VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

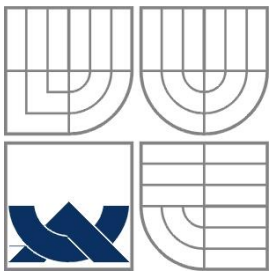
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GENEROVÁNÍ ÚTOKŮ NA SLUŽBU DNS

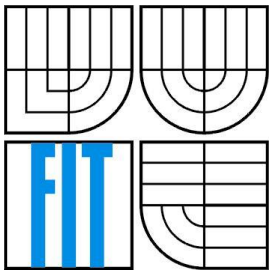
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB TUTKO
AUTHOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GENEROVÁNÍ ÚTOKŮ NA SLUŽBU DNS

GENERATING OF DNS SERVICE ATTACKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB TUTKO

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL KOVÁČIK

BRNO 2015

Abstrakt

Dostupnost a bezpečnost patří mezi nejdůležitější požadavky internetových služeb. Proto je nezbytné detekovat síťové anomálie, které mají na tyto potřeby největší vliv. Aby detekční mechanismy držely krok se stále propracovanějšími anomáliemi, musí být na jejich vývoj kladen velký důraz. Cílem této práce je analýza a replikování nejčastějších útoků na službu DNS. Sesbíraná data z těchto generátorů útoků mohou byt použitá pro lepší pochopení jejich chování, co vede ke tvorbě kvalitnějších detekčních mechanizmů.

Abstract

Availability and security are the most important requirements for Internet services. It is therefore necessary to detect the network anomalies, which have the greatest impact on these requirements. Therefore the great emphasis must be placed on development of detection mechanisms that keep pace with increasingly sophisticated network anomalies. The aim of this work is to analyze and replicate the most common DNS service attacks. Collected data from these attack generators can be used for better understanding of their behavior, what leads to improved and more effective detection mechanisms.

Klíčová slova

DNS útoky, generování útoků, analýza útoků, reflexívní DNS útok, cache poisoning, DNS tunelování, škodlivé domény

Keywords

DNS service attacks, generating attacks, analysis of attacks, DNS reflection attack, cache poisoning, DNS tunneling, malicious domain

Citace

Jakub Tutko: Generování útoků na službu DNS, bakalářská práce, Brno, FIT VUT v Brně, 2015

Generování útoků na službu DNS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Kováčika. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jakub Tutko
20.05.2015

Poděkování

Chcel by som sa poďakovať svojmu vedúcemu Ing. Michalovi Kováčikovi za rady pri písaní tejto práce. Vďaka patrí taktiež rodine, priateľke a kamarátom za ich podporu.

Obsah

1 Úvod

Rozvoj počítačových sietí so sebou prináša stále prísnejšie potreby pre ich správu. Hlavný dôraz sa kladie najmä na dostupnosť a bezpečnosť služieb internetu. Cez sieť sa posiela čoraz viacej citlivých informácií a aby sa užívatelia nemuseli báť, že by sa tieto informácie dostali do nesprávnych rúk, správcovia sietí musia dbať na ich zabezpečenie. Rozvoj sietí totiž zapríčiňuje stále častejšie výskyty internetových útokov. Tieto útoky sú navyše stále viac sofistikovanejšie a deštruktívnejšie.

Jednou z dnešných najpoužívanejších služieb internetu je práve služba DNS. Túto službu využívajú takmer všetky dnešné sieťové aplikácie a aj viacero iných služieb. Tohto trendu sú si vedomí aj útočníci, preto sú ich útoky vedené buď priamo na túto službu alebo ju využívajú pre lepšiu efektivitu svojich útokov.

Aby sa správcovia sietí vedeli proti týmto útokom brániť, musia najskôr tieto útoky čo najpodrobnejšie analyzovať a podľa zozbieraných informácií implementovať čo najúčinnejšie detekčné mechanizmy, ktoré by útoky včas odhalili a zabránili im. Momentálne však nie je jednoduché dostať sa ku kvalitným dátam z jednotlivých útokov a čakať na reálny útok, aby sa tieto mechanizmy otestovali nie je ideálnym spôsobom. Preto vývoj kvalitných detekčných mechanizmov v sebe musí zahŕňať aj vývoj programov, ktoré by útoky rekonštruovali. Vďaka kvalitným generátorom útokov je jednoduchšie pochopiť správanie sa jednotlivých anomálií a otestovať zraniteľnosť sietí.

Takýmito generátormi útokov sa zaoberá aj táto práca. Práca obsahuje jednoduchý popis fungovania služby DNS, analýzu najčastejších útokov na túto službu a popis ich implementácie.

Na začiatku práce v [kapitola 2] je čitateľ oboznámený so základným popisom služby DNS, so zabezpečením a zraniteľnosťou tejto služby a sú mu vysvetlené princípy najbežnejších útokov na túto službu. V [kapitola 3] sú jednotlivé útoky analyzované. Je tam vysvetlené ako vyzerajú pakety týchto útokov a podrobnejšie sa vysvetľuje ich chovanie. Implementácia útokov je rozobraná v [kapitola 4]. V tejto kapitole je popísané prostredie, ktoré bolo použité na testovanie, konfiguračný súbor, vďaka ktorému je možné meniť chovanie programov, výstup programov a problémy pri implementovaní jednotlivých programov.

2 Služba DNS

Pre plnú funkčnosť internetu je potrebné, aby na ňom bežalo viacero služieb. Sú to služby, ktoré nám umožňujú posielať elektronickú poštu, prenášať súbory medzi počítačmi, prehľadávať internetové stránky a mnoho ďalších služieb. Jednou z takto potrebných služieb je aj služba DNS[1], ktorá beží na DNS servery. Bez tejto služby by však internet mohol existovať, ale značne by znepríjemňoval užívateľom jeho používanie a nemohol by byť dnes takto rozšírený.

Základnou úlohou DNS služby je mapovanie doménových adries na IP adresy[2]. Bez doménových adries by si užívatelia museli pamätať IP adresy všetkých služieb na internete, ku ktorým by chceli pristupovať, čo by bolo pre väčšinu prakticky nemožné. Väčší problém by navyše nastal pri zavedení IPv6 adries, kde by už bol problém zapamätať si jednu konkrétnu adresu a všetky adresy by si užívatelia museli niekde zapisovať a mať ich stále pri sebe. Naopak, ak by sieťové zariadenia mali pracovať s textovými reťazcami značne by ich to zaťažovalo. Pre tieto dôvody bolo nevyhnutné zaviesť službu DNS.

Služba DNS obsahuje databázu všetkých registrovaných doménových mien. Keďže táto databáza je veľmi rozsiahla, je distribuovaná na viacero DNS serverov. K týmto serverom sa potom pristupuje pomocou DNS dotazu.

Systém DNS využívajú všetky aplikačné protokoly. Pri spustení aplikácie, ktorá tieto protokoly využíva sa najprv posiela dotaz na DNS server a až po odpovedi sa aplikácia pripája na cielený server.

2.1 DNS server

DNS server je zariadenie, ktoré obsahuje dáta z priestoru doménových mien. Tento priestor mien je uložený ako množina záznamov na DNS serveroch. Priestor sa navyše rozdeľuje na zóny a jednotlivé zóny sú potom uložené na rôznych DNS serveroch. Jeden DNS server však môže spracovávať aj viacero zón.

Ak príde dotaz na DNS server a server obsahuje hľadaný záznam vo svojom lokálnom súbore, vráti ho ako odpoveď, ak záznam neobsahuje, načíta ho z DNS serveru spravujúci zónu, kde sa hľadaná adresa nachádza. Informácie, ktoré DNS server spracováva, a za ktoré je zodpovedný sa nazývajú autoritatívne. DNS servery sa delia na tieto skupiny:

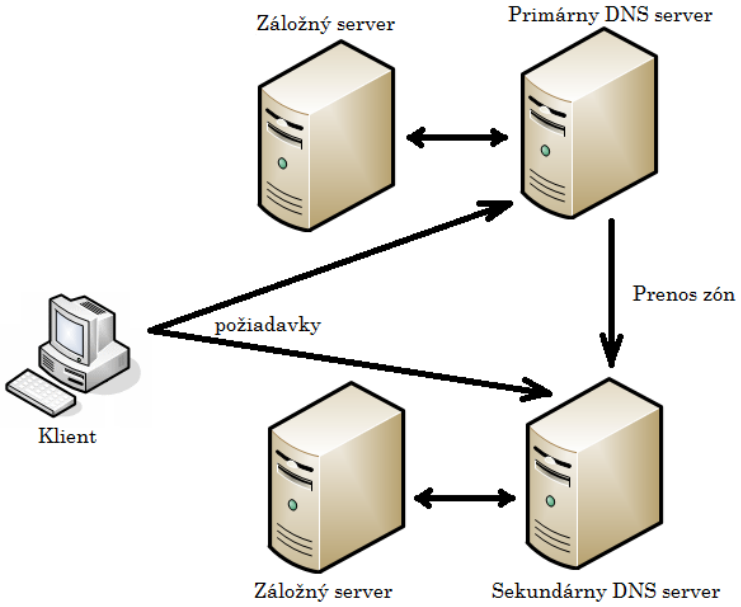
[1] Domain Name Server – Systém názvov domén

[2] napríklad doménová adresa fit.vutbr.cz je mapovaná na adresu 147.229.9.23

- **Primárny DNS server** – obsahuje úplné záznamy o doménach. Jeho odpovede pre tieto domény sú vždy autoritatívne. Pre každú doménu existuje práve jeden primárny server.

- **Sekundárny DNS server** – server získava dáta od primárneho servera a ukladá si ich. Musí udržovať aktuálnosť všetkých získaných dát. Jeho odpovede sú taktiež autoritatívne.

- **Záložný DNS server (cache server)** – server slúži na zrýchlenie procesu rezolúcie doménového mena. Prijatý dotaz preposiela ďalej serverom DNS. Prijatú odpoveď si potom ukladá a pri ďalších dotazoch na tento záznam vracia takto uložený záznam. Jeho odpovede sú však neautoritatívne.



Obr. 2.1.1 : Typy DNS serverov

Platnosť záznamov na sekundárnych a záložných serveroch býva obmedzená. Doba, po ktorú je záznam platný je uložená priamo v každom zázname. Po vypršaní tejto hodnoty musí server buď, záznam vymazať alebo požiadať primárny server o aktualizáciu.

S touto platnosťou záznamov musí rátať aj primárny DNS server. Ak aktualizuje svoje záznamy, tak sa tieto zmeny môžu preniesť do sekundárnych serverov až o niekoľko hodín neskôr. Je to spôsobené tým, že primárny server neinformuje sekundárne servery o zmenách, ale sekundárne

servery sa po istej dobe pýtajú primárneho či k nejakým zmenám došlo. A ak by sa sekundárne servery pýtali na zmeny príliš často, značne by to zaťažovalo primárny server.

2.2 DNS záznamy

Pre ukladanie informácií v dátovom priestore DNS slúžia záznamy DNS. Záznamy sa ukladajú v textovej podobe v zónových súboroch na serveroch DNS. Každý záznam musí dodržovať formát definovaný štandardom RFC 1035 [6]. Záznam obsahuje meno, typ, triedu, ttl[3] a dĺžku záznamu, za ktorými nasledujú informácie, ktoré sa líšia podľa typu záznamu. Najbežnejšie záznamy sú:

- **SOA záznamy -** záznam obsahuje informácie o autoritatívnych dátach pre danú zónu. Každá zóna má práve jeden typ tohto záznamu. Záznam obsahuje meno primárneho serveru DNS pre danú doménu, kontakt na správcu domény (emailovú adresu), sériové číslo pre identifikáciu zmien záznamu a ďalšie informácie pre iné DNS servery.

```
$TTL 3600
        @ IN SOA isa.fit.vutbr.cz. root.isa.fit.vutbr.cz.(
                2007110901 ; Serial
                3600 ; Refresh
                900 ; Retry
                3600000 ; Expire
                3600 ) ; Minimum
```

- **NS záznam** – záznam obsahuje autoritatívne servery pre danú doménu. Autoritatívny server obsahuje platné záznamy pre danú doménu.

```
fit.vutbr.cz.    IN NS gate.feec.vutbr.cz
                 IN NS guta.fit.vutbr.cz
                 IN NS kazi.fit.vutbr.cz
                 IN NS rhino.cis.vutbr.cz
```

[3] Time to live – parameter DNS záznamu, ktorý obsahuje dobu platnosti záznamu

- **A záznam** – základný záznam, ktorý obsahuje preklad doménovej adresy na IP adresu.

  eva.fit.vutbr.cz. IN A 147.229.176.14

- **AAAA záznam** – záznam podobný záznamu typu A, s tým rozdielom, že prekladá doménové mená na IPv6 adresy.

  isa.fit.vutbr.cz. 14400 IN AAAA 2001:67c:1220:8b0::93e5:b012
  www.cesnet.cz. 85443 IN AAAA 2001:718:1:101::4

- **MX záznam** – obsahuje informácie o poštovom serveri danej domény. Doména môže obsahovať viacero poštových serverov a je možné im priradiť rôzne priority.

  fit.vutbr.cz. IN MX 10 kazi.fit.vutbr.cz
  IN MX 20 eva.fit.vutbr.cz

- **CNAME záznam** – keďže pre jeden počítač môže byť vytvorených viacero doménových adries, tento záznam nám oznámi kanonické meno daného počítača.

  www IN CNAME tereza.fit.vutbr.cz
  ldap IN CNAME tereza.fit.vutbr.cz

- **PTR záznam** – tento záznam prevádza opačné mapovanie adries. To znamená, že prevádza IP adresy na doménové adresy.

