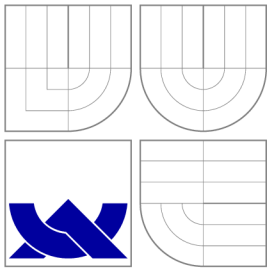
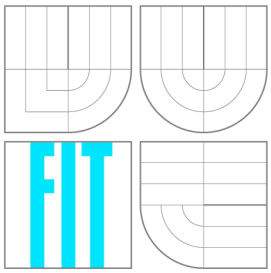


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

OBFUSKACE ANOMÁLIÍ A BEZPEČNOSTNÍCH INCIDENTŮ PŘI PROVOZU DNS

OBFUSCATION OF ANOMALIES AND SECURITY INCIDENTS IN DNS TRAFFIC

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ONDŘEJ ŠTĚRBA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IVAN HOMOLIAK

BRNO 2016

Zadání diplomové práce/18979/2015/xsterb08

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Štěrba Ondřej, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Obfuskace anomálií a bezpečnostních incidentů při provozu DNS**
Obfuscation of Anomalies and Security Incidents in DNS Traffic
Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s problematikou detekce síťových anomálií a bezpečnostních incidentů v DNS a nastudujte metody používané pro odhalování anomálií v DNS datech.
2. Navrhněte obfuskační techniky nastudovaných metod.
3. Navržené techniky implementujte do formy automatizovaného nástroje.
4. Otestujte navržené techniky na zvolených zranitelnostech s využitím dostupných metod detekce anomálií.
5. Zhodnoťte dosažené výsledky a navrhněte rozšíření.

Literatura:

- Yadav, Sandeep, et al.: Detecting algorithmically generated domain-flux attacks with DNS traffic analysis, 2012.
- Whyte, David, at al.: DNS-based Detection of Scanning Worms in an Enterprise Network, 2005.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 2 zadání.

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

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

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

Vedoucí: **Homoliak Ivan, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

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

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Práce se nejdříve zabývá analýzou současných metod detekce anomálií a bezpečnostních incidentů v DNS provozu. Později jsou v práci navrženy obfuskační techniky, pomocí kterých je možné obejít současnou anomální detekci v DNS. Pro implementační část práce byly vybrány útoky zneužívající DNS protokol na tunelování jiné síťové komunikace - konkrétně bylo uvažováno využití tunelování pro řízení a kontrolu botnetu. Hlavním cílem práce je poukázat na nutnost objevování nových přístupů pro detekci anomálií a bezpečnostních incidentů v DNS provozu.

Abstract

The work analyze current detection methods of anomalies and security incidents in DNS traffic, and than design new obfuscation techniques which are capable of evading anomaly detection. Network attacks, exploiting the DNS protocol for tunneling of other network traffic, were selected for implementation part of the work. Control of botnet is considered as malicious application of tunneling through the DNS protocol. The main result of the work is to emphasize the necessity of discovering new detection principles of anomalies and security incidents in DNS traffic.

Klíčová slova

DNS, obfuskační techniky, DNS anomálie, bezpečnostní incident v DNS, tunelování, botnet.

Keywords

DNS, obfuscation techniques, DNS anomalies, DNS security incident, tunnelling, botnet.

Citace

Ondřej Štěrba: Obfuskace anomálií a bezpečnostních
incidentů při provozu DNS, diplomová práce, Brno, FIT VUT v Brně, 2016

Obfuskace anomálií a bezpečnostních incidentů při provozu DNS

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Ivana Homoliaka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Ondřej Štěrba
27. května 2016

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu Ing. Ivanu Homoliakovi za jeho čas, cenné příspěvky a odbornou pomoc na konzultacích k této diplomové práci.

Obsah

Kapitola 1

Úvod

Obor bezpečnost síťové komunikace vznikl téměř současně jako sama síťová komunikace. Již od začátku spolu soupeří bezpečnostní experti s útočníky, kteří na síťovou infrastrukturu útočí zpravidla pro svůj profit. Nejčastěji je nové bezpečnostní opatření zavedeno až po jeho detekci. Jiným přístupem je zkoumání slabin a anomálií v zavedených postupech a protokolech a předcházení tak útokům, neboli pokoušet se přemýšlet jako útočník. Právě tento přístup byl zvolen pro vypracování této diplomové práce, která se věnuje jedné z částí síťové komunikace. Touto komunikací je DNS provoz.

DNS komunikace je jednou z nejméně monitorovaných a typický stav je, že IDS a Firewally propouštějí komunikaci obousměrně bez kontroly. Na druhou stranu velké množství síťových služeb začíná právě DNS komunikací a její využití pro nelegitimní účely je stále na vzestupu.

Jednou z metod používanou útočníky je obfuskace zlomyslného provozu. Obfuskace v kontextu této práce znamená zamaskování příznaků škodlivé komunikace takovým způsobem, aby vypadala jako legitimní komunikace. Obfuskace může probíhat na úrovni síťové komunikace bez ohledu na aplikační protokol, ale i na úrovni analýzy obsahu aplikačních dat služby DNS.

1.1 Cíl práce

Hlavní náplní této práce je analýza metod pro detekci anomálií a bezpečnostních incidentů v DNS provozu. Na základě analýzy metod jsou navrženy obfuskační techniky pro vybrané metody detekce. Dalším krokem je vytvoření automatizovaného nástroje využívajícího navržené obfuskace a otestování implementovaného nástroje. Cílem práce je odhalení slabin ve stávajících metodách pro detekci anomálií a bezpečnostních incidentů v DNS provozu, které mohou být v budoucnu využívány útočníky, případně již využívány jsou. Obejití detekčních metod poukáže na potřebu hledat univerzálnější způsoby detekce útoků.

1.2 Struktura práce

V první části práce je popsána služba DNS, anomálie a bezpečnostní incidenty týkající se této služby a metody detekce popsaných anomálií a bezpečnostních incidentů. Další kapitola popisuje infrastrukturu botnetu, na jehož řízení a kontrolu je zaměřen návrh obfuskací. Poté je popsán návrh protokolu pro řízení botnetu, který slouží pro otestování navržených obfuskací. V další kapitole jsou identifikována pole DNS protokolu, která jsou vhodná pro

využití při návrhu obfuskací a v další kapitole jsou popsány způsoby zakódování protokolu pro řízení botnetu do identifikovaných polí. Následující kapitola popisuje implementaci automatického nástroje. V poslední kapitole je uveden způsob testování, vyhodnocení testování a jsou uvedena případná rozšíření.

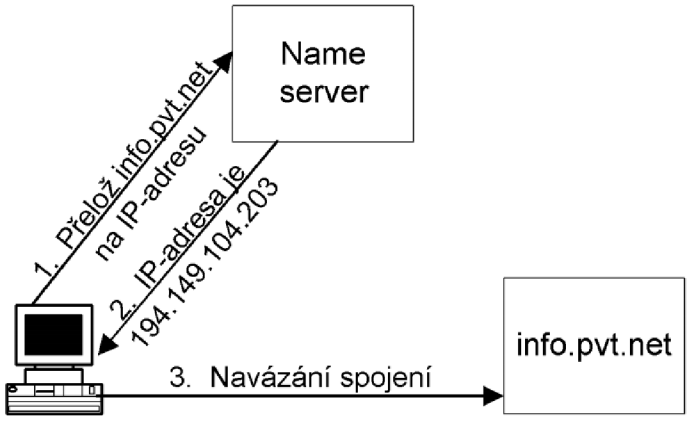
Kapitola 2

Systém DNS

DNS, neboli Domain Name System, je hiearchický systém doménových jmen. Hlavním důvodem vzniku je převod doménových jmen na IP adresy a opačně. Dnes má DNS více jak 30 různých typů záznamů. Příkladem jsou záznamy typu A pro překlad doménových jmen na IPv4 adresy, CNAME pro nastavení aliasu pro jiný doménový název, MX pro emailové servery, nebo AAAA záznamy pro převod doménových jmen na IPv6 adresy. Systém DNS se skládá ze tří základních částí, jež jsou resolver, DNS protokol a DNS servery. Každá část je popsána v následující kapitole.

2.1 Využití DNS

Pokud nějaký bod v internetové síti chce komunikovat s dalším bodem, musí znát jeho IP adresu. Tyto adresy jsou pro lidi často těžko zapamatovatelné a proto byl vymyslen DNS systém, který poskytuje mapování IP adres na doménová jména, která jsou lépe zapamatovatelná. Pokud je však některý bod v síti identifikován doménovým jménem, musí nejprve proběhnout překlad doménového jména na IP adresu. Překlad má na starost resolver.



Obrázek 2.1: DNS rezoluce [24]

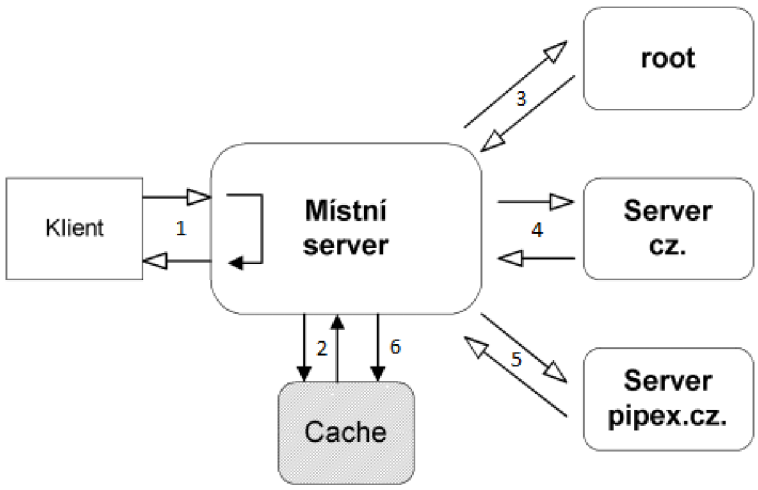
2.2 Resolver

Resolver je klient, který se dotazuje doménového serveru. Jeho činnost je naznačena na obrázku 2.1. Databáze doménových jmen je celosvětově distribuovaná a nejbližší doménový server nemusí znát odpověď. V takovém případě server spolupracuje s dalšími servery na získání odpovědi. Typ spolupráce se dělí na iterativní nebo rekurzivní zpracování.

2.2.1 Iterativní zpracování

Při iterativním zpracování dotazů vrací každý spolupracující server nejlepší možnou odpověď. Dotazovaný server pokračuje v hledání odpovědi na jednom z vrácených serverů, případně vrací odpověď při úspěšném vyhledání na jednom ze spolupracujících serverů. Na obrázku 2.2 je uveden příklad iterativního dotazu pro překlad doménového jména `test.pipex.cz` na IP adresu:

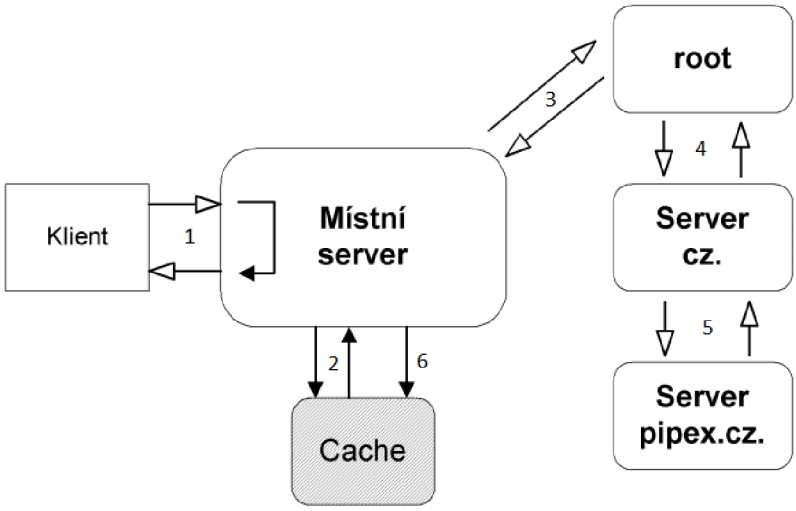
1. Klient posílá dotaz na nejbližší doménový server a čeká na odpověď.
2. Doménový server hledá odpověď ve své cache paměti. V paměti jsou jak autorizovaná data z databází na disku dotazovaného serveru, tak i neautorizovaná data ze zpracování předchozích dotazů.
3. Pokud není nalezena odpověď v cache paměti, přepošle dotaz na kořenový DNS server, který odpověď také nezná, a v odpovědi vrací servery spravující doménu `cz`.
4. Místní server se obrátí s dotazem na jeden ze serverů spravujících doménu cz. Odpovědí jsou informace o doménových serverech spravujících doménu `pipex.cz`.
5. V dalším kroku rezoluce je dotaz přeposlán na doménový server spravující doménu `pipex.cz`. Odpovědí jsou IP adresy odpovídající doménovému jménu `test.pipex.cz`.
6. Posledním krokem je uložení odpovědi do cache paměti a předání odpovědi klientovi.



Obrázek 2.2: Iterativní zpracování DNS dotazu [24]

2.2.2 Rekurzivní zpracování

Při rekurzivním zpracování dotazu se klient dotazuje místního doménového serveru, který, v případě že odpověď nezná, přeposílá dotaz kořenovému serveru. Tento server dotaz přeposílá na další servery. Dotaz se tak postupně dostane až na server spravující požadovanou doménu. Rekurzivní server při zpracování dotazu může kontaktovat i několik autoritativních serverů. Ideální pro rekurzivní zpracování by bylo, pokud by rekurzivní servery přijímaly požadavky pouze od lokálních klientů. Bohužel tomu tak není, a často přijímají všechny příchozí požadavky, které předávají na další servery podle potřeby. Takovéto rekurzivní servery se nazývají open resolvers [34].



Obrázek 2.3: Rekurzivní zpracování dotazu [24]

Na obrázku č. 2.3 je uveden příklad rekurzivního zpracování dotazu pro překlad doménového jména `test.pipex.cz` na IP adresu:

1. Resolver posílá dotaz na nejbližší doménový server a očekává odpověď.
2. Doménový server hledá odpověď ve své cache paměti.
3. Pokud nenajde odpověď na dotaz ve své cache paměti, přepošle dotaz na kořenový DNS server a očekává odpověď.
4. Kořenový doménový server nezná odpověď a dotaz přepošle na jeden ze serverů spravujících doménu `cz`. Po přeposlání dotazu čeká na odpověď.
5. Doménový server spravující doménu `cz` také nezná odpověď, ale zná správce domény `pipex.cz`. Přepošle tedy odpověď a čeká na odpověď. Doménový server pro doménu `pipex.cz` zná IP adresu a odpověď odešle zpět.
6. Posledním krokem je uložení odpovědi do cache paměti.

Doménový server u iterativního zpracování zjistí nejlepší možnou odpověď ze své databáze a tu odešle zpět. V té chvíli se veškeré jeho zdroje nutné pro zpracování uvolní a jsou k dispozici pro zpracování dalších dotazů. Oproti tomu rekurzivní zpracování je značně náročné na zdroje doménového serveru. Server čeká na odpověďi od doménových serverů,

které jsou postaveny nížeji v doménové hiearchii. Rekurzivní zpracování tedy není vhodné pro kořenové servery.

2.2.3 Autoritativní a neautoritativní odpověď

Autoritativní odpovědi poskytují primární a sekundární doménové servery pro dotazy, na které mají odpověď v databázi. Odpovědi, které jsou v cache paměti jsou neautoritativní, jelikož jejich platnost mohla být ukončena v primárním serveru.

2.3 DNS protokol

DNS protokol, popsaný v RFC 1035 [7], je založen na architektuře klient – server. Klient posílá dotaz na server a ten mu odpovídá. DNS dotazy a odpovědi se označují jako zprávy. DNS protokol patří do aplikační vrstvy a je přenášen protokoly transportní vrstvy. Pro přenos se používá jak TCP[1], tak UDP[2] protokol na portu 53. Typické je využití UDP protokolu pro přenos dotazů a odpovědí, a TCP protokolu pro přenos zónových souborů mezi primárními a sekundárními servery [24]. Ukázka DNS protokolu pro dotaz a odpověď je v příloze A.

