fast-flux-final-report-06aug09-en.pdf>

[15] Larson, M.; Barber, P.: Observed DNS Resolution Misbehavior. RFC 4697, 2006.

[16] Laurie, B.; Sisson, G.; Arends, R.; aj.: DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155, 2008.

[17] Li, H. B.; Wang, W.; Ding, H. W.; aj.: Trees Weighting Random Forest Method for Classifying High-Dimensional Noisy Data. *IEEE International Conference on E-Business Engineering*, 2010: s. 160–163.

[18] Matoušek, P.: Studijní opora předmětu ISA na FIT VUT v Brně, 2010.

[19] Mockapetris, P.: Domain names - concepts and facilities. RFC 1034, 1987.

[20] Mockapetris, P. V.: Domain names - implementation and specification. RFC 1035, 1987.

[21] Project, T. H.: Know Your Enemy: Fast-Flux Service Networks. http://www.honeynet.org/node/138, 2008, [Online].

[22] Rich, C.; Alexandru, N.-M.: An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, New York, NY, USA: ACM, 2006, ISBN 1-59593-383-2, s. 161–168.

[23] Rose, S.; Nakassis, A.: Minimizing Information Leakage in the DNS. Technická zpráva, National Institute for Standards and Technology, 2010, [Online].
URL <http://w3.antd.nist.gov/pubs/892-papers/WorkingPapers-WERB-approved/DNS-information-leakage-Rose-Nakassis.pdf>

[24] Rychlý, M.: Klasifikace a predikce. 2005, [Online].
URL <http://www.fit.vutbr.cz/~rychly/docs/classification-and-prediction/classification-and-prediction.pdf>

[25] SSAC: Advisory on Fast Flux Hosting and DNS. Technická zpráva, ICANN, 2008, [Online].
URL <http://www.icann.org/en/committees/security/sac025.pdf>

[26] Wessels, D.; Fomenkov, M.: Wow, That's a Lot of Packets. 2003, [Online].
URL <http://dns.measurement-factory.com/writings/wessels-pam2003-paper.pdf>

[27] Yuchi, X.; Wang, X.; Lee, X.; aj.: A new statistical approach to DNS traffic anomaly detection. In *Proceedings of the 6th international conference on Advanced data mining and applications - Volume Part II*, ADMA'10, Springer-Verlag, 2010, ISBN 3-642-17312-8, 978-3-642-17312-7, s. 302–313.

[28] Zaiane, O.: Data Classification. 1999, [Online].
URL <http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter7>

Příloha A

Obsah DVD

Popis obsahu adresářů DVD disku přiloženého k této práci:

- `data` – obsahuje transformační program, který ze vstupních souborů v pcap formátu vytvoří soubory ve formátu vhodném jako vstup pro klasifikátor, vzhledem k citlivé povaze DNS dat zde nejsou uloženy žádné záznamy reálně zachycené z provozu
- `tex` – obsahuje tuto práci v elektronické podobě a zdrojový text napsaný v LaTeXu
- `src` – obsahuje zdrojové soubory a vše nutné pro překlad

Příloha B

Výstup programu SLOCCount na zdrojové kódy

Jedná se o program již staršího data, avšak jistou vypovídající hodnotu má. Do těchto statistik není zahrnut program pro transformaci pcap souborů.

- Total Physical Source Lines of Code (SLOC) = 1,791
- Development Effort Estimate, Person-Years (Person-Months) = 0.37 (4.43)
  (Basic COCOMO model, Person-Months = 2.4 * (KSLOC**1.05))
- Schedule Estimate, Years (Months) = 0.37 (4.40)
  (Basic COCOMO model, Months = 2.5 * (person-months**0.38))
- Estimated Average Number of Developers (Effort/Schedule) = 1.01
- Total Estimated Cost to Develop = 49,819$

Příloha C

Slovník pojmů a zkratek

- ASN *(Autonomous System Number)* – číslo autonomního systému.
- Atribut – viz prediktor.
- BGP *(Border Gateway Protocol)* – směrovací protokol používaný mezi autonomními systémy.
- DNS *(Domain Name System)* – hierarchický systém doménových jmen, který převádí IP adresy na doménové jména a naopak. Protokol používá port 53 a transportní protokoly TCP i UDP.
- (D)DoS *((Distributed) Denial of Service)* – typ útoku na počítač nebo síť, který má způsobit nedostupnost dané služby, typicky síťového připojení. Obvykle se provádí tak, že je cílový systém zahlcen ohromným množstvím nesmyslných dotazů.
- Doména – Doménou rozumíme podstrom v grafu doménových adres. Jméno domény je cesta mezi uzlem, který tvoří vrchol domény, a kořenem DNS stromu.
- IP *(Internet Protocol)* – protokol pracující na síťové vrstvě, v současné době existuje ve dvou verzích IPv4 a IPv6.
- OOB *(Out-of-bag)* – pozorování, která nebyla použita při tvorbě stromu
- Prediktor – též proměnná či atribut, jistá sledovaná vlastnost vstupních dat (DNS dotazů), která v kontextu této práce je např. hodnota TTL
- RF *(Random Forest)* – náhodný les, detailnější popis je možno nalézt v části 5.6
- RR *(Resource Record)* – DNS záznam, blíže popsán v části 2.2
- RRset – Množina RR majících stejné jméno, třídu a typ, avšak liší se daty (RDATA)
- TLD *(Top Level Domain)* – doména nejvyšší úrovně je doména s vrcholem v uzlu ve vzdálenosti 1 od kořene grafu. Dělí se na národní (ccTLD) a generické (gTLD).
- TCP *(Transport Control Protocol)* – protokol transportní vrstvy. Zaručuje spolehlivé spojení a doručování ve správném pořadí.
- UDP *(User Datagram Protocol)* – protokol transportní vrstvy. Přenáší data mezi počítači v síti, ale narozdíl od TCP nezaručuje, že budou data doručena, případně že budou doručena v pořadí, či doručena vícekrát.