  14.176.229.147.in-addr.arpa. IN PTR eva.fit.vutbr.cz

- **TXT záznam** – ukladá v sebe dodatočné textové informácie o hľadanej doméne.

  fi.muni.cz. 1773 IN TXT ‘‘Masaryk University Brno’’
  fi.muni.cz. 1773 IN TXT ‘‘Botanicka 68a, 602 00, Brno, Czech republic’’
  fi.muni.cz. 1773 IN TXT ‘‘Faculty of Informatics’’

- **SRV záznam** – záznam obsahuje informácie o službách, ktoré bežia na hľadanom počítači.

```
_http._tcp.www.hello.edu.      IN SRV 0 2 80 www.hello.edu
                               IN SRV 0 1 80 www2.hello.edu
                               IN SRV 1 1 8000 backup.hello.edu
_gopher._tcp.hello.edu         IN SRV 0 0 0
```

- **NAPTR záznam** – slúži na mapovanie textových reťazcov na dáta. Môže napríklad prekladať telefónne čísla. Tento typ záznamu môže obsahovať regulárne výrazy.

```
$ORIGIN 4.1.1.4.5.0.2.4.e164.arpa.      IN NAPTR 10 50 "u"
                                        "E2U+sip" "!^\\+(.*)$!sip:\\1@vutbr.cz!"
```

2.3 Zabezpečenie DNS

Služba DNS je verejne dostupná, preto vznikli mechanizmy ako ju dostatočne zabezpečiť. Na zabezpečenie tejto služby slúži napríklad mechanizmus TSIG alebo štandard DNSSEC.

TSIG je mechanizmus, ktorý popisuje autentizáciu transakcií služby DNS pomocou symetrickej kryptografie. Mechanizmus pridáva nový typ záznamu, záznam TSIG, ktorý obsahuje autorizačný kód DNS záznamu a taktiež názov algoritmu, čas podpisu a ďalšie položky. Toto zabezpečenie umožňuje bezpečne prenášať záznamy medzi DNS servermi bez toho, aby nejaký útočník tieto záznamy podvrhol. Klienti tento mechanizmus nevyužívajú kvôli problémom s distribúciou kľúču.

Štandard DNSSEC vznikol v roku 2005 a definuje rozšírenie protokolu DNS pre zabezpečený prenos dát v systéme DNS s použitím verejného a súkromného kľúča a asymetrickej kryptografie. DNSSEC pridáva do DNS ďalšie typy záznamov: záznam DNSKEY pre ukladanie verejného kľúča, záznam RRSIG obsahujúci podpis záznamu, záznam NSEC na zreťazenie záznamov a DS záznam, ktorý podpisuje DNSKEY záznam.

Vďaka týmto mechanizmom je možné dostatočne zabezpečiť systém DNS, no stále sa nájdu cesty ako tieto prekážky prekonať. Navyše nie všetky služby môžu byť dostatočné zabezpečené. Preto táto služba bude naďalej istým bezpečnostným rizikom pre užívateľov.

2.4 Zraniteľnosť DNS

Služba DNS je využívaná každodenne takmer všetkými užívateľmi internetu, preto býva častou voľbou pre útoky. Je verejná, čiže s ňou môže pracovať ľubovoľný užívateľ a cieľový užívateľ ju považuje za dôveryhodnú, čo sú hlavné výhody pre útočníkov.

Útoky môžu byť zamerané priamo na službu DNS alebo môžu len využívať vo svoj prospech jej správanie. Ak útočník ochromí nejaký frekventovaný DNS server spolu s jeho zálohami, môže znemožniť prístup k internetu pre obrovské množstvo užívateľov.

Časté útoky, ktoré nebývajú zamerané priamo na službu DNS využívajú práve správanie tejto služby, ktorá poskytne odpovede každému, kto na ňu zašle dotaz. Navyše ak je dotaz zaslaný cielene, je možné dostať niekoľko násobne väčšiu odpoveď. Vďaka serverom DNS môžu útočníci svoj útok niekoľko násobne zosilniť a navyše docieliť väčšej anonymity.

Nebezpečnou súčasťou DNS služby zvyknú byť ich cache pamäte. Ak nie sú cache pamäte dostatočne zabezpečené, sú jednoduchým cieľom útočníka. Ak sú im podvrhnuté nejaké falošné záznamy, tak užívateľ bez toho, aby si niečo všimol, navštevuje stránky, o ktoré nemal vôbec záujem a v horšom prípade im poskytne aj svoje citlivé informácie ako napríklad prihlasovacie údaje alebo čísla účtov.

Ku väčšine útokov sa používajú nejaké náhodne vygenerované domény, aby útok mohol prebiehať aj po zablokovaní tejto domény. Útočný program si po zablokovaní tejto domény vygeneruje doménu novú. O tomto generovaní je viac popísané v kapitole 3.4.

Existuje mnoho ďalších útokov využívajúcich túto službu, niektoré sú viac, iné menej nebezpečné no všetkým je potrebné sa vyhýbať. Tie najbežnejšie útoky budú podrobnejšie prebrané ďalej v texte.

2.4.1 Reflexívny DNS útok

Medzi najbežnejšie útoky na internete patria DoS[4] útoky. Sú to útoky, ktoré sa snažia o ochromenie určitej služby na internete. Na službu začnú posielať také veľké množstvo dotazov, že služba už nie je schopná odpovedať iným užívateľom, prípadne službe dôjdu zdroje a skolabuje. Takémuto útoku je však ľahké sa ubrániť. Stačí obmedziť počet dotazov z jednej IP adresy za nejaký čas. Preto vznikol DDoS[5] útok, kde už dotazy neposiela len jeden počítač ale útočník na zosilnenie útoku využíva predom infikované počítače, ktoré má pod svojou kontrolou, takzvané botnety. Takýto útok sa už ťažko odhaľuje, a navyše je lepšie zachovaná útočníkova anonymita.

[4] Denial of Service – odoprenie služby

[5] Distributed Denial of Service – distribuované odopretie služby.

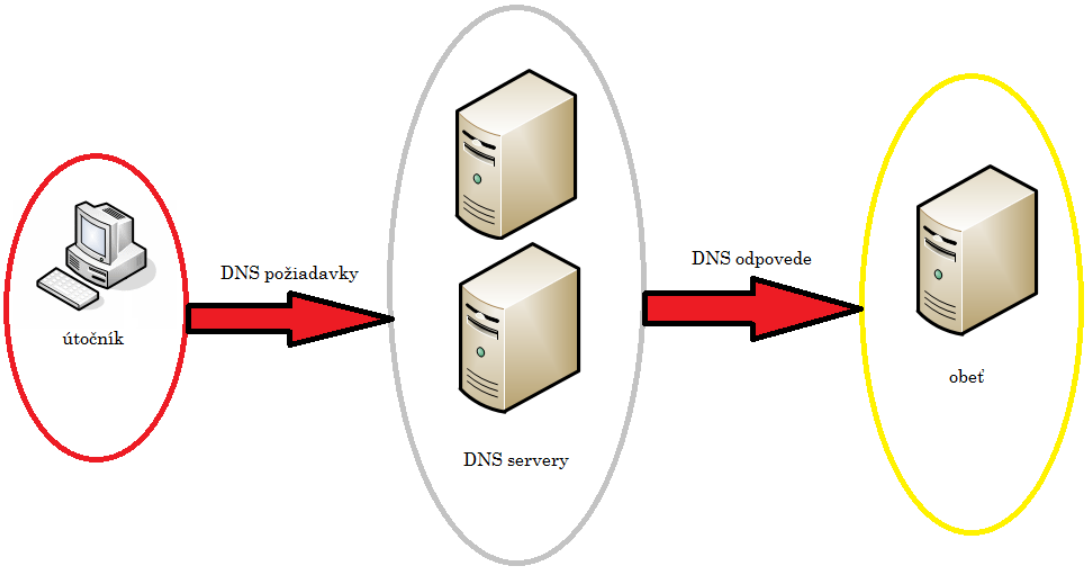
Medzi DoS útoky patrí aj reflexívny DNS útok. Vo svoj prospech však využíva správanie DNS služby, kde pri malom dotaze môže byť vygenerovaná aj niekoľko násobne väčšia odpoveď a tým dosahuje amplifikačný efekt[6]. Preto sa tento útok môže nazývať aj reflexívny, amplifikačný útok. Útok nemusí byť zameraný len na DNS službu, ale môže byť vykonaný na ktorúkoľvek inú, verejne dostupnú službu.

Útočník vygeneruje veľké množstvo dotazov na jeden, prípadne viacero DNS serverov. V dotazoch však podvrhne zdrojovú IP adresu IP adresou obete, takzvaný spoofing[3]. Podvrhnutie je možné kvôli UDP protokolu, ktorý spojenie nijako neoveruje. Jednotlivé DNS servery potom začnú všetky svoje častokrát niekoľko násobne väčšie odpovede posielať nevedome obeti, ktorá si o tieto odpovede nikdy nežiadala. Tým je možné aj pomocou relatívne malého úsilia dosiahnuť efektívneho útoku. Útočník si väčšinou predtým dokáže zistiť, ktoré dotazy dostávajú najväčšie odpovede a zameria sa práve na tento typ dotazov. Najrozsiahlejšie odpovede bývajú na dotazy typu ANY[7], DNSKEY, NS a RRSIG. Amplifikačný efekt sa ešte zväčšuje, ak daný server využíva pre lepšiu bezpečnosť DNSSEC. Avšak požadovanie stále tej istej odpovede môže viesť k ľahkej detekcii útoku, preto sa väčšina útokov snaží dotazy postupne obmieňať. Útočník môže taktiež využiť vopred infikované počítače, ktoré má pod svojou kontrolou, aby taktiež generovali DNS dotazy s podvrhnutou IP adresou obete, čím sa útok ešte viac zefektívni. Po začatí útoku obeti začína dochádzať také množstvo nevyžiadaných odpovedí, na ktoré musí reagovať, že je úplné odrezaná od internetového spojenia. Tento útok môže buď zahltiť celé šírkové pásmo siete obete, kvôli čomu sa znemožní posielanie alebo prijímanie ďalších paketov, prípadne útok zahltí všetky zdroje obete ako pamäť a procesor, ktoré sa budú snažiť prijímané dáta spracovávať a tým ochromia celý systém.

Výhodou takto generovaného útoku oproti nereflexívnemu útoku je nie len väčšia efektivita, ale aj anonymita útočníka, keďže obeti sa útok javí akoby na ňu útočil legitímny DNS server. Preto je veľmi obťažné zistiť, odkiaľ útok skutočne prišiel.

[6] Amplifikačný efekt sa dá vyrátať ako pomer veľkosti odpovede ku veľkosti dotazu.

[7] ANY dotaz vráti všetky typy záznam korešpondujúcich s danou doménou.



Obr. 2.4.1.1 : Reflexívny DNS útok

Existuje viacero možností ako sa proti týmto útokom brániť.

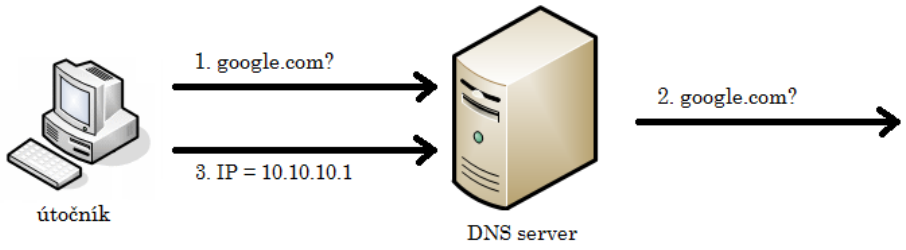
Najjednoduchšou ochranou proti tomuto útoku zvykne byť brána Firewall. Pomocou tejto brány sa dajú filtrovať pakety na ktoré počítač nebude reagovať. Brána môže byť nastavená aby neprepúšťala pakety, ktoré bývajú často používané na tento typ útoku, napríklad pakety obsahujúce DNS záznamy typu ANY. Tento typ ochrany však znemožní prijímanie legitímnych DNS odpovedí. Ak útočník vyberie na útok typ záznamu, ktorý práve nie je blokovaný bránou, ochrana je neúčinná. Preto táto ochrana nie je veľmi efektívna. Ďalšou možnosťou ochrany je napríklad využitie mechanizmu BCP38 [3] na smerovačoch. Tento mechanizmus kontroluje zdrojové adresy v paketoch v danom sektore siete, a ak by sa nejaká adresa v tomto sektore nachádzať nemala, paket zahodí. Vďaka tomuto mechanizmu sa útočníkovi znemožňuje využitie spoofingu. Problém však nastáva, ak útočník útočí na adresu v rovnakom sektore. Iným typom ochrany sa zaoberá metóda DNS Dampening [4], v ktorej existujú trestné body za posielanie rovnakých dotazov na tú istú zdrojovú IP adresu. Ak počet trestných bodov prekročí určitú hranicu, server na tieto dotazy prestane odpovedať pokiaľ počet trestných bodoch opäť neklesne pod stanovenú hranicu. Metóda je implementovaná na DNS serveroch, čo zabráni týmto serverom, aby boli použité k útoku. Podobnou metódou je metóda RRL, ktorá limituje počet poslaných unikátnych odpovedí na jednu adresu a po prekročení tohto limitu buď na danú adresu prestane odpovede posielať, alebo si vyžiada TCP spojenie.

2.4.2 Cache Poisoning

Doteraz zmienené útoky neboli priamo zamerané na službu DNS, ale využívali jej chovanie, aby útok napríklad zosilnili prípadne umožnili funkčnosť iného útoku. Útok zvaný cache poisoning sa už zameriava priamo na DNS servery, konkrétnejšie na ich cache pamäte.

Ak užívateľ posiela nejaký dotaz na DNS server, server najprv prehľadá svoju databázu či sa v nej nenachádza hľadaná doména. Ak doménu nenašiel pýta sa serveru zodpovedný pre zónu do ktorej hľadaná IP adresa patrí. Po získaní odpovede informácie vráti užívateľovi a navyše si odpoveď uloží do svojej dočasnej pamäte, aby sa pri opätovnom dotaze na danú doménu nemusel dopytovať iného serveru. Výhodou takéhoto chovania je zmiernenie záťaže na autoritatívny server, no jeho nevýhodou je ľahké podvrhnutie odpovede, ak táto komunikácia nie je dostatočne zabezpečená. Keď sa útočníkovi podarí takúto odpoveď podrhnúť, užívateľ dostane nepravdivú odpoveď a ani o tom nebude vedieť. Následne po zadaní internetovej stránky, je presmerovaný na stránku vytvorenú útočníkom, ktorá vyzerá identicky s pravou stránkou, a užívateľ bez toho aby niečo tušil, na tejto stránke poskytuje útočníkovi svoje osobné informácie ako heslá k účtom, čísla bankových účtov atď.