2.3.1 DNS zpráva

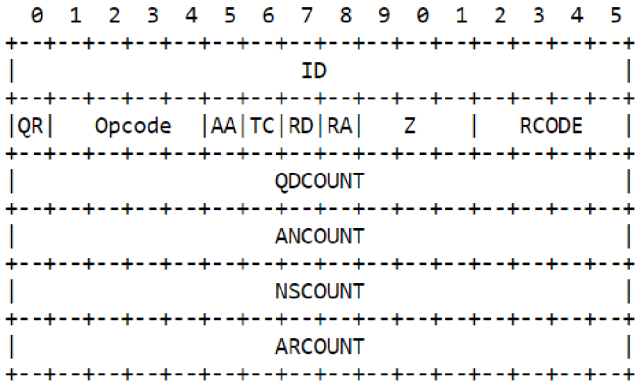
Formát DNS zprávy je následující:

- HEADER - hlavička DNS zprávy
- QUESTION - sekce s dotazy pro doménový server
- ANSWER - sekce s odpověďmi od doménového serveru
- AUTHORITY - záznamy o autoritativním serveru
- ADDITIONAL - dodatečné informace

Formát hlavičky DNS zprávy, obrázek 2.4:

- ID - 16 bitový identifikátor přiřazený klientem, který vygeneroval dotaz
- QR - jednobitový příznak identifikující dotaz nebo odpověď
- OPCODE - 4 bitové pole specifikující typ zprávy
- AA - příznak pro autoritativní odpověď
- TC - příznak pro zkrácené zprávy kvůli omezení na přenosovém kanálu
- RD - nastavení, zda má překlad probíhat rekurzivně
- RA - příznak, pokud doménový server umí rekurzivní zpracování dotazů
- Z - rezervovaná hodnota pro budoucí použití, musí být vždy 0

[1] Transmission Control Protocol
[2] User Datagram Protocol

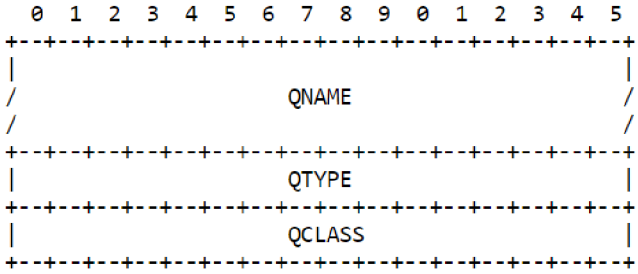


Obrázek 2.4: Hlavička DNS protokolu [7]

- AD - příznak nastaven, pokud žádná odpověď nebo záznam nebyl kryptograficky ověřen, nebo jinak nesplňuje zásady zabezpečení místního serveru (RFC 3655)
- CD - zakázání ověření podpisu
- RCODE - 4 bitové pole s návratovou hodnotou překladu
- QDCOUNT - 16 bitové číslo určující počet dotazů
- ANCOUNT - 16 bitové číslo určující počet odpovědí
- NSCOUNT - 16 bitové číslo určující počet záznamů o autoritativních serverech
- ARCOUNT - 16 bitové číslo určující počet doplňkových záznamů

Formát sekce dotazu, obrázek 2.5:

- QNAME - doménové jméno prezentované jako sekvence názvů, kde každý název je složen z počtu bytů následovaný tímto počtem uvedených bytů ukončených nulou
- QTYPE - 16 bitový kód specifikující typ dotazu
- QCLASS - 16 bitový kód specifikující třídu dotazu

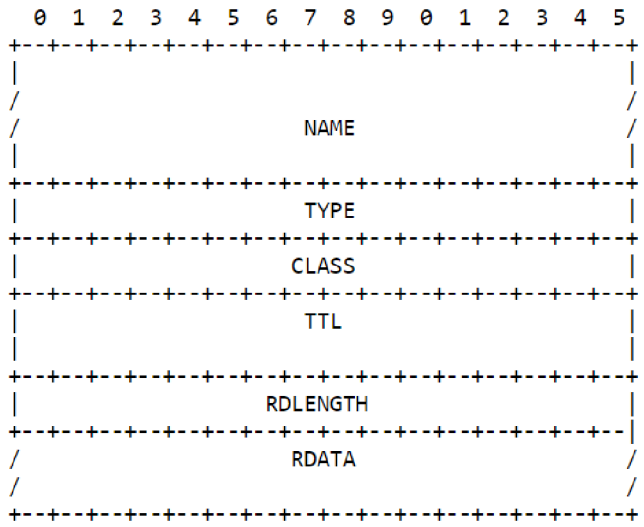


Obrázek 2.5: Formát dotazu DNS protokolu [7]

Formát sekce s odpověďmi, obrázek 2.6:

- NAME - doménové jméno, k němuž se zdrojový záznam vztahuje

- TYPE - typ záznamu odpovědi
- CLASS - třída dat
- TTL - doba platnosti záznamu v sekundách
- RDLENGTH - 16 bitové číslo udávající délku pole RDATA
- RDATA - řetězec popisující zdrojový záznam, formát je závislý na polích TYPE a CLASS



Obrázek 2.6: Formát odpovědi DNS protokolu [7]

Některá pole v DNS zprávě mají danou hodnotu, která se v současnosti typicky používá. Takovým příkladem je pole 16 bitové pole CLASS, které při použití internetové sítě musí mít vždy hodnotu IN (0x0001). Dalším příkladem je pole Z v hlavičce protokolu, které je rezervované pro budoucí použití. Jakákoliv odchylka od těchto typických nebo vyžadovaných hodnot může být považována za anomálii.

2.4 DNS servery

Každá úroveň v hiearchii doménových jmen může být na jiném serveru a mít jiného vlastníka. V této kapitole jsou popsány kořenové servery, které jsou v hiearchii nejvýše, dále jsou popsány primární a sekundární servery, které jsou používány níže v doménové hiearchii. Zvláštním typem doménového serveru je caching only server.

- Kořenový doménový server - tyto servery obsluhují kořenovou doménu. Pro internet je 13 kořenových serverů a jsou označny písmeny A až M. Každý z těchto serverů je zároveň primárním serverem.
- Prrimární server - primární server udržuje data v databázi na disku. Data primárního serveru jsou jediná, která má smysl modifikovat, jelikož právě tato data jsou autoritativní a jsou šířena dál do ostatních serverů.

- Sekundární server - sekundární servery slouží pro rozložení zátěže DNS komunikace. Pracují s kopií databáze primárního serveru, kterou si v pravidelných intervalech aktualizují. Změny v sekundárním serveru jsou pouze dočasné a budou přepsány při další aktualizaci. Aktualizace sekundárního serveru se provádí pomocí přenosu zónových souborů.
- Caching only servery - tyto servery si ukládají procházející odpovědi do paměti cache. Cache paměť mají primární i sekundární servery, ale na rozdíl od caching only serverů disponují i databází s odpověďmi.

2.5 Rozšíření DNS

DNS systém se stále vyvíjí a od své první definice byl rozšířen kvůli novým požadavkům na funkčnost systému. V této kapitole jsou popsána rozšíření DNS Security Extension, Dynamic DNS a Extension Mechanisms for DNS.

2.5.1 DNSSEC

Spolehlivá funkčnost DNS služby se stala kritickou operací pro fungování internetové infrastruktury. V původním návrhu DNS byly navrženy pouze slabé bezpečnostní mechanismy pro integritu a ověření původu dat. Rozšíření DNSSEC definované v RFC 2065 [12] umožňuje podepisování technických údajů o doménách pomocí asymetrické šifry, ale nikoliv šifrování přenášených dat. Zavádí hiarchii v ukládání veřejných klíčů v nadřazených autoritách. Proces podepisování kořenové zóny byl dokončen v roce 2010.

2.5.2 Dynamic DNS

Systém DNS byl původně navržen pro podporu dotazů na staticky konfigurované databázi. Tento návrh počítal se změnami dat v databázi, ale byla očekávána nízká frekvence těchto změn a všechny aktualizace byly provedeny v zónovém souboru. Rozšíření DDNS, popsané v RFC 2136 [9], umožňuje přidávat nebo odstraňovat záznamy z požadované zóny. Požadavky jsou specifikovány odděleně od operace aktualizace a lze určit závislost na předchozí existenci nebo neexistenci záznamu, případně existenci jediného záznamu.

2.5.3 EDNS

DNS protokol zahrnuje množství pevně daných polí, jejichž rozsah byl nebo brzy bude vyčerpán. Protokol zároveň neumožňuje odpovídající straně přijímat informace o podporovaných rozšířeních dotazující se strany. RFC 2671 [10] popisuje zpětně kompaktibilní mechanismus umožňující růst protokolu. Umožňuje objektům odesílajícím dotazy službě DNS inzerovat velikost jim odpovídajících paketů UDP a dále usnadňuje přenosy paketů větších než 512 bytů, což je omezení definované v prvním návrhu DNS služby.

Kapitola 3

Anomálie a bezpečnostní incidenty v DNS provozu

V této kapitole jsou popsány útoky na DNS službu rozdělené do dvou základních skupin [33]. Do první skupiny patří útoky mířené přímo na službu DNS, do druhé pak útoky zneužívající DNS pro vedení jiných útoků.

3.1 Útoky cílené přímo na DNS

Cílem útoku mířeného přímo na DNS může být zamezení přístupu k samotné službě, nebo falšování odpovědí. Byly identifikovány různé útoky typu DDoS, které různými způsoby zamezují přístup ke službě DNS.

3.1.1 DoS útok

Cílem útoku DoS (Denial of Service) je zamezení přístupu ke službě DNS. Speciálním případem útoku je DDoS (Distributed Denial of Service), kdy útok provádí více různých zařízení, nejčastěji spojená v tzv. Botnet. Těchto zařízení mohou být až statisíce [1]. DoS útok je charakteristický nadměrným množstvím dotazů nebo spojení, které vyčerpávají zdroje doménového serveru. Útok lze rozdělit podle různých způsobu provedení:

- DNS flood útok – vyčerpání zdrojů cílového DNS serveru za pomoci velkého množství dotazů. Nečastěji se používá nespojovaného UDP protokolu, kde útočník může podvrhnout zdrojovou IP adresu a skrývat tak svoji identitu. Jeden dotazující se uzel může generovat až 100 dotazů za sekundu. Pro úspěšné provedení útoku tedy nemusí být potřeba velké množství uzlů generujících dotazy [33].
- Rekurzivní útok – tento způsob provedení DoS útoku využívá hiearchickou strukturu DNS. Když doménový server, který má povoleno rekurzivní zpracování, dostane požadavek na překlad doménového jména, které nemá v cache paměti, odesílá dotazy na jiné DNS servery a čeká na odpověď. Tak dochází k postupnému vyčerpání zdrojů DNS serveru. Často se pro rekurzivní dotazy využívají neexistující doménová jména, u kterých má útočník jistotu, že odpověď doménový server nenajde ve své cache paměti [1].
- Garbage útok – zahlcení DNS parseru nadměrně velkými dotazy, nebo dotazy, které jsou nesmyslné.

- Reflexivní útok – je DoS útok, který využívá DNS k provedení útoku. Podrobně bude popsán v kapitole 3.2.2.

3.1.1.1 Slow DDoS útok

Zvláštním skupinou DDoS útoků jsou útoky nazvané Slow DDoS. Principem těchto útoků je posílání malého množství provozu v delším časovém úseku. Takové útoky jsou těžko detekovatelné, jelikož vypadají jako normální provoz a je tedy obtížné ohhalit IP adresy, které se na útoku podílejí. Na rozdíl od normálního provozu jsou dotazy odesílány tak, aby co nejvíce vyčerpávaly zdroje serverů. [17]

Důsledky útoku Důsledkem útoku je zpomalení doménového serveru, případně úplné jeho zahlcení. Využití spočívá v zamezení přístupu na určitou doménu, nebo využití pro vedení jiných útoku. Příkladem takového útoku je DNS cache poisoning, při kterém útočník získává čas pro určení ID transakce a pro odeslání podvržené odpovědi, právě pomocí DDoS útoku.

3.2 Útoky zneužívající DNS

U těchto útoků se využívá slabin a vlastností DNS pro vedení jiných útoků. Jako první útok tohoto typu je popsán útok DNS Cache Poisoning, další uvedený útok je DNS Amplification, následuje Domain Fast Flux a poslední je uvedeno DNS tunelování.

3.2.1 DNS cache poisoning

DNS cache poisoning, neboli otrávení cache paměti DNS serveru [16], je útok, při kterém se útočník snaží podvrhnout autoritativní odpověď. Pokud je útok úspěšný a je proveden např. na ISP server, může mít fatální následky. Útok může být podle [30], v závislosti na použitém zpracování DNS dotazu, proveden dvěma způsoby.

Při iterativním zpracování je útok spojen s útokem typu DDoS, kdy cílem útočníka je odeslat odpověď dříve, než dotazovaný server. K tomu slouží právě DDoS útok, který zahltí dotazovaný doménový server a vytvoří se prostor pro odeslání podvrhnuté odpovědi útočníkem. Při odesílání podvržené odpovědi musí znát útočník ID transakce.

U rekurzivního zpracování dotazů je odeslán dotaz na místní server a ten čeká na poskytnutí odpovědi. Útočník poté může podvrhnout odpověď tak, že ji odešle rychleji než dotazovaný doménový server. Pro úspěšné dokončení útoku si server posílající dotaz uloží podvrhnutou odpověď do své cache paměti.

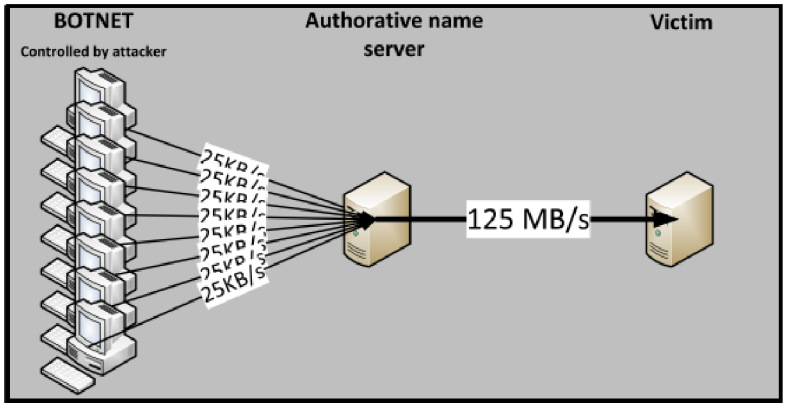
Při útoku DNS Cache Poisoning tedy dochází k závodu v rychlosti odpovědi mezi doménovým serverem a útočníkem. Útočník má závod stížen, jelikož musí uhádnout ID transakce. Pokud je doména, která je cílem útoku, například `www.example.cz`, závod proběhne pouze jednou, jelikož odpověď je uložena do cache paměti. V takovém případě musí útočník počkat, až vyprší TTL záznamu v cache paměti. Pokud je TTL nastaveno na 1 den, útok by musel trvat 84,5 roků, s 50% šancí na úspěch. Jiný přístup je popsán v [26], kde se útočník dotazuje na neexistující subdomény. Tím se zaručí provedení dotazu na každou subdoménu, a tedy závod proběhne tolikrát, kolikrát útočník potřebuje. Tohoto se dá využít například u webových stránek [33], kdy útočník na stránku umístí velké množství odkazů, které se zpracovávají při parsování html kódu. Pro každý odkaz se vytvoří DNS dotaz. Takto vznikne velké množství dotazů, u kterých útočník může hádat ID transakce.

Důsledky útoku Po uložení podvrhnuté odpovědi do cache paměti doménový server nepřeposílá dotaz dalším doménovým serverům, ale odpověď bere právě ze své cache paměti. Tento stav se nezmění do té doby, dokud uložené odpovědi nevyprší TTL (Time to Live) a nebude z paměti cache odstraněna. Tento útok patří do kategorie útoků zneužívajících DNS, jelikož je většinou součástí komplexnějšího útoku, tzv. APT[1]. Příkladem takového útoku může být šíření malwaru, získání identifikačních a autorizačních údajů vytvořením identických stránek na původní adrese, nebo útok typu Man-in-the-middle, kdy útočník filtruje obsah původních stránek, případně pozměňuje obsah [30].

3.2.2 DNS Amplification

Typ reflexního DDoS útoku, který je založen na faktu, že malé DNS dotazy mohou generovat velké odpovědi. Předchůdcem tohoto útoku je Smurf útok využívající ICMP pakety [34]. Tento útok využívá open resolver doménové servery. Aktuální počet open resolver serverů lze nalézt na [2] a dnes je jich okolo 12 milionů. Sílu tohoto útoku vyjadřuje tzv. amplifikační faktor. Tento faktor značí poměr mezi velikostí dotazu a odpovědí. DNS dotaz o velikosti 60 bytů může vygenerovat odpověď o velikosti 512 bytů. Velikost amplifikačního faktoru je v tomto případě 8.5 [34]. Útočník tak směruje velký provoz na systém oběti. Útok se skládá ze dvou částí, v prvním kroku útočník podvrhne IP adresy oběti, v druhém pak vygeneruje dotaz, který způsobí co největší odpověď. Při tomto útoku je složitá detekce útočníka. Další amplifikační útoky jsou popsány v [8].