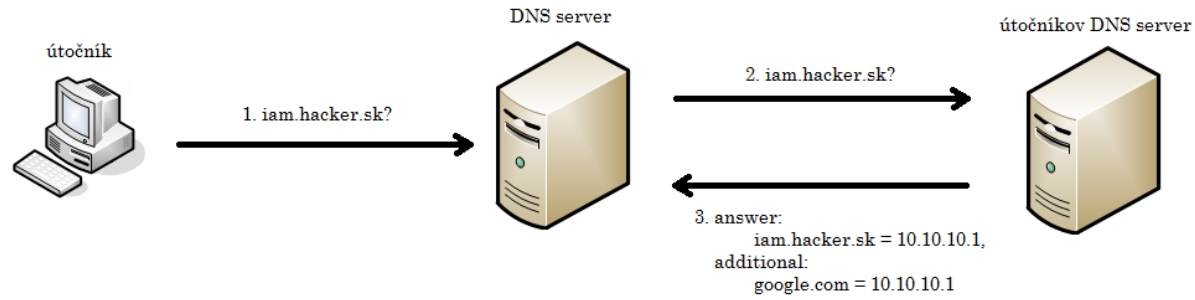
Útočník sa najskôr spýta cieleného DNS servera na záznam pre doménu, ktorú chce podvrhnúť. Hneď nato však na cielený DNS server zašle aj odpoveď s nepravdivou informáciou o tejto doméne. Využije spoofingu, aby si nič netušiaci server myslel, že odpoveď došla od legitímneho serveru. Ak je už doména podvrhnutá, každý ďalší užívateľ, ktorý sa na danú doménu dopytuje dostáva nesprávnu odpoveď a nevedome je presmerovaný na útočníkov web server. Avšak útočník pri tomto útoku naráža na jeden problém. Aby nebolo možné takto jednoducho odpoveď podvrhnúť, server do paketu vkladá aj unikátne 16-bitové číslo a navyše odpoveď očakáva len na konkrétnom porte, čo je taktiež 16-bitové číslo. Obe tieto hodnoty sú generované náhodne. Z toho vyplýva, že ak útočník chce, aby server jeho podvrhnutú odpoveď prijal musí uhádnuť jedno číslo z 2^{32} možností. Buď sa mu podarí odchytiť v sieťovej komunikácie tento paket a získa z neho obidve hodnoty, alebo použije útok hrubou silou. Pri útoku hrubou silou útočník vygeneruje veľké množstvo paketov, každý s inými hodnotami až kým jeden paket nebude obsahovať tie správne hodnoty.



Obr. 2.4.2.1 : Cache poisoning s využitím útoku hrubou silou

Ďalšou možnosťou ako tento útok zrealizovať je využitie poľa additional v DNS odpovedi. Útočník sa opýta cieleného DNS serveru na vymyslenú doménu. Autoritatívnym serverom pre túto doménu je práve útočníkov DNS server. Obetný server sa potom na odpoveď spýta tohto škodlivého DNS serveru a ten mu vráti hľadanú odpoveď, no v odpovedi využije poľa additional, do ktorého

podvrhne iné nelegitímne záznamy na už existujúce domény. Poškodený server si potom navyše okrem vymyslenej domény uloží aj všetky podvrhnuté záznamy a ďalej ich vracia ako záznamy legitímne.



Obr. 2.4.2.2 : Cache poisoning s využitím poľa additional

Najúčinnejšou metódou ako sa proti tomuto útoku brániť je využitie zabezpečovacieho mechanizmu DNSSEC. Vďaka tomuto mechanizmu je možné overovať, že odpoveď prišla od serveru, od ktorého bola očakávaná a nebola podvrhnutá.

2.4.3 DNS tunelovanie

Ak sú osobné počítače pripojené do internetu, sú neustále pod hrozbou internetového útoku. Útočníci sa snažia nejakým spôsobom nabúrať do počítača a mať ho pod svojou kontrolou. Získať prístup k počítaču môžu buď skenovaním jeho otvorených spojení a pomocou nejakého počítač ovládnu, prípadne do počítača vopred nainštalujú svoj program, pomocou ktorého sa s obeťou spoja. Obeť môže takýto program získať nepozornosťou pri užívaní internetu prípadne od nejakého požičaného USB kľúča. Najmä neskúsený užívatelia sa môžu takýmto programom nakaziť ľahšie ako by očakávali. Ak už útočník získal prístup k počítaču, môže ho využívať na ľubovoľné veci, ktoré práve potrebuje. Môže sledovať čo užívateľ práve robí a kradnúť pritom jeho súkromné informácie, prípadne môže jeho počítač využiť k ďalším útokom, aby ich zosilnil.

Samozrejme, proti takýmto útokom už existujú mechanizmy ako sa brániť. Medzi najbežnejšie patrí detekčný a prevenčný systém IDPS [7] a firewall [8]. IDPS obsahuje pravidlá, podľa ktorých sa snaží odhaľovať podozrivú komunikáciu. Ak na takúto komunikáciu narazí, zablokuje ju a oznámi o nej správcovi siete. Firewall obsahuje pravidlá, podľa ktorých vie triediť komunikáciu podľa protokolov, IP adries, portov atď. Na základe týchto pravidiel, jednotlivé pakety či už do siete vchádzajú alebo z nej vychádzajú, buď prepustí, alebo zahodí.

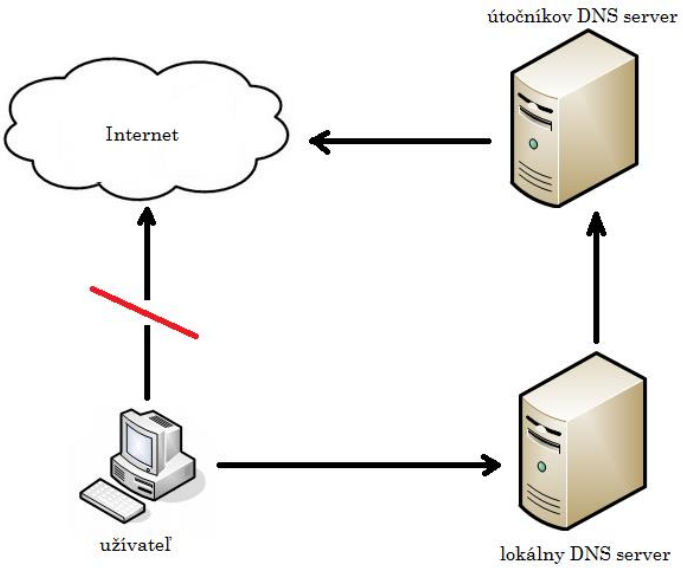
Útočníci však prišli na spôsob ako sa týmto kontrolám vyhnúť. Svoje škodlivé dáta sa snažia zabaľovať do iných protokolov a tým sa vyhnúť kontrolám, ktoré by škodlivé dáta inak odhalili. Najmenej kontrolované komunikácie zvyknú byť DNS dotazy a odpovede. Je to spôsobené tým, že ide

o často používanú službu, ktorej kontrola by značne ovplyvnila priepustnosť siete a navyše DNS pakety obsahujú len limitovanú funkčnosť, takže veľmi často bývajú tieto pakety poslané ďalej bez kontroly.

Prevažná časť lokálnych sietí má aj svoj vlastný DNS server. Ak chce užívateľ získať nejaký DNS záznam, pýta sa práve toho DNS servera. Server prehľadá svoju databázu a ak tento záznam nenájde, má tiež uložené ďalšie DNS serveri, ktorých sa spýta či daný záznam nepoznajú. Ak ani opýtaný server nepozná odpoveď, povie pôvodnému serveru kde by sa mal ďalej pýtať a server sa takto iteratívne pýta rôznych serverov pokiaľ nenájde odpoveď. A týmto iteratívnym spôsobom môže dôjsť až k útočníkom ovládanému serveru, ktorý s pôvodným užívateľom môže takouto cestou jednoducho komunikovať a vyzerať pritom ako obyčajná DNS prevádzka.

Aby útočník čo najlepšie ukryl svoje dáta a aby neboli ľahko odhaliteľné, musí sa svojimi správami čo najviac podobať bežnej DNS prevádzke. Napriek tomu má na výber veľké množstvo typov záznamov nie všetky sú vhodnou voľbou. Jednoduché záznamy ako napríklad A záznam mávajú príliš krátke dĺžky, čo by útočníkovi nestačilo a ak by chcel do záznamu zapísať väčšie množstvo dát, tieto pakety by sa stali ľahko podozrivé. Existujú však záznamy ako CNAME a TXT. Tieto záznamy obsahujú dostatočne dlhé textové reťazce, aby sa s nimi dalo pracovať, preto bývajú najčastejšou voľbou pre ukrývanie dát. Napríklad ak škodlivý program na počítači odhalí užívateľove heslo, môže ho poslať útočníkovmu DNS serveru *nameserver.devildomain.com* ako dotaz pre TXT záznam *passwd.qwerty123.evildomain.com*. Tento dotaz sa iteratívne dostane až k cielenému serveru a ten z nej jednoducho vyberie užívateľovo heslo *qwerty123* bez toho, aby hocijaký filter siete niečo zaznamenal.

Vybranie správneho záznamu na ukrytie dát však nezaručí úplný úspech. Ak by sa v prevádzke začalo objavovať priveľa paketov obsahujúcich záznamy, ktoré sa vyskytujú len zriedkavo, stali by sa takisto ľahko odhaliteľnými. Preto útočník nemôže vyberať typy záznamov len podľa ich dostupného miesta pre informácia, ale navyše ich nemôže poslať viac ako ich býva v bežnej sieťovej prevádzke. V bežnej prevádzke býva CNAME záznamov mnohonásobne viac ako TXT záznamov, preto musí aj útočník spoliehať viac na tieto typy a záznamy s menším výskytom používať rovnako zriedkavejšie.



Obr. 2.4.3.1 : DNS tunelovanie

DNS tunelovanie však nemusí vždy spájať len s útočníkom ktorý sa snaží nepozorovane napadnúť nevinného užívateľa. Tunelovanie môže využiť aj samotný užívateľ vo svoj prospech. Napríklad ak chce využiť komunikáciu ktorá je blokovaná lokálnou sieťou, prípadne ak sa chce vyhnúť poplatkom za nejakú sieť, ktorá mu prístup k internetu umožní až po prihlásení cez nejakú internetovú stránku. Máloktoré siete majú totiž proti DNS tunelovaniu mechanizmu ako mu zabrániť.

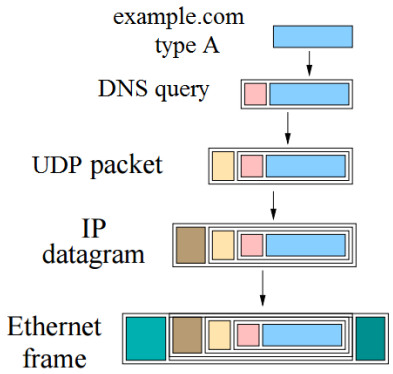
Najúčinnejšou ochranou proti DNS tunelovaniu by bolo kontrolovanie obsahu paketov a hľadanie dát, ktoré by mohli znamenať nejakú podozrivú aktivitu. Toto riešenie by však enormne zaťažilo sieť, takže nie je najvhodnejším. Ďalšími metódami bývajú monitorovanie podozrivých DNS tokov, kde sa kontroluje koľko dotazov sa na akú doménu poslalo, a ak je to príliš veľa tak sa tento tok stáva podozrivý. Napríklad v článku [9] sa autori snažili simulovať DNS tunelovanie a pomocou výsledkov čo namerali, hľadali riešenie ako tunelovanie odhaliť. Pre lepšie odhaľovanie zahodili UDP hlavičku a porovnávali len DNS dáta. Na detekciu tunelovania použili Jensen-Shannovu vzdialenosť a zistili, že normálna DNS prevádzka má túto vzdialenosť vysokú, zatiaľ čo DNS prevádzka s tunelovaním má vzdialenosť menšiu. Zistili taktiež aká je vhodná hranica vzdialenosti na detekciu DNS tunelovania a dokázali, že takéto monitorovanie môže byť použité ako detekcia tunelovania. Ich metóda však nefunguje ak sú dáta šifrované. Autori v [10] zase použili na detekciu tunela n-gramovú frekvenčnú analýzu. Snažia sa porovnávať aktuálnu DNS prevádzku s nejakým súborom, ktorý obsahuje legitímnu DNS prevádzkou. Aby táto metóda bola čo najdôveryhodnejšia, mala by sa použiť prevádzka zo siete, do ktorej chceme detekciu tunela zakomponovať. Aktuálna prevádzka sa rozkladá na n-gramy a tieto sa postupne porovnávajú s n-gramami v pôvodnej legitímnej prevádzke.

3 Analýza DNS anomálií

Analýza DNS anomálií je pred ich samotným implementovaním veľmi podstatná. Pomocou analýzy je možné zistiť, ako vyzerajú pakety vygenerované konkrétnym útokom a lepšie pochopiť ich správanie. Pre niektoré útoky je ťažké nájsť nejaké už existujúce programy, prípadne vygenerované dáta, preto boli pre analýzu použité prevažne vedecké články, ktoré sa týmito útokmi zaoberali.

Pre lepšie pochopenie jednotlivých analýz útokov je dobré si najprv rozobrať, ako vyzerá štandardná štruktúra DNS paketu v sieti.

Paket posielaný po sieti je zabalený do viacerých protokolov TCP/IP modelu[8]. Ak sa jedná napríklad o paket, ktorý v sebe nesie nejaké dáta aplikačného protokolu, tak sú tieto dáta zabelené pomocou hlavičky do aplikačného protokolu (napr. DNS), ten je zabalený do transportného protokolu (napr. UDP), ten je zabalený do internetového protokolu (napr. IP) a nakoniec je všetko zabalené do protokolu fyzického rozhrania (napr. Ethernet). Takto zabalené dáta sú potom poslané na sieť a po doručení do cieľa sú postupne z jednotlivých protokolov rozbalené.



Obr. 3.1 : Zabaľovanie dát do jednotlivých protokolov, zdroj: [5]

Každý jeden protokol musí dodržiavať určité pravidlá, aby sa s ním mohlo na ľubovoľnej stanici v sieti pracovať. Takto je možné pakety posielať po lokálnej sieti, medzi sieťami a rozdeľovať ich jednotlivým bežiacim procesom v počítači, kde sa následne spracujú konkrétne dáta.

Pre smerovanie v lokálnej sieti sa používa MAC adresa zariadenia. Táto adresa sa skladá zo 48 bitov a je pre každé sieťové zariadenie jedinečná. V pakete je MAC adresa uložená v hlavičke fyzického rozhrania. Ak paket smeruje aj mimo lokálnu sieť, tak sa jeho cieľ identifikuje pomocou IP adresy uloženej v internetovej hlavičke paketu. Tá sa skladá zo štyroch 8 bitových hodnôt, dokopy má teda 32

[8] TCP/IP model popisuje sieťovú architektúru a rozdeľuje ju na štyri vrstvy: vrstva fyzického rozhrania, internetová vrstva, transportná vrstva a aplikačná vrstva. Jednotlivé internetové protokoly sa potom rozdeľujú do týchto vrstiev, podľa ich účelu.

bitov. Po dorazení paketu do cieľovej stanice je ešte potrebné, aby bol paket pridelený správnemu procesu. Tento proces je identifikovaný pomocou čísla portu, čo je 16 bitové číslo a je uložené v transportnej hlavičke paketu. Všetky tieto hodnoty sú v pakete uložené ako pre cieľové, tak aj pre zdrojové zariadenie.

Ako všetky aplikačné protokoly, aj protokol DNS má svoje pravidlá, podľa ktorých je možné zostaviť paket obsahujúci dotazy, odpovede a ďalšie informácie služby DNS. Tieto údaje je možné nájsť v [6], kde je okrem stavby paketu popísané aj fungovanie samotnej služby DNS, ako medzi sebou komunikujú server a klient, aké sú obmedzenia služby a mnoho ďalších informácii.

Hlavička DNS protokolu sa skladá z transakčného ID, pomocou ktorého sa priraďujú odpovede k prijatým dotazom. Transakčné ID ma veľkosť 2 byty a môže nadobúdať ľubovoľné hodnoty. Ďalšie 2 byty sú znakové bity. V týchto bitoch je možné nastaviť rôzne príznaky, napríklad že daný paket je odpoveď alebo dotaz, označiť, že odpoveď je od autoritatívneho servera, povoliť alebo zakázať rekurziu, prípadne informovať, že pri rezolúcii došlo k nejakej chybe a ďalšie príznaky. Po znakových bitoch nasleduje ďalších 8 bytov, ktoré hovoria, koľko je v pakete otázok a odpovedí, koľko autoritatívnych serverov obsahuje navyše odpoveď a koľko ďalších informácií v sebe paket nesie. Každý jeden počet je uložený pomocou 2 bytov.

```
Transaction ID: 0x0015
⊟ Flags: 0x8180 Standard query response, No error
    1... .... .... .... = Response: Message is a response
    .000 0... .... .... = Opcode: Standard query (0)
    .... .0.. .... .... = Authoritative: Server is not an authority for domain
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... 1... .... = Recursion available: Server can do recursive queries
    .... .... .0.. .... = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... .... ...0 .... = Non-authenticated data: Unacceptable
    .... .... .... 0000 = Reply code: No error (0)
  Questions: 1
  Answer RRs: 6
  Authority RRs: 0
  Additional RRs: 0
```

Obr. 3.2 : DNS hlavička

Po DNS hlavičke už nasledujú samotné DNS dáta, v ktorých sú zahrnuté všetky dotazy, odpovede, informácie o autoritatívnych serveroch a ďalšie informácie. Počet týchto záznamov sa nesmie líšiť od počtu zadaného v hlavičke, inak je paket zahodený.

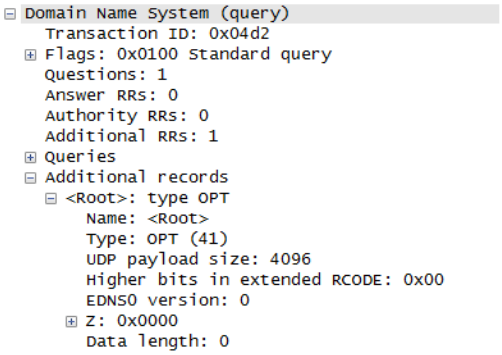
3.1 Reflexívny DNS útok

Ak by sa zo spomínaných útokov mal vybrať jeden, ktorý by bolo programovo na internete najľahšie rozoznať, bol by to práve reflexívny DNS útok. Problém pri detekcii by však mohol nastať, ak by šlo už o distribuovaný útok, to znamená, že by k útoku bolo použitých viacero počítačov, pripadne aj viacero DNS serverov.

Štandardný útok sa vyznačuje veľkým počtom dotazov na ten istý DNS záznam od toho istého počítača. Väčšinou ide o nejaký veľký záznam, aby sa dosiahol čo najväčší amplifikačný efekt. Pri niektorých útokoch je dokonca záznam najprv DNS serveru podvrhnutý záznam s najväčšou veľkosťou, akú serverové zabezpečenie dovoľuje a až potom je útok zahájený.

V poslaných paketoch sa môže meniť transakčné ID prípadne port počítača, no záznam a zdrojová IP adresa sa nemenia.

Ďalším problémom, s ktorým treba pri tomto útoku rátať je maximálna veľkosť DNS UDP paketu. Štandardná maximálna veľkosť DNS dát v UDP pakete je 512 bytov. Ak dáta presiahnu túto veľkosť, DNS server sa ich snaží poslať pomocou TCP protokolu. Je to spôsobené tým, že UDP pakety sa nijako neoverujú, či došli v poriadku a pri veľkých veľkostiach by mohla vzniknúť príliš vysoká stratovosť paketov. Pri zmene UDP protokolu na protokol TCP, by však už nebolo možné podvrhnúť IP adresu obete. V moderných sieťových architektúrach však stratovosť už výrazné klesá, preto sa zaviedol nový mechanizmus, ktorý umožňuje posielať DNS dáta v UDP pakete aj o veľkosti väčšej ako je pôvodných 512 bytov. Vďaka tomuto mechanizmu sa zmierňuje zaťaženie siete, keďže pri zabezpečení DNSSEC bývajú dáta častokrát väčšie ako 512 bytov. Týmto mechanizmom je mechanizmus EDNS(0)[9]. Tento mechanizmus umožňuje posielať DNS dáta v UDP pakete až do maximálnej veľkosti 4096 bytov. EDNS(0) príznak sa potom pridáva ku každému dotazu do poľa s ďalšími informáciami, čím sa oznamuje cieľovému DNS serveru, že stanica podporuje DNS dáta väčšie ako pôvodných 512 bytov. Ak cielený server takisto disponuje týmto mechanizmom, TCP spojenie naviaže až keď dáta presiahnu stanovenú veľkosť.



Obr. 3.1.1 : EDNS(0) v DNS pakete, kde je nastavená maximálna veľkosť DNS dát UDP paketu na 4096 bytov

[9] Extension Mechanisms for DNS

3.2 Cache Poisoning

Cache poisoning sa snaží podvrhnúť DNS serveru nejaký falošný záznam, ktorý bude server ďalej poskytovať užívateľom ako záznam dôveryhodný. Užívateľa je potom možné presmerovať na falošnú stránku, prípadne je možné tento záznam použiť na väčší amplifikačný efekt. Najťažšou časťou pre vykonanie tohto útoku je uhádnutie správneho formátu odpovede, ktorú server očakáva.

Aby server odpoveď prijal za legitímnu, je potrebné aby táto odpoveď splňovala niekoľko faktorov. Prvým faktorom je, aby sedela zdrojová adresa odpovede so zdrojovou adresou dotazu. Túto adresu je relatívne ľahké zistiť. Stačí vedieť, ktorý server je autoritatívny pre doménu, ktorá má byť podvrhnutá a v odpovediach sa podvrhne jeho adresa. Ďalšou podmienkou pre legitímnosť je zhodný dotaz v odpovedi. V službe DNS totiž pakety obsahujú okrem odpovedí a ďalších informácií aj samotnú otázku, na ktorú odpovedajú. Túto podmienku je však veľmi jednoduché splniť, keďže server dotaz, ktorý prijal od klienta nijako nemení. Dotaz sa musí zhodovať v mene aj v type záznamu. Ďalšie dva faktory sa už takto ľahko splniť nedajú. Ide o šestnásť bitové číslo portu, na ktorom server očakáva odpoveď a o takisto šestnásť bitové transakčné ID dotazu. Za predpokladu, že sú obe tieto čísla generované náhodne je šanca na náhodné uhádnutie správnej kombinácie skoro nulová.

V starších implementáciách DNS serverov, do roku 1997, sa používal na dotazy stále ten istý port, prípadne sa pre toho istého klienta používali tie isté porty. Útočníkovi tak stačilo mať vlastný server ktorý je autoritatívny pre nejakú doménu a vedel veľmi jednoducho odhadnúť port, ktorý bude použitý pri jeho ďalšom útoku. S takto zisteným portom stačí vygenerovať pakety so všetkými možnosťami transakčného ID, ktoré sa zašlú DNS serveru. Možností pre transakčné ID je dokopy 65536, čo je s dnešnými výkonnými počítačmi dosiahnuteľné, navyše ak je na DNS server, ktorý má poslať legitímnu odpoveď, vedený nejaký DoS útok, takže odpoveď trvá aj niekoľko sekúnd. Dnešné implementácie však generujú port aj transakčné ID náhodne a takýto útok hrubou silou nie je veľmi efektívny.

Ďalšou možnosťou ako jednoducho uhádnuť správnu kombináciu pre prijatie falošnej odpovede je využitie narodeninového paradoxu. Pri predchádzajúcom útoku sa poslal jeden dotaz na požadovaný záznam a za ním sa poslalo niekoľko vygenerovaných falošných odpovedí. Pravdepodobnosť tohto útoku je potom:

$$p = \frac{x}{2^n}$$

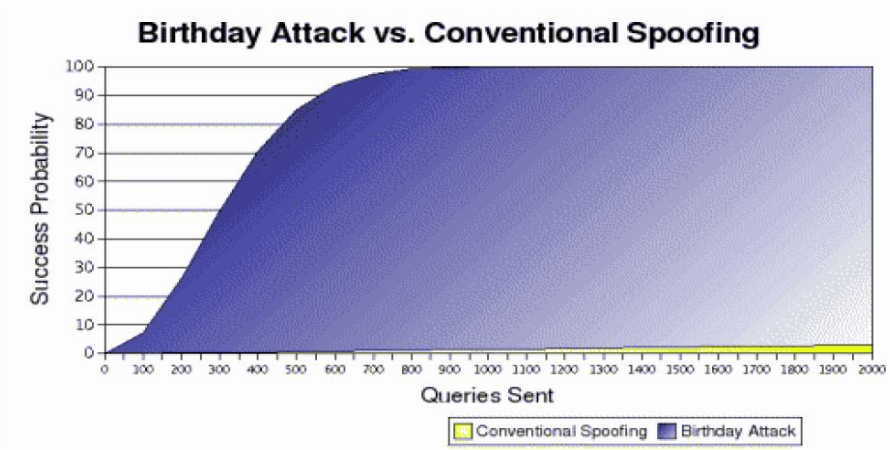
Kde *x* je počet poslaných falošných odpovedí pred príchodom korektnej odpovede a *n* je počet bitov, ktoré je potrebné uhádnuť. Ako je vidieť, táto pravdepodobnosť nie je veľmi efektívna. Narodeninový paradox však hovorí, že:

Pravdepodobnosť že v skupine náhodne vybraných 23 ľudí majú dvaja narodeniny v ten istý deň je väčšia ako 50%.

Za predpokladu, že DNS server pri dotaze na tú istú doménu, preposiela tento dotaz autoritatívnemu serveru stále, pokiaľ ešte nedostal odpoveď, vždy s iným portom a transakčným ID, môže útočník serveru poslať *x* dotazov na záznam, ktorý chce podvrhnúť a zároveň *x* falošných odpovedí. S týmto prístupom, ako je písané v [15], pravdepodobnosť úspešnosti útoku bude vyzerať nasledovne:

$$p = 1 - (1 - \frac{1}{2^n})^{\frac{x * (x-1)}{2}}$$

Kde *x* je počet poslaných odpovedí a *n* je počet bitov, ktoré je potrebné uhádnuť. Pri predchádzajúcej rovnici, ak sme na server odoslali 300 paketov a snažíme sa uhádnuť len transakčné ID, tak pravdepodobnosť úspechu je 0,45%, no pri narodeninovom útoku je táto pravdepodobnosť takmer 50%.



Obr. 3.2.1 : Porovnanie úspešnosti klasického a narodeninového útoku, zdroj: [15]

Žiaľ, od roku 2002, kedy bola táto slabina odhalená, sa pri väčšine nových implementáciách DNS serverov už pre dotazy na tú istú doménu generuje len jeden dotaz na autoritatívny server, takže tento narodeninový paradox sa využiť nedá.

Ak sa takýto útok nepodaril a server si uložil legitímny záznam, ktorý prišiel skôr, do svojej dočasnej pamäte, musí útočník počkať, kým skončí platnosť tohto záznamu a až tak sa pokúsiť útok zopakovať. Pri pokuse úspešnom, bude DNS server vracať miesto legitímneho záznamu, záznam podvrhnutý útočníkom po celú dobu jeho platnosti.