Díky novým RFC specifikacím jako je IPv6, DNSSEC, NAPTR a dalším rozšířením DNS systému, je amplifikační faktor ještě vyšší. To je možné díky využití rozšíření EDNS, které poskytuje mechanismus pro specifikování velikosti odpovědi. Dotaz o velikosti 60 bytů, může generovat odpověď o velikosti více jak 4 000 bytů. Amplifikační faktor takového dotazu je 60 [34].



Obrázek 3.1: DNS Amplification attack

Na obrázku 3.1 je ukázka amplifikačního útoku, kde má útočník k dispozici botnet se 100 stanicemi. Každý boot odesílá DNS dotazy rychlostí 25 kB/s, DNS server odpovídá rychlostí 125 MB/s. Veškeré odpovědi jsou díky podvržení IP adresy v dotazu směrovány k oběti útoku.

Tento útok při použití IP spoofingu spadá do kategorie útoků zneužívajících DNS. Pokud je však použit bez IP spoofingu, jedná se o útok na samotnou DNS službu. Konkrétně útok

[1] Advanced Persistent Threat

typu (D)DoS, při kterém dochází k nadměrnému čerpání zdrojů.

Důsledky útoku Útočník při odesílání DNS dotazu může použít spoofing IP adresy, poté má tento útok za následek zahlcení podvržené IP adresy. Dalším důsledkem je čerpání zdrojů využitých rekurzivních DNS serverů a přenosové kapacity sítě. To může být vedlejším efektem DDoS útoku, nebo přímo cílem útočníka.

3.2.3 Domain Fast Flux

Domain Fast Flux, neboli rychlé přepínání domén, slouží k rychlé změně IP adresy odpovídající doménovému jménu. Takových adres mohou být i tisíce. Single-Flux je technika, při které jsou adresy přiřazovány a odnímány jednotlivým A záznamům. Při Double-Flux dochází ke změně nejen A záznamů, ale i NS záznamů. U obou technik jsou změny prováděny ve velmi vysoké frekvenci, což je umožněno díky nízké hodnotě TTL.

Domain Fast Flux slouží pro legální činnost, ale často je také zneužíváno pro činnost nelegální. Díky této technice může být jméno webové stránky spojeno s velkým množstvím IP adres. Opakovaným zadáním jména webové stránky se prohlížeč může pokaždé připojit k jinému fyzickému stroji. Takto lze rychle reagovat na probíhající útok typu DDoS, nebo selhání serveru na dané IP adrese. Příkladem nelegální činnosti, pro kterou může být útok zneužit, je phishing. Takový útok je popsán v [32]. Útok byl zaměřen na komunitní server MySpace. Útočník vytvořil falešnou webovou stránku login.mylspacee.com, které byla vizuálně stejná s reálnou stránkou MySpace, a požadovala zadání přihlašovacích údajů do služby. Pro ztížení odstavení falešné stránky byla použita technika Double-Flux. Cílem útoku je tedy anonymizace útočníkových strojů pro ztížení odstavení těchto strojů.

Důsledky útoku

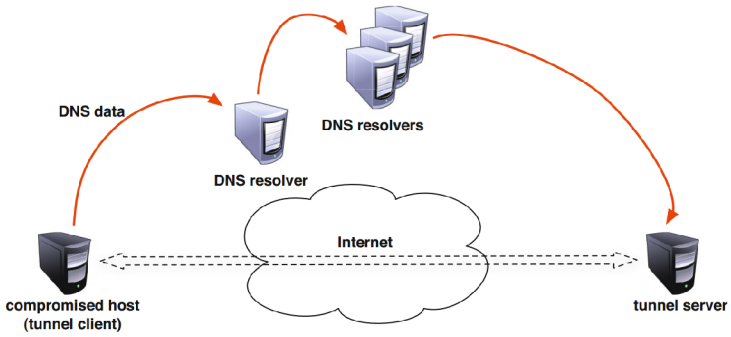
Důsledkem útoku je ztížená identifikace strojů podílejících se na nějakém útoku. Tyto stroje neustále mění svojí IP adresu díky nízké hodnotě TTL v DNS záznamu. Domain Fast Flux obchází primitivní metody detekce, jako je jednoduché filtrování nebo Blacklist.

3.2.4 Tunelování přes DNS

Tunelování přes DNS může být použito pro obejití zabezpečení. Příkladem je ilegální prohlížení webu v zpoplatněných sítích, například v hotelech nebo na letištích. To je umožněno díky faktu, že DNS provoz často není filtrován. Využití DNS tunelu nespočívá však jenom v přístupu na veřejnou část internetu, který je relativně neškodný, ale lze ho využít pro další aktivity, jejichž zneužití může mít kritické dopady. Podle [20] má DNS tunelování tři základní typy využití:

- Přenos souborů
- Interaktivní přenos (např. řízení botnetu)
- Prohlížení webu

Každé z těchto uvedených využití má charakteristické znaky. Při přenosu souborů se dosahuje vysoké rychlosti a existence tunelu není výrazně dlouhá. Interaktivní přenos má minimální přenos dat a existence tunelu je dlouhá. Tunel při prohlížení webového obsahu bude existovat po celou dobu přístupu a přenos bude středně velký.



Obrázek 3.2: DNS tunel

Na obrázku 3.2 je ilustrováno schéma DNS tunelu. Data se zabalí do DNS zprávy a odešlou se na DNS server. Na doménovém serveru dojde k extrakci dat a k provedení požadované komunikace. Požadovanou komunikací může být http dotaz, nebo jakýkoliv jiný protokol. Získaná data zabalí zpět do DNS zprávy a odešle je jako odpověď. Proces tunelování se skládá ze tří hlavních částí:

- Tunelový klient – odesílá DNS dotaz
- DNS server – extrahuje data z DNS dotazu a zabaluje do DNS odpovědi
- Tunelový server – vytváří odpověď pro klienta

Tunelování je podmíněno neexistencí pravidel pro filtrování DNS komunikace. Tato podmínka je často splněna, jelikož filtrování DNS komunikace způsobuje náchylnost k jiným útokům. Další výhodou tunelování je minimální analýza této komunikace. Pro přenos dat přes DNS tunel lze využít některý z existujících nástrojů:

- TCP-over-DNS – tunelování TCP i UDP datagramů [14]. Nástroj obsahuje jak DNS server, tak i DNS klienta. Pro spuštění vyžaduje JVM verze 6. Nástroj podporuje přenos pomocí posuvného okna pro zvýšení rychlosti a spolehlivosti a LZMA pro kompresi přenášených dat.
- Iodine – nástroj pro tunelování přes DNS [13] přenostitelný mezi UNIXovými systémy, stejně dobře jako pro systém Windows. Šířka pásma nástroje Iodine je asymetrická s omezeným uploadem a downloadem až 1Mbit/s. Iodine DNS server může obsluhovat až 16 uživatelů najednou, s podporou přihlašování.

Důsledky útoku

U tunelování je důsledkem obcházení firewallů a placených přístupových bodů do sítě, dále je možné C&C řízení bootnetu nebo krádež dat.

Kapitola 4

Metody detekce vybraných anomálií a bezpečnostních incidentů

Tato kapitola popisuje metody detekce anomálií a bezpečnostních incidentů popsaných v předchozí kapitole. Největší prostor je věnován detekci DNS tunelování, na které je zaměřen zbytek celé práce.

4.1 Metody detekce DNS Cache Poisoning

V [27] je popsán algoritmus pracující s NetFlow daty. Algotirmus pracuje s IP adresami, porty, časovými rozestupy mezi příchody jednotlivých paketů a s posloupností událostí. Na základě těchto dat je generován poplach.

Další možná detekce je popsána v [29]. Je založena na krátké historii, a zaměřuje se na detekci hádání transakčního ID. Unikátní dotazy se zaznamenávají a ukládají. Po příchodu odpovídající odpovědi je dotaz odstraněn z historie. Pokud se odpověď liší v transakčním ID, je možné, že se jedná o narozeninový útok, kterého cache poisoning využívá. Poplach se generuje až po obdržení více než jednoho paketu s různým ID. Tímto způsobem se předchází falešným poplachům. Problémem metody je efektivní ukládání historie, pokud je dotazů podstatně větší množství než odpovědí.

4.2 Detekce Domain Fast-Flux

V [35] je popsán způsob detekce zaměřený na dva různé přístupy. První způsob detekuje domény, na které chodí abnormálně vysoký počet dotazů. V druhém případě se detekují dotazy na doménová jména, která neexistují. Jedná se o odpovědi typu NXDOMAIN (Non-Existent Domain), které mohou být způsobené při snaze kontaktovat C&C server. Další způsob detekce je popsán v [18], kde autoři popisují typické vlastnosti pro Domain Fast Flux. Patří mezi ně například informace o doménových registrátorech, jelikož takovéto domény jsou často registrovány pomocí automatických nástrojů.

4.3 Metody detekce DNS Amplification

Při amplifikačním útoku se jako zdrojová IP adresa uvádí adresa vybraného cíle pro útok. Detekci lze založit na zkoumámí, zda zdrojová adresa patří do privátní sítě, ze které je paket odeslán. Další možnost detekce je popsána v [25]. Autoři pracují s předpokladem, kde každý dotaz je mapován na jednu odpověď. Odpovědi, kterým nepředchází žádný dotaz, jsou označovány jako podezřelé a je vygenerováno varování. Tato metoda dokáže pracovat v reálném čase a dosahuje velice dobrých výsledků.

4.4 Metody detekce DNS tunelu

Metody detekce DNS tunelování lze rozdělit do dvou kategorií. První kategorií je payload analýza, která je zaměřena na data přenášená aplikačním protokolem. Druhou kategorií je traffic analýza, které pro detekci používá měřitelné veličiny DNS komunikace [21].

4.4.1 Payload analýza

Detekční metody spadající do této kategorie analyzují přenášená data. V této kapitole jsou popsány metody detekce založené na entropii textu a detekce podle regulárních výrazů.

Frekvenční analýza

Doménová jména jsou vymýšlena lidmi, a často to jsou slova z běžného jazyka. Entropie doménových jmen by tedy měla být podobná jako u běžných jazyků, tak jak ji popsal Zipf [28]. V [15] je popsána metoda detekce DNS tunelu na základě frekvenční analýzy. Metoda využívá statické funkce a nepracuje v reálném čase. Analýza probíhá na unigramech, bigramech a trigramech z doménových a subdoménových jmen. Je podrobně analyzována frekvence výskytu unigramů, bigramů a trigramů v 1 000 000 nejčastěji navštěvovaných doménových jmen, ve 100 náhodně vybraných a náhodně generovaných doménových jménech. Další způsob detekce je popsán v [31]. Je založen na frekvenční analýze bigramů a na rozdíl od předchozí metody pracuje v realném čase.

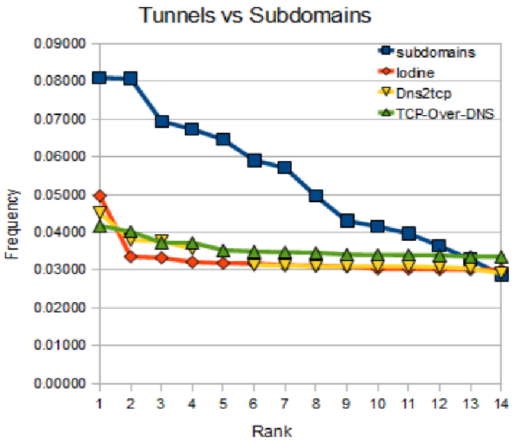
Iodine	
LETTER	FREQUENCY
a	0.04969
c	0.03360
s	0.03329
b	0.03217
q	0.03186
l	0.03177
o	0.03132
n	0.03119
m	0.03096
t	0.03038
w	0.03033
f	0.03020
r	0.03020
g	0.03006

Subdomains	
LETTER	FREQUENCY
a	0.08105
s	0.08074
e	0.06946
o	0.06738
n	0.06463
.	0.05910
i	0.05713
c	0.04961
t	0.04305
l	0.04156
m	0.03972
r	0.03643
g	0.03284
d	0.02883

Obrázek 4.1: Frekvenční chrakteristika nástroje Iodine [13] a běžně používaných jmen [15]

Tyto metody jsou založeny na předpokladu, že data přenášejí nějaký druh informace a jsou šifrována. Jejich entropie je tedy vysoká. Na obrázku č. 4.1 je zobrazena tabulka ukazující porovnání frekvenční chrakteristiky pro tunelovací nástroj Iodine [13] s běžně

používanými doménovými jmény. Přes DNS tunel byl v tomto případě přenesen jeden soubor PDF pomocí SCP přenosu.



Obrázek 4.2: Četnost výskytu znaků tunelovacích nástrojů a běžně používaných doménových jmen [15]

Na obrázku č. 4.2 je znázorněna četnost výskytu znaků pro dříve popsané nástroje pro tunelování. Graf zachycuje 100 dotazů pro každý nástroj s porovnáním frekvenční charakteristiky pro nejčastěji používaná doménová jména. Na grafu je možné pozorovat entropii u tunelovacích nástrojů, která je podstatně vyšší než u běžně používaných doménových jmen.

V [29] je popsána detekce škodlivých domén, při které jsou analyzované domény v rámci předzpracovávání rozděleny na jednotlivé úrovně domén, části kratší než 4 znaky jsou vynechány. Takto rozdělené jméno je podrobeno frekvenční analýze. Dále se analyzuje skladba slov, která se skládá z několika částí. První je analýza délky doménového jména. Hraniční hodnota délky doménového jména je stanovena na základě průměrné délky doménového jména v normálním provozu. Druhou částí je detekce počtu samohlásek, který je porovnáván oproti počtu celkového počtu písmen. V třetí části se sleduje počet opakujících se písmen v názvu domény oproti její délce. Poslední část analyzuje počet číslic v doménovém jméně.

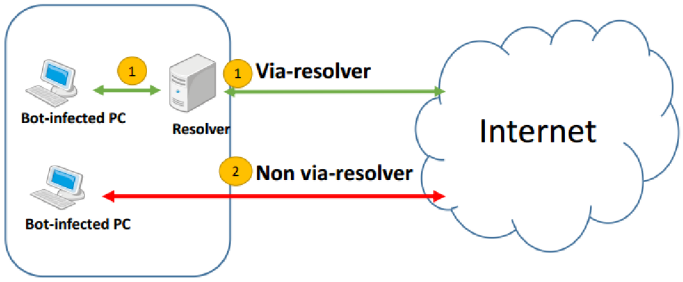
Detekce regulárními výrazy

Tunelovací nástroje lze detekovat pomocí regulárních výrazů. Tyto nástroje zanechávají v generovaných dotazech a odpovědích typické podpisy. Pokud má server nějaká data pro klienta, musí počkat, dokud klient nezačne komunikaci. K tomu se využívá polling. Takový dotaz může být pokaždé stejný a lze ho snadno zachytit pomocí regulárního výrazu. Příkladem je regulární výraz '^a-[a-zA-Z0-9]{6,6}[[.period.]]' [15], který slouží k odhalení poolingu pro nástroj DNScat [11]. Takto lze detekovat i samotný přenos dat přes DNS tunel. Regulární výrazy lze konstruovat z různých vlastností tunelovacích nástrojů, jako je délka názvu subdomén nebo množina určitých znaků na určitých pozicích.

4.4.2 Traffic analýza

Traffic analýza zkoumá několikanásobné dvojice dotaz/odpověď. Analýza provozu je univerzální pro detekci DNS tunelování. Příkladem může být množství a frekvence dotazů pro detekci tunelu. V [21] jsou popsány možnosti detekce DNS tunelu na základě traffic analýzy:

- Množství DNS provozu pro IP adresu - jedním z příkladu sledovaných metrik je množství DNS provozu vygenerovaného určitou IP adresou, jelikož tunel je typicky omezen na 512 bytů na jeden dotaz, z toho důvodu bude pravděpodobně generováno větší množství dotazů.
- Množství DNS provozu pro doménu - další analýzou je sledování provozu pro jednotlivé domény. Tunel může být nastaven tak, že se nepoužívá jenom jedna doména, ale používá se jich více. To umožňuje snížení velikosti provozu pro jednotlivé domény.
- Počet doménových jmen pro doménu - další možností je sledování počtu doménových jmen pro nějakou doménu. Některé tunelovací nástroje používají unikátní doménové jméno pro každý dotaz. V takovém případě má doména neobvykle velké množství doménových jmen, než je typické.



Obrázek 4.3: DNS komunikace s účastí a bez účasti resolveru

Na obrázku 4.3 jsou znázorněny dvě možnosti komunikace botů s C&C serverem přes DNS protokol. V prvním případě se jedná o komunikaci využívající resolver, v druhém případě se jedná o přímé posílání DNS dotazů na C&C server. V [22] je popsán způsob analýzy zaměřený na komunikaci za účasti resolveru. Analýza je zaměřena na dotazy typu DNS TXT, který je nejčastěji využíván pro skryté kanály v DNS protokolu. Pro analýzu je použit 99 denní provoz. Podíl TXT záznamů je 0,6 %, což činí 5,53 milionů záznamů. TXT záznamy využívají některé legitimní nástroje, jako jsou například SPF[1], NTP[2] a NFSv4[3]. 99,96 % z celkového počtu DNS TXT záznamů představovaly dotazy využité některými identifikovanými nástroji. Zbytek představoval 2 293 dotazů, odpovídajících 330 unikátním IP adresám. Z těchto IP adres jich 22,1 % bylo označeno jako podezřelé pomocí nástroje virustotal.com[4]. Tento přístup využívá malware Morto5.3.1, který pomocí DNS resolverů vyhledává C&C servery. V článku je zmínka o možnosti obfuskace neligitimního provozu tak, aby vypadal jako provoz některého legitimního nástroje. V takovém případě by detekce byla velmi obtížná.

[1]Sender Policy Framework
[2]Network Time Protocol
[3]Network File System
[4]online nástroj pro analýzu podezřelých souborů a URL

V [23] je popsána analýza DNS provozu zaměřená na komunikaci bez účasti resolveru. Tento typ DNS komunikace má také legitimní využití některými síťovými nástroji. Zbylý zachycený provoz byl analyzován pomocí nástroje VirusTotal. Komunikaci bez DNS resolveru využívá malware Feederbot5.3.3. V článku je zmínka o možnosti využití záznamů typu A a CNAME. V takovém případě bot odesílá dotaz typu A a C&C server odpovídá záznamem typu CNAME. Přenášená data jsou kódována do názvů subdomén. Nevýhodou tohoto přístupu je možnost přenosu pouze malého množství dat v jedné dvojici dotaz-odpověď.

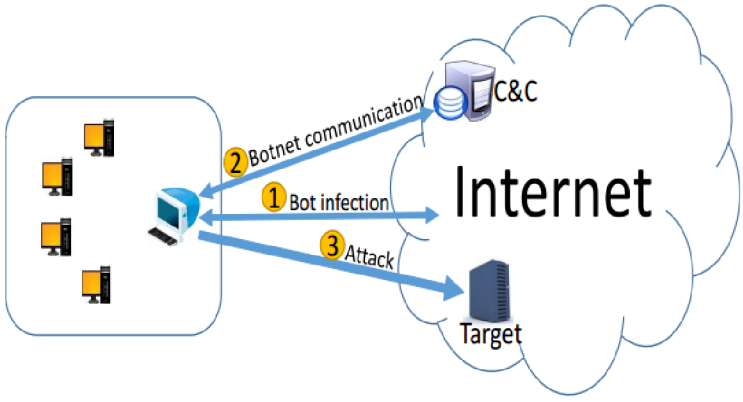
Kapitola 5

Infrastruktura botnetu

Jak již bylo zmíněno v předchozí kapitole, směr této práce je zaměřen na DNS tunelování, konkrétně využití tunelování pro interaktivní přenos při řízení a kontrole botnetu (dále C&C[1]). Pro další práci je tedy nutné alespoň stručně objasnit pojmy botnet a C&C server. Na závěr této kapitoly jsou uvedeny příklady botnetů, které využívají právě protokol DNS.

5.1 Botnet

Botnet je síť počítačů kompromitovaných speciálním softwarem, který je centrálně řízen z jednoho centra zvaného C&C server. Komunikace probíhá typicky přes běžné komunikační kanály. Botnet může provádět nelegální činnosti, jako rozesílání spamu, krádež hesel nebo DDoS útoky. Jednotlivé botnety jsou typicky pojmenovány podle malwaru, který slouží pro napadání dalších počítačů.



Obrázek 5.1: Typické kroky pro útok v botnetu[23]

Na obrázku 5.1 je zobrazena typická činnost v botnetu:

1. Nejdříve dojde ke kompromitování jednotlivých stanic botnetu.
2. Bot zahájí komunikaci s C&C serverem, ten mu pošle konfigurační příkazy.
3. Dle příkazu bot provádí různé činnosti, kterými jsou např. útoky.

[1] Z anglického Command & Control

5.2 Řízení a kontrola botnetu

Botnet je řízen a kontrolován botmasterem pomocí C&C serveru. Botmaster pomocí tohoto serveru posílá příkazy pro zahájení útoku, aktualizaci bota nebo odesílání odcizených dat. Kanál pro řízení botnetu musí být spolehlivý, redundantní, decentralizovaný a měl by vypadat jako legitimní provoz. Spolehlivost, redundance a decentralizace závisí na zvolené architektuře botnetu, legitimnost provozu závisí částečně na architektuře a částečně na obfuskování přenášených dat. Nejvyužívanějším postupem obfuskování přenášených dat je vhodné kódování do jiných legitimních protokolů. V minulosti se nejvíce používal protokol IRC[2], který byl postupně nahrazen protokoly P2P[3] a HTTP[4]. V dnešní době se využívá i DNS protokol. Příklady některých identifikovaných botnetů jsou uvedeny v následující kapitole.

5.3 Příklady Botnetů řízených přes DNS

V této kapitole jsou uvdeny tři příklady botnetů ovládaných přes protokol DNS. Prvním je botnet označovaný jako Morto [4], který přijímá příkazy přes DNS záznamy typu TXT, dalším je Katusha [3] a posledním je Feederbot, který je pravděpodobně prvním objeveným botnetem využívající DNS pro řízení botnetu [19].

5.3.1 Morto

Komunikace botnetu probíhá přes DNS protokol pomocí TXT dotazů a odpovědí. Morto využívá DNS především pro nalezení C&C serveru pomocí DNS resolverů. Odpověď je dekódována pomocí algoritmu podobného Base64 s odlišnou abecedou. Typicky se jedná o seznam modulů, které mají být staženy a provedeny. Stažení modulu, nejčastěji ve formě ddl souboru, pak probíhá například přes HTTP protokol jako stažení obrázku. Morto čeká na vypršení timeoutu pro poslání dalšího dotazu na C&C server. V [4] jsou uvedeny tři typy detekovaných modulů:

- aktualizace kódu viru
- rošíření viru pomocí slabých hesel v Remote Desktop
- spuštění útoku DDoS a zobrazování reklamy na infikovaném systému

Na obrázku 5.2 je příklad DNS dotazu a odpovědi pro malware Morto. Dotaz i odpověď jsou typu TXT , kde odpověď obsahuje informace pro provedení aktualizace.

5.3.2 Katusha/Timestamper

Tento malware využívá DNS komunikaci pro řízení a kontrolu botnetu. Cílem tohoto malwaru je získávání tajných informací a odesílání na vzdálený server. Někdy je také uváděn jako Timestamper. Tento název vychází z toho, že bot používá unixové časové razítko aktuálního data a času v dotazovaném doménovém jménu. V [19] je popsána detekce založená právě na detekci unixového časového razítko v dotazovaném doménovém jménu. Timestamper používá předkonfigurovaný DNS resolver.

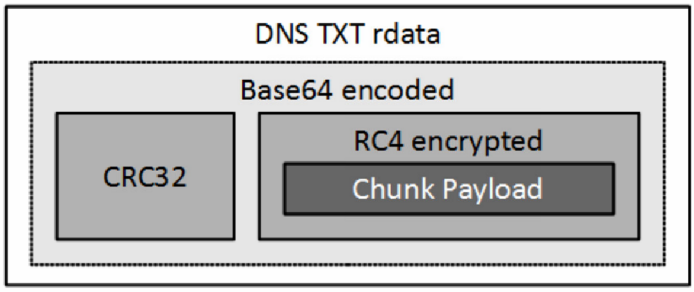
[2]Internet Relay Chat - komunikace v raálném čase po internetu
[3]Peer to Peer
[4]Hypertext Transfer Protocol

```
Domain Name System (response)
    [Request In: 110]
    [Time: 0.019860000 seconds]
    Transaction ID: 0x1639
  ⊞ Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊞ e.ppift.net: type TXT, class IN
  ⊟ Answers
    ⊟ e.ppift.net: type TXT, class IN
        Name: e.ppift.net
        Type: TXT (Text strings)
        Class: IN (0x0001)
        Time to live: 2 minutes, 20 seconds
        Data length: 167
        Text: p6666o66666CD2FCqkRaD4Fc86Jae0fXlp666666667666664kwbF0ZcukPbD!YawmRUD2J
```

Obrázek 5.2: DNS dotaz a odpověď pro Morto malware

5.3.3 Feederboot

Feederbot používá DNS přímo pro komunikaci s C&C serverem bez použití předkonfigurovaného DNS resolveru. K této komunikaci Feederbot používá DNS záznamy typu TXT. Na obrázku 5.3 je znázorněno zapouzdření dat pro řízení a kontrolu botnetu v DNS záznamu typu TXT. Přenášená data jsou rozdělena do částí o maximální délce 220 bytů. Jednotlivé části jsou dále zašifrovány pomocí proudové šifry RC4, s použitím řady různých klíčů. Pro odvození klíče, který byl použit pro zašifrování, se používá specifická část názvu domény v DNS dotazu. Výsledek dešifrování je ověřován pomocí výpočtu hashe algoritmem CRC32.



Obrázek 5.3: Zapouzdření komunikace pro botnet Feederbot[19]

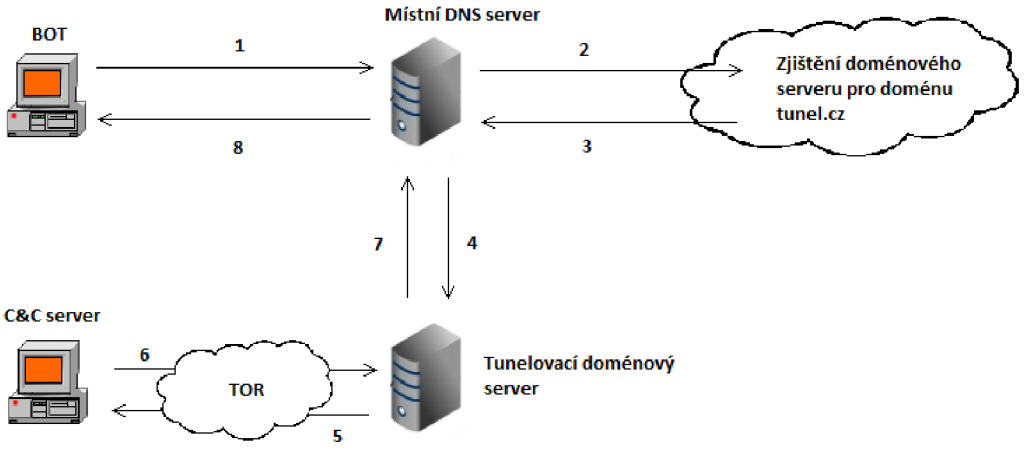
Kapitola 6

Návrh protokolu pro řízení botnetu

V této kapitole je navržen protokol pro řízení a kontrolu botnetu. Návrh samotného protokolu úzce souvisí s návrhem architektury botnetu, který je uveden v první části kapitoly. V druhé části kapitoly je popsán navržený protokol.

6.1 Architektura botnetu

Jednou z částí při návrhu protokolu pro řízení a kontrolu botnetu je návrh architektury samotného botnetu. DNS dotaz generuje bot a odesílá ho na známý doménový server, který čte ze síťové konfigurace hostitelského klienta. Dotaz se následně dostane až k doménovému serveru, který rozpozná, že se jedná o požadavek bota, a předá dotaz C&C serveru. Komunikace mezi farmou tunelovacích doménových serverů a C&C serverem probíhá pomocí sítě Tor [1], kvůli utajení C&C serveru. C&C server z dotazu extrahuje požadavek a vygeneruje odpověď. V odpovědi je zakódován příkaz dle aktuálního nastavení C&C serveru. DNS odpověď se dostane k botovi, který vygeneroval dotaz. Bot poté z odpovědi dekóduje příkaz, který provede.



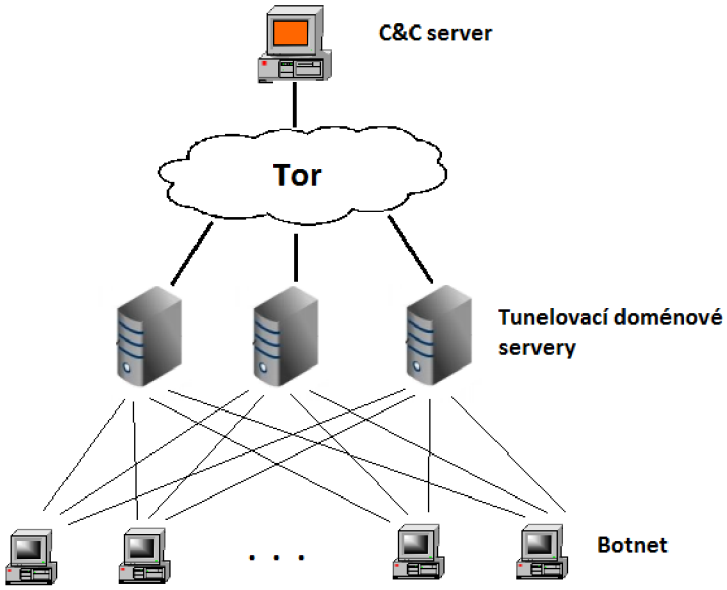
Obrázek 6.1: Příklad iterativní zpracování dotazu pro C&C server

Na obrázku 6.1 je uveden příklad rezoluce DNS dotazu v botnetu:

[1] Systém zajišťující anonymizaci uživatele při pohybu na internetu. https://www.torproject.org/.

1. BOT odesílá DNS dotaz typu A pro doménové jméno `tunel.cz` na místní doménový server.
2. Místní server iterativně zjistí IP adresu doménového serveru spravujícího doménu `tunel.cz`.
3. Doménový server spravující doménu `cz` vrací IP adresu doménového serveru spravujícího doménu `tunel.cz`.
4. Místní server odešle dotaz na doménový server pro doménu `tunel.cz`. Tento server má pod kontrolou útočník.
5. DNS dotaz je přeposlán na C&C server. Komunikace mezi doménovým tunelovacím serverem probíhá přes TOR síť. Z DNS dotazu je dekódován požadavek od bota.
6. C&C vrací vygenerovanou odpověď tunelovacímu serveru, ve které je zakódován příkaz a jeho parametry.
7. Odpověď je přeposlána na místní doménový server.
8. Z místního doménového serveru je dále přeposlána botovi, který z odpovědi dekóduje příkaz a provede ho.

Na obrázku 6.2 je zobrazena architektura celého botnetu. V kapitole 5.2 jsou uvedeny některé požadavky na kanál pro řízení botnetu. Mezi tyto vlastnosti patří redundance a decentralizace. Tato architektura splňuje požadavky na redundanci a decentralizaci díky několika tunelovacím doménovým serverům. Každý bot udržuje seznam těchto serverů a v případě nemožnosti navázat spojení s nějakým serverem ze seznamu může použít jiný. Správu seznamu doménových serverů umožňuje navržený protokol popsaný v následující kapitole. Doménové servery komunikují s C&C serverem pomocí anonymizační sítě Tor.



Obrázek 6.2: Architektura botnetu

6.2 Protokol botnetu

Navržený protokol je bezstavový, stejně jako je DNS protokol. Součástí protokolu je jednoduchá autentizace, zaručení integrity přenášených dat a pevně dané příkazy jak pro bota, tak pro C&C server.

6.2.1 Autentizace

Autentizace probíhá pouze jednostranně a to C&C serveru oproti botovi. Autentizace je důležitá zejména proti odcizení jednotlivých botů z botnetu, případně celého botnetu. Bot a C&C server znají sdílený klíč. C&C počítá hash (MD5) ze všech parametrů příkazu a identifikátoru příkazu společně se sdíleným klíčem. Spočítaná hodnota, která má velikost 128 bitů, je ořezána na 48 bitů a vložena do odpovědi. Tímto mechanismem lze zajistit jednoduchou autentizaci C&C vůči botovi a zároveň integritu přenášených příkazů a jejich parametrů.