3.3 DNS tunelovanie

Pre DNS tunelovanie už na internete existuje viacero programov, ktoré umožňujú užívateľovi DNS tunel využívať bez toho, aby musel nejako do hĺbky rozumieť jeho fungovaniu. Užívateľ môže takto komunikovať mimo lokálnu sieť, kde by inak bola táto komunikácia blokovaná nejakým sieťovým zabezpečením. Avšak, DNS tunel nemusí byť vždy pre užívateľa nápomocný. Využívajú ho taktiež internetoví útočníci, aby získavali od užívateľa citlivé informácie bez toho, aby užívateľ niečo spozoroval a navyše bez toho, aby tieto úniky dát zablokoval nejaký detekčný mechanizmus. Útočníci samozrejme nepoužívajú komerčné programy pre tvorbu tunelu, ale takéto programy si naprogramujú sami, aby vyhovovali ich potrebám a boli čo najťažšie odhaliteľné.

Pre fungovanie DNS tunelu je potrebné mať okrem klientskej stanice navyše vzdialený server s verejnou IP adresou. Tento server musí byť autoritatívny DNS server pre doménu, pomocou ktorej bude tunel fungovať. Po spustení tunelu sa potom dáta posielajú z klientskej stanice na lokálny DNS server, ktorý tieto dáta preposiela vzdialenému serveru. Vzdialený server nemusí byť DNS serverom, stačí ak serverová časť programu beží na porte 53 a je schopná spracovať a odpovedať na prichádzajúce dáta. Niektoré implementácie tunelu umožňujú navyše preposielať dotazy, ktoré sa netýkajú domény pomocou ktorej beží tunel, na vopred zvolený DNS server, prípadne na iný port v počítači. V niektorých implementáciách sa po naviazaní spojenie medzi klientom a serverom dáta posielajú už priamo, bez použitia lokálneho DNS servera. Takéto priame spojenie je potom ustanovené pomocou SSH protokolu, čo poskytuje prenášaným dátam zabezpečenie.

Najväčším problémom DNS tunelu je obmedzená veľkosť prenášaných dát. Tento problém sa týka najmä dát posielaných z klienta na server. Tieto dáta môžu byť posielané len v tvare doménového mena, ktoré má svoju maximálnu dĺžku. Maximálna dĺžka doménového mena je 253 bytov, kde každá subdoména môže byť dlhá maximálne 63 bytov. Ak názov domény, ktorá je použitá na tunel, je napríklad *mydomain.com*, potom na posielanie dát od klienta máme len takýto doménový formát:

$$[48\ znakov].[63\ znakov].[63\ znakov].[63\ znakov].mydomain.com$$

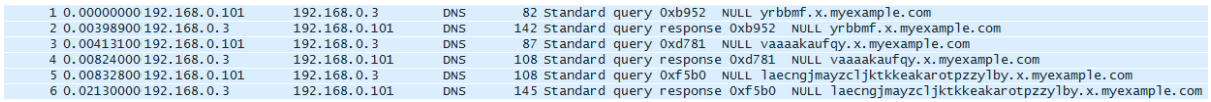
Ako je vidieť, na posielanie dát s takouto doménou ostáva len 237 znakov. Ak by sa z klientskej stanice malo odosielať veľké množstvo dát, bol by tunel ľahko odhalený. V štatistikách by mohol mať takýto klient aj 90% všetkých dotazov poslaných na DNS server, čo by správcovi siete bolo ihneď podozrivé.

Ak je pomocou DNS protokolu potrebné poslať nejaké súbory, to znamená binárne dáta, musí sa navyše pre tieto dáta použiť nejaké kódovanie. Je to potrebné preto, aby v doménovom mene

nevznikali nejaké escape sekvencie[10], ktoré by zapríčinili zahodenie paketu. Najpoužívanejšie kódovanie, ktoré používajú už existujúce programy DNS tunelovania, je kódovanie base32, prípadne base64. Pri kódovaní base32 je možné do jedného znaku vložiť 5 bitov a pri kódovaní base64 až 6 bitov. To však znovu uberá z maximálnej veľkosti dát poslaných v jednom dotaze. Pri použití predchádzajúceho príkladu a za použitia base32 kódovania by bolo možné do jedného dotazu zakomponovať len 237 x 5 bitov, čo je dokopy 148 bytov dát. Ak by sa použilo kódovanie base64, táto veľkosť pri vzrástla na 237 x 6 bitov, takže 177 bytov.

Čo sa týka odpovedí, tam už maximálna veľkosť nie je taká obmedzujúca. Odpovede typu TXT môžu obsahovať až 255 znakov a odpovede typu RRSIG až takmer 512[11] bytov. S mechanizmom EDNS(0) je možné túto veľkosť zväčšiť až na 4096 bytov. Pri veľkých paketov už ale môže dochádzať k vysokej stratovosti, keďže sa stále jedná o UDP protokol.

Medzi najznámejšie programy na DNS tunelovanie, ktoré sú voľne dostupné sú *Iodine*[12], *OzymanDNS*[13], *MagicTunnel*[14] a *Heyoka*[15]. Väčšina takýchto nástrojov používa base32 alebo base64 kódovanie a fungujú na rovnakom princípe.



Obr. 3.3.1 : Ukážka paketov z nástroja Iodine

Obr. 3.3.2 : Ukážka paketov z nástroja OzymanDNS

Na prvý pohľad z obrázkov [3.3.1] a [3.3.2] je vidieť, že pakety z oboch nástrojov vyzerajú približne rovnako. Nástroj *Iodine* navyše používa dotazy so záznamami typu NULL. Tento typ záznamu slúži na testovacie účely a v niektorých sieťach môže byť obmedzovaný.

[10] Escape sekvencia je nejaká kombinácia znakov, na ktoré nereaguje počítač ako na štandardný text, čo môže spôsobiť zmenu chovania počítača.

[11] Táto veľkosť v sebe zahrňuje veľkosť DNS hlavičky, veľkosť dotazu a veľkosť hlavičky záznamu typu rrsig.

[12] http://code.kryo.se/iodine/

[13] http://www.dankaminsky.com/2004/07/29/51/

[14] http://www.magictunnel.net/index.php

[15] http://heyoka.sourceforge.net/

3.4 Škodlivé domény

Škodlivé domény nezapadajú priamo medzi anomálie DNS, ale tieto domény sú častokrát použité na nejakú škodlivú činnosť na internete. Najčastejšie sa využívajú pri rozosielaní spamovej internetovej pošty, pri DNS tunelovaní a pri komunikácii v rámci botnetov.

Aby útok nezávisel len na jednej konkrétnej doméne, čo by mohlo viesť k jej zablokovaniu, útočníci do svojich programov častokrát implementujú nejaký generátor domén. Po zablokovaní domény si tento generátor jednoducho vygeneruje doménu novú a môže pokračovať v škodlivej činnosti.

Pri DNS tunelovaní obsahujú server aj klient rovnaký generátor domén. Klient si potom po spustení pomocou DNS dotazu zistí, ktorá doména je aktuálna a použije ju na komunikáciu so serverom.

Tieto algoritmy na generovanie domén sú plne v rukách programátora a nemajú nejaký všeobecný postup. Takéto domény je však možné spozorovať, pretože nezvyknú obsahovať nejaké slová, ktoré dávajú zmysel. Väčšinou ide len o nejakú náhodnú postupnosť čísel a znakov.

4 Implementácia

Pred implementovaním jednotlivých programov, bolo potrebné vybrať vhodný implementačný jazyk. Hlavnou podmienkou pre výber jazyka bolo, aby vedel pracovať s paketmi aj na nižších vrstvách, ako je aplikačná vrstva a aby táto práca nebola príliš náročná, čo by mohlo do programu zaniesť neželané chyby.

Na implementáciu bol použitý programovací jazyk C++, presnejšie, štandard C++11. C++ obsahuje viacero knižníc, ktoré umožňujú jednoducho pracovať s paktami na nižších úrovniach. Navyše pre tento jazyk existujú kvalitné vývojové prostredia, ktoré prácu značne uľahčujú.

Pre prácu s paketmi bola použitá knižnica *libnet*. Knižnica poskytuje všetky potrebné funkcie, ktoré boli potrebné pre tvorbu programu a práca s nimi je jednoduchá. Navyše je pre knižnicu vytvorená kvalitná dokumentácia. Tu je potrebné poznamenať, že túto knižnicu štandardne jazyk C++ neobsahuje a pred spustením, prípadne preložením programu je potrebné ju nainštalovať.

Pre správne fungovanie programu je navyše potrebné okrem knižnice *libnet*, doinštalovať aj knižnicu *libpcap*, ktorá pomáha pri tvorbe libpcap súborov.

Pre efektívnejšiu prácu s touto knižnicou, boli všetky jej funkcie zabalené do vlastnej triedy. Táto trieda umožňovala prácu len s potrebnou funkčnosťou knižnice a na jej ďalšie možnosti využila nejaké prednastavené hodnoty. Tento prístup značne sprehľadnil jednotlivé úseky kódu a navyše implementáciu jednotlivých programov uľahčil.

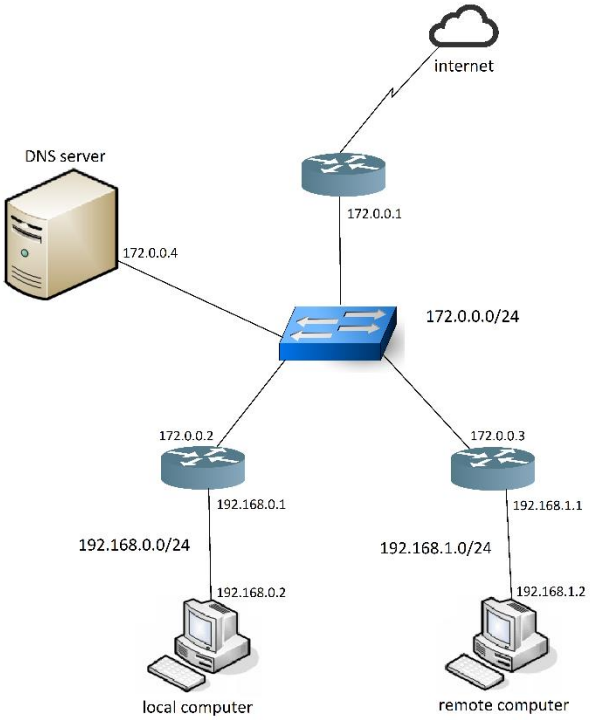
Všetky programy tejto práce boli vyvíjané a testované v linuxovom prostredí a ich fungovanie na iných platformách sa nepredpokladá. Na preklad bol použitý prekladač g++ vo verzii 4.8.2.

Jednotlivé programy sú implementované ako konzolová aplikácia.

4.1 Testovacie prostredie

Vytvorenie si vlastného testovacieho prostredia pre testovanie programov bolo nevyhnutné. Bez DNS serveru by sa funkčnosť programov nijako overiť nedala a ak by sa jednotlivé útoky skúšali v reálnej sieťovej prevádzke na nejakých verejných DNS serveroch, mohli by byť tieto útoky potrestané.

Pre testovacie účely bola preto vytvorená vlastná lokálna sieť s dvomi podsieťami, čo nedovolilo paketom jednotlivých útokov dostať sa do bežnej sieťovej prevádzky. Táto sieť bola vytvorená v sieťovom laboratóriu, kde boli k dispozícii všetky potrebné sieťové prvky. Použité boli smerovače *Cisco 2811 ISR* a prepínač *Catalyst 2960 Series*.



Obr. 4.1.1 : Testovacie prostredie

DNS server bol spojazdnený pomocou softwaru BIND9. Tento program bol vybraný kvôli jeho ľahkej konfigurácii a pretože je to najpoužívanejší software pre správu DNS serveru.

Testovací server bol nakonfigurovaný ako záložný DNS server. Pre testovacie účely nebolo potrebné aby bol server primárny server pre nejakú doménu, ale výhodnejšie bolo získavať aktuálne záznamy z nejakého naozajstného vereného DNS servera. Pre túto potrebu bol vybraný verejný DNS server na adrese 8.8.8.8. Tento server bol vybraný, lebo je verejne dostupný takmer na celom svete a zaručuje pravdivé informácie. Toho sa docielilo pridaním nasledujúceho parametru do súboru *named.conf.options*:

forwarders { 8.8.8.8; };

Takto nastavený server preposielal všetky dotazy na záznamy, ktoré nemal uložené vo svojej dočasnej pamäti na zadaný server. Aby však nebolo potrebné registrovať vlastnú doménu pre potreby DNS tunelovania, bolo potrebné, aby server preposielal dotazy na doménu *mydomain.com* počítaču na adrese 192.168.1.2, kde bežala serverová časť tunelovania. Toto správanie sa docielilo pridaním parametru do súboru *named.conf.local*:

```
zone “mydomain.com” {
        type forward;
        forwarders {
                192.168.1.2 port 53;
        };
};
```

Pre testovacie účely sa do konfiguračného súboru serveru pridalo ešte niekoľko parametrov. Parametre sa zadávali v súbore *named.conf.options* do parametru *options*.

V prvom rade bolo potrebné aby DNS server podporoval rekurzívne dotazy aj od užívateľov v iných sieťach ako je samotný server. Vzhľadom nato, že testovacia sieť nebola verejne prístupná, rekurzívne dotazy sa mohli povoliť od ľubovoľných sietí. Toto správanie sa dosiahlo pomocou parametrov:

```
allow-recursion { any; };
allow-recursion-on { any; };
```

Ďalším parameter, ktorý bolo potrebné nastaviť, bol parameter, ktorý umožňoval serveru pracovať s UDP paketmi o veľkosti až do 4096 bytov. Táto možnosť bola využitá najmä pri reflexívnom amplifikačnom útoku, pre dosiahnutie čo najväčšieho amplifikačného efektu. Toto správanie zaručovali parametre:

```
edns-udp-size 4096;
max-udp-size 4096;
```

Takto nakonfigurovaný server umožňoval vykonať a otestovať všetky implementované programy.