6.2.2 Popis příkazů C&C protokolu

Seznam příkazů pro bota:

1. `bot.keep_alive` - ohlášení bota k C&C serveru
2. `bot.send_data` <`data`> - odeslání dat na C&C server
3. `bot.ddos.fin` <`target`> - informování C&C serveru o ukončení útoku typu DDoS
4. `bot.ampl.fin` <`target`> - informování C&C serveru o ukončení útoku Amplification Attack
5. `bot.slowddos.fin` <`target`> - informování C&C serveru o ukončení útoku Slow DDoS

Příkazy určené C&C serveru se dělí na skupiny oddělené prefixem. Základní rozdělení je na příkazy určené pro konfiguraci bota a příkazy pro řízení útoků. Každý příkaz také obsahuje autentizační data. Seznam příkazů pro C&C server:

1. `cc.keep_alive.ack` - odpověď bez příkazu
2. `cc.conf.remove` - odstranění bota z hosta
3. `cc.conf.update` <`url`> - aktualizace bota ze zadané url
4. `cc.conf.add_domain_name` <`domain_name`> - přidání doménové adresy do seznamu tunelovacích doménových serverů
5. `cc.conf.remove_domain_name` <`domain_name`> - odstranění doménové adresy ze seznamu tunelovacích doménových serverů
6. `cc.conf.set_time` <`time`> - příkaz pro synchronizaci času mezi botem a C&C serverem
7. `cc.conf.traffic_size` <`size`> - velikost generovaného provozu botem při útocích

8. `cc.conf.timeout` <`timeout`> - nastavení doby, po které se má bot ohlásit C&C serveru
9. `cc.ddos.flood` <`target`> <`time`><`duration`> - příkaz pro zahájení útoku typu DDoS na určený cíl, v daný čas a trvající zadanou dobu
10. `cc.ddos.fin.ack` <`target`> - potvrzení o ukončení útoku, parametrem je cíl útoku
11. `cc.ampl.flood` <`spoofed_ip`> <`target`> <`time`><`duration`> - příkaz pro zahájení útoku typu Amplification attack, parametry příkazu jsou podvržená IP adresa, adresa doménového serveru využitého pro útok a čas zahájení útoku
12. `cc.ampl.fin.ack` <`target`> - potvrzení o ukončení útoku, parametrem je cíl útoku
13. `cc.slowddos.flood` <`target`> <`time`><`duration`> - zahájení útoku typu Slow DDoS, parametry příkazu jsou cíl, čas spuštění a doba trvání útoku
14. `cc.slowddos.fin.ack` <`target`> - potvrzení o ukončení útoku, parametrem je cíl útoku

6.2.3 Formální popis protokolu

Pro formální popis navrženého protokolu je použita bezkontextová gramatika.

$$
\begin{aligned}
G &= (N, \Sigma, P, TIMEOUT) \\
N &= \{TIMEOUT, CC\_ACTION, CONF, DDOS\_ATTACK, \\
&\quad DDOS\_FIN, AMPL, AMPL\_ATTACK, AMPL\_FIN, \\
&\quad SLOW\_DDOS, SLOW\_DDOS\_ATTACK, SLOW\_DDOS\_FIN\} \\
\Sigma &= \{bot.keepAlive, cc.conf*, cc.conf, cc.ddos.flood, \\
&\quad bot.ddos.fin, cc.ddos.fin.ack, cc.ampl.flood, \\
&\quad bot.ampl.fin, cc.ampl.fin.ack, cc.slow\_ddos.flood \\
&\quad bot.slow\_ddos.fin, cc.slow\_ddos.fin.ack\} \\
P &:
\end{aligned}
$$

$$
\begin{aligned}
TIMEOUT &\rightarrow bot.keepAlive\ CC\_ACTION \\
CC\_ACTION &\rightarrow cc.keepAlive.ack \mid CONF \mid DDOS \mid AMPL \mid SLOW\_DDOS \\
CONF &\rightarrow cc.conf.^* \\
DDOS &\rightarrow cc.ddos.flood\ DDOS\_ATTACK \\
DDOS\_ATTACK &\rightarrow bot.ddos.fin\ DDOS\_FIN \\
DDOS\_FIN &\rightarrow cc.ddos.fin.ack \mid CC\_ACTION \\
AMPL &\rightarrow cc.ampl.flood\ AMPL\_ATTACK \\
AMPL\_ATTACK &\rightarrow bot.ampl.fin\ AMPL\_FIN \\
AMPL\_FIN &\rightarrow cc.ampl.fin.ack \mid CC\_ACTION \\
SLOW\_DDOS &\rightarrow cc.slow\_ddos.flood\ SLOW\_DDOS\_ATTACK \\
SLOW\_DDOS\_ATTACK &\rightarrow bot.slow\_ddos.fin\ SLOW\_DDOS\_FIN \\
SLOW\_DDOS\_FIN &\rightarrow cc.slow\_ddos.fin.ack\ CC\_ACTION
\end{aligned}
$$

Kapitola 7

Návrh obfuskací

Obfuskací se v kontextu této myslí zamaskování vlastností entit takovým způsobem, aby interpretace těchto vlastností neumožnila jednoznačnou identifikaci entit na základě rozložení pravděpodobnosti výskytu hodnot pro jednotlivé vlastnosti. Z jiného pohledu to je zamaskování příslušnosti entit do klasifikačních tříd.

V předchozí kapitole jsou popsány metody detekce útoků. Obfuskování ale nelze navrhnout pro všechny metody. Například obfuskování DNS Cache Poisoning nemá smysl vzhledem k popisovaným metodám detekce. Nejvhodnějším kandidátem pro návrh obfuskací je DNS tunelování. V této kapitole jsou navrženy obfuskace pro jednotlivé detekční metody DNS tunelování.

7.1 Obfuskace metod detekce DNS tunelu založených na payload analýze

Tyto metody jsou založeny především na frekvenční charakteristice nebo detekci pomocí regulárních výrazů. V této kapitole jsou navrženy obfuskace pro obě varianty.

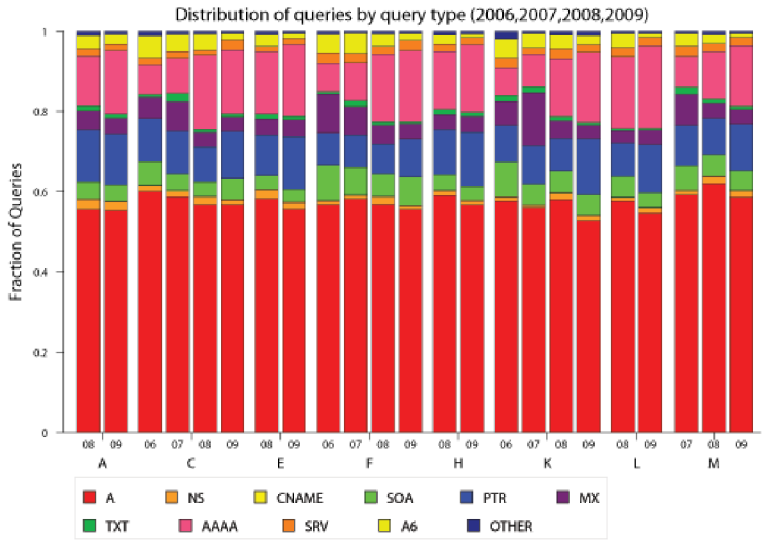
7.1.1 Obfuskace frekvenční analýzy

Výchozí frekvenční chrakteristikou pro porovnávání slouží frekvenční charakteristika nejčastěji používaných doménových jmen. Cílem takové obfuskace by tedy mělo být vhodné kódování, pro co největší se přiblížení frekvenční charakteristice nejčastějších doménových jmen. Kódování je ztíženo faktem, že povolené znaky jsou podmnožina ASCII znaků, která obsahuje znaky a-z, A-Z, čísla od 0 do 9 a pomlčku.

Většina tunelovacích nástrojů, botnetů využívajích DNS pro řízení a kontrolu a dalších případů, kdy se DNS využívá jako skrytý kanál, využívá DNS záznam typu TXT. Právě na tento typ záznamu je zaměřená většina detekčních nástrojů. TXT záznam má v legitimním provozu pouze malé zastoupení (do 1 %) a většina legitimního provozu je známá a dá se rozpoznat pomocí regulárních výrazů. Zbytek takového provozu je podroben analýze. Při návrhu obfuskací je vhodné se vyhnout TXT záznamům a použít záznamy, které se v DNS provozu vyskytují častěji a jsou méně podrobeny analýze. Na obrázku 7.1 je vidět největší zastoupení dotazů typu A. Právě tento typ bude použit při posílání dotazu bota na C&C server.

Další možností je obfuskovat nelegitimní provoz tak, aby vypadal jako legitimní provoz některého ze známých nástrojů využívajících záznamy TXT. Tento přístup je opakem

obfuskace popsané v další kapitole.



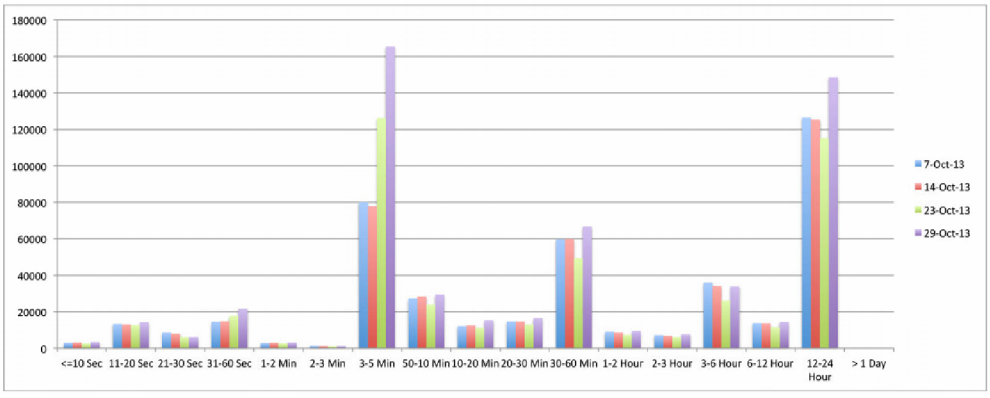
Obrázek 7.1: Nejčastější typy dotazů

7.1.2 Obfuskace detekce pomocí regulárních výrazů

Jak je popsáno v kapitole 4.1.1, některé tunelovací nástroje zanechávají v přenášených datech svůj typický podpis. Tyto nástroje pak lze snadno detekovat pomocí regulárních výrazů. V obfuskačním nástroji by se takový podpis neměl v přenášených datech vyskytovat.

7.1.3 TTL

TTL je časový údaj, který definuje počet sekund, po které má být DNS uložena v cache paměti doménových serverů. Velikost pole v DNS zprávě je 32 bitů.



Obrázek 7.2: Histogram hodnoty TTL v DNS odpovědích

Na obrázku 7.2 je histogram zobrazující hodnoty TTL pro DNS odpovědi. Pro potřeby řízení botnetu je potřeba využít nízké hodnoty TTL, kvůli komunikaci od bota ke C&C

serveru. Na druhou stranu je ale nutné zvolit takovou hodnotu, která nebude nijak abnormální vůči nejčastějším hodnotám. Hodnota by tedy neměla být nižší než 10 sekund, jelikož dle histogramu se tato hodnota nevyužívá často. Nejvhodnější hodnota by měla být zvolena z intervalu 11 až 20 sekund, s ohledem na potřeby botnetu a obfuskace nelegitimního provozu.

7.2 Obfuskace metod detekce DNS tunelu založených na traffic analýze

V této kapitole jsou navrženy obfuskace pro metody detekce založené na traffic analýze. Tyto metody zkoumají měřitelné charakteristiky přenosu, jako je množství DNS komunikace pro IP adresu, množství DNS komunikace pro doménu nebo počet doménových jmen pro doménu.

7.2.1 Použití resolveru

Některé metody se zaměřují na zkoumání, jestli je DNS odeslán přes klientský resolver, nebo je odeslán přímo na doménový server. V rámci obfuskací by bylo nejlepší dotazy posílat přes resolver. To však není možné, pokud se data budou kódovat do některých polí DNS hlavičky. V takovém případě je nutné odesílat dotaz přímo na doménový server.

7.2.2 Množství DNS provozu pro IP adresu

Množství DNS provozu pro IP adresu představuje největší problém a je těžké ho změnit, pokud útočník potřebuje přenést určitá data. Jednou z možností je komprese přenášených dat. Detekční metody založené na sledování DNS provozu typicky používají metodu prahování, kde práh může být stanoven na základě trénovacích dat, ale častěji se jeho hodnota počítá z aktuálního provozu. Pak je cílem útočníka zakrýt zvýšený provoz pro IP adresu, ze které se tuneluje. To může být uskutečněno pomocí generování DNS dotazů s podvrženou zdrojovou IP adresou, což by mělo za následek zvýšení DNS provozu i pro ostatní IP adresy.

V rámci implementace automatického nástroje budou přenášená data kódována z 8 bitů na 6 bitů, což bude mít za důsledek snížení provozu pro danou IP adresu.

7.2.3 Množství DNS provozu pro doménu

Další metriku, kterou lze sledovat, je množství provozu pro domény. U tunelovacích DNS serverů bude provoz nepochybně vyšší. Tato vlastnost se dá skrýt použitím více domén. Jiným způsobem obfuskace může být generování falešného DNS provozu pro jiné domény a zvýšit tak DNS provoz i pro ostatní domény.

7.2.4 Počet doménových jmen pro doménu

Další sledovanou vlastností u DNS tunelů je počet subdomén pro jednotlivé domény. Zde se také jako nejlepší možná obfuskace zdá použití více domén a pevně daný počet subdoménových jmen pro každou doménu.

Kapitola 8

Kódování C&C protokolu

V první části této kapitoly jsou popsány vhodná pole DNS zpráv pro přenos dat protokolu pro řízení a kontrolu botnetu. V druhé části jsou jednotlivé příkazy a další data, které je nutné přenášet, zakódovány do popsaných položek DNS zprávy.

8.1 Pole DNS zprávy vhodná pro skryté kanály

DNS dotaz se skládá z hlavičky, polí identifikujících počty jednotlivých dotazů a odpovědí a samotných dotazů. Pole identifikující počet odpovědí musí být nulové, stejně tak DNS dotaz nesmí obsahovat žádné odpovědi. Vhodnými poli pro přenos skrytého kanálu jsou transakční ID z hlavičky protokolu a název doménového jména pro překlad ze sekce dotazů protokolu. DNS odpověď je složena z hlavičky, dotazu, který je zkopírován z DNS dotazu, a části s odpověďmi. Odpovědi se dělí do tří sekcí. V první sekci jsou přímé odpovědi na dotaz, v druhé sekci jsou odpovědi s informacemi o doménových serverech pro dotazované jméno a v poslední sekci jsou dodatečné informace, které nejsou přímou odpovědí na dotaz, ale vztahují se k dotazu. Jak již bylo uvedeno dříve, s ohledem na statistiku využití jednotlivých typů záznamů, nejvhodnější je využít záznam typu A a CNAME.

8.1.1 Transaction ID

Každá zpráva má přiřazeno transakční ID, které generuje klientský resolver. Jedná se o libovolné číslo, jehož velikost je 16 bitů. Vygenerovaný identifkátor musí být zkopírován do odpovědi, pomocí něho dochází ke spárování dvojice dotaz - odpověď. Pokud bot potřebuje odeslat nějaká data na C&C server, může využít právě toto pole.

8.1.2 Doménové jméno

Doménové jméno je reprezentováno jako sekvence názvů, kde každý název předchází údaj o délce následujícího názvu. Doménové jméno je ukončeno nulovou délkou následujícího názvu, který identifikuje kořenovou doménu. Počet bytů může být lichý a není použito žádné zarovnání. Jednotlivá jména se označují jako domény 1., 2. a 3. řádu. Pro správné fungování navržené architektury botnetu musí jména 1. a 2. (případně 3.) řádu identifikovat tunelovací doménový server. Jméno dalšího řádu lze použít pro zakódování data od bota ke C&C serveru. Takové jméno ale nesmí být příliš dlouhé ani náchylné na detekci pomocí frekvenční analýzy. Kódování dat do doménového jména lze využít jak od bota ke C&C

serveru, kde se jedná o doménové jméno na které se bot dotazuje, tak směrem od C&C serveru k botovi. V druhém případě lze využít odpovědi typu CNAME.

8.1.3 Odpověď typu A

V této odpovědi je vrácena IP adresa verze 4, pro doménu uvedenou v dotazu. IP adresa verze 4 má 32 bitů, které se dají využít pro skrytí přenášených dat. Některé adresy by se však v internetovém provozu vyskytoval neměly. Jsou to broadcastové adresy, které mají hodnotu posledního bytu 255, dále to jsou především privátní adresy, které jsou pro jednotlivé třídy určeny následně:

- 10.0.0.0 až 10.255.255.255 pro třídu A
- 172.16.0.0 až 172.31.255.255 pro třídu B
- 192.168.0.0 až 192.168.255.255 pro třídu C

8.1.4 Záznam CNAME

Záznam typu CNAME slouží pro vytváření aliasů. Vlastníkem daného jména je vždy právě alias. Tento záznam lze využít pro přenos doménových jmen. Tato jména ale musí sloužit výhradně pro účely botnetu a nesmí se takto přenášet doménová jména běžně používaná v internetové síti. Pokud by se přenášela doménová jména, která se běžně využívají, vznikaly by anomálie při ukládání v cache pamětích doménových serverů. Tyto anomálie by bylo možné detekovat. Další využití může být pro přenos dat v těchto jménech. Kódování by ale muselo být navrženo s ohledem na možnost detekce pomocí frekvenčních charakteristik. Při využití záznamu CNAME je důležité dodržet pravidlo, že každé jméno by mělo mít právě jeden alias a následující odpovědí typu A je odpovědí na dotazované jméno, nebo poslední alias.

8.2 Kódování jednotlivých příkazů C&C protokolu

Komunikaci začíná vždy bot ve stanovený čas. Tento čas je určen z hodnoty TTL z předchozí komunikace. Po uplynutí časového intervalu je jisté, že odpověď je odstraněna z cache paměti doménových serverů po cestě mezi botem a tunelovacím doménovým serverem. Bot odesílá hodnotu *nonce*, která slouží k autentifikaci C&C serveru vůči botu. Z navržených příkazů pro bota vyplývají další nutná data, která je potřeba přenášet. Je jím typ příkazu a parametry spojené s daným typem příkazu. Typ příkazu lze kódovat do pole s id transakce.

V návrhu je popsáno 5 typů příkazů. Těchto 5 příkazů lze zakódovat do 3 spodních bitů transakčního id. Kódování příkazů je navrženo následně:

1. `bot.keepalive` - hodnota nonce slouží k autentifikaci C&C vůči botu. Tato hodnota je uložena v poli s transakčním id. Spodní tři bity určují typ příkazu, zbytek 16 bitového id je náhodně generován, případně jsou do něj kódovány další přenášená data popsaná v následujících příkazech
2. `bot.send_data` <`data`> - data jsou uložena do dotazovaného doménového jména.
3. `bot.ddos.fin` <`target`> - cíl útoku je kódován pomocí identifikátoru útoku do pole s transakčním ID.

4. `bot.ampl.fin <target>` - cíl útoku je kódován pomocí identifikátoru útoku do pole s transakčním ID.

5. `bot.slowddos.fin <target>` - cíl útoku je kódován pomocí identifikátoru útoku do pole s transakčním ID.

Všechny odpovědi obsahují minimálně 3 odpovědi. První je typu CNAME, ve které je uložen typ příkazu, další dvě odpovědi jsou typu A a obsahují autentizační údaje. Tyto tři položky nebudou dále uváděny při popisu kódování jednotlivých příkazů.

1. `cc.keepalive.ack` - pouze odpovědi pro určení typu příkazu a pro autentizaci.

2. `cc.conf.remove` - stejně jako příkaz `cc.keepalive.ack`.

3. `cc.conf.update <URL>` - adresa, která má být použita pro aktualizaci bota, je uložena v odpovědi typu CNAME jako alias doménového jména.

4. `cc.conf.add_domain_name <domain_name>` - stejně jako příkaz `cc.conf.update`.

5. `cc.conf.remove_domain_name <domain_name>` - stejně jako příkaz `cc.conf.update`.

6. `cc.conf.settime <time>` - pro zákodování času je potřeba minimálně 11 bitů. Těchto 11 bitů může být zakódováno do spodních 11 bitů IP adresy.

7. `cc.conf.trafficsize <size>` - stejně jako příkaz CC.CONF.SETTIME.

8. `cc.ddos.flood <target> <time> <duration>` - cíl útoku je kódován z 8 bitů na 6 bitů pro jeden znak. Takto zakódovaný řetězec je poté zakódován do IP adres. V další IP adrese je zakódován čas útoku a jeho trvání.

9. `cc.ddos.fin.ack <target>` - pouze typ příkazu a autentizační údaje, identifikátor útoku je již uložen v transakčním ID.

10. `cc.ampl.flood <spoofedIP> <target> <time> <duration>` - cíl útoku je uložen stejně jako v příkazu `cc.ddos.flood`, následuje IP adresa, která nese čas útoku a jeho trvání. Další IP adresa je podvržená IP adresa.

11. `cc.ampl.fin.ack <target>` - pouze typ příkazu a autentizační údaje, identifikátor útoku je již uložen v transakčním ID.

12. `cc.slowddos.flood <target> <time>` <duration> - stejné kódování jako u příkazu `cc.ddos.flood`.

13. `cc.slowddos.fin.ack <target>` - pouze typ příkazu a autentizační údaje, identifikátor útoku je již uložen v transakčním ID.

Kapitola 9

Implementace

Kapitola implementace popisuje zvolené technologie pro vývoj, testování a vyhodnocení automatického nástroje pro obfuskaci tunelování přes DNS protokol. Dále je popsána organizace archivu se zdrojovými kódy, architektura zvolená pro testování a nakonec je uveden popis jednotlivých programů podílejících se na funkčnosti botnetu.

9.1 Použité technologie

V této kapitole jsou uvedeny použité technologie v implementaci a testování. Pro implementaci byl zvolen jazyk C, jako doménový server byl zvolen open source projekt MaraDNS a pro simulaci navržené architektury byl zvolen virtualizační nástroj Oracle VM VirtualBox.

9.1.1 Jazyk C

Jazyk C je nízkoúrovňový a kompilovaný jazyk, který je velice výkonný. Právě proto se nejčastěji využívá pro psaní systémového softwaru. Umožňuje vytváření procesů a vláken, komunikaci pomocí semaforů a mutexů, nebo možnost spolupráce procesů pomocí sdílené paměti. Dále poskytuje možnost využití soketů, které slouží ke komunikaci prostřednictvím internetové sítě.

9.1.2 MaraDNS

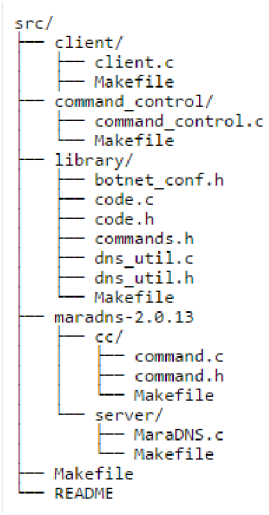
MaraDNS [5] je open source program implementující doménový server. Server může být spuštěn jako autoritativní nebo rekurzivní. Pokud je server spuštěn jako autoritativní, je potřeba provést konfiguraci zónového souboru *db.example.net*. Ten je po instalaci umístěn v adresáři */etc/maradns*. Název zónového souboru lze měnit podle názvu zóny. Změněný název je pak potřeba změnit v konfuguračním souboru *mararc*, který je umístěn v adresáři */etc*. V tomto souboru se také nastavuje IP adresa, na které má doménový server naslouchat.

9.1.3 Oracle VM VirtualBox

Oracle VM VirtualBox [6] je multiplatformní virtualizační nástroj, který umožňuje spuštění dalšího operačního systému. Nejdůležitější vlastnost pro tuto práci je možnost spojovat virtuální stroje do virtuální sítě.

9.2 Organizace archivu se zdrojovými kódy

Na obrázku 9.1 je zobrazena adresářová struktura zdrojových souborů. Kompilování a instalaci lze provádět pomocí Makefilu v nejvyšší složce. Složka *client* obsahuje zdrojový soubor pro program klienta, neboli bota, a soubor Makefile pro kompilování klienta. Složka *command_control* obsahuje zdrojový soubor pro program C&C serveru a soubor Makefile pro kompilování programu. Složka *library* obsahuje sdílené funkce klientem a C&C serverem. V souboru *botnet_conf.h* jsou sdílené konfigurační hodnoty. Knihovna *code.h* implementuje funkce pro kódování a dekódování dat pro ukládání přes IP adresy. Knihovna *dns_util.h* má funkci pro kódování doménového jména do formátu, který se přenáší v DNS protokolu, a druhou funkci, která provádí dekódování. V poslední knihovně *commands.h* jsou definovány jednotlivé příkazy pro řízení a kontrolu botnetu.



Obrázek 9.1: Strom zdrojových souborů

9.3 Klient

Proces klienta běží v nekonečné smyčce. Jeho činnost sestává z odeslání DNS dotazu, příjmu odpovědi, zpracování odpovědi a provedení příkazu. Po vykonání všech těchto kroků je uspán a čeká na vypršení timeoutu. Timeout je nastaven na výchozí hodnotu, nebo ho lze konfigurovat pomocí příkazu. Provedení příkazu je simulováno výpisem hlášky s dekódovaným příkazem a jeho parametry na standardní výstup.

Popis základních funkcí:

- `main()` - hlavní smyčka programu odesílá dotazy, přijímá odpovědi a dekóduje příkazy. Odesílané dotazy jsou vždy typu A, a dotazuje se na doménové jméno z listu doménových jmen. Jména se vybírají náhodně. Pokud není odpověď doručena do vypršení timeoutu, je vygenerován nový dotaz.

- `prepareQuery()` - příprava dotazu pro odeslání na doménový server. Do transakčního ID dotazu je uložen typ příkazu. Do doménového jména mohou být uložena přenášená data.

- `getCommandFromResponse()` - získá typ příkazu od C&C serveru z první odpovědi. První odpověď je vždy typu CNAME. Dle typu příkazu jsou známy typy dalších odpovědí, které jsou čteny následujícími funkcemi.
- `readARecord()` - čte odpověď typu A. Pokud není odpověď A, je to považováno za chybu.
- `readCnameRecord()` - čte odpověď typu CNAME. Pokud odpověď není typu CNAME, je to považováno za chybu.
- `getDnsServers()` - čte nakonfigurované doménové servery ze souboru /etc/resolv.conf, které jsou použity pro odesílání DNS dotazů.
- `parse*Command()` - funkce jsou pojmenovány podle typu příkazu, který má číst. Činnost všech těchto funkcí je podobná, nejprve se načtou všechny parametry příkazu pomocí funkcí readARecord() a readCnameRecord(), z posledních dvou odpovědí se načte hash, který se následně porovná s hashem spočítaným z parametrů příkazu a sdíleného klíče, a připočtené hodnotě pro vypočítání validního hashe. Za validní hash je považován takový hash, po jehož uložení do IP adres nejsou tyto adresy nelegitimní z pohledu výskytu na veřejné internetové síti. Dále je tento hash porovnán s posledním přijatým hashem, podle toho se rozhodne zda byl příkaz už přijat, nebo se jedná o nový příkaz.

Na obrázku 9.2 je uveden příklad použití programu `client`. Program je spuštěn v režimu, kdy periodicky komunikuje s C&C serverem a vypisuje dekódované příkazy. Znak '.' je vypsán, pokud už jednou byl přijat stejný příkaz. Na obrázku jsou vypsány příkazy pro nastavení timeoutu, přidání nového doménového jména pro spojení s C&C serverem a zahájení útoku DDoS.

```
Received Keep Alive Ack command.
.........
Set new timeout '8'.
..
Add new domain name 'new-tunel.cz' to list.
Current list of names is: tunel.cz new-tunel.cz
..
Received command for start of DDoS attack on target '45.46.47.48' in time '15:30' duration '1800'.
..
```

Obrázek 9.2: Program klienta

9.4 Doménový server

Předpokladem funkční architektury navržené v kapitole je nutnost vlastnit doménové servery botmasterem, nebo mít k dispozici kompromitované doménové servery. Kompromitované ve smyslu odklonění procesu vytvoření odpovědi na C&C server. Proces je odkloněn, pokud je detekováno doménové jméno, které je určeno pro dotazování se botů pro příkazy od C&C serveru. Pro testování implementace byla zvolena varianta s kompromitovaným doménovým serverem. Dotaz je tedy předáván procesu C&C serveru přes sdílenou paměť synchronizovanou pomocí semaforů. Po zpracování dotazu je vygenerovaná odpověď uložena do sdílené paměti a odeslána zpět klientovi, který vygeneroval dotaz.

9.5 C&C server

Proces C&C serveru je spuštěn ve dvou vláknech. První vlákno slouží pro interaktivní konfiguraci C&C serveru. Lze zadat příkaz, který má být distribuován do botnetu, a jeho parametry. Po zpracování parametrů dojde k jejich uložení a jsou k dispozici druhému vláknu. Druhé vlákno zpracovává dotazy, které mu doménový server zapisuje do sdílené paměti. Po zpracování dotazu generuje DNS odpověď, která je generována dle aktuálně nastaveného příkazu. Po vygenerování je celá odpověď uložena opět do sdílené paměti.

Popis funkcí programu:

- `threadRead()` - vlákno, které čte a zpracovává zadané příkazy a jejich parametry z příkazové řádky. Parametry jsou zapisovány do sdílených proměnných, které jsou následně použity druhým vláknem při vytváření odpovědí.
- `threadExecute()` - vlákno, které čte dotazy ze sdílené paměti mezi doménovým serverem a C&C serverem. Čtení a zápis do sdílené paměti je řízen pomocí semaforu. Po odemčení semaforu je přečten dotaz ze sdílené paměti a podle příznaku v transakčním ID jsou přečteny data od bota. Následně je dle stavu, ve kterém se program nachází, vygenerována odpověď a uložena do sdílené paměti.
- `createCommandReponse()` - volá funkce pro vytvoření odpovědi s příkazem a jeho parametry.
- `create*Command()` - názvy funkcí jsou odvozeny dle názvu příkazu. Funkce volají funkci createDNSAnswer() pro vytváření odpovědí v přesně specifikovaném pořadí.
- `createDNSAnswer()` - vytvoření jedné odpovědi podle zadaných parametrů.

Po spuštění programu je možné řídit stav pomocí zadávání příkazů a jejich paramterů do příkazové řádky. Ukázka použití programu je na obrázku 9.3. První příkaz `help` vypíše všechny příkazy, další je příkaz `add_domain_name` s parametrem doménového jména, které má být použito jednotlivými boty, pro spojení s C&C serverem. Poslední příkaz je `ddos`, který slouží pro zahájení útoku.

```
>help
        exit - close program
        status - get status
        keepalive_ack - only confirm keepalive from bot
        remove - start removing bots from hosts
        update <url>
        add_domain_name <domain_name>
        remove_domain_name <domain_name>
        set_time - set actual time
        timeout <timeout> - set timeout in sec for bot next request
        traffic_size <size> - size of generated traffic from 1 to 65534
        ddos <target> <time> <duration>
        ampl <spoofedIP> <target> <time> <duration>
        slow_ddos <target> <time> <duration>
>timeout 8
Set timeout '8s' to botnet.
>add_domain_name new-tunel.cz
Add new domain name 'new-tunel.cz' to botnet...
>ddos 45.46.47.48 15:30 1800
Starting of DDoS attack on target '45.46.47.48' at time '15:30' duration '1800'.
>
```

Obrázek 9.3: Program C&C server

Na obrázku 9.4 je zobrazena vygenerovaná odpověď. Jedná se o příkaz pro zahájení amplifikačního útoku s podvrženou IP adresou 78.125.16.123, na doménový server domenovejmeno.cz, čas zahájení útoku je v 15:30 a délka trvání útoku je 1 800 sekund. V první odpovědi CNAME je uložen typ příkazu - `s9`, v následujících čtyřech odpovědích je zakódováno doménové jméno, v další je uložena podvržená IP adresa, poté následuje zakódovaný čas a délka trvání útoku. Poslední dvě adresy nesou spočítaný hash pro typ příkazu a všechny jeho parametry, aby nemohlo dojít k narušení integrity dat.

```
;; ANSWER SECTION:
tunel.cz.               8       IN      CNAME   s9.tunel.cz.
s9.tunel.cz.            8       IN      A       86.248.93.96
s9.tunel.cz.            8       IN      A       136.94.45.247
s9.tunel.cz.            8       IN      A       93.96.232.86
s9.tunel.cz.            8       IN      A       236.11.140.124
s9.tunel.cz.            8       IN      A       78.125.16.123
s9.tunel.cz.            8       IN      A       45.50.7.8
s9.tunel.cz.            8       IN      A       100.81.13.60
s9.tunel.cz.            8       IN      A       100.211.66.149
```

Obrázek 9.4: Ukázka vygenerované odpovědi

9.6 Knihovny

code.h Knihovna poskytuje funkce pro kódování a dekódování řetězců, výpočet a validaci hashe.