4.2 Konfiguračný súbor

Konfiguračný súbor zjednodušuje prácu s jednotlivými programami. Každý program pre svoj beh potrebuje niekoľko povinných parametrov a ak by sa tieto parametre mali písať pomocou príkazovej riadky, značne by to prácu s jednotlivými programami zhoršovalo. Ak by zase program mal mať len minimum parametrov, jeho použitie by sa výrazne obmedzilo.

Hoci každý z programov má inú kombináciu parametrov, ktoré pre svoj beh potrebuje, rovnaké parametre znamenajú stále tú istú vec, čo sa jednotlivých útokov týka. Každý z programov obsahuje parametre pre nastavenie výstupu programu, čo umožňuje generovať z programu priamo súbor

obsahujúci informácie o jednotlivých paketov. Väčšina programov obsahuje taktiež IP adresu a port počítača odkiaľ sú spustené a IP adresu a port DNS serveru, ktorý je k útoku použitý.

Všetky parametre aj s ich jednotlivým vysvetlením je možné nájsť pri každom programe v súbore *README.txt,* prípadne sa parametre vypíšu po zadaní vstupného parametru --help v príkazovom riadku.

Ak sa spustí program bez parametru, program sa snaží načítať prednastavený konfiguračný súbor s názvom *config.conf*. Pre použitie iného konfiguračného súboru sa názov tohto súboru zadá ako jediný vstupný parameter programu.

Konfiguračný súbor môže obsahovať len riadky s jednotlivými parametrami, prázdne riadky a riadky s komentármi, ktoré začínajú dvomi znakmi *"//"*. Riadok s parametrom musí dodržiavať nasledujúci formát:

[názov parametru]=[hodnota]

Za názvom parametru nasleduje bez žiadnej medzery znak *'='* a za ním, takisto bez medzery, hodnota parametru. Všetky iné formáty, prípadne názvy parametrov, ktoré daný program nepodporuje, budú vyhodnotené ako chyba konfiguračného súboru, čo bude mať za následok ukončenie programu.

Povinné parametre, ktoré budú v konfiguračnom súbore chýbať, sa takisto prejavia chybovým ukončením programu. Užívateľ bude na chýbajúci parameter upozornený chybovým výpisom.

Ak sa v konfiguračnom súbore vyskytne niektorý parameter viackrát a tento viacnásobný výskyt nie je programom podporovaný, v úvahu sa bude brať posledná zvolená hodnota v súbore.

4.3 Generovanie libpcap súboru

Každý z programov má dve možnosti ako generovať výstup. Je to buď na sieť alebo do súboru vo formáte libpcap. Niektoré programy umožňujú tieto dva spôsoby aj kombinovať.

Libpcap súbor je štandardný súbor na ukladanie sieťovej komunikácie. Vzhľadom nato, že je to presne stanovený formát, s takýmito súbormi pracuje viacero programov, ktoré umožňujú užívateľovi prehľadne študovať jednotlivé pakety ktoré sa vyskytli na sieti. Najznámejší z takýchto programov je program *Wireshark*[16], ktorý dokáže navyše sieťovú komunikáciu odchytávať a libpcap súbory vytvárať.

Výhodou libpcap súborov je, že obsahujú všetky dáta ktoré sa na konkrétnom sieťovom rozhraní vyskytli. Takto je možné spätne kontrolovať obsahy paketov, čo pri iných formátoch so štatistikami o sieti, ako je napríklad NetFlow, možné nie je.

[16] https://www.wireshark.org/

Nevýhodou libpcap súboru je jeho veľkosť, ktorá sa pri rušnej prevádzke môže vyšplhať až na tisíce megabytov.

V projekte je toto generovanie súboru implementované, aby jednotlivé generátory útokov mohli pracovať aj bez pripojenia do siete, prípadne aby tieto vygenerované útoky nebolo potrebné zachytávať nejakou ďalšou aplikáciou. Takto vygenerovaný súbor je potom možné vložiť do detekčného mechanizmu bez nutnosti behu týchto programov súčasne a testovať ich funkčnosť.

Libpcap súbor sa skladá z globálnej hlavičky, ktorá obsahuje informácie ako časovú zónu, verziu formátu, dĺžku záznamov, typ rozhrania a ďalšie. Táto hlavička sa nachádza v súbore len raz a to na začiatku súboru. Za hlavičkou potom nasledujú hlavičky jednotlivých uložených paketov, za ktorými sú uložené samotné dáta paketu, od najnižšej sieťovej vrstvy. Hlavička paketu obsahuje informácie o čase, kedy bol paket zachytený a o dĺžke paketu.

Obr. 4.3.1 : Formát libpcap súboru, zdroj [14]

Podrobnejšie informácie o formáte libpcap je možné nájsť v [14], kde je okrem formátu súboru popísaný aj základný princíp tvorby takéhoto súboru.

4.4 Generované útoky

Projekt sa skladá z piatich programov, ktoré generujú DNS útoky a jedného obslužného skriptu. Nebolo vhodné tieto útoky implementovať, ako jeden veľký program. Takýto prístup by značne nesprehľadnil prácu s jednotlivými útokmi.

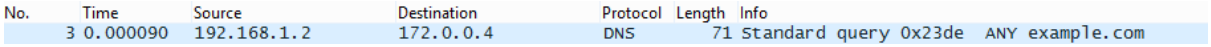
Pre lepšiu prácu s jednotlivými útokmi bol navyše implementovaný jednoduchý obslužný skript. Aby bol tento skript funkčný, je potrebné, aby každý útok bol v samostatnom adresári, ktorý je pomenovaný ako samotný program. V tomto adresári musia byť všetky zdrojové súbory programu, konfiguračný súbor a funkčný *Makefile* programu. Obslužný skript potom umožňuje preložiť všetky programy naraz, vymazať všetky súbory prekladu a vygenerované libpcap súbory naraz, a spúšťať jednotlivé programy z jedného miesta v počítači. Podrobnejšie informácie o fungovaní skriptu sa dajú získať, ak sa skript spustí s parametrom help.

4.4.1 Reflexívny DNS útok

Najľahší útok na implementáciu, z implementovaných útokov je práve reflexívny DNS útok. Tento útok je implementovaný v programe *ampli_attack*.

Hlavným problémom pri implementácii tohto útoku bolo podvrhnutie zdrojovej IP adresy. Vzhľadom nato, že knižnica libnet umožňuje prácu s paketom aj na internetovej vrstve, IP adresa obete

sa do IP hlavičky jednoducho vložila. Takýto paket sa potom jednoducho poslal na sieť a tváril sa ako keby prišiel od počítača, ktoré ho IP adresa bola podvrhnutá. IP adresu a port obetného počítača je možné zadať v konfiguračnom súbore pomocou parametrov *comp_addr* a *comp_port*.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.000090	192.168.1.2	172.0.0.4	DNS	71	Standard query 0x23de ANY example.com

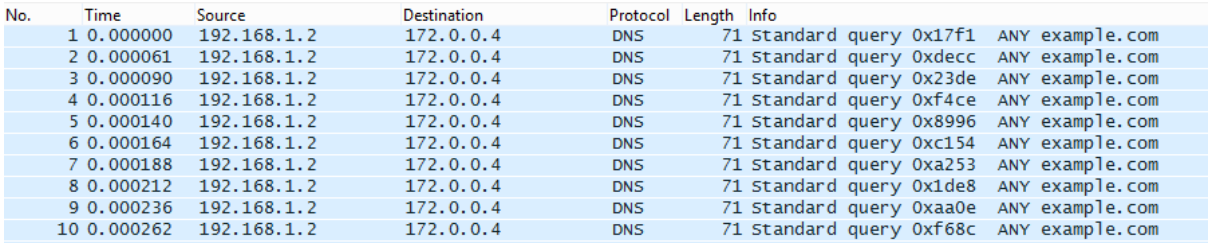
Obr. 4.4.1.1 : Podvrhnutá IP adresa

Ako je vidieť na obrázku [4.4.1.1], paket obsahuje IP adresu *192.168.1.2* hoci bol odoslaný z adresy *192.168.0.2*. Paket vyzerá vierohodne a za vierohodný ho považujú aj všetky sieťové prvky na ceste k DNS serveru, preto, po dorazení k serveru, server posiela odpoveď na túto podvrhnutú adresu.

Ak k útoku využívame záznam väčší ako 512 bytov a DNS server podporuje DNS dáta v UDP pakete väčšie, ako je veľkosť záznamu, k štandardnému dotazu s podvrhnutou IP adresou je potrebné pridať informáciu EDNS(0), že aj cieľová stanica podporuje takéto pakety, inak by do cieľovej stanice odpoveď poslaná nebola. Tohto správania docielime pomocou parametru *udp_size*.

Implementovaný program podporuje len dotazy typu *a*, *rrsig*, *cname*, *txt* a *any*. Tieto dotazy sú v reflexívnom útoku použité najčastejšie, preto nebolo potrebné implementovať ďalšie typy.

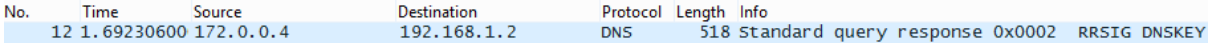
Takto vytvorené pakety sú potom poslané na DNS server. Množstvo odoslaných paketov je definované pomocou parametru *send*. Program sa ukončí až po odoslaní posledného dotazu.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.2	172.0.0.4	DNS	71	Standard query 0x17f1 ANY example.com
2	0.000061	192.168.1.2	172.0.0.4	DNS	71	Standard query 0xdecc ANY example.com
3	0.000090	192.168.1.2	172.0.0.4	DNS	71	Standard query 0x23de ANY example.com
4	0.000116	192.168.1.2	172.0.0.4	DNS	71	Standard query 0xf4ce ANY example.com
5	0.000140	192.168.1.2	172.0.0.4	DNS	71	Standard query 0x8996 ANY example.com
6	0.000164	192.168.1.2	172.0.0.4	DNS	71	Standard query 0xc154 ANY example.com
7	0.000188	192.168.1.2	172.0.0.4	DNS	71	Standard query 0xa253 ANY example.com
8	0.000212	192.168.1.2	172.0.0.4	DNS	71	Standard query 0x1de8 ANY example.com
9	0.000236	192.168.1.2	172.0.0.4	DNS	71	Standard query 0xaa0e ANY example.com
10	0.000262	192.168.1.2	172.0.0.4	DNS	71	Standard query 0xf68c ANY example.com

Obr. 4.4.1.2 : Časť vygenerovaného reflexívneho útoku

Do cieľovej stanice potom dochádza toľko paketov, koľko dotazov bolo na DNS server odoslaných. Odpoveď od DNS servera potom môže vyzerať takto:



No.	Time	Source	Destination	Protocol	Length	Info
12	1.69230600	172.0.0.4	192.168.1.2	DNS	518	Standard query response 0x0002 RRSIG DNSKEY

Obr. 4.4.1.3 : Jedna z odpovedí DNS servera

Obrázok [4.4.1.3] ukazuje prijatú správu v sieti počítača, ktorého IP adresa bola podvrhnutá. Aj pri dotaze na takúto legitímnu doménu je vidieť amplifikačný efekt, ktorý dosiahol takmer osem násobok dotazu. To znamená, že programu stačí vygenerovať, v tomto prípade, osminu dát, aby dosiahol na cieľovej stanici potrebné zahltenie.

4.4.2 Cache poisoning

Útok, ktorý generuje pakety cache poisoningu je implementovaný v programe *cache_poisoning*.

Podstatou tohto útoku je vytvoriť falošnú odpoveď, ktorá sa po zaslaní dotazu pošle na cielený DNS server. Ak táto odpoveď bude totožná s odpoveďou, ktorú server očakáva, server si falošnú odpoveď uloží do svojej dočasnej pamäte.

Program využíva útok hrubou silou, to znamená, že po zaslaní dotazu na server začne okamžite posielať falošné odpovede a snaží sa uhádnuť číslo portu a transakčné ID. Program umožňuje využiť narodeninový paradox, takže miesto jedného dotazu na danú doménu, pošle na server zadané množstvo dotazov. Počet odoslaných dotazov sa dá nastaviť v konfiguračnom súbore pomocou parametru *send_queries*. Keďže program nijako neoveruje úspešnosť útoku, je potrebné pred spustením zadať aj počet poslaných falošných odpovedí. Táto hodnota sa zadáva pomocou parametru *send_responses*, taktiež v konfiguračnom súbore.

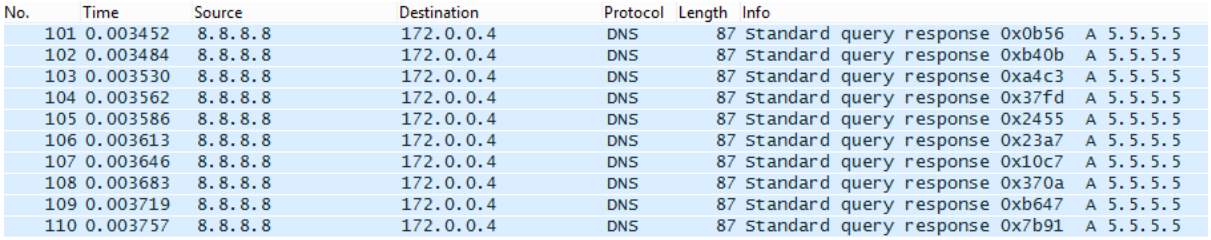
V konfiguračnom súbore je možné nastaviť port a transakčné ID, ktoré je potrebné uhádnuť. Port sa nastavuje pomocou parametru *vict_port_res* a transakčné ID pomocou parametru *dns_res_id*. Ak sa do týchto parametrov zadá hodnota *auto*, hodnoty parametrov sa pri každej odoslanej falošnej odpovedi vygenerujú náhodne. Pre generovanie týchto hodnôt je použitý štandardný c++ pseudo náhodný generátor. Pre generovanie portu sa vynecháva prvých 1023 rezervovaných portov.