- `codeToBuffer()` - kódování 8 bitových znaků na 6 bitů. Každý kódovaný řetězec musí být ukončen znakem '!'. Kódování je možné pouze pro znaky, které se mohou vyskytovat v doménovém jménu. Toto omezení je z důvodu použití kódování pouze pro doménová jména. V případě potřeby je možné množinu kódovaných znaků rozšířit.
- `decodeFromBuffer()` - dekódování z 6 bitových hodnot na 8 bitové.
- `validateHash()` - ověření, zda je vypočítaný hash validní z pohledu anomálního výskytu IP adres, které by se neměly vyskytovat ve veřejné části internetové sítě.
- `hash()` - výpočet hashe algoritmem MD5. Z vypočítané hash hodnoty je použito prvních 6 bytů.

dns_util.h

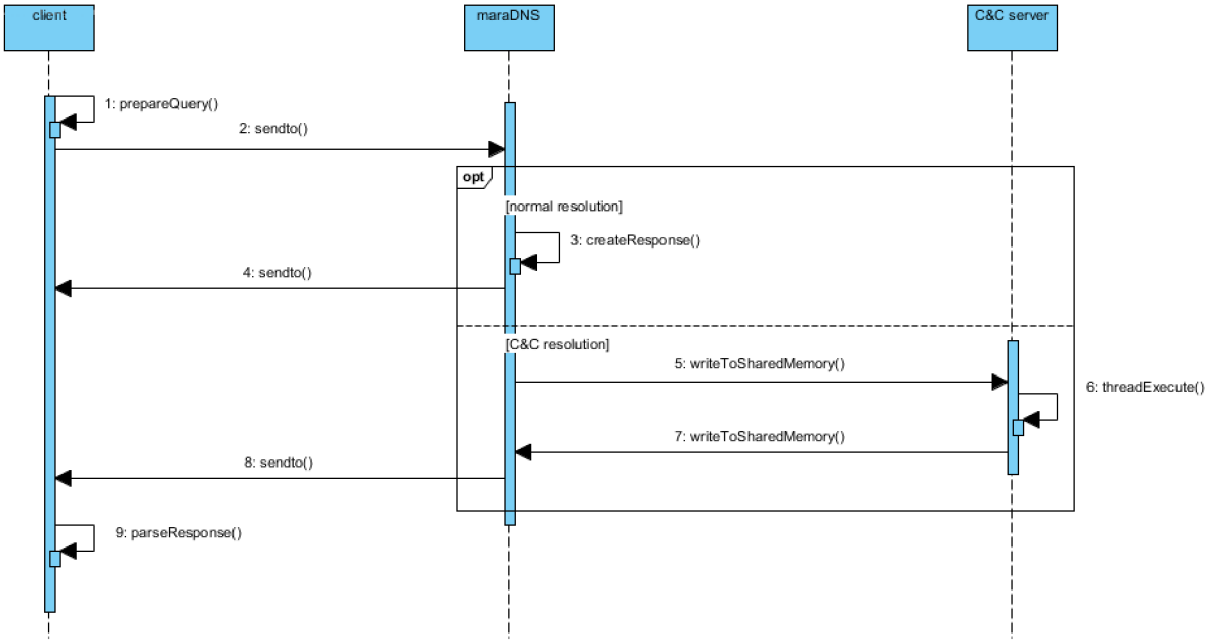
- `changetoDnsNameFormat()` - funkce převádí doménové jméno do formátu, který se používá při přenosu.
- `readName()` - převod z formátu používaného pro přenos doménových jmen v DNS protokolu do původního formátu.

commands.h Definice příkazů a jejich vlastností..

botnet_conf.h Knihovna použitá pro konfiguraci botnetu. Jsou zde konfigurovatelné hodnoty jako offsety použité při ukládání dat do DNS odpovědí nebo klíče pro přístup ke sdílené paměti a semaforům.

9.7 Architektura pro testování

Pro zjednodušení architektury pro testování jsou doménový server a C&C server spuštěny v jednom virtuálním stroji. Oba procesy spolu komunikují přes sdílenou paměť. Program pro klienta je spuštěn na dalším virtuálním stroji. Oba tyto stroje jsou součástí virtuální sítě. Na obrázku 9.5 je uveden případ komunikace programů. Případ začíná vygenerováním DNS dotazu a jeho odeslání klientem. Dotaz obdrží doménový server, který se podle dotazovaného jména rozhodne, jak bude generována odpověď. Pokud jméno patří do množiny jmen zvolených pro tunelování, je dotaz zapsán do sdílené paměti. C&C server poté vygeneruje odpověď, kterou také zapíše do sdílené paměti. Doménový server poté odešle odpověď klientovi, který dekóduje příkaz a vypíše ho.



Obrázek 9.5: Sekvenční diagram pro provedení jedné komunikace

Kapitola 10

Testování, vyhodnocení a možná rozšíření

Tato kapitola popisuje nástroj pro testování navržených obfuskací, dále je uvedeno vyhodnocení naměřených hodnot a jsou uvedena možná rozšíření.

10.1 Nástroj pro testování

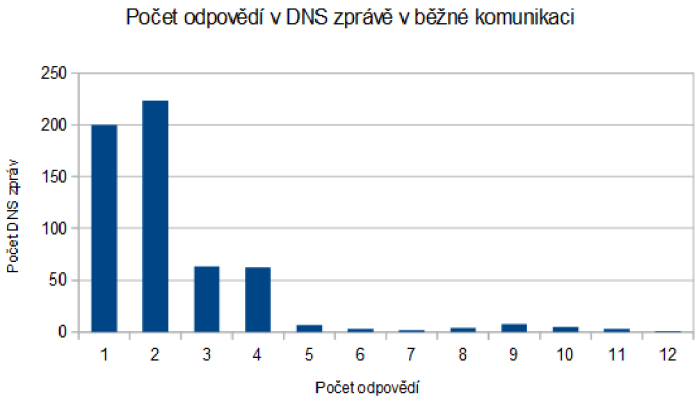
Pro testování byl vyvinut nástroj na analýzu DNS zpráv. Nástroj měří počet a typy odpovědí. Tyto hodnoty jsou zaznamenávány do histogramu. Dále je měřena frekvenční charakteristika znaků doménového jména v každé odpovědi. Poslední měřená vlastnost je navržená detekce obfuskovaného provozu popsaná v kapitole o možných rozšířeních.

10.2 Zhodnocení naměřených výsledků

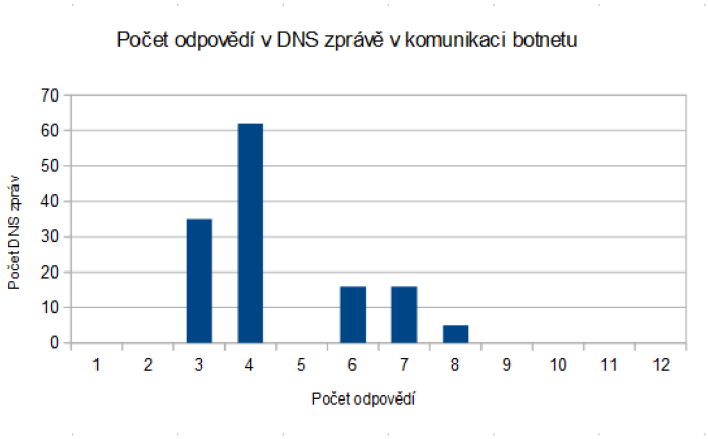
V této sekci jsou vyhodnoceny naměřené výsledky. Nejprve je uvedena analýza počtu odpovědí v DNS zprávách, dále typ těchto odpovědí a nakonec frekvenční analýza doménových jmen.

10.2.1 Měření počtu odpovědí

Množství odpovědí je měřeno pro každou zprávu, která nese odpovědi. Na obrázku 10.1 je zobrazen histogram pro počet odpovědí v DNS zprávách pro legitimní provoz. V legitimním provozu převažuje počet odpovědí od 1 do 4. Zprávy s více odpověďmi jsou méně časté, ale v DNS provozu se vyskytují. Na obrázku 10.2 je zobrazen histogram pro počet odpovědí v DNS zprávách použitých pro C&C botnetu. Nejmenší možný počet odpovědí jsou 3. To vyplývá z nutnosti přenést minimální data, která určují typ příkazu a uložení hashe. Zprávy, které obsahují 4 odpovědi, jsou určené pro konfiguraci bota. Zprávy s více odpověďmi pak obsahují příkazy pro zahájení útoku, a počet odpovědí je určen podle délky kódovaných dat. V legitimním provozu se vyskytují všechny tyto počty odpovědí, které se vyskytují ve zprávách botnetu, a není možné založit detekční metodu na počtu odpovědí, která by vykazovala dobré výsledky.



Obrázek 10.1: Počet odpovědí v DNS zprávách v legitimní komunikaci



Obrázek 10.2: Počet odpovědí v DNS zprávách v komunikaci botnetu

10.2.2 Měření typů odpovědí

Tabulka 10.1 ukazuje typy odpovědí v DNS zprávách. První sloupec tabulky je počet záznamů typů A, druhý sloupec počet záznamů typů CNAME. Třetí a čtvrtý sloupec ukazují výskyty daných kombinací typů záznamů v legitimním provozu a provozu botnetu. V legitimním provozu se používají různé kombinace počtu odpovědí, nejčastější odpověď má jeden záznam typu A a jeden záznam typu CNAME. V provozu botnetu je v odpovědi vždy alespoň jeden záznam typu CNAME a dva záznamy typu A. Do této kombinace jsou kódovány příkazy bez parametrů. Další častou kombinací je odpověď s jedním záznamem CNAME a třemi záznamy A. Tato kombinace je typicky využita pro příkazy s jedním parametrem. Třetí nejčastější kombinací je odpověď s jedním záznamem CNAME a několika zznamy A, kterých je více než tři. To jsou odpovědi v kterých jsou zakódovány příkazy pro zahájení útoku. Z tabulky vyplývá, že není možné založit metodu na detekci podle typu odpovědí.

A	CNAME	legitimní provoz	provoz botnetu
1	0	64	0
1	1	217	0
1	2	27	0
1	4	13	0
2	0	29	0
2	1	12	68
2	2	14	42
3	0	15	0
3	1	8	35
4	0	16	0
4	1	4	10
5	1	2	12
7	1	2	9
8	1	23	8
9	1	21	12

Tabuľka 10.1: Výskyt typů odpovědí v legitimním provozu a provozu botnetu

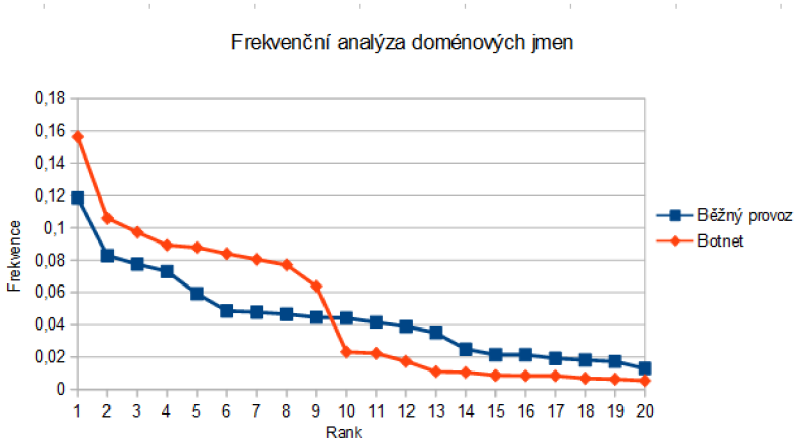
10.2.3 Frekvenční charakteristika znaků v doménovém jménu

Rank	legitimní provoz	frekvence	provoz botnetu	frekvence
1	.	0,12	.	0,16
2	e	0,08	e	0,11
3	c	0,08	n	0,1
4	o	0,07	c	0,09
5	s	0,06	z	0,09
6	a	0,05	t	0,08
7	t	0,05	l	0,08
8	g	0,05	u	0,08
9	i	0,04	s	0,06
10	l	0,04	9	0,02
11	n	0,04	a	0,02
12	m	0,04	o	0,02
13	d	0,03	6	0,01
14	-	0,02	m	0,01
15	w	0,02	d	0,01
16	z	0,02	k	0,01
17	b	0,02	j	0,01
18	r	0,02	0	0,01
19	u	0,02	7	0,01
20	1	0,01	1	0,01

Tabuľka 10.2: Přidělení ranků jednotlivým znakům

Při měření frekvenční charakteristiky znaků v doménových jménech bylo měřeno jméno, kterým začíná každý záznam v sekci s odpověďmi. Pro legitimní provoz a provoz botnetu byl pro každý znak spočítán rank, tak jak ukazuje tabulka 10.2. V této tabulce je v prvním sloupci hodnota ranku, který je dle nejčastějšího výskytu znaku. V druhém a třetím sloupci je znak a jeho frekvence výskytu v legitimním provozu, ve čtvrtém a pátém sloupci je pak znak a jeho frekvence výskytu v provozu botnetu.

Naměřená data z výše uvedené tabulky jsou vynesena do grafu na obrázku 10.3. Z grafu je vidět podobnost frekvenčních charakteristik legitimního provozu a provozu botnetu.



Obrázek 10.3: Frekvenční charakteristika ligitimního a provozu botnetu

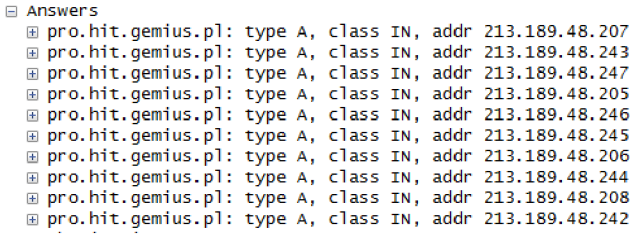
10.3 Rozšíření

V této kapitole jsou popsána rozšíření navržených obfuskací a rozšíření umožňující přenos více dat.

10.3.1 Obfuskace odchylky IP adres

Při ukládání zakódovaných dat a hodnoty hashe do IP adres, mají tyto adresy mezi sebou velké odchylky. V legitimním provozu se však IP adresy liší pouze v několika nejméně významných bitech, jak je ukázáno na obrázku 10.4. Na základě těchto odchylek je možné založit detekční metodu. V rámci testování navržených obfuskací byla tato odchylka změřena na experimentálním vzorku testovacích dat z legitimního provozu a na datech z provozu botnetu. Odchylka byla meřena podle počtu nejméně významných bitů, ve kterých se IP adresy liší. Tato metoda se ukázala jako nadostatečná, jelikož i v legitimním provozu mohou být adresy, které jsou podstatně odlišné. Reálná metoda detekce by nemohla být založena pouze na jednoduché detekci odchylek, ale musela by poskytovat komplexenější analýzu IP adres. Výsledky navržené metody jsou v příloze B.

V rámci rozšíření obfuskací může být navržen algoritmus pro kódování dat do IP adres, který by generoval IP adresy, tak jak ve skutečnosti vypadají v reálném provozu. Při návrhu vylepšení obfuskací je možné pracovat s rozšířeními popsanými v následujících dvou kapitolách.



Obrázek 10.4: Ukázka odchylek IP adres legitimního provozu

10.3.2 Použití IPv6 adres

Nejrozšířenějším internetovým protokolem je v současnosti protokol IPv4. Pro překlad doménových jmen na IPv4 adresy jsou použity DNS záznamy typu A. Nástupcem tohoto protokolu je protokol IPv6. Pro překlad doménových jmen na adresy protokolu IPv6 slouží záznamy typu AAAA. Se vzrůstajícím použitím protokolu IPv6 bude přibývat i využití AAAA záznamů a bude možné využívat tyto záznamy pro obfuskování C&C komunikace. Velikost IPv6 je 128 bitů a umožňují zakódování podstatně více dat než 32bitové IPv4 adresy.

10.3.3 Využití odpovědí v Authority a Additional sekcích

Pro další možné zakódování přenášených dat by bylo možné využít záznam v sekcích *Authority* a *Additional*. V sekci *Authority* se přenášejí záznamy typu NS[1], které distribuují informace o autoritativních doménových serverech. V sekci Additional jsou odpovědi, které nejsou přímou odpovědí na dotaz, ale s dotazem souvisejí. V této sekci jsou převážně odpovědi typu A, AAAA nebo CNAME. Všechny tyto odpovědi lze použít pro přenos zakódovaných dat.

[1]Name Server

Kapitola 11

Závěr

Cílem práce bylo nastudovat síťové anomálie a bezpečnostní incidenty v DNS provozu. Dále analyzovat metody detekce těchto anomálií a bezpečnostních incidentů. Dalším krokem byl návrh obfuskačních technik analyzovaných metod a implementace těchto technik formou automatizovaného nástroje. Posledním krokem práce bylo zhodnocení navržených technik a návrh případných rozšíření.