Po spustení programu sa vyšle na server zadaný počet dotazov na doménu, ktorej záznam chceme podvrhnúť. Týmto dotazom je navyše možné zadať ľubovoľnú zdrojovú IP adresu, čím sa môže dosiahnuť lepšia anonymita útočníka.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.2	172.0.0.4	DNS	71	Standard query 0xa6e7 A example.com
2	0.000070	192.168.0.2	172.0.0.4	DNS	71	Standard query 0xc96f A example.com
3	0.000118	192.168.0.2	172.0.0.4	DNS	71	Standard query 0xcb2e A example.com
4	0.000162	192.168.0.2	172.0.0.4	DNS	71	Standard query 0x4d8c A example.com
5	0.000200	192.168.0.2	172.0.0.4	DNS	71	Standard query 0x9c72 A example.com
6	0.000225	192.168.0.2	172.0.0.4	DNS	71	Standard query 0xd48a A example.com
7	0.000250	192.168.0.2	172.0.0.4	DNS	71	Standard query 0x9fe9 A example.com
8	0.000273	192.168.0.2	172.0.0.4	DNS	71	Standard query 0xea86 A example.com
9	0.000307	192.168.0.2	172.0.0.4	DNS	71	Standard query 0xa643 A example.com
10	0.000349	192.168.0.2	172.0.0.4	DNS	71	Standard query 0x5cdd A example.com

Obr. 4.4.2.1 : Vygenerované dotazy cache poisoningu

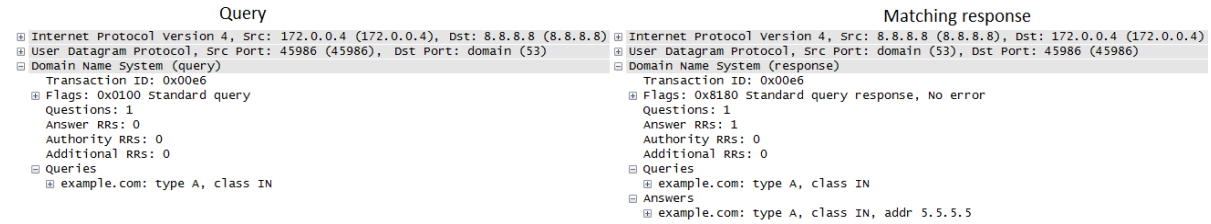
Po odoslaných dotazoch sa na DNS server okamžite začne posielať zadané množstvo falošných odpovedí. Tieto odpovede obsahujú IP adresu servera od ktorého obetný DNS server očakáva odpoveď a náhodne vygenerovaný port a transakčné ID.



No.	Time	Source	Destination	Protocol	Length	Info
101	0.003452	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x0b56 A 5.5.5.5
102	0.003484	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0xb40b A 5.5.5.5
103	0.003530	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0xa4c3 A 5.5.5.5
104	0.003562	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x37fd A 5.5.5.5
105	0.003586	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x2455 A 5.5.5.5
106	0.003613	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x23a7 A 5.5.5.5
107	0.003646	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x10c7 A 5.5.5.5
108	0.003683	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x370a A 5.5.5.5
109	0.003719	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0xb647 A 5.5.5.5
110	0.003757	8.8.8.8	172.0.0.4	DNS	87	Standard query response 0x7b91 A 5.5.5.5

Obr. 4.4.2.2 : Vygenerované odpovede cache poisoningu

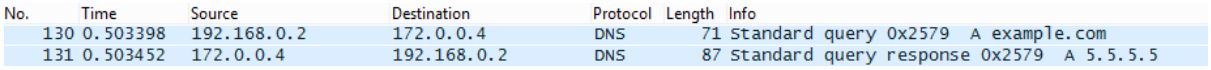
V tomto príklade z obrázkov [4.4.2.1] a [4.4.2.2] sa program snaží serveru podvrhnúť *a* záznam pre doménu *example.com* s odpoveďou *5.5.5.5*.



Obr. 4.4.2.3 : Zhoda dotazu s odpoveďou

Ak sa odpoveď zhodne s odpoveďou, ktorú očakáva DNS server, ako je to na obrázku [4.4.2.3], tento falošný záznam si server uloží do svojej vyrovnávacej pamäte a pravú odpoveď, ktorá príde neskôr bude ignorovať.

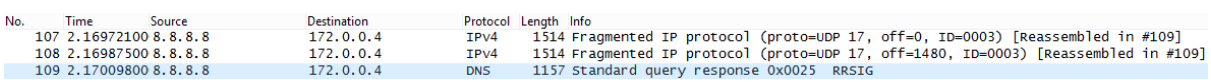
Po skončení programu s úspešne podvrhnutým záznamom server vracia tento falošný záznam.



No.	Time	Source	Destination	Protocol	Length	Info
130	0.503398	192.168.0.2	172.0.0.4	DNS	71	Standard query 0x2579 A example.com
131	0.503452	172.0.0.4	192.168.0.2	DNS	87	Standard query response 0x2579 A 5.5.5.5

Obr. 4.4.2.4 : DNS server vracia falošnú odpoveď

Ďalším implementačným problémom, ktorý bolo potrebné vyriešiť bola fragmentácia paketov. Väčšina sieťových rozhraní totiž neumožňuje prijatie paketu väčšieho ako je 1500 bytov. Ak ale chceme serveru podvrhnúť záznam, ktorý presahuje túto veľkosť, jedinou možnosťou je paket rozdeliť na fragmenty a tieto postupne serveru poslať. Samotnú fragmentáciu knižnica *libnet* nepodporuje, preto bolo potrebné si túto fragmentáciu implementovať vlastným algoritmom. Algoritmus rozdelí dáta aplikačného protokolu na potrebný počet častí a potom každú z týchto častí obalí IP a UDP hlavičkou. V IP hlavičke sa navyše indikuje fragmentácia a posunutie dát pre defragmentáciu. Takto vzniknuté pakety sa pošlú na sieť.



No.	Time	Source	Destination	Protocol	Length	Info
107	2.16972100	8.8.8.8	172.0.0.4	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=0003) [Reassembled in #109]
108	2.16987500	8.8.8.8	172.0.0.4	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=0003) [Reassembled in #109]
109	2.17009800	8.8.8.8	172.0.0.4	DNS	1157	Standard query response 0x0025 RRSIG

Obr. 4.4.2.5 : Paket po fragmentácii

Ako bolo vyššie spomínané, program úspešnosť útoku nijako neoveruje, preto je potrebné zistiť úspešnosť útoku manuálne dotazom na daný server. Úspešnosť útoku je totiž pri dobre zabezpečených serveroch veľmi nízka.

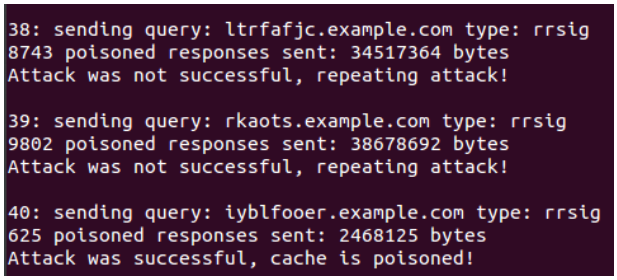
4.4.3 Reflexívny DNS útok s podvrhnutým záznamom

Tento program bol implementovaný ani nie tak pre generovanie dát, ktoré by mohli otestovať detekčné mechanizmy, ako pre ukážkové účely. Program názorne ukazuje ako môže prebiehať reálny útok na internete. Takýto náhľad umožňuje ľahšie pochopenie reflexívneho útoku a cache poisoningu, a spôsobu, ako môžu tieto dva útoky spolupracovať, preto je vhodný napríklad pre výukové účely. Útok je implementovaný v programe *ampli_poison*.

Pre správnu funkčnosť programu je potrebné mať vytvorenú testovaciu sieť a nie je možné ho použiť, ako ostatné programy, bez pripojenia do siete.

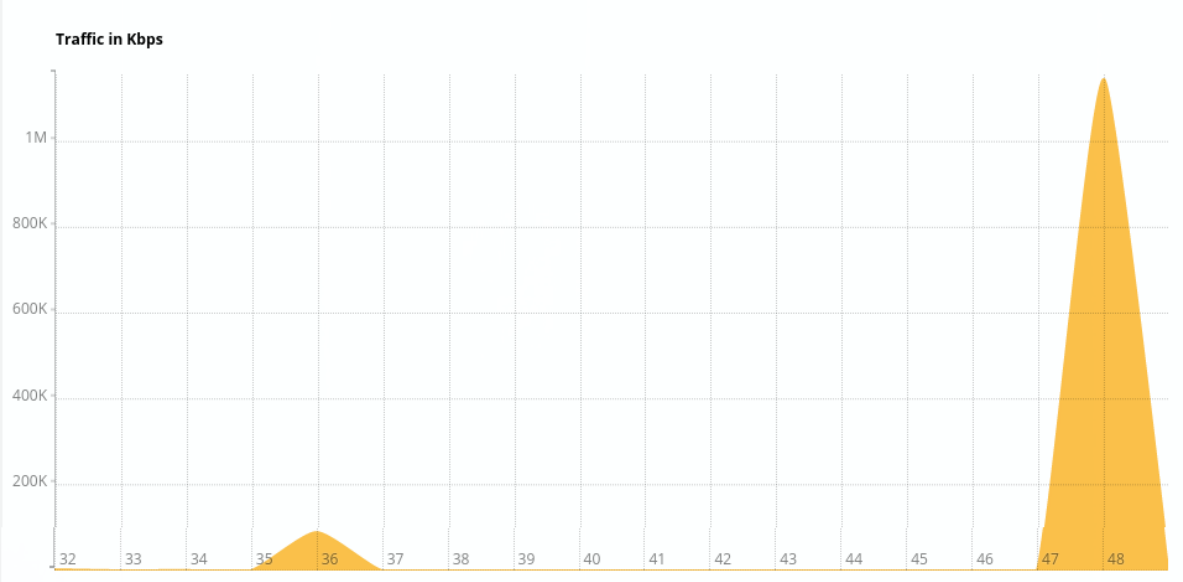
Program sa skladá z dvoch fázy. V prvej fáze sa snaží DNS serveru podvrhnúť nejaký rozmerný záznam, ktorý v druhej fáze využije k čo najväčšiemu amplifikačnému efektu reflexívneho útoku. Veľkosť záznamu, ktorý má byť podvrhnutý sa v konfiguračnom súbore udáva pomocou parametru *response_size*. Táto veľkosť sa dá znásobiť parametrom *multiply*. Keďže dĺžka *txt* záznamu je obmedzená na 255 bytov, pomocou násobiaceho parametru sa odošle niekoľko záznamov so zadanou dĺžkou. Dokopy však paket nesmie presiahnuť veľkosť 4096 bytov.

Cache poisoning v tomto programe funguje automaticky. Program si vytvorí pred poisoningom dve vlákna, kde jedno slúži na odchytávanie paketov a druhé na ich posielanie. Vlákno na posielanie vyšle na DNS server jeden alebo niekoľko dotazov a za nimi, ako pri klasickom cache poisoningu, začne posielať odpovede. Vlákno pre odchytávanie paketov zatiaľ čaká na odpoveď od DNS servera. Po prijatí paketu oznámi vláknu na posielanie, že má prestať posielať pakety a skontroluje prijatú správu. Ak zistí, že prijatá správa obsahuje podvrhnutú odpoveď, cache poisoning sa ukončí, inak sa vygeneruje nová doména, ktorej záznam sa program bude snažiť podvrhnúť a zaháji sa nový cache poisoning.



Obr. 4.4.3.1 : Opakovanie cache poisoningu

Po úspešnom útoku program zaháji amplifikačný útok na podvrhnutú doménu. Týmto spôsobom je možné dosiahnuť maximálneho amplifikačného efektu.



Obr. 4.4.3.2 : Netflow dáta

Na obrázku [4.4.3.2] je vidieť výrazné zahltenie siete pri zahájení útoku. Toto zahltenie by sa dalo zväčšiť distribuovaným útokom z viacerých staníc. Zahltenie bolo navyše limitované výkonom počítača v testovacom prostredí, kde DNS server bežal len vo virtuálnom stroji.

Pri práci s programom je potrebné poznamenať, že chyby vzniknuté za behu cache poisoningu sa v termináli môžu prejaviť aj o 60 sekúnd neskôr. Je to spôsobené časovým limitom prijímacieho vlákna, na ktoré musí program počkať pred svojim ukončením.

4.4.4 DNS tunelovanie

DNS tunelovanie v sebe zahrňuje dva programy. Klientsky program generuje dotazy na serverový program. Klient je implementovaný v programe *dns_tunnel_client* a server v programe *dns_tunnel_server*.

Tieto programy medzi sebou neposielajú reálne dáta, ale snažia sa napodobniť pakety reálnych programov pre tunelovanie. Klientsky program môže pracovať samostatne, aj bez pripojenia do siete, no serverový program bez spustenia programu klienta žiadne odpovede nevygeneruje, takže je potrebné, aby bol spustený vo funkčnej sieti. Navyše musí byť počítač, v ktorom serverový program prebieha, primárnym serverom pre zadanú doménu.

Programy umožňujú využiť pre generovanie dát päť základných kódovaní. Je to klasické kódovanie base32 a base64, a ďalej kódovania zložené z malých písmen, z čísel a s ich kombináciou. Jednotlivé kódovania sa dajú nastaviť pomocou parametru *encoding* v konfiguračnom súbore.

Kódovanie base64 potrebuje, aby DNS server preposielal aj veľké a malé písmená, inak sa kódovaná správa môže pozmeniť.