V rámci studia síťových anomálií a bezpečnostních incidentů byly identifikovány dvě skupiny útoků. První skupinou jsou útoky na samotnou službu DNS. Mezi tyto útoky patří různé varianty útoku DDoS. Druhou skupinou jsou útoky využívající DNS službu pro vedení jiných útoků. Do druhé skupiny patří DNS Cache Poisoning, DNS Amplification, Domain Fast Flux a tunelování přes DNS protokol. Dále jsou uvedeny metody detekce popsaných útoků, kde největší prostor je věnován DNS tunelování. U tunelování je popis metod detekce rozdělen na detekci založenou na analýze payloadu zpráv a na analýze síťové komunikace. Metody detekce payload dat jsou založeny především na frekvenční charakteristice a regulárních výrazech zkoumající data přenášená v TXT záznamech. Na problematiku DNS tunelování se zaměřuje zbytek celé práce, konkrétně na jeho využití pro řízení a kontrolu botnetu. Z tohoto důvodu je nastíněna infrastruktura botnetu a jeho typická činnost, a je navržen protokol pro C&C botnetu. V další kapitole jsou navrženy obfuskace detekce DNS tunelování. Obfuskace jsou zaměřeny především na detekci pomocí frekvenční analýzy, dodržení obvyklých hodnot v DNS protokolu a obfuskací založených na měření množství provozu. Obfuskace frekvenční analýzy je řešena pomocí kódování dat do IP adres, jelikož současné metody se zaměřují pouze na doménová jména a DNS záznamy typu TXT. Dále je uvedeno kódování C&C protokolu a přenášených dat do DNS zpráv. Nejdříve jsou identifikovány pole DNS zpráv vhodná pro přenos dat a následně je popsáno kódování jednotlivých příkazů. Následující část práce popisuje implementaci automatizovaného nástroje a architekturu při testování. Architektura je složena ze tří programů, které jsou C&C server, doménový server a klient. Každý program je popsán a jsou uvedeny příklady použití. Poslední kapitola práce je zaměřena na testování, vyhodnocení a návrh rozšíření. V rámci testování a vyhodnocení byly měřeny tři vlastnosti. První vlastností je počet odpovědí, druhou je typ těchto odpovědí. Třetí vlastností je frekvenční charakteristika doménových jmen. Zhodnocení naměřených výsledků ukazuje odolnost navržených obfuskací vůči uvedeným způsobům detekce.V rámci rozšíření jsou navrženy možnosti pro rozšíření obfuskací. První z nich spočívá v obfuskaci odchylek v IP adresách, kde je i popsána možnost detekční metody. Další návrhy jsou využití IPv6 adres a využití sekcí Authority a Additional v DNS zprávě.

V rámci práce byly navrženy obfuskace metod detekce DNS tunelování, které spočívají

především ve využití nejčastěji se vyskytujících typů záznamů v legitimním DNS provozu. Použité typy jsou A a CNAME. Data se kódují do IPv4 adres (resp. adresy) tak, aby nepůsobily anomálně ve veřejné internetové síti. V záznamech CNAME se přenášejí pouze doménová jména a krátké úseky přenášených dat tak, aby doménová jména byla odolná vůči detekci pomocí frekvenční analýzy. Další důležitou částí je návrh architektury botnetu, který má vliv na obfuskaci metod založených na traffic analýze. Pomocí implementace, testování a vyhodnocení navržených obfuskačních technik byla prokázána nutnost dalšího hledání univerzálnějších způsobů detekce, zejména detekce DNS tunelování.

Literatura

[1] Bootnet Statistic. [online], [cit. 12.1.2016].
URL <https://www.abuse.ch/?p=3294>

[2] Open Resolve Project. [online], [cit. 12.1.2016].
URL <http://www.openresolverproject.org/>

[3] Trojan.Katusha. [online], [cit. 12.3.2016].
URL <http://www.enigmasoftware.com/trojankatusha-removal/>

[4] Win32/Morto - Made in China. [online], [cit. 12.3.2016].
URL <http://www.welivesecurity.com/2012/11/14/win32morto-made-in-china/>

[5] MaraDNS. [online], [cit. 12.5.2016].
URL <http://maradns.samiam.org/>

[6] Oracle VM VirtualBox. [online], [cit. 12.5.2016].
URL <https://www.virtualbox.org/>

[7] Domain Names - Implemantation and Specification. [online], [cit. 16.12.2015].
URL <http://www.ietf.org/rfc/rfc1035>

[8] Domain Name System (DNS) Denial of Service (DoS) Attacks. [online], [cit. 17.12.2015].
URL <http://attrition.org/security/advisory/ciac/j-fy99/ciac.j-063.dns-dos>

[9] Dynamic Updates in the Domain Name System. [online], [cit. 18.12.2015].
URL <http://www.ietf.org/rfc/rfc2136>

[10] Extension Mechanisms for DNS. [online], [cit. 18.12.2015].
URL <http://www.ietf.org/rfc/rfc2671>

[11] DNScat. [online], [cit. 2.12.2015].
URL <https://wiki.skullsecurity.org/Dnscat>

[12] Domain Name System Security Extensions. [online], [cit. 5.1.2016].
URL <http://www.ietf.org/rfc/rfc2065>

[13] Iodine. [online], [cit. 7.1.2016].
URL <http://code.kryo.se/iodine/>

[14] Tcp-over-DNS. [online], [cit. 7.1.2016].
URL <http://analogbit.com/software/tcp-over-dns/>

[15] Born, K.; Gustafson, D.: Detecting dns tunnels using character frequency analysis. *arXiv preprint arXiv:1004.4358*, 2010.

[16] Born, K.; Gustafson, D.: Ngviz: detecting dns tunnels through n-gram visualization and quantitative analysis. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, ACM, 2010, str. 47.

[17] Cambiaso, E.; Papaleo, G.; Chiola, G.; aj.: Slow DoS attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications*, ročník 1, č. 3-4, 2013: s. 300–319.

[18] Chen, C.-M.; Cheng, S.-T.; Chou, J.-H.; aj.: Formulistic Detection of Malicious Fast-Flux Domains. In *Parallel Architectures, Algorithms and Programming (PAAP), 2012 Fifth International Symposium on*, IEEE, 2012, s. 72–79.

[19] Dietrich, C. J.; Rossow, C.; Freiling, F. C.; aj.: On Botnets that use DNS for Command and Control. In *2011 Seventh European Conference on Computer Network Defense*, IEEE, 2011, s. 9–16.

[20] Ellens, W.; Żuraniewski, P.; Sperotto, A.; aj.: *Flow-based detection of DNS tunnels.* Springer, 2013.

[21] Farnham, G.: Detecting DNS tunneling. *InfoSec Reading Room*, 2013.

[22] Ichise, H.; Jin, Y.; Iida, K.: Analysis of via-resolver DNS TXT queries and detection possibility of botnet communications. In *Communications, Computers and Signal Processing (PACRIM), 2015 IEEE Pacific Rim Conference on*, IEEE, 2015, s. 216–221.

[23] Jin, Y.; Ichise, H.; Iida, K.: Design of Detecting Botnet Communication by Monitoring Direct Outbound DNS Queries. In *Cyber Security and Cloud Computing (CSCloud), 2015 IEEE 2nd International Conference on*, IEEE, 2015, s. 37–41.

[24] Kabelová, A.; Dostálek, L.: *Velký průvodce protokoly TCP/IP a systémem DNS*, ročník 3. Computer press, 2008, 1012 s.

[25] Kambourakis, G.; Moschos, T.; Geneiatakis, D.; aj.: Detecting DNS amplification attacks. In *Critical information infrastructures security*, Springer, 2007, s. 185–196.

[26] Kaminsky, D.: Black ops 2008: It's the end of the cache as we know it. *Black Hat USA*, 2008.

[27] Karasaridis, A.; Meier-Hellstern, K.; Hoeflin, D.: NIS04-2: Detection of DNS Anomalies using Flow Data Analysis. In *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*, IEEE, 2006, s. 1–6.

[28] Kingsley, Z. G.: Selective Studies and the Principle of Relative Frequency in Language. 1932.

[29] Kováčik, M.: Detekcia sietových anomálií a bezpečnostných incidentov s vyuzitím DNS dát. In *Sborník příspěvků PAD 2014*, Technická univerzita v Liberci, August 2014, ISBN 978-80-7494-027-9, str. 174.

[30] Olzak, T.: Dns cache poisoning: Definition and prevention. 2006.

[31] Qi, C.; Chen, X.; Xu, C.; aj.: A bigram based real time DNS tunnel detection approach. *Procedia Computer Science*, ročník 17, 2013: s. 852–860.

[32] Riden, J.: Know your Enemy: fast-flux service networks. *The Honeynet Project*, 2008.

[33] Roolvink, S.: Detecting attacks involving DNS servers: a netflow data based approach. 2008.

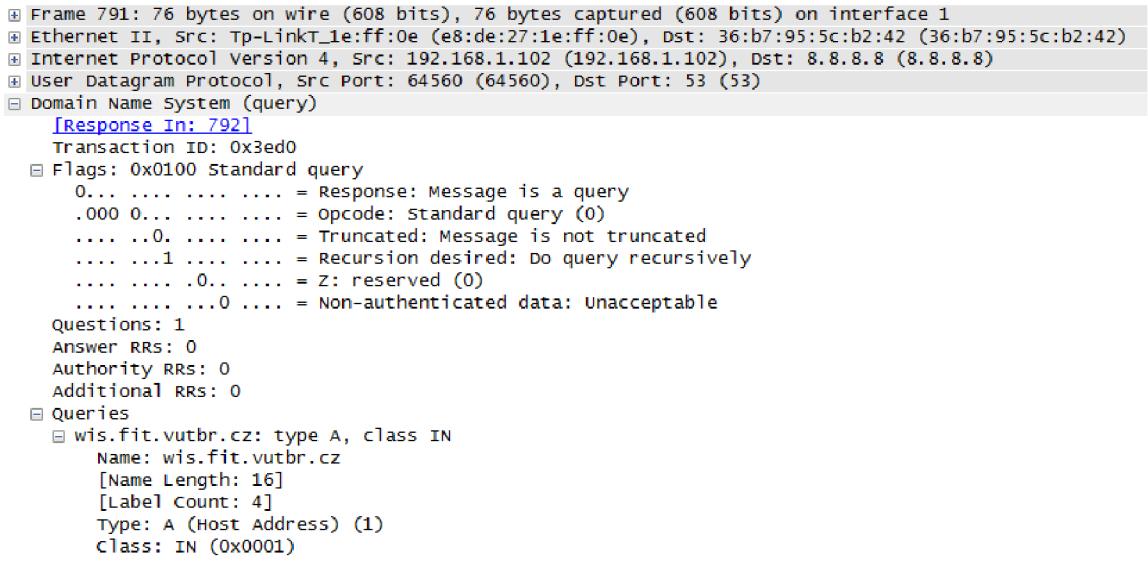
[34] Vaughn, R.; Evron, G.: DNS amplification attacks. *Go online to http://www. isotf. org/news/DNS-Amplification-Attacks. pdf*, 2006.

[35] Villamarín-Salomón, R.; Brustoloni, J. C.: Identifying botnets using anomaly detection techniques applied to DNS traffic. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, IEEE, 2008, s. 476–481.

Příloha A

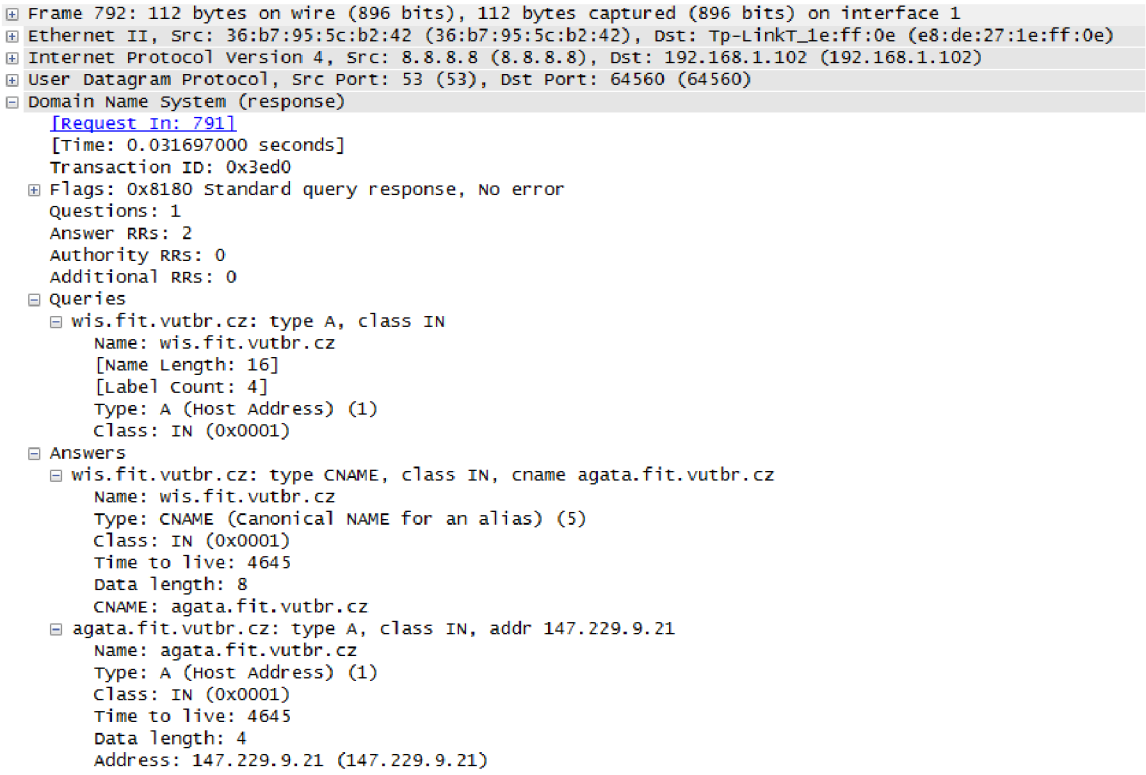
DNS dotaz a odpověď

A.1 DNS dotaz



Obr. A.1: Příklad DNS dotazu.

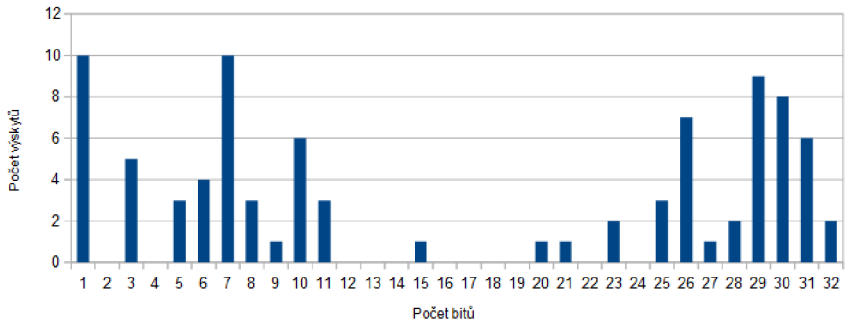
A.2 DNS odpověď



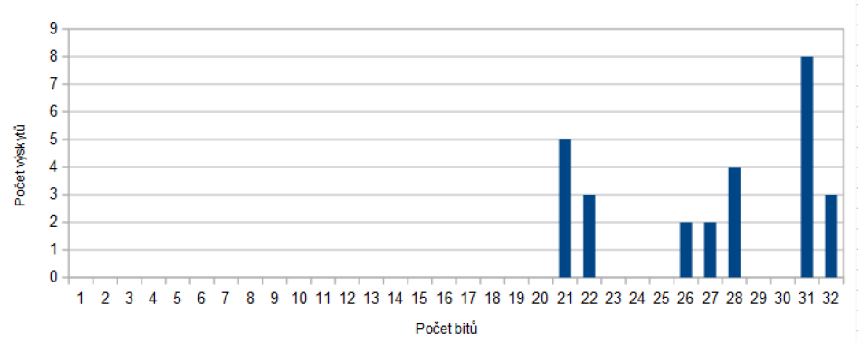
Obr. A.2: Příklad DNS odpovědi.

Příloha B

Grafy metody založené na detekci odchylek IP adres



Obr. B.1: Odchylky IP adres v legitimním DNS provozu



Obr. B.2: Odchylky IP adres v provozu botnetu

Příloha C

Obsah CD

- **doc** - zdrojové kódy LaTeX-u a lyx-u textové části diplomové práce,
- **src** - zdrojové kódy automatického nástroje.