Klientsky program umožňuje navyše generovať dáta do viacerých subdomén. Maximálny počet subdomén s vygenerovanými dátami je možné zmeniť pomocou parametru *subdomains* v konfiguračnom súbore. Program sa snaží tento maximálny počet dodržať čo najpresnejšie, no tak, aby nepresiahol maximálnu dĺžku domény, ktorá je 253 bytov.

```
base32
 93 Standard query 0xcfe0  TXT 2HL7FZ2KUXO4EZE4D2LX.mydomain.com
104 Standard query 0xfd58  CNAME DT6MPHARCH4BNV3XKSFGX6LTBF7ZCA6.mydomain.com
127 Standard query 0x856f  NULL 7UYRI6LYER5GPN3RDGSEFUKDPTCIBU5AIVSQT5IYOG65TZPWAB2FVE.mydomain.com
base64
120 Standard query 0x5d60  A 1ZHnSHj9W2QChR/D+PQHx5XD1M9tlVPbJkopRymCo24wH4Z.mydomain.com
117 Standard query 0x9781  TXT b4wYvlysEKciLLsZO+lbAA3sWzkJ5Ra6JwT5Hfl/Pnho.mydomain.com
 91 Standard query 0xd1f6  CNAME NynOYNFex4p6X66SEq.mydomain.com
alphabetic
 93 Standard query 0x86bb  CNAME rlhuzlapwqzkommffdew.mydomain.com
104 Standard query 0x03c8  NULL kzgttdrmucvqeirtfhspveddljhitpd.mydomain.com
 89 Standard query 0x4800  A zkpqyjvtazctsjcn.mydomain.com
alphanumeric
 81 Standard query 0x73f5  TXT dq2r5hyp.mydomain.com
 78 Standard query 0x5e3d  A fgkzq.mydomain.com
125 Standard query 0x44c4  CNAME oktbe3y3sn9r76jo1qsg4mrpshe87smli5nnyrwx4bu173vezd09.mydomain.com
numeric
124 Standard query 0x7d7b  TXT 585142878057645109377547051417985619098815779301439.mydomain.com
116 Standard query 0xf983  NULL 030473181742843946416666712842475414042l186.mydomain.com
134 Standard query 0xdb04  TXT 9885241809147873540890233649883791217419534433807808022306586.mydomain.com
```

Obr. 4.4.4.1 : Ukážky jednotlivých kódovaní

Klientska časť odosiela dotazy na zvolený DNS server a serverová časť tieto dotazy prijíma a odpovedá serveru. Serverová časť odpovedá len na dotazy typu *rrsig*, *null*, *cname* a *txt*.

4.4.5 Škodlivé domény

Generovanie škodlivých domén je implementované v programe *domain_generator*.

Pre generovanie domén bol implementovaný vlastný algoritmus, ktorý generuje pseudo náhodné čísla. Tento generátor je potom použitý na generovanie pseudo náhodných textových reťazcov. V konfiguračnom súbore je možné tomuto algoritmu zadať tri konštanty, ktoré budú obmieňať výsledok generovania. S rovnakými konštantami program vygeneruje stále tie isté čísla, z čoho sa vygenerujú tie isté doménové mená. Toto správanie tak umožňuje na dvoch rôznych staniciach vygenerovať rovnaké pseudo náhodné textové reťazce.

```
randSeed = parser->getRandomSeed();
unsigned myRandom(const cConfigParser* parser)
{
   randSeed = randSeed * parser->getRandomConstA() + parser->getRandomConstB();
   return randSeed;
}
```

Kód 4.4.5.1 : Generátor pseudo náhodných čísel

Generátor vracia hodnoty od 0 po maximálnu veľkosť typu *unsigned integer*. Konštanty generátoru pseudo náhodných čísel je možné meniť v konfiguračnom súbore pomocou parametrov *seed*, *const_a* a *const_b*.

Program umožňuje navyše vybrať znakovú sadu pre generovanie domén. Je možné vybrať si buď z malých písmen abecedy, čísel alebo kombináciou čísel a písmen.

Škodlivá doména môže byť vygenerovaná aj s viacerými subdoménami. Maximálny počet subdomén je možné meniť pomocou parametru *subdomains* v konfiguračnom súbore. Program sa snaží tento maximálny počet dodržať čo najpresnejšie, no nemôže presiahnuť maximálnu dĺžku domény, ktorá je 253 bytov.

```
Length  Info
   147 Standard query 0xaafc  CNAME klqv4du3oh274p6jolqjk16.8hmf4pazclmfw1qv812bw56fc9qj85ujodmz852.gx6j0p2b01ub8dezgta.com
    91 Standard query 0xa1b6  A ol63o1ibc1yzk1i3o523.p.j0xq.com
   165 Standard query 0xe222  CNAME 0xaf4h2741qbol6bolyzg1avclaf45m34.230pe7wh6b0t6rgx2fgh23opmf0dmvs1yb8t2b4x.n8l67stifwpu7spy78davglif0.com
   168 Standard query 0xfd87  CNAME ynkhursdivwl6f8dunst6jwt6nc1mnstm7oxif4pe7oxevk9ejwh6j4dmbox6v.t2vc5u7khu3cdij4hiz4xi7w5yb4dejc1qj816.4p.com
   169 Standard query 0xe137  CNAME v8luvchmz4x2vs5m741qnodmr8lyjox2fo9e3c.if0h2rk5ubcxqzglq7cp270lurwty38dizohqfcpy3cxavo.6bkd2jkxibg5u7odif.com
   129 Standard query 0xb576  TXT 5mnk5yn0tef0p2jwhm7spuz0pajg9mbk1m34h238.678x6vs9irspq74d.rspiz8l.com
```

Obr. 4.4.5.1 : Vygenerované doménové mená

5 Záver

Cieľom tejto práce bolo generovanie paketov jednotlivých DNS útokov. Implementovali sa generátory pre reflexívny DNS útok, cache poisoning, DNS tunelovanie a generátor škodlivých domén. Tieto programy môžu byť použité pre testovanie detekčných mechanizmov. Na testovanie sa môže využiť buď vygenerovaný libpcap súbor, vďaka ktorému nemusia detekčný program a generátor bežať súčasne, alebo sa tieto programy a detekčný mechanizmus spustia v rovnakom čase.

Tieto programy by nemohli byť implementované bez analýzy jednotlivých útokov. K analýze sa využili už existujúce programy, ktoré zraniteľnosť DNS služby využívajú, alebo analýza prebiehala štúdiom vedeckých článkov, ktoré popisujú chovanie jednotlivých útokov.

V rámci tejto práce som si mohol vyskúšať konfiguráciu a prácu s vlastným DNS serverom a konfiguráciu vlastnej siete v sieťovom laboratóriu, kde som overoval jednotlivé programy. V laboratóriu som si mohol navyše vyskúšať prácu s programom na zbieranie štatistík o sieti.

Rozšírením tejto práce by mohlo byť generovanie už implementovaných útokov iným, taktiež používaným spôsobom. Do budúcnosti by sa navyše k už implementovaným anomáliám mohli dorobiť útoky spojené s inou službou ako je služba DNS. Takto by mohol zvyknúť kompletný generátor, ktorý by testoval detekčné mechanizmy na všetky sieťové anomálie.

Literatúra

[1] ROZEKRANS a DE KONING. *Defending against DNS reflection amplification attacks* [online]. 2013[cit. 2014-12-16]. Dostupné z: http://www.nlnetlabs.nl/downloads/publications/report-rrl-dekoning-rozekrans.pdf

[2] KAMBOURAKIS, Georgios, Tassos MOSCHOS, Dimitris GENEIATAKIS a Stefanos GRITZALIS. *A Fair Solution to DNS Amplification Attacks* [online]. 2010[cit. 2014-12-16]. Dostupné z: http://www.cs.columbia.edu/~dgen/papers/conferences/conference-07.pdf

[3] FERGUSON a SENIE. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. [online]. 2000 [cit. 2014-12-16]. Dostupné z:http://tools.ietf.org/html/bcp38

[4] DONNERHACKE, Lutz. DNS Dampening. [online]. 2012 [cit. 2014-12-16]. Dostupné z:http://lutz.donnerhacke.de/eng/Blog/DNS-Dampening

[5] MATOUŠEK. *Síťové aplikace a jejich architektura*. 1. vyd. Brno: VUTIUM, 2014, 396 s. ISBN 978-80-214-3766-1

[6] P.Mockapetris. *Domain Names — Implementation and Specification*. RFC 1035, November 1987

[7] Di Pietro, R., Mancini, L.V.: *Intrusion Detection Systems*, 1st edn. Springer Publishing Company, Incorporated (2008)

[8] Amon, C., Shinder, T.W., Carasik-Henmi, A.: *The Best Damn Firewall Book Period*, 2nd edn. Syngress Publishing (2007)

[9] Butler, P.; Xu, K.; Yao, D. D.: *Quantitatively Analyzing Stealthy Communication Channels. In Proceedings of the 9th International Conference on Applied Cryptography and Network Security*, ACNS'11, Springer-Verlag, 2011, ISBN 978-3-642-21553-7, s. 238-254.

[10] 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, CSIIRW '10, 2010, ISBN 978-1-4503-0017-9.

[11] Musashi, Y.; Kumagai, M.; Kubota, S.; aj.: *Detection of Kaminsky DNS Cache Poisoning Attack*. In Proceedings of the 2011 4th International Conference on Intelligent Networks and Intelligent Systems, ICINIS '11, 2011, ISBN 978-0-7695-4543-1, s. 121-124.

[12] Alexiou, N.; Basagiannis, S.; Katsaros, P.; aj.: *Formal Analysis of the Kaminsky DNS Cache-Poisoning Attack Using Probabilistic Model Checking*. In High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on, 2010, ISSN 1530-2059, s. 94-103.

[13] SON, Sooel a Vitaly SHMATIKOV. The Hitchhiker's Guide to DNS Cache Poisoning. *Security and privacy in communication networks: 6th Iternational ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings*. Berlin: Springer, 2010, č. 50, s. 466-483. DOI: 10.1007/978-3-642-16161-2_27. Dostupné z: http://link.springer.com/10.1007/978-3-642-16161-

[14] Libpcap File Format. *Wireshark Wiki* [online]. 2013 [cit. 2015-05-02]. Dostupné z: https://wiki.wireshark.org/Development/LibpcapFileFormat

[15] STEWART, Joe. 2007. *DNS Cache Poisoning – The Next Generation* [online]. [cit. 2015-05-05]. Dostupné z: https://www.ida.liu.se/~TDDC03/literature/dnscache.pdf

[16] YU, Huiming, Xiangfeng DAI, Tomas BAXLEY a Jinsheng XU. 2008. A Real -Time Interactive Visualization System for DNS Amplification Attack Challenges. *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. IEEE, : 55-60. DOI: 10.1109/ICIS.2008.42. ISBN 978-0-7695-3131-1. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4529798

[17] VILLAMARIN-SALOMON, Ricardo a Jose Carlos BRUSTOLONI. 2008. Identifying Botnets Using Anomaly Detection Techniques Applied to DNS Traffic. *2008 5th IEEE Consumer Communications and Networking Conference*. IEEE, : 476-481. DOI: 10.1109/ccnc08.2007.112. ISBN 1-4244-1457-1. Dostupné také z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4446410

[18] HÄMMERLI, Bernhard M a Robin SOMMER. *Detection of intrusions and malware, and vulnerability assessment: 4th international conference*, DIMVA 2007, Lucerne, Switzerland, July 12-13, 2007 : proceedings. New York: Springer, 2006, s. pp 129-139. ISBN 3540736131.

[19] Computational intelligence in security for information systems: 4th international conference, CISIS 2011, held at Iwann 2011, Torremolinos-Malaga, Spain, June 8-10, 2011. proceedings. New York: Springer, 2011, s. pp 84-91. ISBN 3642213227.

Príloha A

Obsah CD

/src

Priečinok obsahuje skript pre obsluhu jednotlivých programov s popisom jeho obsluhy a návodom na inštaláciu.

Priečinok ďalej obsahuje priečinky jednotlivých programov pre generovanie útokov, ktoré obsahujú zdrojové kódy programu, Makefile ,vytvorený konfiguračný súbor a popis ich obsluhy a inštalácie.

/bin

Priečinok obsahuje priečinky so spustiteľnými programami pre generovanie jednotlivých útokov a vytvorené konfiguračné súbory.

/data

Priečinok obsahuje vygenerované libpcap súbory jednotlivých programov.

/text

Priečinok obsahuje tento dokument v upraviteľnej forme a vo formáte pdf.

/env

Priečinok obsahuje popis a konfiguráciu testovacieho prostredia.

/libs

Priečinok obsahuje archívy s knižnicami libnet a libpcap.

Príloha B

Práca s obslužným skriptom

Použitie skriptu:

./manager.sh help

- vytlačenie nápovedy skriptu

./manager.sh build all

./manager.sh build "názov programu"

- preloženie všetkých alebo konkrétneho programu

./manager.sh clean all

./manager.sh clean "názov programu"

- vymazanie súborov prekladu všetkých alebo konkrétneho programu

./manager.sh clean_pcap all

./manager.sh clean_pcap " názov programu"

- vymazanie všetkých vygenerovaných libpcap súborov všetkých alebo konkrétneho programu

./manager run " názov programu"

./manager run " názov programu" "názov konfiguračného súboru"

./manager run " názov programu" --help

- spustenie konkrétneho programu s predvoleným alebo špecifickým konfiguračným súborom, prípadne vytlačenie nápovedy programu

Názvy programov:

- *ampli_attack* – reflexívny DNS útok
- *ampli_poison* – reflexívny DNS útok s podvrhnutým záznamom
- *cache_poisoning* – cache poisoning
- *dns_tunnel_client* – klientska časť DNS tunelovania
- *dns_tunnel_server* – serverová časť DNS tunelovania
- *domain_generator* – generátor škodlivých domén

Poznámky:

- každý program musí byť v samostatnom priečinku, ktorý je pomenovaný podľa programu, so všetkými zdrojovými súbormi, funkčným *Makefilom* a konfiguračným súborom
- pre úspešný preklad a beh programov musí mať počítač nainštalovanú knižnicu libnet a libpcap
- podrobnejšie informácie o jednotlivých programoch je možné nájsť v nápovede programu alebo v súbore *README.txt* pri každom programe